



I N T E R W O V E N

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

**Command-Line Tools Reference Guide
for the Windows NT® and Windows 2000® Operating Systems**

© 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, OpenDeploy, TeamSite, 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 # 10-00-40-45-04-110-700

Table of Contents

About This Book 7

- Content and Audience 7
- How This Manual is Organized 7
- Notation Conventions 8

Chapter 1: Overview of CLTs and Command Triggers 11

- CLTs 11
 - Administration Tools 11
 - Development Tools 12
- Command Triggers 13
- Before Using CLTs 13
 - Version Paths 13
 - Relative vpaths 15
 - Directory Paths 16
 - Setting Path Names in Environment Variables 16
- Object IDs 17
- Class IDs 17
- Comments 18
- CLT Location 18

Chapter 2: Administration Tools 19

- iwabort 21
- iwbackup 22
- iwconfig 23
- iwfailsafe 25
- iwfreeze 27
- iwfsck 29
- iwfsconvert 32
- iwfsfix 34
- iwfsshrink 36
- iwfsstripe 38
- iwgetelog 40
- iwgethome 41
- iwgetlocation 42
- iwgetlog 44



iwgetmount 45
iwgetstore 46
iwgettrace 47
iwproxy 48
iwrecentusers 49
iwreset 50
iwrestore 51
iws 52
iwstat 54
iwtestcfg 56
iwuidname 57
iwversion 58

Chapter 3: Development Tools 59

iwaddtaskfile 63
iwattrib 64
iwcallback 73
iwcat 74
iwckrole 75
iwcmp 76
iwcompress 78
iwdecode 80
iwdelcp 81
iwdeploy 83
iwdiffapply 85
iwdiffdir 87
iwencode 89
iwevents 90
iwextattr 94
iwgetwfobj 96
iwinvokejob 103
iwjobc 104
iwjobvariable 105
iwlasted 106
iwlist 107
iwlistlocks 110
iwlistmod 111
iwlock 113

iwlockinfo 114
iwmenu 116
iwmkbr 121
iwmkwa 123
iwnexted 124
iwprv 125
iwpublish 127
iwqueryjobs 129
iwquerytasks 130
iwrcsdiff 132
iwrename 134
iwretryjobop 135
iwrevert 136
iwrlog 137
iwrnbr 139
iwrmed 140
iwrnjob 141
iwrntaskfile 142
iwrnwa 143
iwsubmit 144
iwtaketask 146
iwtaskselect 147
iwundochoice 148
iwunlock 149
iwupdate 150
iwyvpath 151

Chapter 4: Command Triggers 153

Starting Command Triggers 154
Environment Variables 155
iwat 158
iwat-env 159
iwatasgn 161
iwatcreate 162
iwatlock 163
iwatmkbr 164
iwatmkwa 165
iwatpub 166



iwatrmbr 167
iwatrmed 168
iwatrmwa 169
iwatserver 170
iwatsub 171
iwatunlock 172
iwatupdate 173
iwlsat 174
iwrmat 175

Appendix A: Master List 177

Appendix B: Sample Command Trigger Scripts 183

Email Notification Script 183
Replication Script 189

Appendix C: Error Codes 195

Index 207

About This Book

Content and Audience

TeamXpress Command-Line Tools describes each TeamXpress command-line tool (CLT) and command trigger, including syntax and usage examples. It is intended primarily for TeamXpress Administrators and Master users, and for web server administrators and system administrators. Users of this manual should be familiar with either IIS or Netscape web servers, and with basic Windows NT operations such as adding users and modifying ACLs.

