



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

**Interwoven TeamXpress™  
for Multiplatforms V1.1,  
WebSphere™ Edition**

**Administration Guide for Solaris™**

© 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, Inc., which may be registered in certain jurisdictions. TeamXpress, SmartContext, DataDeploy, Content Express, 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

Release 1.1

Part # 20-00-40-45-04-110-300

# Table of Contents

## **About This Book 5**

Notation Conventions 5

Support Information 6

## **Chapter 1: Overview 7**

TeamXpress Elements 7

TeamXpress Users 10

TeamXpress Architecture 11

## **Chapter 2: Installing TeamXpress 13**

Before You Begin 13

Installing TeamXpress 16

Configuring Webservers 23

Configuring Samba 31

Setting Up TeamXpress Clients 32

Loading Content 39

Upgrading TeamXpress 47

If you are upgrading TeamXpress, you do not need to reboot the server. 47

Uninstalling TeamXpress 47

## **Chapter 3: Setting Up Users 49**

Overview 49

Determining User Roles 50

Adding and Removing Users 51

User Role Privileges and Permissions 56

## **Chapter 4: Configuring the TeamXpress Server 63**

Configuring GUI Appearance 65

Configuring GUI Functionality 69

Configuring Server Functionality 85

Configuring Server Performance 99

Changing File Attributes During Submit 102

Configuring the TeamXpress Proxy Server 111

Configuring TeamXpress Failsafe 130



<b>Chapter 5: Configuring TeamXpress Workflow</b>	<b>137</b>
Definitions of Workflow Terms	137
Creating a Job	139
Understanding Workflow Template File Structure	144
Debugging Workflow Templates and Job Specification Files	165
Adding Template Selection Choices to TeamXpress GUI Menus	166
Using Pre-Packaged TeamXpress Workflow Templates	171
Understanding Job Specification File Structure	180
Using Perl Modules	196
<b>Chapter 6: Configuring Metadata Capture and Search</b>	<b>203</b>
Configuring Metadata Capture	203
Configuring Metadata Search	228
<b>Chapter 7: Managing the TeamXpress Server</b>	<b>235</b>
Checking Server Status	235
Reviewing TeamXpress Logs	238
Monitoring the Server Load	239
Starting and Stopping the Server	239
Troubleshooting	239
Managing Server Resources	244
<b>Chapter 8: Backing Up TeamXpress</b>	<b>251</b>
Integrating with Third-Party Backup Solutions	251
Suggested Strategies for Incremental Backups	253
Performing Full Backups with iwbackup	254
Performing Incremental Backups with iwbackup	255
<b>Appendix A: TeamXpress Configuration Files</b>	<b>257</b>
<b>Appendix B: Sample Job Specification File</b>	<b>259</b>
<b>Appendix C: Backing Store Conversion and Distribution</b>	<b>265</b>
About Backing Store Performance Enhancements	265
Consolidating Metadata	265
Distributing the Backing Store Across File Systems	272
<b>Appendix D: Internationalization</b>	<b>281</b>
<b>Index</b>	<b>285</b>

# About This Book

---

The *TeamXpress Administration Guide* is a guide to installing, configuring, and maintaining TeamXpress. It is primarily intended for TeamXpress Administrators and Master users, and for web server administrators and system administrators. Users of this manual should be familiar with basic UNIX commands and be able to use an editor such as emacs or vi.

Many of the operations described in this manual require root access to the TeamXpress server. If you do not have root access to the TeamXpress server, consult your UNIX system administrator.

It is also very helpful to be familiar with regular expression syntax. If you are not familiar with regular expressions, it is recommended that you consult a reference manual such as *Mastering Regular Expressions*, by Jeffrey Friedl.

## Notation Conventions

This manual uses the following notation conventions:

Convention	Definition and Usage
<b>Bold</b>	Text that appears in a GUI element (e.g., a menu item, button, or element of a dialog box) and command names are shown in bold. For example:  Click <b>Edit File</b> in the Button Bar.
<i>Italic</i>	Book titles appear in italics. Terms are italicized the first time they are introduced. Important information may be italicized for emphasis.

Convention	Definition and Usage
Monospaced	<p>Commands, command-line output, and file names are in monospaced type. For example:</p> <p style="padding-left: 40px;">The <code>iwextattr</code> command-line tool allows you to set and look up extended attributes on a file.</p>
<i>Monospaced italic</i>	<p>Monospaced italics are used for command-line variables. For example:</p> <p style="padding-left: 40px;"><code>iwckrole role user</code></p> <p>means that you must insert the values of <i>role</i> and <i>user</i> yourself.</p>
<b>Monospaced bold</b>	<p>Monospaced bold represents user input. The <code>%</code> character that appears before a line of user input represents the command prompt and should not be typed. Your system may not use this command prompt. For example:</p> <p style="padding-left: 40px;"><code>% iwextattr -s project=proj1 //IWSERVER/default/main/dev/WORKAREA/andre/products/index.html</code></p>
<b><i>Monospaced bold italic</i></b>	<p>Monospaced bold italic text is used to indicate a variable in user input. For example:</p> <p style="padding-left: 40px;"><code>% iwextattr -s project=<i>projectname</i> <i>workareavpath</i></code></p> <p>means that you must insert the values of <i>projectname</i> and <i>workareavpath</i> when you enter this command.</p>
[ ]	<p>Square brackets surrounding a command-line argument mean that the argument is optional.</p>
	<p>Vertical bars separating command-line arguments mean that only one of the arguments can be used.</p>

## Support Information

For support information concerning IBM TeamXpress, refer to the following URL:  
<http://www-4.ibm.com/software/webervers/teamxpress/support.html>

# Overview

---

## TeamXpress Elements

### Branches

TeamXpress provides *branches* for different paths of development for a Web site. Branches can be related to each other (e.g. alternate language versions of the same Web site) or they may be completely independent. Each branch contains all the content for a Web site.

A single branch contains archived copies of the Web site as *editions*, a *staging area* for content integration, and individual *workareas* where users may develop content without disturbing one another. Branches can also contain *sub-branches*, so that teams may keep alternate paths of development separate from each other. Content can be easily shared and synchronized across branches and sub-branches. Users may work on one branch or on several, and the number of branches on a system is not limited. In TeamXpress, you cannot create workareas off of the main branch. Additionally, depending on the licensing agreement, you may not be able to add branches or sub-branches.

Branches facilitate distributed workflow because they allow separate teams to work independently on different projects. Because all branches are located on the same TeamXpress server, it is easy for one team to incorporate the work of another into their project.

### Workareas

Each *workarea* contains a virtual copy of the entire Web site, which may be modified in any way without affecting the work of other contributors. Users who have access to a workarea may modify files within that workarea and view their changes within the context of the entire Web site before integrating their work with that of other contributors. Users can lock files in each workarea, eliminating the possibility of conflicting edits. In TeamXpress, you cannot create workareas off of the main branch.



All changes that are made to files in a workarea are kept completely separate from other workareas and the staging area until the user chooses to promote his changes to the staging area. Within a workarea, users may add, edit, or delete files, or revert to older versions of files without affecting other users.

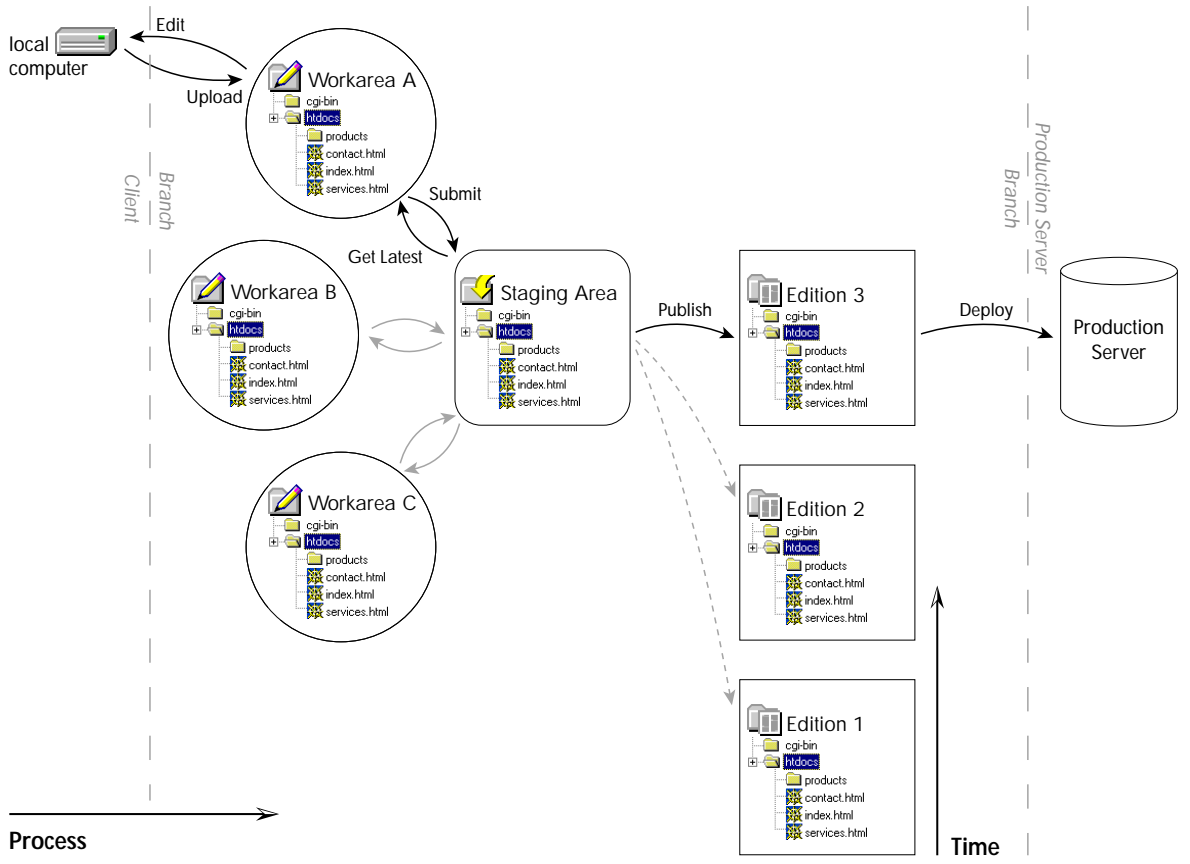
## Staging Areas

Each branch contains one *staging area* where contributors incorporate their changes with the work of others. Users submit files from their workareas to the staging area to integrate their work with other contributions, and test the integrity of the resulting Web site. Because the staging area is an integrated component of the system, conflicts are easily identified and different versions of the same file can be merged, rather than overwritten.

## Editions

*Editions* are read-only snapshots of the entire Web site, taken at sequential points in its development. Contributors can create new editions any time they feel their work is well integrated, or any time they want to create an update to the Web site for reference or deployment. Each edition is a fully functional version of the Web site, so that users may see the development of the Web site over time and compare it with current work.





Process

Time

*TeamXpress elements*

TeamXpress branches contain private workareas, which contain complete virtual copies of the Web site; staging areas, where contributors integrate their work; and editions, which are read-only snapshots of the Web site at various points in its development. Each area contains a virtual copy of the entire Web site. Content is submitted from workareas to the staging area, and the staging area is then published as an edition. Older editions are available for reference.

## TeamXpress Users

### Authors

*Authors* are primary content creators. All work done by Authors goes through an explicit approval step. They can receive assignments from Editors, which are displayed in To-Do lists when Authors log in to TeamXpress. Authors can access TeamXpress from a simple browser-based interface, and do not need to be sophisticated computer users.

In order to test and QA their work, Authors have full access to the content in their Editors' workareas, but do not need to concern themselves with the larger structure and functionality of TeamXpress. The Author role is appropriate for non-technical users, or for more technical contributors who do not need access to TeamXpress's extended functionality, such as TeamXpress's advanced version management features.

### Editors

*Editors* own workareas. They create and edit content, just as Authors do, but they are primarily responsible for managing the development taking place within their workareas. This includes assigning files to Authors and submitting completed content to the staging area, and it may include publishing editions.

Editors have access to specialized TeamXpress content and workflow management functions. Editors are generally "managerial" users, who primarily supervise the work of Authors, or self-managing "power" users, who need TeamXpress's extended functionality to manage their own content.

### Administrators

*Administrators* own branches. They have all the abilities of Editors, but they are primarily responsible for the content and functioning of their branch. Administrators can manage project workflow by creating new workareas for Editors and groups, and by creating sub-branches of their own branch to explore separate paths of development.

An Administrator is the supervisor of the project being developed on his branch. He may be the webmaster for a particular version of the Web site, or a project manager.

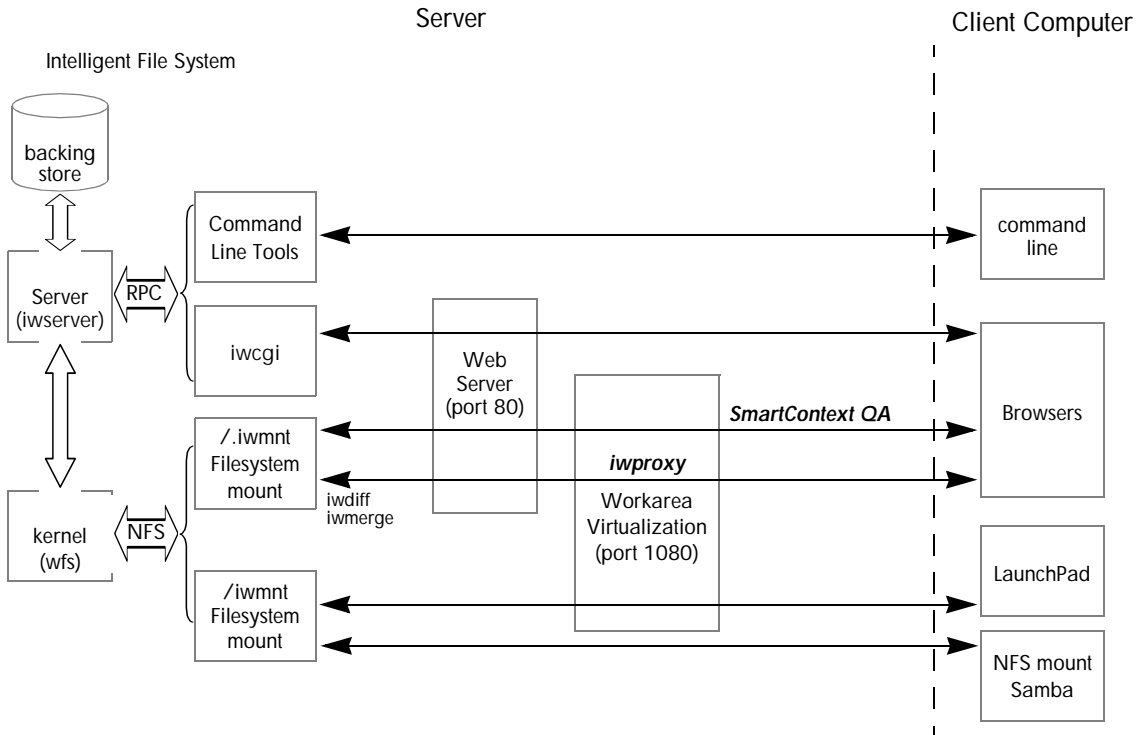
## Masters

*Master users* own the Web site. They can perform all the functions of Editors and Administrators on any branch. The Master user owns the main branch, from which all sub-branches are created. The Master user is generally involved in the installation of TeamXpress, and can reconfigure TeamXpress on a system-wide basis.

## TeamXpress Architecture

TeamXpress's Intelligent File System is composed of the TeamXpress server and kernel, the TeamXpress backing store of files and metadata, a suite of command-line tools, TeamXpress CGI, proxy servers for access through the TeamXpress browser-based GUI, and file system mounts for access through the file system interface.

The Intelligent File System is the core of the TeamXpress system, where detailed information about the Web site, the Web assets, Web asset metadata, the production process and the users is stored. The Intelligent File System collects and maintains metadata on TeamXpress files, directories, and areas, and allows TeamXpress to process and present information according to who is asking for the information, and under what conditions. By using an object oriented design within a file system architecture, TeamXpress combines extensive meta-data tagging with open access and file system performance for Web content.



### TeamXpress architecture

The client computer connects to the TeamXpress server in several ways. Requests from the browsers or LaunchPad are routed through the standard TeamXpress proxy server, which allows consistent views of TeamXpress areas. The double proxy server redirects hard-coded links within the Web site. Requests through the file system interface (NFS mount/Samba) and command-line tools, which do not go through the Web server, are not routed through a proxy server.

# Installing TeamXpress

---

## Before You Begin

### TeamXpress Server Requirements

#### Hardware Requirements

The following recommended hardware configurations are based on the total (not concurrent) number of users and the number of files on the system's largest branch. All CPUs should be at least 300MHz.

		Files per Branch		
		0 to 25,000	25,000 to 75,000	75,000 +
Total users	0 to 25	<ul style="list-style-type: none"><li>• 1 CPU</li><li>• 256MB memory</li></ul>	<ul style="list-style-type: none"><li>• 1 CPU</li><li>• 512MB memory</li></ul>	<ul style="list-style-type: none"><li>• 2 CPUs</li><li>• 1GB memory</li></ul>
	25 to 50	<ul style="list-style-type: none"><li>• 2 CPUs</li><li>• 256MB memory</li></ul>	<ul style="list-style-type: none"><li>• 2 CPUs</li><li>• 512MB memory</li></ul>	<ul style="list-style-type: none"><li>• 2 CPUs</li><li>• 1GB memory</li></ul>
	50 to 100	<ul style="list-style-type: none"><li>• 2 CPUs</li><li>• 256MB memory</li></ul>	<ul style="list-style-type: none"><li>• 2 CPUs</li><li>• 512MB memory</li></ul>	<ul style="list-style-type: none"><li>• 4 CPUs</li><li>• 1GB memory</li></ul>
	100 to 250	<ul style="list-style-type: none"><li>• 4 CPUs</li><li>• 512MB memory</li></ul>	<ul style="list-style-type: none"><li>• 4 CPUs</li><li>• 1GB memory</li></ul>	<ul style="list-style-type: none"><li>• 6 CPUs</li><li>• 2GB memory</li></ul>
	250 +	<ul style="list-style-type: none"><li>• 8 CPUs</li><li>• 1GB memory</li></ul>	<ul style="list-style-type: none"><li>• 8 CPUs</li><li>• 1GB memory</li></ul>	<ul style="list-style-type: none"><li>• 8 CPUs</li><li>• 2GB memory</li></ul>

#### *Recommended hardware configurations*

It is important to have sufficient physical memory for the cache size that you specify in `iw.cfg` as described in “Cache Size” on page 99. Recommended physical memory for the cache is the cache size setting times 1KB plus an additional 25% as a safety margin. For example, if the cache size setting is 45000, physical memory needed just for the cache would be  $(45,000 * 1KB) + 11MB = 56MB$ . If

you encounter a great deal of memory swapping, you should either reduce the cache size setting in `iw.cfg` or install more memory.

### Global Report Center Requirements

The TeamXpress Global Report Center (installation is optional) requires approximately 5 MB of physical memory. The OpenDeploy Global Report Center has the same requirement. Therefore, you should plan on approximately 10 MB of physical memory for the Global Report Center if you install both TeamXpress and OpenDeploy.

### Disk Space Requirements

#### Basic TeamXpress Requirements

All servers must have enough disk capacity for 370MB of TeamXpress program files and five to ten times the total amount you expect the Web site content files to consume. This amount of disk space is required in order to store TeamXpress metadata and multiple versions of your Web site files.

#### Inode requirements

TeamXpress requires a large number of inodes. To estimate how many inodes your server will require, use the following formula:

$$\# \text{ inodes} = (\# \text{ branches})(\# \text{ average files in staging area per branch})(\# \text{ average historical versions/file}) \\ (1 + (\% \text{ of files having extended attributes})/100)(\text{safety-factor})/100$$

For example, if your TeamXpress server has three branches, with 20,000 files in the staging area of each branch, (on average), ten versions of each file in its history list (on average), seven percent of files have extended attributes, and you want to use a 1.5x safety factor:

$$\begin{aligned} \# \text{ inodes} &= 3 * 20,000 * 10 * (1 + .07) * 1.5/100 \\ &= 9630 \text{ inodes} \end{aligned}$$

### Global Report Center Requirements

The TeamXpress Global Report Center (installation is optional) requires an additional 25 MB of disk space, plus 10-50 MB for data storage.

### Software Requirements

The TeamXpress server application runs on the same system as your Web site development server. We recommend that the Web site development server be configured as a dedicated server, running no applications other than the Web server software and TeamXpress. The following software is required or recommended:

- Sun Solaris 2.5.1 (minimum required), 2.6 (recommended), or 32- or 64-bit System 7 (recommended).
- Web server software (i.e., Netscape, Apache, NCSA, etc.).

### TeamXpress Client Requirements

End users access TeamXpress through browser-based thin-client technology. The only hardware requirements for client systems are the RAM, CPU, local storage, and networking capability needed to operate a suitable Web browser and the editing applications of the user’s choice. TeamXpress’s thin-client interface does not require you to install any other client software unless you will be editing files through the TeamXpress GUI.

Not all TeamXpress features are compatible with all browsers on all client platforms. The following table shows compatibility for most popular browsers:

	Windows 95, 98, NT and 2000						Solaris						Macintosh					
	Netscape			Internet Explorer			Netscape			Internet Explorer			Netscape			Internet Explorer		
	4.0 6 - 4.0 8	4.5 x	4.6	4.7	4.x	5.0	4.0 6 - 4.0 8	4.5 x	4.6	4.7	4.x	5.0	4.0 6 - 4.0 8	4.5 x	4.6	4.7	4.0 1	4.5
TeamXpress GUI	Yes	Yes	Yes	Yes*	Yes	Yes*	Yes	Yes	Yes	Yes*	Yes	Yes*	Yes	Yes	Yes	Yes*	Yes	Yes*
SmartContext Editing	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	No	No	No	No	No	No
Merge	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No	Yes	Yes

\*Recommended

### Connecting Through the File System Interface

To connect via the file system interface, users must have a network connection and the ability to interact with a networked file system (i.e., FTP, NFS, PCNFS, SAMBA server, FTP client, Windows networking, etc.). Before installing and configuring any of these protocols, you should be familiar enough with them to perform basic configuration and startup procedures. For more information, see “The File System Interface” on page 34.

## Installing TeamXpress

### TeamXpress File Locations

By default, TeamXpress installs in the following locations (you may select alternate locations for some of these files during installation):

Default Directory	Contents
<code>/usr/iw-home</code>	Default location of TeamXpress program files. The location of this directory may be changed during installation or when the server is stopped.
<code>/local/iw-store</code>	Default location of the TeamXpress backing store (TeamXpress storage of files and metadata for workareas and editions). This directory can consume large amounts of disk space. The location of this directory may be changed during installation or when the server is stopped. To find where this directory is located, use the command-line tool <code>iwgetstore</code> .  <b>Note:</b> The contents of this directory should never be edited by hand in any way. Tampering with this directory can irreparably corrupt the data stored in TeamXpress.
<code>/iwserver</code>	Local file system mount projection directory. Clients mount to this directory to access Web site data. No actual data is stored in this directory, and its location cannot be changed.
<code>/iwmnt</code>	NFS server mount point. TeamXpress This directory is used to access Web site data when working directly from the server. The location of this directory can be changed; however, Web server aliases must be updated to reflect this.



Default Directory	Contents
<code>/.iwmnt</code>	NFS server mount point. This is a noncaching alias used by the Web server. The location of this directory can be changed; however, Web server aliases must updated to reflect this.

## Installing TeamXpress

During installation, any files in the `iw-home/iw-perl` directory are archived into `iw_perl.tar.z` to avoid their being overwritten. If you need to reinstall any of those files, uncompress the file and copy them back into the `iw-perl` directory.

TeamXpress only runs on Sparc or UltraSparc platforms; however, a LaunchPad client is available for the Solaris x86 platform.

To install TeamXpress on your Web site development server:

1. Log in as **root** to the computer where you want to install the TeamXpress server.
2. Uncompress and expand the TeamXpress installation file into the directory under which the TeamXpress application will reside. Decompress the file:

```
% gunzip -c iw-home.tar.gz | (cd /parent_directory; tar xvpf -)
```

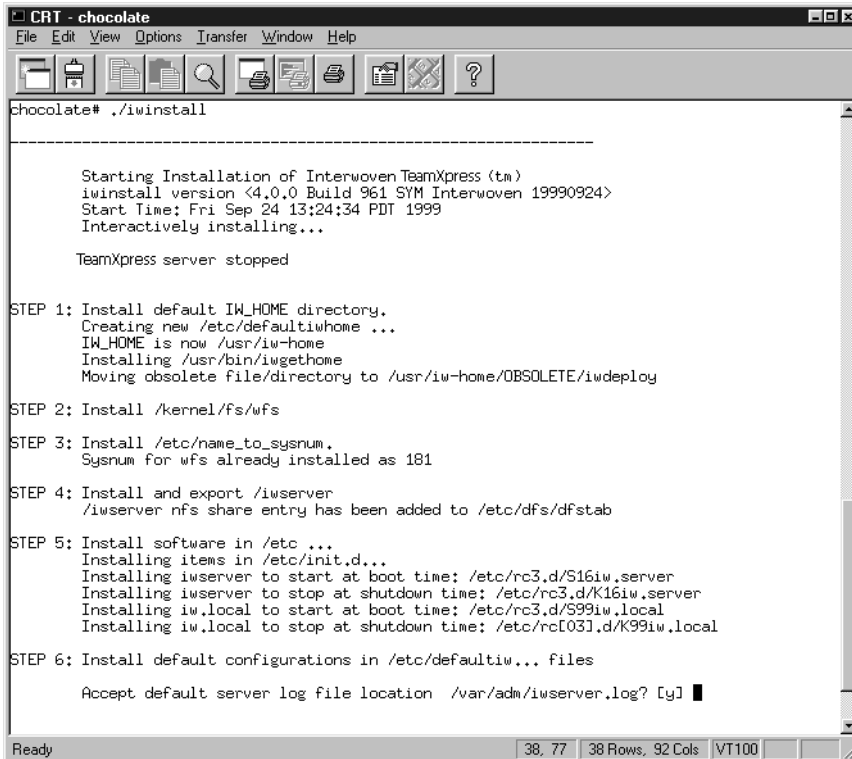
`parent_directory` is the name of the directory under which TeamXpress will be installed. The default parent directory for TeamXpress is `/usr`.

A directory called `iw-home` will be created under the parent directory. This directory contains the TeamXpress program files.

3. The TeamXpress installation file `iwinstall` is located in the sub-directory `iw-home/install`. From this directory, run the command:

```
% ./iwinstall
```

This will install and configure TeamXpress on the server. You will need to provide input during several steps of the process.



```

CRT - chocolate
File Edit View Options Transfer Window Help
-----
chocolate# ./iwinstall

Starting Installation of Interwoven TeamXpress (tm)
iwinstall version <4,0,0 Build 961 SYM Interwoven 19990924>
Start Time: Fri Sep 24 13:24:34 PDT 1999
Interactively installing...

TeamXpress server stopped

STEP 1: Install default IW_HOME directory.
Creating new /etc/defaultiwhome ...
IW_HOME is now /usr/iw-home
Installing /usr/bin/iwgethome
Moving obsolete file/directory to /usr/iw-home/OBSOLETE/iwdeploy

STEP 2: Install /kernel/fs/wfs

STEP 3: Install /etc/name_to_sysnum.
Sysnum for wfs already installed as 181

STEP 4: Install and export /iwserv
/iwserv nfs share entry has been added to /etc/dfs/dfstab

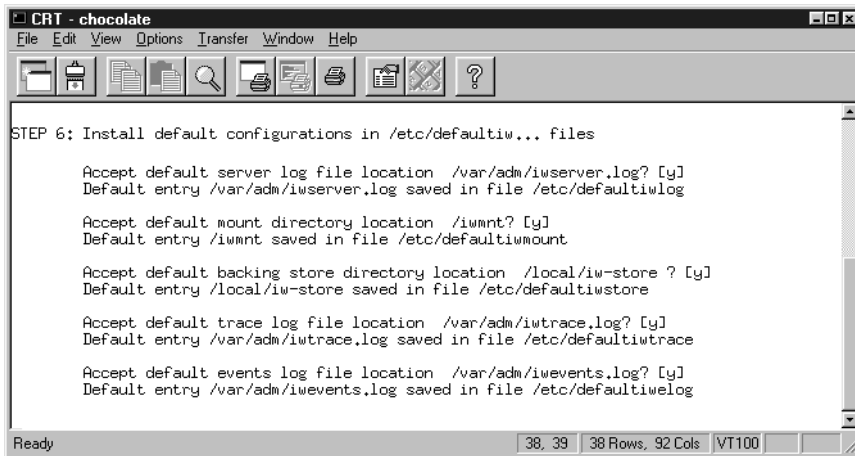
STEP 5: Install software in /etc ...
Installing items in /etc/init.d...
Installing iwserv to start at boot time: /etc/rc3.d/S16iw.serv
Installing iwserv to stop at shutdown time: /etc/rc3.d/K16iw.serv
Installing iw.local to start at boot time: /etc/rc3.d/S99iw.local
Installing iw.local to stop at shutdown time: /etc/rc031.d/K99iw.local

STEP 6: Install default configurations in /etc/defaultiw... files

Accept default server log file location /var/adm/iwserv.log? [y] █
  
```

*Starting the TeamXpress installation*

4. During installation, you must specify locations for the TeamXpress log files, NFS mount point, and backing store. The backing store is a large directory that contains TeamXpress files and metadata. Make sure that the location of the backing store has room for at least three times the size of the entire Web site (e.g., if the Web site will be 200MB in size, the backing store should have at least 600MB of disk space available). For ease of maintenance, it is recommended that the backing store be installed to its own partition.



```
CRT - chocolate
File Edit View Options Transfer Window Help

STEP 6: Install default configurations in /etc/defaultiw... files

Accept default server log file location /var/adm/iwserver.log? [y]
Default entry /var/adm/iwserver.log saved in file /etc/defaultiwlog

Accept default mount directory location /iwmnt? [y]
Default entry /iwmnt saved in file /etc/defaultiwmnt

Accept default backing store directory location /local/iw-store ? [y]
Default entry /local/iw-store saved in file /etc/defaultiwstore

Accept default trace log file location /var/adm/iwtrace.log? [y]
Default entry /var/adm/iwtrace.log saved in file /etc/defaultiwtrace

Accept default events log file location /var/adm/iwevents.log? [y]
Default entry /var/adm/iwevents.log saved in file /etc/defaultiwevents

Ready 38, 39 38 Rows, 92 Cols VT100
```

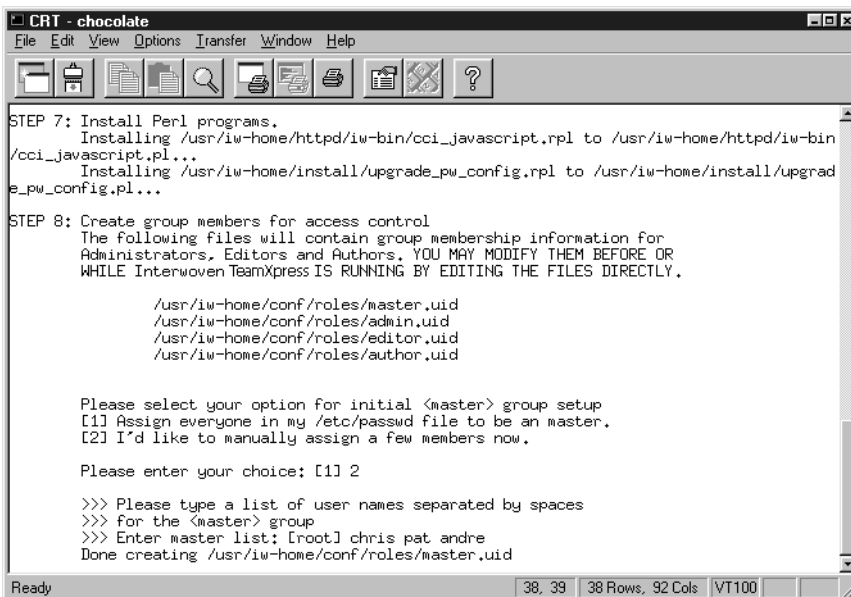
*Specifying log file locations*

5. During installation, you must assign roles to TeamXpress users. The installation program lets you automatically assign all members listed in the password file to a TeamXpress role, or you can manually specify individual users for each role. When manually assigning user names, type each name separated by a space, then **Return** when you are finished.

TeamXpress user roles are stored in the following files:

```
iw-home/conf/roles/master.uid
iw-home/conf/roles/admin.uid
iw-home/conf/roles/editor.uid
iw-home/conf/roles/author.uid
```

You can modify these files at any time (see page 52). Deleting a user name from a file removes TeamXpress access privileges for the role. Adding a user name to a file grants access privileges for the role.



```
CRT - chocolate
File Edit View Options Transfer Window Help

STEP 7: Install Perl programs.
  Installing /usr/iw-home/httpd/iw-bin/cci_javascript.rpl to /usr/iw-home/httpd/iw-bin
/cci_javascript.pl...
  Installing /usr/iw-home/install/upgrade_pw_config.rpl to /usr/iw-home/install/upgrad
e_pw_config.pl...

STEP 8: Create group members for access control
The following files will contain group membership information for
Administrators, Editors and Authors. YOU MAY MODIFY THEM BEFORE OR
WHILE Interwoven TeamXpress IS RUNNING BY EDITING THE FILES DIRECTLY.

  /usr/iw-home/conf/roles/master.uid
  /usr/iw-home/conf/roles/admin.uid
  /usr/iw-home/conf/roles/editor.uid
  /usr/iw-home/conf/roles/author.uid

Please select your option for initial <master> group setup
[1] Assign everyone in my /etc/passwd file to be an master.
[2] I'd like to manually assign a few members now.

Please enter your choice: [1] 2

>>> Please type a list of user names separated by spaces
>>> for the <master> group
>>> Enter master list: [root] chris pat andre
Done creating /usr/iw-home/conf/roles/master.uid

Ready 38, 39 38 Rows, 92 Cols VT100
```

*Assigning roles to users*

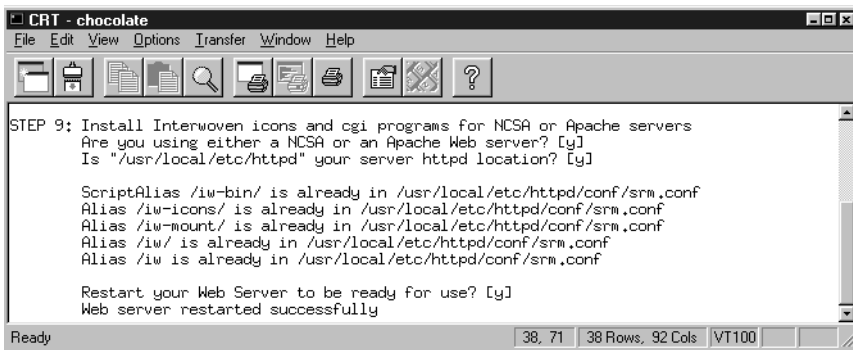
6. During installation, TeamXpress can install aliases required for the TeamXpress browser-based GUI to work with NCSA or Apache Web servers. This allows you to access TeamXpress directly from within the Web server. The TeamXpress installation script will ask:

Are you using NCSA or Apache Web server? [y]

If you are using an NCSA or Apache Web server and want to let TeamXpress install these aliases for you, type **y**. If you are using an NCSA or Apache Web server and do not want to let TeamXpress install these aliases, type **n**. If you are using a Netscape Web server, type **n**.

If you accept this option, you must provide the Web server's `httpd` location. If you do not accept it, you must configure your Web server manually after the TeamXpress installation has finished (see page 23).

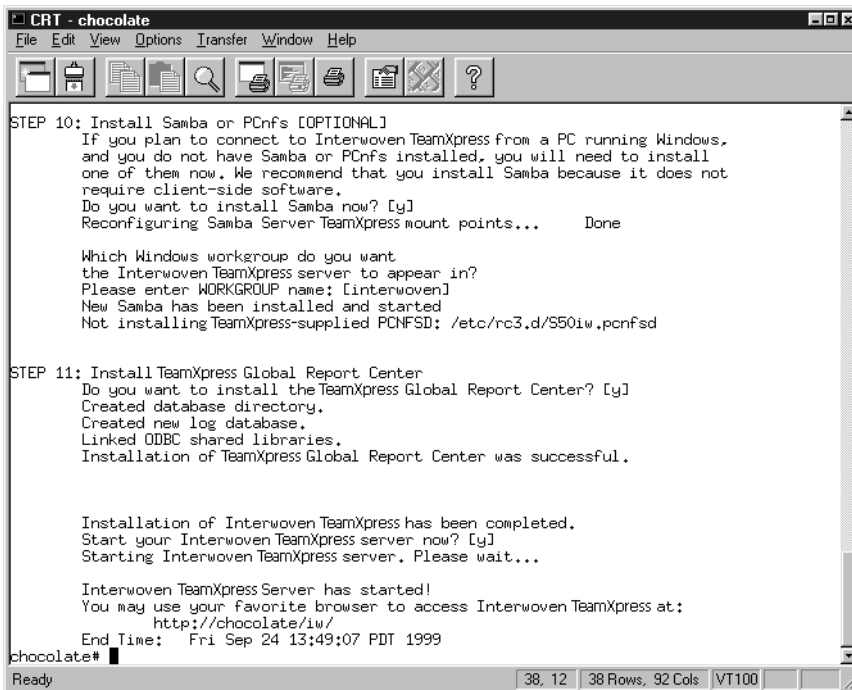
The Web server will have to be restarted after you use `iwinstall` to make your changes become effective. TeamXpress can do this automatically if your `http` server has a restart script and you select the [y] option when prompted by `iwinstall`. You can bypass this step by typing **n** and manually restart the Web server at a later time.



*Installing Web server icons and CGI programs*

7. During installation you will also have the opportunity to install Samba or pcnfsd for Macintosh and Windows access to the TeamXpress server via the file system interface. Samba is the default option. If you decide to install Samba, you must indicate a workgroup name. The remainder of the Samba installation is automatic. If you decide to install pcnfsd, the entire installation is automatic.

The installation script will also ask you if you want to install the Global Report Center. Type **y** to install the Global Report Center.



```

CRT - chocolate
File Edit View Options Transfer Window Help
STEP 10: Install Samba or PCnfs [OPTIONAL]
  If you plan to connect to Interwoven TeamXpress from a PC running Windows,
  and you do not have Samba or PCnfs installed, you will need to install
  one of them now. We recommend that you install Samba because it does not
  require client-side software.
  Do you want to install Samba now? [y]
  Reconfiguring Samba Server TeamXpress mount points... Done

  Which Windows workgroup do you want
  the Interwoven TeamXpress server to appear in?
  Please enter WORKGROUP name: [interwoven]
  New Samba has been installed and started
  Not installing TeamXpress-supplied PCNFSD: /etc/rc3.d/S50iw.pcnfsd

STEP 11: Install TeamXpress Global Report Center
  Do you want to install the TeamXpress Global Report Center? [y]
  Created database directory.
  Created new log database.
  Linked ODBC shared libraries.
  Installation of TeamXpress Global Report Center was successful.

  Installation of Interwoven TeamXpress has been completed.
  Start your Interwoven TeamXpress server now? [y]
  Starting Interwoven TeamXpress server. Please wait...

  Interwoven TeamXpress Server has started!
  You may use your favorite browser to access Interwoven TeamXpress at:
  http://chocolate/iw/
  End Time: Fri Sep 24 13:49:07 PDT 1999
chocolate#
  
```

### *Installing Samba and the Global Report Center*

8. Reboot the server.

The entire installation process is recorded in a log file that can be viewed as standard text called `iw-home/install/iwinstall.log`.

## Installing TeamXpress Templating

See the *TeamXpress Templating and Deployment Guide* for information about installing TeamXpress Templating.

## Configuring Webservers

This section describes how to configure your system's Web server after both TeamXpress and the Web server are installed.

No matter which Web server is installed on your system, you must ensure that the Web server `httpd` user name is included in the `iw-home/conf/roles/master.uid` file. If this user is not included in `master.uid`, the `iwjobc` command fails, and Web-based workflow does not function due to a failed security check. During TeamXpress installation, the user `nobody` is added to `master.uid`. If your system uses a different user name for the Web server `httpd`, you must edit `master.uid` so that it contains the correct name for your system.

### NCSA or Apache Web Servers

If you are using an NCSA or Apache Web server, the TeamXpress installation script allows you to automatically add aliases to the `srm.conf` file. If you choose not to add these aliases during installation, or you are using a Netscape Web server, you must add them manually.

#### Aliases

For NCSA or Apache Web servers, add the following lines to the `srm.conf` file (typically located in `/usr/local/etc/httpd/conf` or `/usr/local/apache/etc`):

<code>ScriptAlias</code>	<code>/iw-bin/</code>	<code>/usr/iw-home/httpd/iw-bin/</code>
<code>Alias</code>	<code>/iw-icons/</code>	<code>/usr/iw-home/httpd/iw-icons/</code>
<code>Alias</code>	<code>/iw-mount/</code>	<code>/.iwmnt/</code>
<code>Alias</code>	<code>/iw/</code>	<code>/usr/iw-home/httpd/iw/</code>
<code>Alias</code>	<code>/iw</code>	<code>/usr/iw-home/httpd/iw/</code>

### CGI Programs

If you need to execute CGI programs in TeamXpress workareas to test your Web site's CGI, you must also add the following lines:

```
<Location ~ "/iw-mount/./cgi-bin/.*">  
SetHandler cgi-script  
</Location>
```

All directories named `cgi-bin` will now be executable. If your CGI programs do not reside in `cgi-bin`, or if you only want certain `cgi-bin` directories to be executable, you must edit the `Location` line accordingly.

After you have added the necessary lines, you must restart the Web server for changes to take effect.

If you reinstall your Web server or lose your automatically modified configuration files, you can refer to the installation log to review the changes previously made.

### Server-Side Includes and Secure Socket Layer

Because server-side include requests do not go through the proxy server, you must install TeamXpress's redirector module to enable SmartContext QA for server-side includes.

**Note:** If your Web site does not use server-side includes, you do not need to install this module.

You have finished configuring your Web server. You will now need to configure Samba (see page 31) and set up TeamXpress clients (see page 32).

### Web Server Auto-Start

After setting up NCSA or Apache, you can optionally configure the Web server to start automatically whenever the system is booted. Consult your Web server's documentation for details.

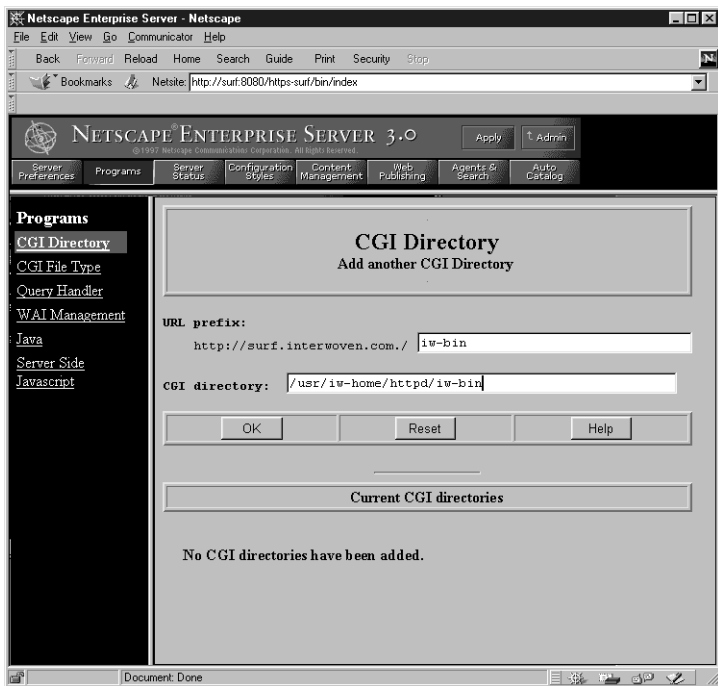


## Netscape Enterprise Server

### Aliases

For Netscape Web servers, use the Server Administrator screens to specify the following aliases:

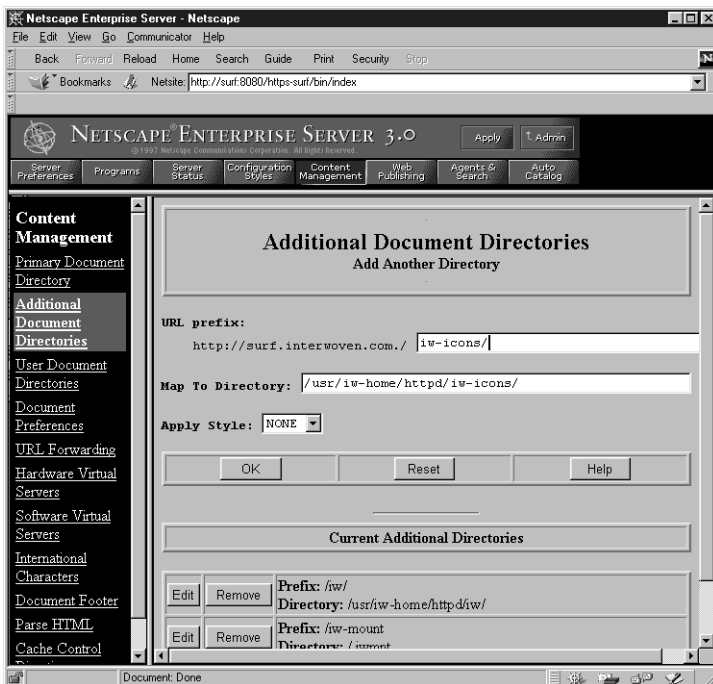
1. From your Netscape Server Administrator console, select the server to configure.
2. Select **Programs** from the top frame.
3. Select **CGI Filetype** from the left frame. In the CGI file type dialog box, click the **Yes** button for “Activate CGI as a filetype?” The Web server will now run, rather than download, all `.exe`, `.cgi`, and `.bat` files.
4. Select **CGI Directory** from the left frame. Create a new alias. The URL prefix is `/iw-bin/` and the CGI directory is `/installation_directory/iw-home/httpd/iw-bin/`.



*Adding the iw-bin alias*

5. Select **Content Management** from the top frame.
6. Select **Additional Document Directories** in the left frame. Add three new document directories:

URL Prefix	Map to Directory
/iw/	/installation_directory/iw-home/httpd/iw/
/iw-icons/	/installation_directory/iw-home/httpd/iw-icons/
/iw-mount/	/.iwmnt/



*Adding document directory aliases*

## MIME Types

TeamXpress also requires two changes to the Netscape server’s MIME types configuration.

1. Select **Server Preferences** from the top frame of the Netscape Server Administrator console.
2. Select **MIME Types** from the left frame.



Viewing global MIME types

3. Remove `exe` from the “magnus-internal/cgi” entry.
4. Add `exe` to the “application/octet-stream” entry.



*Editing a MIME type*

You can also make these changes by editing Netscape's `mime.type` file manually.

## CGI Programs

If you need to execute CGI programs in TeamXpress workareas to test your Web site's CGI, you must manually edit your `obj.conf` file. Add the following lines to the end of `obj.conf`:

```
<Object ppath="/.iwmnt/*/cgi-bin/*">
ObjectType fn="force-type" type="magnus-internal/cgi"
Service fn="send-cgi"
</Object>
```

### Server-Side Includes and Secure Socket Layer

Because server-side include requests do not go through the proxy server, you must install TeamXpress's redirector module, `iw-home/lib/iwrewrite.nsapi.solaris.so`, to enable SmartContext QA for server-side includes. See the `README` file in `iw-home/lib` for installation instructions.

**Note:** If your Web site does not use server-side includes, you do not need to install this module.

To install the redirector module:

1. Copy the `iwrewrite.nsapi.solaris.so` file from `iw-home/lib` to the Netscape server's `https-servername` directory.
2. Add the following lines to the Netscape server's `obj.conf` file:

- a. In the `init` section add the two `init` directives shown below, substituting the pathname to `iwrewrite.nsapi.solaris.so` as appropriate for your installation.

First entry (all on one line):

```
Init fn="load-modules" shlib="/installation_directory/iw-home/lib/  
iwrewrite.nsapi.solaris.so" funcs="iwrewrite"
```

Second entry:

```
Init fn="iwrewrite"
```

- b. In the `default` object description, add the following `NameTrans` as the first `NameTrans`, superseding all others:

```
NameTrans fn="iwrewrite"
```

3. Stop and start the server.

The SSI redirector reads the same branch and docroot configuration from `iw.cfg` that `iwproxy` uses. If you update the `iwproxy` mappings in `iw.cfg` you will have to explicitly stop and start the Web server again to reflect the changes.

You have finished configuring your Web server. You will now need to configure Samba (see page 31), and set up TeamXpress clients (see page 32).

### Redirecting NSAPI HTTPS Requests

If your system contains two Web servers configured such that:

- one is a secure NES server set up to process HTTPS requests
- the other is a non-secure server of any type that processes TeamXpress proxy HTTP requests,

then you can configure TeamXpress to redirect HTTPS requests so that they are served from the proxy over HTTP. Set the following entry in the `[nsapi]` section of `iw.cfg` to enable redirection:

```
redirect_https_to_http=yes
```

When redirection is enabled, all HTTPS requests originating from the browser and received by the secure server's NSAPI plugin are redirected to the proxy. The proxy then sends the requests to the non-secure TeamXpress server just as it would any request originating from the browser. The following conditions must exist for redirection to occur:

- The `iw.cfg` file must be configured as shown above.
- The original request must be an HTTPS request.
- The original request must be for a file in a TeamXpress area.

For example, if the NSAPI plugin on the secure server receives an HTTPS request for a file in a TeamXpress area such as:

```
https://www.example.com/iw-mount/branch1/STAGING/bio.html
```

the request is redirected to `iwproxy` as follows:

```
http://www.example.com:proxyport/iw-mount/branch1/STAGING/bio.html
```

During redirection, some browsers could display a message warning that the request is being sent to an insecure document. This is normal browser behavior. If you see such a message, click **OK** to proceed. Note that HTTPS requests that are redirected to the proxy no longer have full HTTPS security.

## Configuring Samba

TeamXpress includes an optional installation for Samba. Samba is a network protocol that makes it possible for PCs to mount UNIX drives as Windows networked drives. Samba is a server-side solution which does not require client software. This makes TeamXpress relatively easy to administer because the system administrator does not need to update and correct issues with client computers.

The default Samba configuration file is `iw-home/iw-samba/lib/iw.smb.conf`. If you already have Samba installed, the file could be in `/usr/local/samba/etc/smb.conf` or `/usr/local/samba/lib/smb.conf`. No matter where the file is located, you will need to set one global option:

```
security = user
```

and add the following section to your Samba configuration file:

```
[iwmain]
comment = directory main branch of TeamXpress file system
public = no
create mode = 0775
force create mode = 0775
force directory mode = 0775
writable = yes
locking = no
share modes = no
preserve case = yes
short preserve case = yes
path = /iwmnt/default/main
```

Where:

`security=user`

`public=no`

`create mode=0775`

grants access privileges on a user by user basis

means that authentication is required to view the contents of the Samba mount.

sets UNIX permissions on all newly created files to 0775.



<code>force create mode=0775</code>	sets the minimum UNIX permissions on all newly created files to 0775.
<code>force directory mode=0775</code>	sets the minimum UNIX permissions on all newly created files to 0775.
<code>writable=yes</code>	allows users to write to the Samba mount ( <code>writable=no</code> disables the ability to do writes via Samba).
<code>locking=no</code>	turns off locking. Locking must be turned off, because TeamXpress has its own locking.
<code>share modes=no</code>	turns off SMB deny modes.
<code>preserve case=yes</code>	keeps the original case of file names.
<code>short preserve case=yes</code>	keeps the case of 8.3 filenames rather than displaying them as all uppercase.
<code>path=/iwmnt/default/main</code>	is the directory to be mounted.

To mount any other TeamXpress area, copy and paste the above section in your Samba configuration file. Change its name from `[iwmain]` to a unique identifier, and edit the `path` line to point to the area to be mounted.

After making changes to the `iw.smb.conf` file, you must restart Samba for your changes to take effect:

```
% /etc/init.d/iw.samba stop
```

```
% /etc/init.d/iw.samba start
```

## Setting Up TeamXpress Clients

Now that you have installed TeamXpress and configured your Web server, you should set up at least one TeamXpress client to access TeamXpress by both the graphical user interface and the file system interface. This can be used as a first-level check that you have installed and configured TeamXpress properly.



## The Graphical User Interface

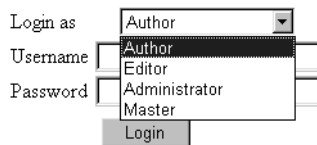
The TeamXpress graphical user interface can be accessed by users through a JavaScript capable Web browser. In order to log in, you must be a TeamXpress user. If you have not yet added users to TeamXpress or changed your own user status, you should do so now (see Chapter 3, “Setting Up Users”). If you do not add users or change your own user status, you are limited to the TeamXpress Master role, which is the default role for the user “root.”

1. To access TeamXpress from a client computer, start your Web browser and enter the URL:

`http://TeamXpress_hostname/iw/`

where `TeamXpress_hostname` is the name of the TeamXpress server (e.g., `TeamXpress1.example.com`). You may want to bookmark this URL for future use.

2. The TeamXpress login screen will appear. If this is your first login and you are using Windows, then the TeamXpress helper application, LaunchPad, will auto-install on your computer.
3. In the login screen, select your user type (Author, Editor, Administrator, or Master) using the pull-down menu. Enter your UNIX user name and password and click **Login**.



The image shows a web form for logging into TeamXpress. It has four main sections: 'Login as', 'Username', 'Password', and a 'Login' button. The 'Login as' section has a pull-down menu with 'Author' selected. The 'Username' and 'Password' sections have text input fields. The 'Login' button is a grey rectangular button.

*The TeamXpress Login screen*

## Installing LaunchPad

Before you can edit files using TeamXpress, you will need to install TeamXpress's helper application, LaunchPad.<sup>1</sup> To download LaunchPad:

1. Log in to TeamXpress through the browser interface.
2. Select **LaunchPad Setup** from the **Edit** menu. Instructions for installing LaunchPad are contained in the "Getting Started" chapter of the the *TeamXpress User's Guide*.

## The File System Interface

The file system interface allows you to manage your Web content in TeamXpress as if it were on a mounted drive on the network. The file system interface is used primarily for file management functions such as moving and copying files, and it can also be used to edit files. It also allows the use of links checkers and scripts that need to be able to access and/or create files. In addition, most TeamXpress operations can be performed from a UNIX command-line interface.

### Windows 95, 98, NT, and 2000

The first time you access TeamXpress from Windows, you may need to mount the TeamXpress server as a network drive. The instructions below show how to access TeamXpress with:

- A networked computer via Samba or a NFS client.
- A networked computer with FTP.

### Microsoft Networks (via Samba)

To access TeamXpress from Windows via Samba, use Windows Explorer to locate the TeamXpress server.

1. Click the **Start** button and select **Find...Computer**.
2. Type the name of the TeamXpress server in the **Named:** box. Click the **Find Now** button.

---

1. If you are using Windows, LaunchPad will auto-install itself on your computer. You do not need to download LaunchPad. See page 70 for details about disabling LaunchPad autoinstallation.

3. Double-click the name of the TeamXpress server when it appears in the list. The server window will open.
4. Double-click the TeamXpress mount point directory (usually `iwmain`). Navigate through the TeamXpress directory structure to find your workarea. Within your workarea, you can edit, move, or rename TeamXpress files as you would any other files. You can also drag and drop files and directories from your local hard drive to directories in your workarea.

### Creating Shortcuts and Mapping Network Drives

To simplify future access to your workarea or to commonly used directories in your workarea, create a shortcut to the directory or directories you access frequently and put it on your desktop.

To simplify future access to the TeamXpress server, and to enable LaunchPad Direct Edit (see the *TeamXpress User's Guide*), map a network drive to the TeamXpress server. To do this from Windows Explorer:

1. Click the **Start** button and select **Find...Computer**.
2. Type the name of the TeamXpress server in the **Named:** box. Click the **Find Now** button.
3. Double-click the name of the TeamXpress server when it appears in the list. The server window will open.
4. Select the TeamXpress mount point (usually `iwmain`) and click the *right* mouse button.
5. Select **Map Network Drive**.
6. Select the letter of the drive you want to map the TeamXpress server to. Click **OK**.

If you are using a WINS server, add the TeamXpress server to its host list. Alternatively, you can add the IP address and host name to the `Windows/LMHOSTS` file. If you are not using a WINS server, you can modify your PC to use the TeamXpress server as a WINS server. In Windows 95:

1. Select the **Network Neighborhood** icon.
2. Click the *right* mouse button and select **Properties**.
3. Select **TCP/IP** from the list.

4. Select **WINS Configuration**.
5. Select **Enable WINS Resolution**.
6. Enter the IP address of the TeamXpress server as the Primary WINS Server.
7. Click **OK** and restart your PC.

After finding your workarea, create a shortcut from the workarea to your local desktop.

### Troubleshooting Windows Networking

If you cannot find the TeamXpress server through Windows Networking, check to see if you have NetBEUI installed. If you do, uninstall it if at all possible.

To uninstall NetBEUI:

1. Select **Start > Settings > Control Panels**.
2. Double-click on the **Network** control panel icon.
3. Select the **Protocols** tab. From the list of adapters and their associated protocols, find the local client Ethernet card adapter. If NetBEUI precedes TCP/IP in the list of bound protocols, you will need to remove it.
4. To remove NetBEUI, select the NetBEUI protocol in the list. Select **Remove**.
5. Close the remaining Network dialog windows.
6. Reboot your computer.

Upon reboot, the client Ethernet card will use TCP/IP to send and receive network transmissions. You will now be able to use the Windows **Start > Find > Computer** utility to locate the TeamXpress server.

You can also use the Advanced tab of the protocol settings to specify TCP/IP as the default protocol binding for Windows Networking, but this solution is not as reliable because it might be upset as network cards are changed and protocols are added and removed.

## Configuring PDC

You can configure TeamXpress to use the Windows NT Primary Domain Controller (PDC) for name and password authentication. This configuration eliminates the need for users to enter their passwords manually whenever they connect to the TeamXpress server.

To use the Windows NT PDC for authentication, modify your Samba configuration as follows:

1. In the `[global]` section `iw.smb.conf`, modify or add the following:

```
# Select a Windows NT PDC for your password server
password server = DNS_name_of_your_PDC
# Use share level security
security = server
# Use password encryption
encrypt passwords = yes
```

2. Stop and restart Samba:

```
% /etc/init.d/iw.samba stop
% /etc/init.d/iw.samba start
```

## NFS Clients

If you are using an NFS client, follow your program's setup instructions to mount `/iwserver` on the TeamXpress server as a networked drive. If you can, you should modify the client configuration to *NFS Version 2* and turn off NFS locking (sharing) on the client. If you do not turn off locking, operations might freeze for long periods of time.

## FTP Clients

If you are using an FTP client, follow your program's setup instructions to install the software. Log in to the TeamXpress server using your UNIX login and password and navigate to your workarea, located at:

```
/iwmnt/default/main/dev/WORKAREA/workareaname
```

## Macintosh

To use the TeamXpress file system interface for Macintosh, you need to have an AppleShare server set up for the server that is running TeamXpress.

1. In the Chooser, select **AppleShare**.
2. Select the name of the TeamXpress server. Click **OK**.
3. If you are asked for your username and password, enter your TeamXpress username and password, and click **OK**.
4. Select the items you want to use, and click **OK**. The TeamXpress server will appear on your desktop.

## UNIX

To access the TeamXpress server via UNIX, log in to the TeamXpress server using your TeamXpress username and password.

If you have UNIX clients that will be accessing TeamXpress's file system, you can mount the TeamXpress directory, or configure the client machine to automatically mount the file system at boot time.

To mount the TeamXpress directory, issue the commands:

```
% mkdir /iwmnt
% mount -overs=2 servername:/iwserver /iwmnt
```

where *servername* is the name of the server on which TeamXpress is running.

Alternatively, you can mount a subdirectory of *iwserver*, e.g., a specific branch or workarea. To mount a workarea, issue the commands:

```
% mkdir /iwmnt
% mount -overs=2 servername:/iwserver/default/branchpath/WORKAREA/workareaname /iwmnt
```

where *servername* is the name of the server on which TeamXpress is running, *branchpath* is the path of the branch your workarea is on (e.g., *main/dev/intranet*), and *workareaname* is the name of your workarea.

To set up a UNIX client (including the TeamXpress server itself) to mount the TeamXpress directory at boot time, edit */etc/vfstab* to include the following line:

```
servername:/iwservr - /iwmnt nfs - yes vers=2,bg
```

## Loading Content

When you install TeamXpress, the main branch is automatically created. It contains a staging area and an empty initial edition. Before you start using TeamXpress for production, however, you must transfer your current Web site files into TeamXpress. Note that in TeamXpress, you cannot create workareas off of the main branch. Additionally, depending on the licensing agreement, you may not be able to add branches or sub-branches. To populate TeamXpress with your content<sup>1</sup> you will need to do the following (detailed directions for each step follow):

1. Create a sub-branch for your Web developers.
2. Create a TeamXpress workarea on this branch.
3. Populate the newly created workarea with existing files.
4. Set permissions on the files, or configure a submit filter (see “Changing File Attributes During Submit” on page 102).
5. Submit the workarea to the staging area.
6. Publish an edition from the staging area.

The newly published edition will then become the foundation of all subsequent work done in TeamXpress.

---

1. For issues regarding multibyte content, see Appendix D, “Internationalization”

## Creating a Sub-branch

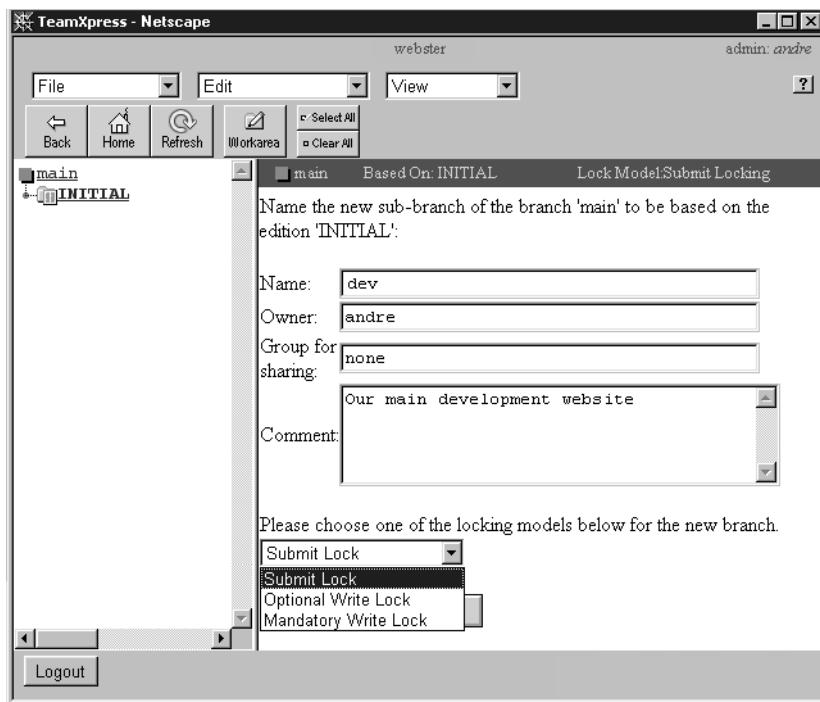
Interwoven recommends that all development take place on sub-branches. The main branch is not usually used for development for several reasons. First, it requires a user with Master privileges to administer. In addition, if you are using TeamXpress to develop multiple Web sites, development of one Web site on the main branch and other Web sites on sub-branches may create a false hierarchy of branches—the sub-branch will not necessarily bear any relation to the parent branch. In TeamXpress, you cannot create workareas off of the main branch. Additionally, depending on the licensing agreement, you may not be able to add branches or sub-branches.

To create a sub-branch using the TeamXpress GUI:

1. Log in to the GUI as a Master user.
2. Select **File > New Branch**. Because there is only one edition on the parent branch (the empty INITIAL edition), this sub-branch will be based on that edition.
3. A Create Branch window will appear.
4. Type in the name of the branch in the **Name** box.  
Avoid using spaces and most punctuation characters in branch names. Branch names should consist only of alphanumeric characters, hyphens, and underscores.
5. Your username will appear in the **Owner** box. If you want to assign the branch to someone else, type the owner's name in this box.
6. If you want this branch to have multiple Administrators, type the name of the group who will be able to administer this branch in the **Group for Sharing** box. The Administrator or Administrators of this branch will be able to create workareas and sub-branches of development. For more information on Administrator privileges, see page 50 and page 56.
7. Use the pull-down menu to select the type of locking you want to be used on this branch (see the *TeamXpress User's Guide* for an explanation of the different types of locking).
8. Add any comments in the **Comment** box (comments cannot be changed). Click **OK**.

Your newly created branch will contain no workareas, a staging area, and an empty edition called INITIAL.





*Create Branch Window*

You can also use the `iwmkbr` command-line tool to create a new branch.

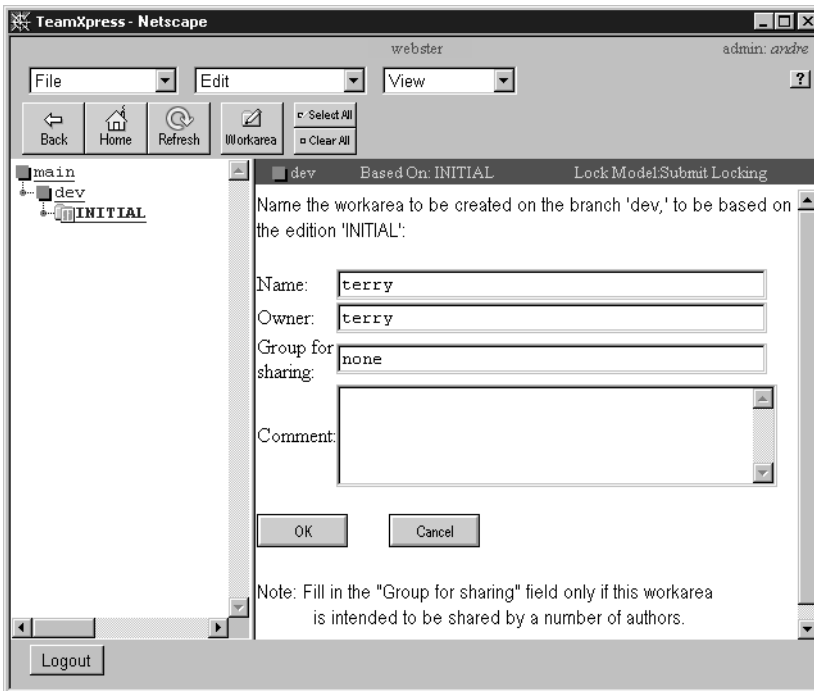
## Creating a Workarea

To create a workarea using the TeamXpress GUI:

1. Click the name of the sub-branch you just created, to navigate into the branch.
2. Select **File > New Workarea**. Because there is only an empty edition on this branch, TeamXpress will create an empty workarea.
3. The Create Workarea window will appear. Type in the name you want to give the workarea in the Name box, and the username of the workarea's owner in the Owner box.

Avoid using spaces and most punctuation characters in workarea names. Workarea names should consist only of alphanumeric characters, hyphens, and underscores.

4. If you want a group to be able to share this workarea, type the name of the group in the Group for Sharing box. If you want this workarea to be private, so that only the owner can modify files in it, leave the default group (none) selected.
5. Add any comments in the Comments box. Click **OK**.



*Create Workarea Window*

You can also use the `iwmkwa` command-line tool.

## Populating the First Workarea

From the UNIXfile system (via telnet, NFS, Samba, etc.):

1. Log in as `root`.
2. Copy all of the original Web site files into the new workarea (default location):
 

```
/iwmnt/default/main/dev/WORKAREA/workareaname
```

where *branchname* is the name of the newly created sub-branch and *workareaname* is the name of the newly created workarea on the sub-branch.

When copying files, use `tar` to maintain file permissions and timestamps (that is, make a tar file of your Web site content, copy the file into the workarea, then untar the file). When you're done, navigate into the workarea and double-check file permissions before submitting the files to the staging area.

3. Set permissions on the files in your Web site. Use `chown` and `chgrp` to limit access to files by changing the owners and groups for these files. For more information on the `chown` and `chgrp` commands, consult a UNIX reference manual.

Because TeamXpress considers a change in permissions to be a change in the file, TeamXpress will store a new version of the file when you change its permissions (new versions are created at the time files are submitted to the staging area). If you wait to set permissions until after your files have been imported into a workarea and submitted to the staging area, you can create a large number of extra versions and unnecessarily clutter each file's version history. To avoid creating unnecessary versions, set permissions immediately after you populate the workarea (but before you submit the files). Interwoven recommends that you configure a submit filter to automate this process (see page 102), but you can also set permissions manually.

! Be sure to set permissions **before** you submit files to the staging area for the first time.

## Submitting Files to the Staging Area

Now that you have populated your workarea with your Web site content, you can submit it to the staging area. You need to submit your content to the staging area before you can publish an edition, which you can use as the basis of all future workareas.

To submit the contents of your workarea to the staging area via the TeamXpress GUI:

1. Go to the top level of the workarea. Do not select any checkboxes.
2. Select **File > Submit Direct**.<sup>1</sup> A dialog box will appear asking if you want to submit the entire directory.

---

1. For first-time submissions of large numbers of files, you should use the direct Submit option rather than workflow Submit.

3. Click **OK**. A Submit window will appear.
4. Enter any comments you have in the comment boxes. The Submit window contains two sections: a Submit Comments section, which consists of a section where you can enter comments for the entire Submit operation and a field where you can enter keywords, e.g., for automatic triggers, and an Individual Comments section, where you can attach comments to each file.
5. Click the **Submit** button.

applies to the entire Submit operation

applies to individual files

*Submit Comments Window*

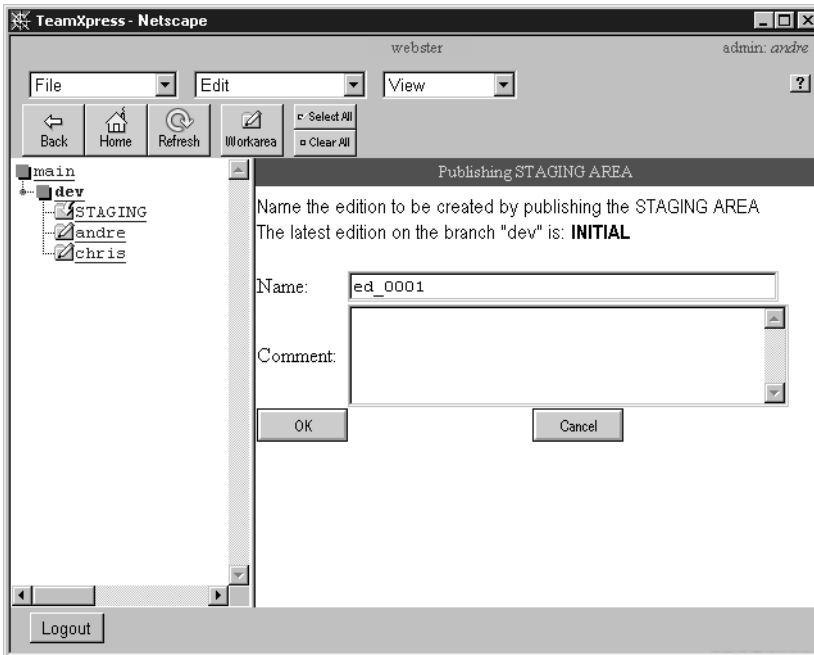
You can also use the `iwsubmit` command-line tool to submit files to the staging area.

## Publishing a New Edition

Publishing an edition creates a snapshot of the staging area at the time of publication. These editions can be used as checkpoints. As part of your initial installation process, you should create an edition to record the state of your Web site at the time that you installed TeamXpress. You can use this new edition as the basis for the other workareas you create on this branch.

To create a new edition from the contents of the staging area via the GUI:

1. Select **File > Publish** menu from anywhere within your branch. The Publish window will appear.
2. TeamXpress will suggest a name for the new edition. If you want to give the edition a different name, enter the name of the new edition in the **Name** box in the Publish window.



*Publish Window*

3. Enter any comments you have in the **Comments** box.
4. Click **OK**. The staging area will be published as a new edition.

You can also use the `iwpublish` command-line tool to publish an edition.

## Upgrading TeamXpress

If you are upgrading TeamXpress, you do not need to reboot the server.

## Uninstalling TeamXpress

To uninstall TeamXpress, you must log in as root. Issue the command:

```
% iwgethome
```

to find the TeamXpress installation directory `iw-home`. Run:

```
% iw-home/install/iwuninstall
```

Use this command with extreme caution! Once completed, this cannot be reversed except by restoring files from your backups. This command will remove all traces of TeamXpress from the host machine.





# Setting Up Users

---

## Overview

Access to TeamXpress is governed by two factors: UNIX-related permissions, and TeamXpress access privileges. UNIX file permissions control who has access to individual files and directories. UNIX password authentication is used when logging in to TeamXpress. However, TeamXpress access privileges govern who can log in under various roles, and who has access to branches and workareas. For example, to edit a file in a workarea, a user must both be able to access that workarea (through TeamXpress access privileges), and have permissions for that file and its parent directory (through UNIX permissions). For a complete list of the TeamXpress and UNIX permissions needed to perform any action in TeamXpress, see page 56.

When adding a new user, you need to take three factors into account:

- Whether the user has access to the server.
- The role the user will play in your Web site operations.
- The portion of the Web site the user will be editing.

If the user does not have access to the TeamXpress server, you will need to add him (see page 51). To decide what TeamXpress role best corresponds to the role he will play in your Web site operations, see page 50.

To decide what groups the new user needs to belong to, and which workareas he needs to access, consider your existing groups and which portions of the Web site and which workareas they can access. Add the new user to the groups that work on the same portion of the Web site that he will be editing, and he will automatically have access to their workareas and to their Web site files. If the new user needs his own workarea, create a private or shared workarea for him, but make sure that he owns or has group-level access to the files that he will be editing. To change ownership or group access of files, see page 53.

## Determining User Roles

To facilitate workflow and security, TeamXpress provides users with four roles, each with varying levels of access to TeamXpress: Master, Administrator, Editor, and Author. To determine which role a user should have, consult this table and find the role that best fits the user's work.

<b>Author</b>	<b>Editor</b>	<b>Administrator</b>	<b>Master</b>
Owns content	Owns workareas	Owns branches	Owns main branch
Edits & creates files	Edits & creates files	Edits & creates files	Edits & creates files
Receives assignments (can assign files to self)	Assigns files	Assigns files	Assigns files
Work is approved by workarea owner	Approves/rejects work of Authors	Approves/rejects work of Authors	Approves/rejects work of Authors
	Uses advanced version management features	Uses advanced version management features	Uses advanced version management features
	Maintains content of workarea	Manages branch	Manages entire Web site
	Submits files to the staging area	Submits files to the staging area	Submits files to the staging area
	Publishes editions (optional)	Publishes editions	Publishes editions
		Creates and deletes workareas	Creates and deletes workareas
		Creates and deletes sub-branches	Creates and deletes sub-branches

In addition, Administrators can perform all the functions that Editors can. Master users can perform all the functions that Administrators and Editors can.

## Adding and Removing Users

### Adding Users

Before adding a user, check to see if the user is a UNIX user on the TeamXpress server. Always consult your system administrator before adding a user. If the user already exists, skip to Step 4.

To add a user to TeamXpress:

1. Add the user to the UNIX password file (`/etc/passwd`) for the TeamXpress server. **Always consult your system administrator before adding a user to this file.** Your system administrator may want to add the new user himself.

You need superuser privileges to add users to the TeamXpress server. To add a user to `/etc/passwd`, add a new line in the following format:

```
username:x:uid:gid:name:default_home_directory:shell
```

where *username* is the user's login name, *uid* is the user's user identifier, *gid* is the group identifier for the group that the user belongs to, *name* is the user's real name, *default\_home\_directory* is the user's home directory, and *shell* is the user's shell.

For example, Pat's entry might look like this:

```
pat:x:500:2000:Pat Smith:/u/iw/pat:/bin/csh
```

2. If applicable, add the user to the UNIX shadow file (`/etc/shadow`) for the TeamXpress server. The shadow file is used for encrypting login passwords.

To add a user to `/etc/shadow`, add a new line in the following format:

```
username:::::::::
```

that is, the user's username followed by eight colons.

3. Set the user's password with the `passwd` command:

```
% passwd username
```

You will be prompted to enter the user's password twice. You can assign the user a password which he can change later, or if the user is present, you can have him enter the password of his choice. If you look at the `etc/shadow` file again, you will find that the encrypted password now appears in the line that you added in Step 2.

1. Add the user's UNIX login name to the appropriate TeamXpress roles file(s).

The four roles files are in the directory `iw-home/conf/roles`:

```
master.uid
admin.uid
editor.uid
author.uid
```

Each file contains a list of the users who have privileges for that role, one user to a line.



```
Telnet - athena
Connect Edit Terminal Help
sol:teamxpress:/private/iw-home/conf/roles:118% more editor.uid
root
test1
test2
test3
test4
questacc
dialins
andre
chris
chris
bobbie
pat
nobody
--More-- (11%)
```

*An editor.uid file*

To give a user multiple roles, include his name in multiple `.uid` files. For example, an Editor may also be able to log in as an Author. A Master user should be able to log in with any role, so you will need to include the name of each Master user in each `.uid` file. Each user's TeamXpress password will be the same as his UNIX password.

After you have edited the TeamXpress roles files, you will need to tell TeamXpress to reread them. type:

```
% iwreset
```

The TeamXpress server will return 0 on success, non-zero on failure.

At this point the new user will now be able to log in to TeamXpress, but he will not have access to any branches or workareas.

2. Add the user to the appropriate UNIX groups files. If you want the user to have access to a shared workarea, add him to the group that has access to that workarea. If the user is an Administrator, and you want him to have Administrator privileges for a branch, add him to the group of Administrators for that branch.

To add a user to a group, edit the `/etc/group` file. Locate the name of the group you want to add the user to, and add his name to the list of usernames that follows it. Usernames must be separated by commas.

3. If you want the user to own a workarea, create a workarea for him on the sub-branch where he will be working (see the *TeamXpress User's Guide*).

## Deleting Users

To remove a user from TeamXpress, remove the user from the `.uid` files for each role that the user. Then use `iwreset` to reread the roles files. You might also want to remove the user's UNIX user account for the TeamXpress server:

1. Delete his username from any groups that he belongs to.
2. Remove him from the `/etc/passwd` file. Always consult your system administrator before altering the `/etc/passwd` file in any way.

## Access Control

To control access to individual files or directories, use `chmod` to change the permission bits. Use `chown` to change the ownership of files or directories, and `chgrp` to change the group. For more information on `chmod`, `chown`, and `chgrp`, consult a UNIX reference manual.

TeamXpress considers a change in file permissions to be a modification of the file. For the most efficient use of disk space, set the permission bits on a file when it is created or imported into TeamXpress *before* you submit it to the staging area for the first time. You may also want to use a submit filter (see "Changing File Attributes During Submit" on page 102) to change permissions.

**Note:** `chmod`, `chown`, and `chgrp` commands, when used recursively, touch every file in the directory, whether they need to or not. This generates excess TeamXpress versions, which use disk space. Instead, target the command using `find`:

```
% find . ! -group webedit -exec chgrp webedit {} \;  
% find . ! -perm -g+w -exec chmod g_w {} \;
```

This will touch only the necessary files and not generate unnecessary versions.

## Group Membership

Many workareas are shared by groups. For a user to have access to a particular workarea, he must either be the owner of the workarea, or a member of the workarea's group. TeamXpress uses UNIX groups for access control. These groups can be managed with standard UNIX commands.

To add a user to a group, edit the `/etc/group` file. Locate the name of the group you want to add the user to, and add his name to the list of usernames that follows it. Usernames must be separated by commas.

To create a group, add a new line to the `/etc/group` file in the following format:

```
groupname:*:gid:username1,username2,username3
```

For example, the entry for the group "allauthors" might look like this:

```
allauthors:*:2000:pat,andre,chris
```

You can add as many users to a group as you want.

## Changing Group Ownership of Workareas

To change which group has access to a workarea:

1. Navigate into the directory containing the workarea.
2. Use the `chgrp` command to change the workarea's group. For more information on the `chgrp` command, consult a UNIX reference manual.

## Example

In this example, user Chris changes the group for one of the workareas on the main branch. First, he navigates into the directory containing the workareas. Then, he looks at the existing workareas and learns that Andre has two workareas, one private and one shared with the group `demoauthor`. Chris has one private workarea, `wal`. He uses the `chgrp` command to change the group on his workarea, then checks the results.

```
% cd /iwmnt/default/main/dev/WORKAREA
% ls -la
total 3
drwxrwxr-x  27 andre    nobody    512 Apr 23 17:07 andre1
drwxrwxr-x   3 andre    demoauthor 512 Apr 17 11:52 andre2
drwxrwxr-x   2 chris    nobody    512 Apr 17 11:37 wal
% chgrp demoauthor wal
% ls -la
total 3
drwxrwxr-x  27 andre    nobody    512 Apr 23 17:07 andre1
drwxrwxr-x   3 andre    demoauthor 512 Apr 17 11:52 andre2
drwxrwxr-x   2 chris    demoauthor 512 Apr 17 11:37 wal
```

## Checking User Roles

The TeamXpress Command Line Tool `iwckrole` allows you to check whether or not a user can log in with a certain role.

### Usage

```
iwckrole [-h|-v] role user
```

<code>-h</code>	Displays usage message.
<code>-v</code>	Displays version.
<code>role</code>	author, editor, admin OR master.
<code>user</code>	Username of the person whose role you are checking.

Exits with `YES` on successful authorization, `NO` on failure.

### Example

```
% iwckrole admin andre
```

returns:

```
YES
```

indicating that user “andre” can log in as an Administrator.

## User Role Privileges and Permissions

When a user tries to perform any action in TeamXpress, the TeamXpress server automatically checks to see whether or not he has permission to perform that action. TeamXpress checks the following factors:

- User roles
- Branch permissions
- Workarea permissions
- File permissions
- Directory permissions

Not all of these factors apply to every action. TeamXpress only checks the factors that apply to the action being attempted.

The table below lists which privileges a user must have in order to perform any action in TeamXpress. To find out whether a user will be able to perform a specific action, check the entry for that action under the user’s role and determine whether or not the specified conditions apply. All conditions listed in each box below must apply in order for a user to perform an action, unless otherwise specified.

*User roles:* If you are attempting to perform any of these actions through the GUI, you must be logged in with the role specified. If you are using the file system interface, you must be able to log in with the specified role.



*Branch permissions:* A user has branch permissions if he is either the primary owner of the branch or a parent branch, or if he belongs to the group that owns the branch or a parent branch. Master users automatically have branch permissions for all branches. Only Administrators and Master users can have branch permissions.

*Workarea permissions:* A user has workarea permissions if he is either the primary owner of a workarea, or if he belongs to the group that has access to the workarea. Workarea permissions are usually synonymous with read-write-access to the root directory of the workarea—the default setting for workarea permissions is 775. If different permissions are specified for the owner and group, some sections of the table below will not apply. If “world” is given read-write-execute permissions, then all users will be considered members of the workarea for those conditions indicated with asterisks (\*).

*File permissions:* File permissions are UNIX read-write-execute permissions (unless otherwise specified) to a file.

*Directory permissions:* Directory permissions are permissions (unless otherwise specified) to a directory.

	<b>Author</b>	<b>Editor</b>	<b>Administrator</b>	<b>Master</b>
Edit file <sup>1</sup>	workarea permissions* parent directory permissions (read/execute) file permissions (write)	workarea permissions* parent directory permissions (read/execute) file permissions (write)	workarea permissions* parent directory permissions (read/execute) file permissions (write)	workarea permissions* parent directory permissions (read/execute) file permissions (write)
View file	workarea permissions* parent directory permissions (read/execute) file permissions (read)	workarea permissions* parent directory permissions (read/execute) file permissions (read)	workarea permissions* parent directory permissions (read/execute) file permissions (read)	workarea permissions* parent directory permissions (read/execute) file permissions (read)



	Author	Editor	Administrator	Master
New file <sup>2</sup>	workarea permissions* parent directory permissions (read/write/execute)	workarea permissions* parent directory permissions (read/write/execute)	workarea permissions* parent directory permissions (read/write/execute)	workarea permissions* parent directory permissions (read/write/execute)
Move file Rename file <sup>3</sup>	workarea permissions* parent directory permissions (read/write/execute)	workarea permissions* parent directory permissions (read/write/execute)	workarea permissions* parent directory permissions (read/write/execute) OR branch permissions	Yes
New directory	workarea permissions* parent directory permissions (read/write/execute)	workarea permissions* parent directory permissions (read/write/execute)	workarea permissions* parent directory permissions (read/write/execute)	workarea permissions* parent directory permissions (read/write/execute)
Move directory Rename directory	workarea permissions* parent directory permissions (read/write/execute)	workarea permissions* parent directory permissions (read/write/execute)	workarea permissions* parent directory permissions (read/write/execute) OR branch permissions	Yes
Delete file <sup>4</sup>	No	workarea permissions* parent directory permissions (read/write/execute)	workarea permissions* parent directory permissions (read/write/execute) OR branch permissions	Yes

	<b>Author</b>	<b>Editor</b>	<b>Administrator</b>	<b>Master</b>
Delete directory	No	workarea permissions* parent directory permissions (read/write/execute)	workarea permissions* parent directory permissions (read/write/execute) OR branch permissions	Yes
Copy <sup>5</sup> (through the TeamXpress GUI)	file permissions (read) directory permissions (destination directory) workarea permissions	file permissions (read) directory permissions (destination directory) workarea permissions	file permissions (read) directory permissions (destination directory) workarea permissions	file permissions (read) directory permissions (destination directory) workarea permissions
Lock file <sup>6</sup>	No	workarea permissions parent directory permissions (read/execute) file permissions (write or ownership)	workarea permissions parent directory permissions (read/execute) file permissions (write or ownership)	workarea permissions parent directory permissions (read/execute) file permissions (write or ownership)
Unlock file	No	creator of lock OR owner of workarea	creator of lock OR owner of workarea OR branch permissions	Yes
Revert	No	workarea permissions	workarea permissions OR branch permissions	Yes



	Author	Editor	Administrator	Master
Get Latest	No	workarea permissions	workarea permissions OR branch permissions	Yes
Copy To	No	workarea permissions (destination workarea)	workarea permissions (destination workarea) OR branch permissions (destination)	Yes
Set Public/Private	No	workarea permissions	workarea permissions OR branch permissions	Yes
View History	No	workarea permissions (read)	workarea permissions (read)	Yes
Compare	No	workarea permissions (read)	workarea permissions (read)	Yes
List Modified	No	workarea permissions (read)	workarea permissions (read)	Yes
List Locks	No	workarea permissions (read)	workarea permissions (read)	Yes
View Submit Log	No	workarea permissions (read)	workarea permissions (read)	Yes
View Update Log	No	workarea permissions (read)	workarea permissions (read)	Yes
Create branch	No	No	branch permissions	Yes
Delete branch	No	No	branch permissions	Yes
Rename branch	No	No	branch permissions	Yes

	<b>Author</b>	<b>Editor</b>	<b>Administrator</b>	<b>Master</b>
Submit files	No	workarea permissions	workarea permissions OR branch permissions	Yes
Publish edition	No	workarea permissions for any workarea on the branch <sup>7</sup>	workarea permissions for any workarea on the branch OR branch permissions	Yes
Delete edition	No	No	branch permissions	Yes
Rename edition	No	No	branch permissions	Yes
Create workarea	No	No	branch permissions	Yes
Delete workarea	No	No	branch permissions	Yes
Rename workarea	No	No	branch permissions	Yes
View reports	No	No	Yes	Yes

1. The ability to edit a file only applies to files that are not already write-locked. If the file is write-locked, then only the lock owner can edit it.
2. The ability to create a file only applies to files that are not already write-locked. You cannot create a file with the same name as a file that is already write-locked. If an Author creates a file, the new file will be assigned to him.
3. The ability to rename or move a file only applies to files that are not already write-locked. If the file is write-locked, then only the lock owner can rename it. A file cannot be renamed with the name of a file that is locked. If an Author renames or moves a file, the renamed or moved version of the file will be assigned to him.
4. The ability to delete a file only applies to files that are not already write-locked. If the file is write-locked, then only the lock owner can delete it.
5. If an Author copies a file, the copied version of the file will be assigned to him.
6. The ability to lock a file only applies to files that are not already locked.
7. Publish capability for Editors may be disabled on a global basis.



# Configuring the TeamXpress Server

Most of the settings for the TeamXpress server are configured in the TeamXpress main configuration file, `/etc/iw.cfg` (default location). Some settings also use

`iw-home/local/config/submit.cfg`, `iw-home/local/config/autopriivate.cfg`, `iw-home/local/config/iwtemplates.cfg`, and `iw-home/local/iwprofiles/`.

Changes to most of these configuration options take effect within a few minutes (although for options that affect the TeamXpress GUI, you may have to clear your browser cache in order to see the changes). For these options to take immediate effect, use the `iwreset` command-line tool (CLT). Configuration options that require TeamXpress to be restarted in order to take effect are marked throughout this chapter.

Option	Configuration file	Page
<b>Configuring GUI appearance</b>		
Configuring TeamXpress area labels	<code>iw.cfg</code>	page 65
Configuring the number of displayed editions	<code>iw.cfg</code>	page 68
Configuring the number of displayed versions	<code>iw.cfg</code>	page 68
Individual user home page settings	<code>iwprofiles</code>	page 69
<b>Configuring GUI functionality</b>		
Enabling/disabling Editor publish capability	<code>iw.cfg</code>	page 69
Selectively enabling or disabling SmartContext Editing	<code>iw.cfg</code>	page 69
Disabling LaunchPad autoinstallation	<code>loginlogo.html</code>	page 70
Configuring LaunchPad autoconfiguration	<code>launchpad.cfg</code>	page 70
Setting the number of GUI preview windows	<code>iw.cfg</code>	page 73
Adding custom menu items	<code>iw.cfg</code>	page 74
Configuring submit button behavior	<code>iw.cfg</code>	page 78



Option	Configuration file	Page
Disabling menu items	iw.cfg	page 79
Disabling directory operations	iw.cfg	page 81
Disabling unlocked file auto-upload	iw.cfg	page 82
Setting the number of jobs listed in the To Do List	iw.cfg	page 83
Configuring job attributes and filters	iw.cfg	page 83
<b>Configuring server functionality</b>		
Setting authentication type	iw.cfg	page 85
Configuring autocompression	iw.cfg	page 87
Setting the Web server UID	iw.cfg	page 88
Setting the main branch locking model, owner and group	iw.cfg	page 88
Configuring submit capabilities on locked files	iw.cfg	page 89
Configuring the events logged in the submit and update logs	iw.cfg	page 89
Setting branch and workarea security	iw.cfg	page 90
Setting default permissions	iw.cfg	page 91
Configuring group remapping	iw.cfg	page 91
Setting TeamXpress file locations	iw.cfg	page 92
Configuring Autoprivate	autoprivate.cfg	page 93
Configuring template files	iwtemplates.cfg	page 96
Configuring use of the proxy server	iw.cfg	page 98
<b>Configuring server performance</b>		
Setting cache size	iw.cfg	page 99
Setting RPC thread count	iw.cfg	page 99
Setting file system threadcount	iw.cfg	page 100
Setting file system active area cache	iw.cfg	page 100
Configuring throughput monitors	iw.cfg	page 101
Detecting low disk space and inodes	iw.cfg	page 101



Option	Configuration file	Page
<b>Configuring submit filtering</b>		page 102
Changing file attributes at submit time	submit.cfg	page 102
RCS macro expansion	submit.cfg	page 107
<b>Configuring the TeamXpress proxy server</b>		page 111
Configuring proxy server operation	iw.cfg	page 113
Resolving relative and absolute paths	iw.cfg	page 113
Resolving fully-qualified paths	iw.cfg	page 117
Redirecting TeamXpress views to different areas	iw.cfg	page 121
Configuring TeamXpress to use different Web servers	iw.cfg	page 124
Configuring external remappings	iw.cfg	page 125
Host header remappings	iw.cfg	page 126
Configuring SSI remappings	iw.cfg	page 127
Configuring proxy failover	iw.cfg	page 127
<b>Configuring TeamXpress Failsafe</b>		page 130
Disabling Failsafe	iw.cfg	page 133
Setting backing cache size	iw.cfg	page 133
Setting backing cache location	iw.cfg	page 134
Setting backing cache flush frequency	iw.cfg	page 135

## Configuring GUI Appearance

### Configuring TeamXpress Area Labels

You can change the labels that appear in the TeamXpress GUI's branch view by editing the area label lines in `iw.cfg`. Use this option with caution, however, because the "branch," "staging area," "edition," and "workarea" terms are used throughout TeamXpress documentation, and changing these labels may cause confusion among users.

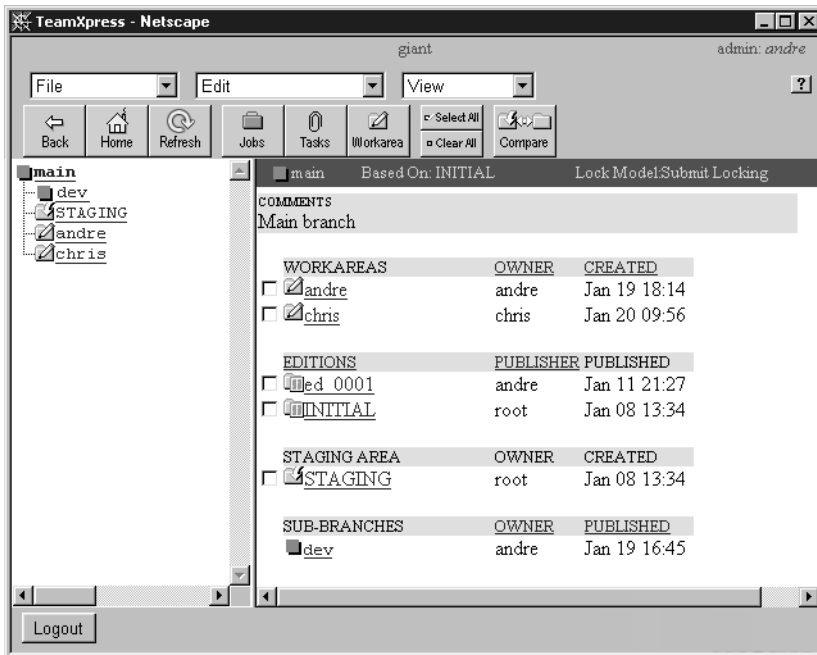
To change these labels, edit the following lines in the `[iwcgi]` section of `iw.cfg`. If these lines do not appear in `iw.cfg`, add them as shown below:

```
branch_label=new_branch_label
staging_label=new_staging_area_label
edition_label=new_edition_label
workarea_label=new_workarea_label
```

For example, with the default values of:

```
branch_label=SUB-BRANCHES
staging_label=STAGING AREA
edition_label=EDITIONS
workarea_label=WORKAREAS
```

The branch view looks like:

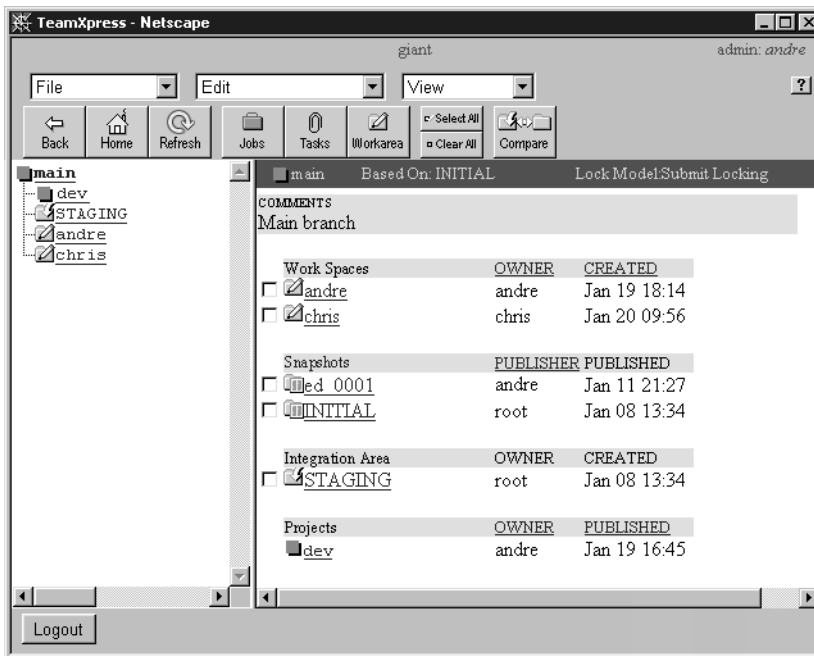


Default TeamXpress labels

However, giving these labels other values, such as:

```
branch_label=Projects
staging_label=Integration Area
edition_label=Snapshots
workarea_label=Work Spaces
```

Would change the labels in the branch view to:



*The TeamXpress Area Labels feature allows you to change the labels that appear in the Work Window in the branch view.*

## Configuring Edition Views

You can configure the number of editions you want to see in the branch view of the GUI. To view prior editions, click the **Show all editions** link in the GUI.