How This Manual is Organized

This manual is organized as follows:

- Chapter 1, “Overview of CLTs and Command Triggers” - Describes the types of operations you can perform with CLTs and command triggers, how CLTs and command triggers are grouped into functional categories, and the general syntax you must use when entering information (e.g., vpaths, object IDs, and so on) common to many CLTs.
- Chapter 2, “Administration Tools” - Contains man pages for the System Information, System Services, and Backing Store CLTs. All man pages in the chapter are presented as one group, arranged alphabetically.
- Chapter 3, “Development Tools” - Contains man pages for CLTs in the following categories: General Development, Deployment, Branch Operation, Edition Operation, Workarea Operation, Version Management, Workflow/Job, and Templating. All man pages in the chapter are presented as one group, arranged alphabetically.
- Chapter 4, “Command Triggers” - Contains man pages for command triggers from the following groups: General Events, File and Directory Events, Branch Events, Edition Events, and Workarea Events. All man pages in the chapter are presented as one group, arranged alphabetically.
- Appendix A, “Master List” - Contains a single alphabetized list of all CLTs and command triggers described in this manual.

- Appendix B, “Sample Command Trigger Scripts” - Contains examples of scripts that can be triggered by command triggers.
- Appendix C, “Error Codes” - Lists the error codes that can be returned by CLTs.

Notation Conventions

This manual uses the following notation conventions:

Convention	Definition and Usage
Bold	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 Edit File in the Button Bar.
<i>Italics</i>	Book titles appear in italics. Terms are italicized the first time they are introduced. Valuable information may be italicized for emphasis.
Monospace	Commands, command-line output, and file names are in monospace type. For example: The <code>iwextattr</code> command-line tool allows you to set and look up extended attributes on a file.
Monospaced italic	Monospaced italics are used for command-line variables. The most common example of this is <i>iw-home</i> , which refers to the directory where TeamXpress is installed. For example: <code><i>iw-home</i>\etc\iw.cfg</code> is the path to the main TeamXpress configuration file, <code>iw.cfg</code> , which is located in the <code>etc</code> directory under the TeamXpress installation directory. <code><i>iwckrole</i> role user</code> means that you must insert the values of <i>role</i> and <i>user</i> yourself.

Convention	Definition and Usage
Monospaced bold	<p>Monospaced bold represents information you enter in response to system prompts. The character that appears before a line of user input represents the command prompt, and should not be typed. The > character that appears before a line of user input represents the command prompt, and should not be typed. For example:</p> <pre data-bbox="462 510 1291 569" style="text-align: center;">>iwextattr -s project=proj1 //IWSERVER/default/main/dev/WORKAREA/andre/products/index.html</pre>
Monospaced bold italics	Used to indicate a variable in response to a system prompt.
[]	Square brackets surrounding a command-line argument mean that the argument is optional.
	Vertical bars separating command-line arguments mean that only one of the arguments can be used.

Overview of CLTs and Command Triggers

CLTs let you use the command line instead of the browser or file system interface to perform most TeamXpress administration and development tasks. For example, TeamXpress provides CLTs for creating and deleting branches and workareas, publishing, deploying, assigning files, invoking the proxy server, interacting with jobs, and dozens of other tasks. CLTs are designed for users and administrators who are comfortable using a command-line interface, and who cannot or prefer not to use the browser or file system interface.

In addition to CLTs, TeamXpress also supports a set of command triggers that let you configure TeamXpress to execute custom scripts whenever certain events occur. For example, you can use command triggers to execute an email notification script whenever a file is assigned, or whenever a new branch is created, or when any of several other supported events occur. The scripts that are triggered can be simple or complex and typically are created by an administrator for a specific site or installation.

You should use CLTs and command triggers only if you have a good understanding of TeamXpress structure, roles, and concepts. When learning CLTs, it is often helpful to monitor your activities using the GUI. If you choose to monitor from the GUI, be sure to refresh your view often. In some cases, it might be helpful to run multiple TeamXpress sessions using the GUI.

The following sections briefly describe the default set of TeamXpress CLTs and command triggers.

CLTs

CLTs fall into two main categories: administration tools and development tools.