To set the number of editions to display, edit the edition list line in the `[iwcgi]` section of `iw.cfg`, as shown below:

```
edition_list_limit=number_of_editions
```

For example:

```
edition_list_limit=10
```

would configure TeamXpress to display only the ten most recent editions.

If this line does not appear in `iw.cfg`, add it as shown above. The default value is 5 (by default, TeamXpress will display the five most recent editions in the branch view).

To show all editions by default, comment out the `edition_list_limit` line by adding a `#` to the beginning of the line.

## Configuring History Views

You can configure the number of versions shown in the History view of the TeamXpress GUI. To configure this option, use the `view_history_limit` parameter in the `[iwcgi]` section of `iw.cfg`. For example:

```
view_history_limit=5
```

would restrict the History view to show only the five most recent versions of a file. All versions would still exist; however, only five would be displayed.

## User Profiles

When a user sets his Home page, that setting is stored in `iw-home/local/iwprofiles/username` on the TeamXpress server; that is, each user's setting is stored in its own file. Do not modify these files manually. These files are automatically generated by the TeamXpress server and updated as needed.

## Configuring GUI Functionality

### Disabling Editor Publish Capability

TeamXpress allows you to turn off Editors' ability to publish. You cannot turn this option off for selected Editors; it applies to all Editors on all branches.

To turn off the Publish capability for Editors:

If applicable, remove the comment mark (#) from the `editor_publish` line in the `[main]` section of `iw.cfg`. If `iw.cfg` does not contain this line, add it as shown below.

```
editor_publish=no
```

### Enabling and Disabling SmartContext Editing

You can selectively enable or disable SmartContext Editing for different workareas or files by adding lines to the `[iwproxy_smartcontextedit_allowed]` section of `iw.cfg`. If this section does not exist, SmartContext Editing is enabled by default.

The `[iwproxy_smartcontextedit_allowed]` section contains one `_default` line, which specifies whether SmartContext Editing is turned on or off in any area or for any file not otherwise specified. This section can also contain any number of `_regex` lines. Each `_regex` line uses a case-insensitive regular expression to specify areas or files, and then specifies whether SmartContext Editing is enabled or disabled for the specified items. A `_regex` line has the following case-insensitive syntax:

```
_regex=regular-expression=yes|no
```

`_regex` lines are order-dependent. For example, the following `[iwproxy_smartcontextedit_allowed]` section turns SmartContext Editing on by default, and it explicitly turns it on for all files in all of Andre's workareas on all branches. It then turns SmartContext Editing off for all CGI files. Because the line turning SmartContext Editing on for Andre's workareas comes first, he will be able to use SmartContext Editing for CGI files in his workarea:

```
[iwproxy_smartcontextedit_allowed]
_default=yes
_regex=(.*)/WORKAREA/andre/.*=yes
_regex=\.cgi(\?.*)?=$no
```

## Disabling LaunchPad Autoinstallation

By default, TeamXpress's helper application, LaunchPad, automatically installs itself when a user logs in through the TeamXpress GUI. This behavior can be turned off. To turn off LaunchPad installation:

1. Navigate to `iw-home/httpd/iw`
2. Move the file `loginlogo.html` to `loginlogo.html.old`.
3. Copy the file `loginlogo_noinstall.html` to `loginlogo.html`.

## Configuring LaunchPad Autoconfiguration

### Windows Platforms

LaunchPad can be pre-configured to use certain applications for certain file types for all Windows clients. For example, you can configure all Windows 95 clients to open HTML files using Notepad. (This requires that Notepad reside in the same location on all Windows 95 clients.)

These settings can be overridden on a user-by-user basis, by configuring the individual file types directly on the client systems. For more information, see the *TeamXpress User's Guide*.

To set up LaunchPad autoconfiguration, create a configuration file in `iw-home/local/config/launchpad.cfg`. This configuration file contains any number of sections which specify the different Windows platforms that you are autoconfiguring. Each section can contain any number of file mappings, and each file mapping has six possible options.

The possible sections for the different Windows platforms are as follows:

```
[win]          # all Windows platforms
[windows]     # Windows 95, 98, 2000(non-NT) platforms
[winnt]       # Windows NT platforms
```

In addition, each section name can be followed by a major or a major/minor version number. For example:

```
[winnt4]
```

refers to all Windows NT 4.x platforms.

```
[winnt4.4]
```

only refers to Windows NT 4.4 platforms.

Each file mapping is of the form:

```
extension=.extension
description=descriptive_text
command=command_setting
DDEMessage=message
DDEApplication=application
DDETopic=topic
```

where **extension** and **command** are required for each entry, and **description**, **DDEMessage**, **DDEApplication**, and **DDETopic** are optional.

Each section can contain multiple file mappings. The file mappings are read from top to bottom, so duplicate mappings later in this file will override earlier ones. Start with the more generic platform mappings and add more specific sections after that. e.g.:

```
[win]
extension=.html
description=HTML files edited with Netscape on Windows
command=c:\programs\netscape.exe
```

```
[winnt4]
extension=html
description=HTML files edited with Netscape on Windows NT 4.x
command=F:\PROGRA~2\Netscape\COMMUN~2\Program\Netscape.exe -edit "%1"
DDEMessage=[edit("%1")]
DDEApplication=NSShell
DDETopic=System
```

This will set the HTML editor for all Windows platforms to `c:\programs\netscape.exe`, except for Windows NT 4, where Netscape is installed in a different location. Note that both long and short-style paths can be used. Also note that if you use both the `.html` and the `.htm` extensions, you need to set those associations up separately:

```
[winnt4.4]
extension=html
description=HTML files edited with htedit on windows NT 4.4
command=C:\PROGRA~1\htedit "%1"
```

```
extension=htm
description=.htm files edited with htedit on windows NT 4.4
command=C:\PROGRA~1\htedit "%1"
```



## UNIX Platforms

To pre-configure LaunchPad associations on UNIX platforms, create an `.iwlanch.cfg` file as described in the *TeamXpress User's Guide*, and place it in the `/etc` directory. To override these settings on a user-by-user basis, create an `.iwlanch.cfg` file in the user's home directory.

## Macintosh

It is possible to pre-configure LaunchPad associations for the Macintosh by modifying the provided copy of Internet Config using ResEdit. However, this requires expertise with ResEdit. Do not attempt this if you are not very familiar with ResEdit.

## Configuring Preview Windows

When you click on the name of a file in the TeamXpress GUI, TeamXpress launches a browser window so you can preview it. By default, a new browser window is launched each time you click on a file name. However, you can configure TeamXpress to reuse the same window each time you preview a file.

To configure the number of browser windows TeamXpress launches, you must edit the `single_browser_window` line in the `[iwcgi]` section of `iw.cfg`. If this line does not exist, add it as shown below.

To reuse the same window each time you preview a file, set `single_browser_window=TRUE`. To launch a new browser window each time you preview a file, set `single_browser_window=FALSE` or comment the line out altogether. This setting will apply to all users on all branches of the TeamXpress server.

```
[iwcgi]
single_browser_window=TRUE
```

## Custom Menu Items

You can add custom menu items to the TeamXpress interface. These menu items can either call CGI scripts or HTML pages.

Note that TeamXpress includes all custom **File** menu items in the **File Options** drop-down list for each file listed in the Task Details screen. The `iw.cfg` file is checked for **File** custom menu items, which are added to the **File Options** drop-down list. Custom menu items for other menus (**Edit** and **View**) are not included in the Task Details screen

### CGI Scripts

CGI scripts that are added to the TeamXpress interface are executed via the TeamXpress CGI launcher/wrapper, which calls the CGI program and sets the environment variables that would be set by the Web server.

### Creating Custom CGI Scripts

The CGI wrapper makes certain variable=value pairs available, depending on the user's current location and any items that are selected.

The user's username and role are always available. Also available is the `vpath` of the current location, the mount path of the TeamXpress mount point, and the directory paths, object ids, and names of the current branch and archive, and (if applicable) of the current sub-branch, area, and directory. The `page_type` variable indicates what type of TeamXpress area the user is currently in, and the `subpage_type` variable (available when the user is not on a branch page), indicates whether the user is currently in a sub-directory or the root directory of his current area. Each item selected has four variables associated with it: type, object id, name, and path.

For example, user `andre` logged in as Master might navigate into the `htdocs` directory in his workarea on the `main/dev` branch, and select the checkboxes next to two directories. The following variable=value pairs would then be available:

<code>vpath=/default/main/dev/WORKAREA/andre/htdocs</code>	—	Vpath of current location
<code>mount_path=/iwmnt</code>	—	TeamXpress mount point
<code>directory_id=0x0000007b00000079000000b9</code>	}	Attributes of current directory
<code>directory_name=htdocs</code>		
<code>directory_path=/iwmnt/default/main/dev/ WORKAREA/andre/htdocs</code>		
<code>area_id=0x00002100000000000000007b</code>	}	Attributes of current area
<code>area_name=andre</code>		
<code>area_path=/iwmnt/default/main/dev/WORKAREA/andre</code>		
<code>branch_id=0x00002250000000000000006d</code>	}	Attributes of current branch
<code>branch_name=dev</code>		
<code>branch_path=/iwmnt/default/main/dev</code>		
<code>archive_id=0x000020200000000000000001</code>	}	Attributes of current archive
<code>archive_name=default</code>		
<code>archive_path=/iwmnt/default</code>		
<code>subpage_type=sub_directory</code>	—	Type of directory
<code>page_type=workarea</code>	—	Type of area
<code>session=AAAAAQAFYW5kcmUAAAAKMjAwMS5hbmRyZTWZjh4A</code>	—	For impersonation use
<code>page_id=8</code>	—	For internal use
<code>user_name=andre</code>	}	User's name and role
<code>user_role=master</code>		

type_0=directory	}	Type, object id, name, and directory path for the first item selected
objid_0=0x0000007b000000b9000000e9		
name_0=corporate		
path_0=/iwmnt/default/main/dev/WORKAREA/andre/htdocs/corporate		
type_1=directory	}	Type, object id, name, and directory path for the second item selected
objid_1=0x0000007b000000b9000000e5		
name_1=iwstore		
path_1=/iwmnt/default/main/dev/WORKAREA/andre/htdocs/iwstore		

To add a custom CGI script to the TeamXpress GUI:

1. Create a CGI program in `iw-home/httpd/iw-bin`.
2. Add the following line to the `[iwcgi]` section of `iw.cfg`:

```
custom_menu_item_identifier="MenuName", "MenuItemName",
"CGIProgramName", "RolesList", "WindowAttributes"
```

where each custom menu item has a unique identifier, *MenuName* is the menu to add the entry to, *MenuItemName* is the name of the menu item as it appears to the user, and *CGIProgramName* is the CGI program to execute. The name of the CGI program may not contain spaces. *RolesList* is the list of roles who will have access to this menu item, and *WindowAttributes* specifies the characteristics of the window. Unless otherwise specified, all window attributes are of the form `value=yes|no`. Possible *WindowAttributes* are:

toolbar	Specifies whether or not the browser toolbar will appear.
location	Specifies whether the Location input field (for entering URLs) will appear.
directories	Specifies whether directory buttons will appear.
status	Specifies whether the status line will appear.
scrollbars	Enables scrollbars.
resizable	Allows the user to resize the window.
menubar	Specifies whether the browser menu bar will appear.

width	Specifies the width of the window (in pixels).
height	Specifies the height of the window (in pixels).

For example:

```
custom_menu_item_show_env="File", "Environment", "show_env.cgi" "admin,
master" "width=640,height=450, scrollbars=yes,resizable=yes"
```

creates a new custom menu item called `show_env`. This menu item will appear in the **File** menu, and it will be called **Environment**. It will call the CGI program `show_env.cgi`, and the menu item will be available only to Administrators and Master users. The window that appears will be 640 pixels wide by 450 pixels high, it will have scrollbars, and it will be resizable.

*RolesList* and *WindowAttributes* are optional. If *RolesList* is not specified, all roles are assumed. If *WindowAttributes* is not specified, the defaults are:

```
resizable=yes,scrollbars=no,menubar=yes,width=640,height=480
```

If *WindowAttributes* is specified, then *RolesList* must be specified.

**Note:** The CGI program must be located in `iw-home/httpd/iw-bin`.

3. Log in and select the menu where you added the new item. You will see the new menu item at the bottom of the menu. When you select this item, a separate window will display the output of your CGI program.

## HTML Pages

You can also launch custom HTML files in a separate window from the TeamXpress GUI. The HTML file will be called directly from the Web browser, without any special preprocessing.

To add an HTML file to the TeamXpress GUI:

1. Create the HTML file.
2. Add the following line to the `[iwcgi]` section of `iw.cfg`:

```
custom_menu_item_identifier="MenuName", "MenuItemName", "file:URL",  
"RolesList", WindowAttributes"
```

where *identifier* is an identifier that is unique compared to the other items on this particular menu. *MenuName* is the menu to add the entry to; *MenuItemName* is the name of the menu item as it appears to the user; and *URL* is the URL of the HTML file to call. For example, an entry that calls the file `iw-home/httpd/iw/localhelp.html` might look like:

```
custom_menu_item_help="View", "Local help...",  
"file:/iw/localhelp.html"
```

## Configuring Submit Button Behavior

The **Submit** button can be configured to either submit files directly to the staging area (Submit-Direct), or to use the default Submit workflow process (Submit-Process). By default, the Submit button will use workflow for all roles.

To configure the Submit button behavior on a per-role basis, add the following section to `iw.cfg`:

```
[submit_button]  
submit_direct=roleslist  
submit=roleslist
```

where *roleslist* specifies the roles that use this option (editor, admin, master, or all—Authors must always use the Submit workflow process).

For example:

```
[submit_button]
submit_direct=admin, master
submit=editor
```

would configure the **Submit** button to submit files directly to the staging area for all Administrators and Master users, but to use the Submit workflow process for all Editors. Authors would still use the Submit workflow process.

If you disable the Submit button for any role (that is, if you include `submit=roleslist` in both the `[ui_remove_menu_items]` section and the `[ui_disable_directories]` section—see below), then this section will have no effect for that role, as there will be no Submit button to configure.

## Disabling Menu Items

You can now disable TeamXpress menu items and buttons on a per-role basis. To disable a TeamXpress menu item, add a new section to TeamXpress's main configuration file, `iw.cfg`, as follows:

```
[ui_remove_menu_items]
menuitemname="roleslist"
```

where `menuitemname` is the name of the menu item you want to disable, and `roleslist` is a comma-separated list of roles (e.g. `author, editor`). The menu item will be disabled for these roles. If the menu item has a corresponding button, it will be removed from the Button Bar for these roles (the **Compare Any** menu item and the **Compare with Staging** button are both governed by the `compare` value of `menuitemname`).

You can add multiple lines to a `[ui_remove_menu_items]` section. For example, the following `[ui_remove_menu_items]` section turns off the **Delete** and **Move** menu items for Editors and Authors.

```
[ui_remove_menu_items]
delete="editor,author"
move="editor,author"
```



This is a complete list of values of *menuitemname* for the [ui\_remove\_menu\_items] section. Items marked with an asterisk (\*) only apply if TeamXpress Templating is installed:

Value	Disabled Menu Item	Disabled Button
compare	<b>File &gt; Compare Any</b>	<b>Compare</b> (compares with the staging area)
compressedition	<b>Edit &gt; Compress</b>	N/A
copy	<b>File &gt; Copy</b>	N/A
copyto	<b>File &gt; Copy to Area</b>	N/A
delete	<b>File &gt; Delete</b>	N/A
*editdcr	<b>Edit &gt; Edit Data Record</b>	N/A
editfile	<b>Edit &gt; Edit File</b>	<b>Edit File</b>
editfilewith	<b>Edit &gt; Edit File With</b>	N/A
file_properties	<b>File &gt; File Properties</b>	N/A
*genHTML	<b>File &gt; Generate HTML</b>	N/A
getlatest	<b>File &gt; Get Latest</b>	<b>Get Latest</b>
history	<b>View &gt; History</b>	N/A
import_files	<b>File &gt; Import Files</b>	N/A
listlocks	<b>View &gt; List Locks</b>	N/A
listmodified	<b>View &gt; List Modified</b>	N/A
lock	<b>Edit &gt; Lock</b>	N/A
move	<b>File &gt; Move</b>	N/A
new_branch	<b>File &gt; New Branch</b>	N/A
*newdcr	<b>File &gt; New Data Record</b>	N/A
newdir	<b>File &gt; New Directory</b>	N/A
newfile	<b>File &gt; New File</b>	N/A
newJob	<b>File &gt; New Job</b>	N/A
new_workarea	<b>File &gt; New Workarea</b>	N/A



Value	Disabled Menu Item	Disabled Button
private	<b>Edit &gt; Private</b>	N/A
public	<b>Edit &gt; Public</b>	N/A
publish	<b>File &gt; Publish</b>	N/A
*regenHTML	<b>File &gt; Regenerate HTML</b>	N/A
rename	<b>File &gt; Rename</b>	N/A
setHomePage	<b>Edit &gt; Set Home Page</b>	N/A
submit	<b>File &gt; Submit</b>	<b>Submit</b> (configurable—see page 78)
submit_direct	<b>File &gt; Submit-Direct</b>	<b>Submit</b> (configurable—see page 78)
submitlog	<b>View &gt; Submit Log</b>	N/A
task_todo	<b>View &gt; To Do List</b>	<b>To Do</b>
uncompressedition	<b>Edit &gt; Uncompress</b>	N/A
unlock	<b>Edit &gt; Unlock</b>	N/A
updatelog	<b>View &gt; Update Log</b>	N/A
viewfile	<b>Edit &gt; View File</b>	<b>View File</b>

## Disabling Directory Operations

You can now disable certain operation abilities to act on directories, on a per-role basis. To disable a TeamXpress menu item's abilities to act on directories, add a new section to TeamXpress's main configuration file, `iw.cfg`, as follows:

```
[ui_disable_directories]
menuitemname="roleslist"
```

where `menuitemname` is the name of the menu item you want to disable for directories, and `roleslist` is a comma-separated list of roles (e.g., `author, editor`). Specify `all` to disable the menu item for all roles. The menu item will no longer act on directories when it is invoked by a user who is logged in with one of these roles.

You can only disable menu item abilities to act on directories if they would ordinarily be able to do so.

You can add multiple lines to a `[ui_disable_directories]` section. For example, the following `[ui_disable_directories]` section disables Editor and Author abilities to delete or move directories.

```
[ui_disable_directories]
delete="editor,author"
move="editor,author"
```

This is a complete list of values of *menuitemname* for the `[ui_disable_directories]` section:

Value	Disabled Menu Item	Disabled Button
copy	<b>File &gt; Copy</b>	N/A
copyto	<b>File &gt; Copy to Area</b>	N/A
delete	<b>File &gt; Delete</b>	N/A
getlatest	<b>File &gt; Get Latest</b>	<b>Get Latest</b>
move	<b>File &gt; Move</b>	N/A
private	<b>Edit &gt; Private</b>	N/A
public	<b>Edit &gt; Public</b>	N/A
rename	<b>File &gt; Rename</b>	N/A
submit	<b>File &gt; Submit</b>	<b>Submit</b> (configurable—see page 78)
submit_direct	<b>File &gt; Submit-Direct</b>	<b>Submit</b> (configurable—see page 78)

## Disabling Unlocked File Auto-Upload

By default, TeamXpress allows you to upload files even if it cannot establish a lock on them. To disable this feature so that files are uploaded only if TeamXpress can establish a lock, add the following line to the `[iwproxy]` section of `iw.cfg`:

```
allow_unlocked_file_upload=no
```

To turn unlocked file uploading back on, either remove this line from `iw.cfg` or change `no` to `yes`.

## Setting the Number of Jobs Listed in the To Do List

The `iw.cfg` configuration file allows you to set the maximum number of jobs to be listed in the To Do List. To configure this option, add the following line to the `[iw_workflow_ui]` section of `iw.cfg`. If this section does not exist, create it:

```
[iw_workflow_ui]
max_job_count_per_page=number_of_jobs|all
```

The default is 55 jobs per page. You can change the number of jobs by specifying a value. If you specify `all`, all the jobs fulfilling a particular view display on a single page. In Windows 95/98, if you have over 100 jobs and specify `all`, the workflow screen may be exceedingly slow.

## Configuring Job Attribute Filters and Settings

Job attributes are settable properties of individual jobs. These attributes show up in the workflow section of the TeamXpress GUI. You can set these attributes through the GUI, and you can choose to display only the jobs that have certain attribute settings. The names of these attributes and their possible settings are configured in the `[iw_workflow_ui]` section of `iw.cfg`.

For example, if you have attributes called `Category`, `Month`, and `Date`, you can use drop-down menus in the GUI to view only jobs with specific values of `Category`, `Month`, and `Date` (e.g. `Category=Marketing, Month=November, Date=24`).

You can configure up to three attributes. Each attribute can have any number of possible values. Values are separated by colons.

The `[iw_workflow_ui]` section of `iw.cfg` has the following format:

```
[iw_workflow_ui]
attribute1=attributename1
values1=valuelist1
attribute2=attributename2
values2=valuelist2
attribute3=attributename3
values3=valuelist2
```

where *attributename1*, *attributename2*, and *attributename3* specify the names of attributes 1, 2, and 3, respectively. *valuelist1*, *valuelist2*, and *valuelist3* are of the format:

```
value1:value2:value3:...valuen
```

To use fewer than three attributes, omit the appropriate attribute and value lines. Attributes must always be numbered sequentially, starting at 1.

For example:

```
[iw_workflow_ui]
attribute1=Category
values1=Sales:Marketing:Engineering:Professional Services:Administration
attribute2=Month Due
values2=Jan:Feb:Mar:Apr:May:Jun:Jul:Aug:Sep:Oct:Nov:Dec
attribute3=Date Due
values3=1:2:3:4:5:6:7:8:9:10:11:12:13:14:15:16:17:18:19:20:21:22:23:24:25
:26:27:28:29:30:31
```

would create three attributes named `Category`, `Month Due`, and `Date Due`. `Category` has possible values of Sales, Marketing, Engineering, Professional Services, and Administration. `Month Due` has possible values of Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, and Dec. `Date Due` can be any number from 1 to 31.

Note that due to page constraints some lines may appear to wrap. Actual lines in `iw.cfg` should never wrap.

## Configuring Server Functionality

### User Authentication

TeamXpress can be configured to use authentication via LDAP, external file, (default) UNIX user passwords, or (on Solaris 2.6 or later) pluggable authentication modules (PAM).

To configure the type of authentication to use, add the following lines to `iw.cfg`:

```
[authentication]
authenticate_by=mode
```

where *mode* specifies one or more of `file`, `ldap`, `local`, or `pam`. Entries may be separated by commas or spaces.

Precedence can be specified, e.g., if you specify `authenticate_by = file ldap`, then `file` will be checked first. If the check fails, `ldap` will be checked next.

If `ldap` is specified as one of the possible modes, you will also need to add the following lines to the `[authentication]` section:

```
ldap_server=ldap-server
ldap_port=ldap-port
ldap_dnbase=search-base-location
```

where

*ldap\_server* is the host machine of the LDAP server

*ldap\_port* is the port for the LDAP server

*ldap\_dnbase* is the specification of DN base location (according to LDAP search syntax) e.g.,  
`ldap_dnbase=ou=people,o=example.com`

If `file` is specified as one of the possible modes, you will also need to add the following line to the `[authentication]` section:

```
password_file=absolute-path-to-file
```

where

`absolute-path-to-file` is the absolute path to a file containing encrypted user-passwords of the same format found in `etc/shadow`.

If `pam` is specified as one of the possible modes, you can perform additional PAM-specific configuration activities. Details are as follows.

### PAM and Account Management

TeamXpress will normally ask a pluggable authentication module (PAM) to perform account management functions on the authenticated user. This is typically used to control expired passwords, login time restrictions, etc. To configure TeamXpress not to perform account management functions, add the following line to the `[authentication]` section of `iw.cfg`:

```
pam_do_acct_mgmt=no
```

### TeamXpress and PAM Configuration File Interaction

By default, PAM will control authentication by using entries tagged with the TeamXpress service name in `/etc/pam.conf`. You can specify that PAM use `/etc/pam.conf` entries tagged other than TeamXpress by changing the `pam_service` line in the `[authentication]` section of `iw.cfg`. For example, to specify that TeamXpress instead use the lines in `/etc/pam.conf` that also control the telnet program, edit the `pam_service` line in `iw.cfg` so that it reads as follows:

```
pam_service=telnet
```

The format of `/etc/pam.conf` is described in detail in the `pam.conf(4)` man page. You should configure TeamXpress with its own entries in `/etc/pam.conf` using the service name TeamXpress (or whatever service name you specify for `pam_service` in `iw.cfg`). Only the `auth` and `account` modules in `/etc/pam.conf` are used for TeamXpress authentication. If no entries are present for TeamXpress in `/etc/pam.conf`, PAM will use whatever settings are specified for the other service. Note that this scenario is not recommended.

On Solaris 2.7 or later, the following lines in `/etc/pam.conf` will produce behavior equivalent to the traditional TeamXpress authentication method:

```
TeamXpressauth    required    /usr/lib/security/pam_unix.so.1
TeamXpressaccount required    /usr/lib/security/pam_unix.so.1
```

On Solaris 2.6 systems, you must add the `use_first_pass` flag as follows:

```
TeamXpress auth    required /usr/lib/security/pam_unix.so.1
use_first_pass
TeamXpress account required /usr/lib/security/pam_unix.so.1
```

To use a third-party PAM module, specify its path instead of `/usr/lib/security/pam_unix.so.1`. For more information about PAM, see <http://www.sun.com/software/solaris/pam/>.

## Autocompression

TeamXpress can be configured to automatically compress editions. TeamXpress Administrators can also manually compress and uncompress editions (see the *TeamXpress User's Guide*). Users can still easily access the files contained in compressed editions, as TeamXpress will automatically uncompress files as needed.

TeamXpress cannot compress certain file formats, such as JPEGs and GIFs, because these files are already compressed. To turn on autocompression, edit the `autocompress` line in the `[iwserver]` section of `iw.cfg` to read `autocompress=yes`. If a comment symbol (`#`) is present at the beginning of this line, remove it. To add file types to the list of file types to remain uncompressed, add a `no_compress_suffix` line to the `[iwserver]` section of `iw.cfg`. `iw.cfg` can include multiple `no_compress_suffix` lines, and each line is of the format:

```
no_compress_suffix=.suffix
```

where *.suffix* specifies the file type to exclude from compression. For example:

```
[iwservr]
autocompress=yes
no_compress_suffix=.compress
no_compress_suffix=.jpg
no_compress_suffix=.jpeg
no_compress_suffix=.jpe
```

## Webserver UID

The Web server uid setting should be set to any uid that allows the Web server to see the Web content as an outside viewer would see it, in order for users to be able to preview the Web site that a normal user would see. Because external browsers access the Web server as `nobody`, this is used as the default. To change the Web server uid setting, edit the `webserver_uid` line in the `[iwservr]` section of `iw.cfg`. If `iw.cfg` does not contain this line, add it as shown below:

```
webserver_uid=nobody
```

## Main Branch Settings

The following settings apply only to the main branch in TeamXpress, not to any of its sub-branches. Because the main branch is not ordinarily used for development, these settings may not apply to your TeamXpress configuration. However, if you have a special need to change the locking model, owner, or group of the main branch, you can use the following settings. In TeamXpress, you cannot create workareas off of the main branch.

### Locking Model

TeamXpress allows you to specify the locking model of each branch at the time that it is created. However, the main branch is created automatically when TeamXpress is installed, or when a new backing store is created. You can specify which locking model to use for the main branch of a new backing store by editing the `main_lock_model` line in the `[iwservr]` section of `iw.cfg` (for a detailed explanation of locking models, refer to the *TeamXpress User's Guide*). When TeamXpress is first installed, it uses the default option of Submit locking for the main branch. The type of locking a branch uses cannot be changed after the branch has been created. However, if you edit the `main_lock_model` line and then create a new backing store, the new settings will take effect on the new backing store. For information about creating a new backing store, see page 244.



```
main_lock_model=locking_model
```

where *locking\_model* is one of `submit_lock`, `optional_write_lock`, or `mandatory_write_lock` (or, more simply, `s`, `o`, or `m`). Submit locking is the default option for the main branch. Optional and mandatory write locking may significantly reduce system performance.

### Owner and Group

You can specify the owner and group of the main branch of a new backing store by editing the `main_owner` and `main_group` lines in the `[iwserver]` section of `iw.cfg`. When TeamXpress is first installed, it uses the default option of `root` for main branch ownership. To change this setting on an existing main branch, you must use the `chown` and `chgrp` commands to change the ownership of the root directory of the main branch. However, if you edit the `main_owner` and `main_group` lines and then create a new backing store, the new settings will take effect on the new backing store. For information about creating a new backing store, see page 244.

```
main_owner=root
main_group=root
```

### Locked File Submission

You can configure TeamXpress to allow only the owner or creator of the lock to submit a locked file to the staging area (as opposed to allowing any member of the workarea where the file is locked). To configure this option, add the following line to the `[iwserver]` section of `iw.cfg`:

```
only_lock_owner_creator_submits=yes
```

### Submit and Update Logs

You can configure the number of events that are contained in the Submit and Update logs for a workarea. To change this number, remove the comment (`#`) symbol from the `event_log_size` line in the `[iwserver]` section of `iw.cfg` and edit the line to specify the number of events you want to record. If this line does not appear in `iw.cfg`, add it as shown below. For example, with this setting, the Submit and Update logs will contain the 64 most recent Submit or Get Latest operations (as opposed to the 64 most recent files that were submitted or updated).

```
event_log_size=64
```

You can also configure whether or not you want all the files contained in new or deleted directories to be listed individually in the Submit and Update logs. To configure this option, remove the comment (#) marks from the `full_submitlog` and `full_updatelog` lines in the `[iwserver]` section of `iw.cfg` and edit the lines to specify `yes` to show all the files that were contained within the directory

that was added or deleted or `no` to show only the directory names. If these lines do not appear in `iw.cfg`, add them as shown below.

```
full_submitlog=no
full_updatelog=no
```

## Branch and Workarea Security

Branch and workarea security determines whether or not a user can see the names of branches and workareas he does not have access to. If a user does not have read access to a branch or workarea, and branch and workarea security is turned off, he will be able to see the name of the branch or workarea, but it will not be linked, and `[N/A]` will appear next to it. However, you can configure TeamXpress to not even show the names of branches and workareas in the TeamXpress GUI if the user does not have read permissions. To set this option, remove the comment (#) marks from the `branch_security` and `workarea_security` lines in the `[iwserver]` section of `iw.cfg` and edit the lines to specify `off` to show all branch and workarea names or `on` to show only the branch and workarea names for which the user has read access.

If these lines do not appear in `iw.cfg`, add them as shown below.

```
branch_security=on
workarea_security=on
```

## Default Permissions

You can configure the default permissions for branches, workareas, directories, and files created using TeamXpress's GUI. Permissions on files created through the file system interface are determined by your file system interface configuration (e.g., the Samba configuration).

To set the permission bits automatically, edit the `branch_default_perm`, `workarea_default_perm`, `directory_default_perm`, and `file_default_perm` lines in the `[iwserver]` section of `iw.cfg` to specify the octal values of the default permission bits for newly created branches, workareas, directories, and files. These settings will only apply to branches,

workareas, directories, and files created after you have edited these lines. If these lines do not appear in `iw.cfg`, add them as shown below.

```
branch_default_perm=permissions
workarea_default_perm=permissions
file_default_perm=permissions
directory_default_perm=permissions
```

where `permissions` specifies the permissions in octal. For example:

```
branch_default_perm=775
```

## Group Remapping

This option provides a workaround for a limitation of NFS. When NFS checks the group that can access a file against the groups that a user belongs to, it only checks the first sixteen groups that the user belongs to. Therefore, if the group on the file is the seventeenth (or more) group that the user belongs to, NFS will incorrectly deny the user access to the file (this applies only to operations performed through the file system).

The `map_secondary_to_primary` option in `iw.cfg` works around this problem. Because TeamXpress does not have NFS's sixteen-group limitation, it first determines whether a user should have group-level access to the file. Then, if the `map_secondary_to_primary` option is turned on, it maps the file's group to the user's primary group. Therefore, if you check the gid on a file via the file system, it could return a gid different from the true gid of the file. To find the file's true gid, use the **File > File Properties** menu item in the TeamXpress GUI, or the `iwattrib` command-line tool.

To turn group remapping on, add the following line to the `[iwserver]` section of `iw.cfg`:

```
map_secondary_to_primary_gid=yes
```

## File Locations

The `[locations]` section of `iw.cfg` may be used to change the locations of various TeamXpress files and directories. To change the location of one of the following files or directories, remove the comment (`#`) marks from its line and edit the line to point to the new location (make sure that the `[locations]` line is not also commented out). When you restart TeamXpress, it will look for the specified file or directory in the new location.

If you change the location of `iwmount`, you will need to edit its Web server alias to point to the new location.

If you change the location of one of the logs, and no file of the specified name is present in the new location, a new file will be created.

If `iw.cfg` does not contain these lines, add the ones you want to configure as shown below. Due to space limitations, some of these lines may appear to wrap. The actual lines in `iw.cfg` should not wrap:

```
[locations]
iwbin=/usr/iw-home/bin
iwmount=/iwmnt
iwcgimount=/.iwmnt
iwroles=/usr/iw-home/local/config/roles
iwstore=/local/iw-store
iwsubmitconfig=/usr/iw-home/local/config/submit.cfg
iwautoprivate=/usr/iw-home/local/config/autopprivate.cfg
iwlogs=/usr/iw-home/local/logs
iwconfigs=/usr/iw-home/local/config
iweventlog=/var/adm/iwevents.log
iwtracelog=/var/adm/iwtrace.log
iwserverlog=/var/adm/iwserver.log
iwdeploylog=/usr/iw-home/local/logs/iwdeploy.log
launchpad=iw-home/local/config/launchpad.cfg
```

where:

<code>iwbin</code>	Specifies the location of TeamXpress binaries (normally <code>iw-home/bin</code> ).
<code>iwmount</code>	Specifies the location of the TeamXpress mount point.
<code>iwcgimount</code>	Specifies the location of the TeamXpress mount point.
<code>iwroles</code>	Specifies the directory containing the TeamXpress roles files.
<code>iwstore</code>	Specifies the location of the TeamXpress backing store.
<code>iwsubmitconfig</code>	Specifies the location of the Submit Filtering configuration file.
<code>iwautopriate</code>	Specifies the location of the Autoprivate configuration file.
<code>iwlogs</code>	Specifies the directory containing TeamXpress logs.
<code>iwconfigs</code>	Specifies the default configuration file directory.
<code>iweventlog</code>	Specifies the location of the TeamXpress event log.
<code>iwtracelog</code>	Specifies the location of the TeamXpress trace log.
<code>iwserverlog</code>	Specifies the location of the TeamXpress server log.
<code>iwdeploylog</code>	Specifies the location of the deployment log.
<code>launchpad</code>	Specifies the location of the LaunchPad autoconfiguration file (default is <code>iw-home/local/config/launchpad.cfg</code> )

## Autoprivate

TeamXpress's Autoprivate feature allows you to prevent certain file types, such as temporary files and Macintosh resource forks, from being submitted to the staging area or copied during a **Copy To** operation. Changes to Autoprivate only apply to files that are created or renamed after the changes are made. Changes do not apply to existing files.

To turn on Autoprivate, create a file named `autopriate.cfg` in your `iw-home/local/config/` directory. The Autoprivate file consists of two sections: files matched by pattern, and files matched by name. Each section is set off by parentheses on their own lines, and the file begins with a `(` (open parenthesis) on its own line and ends with a `)` (close parenthesis) on its own line.

Individual entries in the first section are in the following format:

```
((filenamepattern)(#_characters_to_match_at_beginning)
(#_characters_to_match_at_end))
```

where both *#\_characters\_to\_match\_at\_beginning* and *#\_characters\_to\_match\_at\_end* are in hexadecimal.

For example, to have Autoprivate detect any filename that ends with `.frk`, add the following entry:

```
((x.frk)(0)(4))
```

meaning to match zero characters at the beginning of the filename and the four characters (`.frk`) specified at the end of the filename.

To set Autoprivate to detect any filename that ends in `.backup.fm`, add the following entry:

```
((x.backup.fm)(0)(a))
```

where `0` specifies not to match any characters at the beginning, and `a` (hexidecimal 10) specifies to match ten characters at the end of the filename.

To set Autoprivate to detect any filename that both begins and ends with a number, add the following entry:

```
((#x#)(1)(1))
```

meaning to match one character at the beginning of the filename and one character at the end of the filename. The `#` symbols in the filename pattern indicate that the character must be numeric.

Entries in the second section specify exact filename matches, set off by double parentheses. These filename matches apply across all directories in all workareas on the TeamXpress server. For example, if `autoprivate.cfg` includes:

```
((test))
```

then all files named `test` that are created after this line is added, in all directories in all workareas in TeamXpress, will be marked private.

To have Autoprivate detect a filename that includes spaces, encode the spaces with a `\20`, e.g.:

```
((Network\20Trash\20Folder))
```

to match “Network Trash Folder”

This sample `autoprivat.e.cfg` file includes a few common entries:

```
(  
(  
((x.o)(0)(2))  
((x.a)(0)(2))  
((#x#)(1)(1))  
((x~)(0)(1))  
((.nfsXXX)(4)(0))  
((x.bak)(0)(4))  
((x.tmp)(0)(4))  
)  
(  
((network\20trash\20folder))  
((Network\20Trash\20Folder))  
((.HSAncillary))  
((.HSResource))  
((.hsancillary))  
((.hsresource))  
((.tnatr:intf))  
((.tnatr:reso-fork))  
((resource.frk))  
((trash))  
)  
)
```

For changes to `autoprivat.e.cfg` to take effect, restart the TeamXpress server or use the `iwreset` command-line tool.

## Template Files

If you are using templates with the TeamXpress New File feature, you can configure which templates are to be used in various parts of the Web site. These settings are controlled through the templating configuration file, `iw-home/local/config/iwtemplates.cfg`.

This file governs which templates are accessible from which directories and branches. To configure which directories a template can be used in, add a line to this file. Only non-TeamXpress Templating access is configured in this file (e.g., HTML templates or Microsoft Word templates). To configure TeamXpress Templating, consult the *TeamXpress Templating and Deployment Guide*.

### Syntax

`iwtemplates.cfg` uses the following format:

```
templates
{
  branch
  {
    template_type
    {
      template_identifier
      {
        template=regular-expression
      }
    }
  }
}
```

where *branch* specifies the vpath to a branch, *template\_type* identifies the type of template to be configured in that section, and *template\_identifier* is the individual template identifier that will appear in the New File GUI. The *template* line specifies the full path of each template (rooted in a workarea) and uses case-insensitive regular expression matching to determine which directories this template may be accessed from.

You can have any number of *branch* sections, each of which can have any number of *template\_type* sections. The *template\_type* sections can have any number of *template\_identifier* sections, each of which contains one *template* line.



Each *template* line has the following format:

```
template=regular-expression
```

or

```
template={regular-expression1, regular-expression2,  
regular-expression3...}
```

The left-hand side of each line specifies the template, and the right-hand side contains the case-insensitive regular expression that determines where this template may be used.

Here are some common examples:

To permit the template `global.html` to be used across the entire Web site on a branch, you would add this *template* line to a branch section in `iwtemplates.cfg`:

```
/templates_dir/global.html='/'
```

To permit the template `subdir.html` to be used in any directory path that contains a subdirectory named `subdir`, i.e., `/htdocs/subdir/company` or `/htdocs/products/subdir`, etc, you would add this line:

```
/templates_dir/subdir.html='subdir'
```

To permit the template `company.html` to be used in `/htdocs/company` and all of its subdirectories, you would add this line:

```
/templates_dir/company.html='^/htdocs/company/'
```

To permit the template `products.html` to be used in the `/htdocs/products/` directory but not its subdirectories, you would add this line:

```
/templates_dir/products.html='^/htdocs/products/$'
```

You can also specify multiple patterns to be matched. If you specify multiple patterns, enclose the right-hand side of the line in curly brackets and separate the individual patterns by commas. For example:

```
/templates_dir/products.html={'^/htdocs/products/', 'subdir'}
```

would permit the `products.html` template to be used in `/htdocs/company` and all of its subdirectories, and in any directory path that contains a subdirectory named `subdir`.

### Launching Files Through `iwproxy`

This option enables in-context QA and consistent views of TeamXpress workareas. By default, this option is turned on. However, if your Web site must support SSL, you will need to turn this option off and install the TeamXpress redirector module.

To turn this option off:

1. If a comment symbol (`#`) is present at the beginning of the `use_iwproxy` line of `iw.cfg`, remove it. If `iw.cfg` does not contain this line, add it as shown below.
2. Edit the line to read:

```
use_iwproxy=no
```

## Configuring Server Performance

### Cache Size

To set TeamXpress's cache size, edit the `cachesize` line in the `[iwserver]` section of `iw.cfg`. If a comment symbol (`#`) is present at the beginning of this line, remove it. If this line does not appear in your `iw.cfg` file, add it as shown below. The initial cache size setting should be approximately three times the number of files and directories on the largest branch. For example, if the largest branch contains 15,000 files and directories, you should set cache size to 45000 as shown below. Minimum cache size is 1000; maximum is 500000 (five hundred thousand). Each cache line takes a maximum of 1KB of physical memory. Recommended physical memory is cache size times 1KB plus an additional 25% as a safety margin. In the example shown below, physical memory would be  $(45,000 * 1KB) + 11MB = 56MB$ . If you encounter a great deal of memory swapping, you should either reduce the cache size or install more memory.

You must restart the TeamXpress server for these changes to take effect.

```
cachesize=45000
```

### RPC Threadcount

The RPC threadcount setting determines how many simultaneous requests TeamXpress can handle from users via the GUI or command-line tools. These requests are very short-lived, so that threads are quickly freed for other users. If all threads are currently being used, TeamXpress starts to serialize requests. This setting should not be altered.

```
rpc_threadcount=64
```

## File System Threadcount

The file system threadcount should be set to approximately the number of CPUs on the TeamXpress server. This setting should not be set higher than 2. To change the file system threadcount, edit the `fs_threadcount` line in the `[iwserver]` section of `iw.cfg`. If a comment symbol (`#`) is present at the beginning of this line, remove it. If this line does not appear in your `iw.cfg` file, add it as shown below.

You must restart the TeamXpress server for this change to take effect.

```
fs_threadcount=1
```

## Filesystem Active Area Cache

The file system active area cache should be set to approximately the number of users who are expected to be using TeamXpress concurrently. Note: this is the number of users who are using TeamXpress at one time, not the total number of TeamXpress users. If this value is too large, it will significantly impact memory usage.

To set the file system active area cache, edit the `fs_active_area_cache` line in the `[iwserver]` section of `iw.cfg`. If a comment symbol (`#`) is present at the beginning of this line, remove it. If this line does not appear in your `iw.cfg` file, add it as follows:

```
fs_active_area_cache=8
```

You must restart the TeamXpress server for this change to take effect.

## Throughput Monitors

Throughput monitors can be used in conjunction with the `iwstat` command-line tool to monitor system status and performance. To turn on throughput monitors, remove the comment marks (`#`) from the beginning of the lines for the throughput monitors you want to use in the `[iwserver]` section of `iw.cfg`. By default, there are throughput monitors that return system statistics over the previous minute, fifteen minutes, hour, 8 hours, 24 hours, and for the entire time that the system has been running. There are also two throughput monitors that you can configure with any time interval.

```
thruputmonitoring=on
thruputmonitor1=1      # 1 minute
thruputmonitor2=15    # 15 minutes
thruputmonitor3=60    # 1 hour
thruputmonitor4=480   # 8 hours
thruputmonitor5=1440  # 24 hours
thruputmonitor6=-1    # forever
thruputmonitor7
thruputmonitor8
```

## Detecting Low Disk Space and Inode Count

TeamXpress is configured to freeze the backing store when it detects that free disk space or inode count is low. The backing store remains frozen until sufficient disk space or inode count is restored, at which point the server returns to its normal running state. This feature helps prevent possible corruption of the backing store. While the backing store is frozen, users cannot write to the TeamXpress backing store. Users can still perform read-only operations. The CLT `iwfreeze` can be used to manually freeze the backing store.

The lines shown below in the `[iwserver]` section of `iw.cfg` control the behavior of disk /inode low detection.

The `disklow_mbytes` line gives the server a freeze threshold in MB (the default is 50). The `disklowpercent` line sets the percent of free disk space that is considered “low” (the default is 10). The TeamXpress server does not allow `disklowpercent` to go below 2%. If the server detects a low-disk state based on the threshold set in `iw.cfg`, it does not allow you to manually unfreeze the backing store via the `iwfreeze` command. The `disklow_knodes` line gives the server a freeze threshold in thousands of inodes (the default is 50). To change these settings, edit these lines as shown below.

```
disklow_mbytes=20
disklowpercent=15
disklow_knodes=25
```

## Changing File Attributes During Submit

The TeamXpress server allows you to automatically change file attributes, such as uid, gid, and permissions, at the time that you submit a file. This option allows you to automate the task of specifying the permissions that each file will have in the deployed Web site. The submit filter performs the specified operation on files immediately before they are submitted, so that changes are made to the files in the workarea, which are then submitted.

On startup, the TeamXpress server reads a configuration file named `submit.cfg` in the `iw-home/local/config/` directory (unless the location of this file is otherwise specified in the `[locations]` section of `iw.cfg`). The `submit.cfg` file contains rules to match file and workarea patterns to specific actions to perform when files and directories are submitted.

It has the following format:

```

case-sensitive = [yes|no]
rules
{
    workarea1_pattern
    {
        file_pattern1 { action1 action2 ... }
        file_pattern2 { action3 action4 ... }
        ...
    }
    workarea2_pattern
    {
        file_pattern3 { action5 action6 ... }
        file_pattern4 { action7 action8 ... }
        ...
    }
    ...
}

```

The case-sensitive statement specifies whether or not the rules matching should ignore the case of the path names. If case-sensitive is not specified, the value is assumed to be `yes`, so that rules matching does not ignore the case of the path names.

*workarea pattern* is used to match a workarea to the set of file rules to apply when a submit is done from the workarea. Each pattern can only be specified once, and the first match is used. The syntax of the pattern is `regex(5)` (extended syntax). For more information on regular expressions, consult a reference manual such as *Mastering Regular Expressions*, by Jeffrey Friedl, or read the man page:

```
% man -s 5 regex
```

The match is done against the path name of the workarea, starting with `/default/main`.

*file pattern* is used to match a file or directory to the set of actions to perform on it when it is submitted. Each file or directory pattern can only be specified once, and the first match is used. The syntax of the pattern is `regex(5)` (extended syntax).

The match is done against the path name of the file or directory relative to the workarea.

*action* is one of

```
uid=name
gid=name
perm=octal number
amask=octal number
omask=octal number
expand_rcs_macros = [yes|no] (see page 107)
```

and specifies the operation to perform on the file or directory being submitted. *uid=*, *gid=*, and *perm=* specify new values for these attributes. *amask=* specifies a bit mask to “and” to the existing mode of the file or directory. *omask=* specifies a bit mask to “or” with the existing mode of the file or directory. When you submit files or directories:

1. The server determines what files and directories have actually changed and need to be submitted. It also verifies that none of them are in conflict with the staging area or locked in other workareas.
2. The path name of the workarea from which the submit is being done is matched against the workarea patterns from the configuration file.
3. If the workarea matches one of the workarea patterns, then, for each file and directory that needs to be submitted (as determined in step 1), the file's path name is matched against the file patterns in the matching workarea's section.
4. If a match is found, then the server performs the specified set of actions to the file or directory in the workarea.
5. The server submits the transformed files and directories to the staging area.



**Example**

This is a sample `submit.cfg` file:

```
CASE-SENSITIVE = YES
RULES
{
# SECTION 1
  ^/default/main/dev/WORKAREA/.*$ # any workarea in the main branch
  {
    ^/index\.html$ { gid=test perm=0664 uid=andre }
    # attributes fixed for /index.html
    .*\.sh$ { omask=0111 } # execute bits on all .sh files
    /$ { uid=andre } # andre owns all the directories
    .* { amask=0775 } # don't allow world write access
  }
# SECTION 2
  ^/default/main/dev/TeamXpress/WORKAREA/chris$ # just for chris
  {
    ^/html/chris/.*$ { uid=chris gid=iw } # under /html/chris/
    .* { amask=0775 } # don't allow world write access
  }
# SECTION 3
  .* # any other workarea on any other branch
  {
    .* { amask=0775 gid=test } # no world write access
  }
}
```

**Notes**

Only the first match is applied. That is, the first match wins if multiple rules match the file or directory. `submit.cfg` should be ordered so that the most specific workarea patterns are closer to the top and the most specific file patterns are earlier in each section. You may need to duplicate some actions for them to apply to multiple rules.

For purposes of matching, the path name of a directory must end in a slash (“/”).

Single or double quotes around patterns are optional, but they must be used around workarea and file patterns that contain white space or other special characters, like #, {, }, =, or . . Backslashes (\) are special characters when used within patterns surrounded by quotes; a backslash followed by any character is replaced by the single character. For example, to embed a single quote, double quote, or backslash in a pattern, precede the character with a backslash (\', \", or \\). Backslashes are not special characters in patterns that are not quoted.

Do not specify duplicate workarea patterns multiple times, duplicate file patterns multiple times within a workarea section, or the same action multiple times within a file rule.

Because of client-side attribute caching, the modes, uid, and gid may not appear differently in the workarea immediately after a submit. The correct attributes will appear after a sufficient time-out interval (usually about 30 seconds).

You cannot change the permissions for a symbolic link. The `perm`, `amask`, and `omask` actions will not be performed on a submitted symbolic link. The `uid` and `gid` actions will be performed on a submitted link, not on the actual file.

The permission values should be specified as an octal number (starting with a 0), between 0 and 0777.

Changes to `submit.cfg` do not take effect until the server is restarted or until you use `iwreset`.

## Debugging

The CLT `iwtestcfg` can be used to find out which workarea and file pattern will be applied to a file at the time of submission:

```
% iwtestcfg /default/main/dev/WORKAREA/andre/cgi/test.sh
```

Would return:

```
Matched area pattern "^/default/main/dev/WORKAREA/.*$"  
Matched file pattern ".*\.sh$"  
Actions to do are:  
omask=0111
```

Matched area pattern and Matched file pattern are the case-insensitive regular expressions found in `submit.cfg` that match the workarea and file. Actions to do are the actions (specified in `submit.cfg`) that will be applied to the file.

You can also get debugging information on the submit handling configuration printed to `iwtrace.log`, by adding the following line to the `[server]` section of `iw.cfg`:

```
debug_event_handler=yes
```

This will cause the server to print a configuration map of `submit.cfg` and to print which actions are performed as files are submitted.

## RCS Macro Expansion

TeamXpress provides RCS-style macro substitution at submit time. These macros give you a way to embed version information into files, such as the file name, revision string, last modifier, when it was modified, and submit comments.

To use the macro facility, you must first add new rules to the `submit.cfg` configuration file, then manually insert the macros into files in your workarea. When the files are submitted, the TeamXpress server replaces the macros with the appropriate information, leaving you with a rewritten file in your workarea and the staging area.

### Macros

The macros are a subset of those used in RCS (called “keywords” in RCS documentation). The following description is taken from the UNIX `co(1)` man page, modified for TeamXpress semantics.

Strings of the form `$keyword$` and `$keyword:...$` embedded in the text are replaced with strings of the form `$keyword:value$` where `keyword` and `value` are pairs listed below. Keywords can be embedded in literal strings or comments to identify a revision.

Initially, the user enters strings of the form `$keyword$`. On submit, the server replaces these strings with strings of the form `$keyword:value$`. On subsequent submits, the value fields will be replaced automatically with updated values.

Keywords and their corresponding values:

`$Author$`

The login name of the user who last modified the file. Note that this is a standard RCS keyword and does not refer to the TeamXpress Author role.

`$Date$`

The date and time the file was last modified.

`$Header$`

A standard header containing the path name of the file, the revision string, the date and time, the author. The path is relative to the area root.

`$Id$`

Same as `$Header$`, except that the filename is without a path.

`$Log$`

The comment supplied during submit, preceded by a header containing the filename, the revision string, the date and time when the file was last modified, and the author. The comment is either the submit comment supplied for the individual file or, if this is empty, the comment supplied for the all the files associated with the submit. Existing log messages are not replaced. Instead, the new log message is inserted after `$Log: . . . $`. This is useful for accumulating a complete change log in a file.

Each inserted line is prefixed by the string that prefixes the `$Log$` line. For example, if the `$Log$` line is `// $Log:tan.cc $`, RCS prefixes each line of the log with `" // "`. This is useful for languages with comments that go to the end of the line. The convention for other languages is to use a `*` prefix inside a multiline comment. For example, the initial log comment of a C program is conventionally of the following form:

```
/*
 * $Log$
 */
```

`$Revision$`

The revision string.

`$Source$`

The pathname of the file, relative to the root directory of the area.

These RCS keywords are ignored by TeamXpress:

```
$Locker$
$Name$
$RCSfile$
$State$
```

The following characters in keyword values are represented by escape sequences:

Character	Escape Sequence
end-of-line	<code>\n</code>
<code>\$</code>	<code>\044</code>
<code>\</code>	<code>\\</code>

## Enabling Macro Expansion

To enable RCS macro expansion for a particular set of files, you must include the following action in the `submit.cfg` rules that apply to those files:

```
expand_rcs_macros=yes
```

Here is a sample `submit.cfg` file:

```
CASE-SENSITIVE = YES
RULES
{
    "." # any workarea
    {
        ".*\.[ch]$" { gid=iw, expand_rcs_macros=yes }
        # files ending in .c or .h
        ".*" { gid=iw } # all other files/directories
    }
}
```

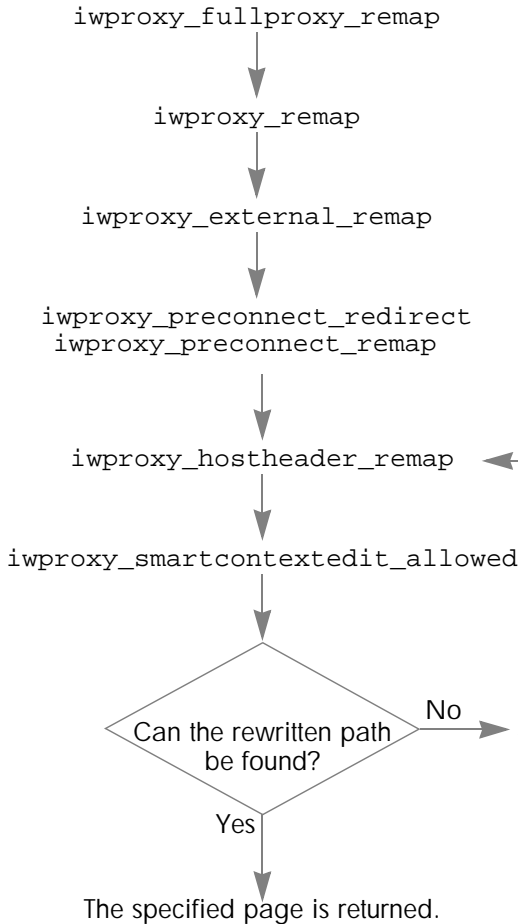
# Configuring the TeamXpress Proxy Server

## About the Proxy Server

TeamXpress uses a proxy server to perform several important functions:

1. Resolve relative and absolute path names in TeamXpress areas in order to present users with a virtualized view of the Web site contained within an area (see page 113).
2. Remap fully-qualified paths into TeamXpress areas (see page 117).
3. Redirect browsing in one branch or workarea into another area (see page 121).
4. Redirect individual workareas to use different Web servers (see page 124).
5. Remap links to external Web servers (see page 125).
6. Modify “Host:” headers (see page 126).
7. Remap SSI requests (page 127).

Each time a request is made through the TeamXpress proxy server, the following sections of `iw.cfg` are processed in the following order. More than one rule may be applied to a request. As a URL gets rewritten by a rule, the rewritten URL is passed to the next section. The first rule that matches in any section prevails; no other rule in that section will be applied.



See *Configuring TeamXpress to Redirect Fully-Qualified Paths*. Applies only if the browser's proxy has been set to the TeamXpress proxy server.

See *Document Roots*.

See *Configuring External Remappings*. Applies only if none of the preceding rules has matched.

See *Redirecting TeamXpress Views to Different Areas*.

See *Host Header Remappings*.

See *Enabling and Disabling SmartContext Editing*.

See *Configuring Proxy Failover*. Sends the rewritten path to be re-processed.



## Configuring Proxy Server Operation

The `[iwproxy]` section of `iw.cfg` is used to configure the operation of TeamXpress's proxy server.

For example:

```
[iwproxy]
iwproxy_port=1080
host_httpd_port=80
ipalias_hostname=hostname.com
```

where:

`iwproxy_port` is the port TeamXpress's proxy server will operate on. It should be set to an open port value (1080 is selected by default). If you set `iwproxy_port=0`, the TeamXpress proxy server will be disabled. However, it will not interfere with the normal operation of the Web server (although LaunchPad will also be disabled).

`host_httpd_port` is the port through which TeamXpress's proxy server communicates with the Web server. It should be set to the value of the port used by the Web server. Port 80 is selected by default.

`ipalias_hostname` is your IP aliasing host. If your site uses IP aliasing, this line should be included and its value should be set to the hostname of the virtual IP alias you want TeamXpress to work under. If you do not use IP aliasing, you do not need to include this line.

## Resolving Relative and Absolute Paths

### About Relative and Absolute Paths

Relative paths specify file locations relative to the referencing file's directory location. Absolute paths specify file locations relative to the Web site's document root directory. For example, the file whose directory path (rooted in a TeamXpress area) is:

```
htdocs/main/dev/index.html
```

might contain a link to the file

```
htdocs/images/banner.gif
```

This link can be specified as either a relative or an absolute path.

If the link were specified as a relative path, it would look like:

```
../images/banner.gif
```

If the link were specified as an absolute path, it would begin with a / and look like:

```
/htdocs/images/banner.gif
```

### Resolving Relative and Absolute Paths

Links in HTML documents are often specified with relative or absolute path names. For example, in a link to an image, the file name might appear as:

```
/images/pic.gif
```

On a typical Web server, this link reference would be mapped by the Web server to its document root, e.g.:

```
/images/pic.gif ==> /usr/httpd/htdocs/images/pic.gif
```

All users attempting to access the file using the absolute path name will be mapped to the same file location on the Web server.

However, TeamXpress supports a system of private workareas, giving each user access to the Web site's files from within their own personal, virtual version of the Web site. TeamXpress uses a proxy server to manage mapping of files to workareas with relative and absolute path references. Going back to our example, the TeamXpress proxy server allows each user attempting to access `/images/pic.gif` from within TeamXpress to be mapped to the copy of `pic.gif` in his own workarea. The redirected mapping would look like:

```
/images/pic.gif ==>  
/iw-mount/default/main/dev/WORKAREA/workareaname/htdocs/  
images/  
pic.gif
```

## Document Roots

TeamXpress maps the initial Web server directory structure (*document root*) of workareas to the top level of the workarea directory by default. You may, however, want to move the document root, or group types of files together for improved clarity, convenience, or efficiency. On a branch-by-branch basis, the TeamXpress proxy server allows you to remap the document root anywhere within the workarea directory. It also allows you to define mappings directly to sub-directories within workareas.

Path mappings are defined by including sections within the TeamXpress main configuration file, `iw.cfg`.

To configure document roots for individual branches:

1. For each branch that you want to configure, add a line to the `[iwproxy_remap]` section of `iw.cfg`, of the form:

```
configsectionname=vpath
```

where *vpath* is the *vpath* to the branch you are configuring, and *configsectionname* is the name of the section of the configuration file that will define the branch remappings.

2. For each line that you added to `[iwproxy_remap]`, create a section in `iw.cfg` named `[configsectionname]`. Add a line to this section that defines the document root:

```
_docroot=dirpath
```

where *dirpath* is a directory path rooted in a workarea.

You can also add lines that bypass the document root, of the format:

```
path=path
```

For example, you might add the following lines to `[iwproxy_remap]`:

```
[iwproxy_remap]
branchrewrite_1=/main/dev
branchrewrite_2=/main/dev/training
```

The first line of the above example tells TeamXpress to use the section `[branchrewrite_1]` to set the document root configuration for the `/main/dev` branch. The second line tells TeamXpress to use the `[branchrewrite_2]` section to set the document root configuration for the `/main/dev/training` branch.

You would then create two new configuration file sections corresponding to the lines in

```
[iwproxy_remap]:
```

```
[branchrewrite_1]
_docroot=/htdocs
/pictures/=/pictures/
/html/=/html/
```

```
[branchrewrite_2]
_docroot=/htdocs
/images/=/images/
```

The first line of both the new sections contains:

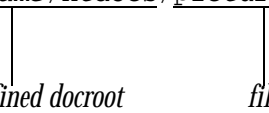
```
_docroot=/htdocs
```

This defines a special directive that sets the mapping of the document root. Any requests from workareas on the `main` branch or the `main/dev/training` branch to the root level directory (`/`) will now start at:

```
.../workareaname/htdocs/
```

Thus, the request for file `/picture1.gif` will now be mapped to:

```
.../workareaname/htdocs/picture1.gif
```



The two docroot configuration sections also contain lines similar to the following:

```
/pictures/=/pictures/
```

This line maps file requests directly to the listed directory `/pictures/`, bypassing the document root defined in the first line. Thus, a request for the file `/pictures/people.gif` gets mapped to:

```
.../workareaname/pictures/people.gif
```

not:

```
.../workareaname/htdocs/pictures/people.gif
```

TeamXpress's proxy server operates using literal string matches and substitutions in path names. To avoid inadvertently rewriting names, always use trailing slashes in your remap definitions.

## Resolving Fully-Qualified Paths

The TeamXpress proxy server can also be configured to resolve fully-qualified paths. For example, a link to the main page of a Web site might appear as

```
http://www.name.com
```

If such a link appears in an HTML file in a TeamXpress workarea, and you follow that link while performing in-context QA, you will be taken out of the workarea and to the actual referenced Web site.

Therefore, if you use fully-qualified paths to reference pages within your own Web site, clicking on these links will take you out of the in-context view of the current workarea, staging area, or edition and into your own currently deployed Web site. To solve this problem, TeamXpress allows you to configure your proxy server. The proxy server will redirect fully-qualified links within your Web site, then pass them to the regular proxy server to ensure the integrity of the in-context view in a workarea, staging area, or edition.

- ! *Only* configure this setting if your Web site uses fully-qualified paths that you need to view in-context! This setting requires you to manually configure your browser, so that you will not be able to view the actual Web site without reconfiguring your browser.

### Configuring TeamXpress to Redirect Fully-Qualified Paths

To configure TeamXpress to redirect fully-qualified paths, you must:

- Configure the TeamXpress server.
- Set your (client) browser's proxy to the TeamXpress proxy server.

### Configuring the TeamXpress Proxy Server to Redirect Fully-Qualified Paths

To configure the TeamXpress server to redirect fully-qualified paths, edit the `[iwproxy_fullproxy_remap]` section of `iw.cfg`. This section contains any number of `_regex` lines. Each line is of the format:

```
_regex=source_regex=dest_ex
```

where `source_regex` is a case-insensitive regular expression specifying a fully-qualified path that might appear in a page, and `dest_ex` is an expression specifying the path that the link will be redirected to. This expression should always be the path to the file specified in `source_regex`, but rooted in a TeamXpress area.

For example:

```
[iwproxy_fullproxy_remap]
_regex=http://www(\.example\.com)?/(.*)=/$2
```

redirects links that specify

```
http://www.example.com
```

in the URL and sends them to the corresponding location in the current TeamXpress area.

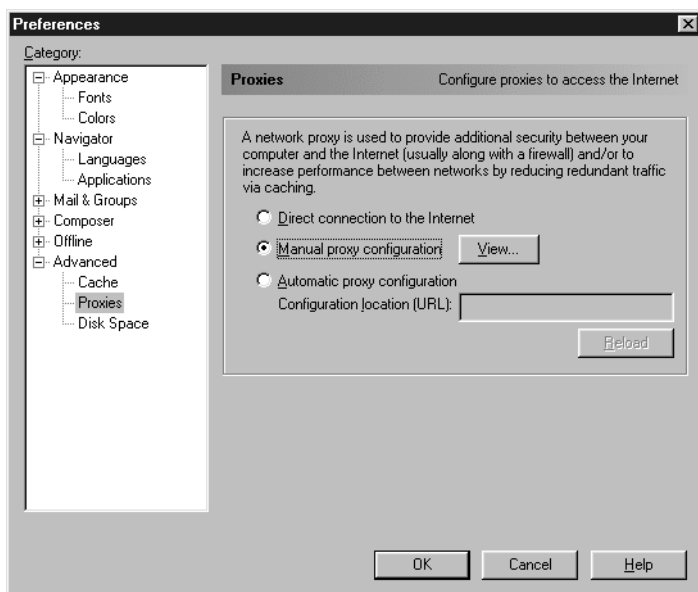
### Configuring the Client for Fully-Qualified Path Redirection

If you are using `[iwproxy_fullproxy_remap]`, you must set up your (client) browsers to go through the TeamXpress proxy server. All requests will then go through `iwproxy`. If you need to browse one of the live Web sites that the TeamXpress proxy server reroutes requests for, you will need to set your browser to not use the TeamXpress proxy server.

## Netscape

To configure your browser to use the TeamXpress proxy server:

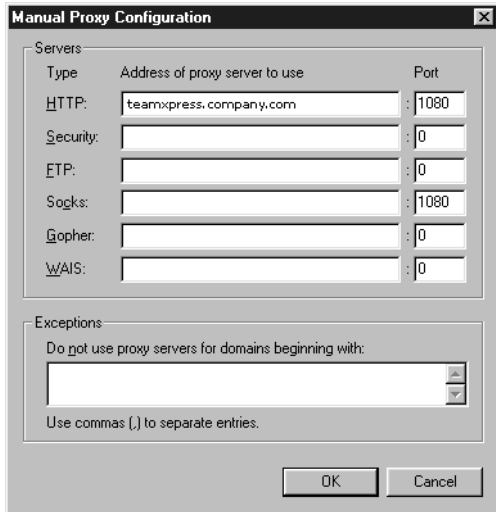
1. In Netscape Navigator, select **Preferences** from the **Edit** menu.
2. The Preferences window will appear. Double-click on the **Advanced** category in the left-hand pane.
3. Select the **Proxies** subcategory in the left-hand pane.



*The Netscape Preferences Window*

4. In the right-hand pane, select **Manual proxy configuration**. Select **View**.

- The Manual Proxy Configuration window will appear. Type the name of your TeamXpress server (e.g. TeamXpress1.example.com) in the **HTTP** section. Type the `iwproxy_port` specified in the `[iwproxy]` section of `iw.cfg` (e.g. 1080) in the **Port** section.



*The Manual Proxy Configuration Window*

- Click **OK**.
- In the Preferences window, click **OK**.

To configure Netscape to not use the TeamXpress proxy server:

- In Netscape Navigator, select **Preferences** from the **Edit** menu.
- The Preferences window will appear. Double-click on the **Advanced** category in the left-hand pane.
- Select the **Proxies** subcategory in the left-hand pane.
- In the right-hand pane, select **Direct connection to the Internet**.
- Click **OK**.
- In the Preferences window, click **OK**.



## Internet Explorer

To configure your browser to use the TeamXpress proxy server:

1. In Internet Explorer, select **Internet Options** from the **View** menu. The Internet Options window will appear.
2. Select the **Connection** tab.
3. Select the **Access the Internet using a proxy server** checkbox.
4. Type the name of your TeamXpress server (e.g. `TeamXpress1.example.com`) in the **Address** section. Type the `iwproxy_port` specified in the `[iwproxy]` section of `iw.cfg` (e.g. 1080) in the **Port** section.
5. Click **OK**.

To configure Internet Explorer to not use the TeamXpress proxy server:

1. In Internet Explorer, select **Internet Options** from the **View** menu. The Internet Options window will appear.
2. Select the **Connection** tab.
3. Deselect the **Access the Internet using a proxy server** checkbox.
4. Click **OK**.

## Redirecting TeamXpress Views to Different Areas

TeamXpress's proxy server allows Web teams to work on branches of development that are populated only with the portion of the Web site that they are developing, but still maintain a fully in-context view of the entire Web site by referencing the staging area or a known edition on another branch of development.

This feature is very flexible in that it can be configured on a per-branch or per-workarea basis, and the redirected view can be configured to take the user to any TeamXpress area on any branch. Redirection can occur in one of two ways:

1. Through `[iwproxy_preconnect_remap]`, which retains your original area as the current working area and directs files there from another area. In this scenario, `docroot` is based on the original area's parent branch.
2. Through `[iwproxy_preconnect_redirect]`, which causes the area you redirect into to become the current working area (and that area's parent branch becomes the basis of `docroot`).

### Using `[iwproxy_preconnect_remap]`

To configure TeamXpress to redirect workarea views as described in Item 1 above, edit the `[iwproxy_preconnect_remap]` section of `iw.cfg`:

```
[iwproxy_preconnect_remap]
_iwregex=source_regex=dest_ex
```

where `source_regex` is a case-insensitive regular expression describing the area to be mapped from, and `dest_ex` is an expression describing the area to be mapped to. These areas are most commonly workareas or staging areas, but you can map to and from any workarea, staging area, or edition. You can add any number of `_iwregex` lines to this section.

For example:

```
_iwregex=(.*)/branch1/WORKAREA/[^/]+/products/(.*)=$1/branch2/
STAGING/products/$2
```

tells the proxy server to remap the `products` directory of any workarea on any branch named `branch1` to the `products` directory of the staging area on its sister branch, `branch2`. To clarify:

In the source regular expression, `(.*)` is used to specify an arbitrary path, and `$1` in the destination expression means that it must follow the same path (and therefore `branch1` can be anywhere in the branch structure, but `branch2` is a sister branch of `branch1`). Also in the source regular expression, `[^/]+` is used to specify a single directory level, of any name (which in this case would be the workarea name, and therefore all workareas on `branch1` are specified). Finally, the source regular expression uses `(.*)` to specify another arbitrary path, and `$2` in the destination expression tells it to follow the same path.

You can also specify the exact location of the areas you want to remap:

```
_regex=^/
iw-mount/default/main/dev/branch1/WORKAREA/[^/ ]+/products/(.*)=/
iw-mount/default/main/dev/branch2/STAGING/products/$1
```

Or, you can specify an individual workarea to remap:

```
_regex=^/
iw-mount/default/main/dev/branch1/WORKAREA/andre/coolstuff/(.*)=/
iw-mount/default/main/dev/branch2/STAGING/coolstuff/$1
```

The TeamXpress proxy server applies the first match it finds, so you can exclude a particular area from a more general rule by creating a separate rule for that area and placing it before the more general rule. For example:

```
_regex=(.*)/branch1/WORKAREA/chris/products/(.*)=$1/branch1/
STAGING/products/$2
_regex=(.*)/branch1/WORKAREA/[^/ ]+/products/(.*)=$1/branch2/
STAGING/products/$2
```

redirects the `products` directory in all workareas on `branch1` except for Chris's to the staging area of `branch2`.

See “Configuring Proxy Failover” on page 127 for a details about configuration rule precedence.

### Using `[iwproxy_preconnect_redirect]`

To configure TeamXpress to redirect workarea views as described in Item 2 above, edit the `[iwproxy_preconnect_redirect]` section of `iw.cfg`:

```
[iwproxy_preconnect_redirect]
_regex=source_regex=dest_ex
```

where `source_regex` and `dest_ex` are as described in “Using `[iwproxy_preconnect_remap]`” on page 122. If you set `[iwproxy_preconnect_redirect]` and then click on a link defined by an absolute path name, the docroot of that link is based on the branch you redirected into (as opposed to the branch of the area you redirected from, which would be the behavior if you had set `[iwproxy_preconnect_remap]`). See “Configuring Proxy Failover” on page 127 for a details about configuration rule precedence.

## Configuring TeamXpress to Use Different Webservers

You can configure TeamXpress to use different Web servers for different workareas or different types of content. For example, Andre might want to make all CGIs in his workarea on `branch1` (subject to no constraints whatsoever on the arguments these CGIs or may not take) be served by `test1.example.com:1234`. This would let Andre test different Web server configurations for his CGIs on `branch1` without disturbing anyone else.

To configure TeamXpress to use different Web servers, edit the `[iwproxy_preconnect_remap]` section of `iw.cfg`:

```
[iwproxy_preconnect_remap]
_regex=source_regex=dest_ex
```

where `source_regex` is a case-insensitive regular expression describing the area and files to be served by the other Web server, and `dest_ex` is an expression describing the area and files on the other Web server. This expression must include the port number.

For this to work properly, the other Web server must have the appropriate NFS TeamXpress directory mounts and privileges. The Web server aliases used by `httpd` on port 1234 of `test1.example.com` must be configured with the TeamXpress aliases and `ScriptAliases`, as well (`/iw-bin/`, `/iw-icons/`, `/iw-mount/`, `/iw`, and `/iw/`).

The following example would allow Andre to test all CGIs in his workarea on `test1.example.com`, as described above:

```
[iwproxy_preconnect_remap]
_regex=^/iw-mount/default/main/dev/branch1/WORKAREA/andre/
(.*).cgi(\?.*)?=$=http://test1.example.com:1234/iw-mount/default/main/
dev/branch1/WORKAREA/andre/$1.cgi$2
```

## Configuring External Remappings

The TeamXpress proxy server allows you to define mappings to directories outside of the TeamXpress system or on different computers altogether. There are two ways to define these mappings:

- Through `[iwproxy_preconnect_remap]`.
- Through `[iwproxy_external_remap]`.

If you use `[iwproxy_preconnect_remap]`, these mappings will follow normal `[iwproxy_preconnect_remap]` precedence rules. However, `[iwproxy_external_remap]` mappings apply *only* if no other remapping rule has been applied.

### `[iwproxy_preconnect_remap]`

To configure TeamXpress to redirect workarea views to external Web servers, edit the `[iwproxy_preconnect_remap]` section of `iw.cfg`:

```
[iwproxy_preconnect_remap]
_regex=source_regex=dest_ex
```

where `source_regex` is a case-insensitive regular expression describing the area to be mapped from, and `dest_ex` is an expression describing the area to be mapped to. These areas are most commonly workareas or staging areas, but you can map to and from any workarea, staging area, or edition.

For example:

```
_regex=(.*)/branch1/WORKAREA/[^/]+/logos/(.*)=
http://corporateidserver.example.com/logos/$2
```

will send all requests for files in the `/logos` directory in all workareas on `branch1` to another server, `corporateidserver.example.com`.

### `[iwproxy_external_remap]`

You can also use `[iwproxy_external_remap]` rules for external remappings. This usage is being phased out; use `[iwproxy_preconnect_remap]` whenever possible.

For example, if all your corporate logo files reside on a separate server, you can use `[iwproxy_external_remap]` to create a mapping to the directory where they reside:

```
[iwproxy_external_remap]
/logos/=http://corporateidserver.example.com/logos/
```

This remapping sends all requests for `/logos/` to a directory on another server, `corporateidserver.example.com/logos/`. You can also create associations using case-insensitive regular expression mapping.

The `[iwproxy_external_remap]` section is read after the `[iwproxy_remap]` section, and it will only be applied if none of the `[iwproxy_remap]` rules were invoked. For example, if you create a mapping for `/logos/` in one of the `[branchrewrite]` sections, all requests on that branch for files in the `/logos/` directory will use that mapping *instead of* the external mapping. Requests on other branches will still be sent to the `corporateidserver`.

## Host Header Remappings

If your Web server manipulates “Host:” headers (e.g. virtual domains), you can configure TeamXpress to have the same behavior. To configure “Host:” header remapping, edit the

```
[iwproxy_hostheader_remap] section of iw.cfg.
[iwproxy_hostheader_remap]
_regex=source_regex=dest_ex
```

where `source_regex` is a case-insensitive regular expression describing the area to be mapped from, and `dest_ex` is an expression describing the new “Host:” header. For example:

```
_regex=^/iw-mount/default/main/dev/branch1/WORKAREA/.*=example.com:1234
```

will change the “Host:” header that the origin server gets from the TeamXpress proxy server to read:

```
Host: example.com:1234
```

whenever content in a workarea on `branch1` is accessed.

## Configuring SSI Remapping

TeamXpress supports the ability to remap SSI requests. This feature is distinct from SSI request virtualization; you must still install the necessary redirector module (`iwproxy_nsapi.dll` or `iwproxy_isapi.dll`) to enable virtualization.

After installing the necessary redirector module as described in “Server-Side Includes and Secure Socket Layer” on page 29, you can configure TeamXpress to remap SSI requests by adding or modifying the `[iwproxy_plugin_remap]` section of `iw.cfg`. In the following example, any SSI request containing the string `/forms/` is mapped to `/default/main/dev/Branch2/STAGING/forms` instead of being referred to the root of the user’s workarea:

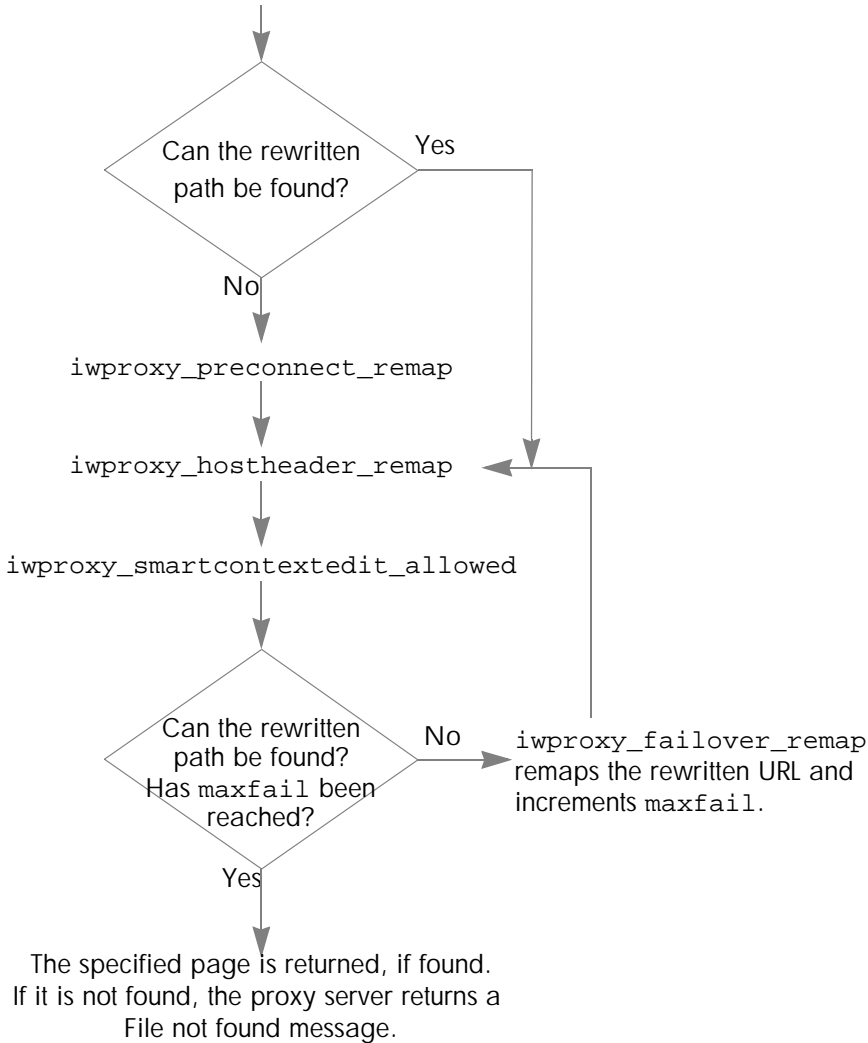
```
[iwproxy_plugin_remap]
_regex=(.*)/forms/(.*)=/default/main/dev/Branch2/STAGING/forms/$2
```

## Configuring Proxy Failover

If a requested page does not exist, the `[iwproxy_failover_remap]` section of `iw.cfg` can be used to specify an alternate location. This section allows you to specify both alternate locations and the number of times to process an URL in an attempt to find a valid location. The figure below illustrates the process by which proxy failover remaps URLs.



The proxy server rewrites the URL according to rules specified in  
iwproxy\_fullproxy\_remap  
iwproxy\_remap  
iwproxy\_external\_remap  
iwproxy\_preconnect\_redirect





The `[iwproxy_failover_remap]` section has the following structure:

```
[iwproxy_failover_remap]
_maxfail=#
_regex=source_regex=dest_ex
_regex=source_regex=dest_ex
```

To specify the number of times to try to remap a URL, edit the `_maxfail` line of the `[iwproxy_failover_remap]` section of `iw.cfg`. The default value of this line is `_maxfail=0`, which turns off proxy failover. Note that proxy failover is seldom needed because files are almost always in locations that can be specified via static, case-insensitive regular expressions during configuration. If you need to enable proxy failover, it is recommended that you do not set `_maxfail` to more than 1 or 2 due to the impact on system performance.

To specify expressions to remap, add `_regex` lines to `[iwproxy_failover_remap]`. These lines specify an incoming pattern to match, and an expression that they should be mapped to. The proxy server will take the first match it finds, remap it as specified, then try to locate the page. If it cannot find the new location, it will try to match the remapped expression to a regular expression specified in `[iwproxy_failover_remap]`. This process will continue until a match is found or the number of iterations specified by the `_maxfail` line is reached.

`_regex` lines in the `[iwproxy_failover_remap]` section follow the same syntax as `_regex` lines specified in the `[iwproxy_preconnect_remap]` section of `iw.cfg`, where `source_regex` is a case-insensitive regular expression describing the area to be mapped from, and `dest_ex` is an expression describing the area to be mapped to. For examples of `_regex` syntax, see “Resolving Relative and Absolute Paths” on page 113.

## Debugging Your Proxy Server Configuration

If your proxy server does not seem to be configured correctly, use the `iwproxy` CLT’s debug option to list all the translations being made by the proxy server:

```
iwproxy [-d|-x]
```

- |                 |  |
|-----------------|--|
| <code>-d</code> | Debug mode (outputs client & server headers)                     |
| <code>-x</code> | Extended (verbose) debug mode (outputs client body text as well) |

`iwproxy` will return debug output which you can redirect to a file. Note that `iwproxy`'s debug mode is single-threaded; it therefore slows the TeamXpress server down tremendously. Use the debug mode only for diagnostic purposes.

One common source of proxy configuration problems is the inclusion of any character or blank space past the end of a branch name in any line in any `[iwproxy*]` section in `iw.cfg`. For example, the following line in the `[iwproxy_remap]` section is illegal because it contains blank spaces and characters after the branch name:

```
[iwproxy-remap]
tag_engspecs=/main/dev/engspecs #This is the engineering spec site
```

## Configuring TeamXpress Failsafe

TeamXpress Failsafe improves TeamXpress reliability by copying TeamXpress cache entries to a temporary disk backup file. If the TeamXpress server terminates abnormally, these cache entry copies are accessed automatically to restore the backing store when you restart the TeamXpress server. This feature significantly reduces the likelihood of metadata inconsistencies caused by abnormal server termination.

### Failsafe Components and Processes

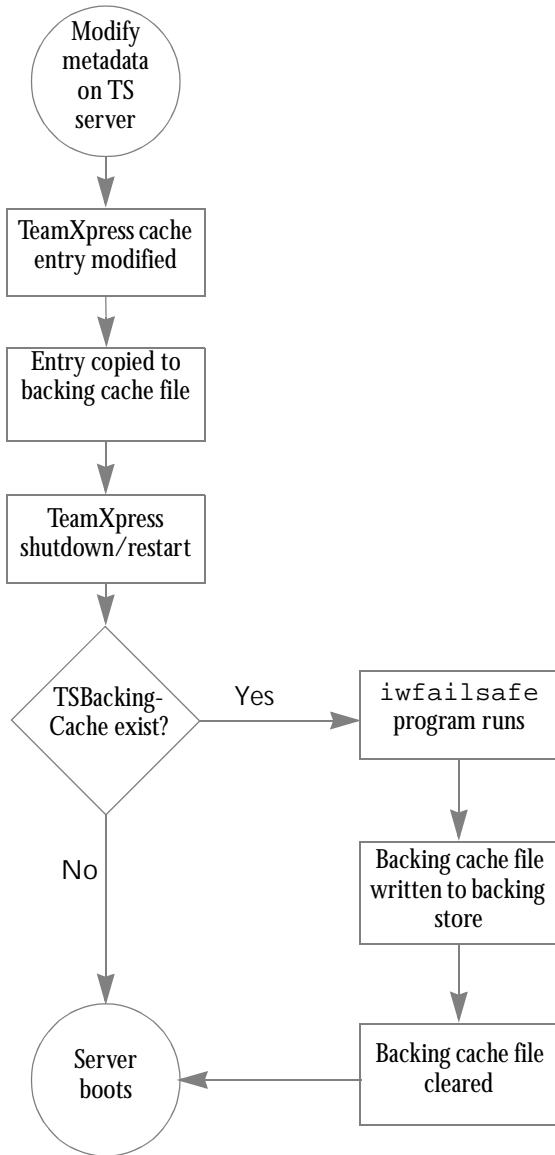
Whenever TeamXpress metadata is modified, it is first stored in the TeamXpress cache. The modified entries remain in the cache until the TeamXpress server load is light enough that they can be written to the backing store. A metadata inconsistency could occur if the TeamXpress server were to terminate unexpectedly before the cache could be written to the backing store.

TeamXpress failsafe addresses this issue by automatically copying all modified TeamXpress cache entries to a backing cache file at the time the entries are modified. Whenever the TeamXpress server is started following a TeamXpress shutdown, the startup script `iwetcbboot TeamXpress` automatically checks for the existence of modified cache entries in the backing cache file. Existence of these entries indicates that the most recent TeamXpress shutdown was abnormal (i.e., due to a power failure, process failure, or some other unexpected condition).

If TeamXpress detects a normal shutdown (i.e., no modified entries exist in the backing cache file), it proceeds with the boot process.

If TeamXpress detects an abnormal shutdown (i.e., modified entries exist in the backing cache file), it runs the `iwfailsafe` restoration program that writes the entries from the backing cache file to the backing store. The boot process then continues.

The following flowchart shows the processes involved in both normal and abnormal TeamXpress shutdowns.



*TeamXpress Failsafe Process Flow*

## Configuration Parameters

Failsafe has a number of configurable parameters that allow the overall performance of the system to be tuned. These parameters are discussed in the following sections. All of these parameters are set in the `[iwserver]` section of `/etc/iw.cfg`.

### Turning Failsafe On and Off

Failsafe is enabled with the `failsafe` parameter. The default value is `yes`, which causes failsafe to be enabled:

```
failsafe=yes
```

Failsafe can be disabled by changing this value to `no`.

### The Failsafe Backing Store: `TSBackingCache`

Failsafe stores its copy of the cache contents in a regular file in the file system named `TSBackingCache`, located in the `$IW_STORE` directory by default. If this file is present, it should never be removed. Operation of the server will not be affected, but any ability to recover from a server crash or system failure will be lost.

### The Size of `TSBackingCache`

In the default configuration, `TSBackingCache` is sized in proportion to the internal cache size used by the server. The file initially starts out with a size, in megabytes, of  $\text{cachesize} / 2$ , or about 15 megabytes for the default `cachesize`. Because some file system metadata objects can grow quite large over time, this file is allowed to grow to up to 8 times the original size or  $\text{cachesize} * 4$  megabytes; about 120 megabytes for the default `cachesize`.

`TSBackingCache` can be sized to suit the needs of a particular installation by specifying the `failsafe_size` parameter in megabytes. For example, to set a `TSBackingCache` size of 128Mb, you would assign a value of 128 to this parameter:

```
failsafe_size=128
```

When used with a value specified for the `failsafe_size` parameter, the file is created at the specified size and does not grow. The value of this parameter is limited to a range of 1Mb to 768Mb.

There is an interaction between the size of the `TSBackingCache` file and overall system performance. A file size that is too small will result in more frequent cache flushes, which will lead to severe degradation of performance. These flushes occur when the backing cache no longer has adequate space to contain the data for a point that is being added. Flushing the cache forces all data to disk and, as a consequence, causes points to be removed from the backing cache, thus allowing additional points to be added to the backing cache. Ideally no cache flushes should be forced by the failsafe subsystem in normal operation.

If a system is experiencing freezes, failsafe can be pinpointed as the source of the problem by issuing an `iwstat` command. If the failsafe system is causing the freezes, `iwstat` will report a `BackingCacheFlush` as the operation in progress. If you find this happening, the size of `TSBackingCache` should be increased by specifying a larger value for `failsafe_size`.

Conversely, the server may have problems allocating memory from the heap or may run out of virtual space. If this happens, the size of `TSBackingCache` is most likely very large and can be reduced by specifying a smaller value for `failsafe_size`.

### The Location of `TSBackingCache`

The location of the `TSBackingCache` file can be changed by assigning a value to the `failsafe_path` variable. This variable specifies the full path name of the location of the `TSBackingCache` file. For example:

```
failsafe_path=$IW_STORE/TSBackingCache
```

If the last element of this pathname is not `TSBackingCache`, then the name of the backing file will not be `TSBackingCache`.

In general, performance of the server will be better if `TSBackingCache` is located on a different disk from `$IW_STORE`. Space is also a consideration: if the file system used for the backing store is small or nearly full, `TSBackingCache` should be placed in a separate file system. Finally, `TSBackingCache` should never be placed in `/tmp` or other directories that are cleaned during the boot process. If this is done, it will not be possible to do a recovery after a system crash or power failure.

### Reliability: Frequency of Flushes

Periodically, the server runs a thread that synchronizes the internal copy of `TSBackingCache` with the on-disk copy. Because there is some cost associated with performing this operation, the `failsafe_delay` parameter is provided. `failsafe_delay` provides the ability to trade performance for increased reliability in the event of a system crash or power failure.

`failsafe_delay` is specified in seconds and has a range of 1 to 60. The default value is 60 seconds:

```
failsafe_delay=60
```

### Performing a Recovery: Running `iwfailsafe`

In a system that does not have the TeamXpress HA package installed, the `iwfailsafe` program is run automatically when the existence of the `TSBackingCache` file is detected during the server boot process. This check is done in `iwetcbboot`, and can be disabled by editing `iwetcbboot`. TeamXpress

On a system that has HA installed, this feature is disabled and `iwfailsafe` must be run from the `iw.powerfail` and `iw.processfail` scripts provided by the HA package. Note that as shipped these scripts do not run `iwfailsafe` and must be edited to meet the needs of the particular installation. See “Running `iwfailsafe` Manually” on page 135 for more details on the use of `iwfailsafe`.

## Running `iwfailsafe` Manually

In addition to running automatically as described in the preceding sections, `iwfailsafe` is executable from the command line. If you suspect a metadata inconsistency, you should check for the following:

- The existence of the TeamXpress backing cache file `$IW_STORE/TSBackingCache`.
- The existence of the file `$IW_STORE/TSRunning` while the server is known to be down.

If these two conditions exist at the same time, you should run `iwfailsafe` from the command line as described in the next section.

## iwfailsafe Command Line Usage

### Usage

`iwfailsafe [-h] [-n] [-v] [-V] [-c filename]`

<code>-h</code>	Display usage message.
<code>-n</code>	Consistency check. Reads the log; exit code indicates log status.
<code>-v</code>	Display version. Has highest precedence if multiple options are specified.
<code>-V</code>	Verbose mode. Prints information about points as they are processed.
<code>-c <i>filename</i></code>	Use <i>filename</i> instead of the default as the log file.

### Exit Status

The following exit values are returned:

0	Success. Log file was processed and no errors were found.
1	Error. Log file could not be opened or errors were found while processing log file.

### Notes

`iwfailsafe` uses a proactive method of error detection. It is capable of processing all of the points contained in the log file up until an error is detected. It is suggested that the program be run with the `-n` option as a basic check on the integrity of the log file. If errors are detected, `iwfailsafe` can be run again with the `-n` and `-v` options to get a complete report on the status of the points in the file.

### Configuration Files

`iw.cfg`



# Configuring TeamXpress Workflow

---

TeamXpress's workflow system consists of two main components:

- A server-side subsystem, which provides a framework for controlling complex Web site production processes.
- A browser based client-side user interface written in Javascript, which lets end users enter data to define and control specific workflow actions.

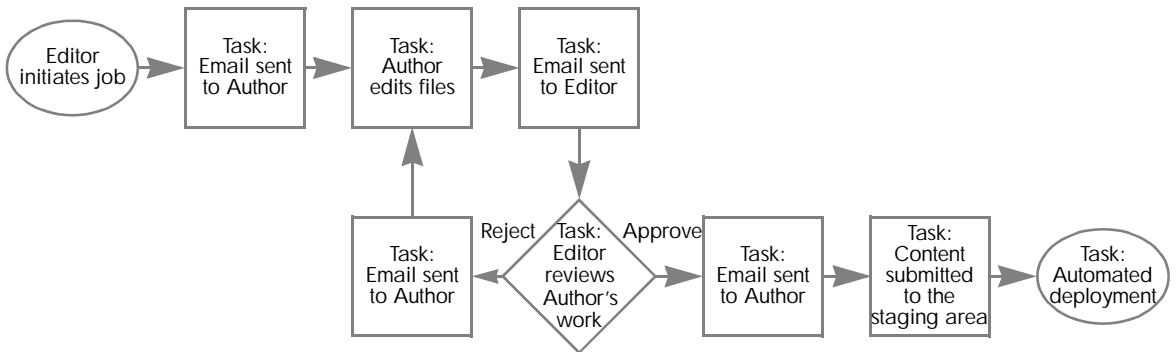
This document describes only the server-side subsystem. See the *TeamXpress User's Guide* for information about the workflow user interface.

## Definitions of Workflow Terms

### Workflow Models

A *workflow model* is a general workflow configuration that can be used repeatedly. Each workflow model describes a process that can include user tasks and a wide variety of automated tasks. Workflow models typically are designed by business managers and configured by a system administrator.

The following diagram shows a very simple assign-edit-approve workflow model. Email is sent to the participants at every stage of the process, and some automated tasks are performed at the end.

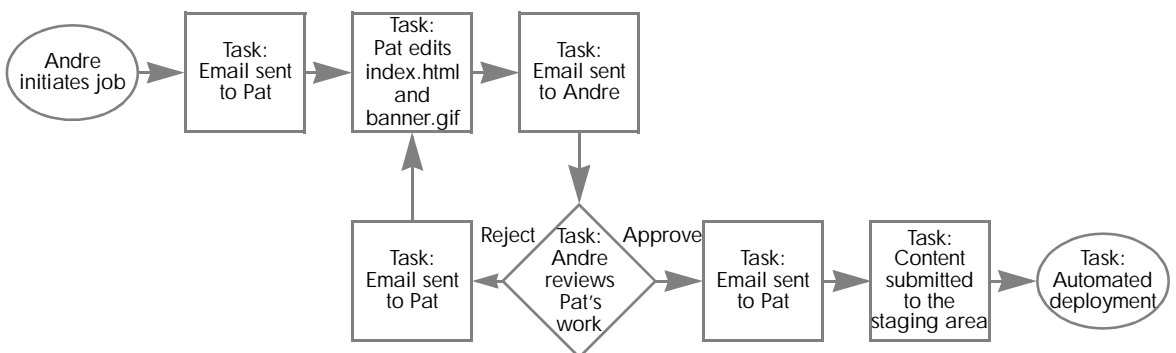


*Workflow model*

## Jobs

A *job* is a set of interdependent tasks. One example of a TeamXpress job would be the set of tasks needed to prepare a new section in a marketing Web site to support a new product launch.