Administration Tools

Administration tools are divided into the following categories:



- **System Information:** Return system-wide information or information about a file's location. They do not provide information about the data in a file (the development tool CLTs provide that type of information).
- **System Services:** Let you manipulate files or objects that affect the entire TeamXpress system. Typical tasks include, but are not limited to, editing configuration files, performing system backups, and starting the proxy server.
- **Backing Store:** Let you perform various tasks on the backing store, including checking fore and fixing problems, converting to and from a multiple file system backing store, and merging metadata files.

See Chapter 2, “Administration Tools,” for details about the CLTs in each category.

Development Tools

Development tools are divided into the following categories:

- **General Tools:** Let you perform operations on, or retrieve information about, the data in a specific file or TeamXpress object.
- **Deployment:** Let you deploy a TeamXpress edition to a production server.
- **Branch Operations:** Let you create and manipulate TeamXpress branches.
- **Edition Operations:** Let you create and manipulate TeamXpress editions.
- **Workarea Operations:** Let you create and manipulate TeamXpress workareas.
- **Version Management:** Return information about file revisions and versions, and let you manipulate specific versions of files and directories.
- **Workflow/Job:** Let you control workflow and job elements such as file locks, assignments to authors, and file approval/rejection.
- **Templating:** Let you update, insert data into, and otherwise manipulate template-based files. Available only if TeamXpress Templating is installed.

See Chapter 3, “Development Tools,” for details about the CLTs in each category.

Command Triggers

All command triggers are described in Chapter 4, “Command Triggers.” Two sample command trigger perl scripts, one that sends email notification and one that replicates files, are shown in Appendix B, “Sample Command Trigger Scripts.”

Before Using CLTs

Most CLTs require that you specify some type of information when you invoke the command. For example, you might have to specify an object ID or the location of a TeamXpress object, or provide some other type of information. The following sections describe how to provide this information.

Additionally, you can enter the name of the CLT followed by `-h` to obtain “help” on that CLT. The help provides a brief description of the CLT and lists the flags and options for that CLT. You can enter the CLT name followed by `-v` to display version information. If `-h` or `-v` are specified, any other flags or options entered at the same time are ignored.

Version Paths

Most CLTs require that you use a version path (vpath) to specify the location of a branch, workarea, staging area, edition, file, or directory in TeamXpress. Vpaths are only used for items within TeamXpress; all other files (e.g., configuration files) must be specified with a directory path (see page 16). Vpaths have the following form:

To specify a server:

```
//servername
```

Example:

The default vpath for the TeamXpress server is:

```
//IWSERVER
```

To specify an archive:

```
//servername/archivename
```

Example:

The default vpath for the TeamXpress archive is:

```
//IWSERVER/default
```

To specify a branch or sub-branch:

```
//servername/archivename{ [/branchname] + }
```

```
//servername/archivename{ [/branchname] + }/subbranchname
```

Examples:

The default vpath for the main branch is:

```
//IWSERVER/default/main
```

The vpath for a sub-branch named dev off of the main branch would be:

```
//IWSERVER/default/main/dev
```

To specify an edition:

```
//servername/archivename{ [/branchname] + }/EDITION/editionname
```

Examples:

The default vpath for the initial edition on the main branch is:

```
//IWSERVER/default/main/EDITION/INITIAL
```

The name of an edition on the subbranch dev would be:

```
//IWSERVER/default/main/dev/EDITION/ed_0001
```

To specify a workarea:

```
//servername/archivename{[/branchname]+}/WORKAREA/workareaname
```

Examples:

The vpath for a workarea named `eng` on the main branch would be:

```
//IWSERVER/default/main/WORKAREA/eng
```

The vpath for a workarea named `qa` on the sub-branch `dev` would be:

```
//IWSERVER/default/main/dev/WORKAREA/qa
```

To specify an individual file within a workarea:

```
//servername/archivename{[/branchname]+}/WORKAREA/workareaname/directory-  
pathname/filename
```

Example:

The vpath for a file named `logo.gif` in the directory `/htdocs/gifs` in workarea `eng` on sub-branch `dev` would be:

```
//IWSERVER/default/main/dev/WORKAREA/eng/htdocs/gifs/logo.gif
```

Relative vpaths

Most TeamXpress CLTs only require you to specify relative vpaths. For example, instead of specifying:

```
//IWSERVER/default/main
```

you would only need to specify:

```
main
```

Instead of specifying:

```
//IWSERVER/default/main/dev/WORKAREA/eng/htdocs/gifs/logo.gif
```

you would only need to specify:

```
main/dev/WORKAREA/eng/htdocs/gifs/logo.gif
```

If you were already in the `htdocs` directory of `workarea eng` on the `main` branch, you would only need to specify:

```
gifs/logo.gif
```

Directory Paths

Some CLTs require you to specify directory paths. These are the full file system mount directory paths of a file in TeamXpress, starting from the server's root directory. When you use a CLT, be sure to check whether it requires a directory path or a `vpath`.

For example, a file whose `vpath` is:

```
//IWSERVER/default/main/dev/WORKAREA/eng/htdocs/gifs/logo.gif
```

would have the directory path of:

```
Y:\default\main\dev\WORKAREA\eng\htdocs\gifs\logo.gif
```

Note that `vpaths` can use forward or back slashes, whereas directory paths use back slashes.

Object IDs

Some CLTs require that you specify object IDs (`objids`). `Objids` are identifiers for each object (file, directory, `workarea`, staging area, edition, or tag) in the TeamXpress system. To find the `objid` for an item, use the `iwattrib` command (see page 64). CLTs that require an `objid` also require further information about the `objid`, such as what type of object it describes or what area it resides in.

Object IDs are 24-digit hexadecimal numbers, such as:

```
0x00002258000000000000006bb
```


Class IDs

Each type of object has its own class ID. Class IDs are 8-digit hexadecimal numbers, whose values for each class are:

server	0x00000051
archive	0x00010010
branch	0x00010250
workarea	0x00010100
staging area	0x00010255
submit event	0x00002258
update event	0x0000225a
edition	0x00010020
file or directory	0x08001000

Comments

CLTs that use comments require them to be specified in one of two ways. For example, a CLT could require a comment to be specified using a tag:

```
iwdone [-h|-v] [-u] [-c comment] vpath
```

Here, you can specify a comment using the `-c` tag. If you do not use the `-c` tag, the change comment will be read from `stdin`. Type a comment, or no comment, then type `Ctrl-Z` to exit. If you do not specify a comment, you must still type `Ctrl-Z`.

The second way of specifying a comment is as a required field:

```
iwassign [-f|-s] vpath comment assignee [assigner] [-f|-s]
```

requires you to type a comment in the comment field. If you do not want to attach a comment, type `' '` in the comment field. If the comment includes spaces, enclose it in a set of quotation marks, e.g., `'please edit this file.'`

CLT Location

CLTs reside in the `bin` subdirectory of the directory returned by `iwgethome`—although not everything contained in this directory is a CLT.

Chapter 2

Administration Tools

This chapter contains man pages for the System Information and System Services administration CLTs. All man pages in this chapter are presented as one group, arranged alphabetically.

System Information CLTs return system-wide information or information about a file's location. They do not provide information about the data in a file (the Development Tool CLTs provide that type of information). CLTs in this category are:

CLT	Description	See...
iwgetelog.exe	Returns location of TeamXpress events log.	page 40
iwgethome.exe	Returns location of TeamXpress program files.	page 41
iwgetlocation.exe	Returns the locations of various TeamXpress log and configuration files.	page 42
iwgetmount.exe	Returns location of TeamXpress mount point	page 45
iwgetstore.exe	Returns location of TeamXpress backing store.	page 46
iwgettrace.exe	Returns location of TeamXpress trace logs.	page 47
iwrecentusers.exe	Displays a list of everyone who has used TeamXpress since the last time the TeamXpress server was started, and a timestamp of each user's most recent TeamXpress operation.	page 49
iwstat.exe	Returns current system activity.	page 54
iwversion.exe	Returns current TeamXpress release information.	page 58