In TeamXpress, a description of a workflow model is called a *job specification*. When a job specification is loaded into the workflow subsystem it becomes a *job instance*. In other words, each job is a specific instance of a workflow model. When a job is created, the job creator must supply all the specific information for that job. For example, the workflow model above might be used to create the following job:



*Job model*

Because jobs follow predefined workflow models, tasks cannot be added to or removed from individual jobs.

## Tasks

A *task* is a unit of work performed by a single user or process. Each task in a job is associated with a particular TeamXpress area and carries a set of files with it. The user or process owning a task can modify, add files to, or remove files from the task (provided the task is not a read-only task for content approval).

Tasks have two possible states: active and inactive. A task becomes active when its predecessor tasks signal it to do so (predecessor tasks and conditions for activation are all configured as part of the workflow model).<sup>1</sup> After the task has been activated, users or external programs can do work on it. For example, after a user task has been activated, the user can work on the files contained in the task. After an external task has been activated, the appropriate external program can run on the files contained in the task. Inactive tasks are tasks that have been completed, or that have not been activated yet. Tasks can also be designated as read-only or editable. A read-only task lets users approve and comment on content without being able to edit, delete, add, or remove files from the task.

## Creating a Job

Jobs can be created in two ways:

1. By directly editing an XML *job specification file*, which defines a single job, or
2. Through a combination of workflow rules defined in an XML *workflow template file* and end-user input from the browser interface. With this method, you can create a single workflow template file to define multiple jobs that differ depending on what input an end user provides. The workflow rules and end-user input are interpreted by a CGI to dynamically create a job specification file. This is the same type of file that you would have to create manually if you used Method 1 described above. Once configured, this method greatly simplifies the process of defining jobs because a) it provides a browser interface for end-user input, and b) it automatically generates separate job specification files for each distinct set of user inputs.

---

1. A workflow task cannot have more than 512 predecessors.

For details about directly editing a job specification file to define a single job, see “Job Creation: Directly Editing a Job Specification File” on page 140. For details about setting up a workflow template file and its associated user interface and CGI, see “Job Creation: Setting up a Workflow Template File” on page 140.

## Job Creation: Directly Editing a Job Specification File

When you directly edit an XML job specification file to define a single job, the file must reside in the TeamXpress home directory. The file must be structured as described in “Understanding Job Specification File Structure” on page 180 and must use elements as described in “Element Definitions” on page 180. See Appendix B, “Sample Job Specification File” for an example of this type of file. For a detailed XML specification, see <http://www.xml.com/axml/testaxml.htm>.

After creating a job specification file to define a workflow model, you must create an instance of it (i.e., a job) on the server by running `iwjobc`. This is known as *instantiating* the job on the server. To execute the job after it has been instantiated on the server, you must execute the `iwinvokejob` utility. Note that even though a job specification file can only define a single workflow model, it is possible to instantiate multiple, identical, concurrent jobs by instantiating and executing the same job specification file more than once using `iwjobc` and `iwinvokejob`. Upon invocation, the job runs until one of its end tasks is activated. Once a job ends, its instance is removed, and you must re-instantiate it to run it again. Various utilities allow you to destroy jobs, view the state of any object in the workflow system, add files to a particular task within the job, and otherwise interact with running jobs.

## Job Creation: Setting up a Workflow Template File

There are two ways to set up a workflow template file to create a job. You can create a completely new template file, or you can edit a sample template file that comes with TeamXpress. Before using either method, you should be familiar with overall workflow template file concepts, and know the details of workflow template structure and elements. See “Understanding Workflow Template File Structure” on page 144 for details about file structure and elements. See “Using Pre-Packaged TeamXpress Workflow Templates” on page 171 for details about sample workflow template files that you can customize for your installation.

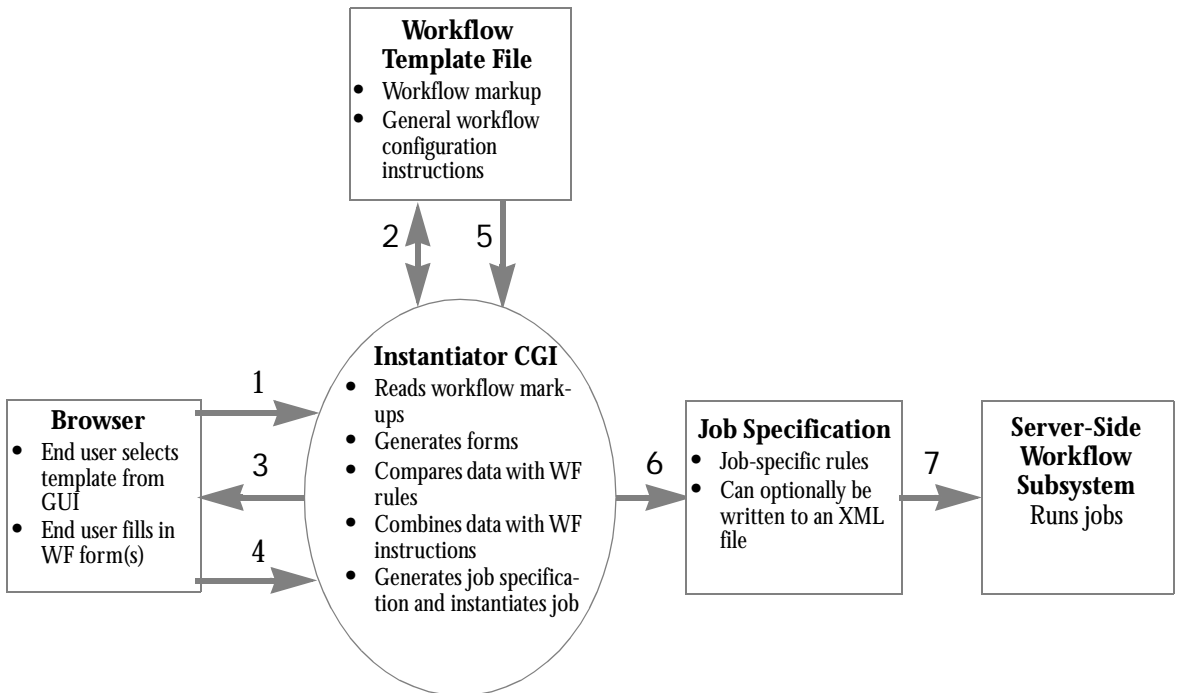
Five main components come into play when you set up a workflow template file to create a job:

- The workflow template file itself, which defines the workflow rules through a set of workflow markups and a set of general workflow configuration instructions. A TeamXpress installation can contain any number of workflow template files. See “Adding Template Selection Choices to TeamXpress GUI Menus” on page 166 for information about how to install workflow template files so that they appear on an end user’s list of template choices in the TeamXpress GUI.
- An instantiator CGI that interprets the workflow rules and data from end users, produces browser graphics and prompts, generates a job specification, and instantiates the job.
- A browser-based GUI for end-user input.
- A job specification file that can be generated by the instantiator CGI.<sup>1</sup>
- The server-side workflow subsystem.

The following diagram shows how these components work together. Sections after the diagram explain each diagram step and component in detail.

---

1. While the workflow subsystem can be configured to create and save a job specification file for any job, the normal scenario is for the job to be instantiated without the job specification being saved in a file. Saving the job specification in a file is a step that is usually taken only when you need to view the file for debugging



### Workflow template overview

#### Diagram Key

1. In the TeamXpress GUI, an end user selects a workflow template from the **File > New File**, **File > Edit File**, or **File > Create New Job** menu item. The instantiator CGI reads the file `available_templates.ipl` to determine which workflow template files are available for that given TeamXpress area or file content.
2. The instantiator CGI goes to the specified workflow template file and reads the workflow markup, which consists of Perl instructions residing in the workflow template file's `<template_script>` element(s). See page 148 for details about `<template_script>` syntax and usage.
3. Based on the workflow markup, the instantiator CGI creates one or more workflow forms into which an end user can enter workflow configuration information via the browser.
4. An end user working through the GUI fills in the workflow form and submits it back to the instantiator CGI.

5. The instantiator CGI consults the rules in the workflow template file's workflow markup to verify the validity of the data entered by the end user. If the data meets all necessary criteria, it is parsed by the instantiator CGI (see Step 6). If the data does not meet all necessary criteria, the interface reprompts the end user so that data can be re-entered (default notification is the invalid field turning red in the workflow form).
6. After determining that the workflow form contains valid data, the instantiator CGI combines the data with the general instructions from the workflow template file to create a job specification (and optionally a job specification file) for this specific job. If a job specification file is created, it is equivalent to the file you would create manually if you defined a job as described in "Job Creation: Directly Editing a Job Specification File" on page 140.

When a job specification file is not created (which is typically the case), the instantiator CGI performs the functional equivalent of writing a job specification file to disk and then invoking the `iwjobc` and `iwinvokejob` commands to instantiate and execute the job instance.

For an explanation of workflow template file structure and supported element syntax, see "Understanding Workflow Template File Structure" on page 144 and "Element Definitions" on page 180. For an example of a job specification file, see Appendix B, "Sample Job Specification File."

7. The job is instantiated into the server and started. These are actions you would execute manually (via `iwjobc` and `iwinvokejob`) if you defined a job as described in "Job Creation: Directly Editing a Job Specification File" on page 140.

The following sections provide more details about each diagram component.

### **Workflow Template File**

A workflow template file is an XML file that can contain any or all of the elements that are valid in a job specification file. These elements form the set of general workflow configuration instructions shown in the diagram on page 142. See "Understanding Workflow Template File Structure" on page 144 for details about these elements.

In addition, the workflow template file can contain `<template_script>` elements and a set of directives to define the workflow markups also shown in the diagram on page 142. All instructions residing within a `<template_script>` element are interpreted by the instantiator CGI as Perl code.

See “Understanding Workflow Template File Structure” on page 144 for details and a sample file illustrating these concepts.

### **Instantiator CGI**

TeamXpress provides a standard instantiator CGI, `iwwft_instantiator.cgi`, to perform the following tasks:

- Create and display the workflow information form based on information in the workflow template file
- Evaluate data entered by end users based on the workflow rules in the workflow template file
- Combine user-entered data with general workflow configuration instructions to create a job specification
- Instantiate the job specification on the TeamXpress server and start the job

### **Browser Interface (GUI)**

The browser-based GUI is a standard, TeamXpress supported GUI through which end users can enter data in a workflow form.

### **Job Specification File**

For an explanation of file structure and supported element syntax, see “Understanding Workflow Template File Structure” on page 144 and “Element Definitions” on page 180. See Appendix B, “Sample Job Specification File” for a sample job specification file.

### **Server-Side Workflow Subsystem**

The server-side workflow subsystem resides on the TeamXpress server and contains all the executable files that provide workflow functionality.

## **Understanding Workflow Template File Structure**

Workflow template files typically reside in `iw-home/local/config/wft` and (by convention) end with a `.wft` extension. See “Configuring Metadata Capture” for information about configuring the TeamXpress GUI to show lists of available templates.



A workflow template file may have the following components:

- `<template_script>` element(s) containing arbitrary Perl code.
- `CGI_info` directives to control the look and feel of workflow forms generated by the instantiator CGI.
- `TAG_info` directives to control the data collection, validation, and error messages displayed in workflow forms.
- `__ELEM__(tagname)`; directives to return the number of elements in a tag.
- `__TAG__(tagname)`; directives to insert the HTML-encoded data associated with the form POST/GET key `tagname` into the job specification.
- `__INSERT__(string)`; directives to insert text into a job specification programatically.
- `__VALUE__(tagname)`; directives to return unescaped POST/GET data associated with `$tagname`.
- Other elements identical to those used by job specification files.

The following section shows an excerpt from a basic sample workflow template file, followed by explanations of all file components (some of which are not included in the basic sample file). A second, more sophisticated sample file (“Sample Workflow Template File 2” on page 163) shows how to use more of these file components.

## Sample Workflow Template File 1

Workflow templating makes generalizing jobs very easy. The following is a fragment from a simple workflow template file that fills in the blanks (indicated by `__TAG__` directives) with HTML-encoded CGI data.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE workflow SYSTEM "iwwf.dtd">

<template_script><![CDATA[

    TAG_info(

        description  => "<input type='text' value='dark and stormy night'>",
        job_name     => "<input type='text' value='edit story'>",

    );

]]></template_script>

<workflow name="__TAG__('job_name');"
           owner="__TAG__('iw_areaowner');"
           description="__TAG__('description');">
... and so on...
```

Things to note in the preceding example:

- POST/GET data keynames appear on the left hand side of the arrow in the `TAG_info` directive.
- The HTML form input field that collects data for the template appears on the right hand side of the arrow in the `TAG_info` directive.
- The `TAG_info` directive appears within a `<template_script>` element.
- You can refer to POST/GET data that was not explicitly collected by the HTML form input fields you specified in `TAG_info`. For example, `iw_areaowner` is provided by default, so you do not need to give the template instantiator CGI an input field HTML snippet for `iw_areaname` within the `TAG_info` directive.

Suppose that, in the user interface for this form, you want the field that collects data for `job_name` to have a label of “Job Name” instead of “`job_name`”. The following template file section would accomplish that:

```
TAG_info(
    description => "<input type='text' value='dark and stormy night'>",
    job_name    => [ html => "<input type='text' value='edit story'>",
                    label => "Job Name",
                    ],
);
```

This example illustrates the `TAG_info` attributes `html` and `label`. There are many more, but all of them follow the same simple pattern:

```
[    ...someattribute... =>    ...avalue...,
    ...anotherattribute... =>  ...anothervalue...,
    ... and so on...
],
```

As described later in this chapter, the template writer can do far more sophisticated things than just filling in the blanks. For example, you can generate workflow dynamically, and/or intersperse dynamically generated workflow, data, and tags with hard-coded information. The following sections explain the details of workflow template file components and how you can use them to create workflow templates ranging from simple to elaborate. See “Sample Workflow Template File 2” on page 163 for another example of how to use many of these components.

## <template\_script> Element

A workflow template file can contain any number of <template\_script> elements. Each <template\_script> element contains arbitrary Perl code that can perform the following actions:

- Define the rules that the instantiator CGI employs to combine user-entered data with:
  - hardcoded workflow XML from the workflow template file.
  - programatically generated workflow XML produced within <template\_script> elements.
- Programatically generate a job specification (and/or intersperse hard-coded XML job specification information with programatically generated XML).
- Optionally send the job specification to a file in addition to, or instead of, instantiating it into the workflow subsystem. This can be helpful if you need to see exactly what XML is being produced by the workflow template file and instantiator CGI.
- Defines the rule that validate user-entered data.
- Define the custom error messages displayed when the template's data validation rules are not satisfied (see “TAG\_info Directive” on page 151).
- Inspect incoming POST/GET data, and (under some conditions) execute code on the basis of this data. For example, rules in a <template\_script> element can tell the instantiator CGI to generate different job specifications depending on what a user's name is. Thus, if the Author “Andre” needs three approvers for his work, but the Author “Jon” needs only one approver, you can define rules in a <template\_script> element to perform this job customization automatically based on whether Jon or Andre is the Author.
- Merge POST/GET data with the hard-coded workflow XML from the workflow template file.
- Determine the look and feel of the automatically generated workflow form that collects end-user data for a job (see “CGI\_info Directive” on page 150).

Because TeamXpress workflow template files must be valid XML documents, all content in a <template\_script> element must be declared as CDATA to protect it from interpretation by an XML parser. For example:

```
<template_script><![CDATA[  
  
    # arbitrary Perl code  
  
]]></template_script>
```

Together, all of the `<template_script>` elements in a workflow template file form a single Perl program. If a workflow template file contains more than one `<template_script>` element, all variables, functions, and libraries set in one element are available in the others. For example:

```
<...hard-coded workflow XML...>
```

```
<template_script><![CDATA[  
    use Lib1;    # you can import libraries  
  
    sub some_function    # you can define functions  
    {  
        return "Please enter beverage choice";  
    }  
  
    my $beverage = "tea";# you can define variables  
]]></template_script>
```

```
<...hard-coded workflow XML...>
```

```
<template_script><![CDATA[  
    # The variable $beverage is accessible in this  
    # section, and contains the value "tea".  
    # The function some_function() may also be called here.  
]]></template_script>
```

## CGI\_info Directive

### Usage:

```
CGI_info( ...list of key/value pairs... );
```

### Description:

Sets various defaults that effect the look and feel of workflow forms generated by the instantiator CGI. The CGI\_info directive may only appear within a <template\_script> element. Properties that you can set are as follows:

Property	Description
error_data_bgcolor	Data field background color displayed if an end user enters invalid data (validity is determined by the instantiator CGI). By default, all non-empty fields are valid, but you can customize this on a field-by-field basis. Color in this property and all other color properties shown in this table can be specified using standard HTML color syntax (e.g., "red", "green", "#FFFFFF", etc.).
error_label_bgcolor	Label field background color displayed if an end user enters invalid data in the data field.
error_text_color	Error message text color.
valid_bgcolor	Background color displayed if an end user enters valid data.
title	Browser window title.
html_body_options	Sets the options for the <body> element of the instantiator CGI. For general information about <body> elements, see <a href="http://www.w3.org/TR/REC-html40/struct/global.html#edef-BODY">http://www.w3.org/TR/REC-html40/struct/global.html#edef-BODY</a>
tag_table_options	Sets the options for the <table> element of the instantiator CGI. For general information about <table> elements, see <a href="http://www.w3.org/TR/REC-html40/struct/tables.html#h-11.2.1">http://www.w3.org/TR/REC-html40/struct/tables.html#h-11.2.1</a>
pre_tagtable_html	Defines what is displayed in a workflow form between the banner and tag table areas.
post_tagtable_html	Defines what is displayed in a workflow form after the tag table area.

**Note:** TeamXpress comes with a set of standard defaults to govern the look and feel of workflow forms.

**Example:**

```
CGI_info(  
  error_bgcolor    => "red",  
  valid_bgcolor   => "green",  
  title            => "TeamXpress Workflow Template",  
  html_body_options => "bgcolor='yellow'",  
  tag_table_options => "border=5 cellspacing=2 cellpadding=8",  
  
  pre_tagtable_html => "<h2>Whatever you want...</h2>",  
  post_tagtable_html => "...this appears after the tagtable...",  
);
```

## TAG\_info Directive

**Usage:**

```
TAG_info(list of key/value pairs);
```

**Description:**

Establishes a relationship between a list of tag names and the information the instantiator CGI uses to collect data for them. There are two ways to build these associations:

**Style 1 (simple):**

```
tagname => "...html that collects data for tagname...";
```

**Style 2 (highly flexible):**

```
tagname => [ html          => "...html that collects data for tagname...",  
             is_required => "true",  
             valid_input => "...Perl expression...",  
             label       => "...html label...",  
             error_msg   => "...html error message...",  
           ];
```

When the instantiator CGI processes the `TAG_info` directive, the `name` attribute in the resulting HTML code is automatically set to `tagname`. For example, given the following `TAG_info` directive:

```
TAG_info(
    beverage => "<input type='text' value='tea'>",
);
```

The internal representation of the resulting HTML code is:

```
"<input type='text' name='beverage' value='tea'>"
```

Because this is done automatically, it is impossible for the tag names to get out of sync with the resulting HTML code. For example, if you attempted to explicitly set the `name` attribute to something other than `tagname`:

```
TAG_info(
    beverage => "<input type='text' name='drink' value='tea'>",
);
```

then `name='drink'` gets stripped out and automatically replaced by `name='beverage'`.

The `TAG_info` directive may appear only within a `<template_script>` element. While it is legal to have any number of `TAG_info` directives in a workflow template file, it is often convenient to consolidate all necessary data into one `TAG_info` directive.

Properties that you can set for each tag in a `TAG_info` directive are as follows:

Property	Description
<code>html<sup>1</sup></code>	Valid HTML input form field (which optionally may contain a default value).
<code>is_required</code>	Whether end-user input in the tag is required. Defaults to <code>true</code> if not set.
<code>valid_input</code>	Rules to check input validity. Default is to check for an empty string if not set.



Property	Description
<code>error_msg</code>	An HTML message returned if end-user input is invalid. Default message is <code>Valid input required</code> if not set.
<code>label</code>	The HTML label displayed beside the HTML input field for this tag. Defaults to the value of <code>tagname</code> if not set.

1. Required if using Style 2.

### Array Validators

When validating input in a `valid_input` expression, both `$_` and `@_` are set appropriately. Therefore, when collecting information in a multi-select list, a tag's validator can be implemented as follows:

```

TAG_info(
    a_tag_name => [ html => "html that collects data for a_tag_name...",
                  valid_input => 'a_tag_validator(@_)',
                  ]
)
    
```

### Example:

The following example shows definitions for three tags (named `food`, `beverage`, and `music`). Each tag can be used any number of times by the instantiator CGI to prompt for and collect end-user input in a workflow form.

The definition for tag `food` specifies that the HTML element used to collect data for this CGI variable is a text field.

Tag `beverage` has the following characteristics:

- It will accept only text input.
- It automatically displays a default value of `Beverage: tea` in its entry field.
- A value in its entry field, either end-user input or the default, is required.
- The label `Enter beverage choice` is displayed beside the text field that collects user input.

- `valid_input` specifies that all data entered by an end user must begin with the string `Beverage:`.
- `error_msg` specifies the error message to be displayed if end-user input does not begin with `Beverage:`.

Tag `music` displays a default value of `Classical`.

```
TAG_info(  
  
  food => "<input type='text'>",  
  
  beverage => [ html      => "<input type='text' value='Beverage: tea'>",  
                 is_required=> 'true',  
                 label      => '<strong>Enter beverage choice</strong>',  
                 valid_input=> '/^Beverage:/',  
                 error_msg  => '<br><strong>'.  
                               'ERROR: input must begin with "Beverage:"'.  
                               '</strong>',  
                 ],  
  
  music => "<input type='text' value='Classical'>",  
);
```

## \_\_ELEM\_\_ Directive

### Usage:

```
__ELEM__($tagname);
```

### Description:

Returns the number of data elements associated with tag `tagname`. If `tagname` is undefined, 0 is returned. The `__ELEM__` directive may appear inside and/or outside of a `<template_script>` element. You can also embed an `__ELEM__` directive within an `__INSERT__` directive. A workflow template file can contain any number of `__ELEM__` directives.

**Example:**

The following `TAG_info` directive defines the tag `reviewers` to accept multiple selections. Thus, this one tag can have multiple values. By default, two reviewers (Dick and Harry) have been selected. If an end user accepts these default values, `__ELEM__('reviewers');` will yield 2. If an end user also selects Tom as a reviewer, `__ELEM__('reviewers');` will yield 3.

```
TAG_info(
    reviewers => [ html => "<select multiple>"
                  "      <option>Tom"
                  "      <option selected>Dick"
                  "      <option selected>Harry"
                  "</select>",
                  label => "Pick reviewers",
    ],
);
```

**\_\_TAG\_\_ Directive****Usage:**

```
__TAG__($tagname);
```

**Description:**

When the instantiator CGI creates a job specification, it uses each `__TAG__` directive in the workflow template file as an insertion point for the HTML-encoded value associated with the form input key `tagname`. Thus, user input from any tag can be inserted at any point in a job specification file in HTML-encoded form.

In addition, the `__TAG__` directives can mention form input key names that are not defined in `TAG_info` as long as the POST/GET data is provided for these keys programatically. The following POST/GET keys are always passed, and are therefore always available for use in a workflow template file or job specification file. The set of passed tags differs depending on how the job is started.

If started via **Submit**:

Key Name	Description
<code>iw_areaowner</code>	The owner of the workarea.
<code>iw_branch</code>	The branch's vpath (i.e., <code>/default/main/dev/subbranch1</code> ).
<code>iw_home</code>	The <code>iw-home</code> directory.
<code>iw_role</code>	The user's role.
<code>iw_session</code>	The session string.
<code>iw_template_file</code>	The template file's path and name relative to <code>iw-home/local/config/wft</code> .
<code>iw_template_name</code>	The template name to be displayed in TeamXpress GUI.
<code>iw_use_default</code>	Use the default argument of the template (defaults to <code>true</code> ).
<code>iw_user</code>	The user's name.
<code>iw_workarea</code>	The workarea's vpath (i.e., <code>/default/main/dev/WORKAREA/user1</code> ).

If started via **New Job**:

Key Name	Description
<code>iw_home</code>	The <code>iw-home</code> directory.
<code>iw_role</code>	The user's role.
<code>iw_session</code>	The session string.
<code>iw_template_file</code>	The template file's path and name relative to <code>iw-home/local/config/wft</code> .
<code>iw_template_name</code>	The template name to be displayed in TeamXpress GUI.
<code>iw_use_default</code>	Use the default argument of the template (defaults to <code>true</code> ).
<code>iw_user</code>	The user's name.

In addition, the `iw_overwrite` POST/GET key makes the status of the Overwrite button in the TeamXpress GUI available to the workflow subsystem. For example, if Overwrite is selected, an `iw_overwrite` value of `true` is passed as POST/GET data to the instantiator CGI, making it available for use in a job specification. If Overwrite is not selected, the value of `iw_overwrite` is `false`.

Additional POST/GET keys could also be available, depending on the job's configuration. To display a list of all POST/GET keys available in a specific job, run `show_env.cgi` by naming it in the job's `<cgitask>` element. For example:

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE workflow SYSTEM "iwwf.dtd">

  <workflow name="minimal"
    owner="jon" creator="jon"
    description="This is a minimal example of a CGI task">

    <cgitask name="cgi" start="t" owner="chris"
      description="First CGI task" immediate="t">

      <areavpath v="/default/main/dev/wfregr2/WORKAREA/chris"/>

      <successors>
        <successorset description="Success">
          <succ v="end"/>
        </successorset>
      </successors>

      <command v="show_env.cgi"/>

    </cgitask>

    <endtask name="end"/>

  </workflow>
```

The `__TAG__` directive may appear inside and/or outside of a `<template_script>` element. You can also embed a `__TAG__` directive within an `__INSERT__` directive. A workflow template file can contain any number of `__TAG__` directives. To determine how many elements a tag contains, refer to the `__ELEM__` directive. See “Using Variables in Strings” on page 161 for details about the syntax of variables used within the `__TAG__` directive.

### Examples:

If your workflow template file contained the following text:

```
I wish I had a __TAG__('beverage');
```

and the instantiator CGI POST/GET data for tag `beverage` was `cup of tea`, the job specification would contain:

```
I wish I had a cup of tea
```

Similarly, if `beverage` were an array tag (e.g., multi-select, checkbox, etc.), and `2` were a valid index, the following would be a valid entry in the workflow template file:

```
I wish I had a __TAG__('beverage[2]');
```

In this case, the third element from tag `beverage` would be inserted by `__TAG__`. The third element is chosen because arrays start at element 0.

## `__INSERT__` Directive

### Usage:

```
__INSERT__($string);
```

### Description:

Inserts the value of the variable (or hard coded string) `$string` into the workflow template file, where `$string` can be any arbitrary text (typically, workflow XML). `$string` can optionally include embedded tags (via the `__TAG__` directive) and/or elements (via the `__ELEM__` directive). Embedding tags within an `__INSERT__` directive is especially useful when the template’s output needs to be generated dynamically. See “Using Variables in Strings” on page 161 for details about the syntax of variables used within the `__INSERT__` directive.

**Example:**

The following example shows the portion of a workflow template file that sequentially inserts the values of tags `a`, `b`, and `c` into the job specification file.

```
<template_script><![CDATA[

my $i;
my @tag_array = ('a','b','c');
for ($i=0; $i<3; ++$i)
{
    __INSERT__("I am __TAG__($tag_array[$i]); pleased!\n");
}

]]></template_script>
```

Note that an `__INSERT__` directive can also process complex expressions both inside and outside of a `<template_script>` element (e.g., it can process quoted fragments containing nested `__TAG__(...);` directives, possibly joined by `'.'` etc.).

**\_\_VALUE\_\_ Directive****Usage:**

```
__VALUE__($tagname);

__VALUE__($tagname,$encoding);
```

**Description:**

By default, returns the unescaped POST/GET data associated with tag `$tagname`, but unlike `__TAG__($tagname)`, it does not insert anything into the job specification when the instantiator CGI processes the workflow template file. If the value of the optional parameter `$encoding` is set to `html`, the HTML-encoded version of the data is returned instead of the raw value.

This is useful when the template's output needs to be generated dynamically based on the POST/GET values the instantiator CGI receives. The values returned by the variable `$tagname` are as follows:

- If `$tagname` does not refer to a defined POST/GET key name, `undef` is returned.
- If `$tagname` is a scalar POST/GET key name, a scalar is returned.

- If *\$tagname* is an array POST/GET key name, a list is returned.
- If *\$tagname* is a subscripted array POST/GET key name, a scalar is returned.

**The `__VALUE__` directive may only appear within a `<template_script>` element. In a `__VALUE__($tagname);` directive, if the tagname is a subscripted array, the subscript can be an expression.**

#### Example:

The following example uses `__VALUE__` of tag `x` to set the upper limit of `$i`. This example assumes that the form input key name `x` contains an integer.

```
<template_script><![CDATA[

for (my $i=0; $i < __VALUE__("x"); ++$i)
{
    __INSERT__("very nice $i\n");
}

# Advanced users:  if x were an array tag (CGI form input keyname),
# then it could be subscripted as follows, assuming 2 is a valid
# array index (cf: __ELEM__):
#
#     for (my $i=0; $i < __VALUE__("x[2]"); ++$i)
#
]]></template_script>
```

## Other Elements

A workflow template file can also contain any element that is legal in a job specification file. These elements, described in “Element Definitions” on page 180, make up the set of general workflow configuration instructions shown in the workflow template file box in the diagram on page 142.



## Using Variables in Strings

The following scenarios describe syntax rules that apply to variables in strings used by `__TAG__` and `__INSERT__` directives. The scenarios start with the simplest method of variable substitution and end with the most advanced.

### Scenario 1: Basic Variable Usage

When inside a quoted string, the argument for a `__TAG__` directive does not need any kind of quoting at all.

For example, assuming you have a POST argument named `color1`, you can just say:

```
__INSERT__("shirtcolor='__TAG__(color1);' accepted!!");
```

Other valid usage examples are:

```
__INSERT__("... __TAG__('color1'); ...");
__INSERT__('... __TAG__("color1"); ...');
__INSERT__("... __TAG__(color1); ...");
__INSERT__('... __TAG__(color1); ...');
__INSERT__("... '__TAG__(color1);' ...");
__INSERT__('... "__TAG__(color1);" ...');
__TAG__("color1");
__TAG__('color1');
```

The following is **not** valid because the argument `color1` is not quoted in any way, and `__TAG__` is not nested within an `__INSERT__` directive:

```
__TAG__(color1);
```

### Scenario 2: Including Quotation Marks in Insertions

Continuing with the preceding example and adding the following information:

- you have a Perl variable named `$var1` whose value is `workarea`
- a POST input key named `workarea` whose value is `jon`

then the following statements all insert the string `jon` into the job:

```
__INSERT__ ("... __TAG__($var1); ...");  
__INSERT__ ("... __TAG__(' $var1 '); ...");  
__TAG__($var1);
```

The following expression inserts the string `'jon'` into the job:

```
__INSERT__ ("... ' __TAG__($var1);' ...");
```

Hence, to insert a tag into a job within single quotes you could say:

```
__INSERT__ ("var1=' __TAG__(color1);' accepted!!");
```

And to insert a tag into a job within double quotes, you could say:

```
__INSERT__ ('var1=" __TAG__(color1);" accepted!!');
```

### Scenario 3: Preferred Ordering of Single and Double Quotes

If you specify either of the following:

```
__INSERT__ (' __TAG__("$var1");');  
__INSERT__ (' __TAG__($var1);');
```

you will probably get an error message about not finding data for the FORM input keyname `$var1` because the outer-most quotation marks on the `__INSERT__` directive are single quotes. In Perl, single quotes are interpreted as:

“Do not interpolate anything in this string as a Perl variable.”

Hence `$var1` is literally the set of characters `$, v, a, r, 1` (and not a variable named `$var1` whose value is `workarea`).

In general you should place the double quotes outside and the single quotes inside:

```
__INSERT__("var1='__TAG__(color1);' accepted!!");
```

For example:

```
<template_script><![CDATA[
    my $status = "not in stock.";
    __INSERT__("var1='__TAG__(color1);' currently $status");
]]></template_script>
```

### Sample Workflow Template File 2

The following workflow template file is more elaborate than the sample file shown in “Sample Workflow Template File 1” on page 146; it shows the use of several additional file components as explained in the preceding sections. Specifically, this file:

- Generates a form that captures the deployment date for this job.
- Ensures that the “Timed Deployment” field does not allow a user to just enter “later”.
- Sets the label in the form that collects data for the deploy date to “Timed Deployment”.
- Sets the owner attribute for the XML element <workflow> to the HTML-encoded data associated with the form input key iw\_areaowner (and similar operations for the other \_\_TAG\_\_ directives).
- For each file that has been selected in the TeamXpress GUI, it inserts a line that reads:

```
<file path='...filename...' comment='File to time deploy'/>
```

**Note:** When the job specification is generated, these lines appear between the XML tags <files> and </files>.



```

<?xml version="1.0" standalone="no"?>
<!DOCTYPE workflow SYSTEM "iwwf.dtd">

<template_script><![CDATA[

    TAG_info(
        deploy_date => [ html          => "<input type='text' value=''>",
                        valid_input => '$_ ne "later"',
                        label        => "Timed Deployment",
                        ],
    );

]]></template_script>

<workflow name="TimedDeploy" owner="__TAG__('iw_areaowner');"
    creator="__TAG__('iw_areaowner');"
    description="Timed Deployment">

    <usertask name="AuthorWork" owner="__TAG__('iw_areaowner');"
        description="Editing files to time deploy" start="t">

        <areavpath v="__TAG__('iw_workarea');"/>
        <successors>
        <successorset description="One Minute">
        <succ v="Submit"/>
        </successorset>
        </successors>
        <files>
        <template_script><![CDATA[

            for (my $i=0; $i < __ELEM__('iw_file'); ++$i)
            {
                __INSERT__("<file path='__TAG__(iw_file[$i]);' comment='File to
                    time deploy'>\n");
            }

        ]]></template_script>
        </files>
    </usertask>

    <submittask name="Submit" owner="__TAG__('iw_areaowner');"
        description="Staging for deployment.">
        <areavpath v="__TAG__('iw_workarea');"/>
    ...

```

## Debugging Workflow Templates and Job Specification Files

Two additional POST/GET keys are available for debugging workflow template and job specification files. Details are as follows.

### **iw\_debug\_mode**

The `iw_debug_mode` key instructs the instantiator CGI to process input data from a submitted form as it normally would, and then display job-specific information in a Debug Mode page rather than instantiate the job on the server. The Debug Mode form always contains two default sections: the Perl code (including line numbers) that generates the job specification, and the XML job specification itself. This job specification is what would have been instantiated on the server if debug mode had been turned off. A third section showing syntax errors appears in the Debug Mode form if the instantiator CGI found errors in the Perl code it generated based on form input.

### **iw\_output\_file**

The `iw_output_file` key instructs the instantiator CGI to process input data from a submitted form as it normally would, and then capture the output in an XML job specification file rather than instantiate the job on the server. After it is created, you can manually instantiate the job specification file on the server at any time via `iwjobc`.

### **Usage**

You can define the `iw_debug_mode` and `iw_output_file` key names in a `TAG_info` directive (causing the keys to appear in the workflow form), or you can provide definitions programmatically via POST/GET data.

### **Example**

The following example shows definitions that are set within a `TAG_info` directive:

```
TAG_info(  
    iw_debug_mode => "<input type='text' value='true'>",  
    iw_output_file => "<input type='text' value='/tmp/my_job.xml'>",  
);
```

Items to note in the preceding example:

- For either element, when you set `type` to `text`, the element appears on the workflow form. Setting `type` to `hidden` causes the element not to be displayed on the workflow form.
- You can toggle debug mode on or off by setting `value` to `true` or `false`.

The file named in `value` must be writable by `httpd`.

## Workflow Log File

Output from workflow runtime diagnostics is logged in `/var/adm/iwjoberrors.log`.

## Adding Template Selection Choices to TeamXpress GUI Menus

By default, workflow templates reside in `iw-home/local/config/wft`. This directory also contains a file, `available_templates.ipl`, which consists of a single Perl function that lets administrators configure which templates appear on any user's template choice menu in the TeamXpress GUI. You can modify this configuration function to be as simple or as sophisticated as necessary. For example, the function can contain just a hard-coded list of files that will appear on a user's template choice menu. Or, a more elaborate configuration might include actions like fetching template list information from a relational database, or scanning a directory or flat file.

The `available_templates.ipl` file can include rules that return different lists of available templates depending on the current user, role, `vpath`, and `command`. This design allows TeamXpress's workflow template mechanism to handle dynamic business rules and scale to systems that have a very large number of workflow processes.

The following sections describe three configuration scenarios, starting with the simplest and ending with more advanced selection criteria.

### Scenario 1: Hard-Coded Template Names

The following example shows an `available_templates.ipl` file containing a hard-coded list of templates that will appear on an end-user's template choice menu in the TeamXpress GUI. See the "Diagram Key" section following the file for details about referenced items.

```

#-----
# File   available_templates.ipl
#
# Use    An administrator-configurable Perl function that returns a list of
#        available workflow template files given a user, role, vpath, command
#        (and possibly post_command).
#-----
#-----
sub available_templates
{
    my ($user,$role,$vpath,$command,$post_command) = @_;
    if ($command eq "submit"&&    $vpath =~ m[default/main/dev/sales]
    {
        return
        [
            "demo.wft",           # note: relative paths are rooted in
            "demo2.wft",         # iw-home/local/config/wft/
            "examples/demo3.wft",
            "/ray/size/demo4.wft", # absolute paths are ok too
        ];
    }
    elsif ($command eq "edit")
    {
        return
        [
            "demo.wft",
            "demo5.wft",
        ];
    }
    else { fail("$0 cannot get templates for this command\n");}
}
#-----
#-----
1;    # <-- Do not delete this line.

```

Diagram Key:

- ← Values available as selection
- Chosen selection criteria.<sup>2</sup>
- List of templates displayed. (points to the array of template names)
- Failure (points to the `fail` statement)

## Diagram Key

1. The values of `$user`, `$role`, `$vpath`, `$command`, and `$post_command` can be used by `available_templates.ipl` as criteria for selecting the circumstances under which the hard-coded list of templates will be displayed on the template choice menu.
2. This line names the specific selection criteria and their corresponding values. This example specifies that the templates shown in Item 3 are to be displayed when the value of `$command` is "submit" and the value of `$vpath` is "default/main/dev/sales".
3. The templates shown here are displayed in the template choice menu if the value of `$command` is "submit" and the value of `$vpath` is "default/main/dev/sales". If a workflow template file resides in `iw-home/local/config/wft`, you can specify just the file name (as shown by `demo.wft` and `demo2.wft`). If a workflow template file resides in a different directory, you can name it using either a relative path rooted in `iw-home/local/config/wft` (as shown by `examples/demo3.wft`) or an absolute path (as shown by `ray/size/demo4.wft`).

Note that TeamXpress converts hard-coded names as follows:

- The `.wft` extension is removed.
- The `_` character is replaced by a space.
- Each space-delimited token has the first letter capitalized.

For example, the name `Author Assignment` appears on the list of available templates if `available_templates.ipl` is coded as follows:

```
return
[
"default/author_assignment.wft",
];
```

4. This section specifies that the templates `demo.wft` and `demo5.wft` be displayed on the template choice menu if the value of `$command` is "edit".
5. This message is displayed if the value of `$command` is not "submit" or "edit".



## Scenario 2: Hard-Coded File Lists with Explicit Display Names

You can also hard-code a list of workflow template files, but specify a display name for the template choice menu that differs from the base file name. For example, the configuration file could be as follows. See the “Diagram Key” section following the file for details about referenced items.

```

#-----
sub available_templates
{
    my ($user,$role,$vpath,$command,$post_command) = @_ ;

    if ($command eq "submit")
    {
        return
        [
            "demo.wft",           ← Template file names1
            "demo2.wft",        ←
            [ file => "examples/minimal_cgi.wft",   ← Template file name2
              name => 'CGI Task Demo'             ← Display name3
            ],
            [ file => "examples/author.wft",       ← Template file name2
              name => 'Author Demo'               ← Display name3
            ],
        ];
    }
    elsif ($command eq "edit")
    {
        return
        [
            [ file => "examples/author.wft",       ← Template file name2
              name => 'The Same Old Author Demo' ← Display name3
            ],
        ] ;
    }
    else { fail("$0 does not know how to get templates for this command\n");}
}
#-----
#-----
!; # <-- Do not delete this line.

```

## Diagram Key

1. These template names appear as is on the template choice menu as described in Scenario 1, Item 2.
2. The template files named by `file` appear on the template choice menu as defined by `name`. For example, the template `examples/minimal_cgi.wft` appears on an end user's template choice menu as `CGI Task Demo`.

## Scenario 3: Rules-Based Template Selection

Advanced configuration methods involve setting up `available.templates.ipl` to return a programmatically generated list of which templates are available under what circumstances. For example, the list of template files might be obtained by scanning a directory, reading from a flat file, querying a database, etc. Modifications to `available.templates.ipl` must ensure that the list of templates to display is formatted as described earlier in Scenarios 1 and 2. The file format is as follows:

```
#-----  
sub available_templates  
{  
    my ($user,$role,$vpath,$command,$post_command) = @_;  
  
    # grab the list from a database, or perhaps by reading a config file  
    # or perhaps looking at all the files in a particular directory, etc.  
  
    if ( $user eq "jon" && $command eq "submit")  
    {  
        return ...a list of templates...  
    }  
    elsif (...other constraint...) {return ...a list of templates...}  
    elsif (...other constraint...) {return ...a list of templates...}  
    else { fail("$0 cannot get templates for this command\n");}  
}  
#-----  
#-----  
1; # <-- Do not delete this line.
```

In this example, *...a list of templates...* can use any return syntax shown in Scenarios 1 and 2. The value of *...other constraint...* can use any `if` or `elsif` syntax shown in Scenarios 1 and 2.

## Using Pre-Packaged TeamXpress Workflow Templates

TeamXpress comes with pre-packaged workflow templates for several common jobs. Some of these templates are available by default from the TeamXpress GUI, while others require that you configure TeamXpress to make them available. The following sections describe where each template resides, the job that is configured by each template, and how to make each template available through the TeamXpress GUI.

### Template Locations

Pre-packaged workflow templates reside in two directories:

- `iw-home/local/config/wft/default`
- `iw-home/local/config/wft/examples`

Following a TeamXpress installation, templates in the `default` directory automatically appear as choices in the New Job window when a user selects **File > New Job** in the TeamXpress GUI. Templates in the `examples` directory require that you configure workflow to make them available from the TeamXpress GUI. After you make them available, they appear together with the default templates in the New Job window. The following sections describe the default and example templates in detail.

## Default Templates

The following templates reside in `iw-home/local/config/wft/default`:

Template Name	Description
<code>author_assignment.wft</code>	Lets an Editor, Administrator, or Master assign a job to an Author. The assigner selects an author and enters a task description. The assigner also selects a branch and workarea if the job is initiated from the To Do List view. An approval sequence is also included for the author assignment.
<code>author_submit_dcr.wft</code>	Submits a data content record (DCR) to the staging area when an Author clicks <b>Save and Exit</b> in TeamXpress Templating Data Content Record window. This automates the submission process, eliminating the need for the Author to initiate the submission manually after creating or editing a DCR. Requires the presence of TeamXpress Templating.
<code>default_submit.wft</code>	Performs the same actions as the <b>Submit Direct</b> button, plus provides support for pre-submit activities such as approval, file type recognition, user-specific destinations, etc.
<code>dual_work_order.wft</code>	Configures a job that lets an Editor, Administrator, or Master assign to another user the task of setting up the job defined by the Work Order template. Requires additional configuration as described in “Configuring the Dual Work Order Template” on page 176.
<code>work_order.wft</code>	Configures a template that lets an Editor, Administrator, or Master define work assignments and approval chains for a job of any length. Requires additional configuration as described in “Configuring the Work Order Template” on page 173.

By default, these workflow templates are referenced in the `available_templates.ipl` file as described in “Adding Template Selection Choices to TeamXpress GUI Menus” on page 166. Therefore, all are available via the **File > New Job** command following a TeamXpress installation.

Note that the `work_order` and `dual_work_order` templates are highly configurable. It is recommended that you configure TeamXpress as described in the following sections before allowing end users to run the Work Order and Dual Work Order jobs.

### Configuring the Work Order Template

The Work Order template configures a job that lets an Editor, Administrator, or Master define work assignments and approval chains for a job of any length. The job can contain:

- Single or multiple contributors.
- Single or multiple approvers.
- Serial contributors and approvers.
- Concurrent contributors and approvers.
- Metadata assignments.
- Task attribute assignments (whether files are locked, read-only, etc.).
- Recursion (jobs that run again at specified intervals of time).
- Archiving (saving the job for a specified amount of time following its completion).

After you perform the configuration tasks described in this section, the Work Order job is ready for use by job creators. A job creator running the job can then fill in the Work Order template via the TeamXpress GUI to control the following parameters for the current instance of the job:

- Which users will be content contributors.
- Which users will be content approvers.
- Which users will receive email notification when a task is done.
- Whether contributors and approvers will work serially or concurrently.
- Whether files are locked.
- Whether files are read-only.
- Whether contributors can add metadata to a file.
- How many days will elapse before the job runs again.
- How long the job will be saved after it is completed.

To configure the Work Order template, edit

`iw-home/local/config/wft/default/work_order.wft` as follows:

1. Set the `$number_of_contributors` variable as described in the file's comments to specify how many times the contributor field appears in the Work Order form. The default value is 2.
2. Set the `$number_of_approvers` variable as described in the file's comments to specify how many times the approver field appears in the Work Order form. The default value is 2.
3. Set the `@possible_contributors` variable as described in the file's comments to specify which TeamXpress role(s) will be used as the basis for the drop-down list of possible contributors. The default roles are Author and Editor.
4. Set the `@possible_approvers` variable as described in the file's comments to specify which TeamXpress role(s) will be used as the basis for the drop-down list of possible approvers. The default role is Master.
5. Set the `$skip_metadata` variable as described in the file's comments to specify whether the Work Order form will contain a metadata field. The default value is `FALSE`, which creates a metadata field in the Work Order form. If a metadata field exists in the form, the person using the form to set up the job instance can specify whether content contributors will be prompted to set metadata for a file.
6. Set the `$skip_email` variable as described in the file's comments to specify whether the Work Order form will contain a field for email addresses. The default value is `FALSE`, which creates an email address field in the Work Order form. If an email address field exists in the form, the person using the form to set up the job instance can specify who receives email notification upon task assignment.
7. Set the `$skip_save_job` variable as described in the file's comments to specify whether and for how long the job is saved following completion. The default value is `FALSE`, which creates a save job field in the Work Order form. If a save job field exists in the form, the person using the form to set up the job instance can specify the duration (in days) that the job is saved.
8. Set the `$skip_recurring` variable as described in the file's comments to specify whether and how often the job reoccurs. The default value is `FALSE`, which creates a job recursion field in the Work Order form. If a job recursion field exists in the form, the person using the form to set up the job instance can specify how many days elapse before this job runs again.

9. Set the `$skip_branch` variable as described in the file's comments to specify whether the person filling in the Work Order form is prompted for branch and path information. The default is `FALSE`.

10. You can optionally set default values for any of the following variables:

Variable	Current Default Value
<code>\$workflow_name</code>	Ordered Change Request
<code>\$contributor_default</code>	No default.
<code>\$task_desc_default</code>	Do this.
<code>\$contrib_email_default</code>	No default.
<code>\$contrib_perform_tasks_default</code>	serial
<code>\$metadata_default</code>	no
<code>\$contrib_lock_default</code>	yes
<code>\$approver_default</code>	No default.
<code>\$approver_email_default</code>	No default.
<code>\$approver_perform_tasks_default</code>	serial
<code>\$approver_lock_default</code>	no
<code>\$approver_read_only_default</code>	yes
<code>\$workarea_path_default</code>	No default.
<code>\$recurring_days</code>	No default.
<code>\$numrows_default</code>	1

A line already exists for each variable's default in `work_order.wft`. To set no default for a variable, set its value to "" (an empty string). Do not comment out any of the variables. See the comments in `work_order.wft` for more information.

### Configuring the Dual Work Order Template

The Dual Work Order template configures a job that lets an Editor, Administrator, or Master assign to another user the task of setting up the job defined by the Work Order template. For example, a second-level manager could use the Dual Work Order template to delegate a job's setup to a first-level manager. The second-level manager would fill in the initial Dual Work Order form, stating which first-level manager should complete the job setup. When the second-level manager starts the job, the first-level manager receives the job setup assignment and the Dual Work Order template automatically starts the Work Order job as an external task for the first-level manager to complete. At completion of the entire job, the second-level manager can approve the entire job before it is submitted.

To configure the Dual Work Order template, edit

`iw-home/local/config/wft/default/dual_work_order.wft` as follows:

1. Set the `$areavpath` variable so that it specifies a path to any workarea on the system. This step is necessary so that the CGI task that runs as part of the Dual Work Order job is directed to TeamXpress. Therefore, you can specify any valid TeamXpress workarea when setting `$areavpath`.
2. Set the `$number_of_contributors` variable as described in the file's comments to specify how many times the contributor field appears in the Dual Work Order form. The default value is 2.
3. Set the `@possible_contributors` variable as described in the file's comments to specify which TeamXpress role(s) will be used as the basis for the drop-down list of possible contributors. The default roles are Author and Editor.



4. You can optionally set default values for any of the following variables:

Variable	Current Default Value
<code>\$workflow_name</code>	Coordinated Change Request
<code>\$contributor_default</code>	No default.
<code>\$task_desc_default</code>	Do this.
<code>\$contrib_email_default</code>	No default.
<code>\$self_approve_default</code>	yes
<code>\$self_email_default</code>	No default.
<code>\$numrows_default</code>	1

A line already exists for each variable’s default in `dual_work_order.wft`. To set no default for a variable, set its value to " " (an empty string). Do not comment out any of the variables. See the comments in `dual_work_order.wft` for more information.

5. Because the Dual Work Order job runs the Work Order job as an external task, you must also make sure that `work_order.wft` is configured before running Dual Work Order.

### Example Templates

The templates residing in `iw-home/local/config/wft/examples` are included as reference examples that are applicable to some installations. The functionality provided by these examples is included in a more generalized form in the `work_order.wft` template. The templates in the `examples` directory are provided as shorter, more modular examples of how you can develop custom workflow templates. To make the example templates available to end users, you must edit `available_templates.ipl` as described in “Adding Example Templates to the TeamXpress GUI” on page 179.

The following templates reside in `iw-home/local/config/wft/examples`:

Template Name	Description
<code>author_assignment_with_email.wft</code>	Lets Editors, Administrators, and Masters assign a job to an Author, and then notifies the Author via email.
<code>author_submit.wft</code>	Lets an Author submit files to a staging area.
<code>concurrent_approval.wft</code>	Lets Editors, Administrators, and Masters assign a task to a content contributor and specify a single user or group as the approver.
<code>concurrent_approval_with_email_with_metadata.wft</code>	Performs the same activities as Concurrent Approval, plus allows the content contributor to set metadata for a file, and sends email to the approver(s).
<code>concurrent_approval_with_metadata.wft</code>	Performs the same activities as Concurrent Approval, plus allows the content contributor to set metadata for a file.
<code>serial_approval.wft</code>	Lets Editors, Administrators, and Masters assign a task to a content contributor and specify one or more users or groups as the approvers.
<code>serial_approval_with_email_with_metadata.wft</code>	Performs the same activities as Serial Approval, plus allows the content contributor to set metadata for a file, and sends email to the approver(s).
<code>serial_approval_with_metadata.wft</code>	Performs the same activities as Serial Approval, plus allows the content contributor to set metadata for a file.

It is recommend that you examine each `.wft` file for details about its construction and the features of the job it configures. After examining each file, you can choose to use it as is, or modify it for your specific installation using the information from “Understanding Workflow Template File Structure” on page 144.

## Adding Example Templates to the TeamXpress GUI

Each template in `iw-home/local/config/wft/examples` is listed as a commented entry in `available_templates.ipl`. For example:

```
elseif ($command eq "new_job")
{
    return
    [

#         "examples/author_assignment_with_email.wft",
#         "examples/author_submit.wft",
#         "examples/concurrent_approval.wft",
#         "examples/concurrent_approval_with_email_with_metadata.wft",
#         "examples/concurrent_approval_with_metadata.wft",
#         "examples/serial_approval.wft",
#         "examples/serial_approval_with_email_with_metadata.wft",
#         "examples/serial_approval_with_metadata.wft",
```

To add a template to the TeamXpress GUI, uncomment the line for that template.

A duplicate of the original `available_templates.ipl` file is included in `iw-home/local/config/wft`. You can use `available_templates.example` if you need to revert to the configuration originally contained in `available_templates.ipl`.

## Understanding Job Specification File Structure

A job specification file describes a single job. It is structured as a hierarchy of sections, each containing an element definition that lets you control a job parameter. An initial `<workflow>` section defines the overall characteristics of the job. It is followed by one or more task sections describing specific tasks that occur as part of the job. The following list shows all of the possible elements that can define sections in a job specification file. Indentation shows nesting levels:

```
workflow
  usertask
  updtatetask
  submittask
  externaltask
  endtask
  grouptask
  cgitask
  dummytask
  locktask
```

All of these elements, their attributes, and their subelements are described in the following section. See Appendix B, “Sample Job Specification File” for examples of files that use these elements.

### Element Definitions

The following DTD excerpts describe the syntax for each job specification file element. These elements are also valid in a workflow template file.

**Note:** Subelements within an element must be ordered as shown in the DTD. See `iw-home/local/config/wft/iwwf.dtd` for the complete workflow DTD.

### <workflow> Element

A <workflow> element defines a job's name and owner.

```
<!ELEMENT workflow (description?, variables?,
(usertask|submittask|updatetask|externaltask|cgitask|endtask|grouptask|
dummytask|locktask)+)>
  <!ATTLIST workflow name ID #REQUIRED
                    owner CDATA #REQUIRED
                    creator CDATA #REQUIRED
                    description CDATA #IMPLIED>
```

#### Attributes:

name	Name of the job. Job names are not unique identifiers. However, each job that is instantiated is identified by a unique ID number.
owner	The owner responsible for the job (defined in workflow template file rules).
creator	The user who started the job via the TeamXpress GUI's workflow form.
description	A description of what the job does. Can be specified as both an attribute and a subelement of <workflow>.

#### Subelements:

<description>

A description of what the job does. Can be specified as both an attribute and a subelement of <workflow>. Syntax is as follows:

```
<!ELEMENT description (#PCDATA)>
```

<variables>

Workflow variables are key-value pairs that can be stored in and retrieved from job instances. They are used to allow separate CGI tasks and external tasks to communicate with each other during job execution. Workflow variables are manipulated using the `iwjobvariable` CLT or by specifying them at job creation time. Syntax is as follows:

```
<!ELEMENT variables (variable+)>
  <!ELEMENT variable EMPTY>
    <!ATTLIST variable key NMTOKEN #REQUIRED
      value CDATA #REQUIRED>
```

### Parameters Common to All Tasks

The following parameters apply to all *task* elements (i.e., `<usertask>`, `<updtatetask>`, `<submittask>`, `<externaltask>`, `<grouptask>`, and `<cgitask>`). In this section, the term *task* represents any of these elements. For information about parameters that apply only to a specific *task* element, see that element's section later in this chapter.

```
<!ELEMENT task (description?, areavpath, successors, timeout?,
  files?, activation?, inactivate?, resets?,
  eastartop*, eafinishop*, variables?)>
  <!ATTLIST task owner CDATA #REQUIRED
    name ID #REQUIRED
    start (t|f) "f"
    description CDATA #IMPLIED
    lock (t|f) "f"
    readonly (t|f) "f">
```

**Attributes:**

owner

The owner of the task.

name

The name of the task. A task is uniquely identified within its job by its name.

start

Specifies whether the task should be active upon its containing job's invocation. The default is `f`.

description

A description of what the task does. Can be specified as an attribute as well as a subelement of `task`.

lock

When the `lock` attribute is set to `t` the task will acquire TeamXpress locks on all the files it contains when it becomes active. If the task cannot acquire locks for one or more of the files it contains it will release any locks it has already acquired and try again every five minutes until it successfully acquires all locks. When a locking task tries to acquire a lock for a file it checks first to see if that file is locked by some other task in its own job. If it is, the locking task "steals" the lock from the other task. This behavior can result in submit time conflicts. In general it is best to ensure that no task will try to acquire locks that could already be owned by another active task.

readonly

Marking a task read only disallows users from adding, removing, or modifying files. Note: With this release of TeamXpress, `readonly` is used only by `<usertask>` and `<grouptask>`.

## Subelements:

<areavpath>

The <areavpath> subelement specifies the TeamXpress area associated with this task. Syntax is as follows:

```
<!ELEMENT areavpath EMPTY>
  <!ATTLIST areavpath v CDATA #REQUIRED>
```

<timeout>

A timeout is an optional time limit for the completion of a task. When time runs out the task is inactivated and the <succ> elements are signalled to become active. The time value for <timeout> is specified as the *v* attribute in two possible forms: *+HHHHMM*, which is the number of hours and minutes after the task becomes activated that the timeout should occur, or *MMDDYYYYHHMM*, which is the month, day, year, hour, and minute at which the timeout should occur. When using *+HHHHMM*, you must use all six digits, including leading zeros if necessary. Syntax is as follows:

```
<!ELEMENT timeout (succ)+>
  <!ATTLIST timeout v CDATA #REQUIRED>
  <!ELEMENT succ EMPTY>
    <!ATTLIST succ v IDREF #REQUIRED>
```

<files>

These are the files that the actions of a task affect. The files can be specified at configuration time (but only on <start> tasks) or dynamically (but only on active tasks). It is expected that the user interface will allow users to modify and/or add to the comment field. Syntax is as follows:

```
<!ELEMENT files (file+)>
  <!ELEMENT file EMPTY>
    <!ATTLIST file path CDATA #REQUIRED
      comment CDATA #REQUIRED>
```



```
<activation>
```

The `<activation>` element specifies the conditions under which a task will become active. The body of the `<activation>` element specifies a logical expression. When a finishing task signals a successor task, the successor task notes that the finishing task has signaled and then evaluates the logical expression to determine if it should become active. Syntax is as follows:

```
<!ELEMENT activation (and|or|not|pred)>
```

```
<!ELEMENT and (and|or|not|pred)*>
```

```
<!ELEMENT or (and|or|not|pred)*>
```

```
<!ELEMENT not (and|or|not|pred)>
```

```
<!ELEMENT pred EMPTY>
  <!ATTLIST pred v IDREF #REQUIRED>
```

For example, given tasks A, B, C that can signal task D, the `<activation>` element for D looks like this:

```
<activation>
  <and>
    <pred v="A"/>
    <or>
      <pred v="B"/>
      <pred v="C"/>
    </or>
  </and>
</activation>
```

This means (in more conventional notation):  $A \& (B \mid C)$ . When A, B or C signals D, D notes the fact that it has been signalled and evaluates  $A \& (B \mid C)$  where the values of A, B and C are whether they have signalled D. In this example, D will become active if and only if A has signalled and B or C have signalled.

```
<inactivate>
```

A task becomes inactive at the time it signals its successors. However it is often necessary to inactivate tasks other than those which have signalled a task when that task becomes active. For example, suppose a user completes a task and routes it simultaneously to five user for review. If one of those reviewers rejects the work, the task should be inactivated and removed from the lists of the other four reviewers. Syntax is as follows:

```
<!ELEMENT inactivate (pred+)>
```

For example, given tasks A and B that can signal it, task C has the following `<activation>` and `<inactivate>` sections:

```
<activation>
  <or>
    <pred v="A"/>
    <pred v="B"/>
  </or>
</activation>
<inactivate>
  <pred v="A"/>
  <pred v="B"/>
</inactivate>
```

When C becomes active, by being signalled by either A or B, it inactivates both A and B. Specification of the `<inactivate>` element is optional. If the `<inactivate>` element is unspecified it is the same as specifying an `<inactivate>` element containing all possible predecessor tasks.

```
<resets>
```

A task can be configured to reset the activation state of an arbitrary set of other tasks when it becomes active. Resetting the activation state of a task simply means that such tasks are set to a state of no tasks having activated them. This capability is useful in certain parallel task configurations. Syntax is as follows:

```
<!ELEMENT resets (reset+)>
  <!ELEMENT reset EMPTY>
  <!ATTLIST reset v IDREF #REQUIRED>
```

<eastartop>, <eafinishop>

Both when a task becomes active (<eastartop>) and when a task becomes inactive (<eafinishop>), TeamXpress extended attributes can be set, modified, or deleted on the files contained by the task. Syntax is as follows:

```
<!ELEMENT eastartop EMPTY>
  <!ATTLIST eastartop op (set|append|delete) #REQUIRED
                    name CDATA #REQUIRED
                    value CDATA #REQUIRED>

<!ELEMENT eafinishop EMPTY>
  <!ATTLIST eafinishop op (set|append|delete) #REQUIRED
                    name CDATA #REQUIRED
                    value CDATA #REQUIRED>
```

If the *op* attribute of the <ea $XXX$ op> element is *set*, the extended attribute with key *name* will be set to *value*. If *op* is *append*, *value* will be appended. If *op* is *delete*, the extended attribute with key *name* will be deleted. The *value* attribute of the <ea $XXX$ op> element can contain the following macros of the form %*name*; that will be expanded before being set as an extended attribute:

Macro Name	Description
%workflow;	Name of the job.
%workflowid;	ID of the job.
%task;	Name of the task.
%taskid;	ID of the task.
%taskowner;	Owner of the task.
%time;	The current wall clock time.
%area;	VPATH of the task's area.
%path;	Path of the file from area root.
%fullpath;	Full path of the file from server root.

Macro Name	Description
<code>%taskcomment;</code>	Task-specific comment added to the extended attribute.
<code>%filecomment;</code>	File-specific comment added to the extended attribute.

### <usertask> Element

A <usertask> element defines user tasks that appear on a user's task list.

```
<!ELEMENT usertask (description?, areavpath, successors, timeout?,
    files?, activation?, inactivate?, resets?,
    eastartop*, eafinishop*, variables?)>
<!ATTLIST usertask owner CDATA #REQUIRED
    name ID #REQUIRED
    start (t|f) "f"
    description CDATA #IMPLIED
    lock (t|f) "f"
    readonly (t|f) "f">
```

### Attributes:

There are no attributes unique to user tasks. See “Parameters Common to All Tasks” on page 182 for descriptions of the attributes shown in the preceding DTD excerpt.

### Subelements:

See “Parameters Common to All Tasks” on page 182 for descriptions of the subelements from the preceding DTD excerpt that are common to other tasks. Usage of the <successors> subelement is as follows:

```
<successors>
```

The `<successors>` subelement specifies the possible alternative sets of successor tasks to signal when the user task is finished. The GUI presents the user with a set of options for finishing a task. The text of the description of the task is used to label the alternatives for the user (e.g., “Mark Done”, “Reject”, “Approve” and so on).

```
<!ELEMENT successors (successorset+)>
<!ELEMENT successorset (description?, succ+)>
  <!ATTLIST successorset description CDATA #IMPLIED>
<!ELEMENT succ EMPTY>
  <!ATTLIST succ v IDREF #REQUIRED>
```

### **<grouptask> Element**

A group task is similar to a user task in that it appears on a user’s task list. A group task, however, belongs to an arbitrary group of users and therefore shows up in the task list of every user who belongs to that arbitrary group. A group task becomes identical in behavior to a user task when one user from the group takes ownership of the task via the GUI or the CLT `iwtaketask`.

```
<!ELEMENT grouptask (description?, areavpath, successors, sharedby,
  timeout?, files?, activation?, inactivate?,
  resets?, eastartop*, eafinishop*, variables?)>
  <!ATTLIST grouptask name ID #REQUIRED
    start (t|f) "f"
    description CDATA #IMPLIED
    lock (t|f) "f"
    readonly (t|f) "f">
```

### **Attributes:**

There are no attributes unique to group tasks. See “Parameters Common to All Tasks” earlier in this chapter for descriptions of the attributes shown in the preceding DTD excerpt.

**Subelements:**

See “<usertask> Element” on page 188 for a description of the <successors> subelement. See “Parameters Common to All Tasks” on page 182 for descriptions of the other subelements from the preceding DTD excerpt that are common to other tasks. Subelements that are unique to group tasks are as follows:

<sharedby>

The <sharedby> element specifies the arbitrary set of users who share this group task. The element allows an arbitrary combination of individual TeamXpress users and OS groups to be shared owners of the group task. Syntax is as follows:

```
<!ELEMENT sharedby (user|group)*>
  <!ELEMENT user EMPTY>
    <!ATTLIST user v CDATA #REQUIRED>
  <!ELEMENT group EMPTY>
    <!ATTLIST group v CDATA #REQUIRED>
```

**<externaltask> Element**

An external task runs external programs when it becomes active.

```
<!ELEMENT externaltask (description?, areavpath, successors,
  command, timeout?, files?, activation?,
  inactivate?, resets?, eastartop*,
  eafinishop*, variables?)>
  <!ATTLIST externaltask owner CDATA #REQUIRED
    name ID #REQUIRED
    start (t|f) "f"
    description CDATA #IMPLIED
    lock (t|f) "f"
    readonly (t|f) "f">
```

**Attributes:**

There are no attributes unique to external tasks. See “Parameters Common to All Tasks” on page 182 for descriptions of the attributes shown in the preceding DTD excerpt.

**Subelements:**

See “<usertask> Element” on page 188 for a description of the <successors> subelement. See “Parameters Common to All Tasks” on page 182 for descriptions of the other subelements from the preceding DTD excerpt that are common to other tasks. Subelements that are unique to external tasks are as follows:

**<command>**

The <command> element specifies the full path of the program to be run on activation followed by any initial arguments. When the program is run by the workflow system, the following arguments are passed as separate arguments: the containing job’s ID (in decimal), the ID of the task, and each file from the task’s file list. On Solaris the program will be run as the owner of the task. On Windows NT it runs as the SYSTEM user.

Syntax for use of the <command> subelement is as follows:

```
<!ELEMENT command EMPTY>
  <!ATTLIST command v CDATA #REQUIRED>
```

When an external program finishes it must run the `iwcallback` program, passing the job and task IDs and a return code as arguments, to tell the server that it is finished. The server does not wait for an external task to finish. The server uses the return code passed to `iwcallback` to choose the set of successors to signal. If the return code is out of the range  $0..n-1$  (where  $n$  is the number of <successorset> elements), the last successor set is used.

**<cgitask> Element**

A CGI task behaves much like an external task. The only difference is that a CGI task does not run its `<command>` element (it relies on the user interface for that). A CGI task expects to have `iwcallback` called to notify it of program completion.

```
<!ELEMENT cgitask (description?, areavpath, successors, command,
                  timeout?, files?, activation?, inactivate?,
                  resets?, eastartop*, eafinishop*, variables?)>
<!ATTLIST cgitask owner CDATA #REQUIRED
                  name ID #REQUIRED
                  start (t|f) "f"
                  description CDATA #IMPLIED
                  lock (t|f) "f"
                  immediate (t|f) "f"
                  readonly (t|f) "f">
```

**Attributes:**

There are no attributes unique to CGI tasks. See “Parameters Common to All Tasks” on page 182 for descriptions of the attributes shown in the preceding DTD excerpt.

**Subelements:**

There are no subelements unique to CGI tasks. See “<usertask> Element” on page 188 for a description of the `<successors>` subelement. See “Parameters Common to All Tasks” on page 182 for descriptions of the other subelements from the preceding DTD excerpt.



**<submittask> Element**

A submit task performs a submit operation on its contained files.

```
<!ELEMENT submittask (description?, areavpath, successorset,
    timeout?, files?, activation?, inactivate?,
    resets?, eastartop*, eafinishop*,
    variables?)>
    <!ATTLIST submittask owner CDATA #REQUIRED
        name ID #REQUIRED
        start (t|f) "f"
        skipconflicts (t|f) "f"
        skiplocked (t|f) "f"
        override (t|f) "f"
        unlock (t|f) "f">
        description CDATA #IMPLIED
```

If the submit task succeeds, the successor tasks specified in the <successorset> element are signaled. If the submit task fails, the submit task goes into a special state that the user interface can detect. When the user interface has resolved any conflicts it retries the operation so that the job can continue. For the purposes of workflow, a submit task is considered successful even if some of its contained files were not submitted because of being up to date with the staging area.

**Attributes:**

See “Parameters Common to All Tasks” on page 182 for descriptions of the attributes shown in the preceding DTD excerpt that are common to other tasks. Attributes that are unique to submit tasks are as follows:

skipconflicts	Does not submit conflicting files.
skiplocked	Does not submit locked files.
override	Overwrites the staging area version of conflicting files.

**Subelements:**

There are no subelements unique to submit tasks. See “<usertask> Element” on page 188 for a description of the <successorset> subelement. See “Parameters Common to All Tasks” on page 182 for descriptions of the other subelements from the preceding DTD excerpt.

### <updatetask> Element

An update task does the equivalent of Get Latest (if the source is the staging area) or Copy To (if the source is another workarea or edition) on its contained files.

```
<!ELEMENT updatetask (description?, areavpath, successorset,
                    srcareavpath, timeout?, files?, activation?,
                    inactivate?, resets?, eastartop*,
                    eafinishop*, variables?)>
<!ATTLIST updatetask owner CDATA #REQUIRED
                    name ID #REQUIRED
                    start (t|f) "f"
                    delete (t|f) "t"
                    overwritemod (t|f) "f"
                    description CDATA #IMPLIED
                    lock (t|f) "f">
```

If the update task fails because of conflicts, it goes into a state like that of a failed submit task. The user interface is responsible for resolving conflicts and retrying the update task.

### Attributes:

See “Parameters Common to All Tasks” on page 182 for descriptions of the attributes shown in the preceding DTD excerpt that are common to other tasks. Attributes that are unique to update tasks are as follows:

delete	Propagates deleted files to the destination area.
overwritemod	Overwrites conflicting versions of files in the destination area.

### Subelements:

See “<usertask> Element” on page 188 for a description of the <successorset> subelement. See “Parameters Common to All Tasks” on page 182 for descriptions of the other subelements from the preceding DTD excerpt that are common to other tasks. Subelements that are unique to update tasks are as follows:

#### <srcareavpath>

The area from which files are copied.

**<endtask> Element**

An end task is a marker for the end of a job. When an end task becomes active, its containing job is terminated and all locks held in the job are released.

```
<!ELEMENT endtask (activation?, eastartop*, eafinishop*)>
  <!ATTLIST endtask name ID #REQUIRED
    description CDATA #IMPLIED>
```

**Attributes:**

There are no attributes unique to end tasks. See “Parameters Common to All Tasks” on page 182 for descriptions of the attributes from the preceding DTD excerpt.

**Subelements:**

There are no subelements unique to end tasks. See “Parameters Common to All Tasks” on page 182 for descriptions of the subelements from the preceding DTD excerpt.

**<dummysubtask> Element**

A <dummysubtask> element is a task that waits for its mandatory timeout to expire. If "+000000" is specified as a timeout value, <dummysubtask> becomes a spacer task. Dummy tasks let a workflow designer create a time interval unrelated to any actual job activity. A dummy task does not have an owner or areavpath.

```
<!ELEMENT dummysubtask (description?, timeout, files?, activation?,
  inactivate?, resets?, eastartop*,
  eafinishop*, variables?)>
  <!ATTLIST dummysubtask name ID #REQUIRED
    start (t|f) "f"
    description CDATA #IMPLIED>
```

**Attributes:**

There are no attributes unique to dummy tasks. See “Parameters Common to All Tasks” on page 182 for descriptions of the attributes from the preceding DTD excerpt.

**Subelements:**

There are no subelements unique to dummy tasks. See “Parameters Common to All Tasks” on page 182 for descriptions of the subelements from the preceding DTD excerpt.

### <locktask> Element

A <locktask> element is a task that attempts to acquire locks on the files it owns. If it succeeds, it transitions to the successors specified in its `success` element. If it fails, it transitions to the successors specified in its `failure` element. This provides users with a way of backing out of a job or choosing an alternate path in a job that cannot acquire its locks.

```
<!ELEMENT locktask (description?, areavpath, success, failure,
                    files?, activation?, inactivate?, resets?,
                    eastartop*, eafinishop*, variables?)>
  <!ATTLIST locktask owner CDATA #REQUIRED
                    name ID #REQUIRED
                    start (t|f) "f"
                    description CDATA #IMPLIED>
  <!-- Locking is implied -->
  <!ELEMENT success (succ+)>
  <!ELEMENT failure (succ+)>
```

### Attributes:

There are no attributes unique to lock tasks. See “Parameters Common to All Tasks” on page 182 for descriptions of the attributes from the preceding DTD excerpt.

### Subelements:

See “Parameters Common to All Tasks” on page 182 for descriptions of the subelements from the preceding DTD excerpt that are common other tasks. Subelements that are unique to lock tasks are as follows:

#### <success>

Names the successor tasks that become active when the lock task succeeds.

#### <failure>

Names the successor tasks that become active when the lock task fails.

## Using Perl Modules

These Perl modules are provided as reference for workflow template developers. Refer to the Perl modules for the latest documentation, or see `iw-home/iw-perl/bin/perldoc module`.

## Name

`TeamSite::WFsystem`

## Synopsis

Utilities for accessing the TeamXpress workflow engine. This provides access to functions for querying the entire workflow system.

```
use TeamSite::WFsystem;  
$system = new TeamSite::WFsystem();
```

## Functions

<code>new()</code>	Creates a new <code>WFsystem</code> object. This only works on the local TeamXpress server.
<code>GetWorkflows()</code>	Returns an array of <code>WFworkflow</code> objects corresponding to all jobs in the system.
<code>GetActiveWorkflows()</code>	Returns an array of <code>WFworkflow</code> objects corresponding to all active jobs in the system.
<code>GetTasks()</code>	Returns an array of <code>WFworkflow</code> objects corresponding to all tasks in the system.
<code>GetActiveTasks()</code>	Returns an array of <code>WFworkflow</code> objects corresponding to all active tasks in the system.
<code>CreateWorkflow(\$spec)</code> , <code>CreateWorkflow(\$spec, \$tmpfile)</code>	Creates a workflow instance from <code>\$spec</code> (a workflow specification in the form of a string). Returns a <code>TeamSite::WFworkflow</code> . If the <code>\$tmpfile</code> arg is set, <code>tmpfile</code> is used instead of <code>stdin</code> . Using this option in the context of HTTP may result in timing security issues.
<code>Refresh()</code>	Call after changes have been made.

## Examples

```
$system = new TeamSite::WFsystem();  
$wfs = $system->GetWorkflows();  
$wfs = $system->GetActiveWorkflows();
```

`$wfs` is a reference to an array containing references to `TeamSite::WFworkflow` objects.

```
$tasks = $system->GetTasks();  
$tasks = $system->GetActiveTasks();
```

`$tasks` is a reference to an array containing `TeamSite::WFtask` references.

```
my $wf_system = TeamSite::WFsystem->new();  
my $wf        = $wf_system->CreateWorkflow($spec);  
if ((!defined $wf) || !$wf->IsValid() || $wf->GetError())  
{  
    ... handle error...  
}
```

## Name

`TeamSite::WFworkflow` - interface to TeamXpress workflow objects.

## Synopsis

Utilities for using the TeamXpress workflow engine. This supplies functions for manipulating and querying workflows.

```
$workflow = new TeamSite::WFworkflow($id)
```

## Functions

<code>new(\$id)</code>	Creates a new <code>WFworkflow</code> object.
<code>GetId()</code>	Fetches the workflow ID.
<code>GetError()</code>	Fetches the last error message (if any).
<code>SetError(\$error_string)</code>	Sets the error message to <code>\$error_string</code> and returns the previous error message (if any).
<code>IsValid()</code>	Determines whether this a valid workflow object.
<code>GetTasks()</code>	Gets the tasks owned by this workflow.
<code>GetOwner()</code>	Returns owner of workflow.
<code>GetCreator()</code>	Returns the creator of this workflow.
<code>GetName()</code>	Returns the name of the workflow.
<code>GetDescription()</code>	Returns the description for this workflow.
<code>Invoke()</code>	Starts this workflow running. Returns a <code>TeamSite::WFtask</code> object. If the returned object is valid, then a CGI task that wishes to be run.
<code>GetVariable(\$name)</code>	Gets the value of a workflow variable.
<code>SetVariable(\$name, \$value)</code>	Sets the value for a workflow variable. Returns exit status of underlying CLT (non-zero indicates an error occurred).
<code>CreateVariable(\$name, \$value)</code>	Creates a workflow variable. If the variable already exists, this fails.
<code>DeleteVariable(\$name)</code>	Deletes a workflow variable.
<code>Refresh()</code>	Call when workflow object has been modified.

## Examples

```
$workflow = new TeamSite::WFworkflow($id);  
$tasks = $workflow->GetTasks();
```

`$tasks` is a reference to a list containing `TeamSite::WFtask` objects.

## Name

`TeamSite::WFtask` - interface to TeamXpress task objects.

## Synopsis

Utilities for using the TeamXpress workflow engine. This supplies functions for manipulating and querying tasks.

```
$task = new TeamSite::WFtask($id);
```

## Functions

<code>new(\$id)</code>	Creates a new <code>WFtask</code> object.
<code>GetId()</code>	Fetches the task ID.
<code>GetType()</code>	Returns the task type.
<code>GetOwner()</code>	Gets the owner of this task.
<code>GetDescription()</code>	Returns the description for this task.
<code>GetWorkflowId()</code>	Returns the ID of the job that owns this task.
<code>AddFile(\$path, \$comment)</code>	Adds a file with comment to a task.
<code>SetComment(\$comment)</code>	Sets comment on task.
<code>(\$success, \$immediatetask) SelectTransition(\$which, \$comment)</code>	Selects a transition for this task. <code>\$success</code> is a boolean and <code>\$immediatetask</code> is a possibly invalid <code>TeamSite::WFtask</code> to run.
<code>(\$success, \$immediatetask) = CallBack(\$rcode, \$comment)</code>	Callback from a CGI task or external task.



	<code>\$immediatetask</code> is a possibly invalid <code>TeamSite::WFtask</code> to run.
<code>GetCommand()</code>	Gets the command string for an external task.
<code>Refresh()</code>	Call when the server side object has been changed.
<code>IsValid()</code>	Returns true if this is a valid task.
<code>GetSubmitEvents()</code>	Returns a (possibly empty) array of <code>SubmitEvent</code> objids (as strings). It returns an array because there may have been conflicts or other problems which could produce multiple events.
<code>GetUpdateEvents()</code>	Returns a (possibly empty) array of <code>UpdateEvent</code> objids (as strings). It returns an array because there may have been conflicts or other problems that could produce multiple events.
<code>GetFiles()</code>	Returns a (possibly empty) array of file names.
<code>GetArea()</code>	Gets the area for the task, such as <code>/default/main/dev/WORKAREA/andre</code>
<code>GetError()</code>	Fetches the last error message (if any).
<code>SetError(\$error_string)</code>	Sets the error message to <code>\$error_string</code> and returns the previous error message (if any).

**Example**

```
$task = new TeamSite::WFtask($id);
```



# Configuring Metadata Capture and Search

---

TeamXpress metadata capture lets end users add metadata information to files. After the metadata is deployed to a database via DataDeploy, end users can use TeamXpress metadata search to query the database and locate files having specific metadata characteristics.

You must configure TeamXpress to enable metadata capture or search; they do not appear by default in the TeamXpress GUI. Configuration involves editing a set of configuration files to specify the appearance and behavior of the metadata forms, and then editing the main TeamXpress configuration file (`iw.cfg`) to add metadata capture or search to a TeamXpress GUI menu. After configuration is complete, end users enter information in either a metadata entry form or a metadata search form. Following data entry, the forms are processed by the metadata capture or metadata search subsystem residing on the TeamXpress server. Metadata capture and search exist as separate entities, each accessed via its own TeamXpress GUI menu item. Metadata capture can exist without metadata search. However, metadata search requires that you also configure metadata capture.

The rest of this chapter describes how the metadata capture and search subsystems work, and how to configure them. For details about using metadata capture and metadata search, see the *TeamXpress User's Guide*.

## Configuring Metadata Capture

The following sections describe:

- A metadata capture overview.
- The main components that make up metadata capture.
- How to configure metadata capture.

## Overview

Metadata capture is a file-specific feature. That is, you must explicitly select the file(s) on which you intend to set metadata. You cannot globally set metadata for an entire area or branch. For example, to set metadata on all files in a workarea, you must select each file in that workarea (by choosing **Select All**, or by clicking the checkbox next to each file, etc.) and then initiate a metadata capture session. See the *TeamXpress User's Guide* for more information.

Metadata capture can be initiated in one of two ways:

- Through a job as part of a `<cgitask>` element (see “Initiating Metadata Capture from a Job Specification File” on page 227), or
- From a menu item in the TeamXpress GUI. A menu item for metadata capture is not on a TeamXpress menu by default; you must add it as described later in this chapter.

## Components

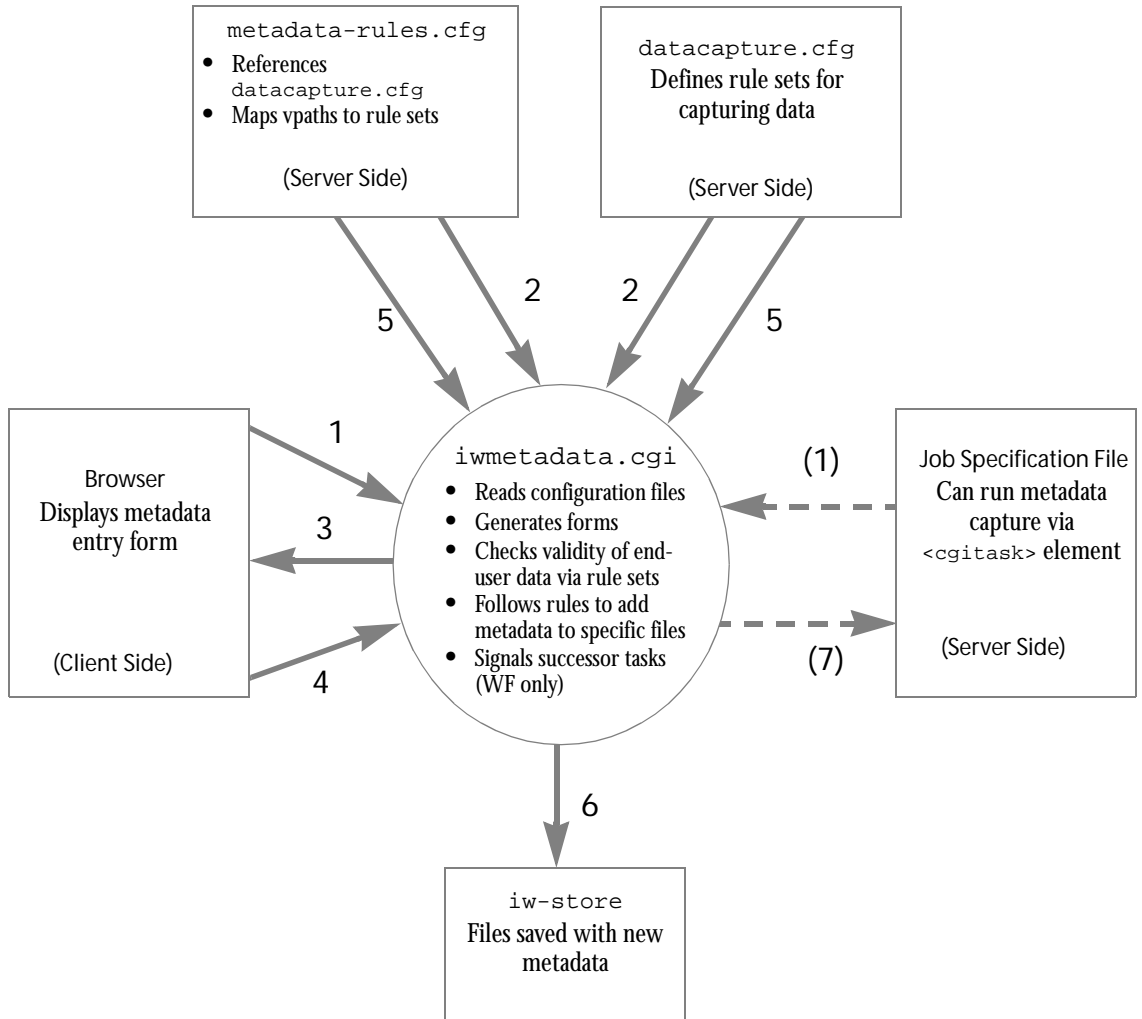
No matter how metadata capture is initiated, it relies on four main components:

- The `iw-home/local/config/metadata-rules.cfg` configuration file, which maps vpaths to the data capture rules defined in `datacapture.cfg`.
- The `iw-home/local/config/datacapture.cfg` configuration file, which defines rule sets for capturing data.
- The metadata capture CGI `iwmetadata.cgi`, which interprets data from end users and rules in `datacapture.cfg` and `metadata-rules.cfg`, produces browser graphics and prompts, and acts as an interface with workflow configuration files (if metadata capture is running as part of a job).
- A browser interface for end-user input.

Two configuration files (`metadata-rules.cfg` and `datacapture.cfg`) allow you to configure the following on a per-user or per-vpath basis:

- The metadata item name that is displayed in the metadata entry form.
- The interface through which an end user enters input (e.g., a checkbox, data field, etc.).
- The type of data that is acceptable or unacceptable in any given field.
- Whether input is required for any given field.

The following diagram shows how these components work together. Sections following the diagram explain each diagram step and component in detail.



Metadata Capture Overview

### Diagram Key

1. The metadata CGI receives a list containing the names of the files that will have metadata added to them. The list can come from an instantiated job (if metadata capture is initiated from a job) or from the browser (if initiated from the TeamXpress GUI).
2. The metadata CGI reads both configuration files (`metadata-rules.cfg` and `datacapture.cfg`) to determine what information it should display in the metadata entry form. It makes this determination on a per-file basis, so that the entry form can contain different prompts and actions for different files.
3. The metadata CGI displays the metadata entry form on the client system via the GUI.
4. An end user fills in data and submits the entry form back to the metadata CGI.
5. The metadata CGI consults the rules in both configuration files to verify the validity of the data entered by the end user. If the data does not meet all necessary criteria, notification is sent to the end user so that data can be re-entered.
6. If the data meets all necessary criteria, the metadata CGI adds the new metadata (in the form of TeamXpress extended attributes) to the specified files. The metadata CGI interfaces directly with the backing store to update the files with the new metadata.
7. If metadata capture was initiated from a job, the metadata CGI notifies the workflow subsystem, which starts successor task 0 (zero) as defined in the job specification file.

## Configuring Metadata Capture

You must perform three main activities to configure metadata capture:

1. Create a `metadata-rules.cfg` file in `iw-home/local/config` for your site.
2. Create a `datacapture.cfg` file in `iw-home/local/config` for your site.
3. Add a **Set Metadata** item to the TeamXpress GUI so that end users can access metadata capture.

The following sections describe these steps in detail.

**Configuring metadata-rules.cfg**

The `metadata-rules.cfg` file maps vpaths to data capture rules that are defined in `datacapture.cfg`. The `metadata-rules.cfg` file consists of a series of `<cond>` (conditional) elements. A `<cond>` element can contain `<rule>` elements and other `<cond>` elements. Each vpath is run through `metadata-rules.cfg`, resulting in a one-to-many mapping from vpaths to named rules. Whenever a list of `<cond>` elements is found, the first to match the current vpath takes effect, and the rest of the elements in the list are discarded.

To set up `iw-home/local/config/metadata-rules.cfg` for your site, it is recommended that you copy and edit the example file provided with TeamXpress (`iw-home/local/config/metadata-rules.cfg.example`). Use the following DTD and annotated examples as references for your own site-specific configuration.

**DTD: metadata-rules.cfg**

The `metadata-rules.cfg` file uses the following DTD:

```
<!ELEMENT metadata-rules (cond)*>
<!ELEMENT cond (cond|rule)*>
  <!ATTLIST cond
    vpath-regex CDATA #REQUIRED
  >
<!ELEMENT rule EMPTY>
  <!ATTLIST rule
    name CDATA #REQUIRED
  >
```

### Sample metadata-rules.cfg File 1

The following `metadata-rules.cfg` file is distributed with TeamXpress as `iw-home/local/config/metadata-rules.cfg.example`.

```
<?xml version="1.0" encoding="UTF-8" ?> ← International Encoding 1
<metadata-rules>
  <cond vpath-regex="."> ← Vpath Identifier 2
    <rule name="Default Rule" /> ← Rule Identifier 3
  </cond>
</metadata-rules>
```

### Sample metadata-rules.cfg File 1 Notes

- 1. International Encoding:** UTF-8 is an encoding of Unicode, a standard for encoding the character sets of international languages. All Web assets should specify their encoding as UTF-8. For details about Web asset encoding, see Appendix D, “Internationalization”.
- 2. Vpath Identifier:** Names the vpath (in this case all directories) to which the rule(s) named in the following subelement(s) will be applied.
- 3. Rule Identifier:** Names the rule that applies to the preceding vpath. The rule itself is defined in the `<ruleset>` element in `iw-home/local/config/datacapture.cfg`. In this example, the `Default Rule` rule defined in `datacapture.cfg` will always apply to all directories.



### Sample metadata-rules.cfg File 2

The following metadata-rules.cfg file illustrates a more sophisticated example:

```

<metadata-rules>
  <cond vpath-regex="/^/default/main/dev/syndication">
    <rule name="Default" />
    <rule name="Syndication" />
    <cond vpath-regex="/\\.pdf$" >
      <rule name="PDF Files" />
    </cond>
    <cond vpath-regex="/\\.doc$" >
      <rule name="MS Word Files" />
    </cond>
  </cond>

  <cond vpath-regex="/^/default/main/dev/www">
    <rule name="Default" />
    <rule name="Web Content" />
    <cond vpath-regex="/\\.html$" >
      <rule name="HTML Files" />
    <cond vpath-regex="/pr/" >
      <rule name="PR" />
    </cond>
    <cond vpath-regex="/corp/" >
      <rule name="Corporate" />
    </cond>
  </cond>
</metadata-rules>

```

### Sample metadata-rules.cfg File 2 Notes

1. **Vpath Identifier:** Files on the /main/dev/syndication branch will always receive the rules named in the following subelements.
2. **Rule Identifiers:** The Default and Syndication rules defined in datacapture.cfg will always apply to the /main/dev/syndication branch.



- 3. Vpath Identifier:** Files ending in `.pdf` on the `/main/dev/syndication` branch will receive rules in addition to those defined by `Default` and `Syndication`.
- 4. Rule Identifier:** The `PDF Files` rule defined in `datacapture.cfg` will apply to files ending in `.pdf` on the `/main/dev/syndication` branch.
- 5. Vpath Identifier:** Files ending in `.doc` on the `/main/dev/syndication` branch will receive rules in addition to those defined by `Default` and `Syndication`.
- 6. Rule Identifier:** The `MS Word Files` rule defined in `datacapture.cfg` will apply to files ending in `.doc` on the `/main/dev/syndication` branch.
- 7. Vpath Identifier:** The `/main/dev/www` branch will always receive the rules named in the following subelements.
- 8. Rule Identifiers:** The `Default` and `Web Content` rules defined in `datacapture.cfg` will apply to the `/main/dev/www` branch.
- 9. Vpath Identifier:** Files ending in `.html` on the `/main/dev/www` branch will receive rules in addition to those defined by `Default` and `Web Content`.
- 10. Rule Identifier:** The `HTML Files` rule defined in `datacapture.cfg` will apply to files ending in `.html` on the `/main/dev/www` branch.
- 11. Vpath Identifier:** Files ending in `.html` in the `pr` directory on the `/main/dev/www` branch will receive rules in addition to those defined by `Default` and `Web Content`.
- 12. Rule Identifier:** The `PR` rule defined in `datacapture.cfg` will apply to files ending in `.html` in the `pr` directory on the `/main/dev/www` branch.
- 13. Vpath Identifier:** Files ending in `.html` in the `corp` directory on the `/main/dev/www` branch will receive rules in addition to those defined by `Default` and `Web Content`.
- 14. Rule Identifier:** The `Corporate` rule defined in `datacapture.cfg` will apply to files ending in `.html` in the `corp` directory on the `/main/dev/www` branch.

### Configuring `datacapture.cfg`

The `datacapture.cfg` file defines rule sets for capturing data. Rules are referred to by name in `metadata-rules.cfg` (see “Configuring `metadata-rules.cfg`” on page 207).

Rules contain *items*, where each item is a single set of data that is to be captured from the end user. An item consists of one or more *instances*. Each instance encapsulates how to capture the data for the item, and each instance defines an ACL that determines which (if any) instance a particular user is allowed to use to enter the data.

The metadata capture form is a data capture template (DCT) that is configured specifically for metadata capture. The DCT subsystem that generates the metadata capture form is the same subsystem that generates DCTs for TeamXpress Templating. A main difference between the two implementations is the location of the `datacapture.cfg` file. TeamXpress Templating relies on multiple `datacapture.cfg` files (one for each data type), while metadata capture relies on a single `datacapture.cfg` file (in `iw-home/local/config`).

See for a complete explanation of `datacapture.cfg` files, including annotated examples and explanations of elements and attributes. Note that even though the examples are specific to TeamXpress Templating, they are useful as reference points for setting up `datacapture.cfg` for metadata capture.

An example `datacapture.cfg` file configured specifically for metadata capture is also included with TeamXpress (refer to `iw-home/local/config/datacapture.cfg.example`). An annotated explanation of that file is shown in “Sample `datacapture.cfg` File 1” on page 214.

To set up `iw-home/local/config/datacapture.cfg` for your site, it is recommended that you copy and edit the `datacapture.cfg.example`, using the following DTD and annotated examples as reference points for your own site-specific configuration.