System Services CLTs let you manipulate files or objects that affect the entire TeamXpress system. Typical tasks include, but are not limited to, editing configuration files, performing system backups, and starting the proxy server. CLTs in this category are:

CLT	Description	See...
iwabort.exe	Stops a current operation.	page 23
iwchgrp.exe	Changes the group of a TeamXpress workarea or branch.	page 22

iwconfig.exe	Reads or writes to TeamXpress's main configuration file.	page 23
iwfreeze.exe	Freezes and unfreezes all system writes.	page 27
iwreset.exe	Rereads TeamXpress configuration files.	page 50
iwsi.ipl	Collects system state data.	page 52
iwteamsite.exe	Installs or uninstalls the iwtock watchdog daemon.	page 54
iwtestcfg.exe	Returns the operation that will be performed upon a file at submission time.	page 56

Backing Store CLTs let you perform various tasks on the backing store. CLTs in this category are:

CLT	Description	See...
iwfailsafe.exe	Provides for error detection in the backing cache.	page 25
iwfsck.exe	Diagnoses backing store problems.	page 29
iwfsconvert.exe	Merges multiple metadata files into a single file.	page 32
iwfsfix.exe	Fixes problems found by iwfsck.	page 34
iwfsshrink.exe	Finds and removes duplicate data in the backing store.	page 36
iwfsstripe.exe	Converts existing backing store to a multiple file systems.	page 38

iwabort.exe

Provides a method for terminating a long-running server operation. The command works on such operations as submit, update, create branch, delete branch/workarea/edition, compress edition, and freeze. Before issuing `iwabort`, use `iwstat` to obtain the identifier for the operation.

- Use with submit and update operations – There are occasionally some places in submit and update where they cannot be terminated by using `iwabort`. After issuing `iwabort`, you may end up with some but not all the files submitted or updated.
- Use with create and delete operations – The branch will not be created and may or may not be deleted.

Man Page Group:

System Services.

Usage:

```
iwabort operation_id
```

operation_id The operation_id displayed by iwstat.

Example:

Use `iwstat` to obtain the server status. Then, using the ID shown in the first column, abort the submit operation.

```
>iwstat
```

```
Status: Server running
```

ID	Thread	User	Duration	Operation
0x24caee4b	0x13	andre	560.006	SubmitFSE 0xef70087ced505b4004a9bc98
0x24caee58	0xf	-	0.000	RunFastVacuum
0x24caee57	0x15	chris	0.000	GetArchiveStatus

```
>iwabort 0x24caee4b
```

iwchgrp.exe

`iwchgrp` is `chgrp` for TeamXpress for the Windows NT Operating System. Because TeamXpress retains the concept of group ownership of branches and workareas, `iwchgrp` allows you to change the group of a branch or workarea. To change the group of a branch or workarea, you must be the owner of that branch or workarea.

Man Page Group:

System Services.

Usage:

```
iwchgrp [-h|-v] group path
```

<code>-h</code>	Displays usage message.
<code>-v</code>	Displays version.
<code>group</code>	Specifies the new domain and group name.
<code>path</code>	Specifies the directory path (including the drive) to the workarea.

Examples:

```
>iwchgrp Administrators Y:\default\main\dev\Workarea\andre
```

changes the group of workarea `andre` on the `dev` branch to the local group of administrators.

```
>iwchgrp WEBSTER\Marketing Y:\default\main\dev\Workarea\andre
```

changes the group of workarea `andre` on the `dev` branch to the `Marketing` group in the `WEBSTER` domain.

iwconfig.exe

A command line interface to TeamXpress's main configuration file, `iw.cfg`. Allows you to read the configuration file, list the names of all the sections in the configuration file, or write to the configuration file. For information on configuring `iw.cfg`, see the "Configuring TeamXpress" chapter of the *TeamXpress Administration Guide*.