## DTD: `datacapture.cfg`

The `datacapture.cfg` file uses the following DTD. This DTD is also available online in `iw-home/local/config`.

```
<!ELEMENT data-capture-requirements (ruleset)*>
  <!ATTLIST data-capture-requirements
    name CDATA #REQUIRED
    type(metadata|content|workflow) #REQUIRED
  >

<!ELEMENT ruleset (item)*>
  <!ATTLIST ruleset
    name CDATA #REQUIRED
  >

<!ELEMENT item (database?,(%instance;)*)>
  <!ATTLIST item
    name CDATA #REQUIRED
    description CDATA #IMPLIED
  >

<!ENTITY % instance "(checkbox|radio|text|textarea|select|replicant)" >

<!ELEMENT checkbox(allowed|option)*>
  <!ATTLIST checkbox
    required (t|f)"f"
    delimiter CDATA", "
  >

<!ELEMENT radio (allowed|option)*>
  <!ATTLIST radio
    required (t|f) "f"
  >

<!ELEMENT text (allowed)*>
  <!ATTLIST text
    required (t|f) "f"
    maxlength NUM
    size NUM
    validation-regex CDATA -- regex(5) for validating this element -
  >
```

```

<!ELEMENT textarea (allowed)*>
  <!ATTLIST textarea
    required (t|f) "f"
    rows NUM
    cols NUM
    validation-regex CDATA -- regex(5) for validating this element -
  >
<!ELEMENT select (allowed|optgroup|option)*>
  <!ATTLIST select
    required (t|f) "f"
    size NUM
    multiple (t|f) "f"
    delimiter CDATA ", "-- for multiple=t only --
  >
<!ELEMENT replicant (allowed|item)*>
  <!ATTLIST replicant
    min NUM
    max NUM
    default NUM
  >

<!ELEMENT optgroup (optgroup*, option*)+>
  <!ATTLIST optgroup
    label CDATA #REQUIRED
  >

<!ELEMENT option EMPTY>
  <!ATTLIST option
    selected (t|f) "f"
    value CDATA #IMPLIED
    label CDATA #REQUIRED
  >

<!ELEMENT database EMPTY >
  <!ATTLIST database
    deploy-column(t|f) "t"
    searchable (t|f) "t"
    data-type CDATA "VARCHAR(255)"
    data-format CDATA #IMPLIED
  >

```

```
<!ELEMENT allowed (cred|and|or|not)>
```

```
<!ELEMENT cred EMPTY>
<!ATTLIST cred
  role CDATA #IMPLIED
  user CDATA #IMPLIED
>
```

```
<!ELEMENT and (cred|and|or|not)+>
```

```
<!ELEMENT or (cred|and|or|not)+>
```

```
<!ELEMENT not (cred|and|or|not)>
```

### **Sample datacapture.cfg File 1**

The following `datacapture.cfg` file is distributed with TeamXpress as `iw-home/local/config/datacapture.cfg.example`. See the section immediately following the file for an explanation of the numbered callouts. See the *TeamXpress Templating and Deployment Guide* for a complete explanation of `datacapture.cfg` files.

```
<?xml version="1.0" encoding="UTF-8" ?> ← International Encoding 1
```

```
<!-- A <data-capture-requirements> element with type="metadata"
      can contain multiple <ruleset> elements.
```

Note: The <database> elements have no effect on the metadata capture process. These optional elements are used to help integrate metadata capture with Data Deploy. Data Deploy configuration files can be automatically generated from datacapture.cfg files. The <database> tags ensure the database tables are built using the appropriate datatype.

```
-->
```

```
<data-capture-requirements type="metadata"> ← Metadata Identifier 2
```

```
<ruleset name="Default Rule"> ← Rule Identifier 3
```

```
<description>
```

```
    This rule applies to all files on all branches.
```

```
</description>
```

```
<item name="Title">
```

```
    <database searchable="t" data-type="VARCHAR(60)" /> ← database Element 4
```

```
    <text required="t" maxlength="60" /> ← Instance (text) 5
```

```
</item>
```

```
<item name="Description">
```

```
    <database data-type="VARCHAR(100)" />
```

```
    <text required="t" maxlength="100" />
```

```
</item>
```

```
<item name="Type">
```

```
    <database data-type="VARCHAR(30)" />
```

```
    <select>
```

```
        <option label="White Paper" value="white_paper" />
```

```
        <option label="Datasheet" value="datasheet" />
```

```
        <option label="Press Release" value="press_release" />
```

```
        <option label="Architecture Overview" value="architecture" />
```

```
        <option label="Futures Overview" value="futures" />
```

```
        <option label="Program Material" value="program_material" />
```

```
    </select>
```

```
</item>
```



```
<item name="Category">
  <database data-type="VARCHAR(40)" />
  <!-- To use the example callout,
    1. Comment out this select element.
    2. Uncomment the text element.
  -->
  <select>
    <option label="Internet - Financial" value="financial_internet"/>
    <option label="Internet - Manufacturing" value="manufacturing_in
      ternet" />
    <option label="Internet - Services" value="services_internet" />
    <option label="Extranet - Tier 1" value="tier_1_extranet" />
    <option label="Extranet - Tier 2" value="tier_2_extranet" />
    <option label="Extranet - Tier 3" value="tier_3_extranet" />
  </select>
  <!--
  <text>
    <callout type="cgi"
      label="Query for Categories"
      url="/iw-bin/iw_cgi_wrapper.cgi/example_datacapture_callout.i
        pl/metadata-category-options.txt" />
  </text>
  -->
</item>

<item name="Languages">
  <database data-type="VARCHAR(20)" />
  <checkbox>
    <option label="English" value="English" />
    <option label="German" value="German" />
    <option label="French" value="French" />
    <option label="Japanese" value="Japanese" />
    <option label="Chinese" value="Chinese" />
  </checkbox>
</item>

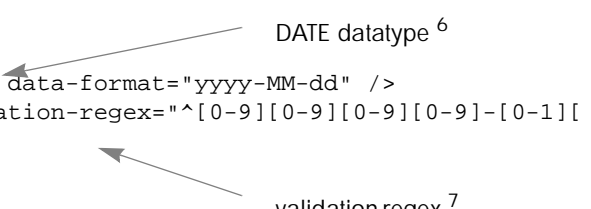
<item name="Source">
  <database data-type="VARCHAR(50)" />
  <text maxlength="50" />
</item>
```



```
<item name="Launch Date">
  <database data-type="DATE" data-format="yyyy-MM-dd" />
  <text required="t" maxlength="10" validation-regex="^[0-9][0-9]
    [0-9][0-9]-[0-1][0-9]-[0-3][0-9]" />
</item>

<item name="Expiration Date">
  <database data-type="DATE" data-format="yyyy-MM-dd" />
  <text maxlength="10" validation-regex="^[0-9][0-9][0-9][0-9]-[0-1][
    0-9]-[0-3][0-9]" />
</item>

<item name="Keywords">
  <database data-type="VARCHAR(100)" />
  <text maxlength="100" />
</item>
</ruleset>
</data-capture-requirements>
```



DATE datatype <sup>6</sup>

validation-regex <sup>7</sup>

## Sample `datacapture.cfg` File 1 Notes

The following information is specific to the example file shown in the preceding section. For more detailed information about `datacapture.cfg` files, see the *TeamXpress Templating and Deployment Guide*.

- 1. International Encoding:** UTF-8 is an encoding of Unicode, a standard for encoding the character sets of international languages. All Web assets should specify their encoding as UTF-8. For details about Web asset encoding, see Appendix D, “Internationalization”.
- 2. Metadata Identifier:** When configuring `datacapture.cfg` for metadata capture, you must specify `"type=metadata"` in the `<data-capture-requirements>` element as shown here.
- 3. Rule Identifier:** The `<ruleset>` element contains all of the items that make up the rule set that defines the appearance and behavior of the data capture form. A `datacapture.cfg` file that is configured for metadata capture can contain any number of `<ruleset>` elements (as opposed to TeamXpress Templating `datacapture.cfg` files, which can contain just one `<ruleset>` element). This example file happens to contain just one `<ruleset>` element; it could contain more if necessary. The rule defined here is named `Default Rule`, and is referenced by the `metadata-rules.cfg` file shown page 208. The name attribute is required and its value appears in the TeamXpress GUI as the name of the data capture form. More than one form can appear on a single page. Optional subelements `<label>`, `<description>`, `<item>`, and `<itemref>`. The `<label>` subelement is used to provide a label on the data capture form. The `<itemref>` subelement requires the name attribute and is used as a stand-in for the `<item>` subelement in a `<symbol-table>` element.
- 4. database Element:** The optional `<database>` element facilitates the use of the appropriate data type in DataDeploy and is used only for generation of the `mdc_dd.cfg` file. It does not control any aspects of the metadata capture or search forms. The `<data-type>` and `<searchable>` information in the `<database>` element are passed on to `mdc_dd.cfg`, which in turn uses that information to control how metadata is deployed to a database via DAS. The `<database>` element has four attributes. Because attribute order in XML documents is important; these attributes should occur in the order they are listed below:
  - `deploy-column` can be either `"t"` (default) or `"f"` and allows you to set whether or not data entered for the item is deployed to a database column.

- `searchable` can be either "t" (default) or "f" and allows you to set whether or not users can search against this item.
- `data-type` is required and is any JDBC database type. If you do not set the `data-type` attribute, a default datatype of `VARCHAR (255)` is set in `mdc_dd.cfg`.
- `data-format` describes the format if `date` or `time` is specified for the `data-type` attribute (see callout 6). If a value for `data-format` is specified, the instance should contain a validation regex to force a valid entry in the field (see callout 7).

In this example, `deploy-column` would be the first attribute if it were set. It is not, so all input for this item will be deployed to a database column. Next, the `searchable` attribute is specified as "t"; however, because this is the default value for the attribute, `searchable` need not be included here. Following `searchable` is `data-type`, here specifying the input to be stored as a string. If `date` or `time` is set as the value for the `data-type`, a `data-format` attribute should end the element.

**5. Instance (text):** The optional `<text>` element controls the length of text entry fields in metadata capture and search forms. It also controls whether an end user is required to enter text in a field. If the datatype is `date` or `time` and a format has been specified, it is best to include a `validation-regex` to force users to input data in the correct format (see callout 7). In this example the user is required enter a string of between 1 and 60 characters in the text field. The data entered for this item is stored in the database as `VARCHAR` and is searchable.

**6. DATE datatype:** If the datatype is set to `date` or `time`, it is recommended you specify a `data-format` and include a validation. Because there are many formats for date and time, specifying a format forces the user to enter data in that format and reduces the chance of user error. The value for `data-format` can be any valid Java format for a date or time.

**7. validation-regex:** The user can be forced to enter a date or time in the format you specify by including a validation regex. The value for the `validation-regex` attribute must match the format specified in for `data-format`. The regex in this example specifies the range of digits that can be entered for `yyyy-MM-dd` and that dashes must separate year, month and day. The following table shows validation regex examples for several supported datatypes. The `<database>` and `<text>` elements shown in the table are subelements of the `<item>` element. Some regex lines are wrapped due to formatting constraints. You should enter them all on one line in your configuration file.

Datatype	Notes	Example
DATE	If data-type is DATE, the data-format must be a format string that is valid for the Java simple date format class. Formats do not have to be year-month-day, any valid format will work.	<pre>&lt;database data-type="DATE" data-format="yyyy-MM-dd" /&gt; &lt;text maxlength="10"   validation-regex="^[0-9][0-9][0-9][0-9]-[0-1][0-9]-[0-3][0-9]\$" /&gt;</pre>
INT	Allows any integer up to 7 digits. This example assumes that you want to store data as integers, not dollars and cents.	<pre>&lt;database data-type="INT" /&gt; &lt;text maxlength="7"   validation-regex="^[0-9]\\{0,\\}\$" /&gt;</pre>
REAL	Allows any decimal up to 8 digits (including decimal). The regex allows 0 or more digits, followed by a decimal point, followed by zero or more digits.	<pre>&lt;database data-type="REAL" /&gt; &lt;text required="t" maxlength="8"   validation-regex="^[0-9]\\{0,\\}\\.[0-9]\\{0,\\}\$" /&gt;</pre>

### Sample datacapture.cfg File 2

The following `datacapture.cfg` file is written to work with the file shown earlier in “Sample metadata-rules.cfg File 2” on page 209.

```

<!-- This config file defines rulesets for capturing data.
Rules are referred to by name in other config files, such as
metadata-rules.cfg. Rules contain "items"; one item is a single
(set of) data that is to be captured from the end user.
An item consists of one or more "instances". Each instance
encapsulates how to capture the data for the item, and each
instance defines an ACL that determines which (if any)
instance a particular user is allowed to use to enter the
data. Instances are text, textarea, radio, checkbox, select,
and replicant. (Others are coming.)
Replicants are very special kinds of instances; they are
repeatable. Replicants contain _items_ instead of just an ACL
like the other types of instances.
-->
<data-capture-requirements type="metadata">
  <ruleset name="Default">
    <item name="Author">
      <database data-type="VARCHAR(12)" />
      <!-- This item is represented by a text box. -->
      <text size="12" required="t" />
      <!-- no ACL means open access for everyone -->
    </item>
  </ruleset>
  <ruleset name="Syndication">
    <item name="Category">
      <database data-type="VARCHAR(10)" />
      <!-- This item is represented by a series of four
checkboxes. -->
      <checkbox required="t" delimiter="/"> ← Instance (checkbox) with delimiter 1

      <!-- We want the TeamXpress extended attribute
to use "/" as the value delimiter when
concatenating all the selected values,
e.g., "Partners/Customers." -->

      <option label="Partners" />
      <option label="Suppliers" />
      <option label="Customers" />
      <option label="Internal" />

```



```

<ruleset name="PDF Files">
  <item name="All Keywords">
    <!-- All nested <item> elements must be "type compatible"
         if the fields are going to be deployed to a database. -->
    <database data-type="VARCHAR(20)" searchable="f" />
    <!-- Because any number of keywords may apply to a
         single file, we use a replicant instance for
         the "keywords" item. -->
    <replicant default="3" min="1" max="12"> ← Instance (replicant) 2
      <!-- We allow from 1 to 12 keywords. -->
      <!-- This replicant instance contains just one item,
           which has two instances. -->
      <!-- When there are multiple instances, the first
           instance whose ACL allows the current user
           will be the instance used for that user. -->
      <item name="Keyword">
        <text size="20" required="t">
          <!-- This ACL allows "joe" and masters
               to type anything she wants. -->
          <allowed> ← Access Control Limiter (ACL) 3
            <or>
              <cred user="joe" /> ← Access Identifier 4
              <cred role="master" />
            </or>
          </allowed>
        </text>
        <select required="t">
          <!-- Everyone but joe has to choose from
               pre-determined choices. -->
          <allowed>
            <not>
              <cred user="joe" />
            </not>
          </allowed>
          <option label="supply chain" />
          <option label="marketing" />
          <option label="sales promotions" />
          <option label="earnings" />
          <option label="facilities" />
          <option label="eCommerce" />
        </select>
      </item>
    </replicant>
  </item>

```

```

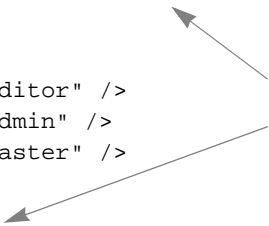
<ruleset name="MS Word Files">
  <!-- ... -->
</ruleset>
<ruleset name="Web Content">
  <!-- ... -->
</ruleset>
<ruleset name="HTML Files">
  <!-- ... -->
</ruleset>
<ruleset name="PR">
  <item name="Go-Live Date">
    <!-- The database column must allow strings like "Today" and
         "Tomorrow", so the DATE datatype cannot be used.
         This configuration decreases the usefulness of searching
         this database column -->

    <database searchable="f" data-type="VARCHAR(10)" />

    <!-- This text instance has a regular expression that
         determines validity of user-entered data.
         In this case, the regex requires the user
         to enter "##/##/####". -->
    <text size="10" validation-regex="^[0-9][0-9]/[0-9][0-9]/[0-9][
    0-9] [0-9][0-9]$" >
      <allowed>
        <or>
          <cred role="editor" />
          <cred role="admin" />
          <cred role="master" />
        </or>
      </allowed>
    </text>
    <select required="t">
      <option label="Today" />
      <option label="Tomorrow" selected="t" />
      <option label="Next Week" />
      <allowed>
        <cred role="author" />
      </allowed>
    </select>
  </item>
</ruleset>
<ruleset name="Corporate">

```

Variable Instances <sup>5</sup>



### Sample `datacapture.cfg` File 2 Notes

The following information is specific to the example file shown in the preceding section. For more detailed information about `datacapture.cfg` files, see the *TeamXpress Templating and Deployment Guide*.

- 1. Instance (checkbox or select) with delimiter:** Specifies the delimiting character used when data from all check boxes is concatenated by the data capture subsystem. The default delimiter is a comma (.). In this example a “/” is used as the delimiter to separate the concatenated values for the checked `<option>` elements.
- 2. Instance (replicant):** Specifies a repeatable instance that can contain multiple nested items and instances. When there are multiple instances, the first instance whose ACL allows the current user to enter data will be the instance used for that user. `<replicant>` is the only instance that can contain nested items and instances. Whenever additional iterations of the instance can be displayed (i.e, if the `max` threshold has not yet been reached), an **File > Add Above** and **File > Add Below** menu items are active. Whenever iterations of the instance can be removed (i.e, if the `min` threshold has not yet been reached), The **File > Delete** menu item is active. If a `<replicant>` has four items, the **Add** menu item displays another set of four items in the data capture form. In this example, if the user’s username is `joe` or role is `master`, three keyword text fields display; if the user is not a master or “joe”, then three drop-down selection boxes display. Keyword instances can be added or removed to a minimum of one and a maximum of twelve using the **Add** or **Delete** options in the **File** menu.
- 3. Access Control Limiter (ACL):** The `<allowed>` element lets you set an ACL to specify which users can or cannot use a specific instance to enter data. If `<allowed>` is not set, the instance is visible to and can be used to input data by any user. The `allowed` element can have any of the following elements:
  - `<cred>` lets you name a user or role in the ACL.
  - `<and>` defines multiple users or multiple roles that can use the instance.
  - `<or>` defines users and roles that can use the instance.
  - `<not>` defines a user or role that is not allowed to use the instance. The instance does not display for users not allowed to use that instance.

In this example **Keyword** text fields display for the user with username `joe` or the role `master`; for other users only **Keyword** selection drop-down menus display.



- 4. Accessor Identifier:** The `<cred>` element is a child element of `<and>`, `<not>`, or `<nor>` and lets you identify the accessor by role and username. Note that `<cred>` requires exactly one attribute, either `role` or `user`. In this example two `<cred>` elements are combined under the parent `<and>` element so that the ACL applies to both the username `joe` and role `master`. Text fields display for users with either identification, while drop-down selection menus display for others.
- 5. Variable Instances:** Within an `<item>` an instance can be made available to certain users or roles and not to others. In this example all Editors, Administrators, or Masters are offered text fields and can input variable text strings, while Authors are offered drop-down menus with predefined choices. Note that because there is one `<database>` element for each `<item>`, and that input for this item could be either dates or a character string, the datatype must be set to `VARCHAR`. It is not recommended that you create a `VARCHAR` database in which numerical data, such as dates or time, might be stored; operands useful for retrieving dates and time such as “between”, “less than”, “greater than” cannot be used to search such information.

### Adding Metadata Capture to the TeamXpress GUI

Because metadata capture is a file-specific feature, it is recommended that end users access it via the **File** menu in the TeamXpress GUI. To add a **Set Metadata** item to the TeamXpress GUI's **File** menu, add the following line to the `[iwcgi]` section of `iw.cfg`:

```
custom_menu_item_metadata="File", "Set Metadata", "iwmetadata.cgi" "all"
"width=800,height=570,scrollbars=yes,resizable=yes"
```

This line specifies the following:

- The TeamXpress GUI menu (**File**) to which the item will be added.
- The name of the new item (**Set Metadata**).
- The CGI (`iwmetadata.cgi`) that will execute when the item is selected.
- Which users (`all`) can see the menu item.
- The appearance and behavior of the window in which the CGI runs.

See “Custom Menu Items” on page 74 for more information about adding and enabling custom menu items.

## Metadata Capture End Result

After you configure metadata capture, end users can access it via the TeamXpress GUI to set metadata on files. The end result of a metadata capture session is the addition of TeamXpress extended attributes to one or more files. For example:

**File:** /default/main/dev/www/WORKAREA/jk/pr/BigAnnouncement.html

Name	Value
TeamXpress/Metadata/Author	jk
TeamXpress/Metadata/Go-Live Date	07/04/2000

**File:** /default/main/dev/syndication/WORKAREA/bill/fall2000.pdf

Name	Value
TeamXpress/Metadata/Author	bill
TeamXpress/Metadata/Category	Partners/Customers/Internal
TeamXpress/Metadata/Category/Partners	Y
TeamXpress/Metadata/Category/Customers	Y
TeamXpress/Metadata/Category/Internal	Y
TeamXpress/Metadata/Keywords/0/Keyword	supply chain
TeamXpress/Metadata/Keywords/1/Keyword	earnings
TeamXpress/Metadata/Keywords/2/Keyword	eCommerce

These are the extended attributes that would be displayed via the TeamXpress GUI from the **File > File Properties** menu. These extended attributes can now be deployed to a database via DataDeploy. the deployment can be manual, or automatic through Database Auto-Synchronization (DAS). See the *TeamXpress Templating and Deployment Guide* for more information.

## Metadata Capture and TeamXpress Workflow

The following sections describe key interactions between metadata capture and TeamXpress workflow.

### Specifying Files in Workflow Tasks

Metadata capture includes the ability to self-filter a list of files on which to capture metadata. For example, a user task or a job task can name a set of files upon which metadata will be set. All symlinks, directories, and deleted files that are part of the file set will be filtered out and ignored. Only actual files will have metadata set.

### Initiating Metadata Capture from a Job Specification File

The following sample `<cgitask>` section from a job specification file shows the syntax necessary to initiate a typical metadata capture process from within a job. The task owner in this example is `jk`. The task in this example is associated with the area shown in `areavpath`. See “`<cgitask>` Element” on page 192 for the `<cgitask>` DTD and other general information.

```
<cgitask name="metadata" owner="jk">
  <description>apply metadata.</description>
  <areavpath v="/default/main/dev/test/WORKAREA/jk" immediate
="+" />
    <successors>
      <successorset description="set">
        <succ v="confirm" />
      </successorset>
    </successors>
    <command v="/iw_cgi_wrapper.cgi/iwmetadata.cgi" />
    <activation>
      <or>
        <pred v="start" />
      </or>
    </activation>
</cgitask>
```

## Configuring Metadata Search

The following sections describe:

- A metadata search overview.
- The prerequisites for configuring metadata search.
- The main components that make up metadata search.
- How to configure metadata search.

### Overview

The metadata search subsystem uses search parameters supplied by an end user via a search form to query a database containing metadata. The end result is a list of files, displayed in the TeamXpress GUI, that contain metadata tags matching the search parameters. The search form is based on configuration files also used by metadata capture and generated by DAS. This relationship ensures that the search form contains fields only for data that is already stored in the metadata database. Details about these files are presented later in this section.

Metadata search is an area-specific feature. That is, it performs a search for metadata tags on files in the entire area and all subareas from which it was executed. For example, if you execute metadata search from `/default/main/dev/www/WORKAREA/w1`, all files in `w1` and its subdirectories are searched. If you execute metadata search from `/default/main/dev/www/WORKAREA/w1/marketing`, all files in `marketing` and its subdirectories are searched. You can execute metadata search from a workarea or any subdirectory within a workarea. You cannot execute it from a staging area, edition, or branch. See the *TeamXpress User's Guide* for more information about metadata search usage.

## Prerequisites

It is essential that you configure the following features before configuring and running metadata search:

- Metadata capture, as described earlier in this chapter. This is required because metadata search relies on the same `datacapture.cfg` file as metadata capture. If this file is not configured correctly, metadata search will not run.
- DataDeploy's Database Auto-Synchronization (DAS) module as described in the *TeamXpress Templating and Deployment Guide*. This is required because metadata search relies on the `mdc_dd.cfg` file, which is generated automatically when DAS is configured. If DAS is not configured correctly, metadata search will not run.

In addition, you must already have deployed metadata to a database via DAS before running metadata search. This is required because metadata search searches the database specified in the DAS configuration files. It does not search the TeamXpress backing store or any database not specified in the DAS configuration files.

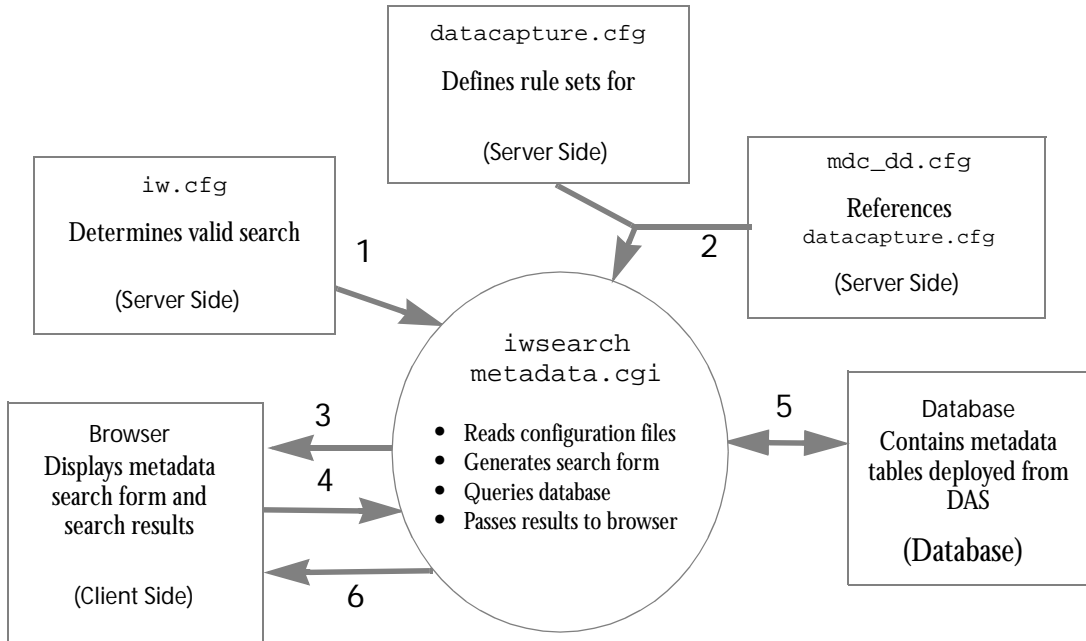
If all of these prerequisites are met, you can proceed with the metadata search configuration as described in "Configuring Metadata Search" on page 231. It is recommended that you read the following "Components" section before performing the configuration.

## Components

Metadata search relies on six main components:

- The same `iw-home/local/config/datacapture.cfg` configuration file used by the metadata capture subsystem.
- The DataDeploy configuration file `iw-home/local/config/mdc_dd.cfg`, which is generated automatically when you configure DataDeploy Database Auto-Synchronization (DAS) or when you execute the `iwsyncdb.ipl -mdcddgen` command.
- The metadata search CGI `iwsearchmetadata.cgi`, which interprets data from end users and rules in `datacapture.cfg` and `mdc_dd.cfg`, and produces browser graphics and prompts.
- The `[valid_search_paths]` section of `/etc/iw.cfg`.
- A browser interface for end-user input.
- The database containing metadata deployed via DAS.

The following diagram shows how these components work together. Sections following the diagram explain each diagram step and component in detail.



### Diagram Key

1. The metadata search CGI reads the `[valid_search_paths]` section of the `/etc/iw.cfg` configuration file to determine the paths where metadata search is valid. By default, all paths are considered valid search paths. See “Changing Valid Search Paths” on page 233 for more information.
2. The metadata search CGI reads the `mdc_dd.cfg` and `datacapture.cfg` configuration files. The information from these files is used by the search CGI to determine what should be displayed in the metadata search form. The information provided by `mdc_dd.cfg` controls whether a metadata field is searchable, what label each field has, and which operators are valid for each field in the search form. For example, if a metadata tag residing on the database uses a data type of `CHAR`, the

search form will contain operators such as `Contains`, `Does contain`, etc. for that specific field. The end user can then select one of these field-specific operators to set the search parameters for that field. However, if a metadata tag uses a data type of `INTEGER`, the search form will contain operators such as `Equals`, `Does not equal`, etc. for that specific field. By using both `data-capture.cfg` and `mdc_dd.cfg`, the search CGI ensures that each field in a search form will always contain the appropriate set of operators from which an end user can choose.

See “Configuring Metadata Search” on page 231 for more information.

3. The metadata search CGI displays the search form on the client system via the GUI.
4. An end user fills in search parameters and submits the search form back to the metadata CGI.
5. The metadata search CGI constructs the appropriate query statements and queries the database.
6. The search results are displayed in the TeamXpress GUI.

## Configuring Metadata Search

You must perform two main activities to configure metadata search:

1. Ensure that metadata capture and DAS are synchronized.
2. Add a **Search Metadata** item to the TeamXpress GUI so that end users can access metadata search.

The following sections describe these steps in detail. Additional configuration information is included in subsequent sections in case you need to customize the metadata search form or other characteristics of metadata search.

## Synchronizing Metadata Capture and DAS

Metadata search relies on correct synchronization of metadata capture and DAS. If these two features are not synchronized, metadata search will not run correctly. The issue is as follows:

When you configure DAS, you execute the `iwsyncdb.ipl -initial` command. This command generates several files, including `mdc_dd.cfg`. The `mdc_dd.cfg` file is in turn based on information from `datacapture.cfg`. The `datacapture.cfg` file must be configured specifically for metadata capture at your site to ensure that `mdc_dd.cfg` is generated correctly for use with metadata capture and search at your site. Therefore, it is *essential* that `mdc_dd.cfg` be generated *after* you set up `datacapture.cfg` in `iw-home/local/config` as described earlier in this chapter. There are two ways to ensure that this is the case:

1. Configure metadata capture as described earlier in this chapter before configuring DAS as described in the *TeamXpress Templating and Deployment Guide* or
2. Execute the following command if DAS was already configured prior to your configuring `iw-home/local/config/datacapture.cfg`:

```
iwsyncdb -mdcddgen [-force]
```

You can execute this command whenever you need to resynchronize DAS and metadata capture/search. See the *TeamXpress Templating and Deployment Guide* for details about `iwsyncdb` usage.

**Note:** Regenerating `mdc_dd.cfg` overwrites the existing version of the file, including any changes you might have made to it.

## Adding Metadata Search to the TeamXpress GUI

Because metadata search is an area-specific feature, it is recommended that end users access it via the **View** menu in the TeamXpress GUI. To add a **Search Metadata** item to the TeamXpress GUI's **View** menu, add the following line to the `[iwcgi]` section of `/etc/iw.cfg`:

```
custom_menu_item_metadata="View", "Search Metadata",  
"iwsearchmetadata.cgi" "all"  
"width=800,height=570,scrollbars=yes,resizable=yes"
```

Due to space limitations, this line appears to wrap. The line in the configuration file should not wrap.



The preceding line specifies the following:

- The TeamXpress GUI menu (**View**) to which the item will be added.
- The name of the new item (**Search Metadata**).
- The CGI (`iwsearchmetadata.cgi`) that will execute when the item is selected.
- Which users (`all`) can see the menu item.
- The appearance and behavior of the window in which the CGI runs.

See “Custom Menu Items” on page 74 for more information about adding and enabling custom menu items.

### Changing Valid Search Paths

Valid paths for metadata search are set in the `[valid_search_paths]` section of `/etc/iw.cfg`. By default, all paths are searchable. You can use regular expressions to specify that only certain paths are searchable. See comments in `iw.cfg` for more information.

### Making Individual Fields Non-Searchable

You can specify whether any field in a DCT is searchable. By default, all fields are searchable. To make a field non-searchable, specify `searchable="f"` the `<database>` element for that field in `datacapture.cfg`. If you make this change after `mdc_dd.cfg` was generated, you must regenerate it via the `iwsyncdb -mdcddgen` command.

**Note:** It is not advisable to edit `mdc_dd.cfg` itself. Such action could result in inconsistencies between DAS and metadata capture/search.



# Managing the TeamXpress Server

---

## Checking Server Status

### Verifying Server Operation

Verify that the TeamXpress server is running correctly by typing:

```
% /bin/ps -ef | grep iwserver | grep -v grep
```

You will see a response similar to this:

```
root 18309 18304 1 08:03:10 ? 2:56 /usr/iw-home/bin/iwserver
local/iw-store
```

If you do not see this response, the main TeamXpress process is down. Check the `iwserver.log` log file to see what happened. If TeamXpress seems to have died abnormally, run

```
% /etc/init.d/iw.server stop
```

to try to clear out the kernel module.

You can also run:

```
% modinfo | grep wfs
```

If anything returns, `wfs` is loaded. The number in the first column is the module id.

If wfs is loaded, stop the server with:

```
% /etc/init.d/iw.server stop
```

Finally, attempt to start TeamXpress with:

```
% /etc/init.d/iw.server start
```

## Checking for Multiple Servers

Only one server process should be running at any given time. If there are no processes running, then the server is not running. If more than one server process is running, there is a problem with the server and it should be restarted.

To reset the server to ensure that only one server process is running:

1. Issue the following command to stop the server:

```
% /etc/init.d/iw.server stop
```

2. Verify that all server processes have stopped. If not, manually kill any remaining processes. For more information on the `kill` command, consult a UNIX reference manual.

3. Restart the server with the following command:

```
% /etc/init.d/iw.server start
```

## Checking Request Handling

Verify the server is answering requests correctly by issuing the command:

```
% iw-home/bin/iwversion
```

You will see a response similar to this:

```
iwserver: 1.0.0 Build 1738 Interwoven 2000714
```

If the server does not respond or stops, then the server is not handling requests correctly. Restart the server, as described above.

## Verifying the Server Mount

Verify the server is mounted to the correct drive partition with the command:

```
% df -k | grep iwserver
```

The output should contain a line similar to this:

```
Filesystem          kbytes  used   avail  capacity Mounted on
servername:/iwserver 3141968 1542472 1285336 55%      /iwmnt
```

If the server does not respond properly, restart it as described above.

## Finding the Installation Directory

To find the TeamXpress installation directory, use the command-line tool (CLT) `iwgethome`.

### Usage:

```
iwgethome [-h|-o|-v]
```

- h Displays usage message.
- v Displays version.
- o Returns original factory setting value.

### Example:

```
% iwgethome
```

returns

```
/usr/iw-home
```

## Reviewing TeamXpress Logs

TeamXpress records events in TeamXpress log files as described below.

Log file	Default location	Contents
Installation log	<code>iw-home/install/iwinstall.log</code>	Record of the TeamXpress installation process.
Server log	<code>/var/adm/iwserver.log</code>	Record of the state of TeamXpress over time. Tracks when the TeamXpress server is started, stopped, mounted, etc. The location of this file is contained in <code>/etc/defaultiwlog</code> . You can also find this file using the CLT <code>iwgetlog</code> .
Trace log	<code>/var/adm/iwtrace.log</code>	Record of any irregularities on the TeamXpress server. The location of this file is contained in <code>/etc/defaultiwtrace</code> . You can also find this file using the CLT <code>iwgettrace</code> .
Event log	<code>/var/adm/iwevents.log</code>	Record of activities on TeamXpress. Tracks when files are submitted, published, branches created, etc., including DiskLow, Freeze, ShutDown, StartUp and Thaw events. Used with TeamXpress triggering scripts. The location of this file is contained in <code>/etc/defaultiwelog</code> . You can also find this file using the CLT <code>iwgetelog</code> .
Workflow log	<code>/var/adm/iwjoberrors.log</code>	Record of output from workflow runtime diagnostics.

## Monitoring the Server Load

The TeamXpress CLT `iwstat` returns a list of all current TeamXpress processes.

## Starting and Stopping the Server

To stop the TeamXpress server:

```
% /etc/init.d/iw.server stop
```

To restart the TeamXpress server:

```
% /etc/init.d/iw.server start
```

## Troubleshooting

### Repairing the Backing Store

The following section contains information about the backing store repair tools provided with TeamXpress. If you are experiencing problems with the TeamXpress backing store such as missing TeamXpress areas or missing file versions, use `iwfsck` to check the backing store. You can use `iwfsck -y` and `iwfsfix` to repair the backing store depending on the results of your backing store check.

## iwfsck

Diagnoses backing store problems and allows repair of some of the problems found.

### Usage:

```
iwfsck [-h] [-v] [-x|-xx|-xxx] [-l] [-y] [-b path] [-z]
[-d [[-s] | [-f] [-m] [-p]] [-r]]
[-o file] [-e file] [-t file] [-u file] [vpath]
```

- h Displays usage message.
- v Displays version.
- x Requests extra output and increments verbosity level. Prints additional information about what `iwfsck` is doing as it operates. Each `x` increments the verbosity level by 1. The highest level of verbosity is level 3 (`-xxx`). In the higher levels of verbosity, an extremely large quantity of output may be produced.
- l Prints output as HTML. This option is used by the `iwfsckcgi.cgi` program.
- y Repairs damaged files while running. In this mode, damaged files are deleted while `iwfsck` is running. The TeamXpress server must be down when specifying this option. If the TeamXpress server is running when this option is specified, a warning displays and this option is ignored.
- b *path* Uses *path* as the backing store location. The default is the configured backing store location returned by `iwgetstore` for the TeamXpress server.
- z Checks events in branches.
- d Checks directories and files in addition to the normal checking of branches and areas. All directories and files from the *vpath* are walked. If a *vpath* is not specified on the command line, the walk begins at the `/vpath`.



The following options are only allowed when `-d` is specified:

- f


Provides a fast reference check (not allowed with `-p`, `-m`, or `-s`). All references from the root are walked aggressively looking for missing references. If a missing reference is found, that part of the tree is marked *suspect*, and a more expensive walk with `vpaths` is done on that part of the tree to determine the directories and files affected by the problem.
- s


Provides a stack walk, which is slower but uses less memory than the default (not allowed with `-ε`). This mode uses the least amount of memory, but it is the least efficient for walking the entire tree of files and directories from the root.
- m


Checks ModLists for directories. A ModList is a data structure that is a shadow tree to the directory structure within a workarea. This shadow tree allows the modified files within a workarea to be determined quickly without having to traverse every file and directory within a workarea.
- p


Checks protopaths. A protopaths is a data structure that allows file names and history information to be determined without the expense of walking up to the root of an area through directories; however, it can be expensive.
- r


Checks parents. Parents and anti-parents are the reference counting mechanism used by the TeamXpress server. If zero parents are found for a file, it indicates a problem. It can be expensive.

The following options specify where output goes (note that `stdout` and `stderr` may be redirected in the normal way in a command line shell and that `-o` and `-e` are provided to allow redirection when shell redirection is not available):

<code>-o file</code>	Specifies output file for server startup information.
<code>-e file</code>	Specifies the file to write error messages to.
<code>-t file</code>	Specifies the file to write reports to.
<code>-u file</code>	Specifies the summary file.
<code>vpath</code>	Specifies the starting vpath to walk directories when <code>-d</code> is used.

### Examples:

To check areas and branches, issue the command:

```
% iwfsck
```

To check directories and files in addition to branches and areas, issue the command:

```
% iwfsck -d
```

Use the following command to check protopath and parents in addition to branches, areas, directories, and files. This command can be very resource intensive.

```
% iwfsck -d -p -r
```

### `iwfsckcgi.cgi`

This program provides an optional GUI interface to run `iwfsck`. You can access this interface through a browser:

```
server_name/iw-bin/iwfslogin.cgi
```

You will be prompted for the root or Administrator password. Once you have been authenticated, a screen will provide two choices:

- Perform content recovery
- Perform backing store check

To run `iwfsckcgi.cgi`, select **Perform backing store check**.

### **iwfsfix**

If `iwfsck` finds problems that cannot be repaired or the TeamXpress server is running when the backing store is diagnosed, it outputs lines in the format:

```
FIX iwfsfix repair args
```

The repairs and their arguments are shown below. To perform necessary repairs, copy the `FIX` line issued by `iwfsck` and paste it on the command line, with the word `FIX` removed.

There are also repairs for ModLists that must be performed when the TeamXpress server is running. On Solaris, these lines are in the format:

```
FIX /bin/touch junkfile; /bin/rm junkfile
```

`junkfile` is a uniquely named file that is created and removed from an affected directory.

The repairs that can be performed with `iwfsfix` are:

```
delete_tag branch_id tag_id
```

Removes the reference to a tag (lock) from a branch. This is done when the tag point itself is missing.

```
delete_tag_and_point branch_id tag_id
```

Deletes the reference to a tag (lock) from a branch and removes the tag point itself. Generally this is done when a tag duplicates or conflicts with another tag within a branch.

```
delete_direntry directory_id diritems_index filename
```

Deletes the directory entry for a damaged or missing file.

```
replace_direntry directory_id diritems_index filename new_standin_id
```

Repairs a directory entry to point to a correct standin ID.

```
delete_area area_id
```

Deletes the point for an area.

```
delete_area_from_branch branch_id area_id workarea / edition
```

Deletes the reference to an area (workarea or edition) from a branch. This cannot be done on a staging area because a branch by definition always contains a staging area.

```
null_previous point_id
```

Sets to null (-1) the PreviousPoint reference within a point. This is done when the PreviousPoint reference for a point is incorrect.

```
clone_diritems directory_id diritems_index new_gen_id new_dot_dot
```

Clones a set of directory items within a directory to create a new set. This is done when a set of directory items is shared between areas, but it should not be shared.

## Managing Server Resources

### Disk Space

#### TeamXpress Data Store

The location of the TeamXpress backing store is contained in `/etc/default/iwstore` (alternatively, you can use the CLT `iwgetstore` to find this location). This file can only contain one address at a time. Changing the address contained in this file changes the location of the backing store.

To move the TeamXpress backing store:

1. Shut down the server.
2. Move the entire backing store to the new location.

3. Edit `/etc/defaultiwstore` so that it now points to the new location. **Note:** Never modify `/etc/defaultiwstore` while the TeamXpress server is running.
4. Restart the TeamXpress server.

To create a new, empty, TeamXpress backing store:

1. Shut down the TeamXpress server.
2. Edit `/etc/defaultiwstore` to point to an empty directory on a partition with enough space for the TeamXpress backing store.
3. Restart the TeamXpress server.

TeamXpress will create a new (empty) backing store when you restart the server. The new backing store will contain only a main branch, which will have an empty initial edition, a staging area, and no workareas. You can populate this new backing store with content just as you did your old one. If you point `/etc/defaultiwstore` back to the older backing store, all the content contained in the older backing store will reappear.

### Checking Disk Space Usage

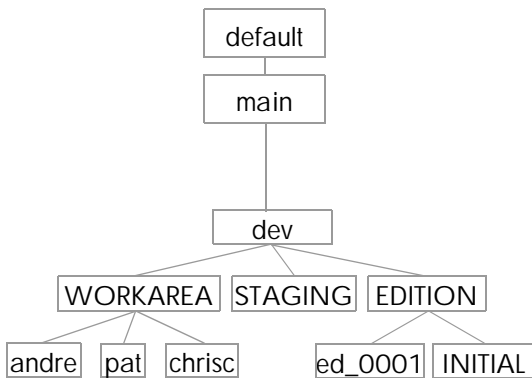
When checking disk space usage, you can use

```
% df -k `cat /etc/defaultiwstore`
```

to check the size of the directory contained in `/etc/defaultiwstore` or `/iwmnt`. That is, you can check the size of either the backing store *or* the mount point. Although the mount point contains many virtual copies of files in workareas, staging areas, and editions, `df -k` will only check the actual disk space used.

## TeamXpress File System Mount

The TeamXpress file system mount contains a file system view of all the branches, workareas, staging areas, and editions on the TeamXpress server. TeamXpress areas do not contain physical copies of the entire Web site, but rather pointers to the files contained in the Web site. The only physical files contained within TeamXpress areas are the files that have actually been modified in those areas. That is, the only files actually contained in a workarea are those files that have been modified in that workarea but not yet submitted; the only files contained in the staging area are the files that have been submitted since it was last published; the only files in an edition are the files that have changed since the previous edition was published.



*Sample TeamXpress file system structure*

Each branch contains three directories: `WORKAREA`, containing all the workareas on the branch; `STAGING`, containing the staging area for the branch, and `EDITION`, containing all editions on the branch. It may also contain directories that hold sub-branches. In the example above, the `main` branch contains one workarea, a staging area, an initial edition, and a sub-branch (`dev`). The sub-branch contains three workareas (`andre`, `pat`, and `chris`), a staging area, and two editions.

Although many of the files contained within this file system structure are virtual, they can be treated as if they were real. They will appear to exist even when you run links checkers and scripts against them. However, staging areas, editions, and container directories (e.g. `WORKAREA`, `EDITION`, `main`, or `dev`) are all read-only. Only workareas can be written to.

### Compression and Recovering Disk Space

TeamXpress supports edition compression, but due to the small number of actual files contained in a typical edition and the fact that many Web site files are already in a compressed format (e.g., GIFs and JPEGs), this procedure might only recover small amounts of disk space. You can also delete old editions, which will delete all files actually contained in that edition, in addition to all intermediate submissions between publication of editions.

### Routine Maintenance: Metadata Forking

Metadata forking conserves disk space by reducing the number of files whose content is duplicated throughout the TeamXpress backing store. That is, if you have an old version of a file in one branch, and an identical file version on another branch, the same data may appear twice in the backing store. Metadata forking eliminates this type of duplication. This operation results in no user-visible changes to the TeamXpress virtual file system. For example, file histories are unchanged.

To use metadata forking, run the `iwfsshink` utility. The `iwfsshink` utility may be run while the TeamXpress server is running; however, TeamXpress may experience some performance degradation while it is running. Also, `iwfsshink` may not remove all duplicates (for example, it will not remove any duplicates created by TeamXpress users while the utility is running).

1. Issue the `iwfsshink` command:

```
% iwfsshink run
```

2. The utility may take several hours to run. Use the `status` option to view the current status. You can also pause the operation with the `pause` option, then restart it with the `run` option. See “`iwfsshink` Syntax” for a full list of options.

The `iwfsshink` utility should be run every few months.

**iwfsshink Syntax**

<code>iwfsshink [-h] [-v] [run   pause   abort   status]</code>	
<code>-h</code>	Displays usage message.
<code>-v</code>	Displays version.
<code>run</code>	Starts the <code>iwfsshink</code> process.
<code>pause</code>	Temporarily stops the <code>iwfsshink</code> process. It can be restarted with the <code>run</code> option. Because <code>iwfsshink</code> takes a long time to run, you may want to start it during off-hours. When activity increases, you can pause it until the next period of inactivity.
<code>abort</code>	Terminates the <code>iwfsshink</code> process.
<code>status</code>	Shows information about the latest <code>iwfsshink</code> process.

**Examples:**

```
% iwfsshink status
```

when `iwfsshink` has finished running, returns a message similar to:

```
Not currently running.  
Last started Mon Jun 26 15:47:53 2000  
Last completed Tue Jun 27 00:40:04 2000  
Files examined: 317974  
Bytes examined: 75936814830  
Files found to be duplicates: 233430  
Files converted: 198352  
Bytes removed: 23455046531
```



### **Moving the Backing Store and Removing Old Versions**

If you are running out of disk space and `iwfsshrink` doesn't recover enough extra space, you might need to move the TeamXpress backing store (see page 244). The TeamXpress backing store must reside on a single logical volume, e.g., a single disk or an array of disks.

Alternatively, if you have unused branches in TeamXpress, you can delete these branches to recover disk space.

Over time, individual branches take up more and more disk space, as the number of versions and files on the branch grows. If you do not need any of your old version history, you can create a new (empty) branch, create a workarea, copy all the old content into the workarea, then delete the old branch. Exercise extreme caution when doing this, as all versioning and metadata information will be irrevocably lost.



# Backing Up TeamXpress

---

The TeamXpress backing store represents a tremendous investment in resources and is a valuable corporate asset. As such, it should be backed up daily, or even more frequently, to minimize the possibility of damaged or lost data.

Any backup mechanism that can guarantee exact time and state directory content recovery can be used effectively for this purpose. It is recommended you use professionally supported third-party backup solutions for this purpose rather than the basic `iwbackup` and `iwrestore` utilities provided with TeamXpress. These are provided for use only when a better solution is not yet available at a specific installation.

## Integrating with Third-Party Backup Solutions

It is recommend you use a high quality third party backup solution for protecting the backing store data. When evaluating a backup solution, the following criteria are essential:

- The backup method must provide a way to perform an `iwfreeze` operation prior to performing the backup. This must be done to assure that the backing store does not change during the backup. The backup method must then perform an `iwfreeze --` operation to allow writes to the backing store when the backup is finished.
- The backup method must be fast enough to perform a full or incremental backup of the backing store within a reasonable length of time. The maximum allowable length of time depends on the requirements of the particular installation, but should probably be less than 12 hours.
- The restore method must provide a way to do a complete state-restore of a directory as of a given time. This means that when a directory is recovered, the contents must match exactly what was in the directory at the time the backup was performed. Only files that were present at the time of the backup must be present in the restore. That is, if a file was deleted from the original directory

between backups, it should not be present in the restore. Some backup and restore products regard all backed-up files to be “sticky,” i.e., as long as a file ever existed, it will be present in the restoration regardless of whether it was deleted prior to the last backup.

Additional criteria to consider are:

- An automated backup execution facility capable of performing full backups followed by level (preferred) or incremental backups to provide a customizable backup strategy.
- Automated backup media management and manipulation (e.g., a tape jukebox or silo).
- The ability to make copies of completed backups for offsite storage.

If the available backup method is efficient and inexpensive (compared to the value of the data being protected), the TeamXpress workareas can also be backed up to allow users to recover individual files or directories from their workareas, rather than having to recover the entire backing store. This is a very convenient feature for users, but can come at a relatively high price in terms of extra time and space needed for these redundant backups. Although the virtual files which comprise much of TeamXpress’s file system mount (`/iwmnt`) take up no extra space on the TeamXpress server, if the actual TeamXpress workareas are backed up, the virtual files in the workareas will be treated as actual files and will take up space in the backup media.

It is not absolutely necessary to freeze the TeamXpress backing store while you are backing up workareas; however, failure to freeze the backing store while you are backing up the backing store itself can result in possible data loss and corruption.

Backing up workareas alone is not a substitute for backing up the TeamXpress backing store. If you only back up the files that appear in the TeamXpress file system mount, you will lose important metadata such as version histories and file status. Always back up the actual TeamXpress backing store whether or not you back up individual workareas.

## Suggested Strategies for Incremental Backups

It is possible to implement a “level-oriented” backup if a sufficiently sophisticated backup solution is available. For example, a full backup can be performed on the first Saturday of the month, then incremental backups that build on each other can be performed for the rest of the week. On the second Saturday of the month, a “super-incremental” backup based on the original full backup done on the first Saturday is performed. The super-incremental backup supersedes all of the previous incremental backups. Only the first full backup and super-incremental are needed to completely recover the backing store. For the subsequent week, incremental backups are again performed based on the super-incremental backup done on the second Saturday. The following Saturday, another super-incremental backup based on the previous super-incremental file is performed, again eliminating the need for the previous week’s incrementals to recreate the backing store. To perform a recovery at this point, restore the original full backup, then each super-incremental in sequence, and finally the balance (if any) of the current week’s incrementals.

This tiered, or level-oriented backup can be repeated on a monthly basis to produce a week-by-week archive of the backing store. To reproduce the backing store as of any particular Saturday, recover the full backup from the beginning of the month, then apply each Saturday backup in turn until the desired Saturday is reached.

To determine your optimal backup strategy, you must analyze the tradeoffs of convenience and speed in backing up versus simplicity and speed of restoration, and decide what best suits your needs. A strategy using a single full backup and an indefinite string of incrementals is optimized for backup speed, but the amount of time required to perform a full recover of the backing store grows with each passing day as a new incremental is added to the list. Every backup must be preserved to be able to recover the backing store. One benefit of this method is that a complete daily archive of the backing store will be preserved.

The opposite extreme is to perform a full backup every day. Each backup will take the maximum amount of time to perform, but only one recover needs to be done to completely recreate the backing store. If you only preserve the previous day’s backup, no history of the backing store will be retained, but the amount of storage space used by the backups is minimized.

## Performing Full Backups with iwbackup

A complex backup strategy using `iwbackup` will require a lot of customization, preparation and human intervention. `iwbackup` is incapable of spanning backup volumes automatically. This necessitates splitting up large backing stores into smaller chunks for backups. Automation of backups will require the development of an automation mechanism. For these reasons, it is again recommended that you implement a full-featured third party backup solution to protect the backing store.

If `iwbackup` is the only solution available, the following guidelines are provided.

Before any full or incremental backup of the TeamXpress backing store is performed, `iwfreeze` must be used to prevent modifications during the backup process, in order to save a coherent image of the backing store. When the backup is finished, `iwfreeze` can unfreeze TeamXpress. `iwbackup` produces a data stream on `stdout` that is suitable for writing to a disk file, tape or any other medium capable of sequential access.

A good method for using `iwbackup` would be in a script invoked by `cron` during a time of day when no one normally makes modifications to the backing store. The script would automatically invoke `iwfreeze`, then use `iwbackup` to write the backup data to a tape or to disk, and finally invoke `iwfreeze` again to unfreeze TeamXpress for production use.

### Usage:

```
iwbackup -full log path
```

<code>-full</code>	Perform a full backup of the backing store.
<code>log</code>	Path to write log file for this backup.
<code>path</code>	Location of backing store to be backed up.

It is *extremely* important to save the `log` file. It contains a description of the backing store and is required for doing a restore with `iwrestore`. If you lose this file, you will be unable to perform incremental backups or recover your backing store. It is recommended that you make multiple copies of it and store them in separate locations.

## Performing Incremental Backups with `iwbackup`

After the backing store has been fully backed up, it is not necessary to do another full backup unless checkpoint archives are desired, or to collapse a previous cycle of full-plus-incremental backups into a single new full backup for convenience. Note that the more incremental backups performed since the last full backup, the longer and more complicated the recovery process will be.

### Usage:

```
iwbackup -incremental prev new path
```

<code>-incremental</code>	Backs up incrementally, relative to <code>prev</code> .
<code>prev</code>	Log file from previous backup.
<code>new</code>	Path to write log file for this backup to.
<code>path</code>	Path of backing store to be backed up.

A simple but effective backup strategy would be to perform a full backup once a week, then perform an incremental backup every day. Previous weeks' backups need not be retained except for historical reference.

## Restoring with `iwrestore`

Recovering a backing store with `iwrestore` is conceptually simple. In the simplest case of a single full backup followed by sequential incrementals, the full backup is recovered first, then each incremental backup is applied in chronological order (oldest to newest).

If many weeks separate the full backups, the task of restoring dozens (or even hundreds) of incremental backups will become impractical. This is why it is recommended that in the simple case of a full backup followed by sequential incrementals, no more than a few weeks should be allowed to separate the full backups.

**Usage:**

```
iwrestore [-full | -incremental] path < archive_file
```

<code>-full</code>	Restore from a full backup.
<code>-incremental</code>	Restore from an incremental backup.
<code><i>path</i></code>	Path of backing store that was backed up (see <code>iwbackup</code> ). Must exist, or <code>iwrestore</code> will fail.
<code><i>archive_file</i></code>	Backup archive created by <code>iwbackup</code> , read from <code>stdin</code> .

If a more complex backup strategy is used, as in the “super-incremental” example described earlier, the incremental backups performed between the super-incremental backups can be ignored during a restore. Only the incrementals performed since the last super-incremental need to be applied. This strategy is a trade-off between backup speed and recovery speed. Backups would be faster in the case of a “full-plus-sequential-incrementals” strategy, but recovery would become impossibly time-consuming after just a few weeks. Recoveries will be simple in the case of a “nothing-but-full” backup strategy, but full backups are very time- and space-consuming.



# TeamXpress Configuration Files

The following files contain information about your TeamXpress server configuration:

Configuration File	Function
<code>/etc/defaultiwhome</code>	Describes the location of the TeamXpress application software. The installation default value is <code>/usr/iw-home</code> .
<code>/etc/defaultiwstore</code>	Describes the location of the TeamXpress server backing store directory. The installation default value is <code>/usr/iw-store</code> .
<code>/etc/defaultiwmount</code>	Describes the location of the TeamXpress virtual mount point. The installation default value is <code>/iwmnt</code> .
<code>/etc/defaultiwlog</code>	Describes the location of the <code>iwserver.log</code> file. The installation default value is <code>/var/adm/iwserver.log</code> .
<code>/etc/defaultiweventlog</code>	Describes the location of the <code>iwevents.log</code> file. The installation default value is <code>/var/adm/iwevents.log</code> .
<code>/etc/defaultiwtrace</code>	Describes the location of the <code>iwtrace.log</code> file. The installation default value is <code>/var/adm/iwtrace.log</code> .
<code>iw-home/etc/iw.cfg</code> (default location—see below for more information)	Contains various parameters necessary for the operation of TeamXpress, as described in the Chapter 4, “Configuring the TeamXpress Server.”
<code>iw-home/iw-samba/lib/iw.smb.conf</code>	Contains Samba configuration.
<code>iw-home/local/config/submit.cfg</code>	Specifies all file permissions that will automatically be changed at submit time.

Configuration File	Function
<code>iw-home/local/config/autopriate.cfg</code>	Specifies what types of files will automatically be marked private.
<code>iw-home/local/config/templates.cfg</code>	Specifies which templates will be used in which TeamXpress areas.
<code>iw-home/conf/roles/master.uid</code>	Contains a list of all users who can log in as a Master user.
<code>iw-home/conf/roles/admin.uid</code>	Contains a list of all users who can log in as an Administrator.
<code>iw-home/conf/roles/editor.uid</code>	Contains a list of all users who can log in as an Editor.
<code>iw-home/conf/roles/author.uid</code>	Contains a list of all users who can log in as an Author.

The locations of most of these files can be changed (see “File Locations” on page 92).

If `iw.cfg` does not exist in the default location, TeamXpress will look for it in the following locations, in order:

`/etc/iw.cfg`

`iw-home/config/iw.cfg`

`iw-home/local/etc/iw.cfg`

`iw-home/etc/iw.cfg`

If `iw.cfg` is not found in any of these places, TeamXpress will assume the default values for `iw.cfg` settings.

# Sample Job Specification File

---

This appendix contains a sample job specification file as described in Chapter 5, “Configuring TeamXpress Workflow.”

The following job specification file could be created by direct editing (see “Job Creation: Directly Editing a Job Specification File” on page 140) or by configuring a workflow template file to generate it based on data provided by an end user (see “Job Creation: Setting up a Workflow Template File” on page 140). This file defines a workflow for this sequence of events:

1. A worker named Mark generates a set of documentation about a new product called B4000.
2. A worker named Bill then receives this documentation and prepares it for the Web.
3. Bill’s manager and the legal department review Bill’s and Mark’s efforts.
4. Material is submitted and pushed out to the live Web server.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE workflow SYSTEM "iwwf.dtd">

<!-- Sample workflow for B4000. -->

<workflow name="B4000" owner="BillsManager"
  description="Standard workflow for new product information.">
  <usertask name="MarkWork" owner="Mark"
    description="Write copy for B4000" start="t">
    <areavpath v="/default/main/dev/WORKAREA/Mark"/>
    <successors>
      <successorset description="Done">
        <succ v="MarkToBill"/>
      </successorset>
    </successors>
    <activation>
      <or>
        <pred v="BillToMark"/>
```



```
        <pred v="ReviewToMark"/>
    </or>
</activation>
</usertask>

<updatetask name="MarkToBill" owner="Bill"
    description="Update Bill's Workarea">
    <areavpath v="/default/main/dev/WORKAREA/Bill"/>
    <successorset>
        <succ v="BillWork"/>
    </successorset>
    <srcareavpath v="/default/main/dev/WORKAREA/Mark"/>
    <activation>
        <pred v="MarkWork"/>
    </activation>
</updatetask>

<usertask name="BillWork" owner="Bill"
    description="Webify this doc.">
    <areavpath v="/default/main/dev/WORKAREA/Bill"/>
    <successors>
        <successorset description="Done">
            <succ v="BillToReview"/>
        </successorset>
        <successorset description="Send back to Mark">
            <succ v="BillToMark"/>
        </successorset>
    </successors>
    <activation>
        <or>
            <pred v="MarkToBill"/>
            <pred v="ReviewToBill"/>
        </or>
    </activation>
</usertask>

<updatetask name="BillToReview" owner="Manager"
    description="Update the Review area from Bill's
        Workarea.">
    <areavpath v="/default/main/dev/WORKAREA/Review"/>
    <successorset>
```

```

        <succ v="LegalReview"/>
        <succ v="ManagerReview"/>
    </successorset>
    <srcareavpath v="/default/main/dev/WORKAREA/Bill"/>
    <activation>
        <pred v="BillWork"/>
    </activation>
</updatetask>

<usertask name="LegalReview" owner="Legal "
    description="Limit exposure." readonly="t">
    <areavpath v="/default/main/dev/WORKAREA/Review"/>
    <successors>
        <successorset description="Okay">
            <succ v="Submit"/>
        </successorset>
        <successorset description="Legal problem">
            <succ v="ReviewToMark"/>
        </successorset>
    </successors>
    <activation>
        <pred v="BillToReview"/>
    </activation>
</usertask>

<usertask name="ManagerReview" owner="Manager "
    description="Final Approval" readonly="t">
    <areavpath v="/default/main/dev/WORKAREA/Review"/>
    <successors>
        <successorset description="Okay">
            <succ v="Submit"/>
        </successorset>
        <successorset description="Send back to Mark">
            <succ v="ReviewToMark"/>
        </successorset>
        <successorset description="Send back to Bill">
            <succ v="ReviewToMark"/>
        </successorset>
    </successors>
    <activation>
        <pred v="BillToReview"/>

```



```
</activation>
</usertask>

<submittask name="Submit" owner="Manager"
  description="Final submission.">
  <areavpath v="/default/main/dev/WORKAREA/Review"/>
  <successorset>
    <succ v="Deploy"/>
  </successorset>
  <activation>
    <and>
      <pred v="LegalReview"/>
      <pred v="ManagerReview"/>
    </and>
  </activation>
</submittask>

<externaltask name="Deploy" owner="Manager"
  description="Deploy to live server.">
  <areavpath v="/default/main/dev/STAGING"/>
  <successors>
    <successorset description="Successful Deployment">
      <succ v="End"/>
    </successorset>
    <successorset description="Deployment failed">
      <succ v="End"/>
    </successorset>
  </successors>
  <command v="/scriptorium/do_deploy.pl"/>
  <activation>
    <pred v="Submit"/>
  </activation>
</externaltask>

<endtask name="End">
  <activation>
    <pred v="Deploy"/>
  </activation>
</endtask>
```

```

<!-- Various send back updates -->

<updatetask name="ReviewToBill" owner="Bill"
  description="Update Bill's workarea form the Review
              workarea.">
  <areavpath v="/default/main/dev/WORKAREA/Bill"/>
  <successorset>
    <succ v="BillWork"/>
  </successorset>
  <srcareavpath v="/default/main/dev/WORKAREA/Review"/>
  <activation>
    <pred v="ManagerReview"/>
  </activation>
</updatetask>

<updatetask name="BillToMark" owner="Mark"
  description="Update Mark's workarea from Bill's">
  <areavpath v="/default/main/dev/WORKAREA/Mark"/>
  <successorset>
    <succ v="MarkWork"/>
  </successorset>
  <srcareavpath v="/default/main/dev/WORKAREA/Bill"/>
  <activation>
    <pred v="BillWork"/>
  </activation>
</updatetask>

```



```
<updatetask name="ReviewToMark" owner="Mark"  
  description="Update Mark's workarea from Review">  
  <areavpath v="/default/main/dev/WORKAREA/Mark"/>  
  <successorset>  
    <succ v="MarkWork"/>  
  </successorset>  
  <srcareavpath v="/default/main/dev/WORKAREA/Review"/>  
  <activation>  
    <or>  
      <pred v="ManagerReview"/>  
      <pred v="LegalReview"/>  
    </or>  
  </activation>  
</updatetask>  
  
</workflow>
```



# Backing Store Conversion and Distribution

---

## About Backing Store Performance Enhancements

TeamXpress 1.0 includes several features that allow significant performance and stability to the TeamXpress backing store.

- Metadata consolidation reduces the number of inodes required, and may also conserve space in the backing store. It also improves performance and reliability.
- Backing store distribution allows your TeamXpress backing store to reside on multiple file systems, reducing the time required for backups and restoration.

## Consolidating Metadata

Metadata consolidation consolidates many small metadata files into a single metadata file. This reduces the number of inodes required, and may also conserve space in the backing store. To consolidate metadata, use the `iwfsconvert` CLT.

The `iwfsconvert` CLT has two modes: normal mode and transition mode. Normal mode converts the backing store while the TeamXpress server is stopped (this may take several hours, depending on the size of the backing store). Transition mode converts a copy of the backing store, so that you can still work while it is running. It logs changes made to the backing store and makes the necessary changes to the deltas.

While converting the backing store, you will need some extra disk space (approximately 25% of the size of your current backing store).

## Before You Begin

Before you convert the backing store, you need to check for metadata inconsistencies, determine whether you want to use normal mode or transition mode, and ensure that you have enough disk space.

1. Stop the TeamXpress server:

```
% /etc/init.d/iw.server stop
```

2. Check the backing store to make sure there are no metadata inconsistencies:

```
% iwfsck -d
```

3. Fix any inconsistencies using `iwfsfix`. For more information on the `iwfsck` and `iwfsfix` commands, see “Repairing the Backing Store” on page 239.

4. Issue the `iwfsconvert` command in pre-check mode to determine how long the conversion will take, and how much disk space you will need to perform the conversion. For example:

```
% iwfsconvert -p
```

```
Scanning: 100 of 100000
```

```
Scanning: 10 of 1000
```

```
Disk space needed: 10Mb
```

```
Free inodes needed: 256
```

```
It will take approximately 20 minutes to perform the conversion.
```

Make sure that you have at least as much free space as the pre-check mode indicates before beginning the conversion.

5. The TeamXpress server will need to be off-line for the entire conversion, so decide whether or not it is practical to have the TeamXpress server off-line for as long as the pre-check mode indicates it will take. If the TeamXpress server can be off-line for that long, use normal mode (see page 267). If it can't, use transition mode (see page 268).

## Converting the Backing Store Using Normal Mode

To convert the backing store using normal mode:

1. Stop the TeamXpress server:

```
% /etc/init.d/iw.server stop
```

2. Perform the conversion:

```
% iwfsconvert -c
```

The following is sample output:

```
Converting 100 of 100000
Converting 10 of 1000
Compacting metadata files...
Updating backing store version number...
Removing old format metadata files... 100
Removing old format metadata files... 10
```

**Note:** If you have not already run the pre-check, `iwfsconvert -c` will run it automatically. However, if you run the pre-check separately, you can make sure that you have enough disk space in advance.

3. If there are any bad points that are not converted, the `iwfsconvert` tool will list them, or you can use the `-b` option to see a list. For example:

```
% iwfsconvert -b
```

The following points in the default archive were not converted:

```
0x64 /local/iw-store/default/d0/d0/d0/f64_
0x1024 /local/iw-store/default/d0/d0/d10/f24
```

4. Use the `-f` option to fix any bad points that were not converted. For example:

```
% iwfsconvert -f
```

```
Fixing point (0x0, 0x1024): Success!
```

5. Restart the TeamXpress server:

```
% /etc/init.d/iw.server start
```

## Converting the Backing Store Using Transition Mode

To convert the backing store using transition mode:

1. Stop the TeamXpress server:

```
% /etc/init.d/iw.server stop
```

2. Copy the backing store to another location (in this example, the original backing store is located at `/local/iw-store` and it gets copied to `/local1/iw-store1`).

3. Issue the `iwfsconvert` command in transition mode. For example:

```
% iwfsconvert -t start -s /local/iw-store -d /local1/iw-store1
```

```
Entering transition mode for backing store /local/iw-store
```

4. Restart the TeamXpress server:

```
% /etc/init.d/iw.server start
```

5. Convert the copy, using the `-t convert` option. For example:

```
% iwfsconvert -t convert -d /local1/iw-store1
```

```
Converting 100 of 100000
```

```
Converting 10 of 1000
```

```
You have started an offline conversion and need to finish it  
before you can complete this process. Please run this program  
with the -t finish option to complete the conversion process.
```

```
Compacting metadata files...
```

If there are any bad points that are not converted, the `iwfsconvert` tool will list them, or you can use the `-t showbad` option to see a list, as described in “Troubleshooting Backing Store Conversion” on page 269.

6. When the copy backing store has been converted, stop the TeamXpress server again, as described in step 1.

## 7. Convert the differences between the original backing store and the copy:

```
% iwfsconvert -t finish -s /local/iw-store -d /local1/iw-store1
Converting 100 of 100000
Converting 10 of 1000
Compacting metadata files...
Updating backing store version number...
Removing old format metadata files... 100
Removing old format metadata files... 10
```

If there are any bad points that are not converted, the `iwfsconvert` tool will list them, or you can use the `-t showbad` option to see a list, as described in “Troubleshooting Backing Store Conversion” on page 269.

## 8. Restart the TeamXpress server, as described in step 4.

### Troubleshooting Backing Store Conversion

`iwfsconvert` will not remove any of your existing metadata files until it has successfully converted your entire backing store. If the conversion is interrupted, restart it using the `-r` option (or, for transition mode, the `-t restart` option):

```
iwfsconvert -r [-a archive]
iwfsconvert -t restart -s source -d destination
```

Sometimes `iwfsconvert` will be unable to convert a point, usually because it cannot be read. In this case, `iwfsconvert` will skip the point it can't convert and continue converting the rest of the backing store. When it finishes, it will list any points that it could not convert. To view this list again use the `-b` option or, for transition mode, the `-t showbad` option:

```
iwfsconvert -b [-a archive]
iwfsconvert -t showbad -s source -d destination
```

Once you have solved the problems which prevented the listed points from being converted, use the `-f` option to fix them (or, for transition mode, the `-t fix` option):

```
iwfsconvert -f [-a archive]
iwfsconvert -t fix -s source -d destination
```

You must make sure that all points are converted before attempting to restart the server.

## iwfsconvert Syntax

### Normal Mode Usage:

```
iwfsconvert [-h] [-v] [-V] [-p|-r|-b|-f|-c] [-a archive]
```

<code>-h</code>	Prints this message.
<code>-v</code>	Prints version.
<code>-V</code>	Verbose mode.
<code>-p</code>	Does pre-check. Not valid when used with <code>-t phase</code> .
<code>-b</code>	Lists bad points which were not converted.
<code>-f</code>	Fix mode. Attempts to convert points which were not converted on the first pass.
<code>-c</code>	Begins conversion. Automatically restarts conversion if already started.
<code>-a archive</code>	Specifies the root directory of the archive to convert.
<code>-r</code>	Restarts an interrupted conversion.

**Transition Mode Usage:**

```
iwfsconvert [-V] [-h] [-v] -t phase [-s source] -d destination
```

-h	Prints this message.
-v	Prints version.
-V	Verbose mode.
-t <i>phase</i>	Specifies the phase of the conversion process in transition mode, where <i>phase</i> is one of: <i>start</i> starts the conversion process in transition mode <i>convert</i> converts the backing store specified with -d <i>destination</i> <i>finish</i> finishes the transition mode conversion <i>restart</i> continues an interrupted conversion <i>fix</i> fixes unconverted points <i>showbad</i> shows a list of unconverted points
-s <i>source</i>	Specifies the path to the original backing store.
-d <i>destination</i>	Specifies the path to the copy of the backing store.

**Examples:**

The following command starts transition mode:

```
% iwfsconvert -t start -s /local/iw-store -d /local1/iw-store
```

The following command converts the copy of the backing store:

```
% iwfsconvert -t convert -d /local1/iw-store
```



The following command finishes the transition mode conversion:

```
% iwfsconvert -t finish -s /local/iw-store -d /local1/iw-store
```

The following command displays a list of unconverted points for both the source and destination copies of the backing store:

```
% iwfsconvert -t showbad -s /local/iw-store -d /local1/iw-store
```

## Distributing the Backing Store Across File Systems

TeamXpress allows you to distribute, or “stripe” its backing store across multiple file systems. This can greatly reduce the time required to back up and restore the backing store. It can also help alleviate some file system limitations, such as inode requirements and disk space limits. This feature is designed for use with backup systems that can back up multiple file systems in parallel.

**Note:** This is not a RAID system. Backing store distribution provides no redundancy, by itself. However, if you want to use RAID in conjunction with this feature, you can configure each file system as a RAID system.

**Note:** If any one of the file systems becomes unavailable, the entire TeamXpress backing store will be effectively unavailable.

The backing store distribution CLT, `iwfsstripe`, has three modes: initialization, normal, and transition.

Initialization mode (see page 273) will create an empty new backing store striped across multiple file systems. Use this mode if you want to use this feature but don't have an existing backing store to convert.

Normal mode (see page 275) creates an empty new backing store and populates it with the contents of an existing backing store. Use this mode if you want to stripe your existing backing store across multiple file systems.



Transition mode (see page 276) creates an empty new backing store and populates it with the contents of an existing backing store, with minimal down time during the conversion process (although extra down time may be required to make a copy of the existing backing store). Use this mode if you want to stripe your existing backing store across multiple file systems, and you can make a copy of your backing store very quickly (by breaking a mirrored drive, for example).

## Initialization Mode: Creating a New Distributed Backing Store

This procedure creates an empty new backing store distributed across multiple file systems. Use this mode only if you do not have an existing backing store that you want to use.

1. Create a file that contains all the file system locations you want to stripe the TeamXpress backing store across, one to a line. In this example, the file is named `stripefile`. For example:

```
/local1/iw-store-1
/local2/iw-store-2
/local3/iw-store-3
```

All paths listed in the stripe file must be absolute pathnames (they must start with `/`). The parent directories must all exist, they must be mounted, and they must be empty, or else the `iwfsstripe` CLT will not run. Furthermore, they must all reside on different file systems. If you want to override this restriction, you can use the `-w` option when you invoke `iwfsstripe`.

Make sure that the location you want to use for the “primary” stripe location is on the first line of this file. The primary stripe location is the file system whose name will be returned by CLTs such as `iwgetstore`.

2. Use the `iwfsstripe` CLT to create a new backing store distributed across the file systems listed in `stripefile`. For information about the `iwfsstripe` CLT, see “`iwfsstripe` Syntax” on page 279.

**Note:** You must run `iwfsstripe` as root.

```
% iwfsstripe -s stripefile
Initializing new backing store...
Activating new backing store...
Done
```

3. Stop the TeamXpress server:

```
% /etc/init.d/iw.server stop
```

4. Configure the TeamXpress server to use the new backing store:

Edit `/etc/default/iwstore` to contain the location of the primary stripe (in this example, `/local1/iw-store-1`).

5. Restart the TeamXpress server:

```
% /etc/init.d/iw.server start
```

6. Use `iwgetstore` with the `-a` option to return a list of all file system locations of the TeamXpress backing store. In this example,

```
% iwgetstore -a
```

would return:

```
/local1/iw-store-1
```

```
/local2/iw-store-2
```

```
/local3/iw-store-3
```

Files will be evenly distributed across all the file systems.

## Normal Mode: Distributing an Existing Backing Store

To distribute your backing store across multiple file systems:

1. Create a file that contains all the file system locations you want to stripe the TeamXpress backing store across, one to a line. In this example, the file is named `stripefile`. For example:

```
/local1/iw-store-1
/local2/iw-store-2
/local3/iw-store-3
```

All paths listed in the stripe file must be absolute pathnames (they must start with `/`). All of these locations must exist, they must be mounted, and they must be empty, or else the `iwfsstripe` CLT will not run. Furthermore, they must all reside on different file systems. If you want to override this restriction, you can use the `-w` option when you invoke `iwfsstripe`.

Make sure that the location you want to use for the “primary” stripe location is on the first line of this file. The primary stripe location is the file system whose name will be returned by CLTs such as `iwgetstore`.

2. Stop the TeamXpress server:

```
% /etc/init.d/iw.server stop
```

3. Use the `iwfsstripe` CLT to distribute your backing store across the file systems listed in `stripefile`. For information about the `iwfsstripe` CLT, see “`iwfsstripe` Syntax” on page 279. In this example, the backing store is located in `/local/iw-store`.

**Note:** You must run `iwfsstripe` as root.

```
% iwfsstripe -s stripefile -b /local/iw-store
Initializing new backing store...
Copying non-striped files...
Copying striped files...
default archive: 43264 of 43264 point ids copied (100%)
workflow archive: 9728 of 9728 point ids copied (100%)
Activating new backing store...
Done
```

Your existing backing store will not be deleted. You can delete the old backing store at your convenience.

4. Configure the TeamXpress server to use the new backing store:

Edit `/etc/default/iwstore` to contain the location of the primary stripe (in this example, `/local1/iw-store-1`).

5. Reboot the TeamXpress server:

```
% /etc/init.d/iw.server start
```

6. Use `iwgetstore` with the `-a` option to return a list of all file system locations of the TeamXpress backing store. In this example,

```
% iwgetstore -a
```

would return:

```
/local1/iw-store-1
```

```
/local2/iw-store-2
```

```
/local3/iw-store-3
```

Files will be distributed across all the file systems, approximately evenly. You can delete the old backing store at your convenience.

## Transition Mode: Distributing an Existing Backing Store

Use transition mode only if you have a quick method of making a copy of your backing store (for example, breaking a mirrored drive). Otherwise, transition mode may not save you significant time over copying your backing store to another location.

To distribute your backing store across multiple file systems:

1. Create a file that contains all the file system locations you want to stripe the TeamXpress backing store across, one to a line. In this example, the file is named `stripefile`. For example:

```
/local1/iw-store-1
```

```
/local2/iw-store-2
```

```
/local3/iw-store-3
```

All paths listed in the stripe file must be absolute pathnames (they must start with `/`). All of these locations must exist, they must be mounted, and they must be empty, or else the `iwfsstripe` CLT will not run. Furthermore, they must all reside on different file systems. If you want to override this restriction, you can use the `-w` option when you invoke `iwfsstripe`.

Make sure that the location you want to use for the “primary” stripe location is on the first line of this file. The primary stripe location is the file system whose name will be returned by CLTs such as `iwgetstore`.

2. Stop the TeamXpress server:

```
% /etc/init.d/iw.server stop
```

3. Copy the backing store to another location.

4. Start transition mode logging on the original backing store. In this example, the original backing store is located in `/local/iw-store`.

**Note:** You must run `iwfsstripe` as root.

```
% iwfsstripe -s stripefile -b /local/iw-store -t start
```

```
Initializing new backing store...
```

```
Initializing transition mode logging of /local/iw-store...
```

For information about the `iwfsstripe` CLT, see “`iwfsstripe Syntax`” on page 279.

**Note:** Transition mode logging keeps track of the files that are modified in the original backing store while transition mode logging is turned on. After the copy of the backing store is converted, `iwfsstripe` will use the log to convert the differences between the original backing store and the copy. Transition mode logging relies on the presence of the `iwtransitionlog` file in each of the backing store archives. Do not remove this file, or else transition mode conversion will not be able to complete.

5. Restart the TeamXpress server:

```
% /etc/init.d/iw.server start
```

6. Use the `iwfsstripe` CLT in transition mode to distribute the copy of the backing store across the file systems listed in `stripefile`:

```
% iwfsstripe -s stripefile -b /local/iw-store-copy -t convert
```

```
Copying non-striped files...
```

```
Copying striped files...
```

```
default archive: 43264 of 43264 point ids copied (100%)
```

```
workflow archive: 9728 of 9728 point ids copied (100%)
```

```
Done
```

7. When `iwfsstripe` finishes converting the copy, stop the TeamXpress server again.

```
% /etc/init.d/iw.server stop
```

8. Finish the transition mode conversion:

```
% iwfsstripe -s stripefile -b /local/iw-store -t finish
```

```
Copying non-striped files...
```

```
Copying logged files...
```

```
Activating new backing store...
```

```
Done
```

9. Configure the TeamXpress server to use the new backing store:

Edit `/etc/defaultiwstore` to contain the location of the primary stripe (in this example, `/local1/iw-store-1`).

10. Restart the TeamXpress server:

```
% /etc/init.d/iw.server start
```

11. Use `iwgetstore` with the `-a` option to return a list of all file system locations of the TeamXpress backing store. In this example,

```
% iwgetstore -a
```

would return:

```
/local1/iw-store-1
```

```
/local2/iw-store-2
```

```
/local3/iw-store-3
```

Files will be distributed approximately evenly across all the file systems.

You can delete the old backing store at your convenience.

## Returning Your Backing Store to a Single File System

If you have distributed your backing store across multiple file systems, as described above, you can return it to a single file system at any time.

To return your backing store to a single file system, edit `stripefile` to contain only the location of the single file system. Proceed with the directions for converting an existing backing store, as shown above.

## iwfsstripe Syntax

### Usage:

```
iwfsstripe [-h] [-v] [-V] [-w] -s stripefile [-b storepath]  
[-t {start|convert|finish}]
```

<code>-h</code>	Displays usage message.
<code>-v</code>	Displays version.
<code>-V</code>	Verbose mode.
<code>-w</code>	Overrides unique file system check.
<code>-s <i>stripefile</i></code>	Initializes a multiple file system backing store using the pathnames listed in <i>stripefile</i> .
<code>-b <i>storepath</i></code>	Copies the contents of the backing store located at <i>storepath</i> into the new backing store.

<code>-t start</code>	Phase 1 of transition mode; initiates logging of TeamXpress activity on the backing store specified by the <code>-b</code> option.
<code>-t convert</code>	Phase 2 of transition mode; copies the contents of the backing store specified by the <code>-b</code> option into the new backing store.
<code>-t finish</code>	Phase 3 of transition mode; ends logging of TeamXpress activity on the backing store specified by the <code>-b</code> option, and copies the logged changes into the new backing store.

where *stripefile* is the file containing the list of file systems, and *storepath* is the root path to the source backing store (the root path is the pathname returned by `iwgetstore`).

**Examples:**

The following command initializes a backing store on multiple file systems:

```
% iwfsstripe -s stripefile
```

The following command initializes a backing store on multiple file systems and copies the contents from an existing backing store into the new backing store:

```
% iwfsstripe -s stripefile -b /local/iw-store
```



# Internationalization

---

This appendix contains the following information:

- An overview of TeamXpress multibyte character support.
- General recommendations and information regarding specifying the encoding of Web assets and the browser behavior when interpreting encoding.
- Information regarding multibyte support for TeamXpress Templating, DataDeploy, and OpenDeploy.

## Overview

TeamXpress 1.0 supports multiple languages. End users can enter data into TeamXpress in the following four languages:

- US English
- Japanese
- Traditional Chinese
- Simplified Chinese

TeamXpress GUI elements, such as buttons and drop down menus, retain English names but may look slightly different because all HTML pages of our browser-based GUI are now UTF-8 encoded, even for US English installations. Your client browsers may therefore choose different fonts to render UTF-8 HTML pages.

Note that TeamXpress users can interact with the GUI using any one of the supported four languages, but the TeamXpress server *must* use an US English operating system. All TeamXpress configuration files (including workflow templates) must be ASCII. TeamXpress does not currently support non-ASCII characters in branch, area, directory, or file names.

## About UTF-8

UTF-8 is the encoding for Unicode. Unicode is a system for exchanging, processing, and displaying diverse written languages. Unicode supports the principal written languages of the world as well as many classical languages.

## Recommendations

It is strongly recommended that you specify the encoding on all pages. Do this by using the `charset` parameter within the `META` tag:

```
<META HTTP-EQUIV="Content-type" CONTENT="text/html; charset=UTF-8">
```

Specify the encoding of your Web content so that browsers may display this content consistently without error. When passed content that is not specified, browsers rely on default settings to interpret content encoding; if the default does not match the passed content's encoding, the browser may display nonsense.

### Using Notepad and Wordpad

Notepad on Windows 2000 defaults to ANSI when a file is saved; Wordpad defaults to Rich Text Format (RTF). To declare UTF-8 encoding when using Notepad, select **File > Save As**, then select **UTF-8** in the **Encoding** drop down menu. When using Wordpad, select **File > Save As**, then select **Unicode Text Document** in the **Save as type** drop down menu.

Wordpad on both the Traditional and Simplified Chinese versions of Windows NT 4.0 can neither save nor render text as UTF-8.

In Notepad, you can begin the first line of text with a š character. In Wordpad, do not begin the first line of text with š.

### Behavior of Netscape Navigator

Once Netscape finds a UTF-8 page, it uses UTF-8 as its default encoding for pages that do not specify their encoding. This may cause the browser to display pages incorrectly if the user browses pages that do not specify their encoding, or creates pages without specifying the encoding.

### **Scenario 1**

1. A Japanese user goes to a Japanese site which does not specify its encoding. Netscape defaults to 'Japanese (Auto-Detect).'
2. The Japanese user logs into TeamXpress (UTF-8 pages). Netscape switches to UTF-8.
3. The Japanese user opens a new window and returns to the Japanese site which does not specify its encoding. Now Netscape defaults to UTF-8.

This would not happen if the site specified the encoding of its Web pages.

### **Scenario 2**

1. A Japanese user logs into TeamXpress (UTF-8 pages). Netscape switches to UTF-8.
2. The Japanese user's content in TeamXpress does not include the 'Content-type' META tag.
3. Upon entering SmartContext QA, Netscape tries to render the content as UTF-8, which is probably wrong. The solution to this problem is to always specify the encoding for all HTML content.

### **Summary**

All browsers rely on default settings to "guess" the encoding of pages whose encoding is not explicitly declared. The default setting is established differently in Internet Explorer than in Netscape Navigator. If the browser's default setting is different than that of the actual encoding of the page passed to the browser, the browser may render the page incorrectly. Therefore, your Web pages should always declare their encoding. Not only will this prevent Netscape from guessing incorrectly when you use TeamXpress, but it will also ensure that your Web site viewers' browsers will not have to guess which encoding they should use.

## Multiple Languages, One Website

TeamXpress allows contributors to use any or all of the four supported languages in its GUI. However, users who want to see multi-language content must have browsers capable of doing this. For example:

1. In TeamXpress, a Japanese user enters some submit comments in Japanese.
2. If an American user wants to view the version history, then the American user's browser must be capable of correctly displaying Japanese characters.

# Index

---

## Symbols

.uid files 20, 52

## A

absolute paths 113  
access privileges 49  
    TeamXpress 49  
accessing TeamXpress 34  
adding a new user 49  
adding metadata capture to  
    TeamXpress GUI 225  
adding metadata search to  
    TeamXpress GUI 232  
Administrators 10  
    about 50  
aliases  
    web server 23  
applications  
    editing 15  
area labels  
    configuring 65  
assigning files 10  
attributes  
    of windows 76  
authentication  
    external file 85  
    LDAP 85  
    password 49  
    setting type 85  
    user 85

## Authors 10

    about 50  
autocompression 87  
autoconfiguration, of  
    LaunchPad 70  
Autoprivate 93  
autoprivate.cfg 63, 93, 258  
available\_templates.ipl 166

## B

backing store 19, 244  
    converting 265  
        normal mode 267  
        requirements 266  
        transition mode 268  
        troubleshooting 269  
    location 16  
    repairing 239  
    returning to a single file  
        system 279  
backing store distribution  
    CLT. *see* iwfsstripe  
backups  
    of workareas 252  
    strategies 253  
branch 7, 46  
    permissions 57  
    remappings  
        configuring 115  
    structure 246  
branch and workarea security 90

## browser windows

    configuring 73

## C

cache size 99  
CGI scripts  
    adding to the GUI 74  
CGI\_info directive 150  
cgitask element 192  
changing valid search paths 233  
checking  
    disk space usage 245  
    for multiple servers 236  
    request handling 236  
    server status 235  
chgrp command 54  
clients  
    Microsoft network 34  
    NFS 37  
    TeamXpress 34  
comments  
    publish 47  
    submit  
        individual file 44  
        keywords 44  
        submit operation 44  
configuration files 257, 259  
    locations 92  
configuring  
    area labels 65  
    autocompression 87



- Autoprivate 93
- backing store freezes 101
- branch and workarea
  - security 90
- branch remappings 115
- cache size 99
- custom menu items 74
- default permissions 91
- directory operations 81
- disabling Editor publish capability 69
- edition views 68
- external remappings 125
- Failsafe 130
- file locations 92
- file system active area
  - cache 100
- file system threadcount 100
- group remapping 91
- history views 68
- job attributes 83
- jobs in the GUI 83
- launching files through
  - iwproxy 98
- lock behavior 89
- main branch locking model 88
- main branch ownership 89
- main configuration file 63
- menu items 79
- metadata capture 203, 206
- new browser windows 73
- preview windows 73
- RPC threadcount 99
- rule sets for metadata
  - capture 211
- Submit and Update logs 89
- submit button 78
- submit logs 89

- templates 96
- throughput monitors 101
- TSTackingCache 133
- update logs 89
- web server uid 88
- conserving disk space 247
- conventions
  - notation 5
- converting
  - backing store 265
  - normal mode 267
  - requirements 266
  - transition mode 268
  - troubleshooting 269
- copying files 42
- creating
  - branches 40
  - workareas 41
- custom menu items 74
  - adding 74

## D

- data store 244
- datacapture.cfg
  - annotated example 214
  - database element 218
  - DATE datatype 219
  - instance 219
  - metadata identifier 218
  - rule identifier 218
  - UTF-8 encoding 218
  - validation-regex 220
- configuring 211
- DTD 212
- default file locations 16
- default permissions 91
- deleting branches 249
- directory operations

- disabling 81
- directory permissions 57
- disabling
  - directory operations 81
  - LaunchPad autoinstallation 70
  - menu items 79
  - SmartContext Editing 69
- disk space 244
  - checking usage 245
  - conserving 247
  - data store 244
  - file system mount 246
  - moving the backing store 249
  - removing old versions 249
  - requirements 14
  - usage 245
- document root 115

## E

- editing applications 15
- edition views
  - configuring 68
- editions 8
  - initial 40
  - new 46
  - publishing 46
  - viewing 68
- Editors 10
  - about 50
  - disabling publish capability 69
- ELEM directive 154
- enabling 98
  - SmartContext Editing 69
- encoding
  - Unicode 282
  - UTF-8 282
- endtask element 195
- errata 6

external remappings  
  configuring 125  
externaltask element 190

## F

Failsafe 130  
  about 130  
file locations  
  changing 92  
file permissions 57  
file system active area cache 100  
file system interface  
  network connection 16  
  using 16  
file system mount 246  
file system threadcount 100  
  configuring 100  
files  
  assigning 10  
  permissions 57  
  submitting to the staging  
    area 43  
  virtual 246  
finding installation directory 237  
FTP 16, 34, 37

## G

Global Report Center 14  
  installing 22  
groups  
  creating 54  
  files 53  
  membership 54  
  remapping 91  
grouptask element 189

## H

hardware requirements 13  
history views  
  configuring 68  
HTML pages  
  adding to the GUI 78  
httpd user name 23  
HTTPS requests  
  redirecting 30

## I

in-context QA 98  
initial edition 40  
inode requirements 14  
INSERT directive 158  
installation directory  
  finding 237  
installation files 18  
installation log file 22  
installing  
  TeamXpress Templating 23  
installing TeamXpress 17  
internationalization  
  browser behavior 282  
  recommendations 282  
  Unicode 282  
  using Notepad and  
    Wordpad 282  
  UTF-8 282  
iw.cfg 63, 115  
iwckrole 55  
iwfsconvert 265  
  normal mode 267  
  requirements 266  
  syntax 270  
  transition mode 268  
  troubleshooting 269  
iwfsstripe

  initialization mode 273  
  normal mode 275  
  overview 272  
  transition mode 276

iwgetelogs 238  
iwininstall 18  
iwinvokejob 140  
iwjobc 140  
iwproxy 113  
  debug option 129  
  launching files 98  
iwstat 239  
iwtemplates.cfg 63, 96

## J

JavaScript 33  
job specification 138  
  defined 138  
  file 139  
job specification file  
  DTD 180  
jobs 138  
  attributes 83  
  defined 138  
  filters 83  
  listing in the GUI 83  
  *see also* workflow  
  settings 83

## L

languages  
  browser behavior when  
    interpreting web content  
    encoding 282  
LaunchPad 34  
  autoconfiguration 70  
  disabling autoinstallation 70  
LDAP 85



- loading content 39
- locations
  - of TeamXpress files 92
- locking model
  - setting 40
- locks
  - configuring behavior 89
- login names 52
- logs 238
  - submit 89
  - update 89
- low disk space
  - detecting 101
- low inode count
  - detecting 101
- M**
- main branch
  - locking model 88
- main branch ownership 89
- making fields non-searchable 233
- managing server resources. *see* server resources
- Masters
  - about 11, 50
- memory requirements
  - and cache size 13
- menu items
  - disabling 79
- metadata
  - capture form 211
- metadata capture 203
  - about 204
  - adding to TeamXpress GUI 225
  - and DAS
    - synchronizing 232
  - components 204
  - configuring 206
  - DTD 207, 211
  - extended attributes 226
  - initiating 227
  - results of 226
  - rule sets 211
  - schematic 205
- metadata consolidation
  - about 265
- metadata search
  - adding to TeamXpress GUI 232
  - changing valid search paths 233
  - components 229
  - configuring 231
  - making fields non-searchable 233
  - overview 228
  - prerequisites 229
- metadata-rules.cfg
  - configuring 207
  - examples 208
  - rule identifier 208
  - UTF-8 encoding 208
  - vpath identifier 208
- MIME types 27
  - exe 27
- mounting
  - TeamXpress server 237
- moving 244
- multibyte characters
  - browser behavior when interpreting web content encoding 282
- multiple servers
  - checking 236
- N**
- NetBEUI 36
- Netscape 25
  - CGI programs 28
  - configuring aliases 25
  - configuring MIME types 27
  - mime.type file 28
  - NSAPI redirector module 29
- Netscape Server
  - Administrator 25
- network drive 34
- network file system 16
- new users
  - adding 49
- NFS 16, 34, 37
  - group limitations 91
- notation conventions 5
- Notepad
  - saving documents as UTF-8 282
- NSAPI redirector module 29
- O**
- ownership
  - of workareas, changing 54
- P**
- passwords 51
  - authentication 49
  - enabling plain-text 37
  - Windows encrypted 37
- paths
  - absolute 113
  - relative 113
- pcnfsd 16
  - installing 22
- permissions 56
  - branch 57
  - directory 57
  - file 49, 57
  - workarea 49, 57



- preview windows
  - configuring 73
- profiles, user
  - about 69
- program files
  - location 16
- proxy server 111
  - configuring basic operation 113
  - configuring to use different webservers 124
  - debugging 129
  - external remappings 125
  - fully-qualified paths 117
    - configuring Internet Explorer 121
    - configuring Netscape 119
  - host header remappings 126
  - redirecting TeamXpress views 121
  - relative and absolute paths 113
  - rules of precedence 111
- publishing 10, 46
  - first edition 46
- R**
- redirecting HTTPS requests 30
- redirector module 98
- regular expressions 96, 103
  - about 5
- relative paths 113
- removing users 53
- repairing backing store 239
- request handling
  - checking 236
- requirements
  - disk space 14
  - hardware 13
  - inode 14
  - memory 13
  - software 15
  - system 13
  - web server 15
- returning backing store to a single file system 279
- reviewing TeamXpress logs
  - overview 238
- roles
  - TeamXpress 49
- roles files 52
  - master users 23
- RPC threadcount
  - configuring 99
- rule sets
  - configuring 211
- S**
- Samba 16, 34
  - configuring 31
  - installing 22
- searching metadata
  - components 229
  - configuring 231
  - overview 228
  - prerequisites 229
- secure socket layer 24, 29
- server load 239
  - monitoring 239
- server mount
  - verifying 237
- server operation
  - verifying 235
- server resources
  - managing 244
    - disk space 244
- server status 235
- server-side includes 24, 29
  - configuring Netscape web servers 24, 29
- SmartContext Editing
  - disabling 69
  - enabling 69
- SmartContext QA 24, 29
- software requirements 15
- srm.conf 23
- staging area 8, 40
- starting TeamXpress server 239
- stopping TeamXpress server 239
- Submit and Update logs 89
- submit button
  - configuring 78
- submit filtering
  - when populating a workarea 43
- submit.cfg 63, 102
- submittask element 193
- submitting files 43
- synchronizing metadata capture and DAS 232
- T**
- TAG directive 155
- TAG\_info directive 146, 151
- tasks
  - defined 139
- TeamXpress
  - accessing
    - through the file system 34
    - through the GUI 33
  - troubleshooting 36
  - adding metadata capture to GUI 225
  - adding metadata search to GUI 232



- clients 32, 34
- configuration files 257, 259
- installation directory 17
- mounting 34
- populating 42
- proxy server 111
- roles 49
- user roles 50
- TeamXpress server
  - answering requests 236
  - mounting 237
  - process 236
  - restarting 236
  - starting 239
  - stopping 239
- TeamXpress Templating
  - installing 23
- template file 139
  - components 141
- template\_script element 148
- templates
  - configuring 96
- throughput monitors 101
- troubleshooting
  - backing store conversion 269
  - overview 239
  - repairing backing store 239
- TSSBackingCache 133
- U**
- Unicode 282
- uninstalling TeamXpress 47
- updatetask element 194
- user authentication 85
- user profiles
  - about 69
- user roles 50, 56
  - Administrator 50
  - assigning 20
  - Author 50
  - checking 55
  - Editor 50
  - Master 50
  - roles files 20
- users
  - adding 51
  - removing 51, 53
- usertask element 188
- UTF-8 282
  - recommendations 282
- V**
- VALUE directive 159
- verifying server mount 237
- virtual files 246
- W**
- web browsers
  - behavior when interpreting
    - web content encoding 282
- web server
  - uid 88
- web servers
  - aliases 16, 21, 23
  - Apache 23
  - NCSA 23
  - Netscape 25
    - CGI programs 28
    - configuring aliases 25
    - configuring MIME types 27
  - plugin 98
  - requirements 15
  - restarting 21
- window attributes 76
- Windows networking 16
- WINS server 35
- Wordpad
  - saving documents as UTF-8 282
- workareas 7
  - changing group ownership 54
  - creating 41
  - permissions 57
  - populating 42
  - private 42
  - shared 42
  - submitting to the staging area 43
- workflow
  - CGI\_info directive 150
  - cgitask element 192
  - ELEM directive 154
  - endtask element 195
  - external task element 190
  - grouptask element 189
  - INSERT directive 158
  - instantiating a job 140
  - job specification 138
  - job specification DTD 180
  - job specification file 139
  - jobs 138
  - model 137
  - POST/GET data 146
  - sample template file 146, 161
  - schematic of 142
  - submittask element 193
  - TAG directive 155
  - TAG\_info directive 151
  - tasks 139
    - specifying files in 227
  - template file 139
    - components 141
    - sample 146, 161

- structure and syntax 144
- template\_script element 148
- troubleshooting 23
- updatetask element 194
- usertask element 188
- VALUE directive 159
- variables in strings 161
- variables passed via POST/
  - GET 155
- workflow element 181