Man Page Group:

System Services.

Usage:

```
iwconfig [options] section varname [value]
```

where options are:

<code>-show</code>	Displays the entire configuration file.
<code>-l</code>	Lists section names only.
<code>-w</code>	Writes value into <code>/etc/iw.cfg</code> (creates new section and variable if necessary).
<code>-d</code>	Delete a name, value pair from a specified section.
<code>-h</code>	Displays usage message.
<code>-v</code>	Displays version.
<code>section</code>	Section of <code>iw.cfg</code> to read or write to.
<code>varname</code>	Variable name within <code>section</code> to read or configure.
<code>value</code>	Value to assign to <code>varname</code> (use only with <code>-w</code>).

Example:

```
>iwconfig -l
```

returns:

```
main  
iwcgi
```

```
iwproxy  
iwproxy_remap  
global_default_map  
iwproxy_external_remap  
iwserver
```


iwfailsafe.exe

Provides for error detection in the backing cache. If you suspect a metadata inconsistency, you should check for the following:

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

If these two conditions exist at the same time, run `iwfailsafe`.

Usage

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

<code>-h</code>	Display usage message.
<code>-v</code>	Display version. Has highest precedence if multiple options are specified.
<code>-n</code>	Consistency check. Reads the log; exit code indicates log status.
<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.

iwfreeze.exe

Freezes and unfreezes all system writes. Unlike a full server stop, users can still log into the system and read data.

To find out whether the TeamXpress server is currently frozen, use the `iwstat` (page 54) command-line tool.

Man Page Group:

System Services.

Usage:

```
iwfreeze [-h|-v|+N|-j|--]
```

-h	Displays usage message.
-v	Displays version.
+N	Freeze for <i>N</i> seconds.
-j	Freeze or unfreeze only batch jobs, not the entire archive. As currently supported, batch jobs result from a Delete area operation and accomplish the following: clean up destroyed workareas, editions, and branches; compress editions.
--	Unfreeze archive immediately.

Examples:

```
>iwfreeze +30
```

returns:

```
Freeze archive default for 30 seconds...
```

and freezes the TeamXpress archive for thirty seconds.

```
>iwfreeze --
```

returns:

Archive default unfrozen.

and unfreezes the archive.

Notes:

When `iwfreeze` is issued, the system performs the following activities:

- blocks new operations
- completes all current operations
- flushes the cache
- issues a message telling users an `iwfreeze` has occurred
- returns write operations with a failure status message
- performs read-only operations

iwbsck.exe

Diagnoses backing store problems and allows repair of some of the problems found.

Man Page Group:

Backing Store.

Usage:

```
iwbsck [-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 iwbsck 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 iwbsckcgi.cgi program.
-y	Repairs damaged files while running. In this mode, damaged files are deleted while iwbsck 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 <i>path</i>	Uses <i>path</i> 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 `-f`). 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 protopath. A protopath 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
```

iwfsconvert.exe

Merges multiple metadata files into a single file in the backing store. Converting the backing store results in improved performance (i.e., reduced inode usage) and reliability.

Man Page Group:

Backing Store.

Normal Mode Usage:

```
iwfsconvert [-h] [-v] [-V] [-p|-r|-b|-f|-c] [-a archive]
```

-h	Prints this message.
-v	Prints version.
-V	Verbose mode.
-p	Does a pre-check. Not valid with -t <i>phase</i> .
-r	Restarts an interrupted conversion.
-b	Lists bad points that were not converted.
-f	Fix mode. Attempts to convert points that were not converted on the first pass.
-c	Begins conversion. Automatically restarts conversion that had already been started.
-a <i>archive</i>	Specifies the root directory of archives to convert.

Note: The disk space requirement to run `iwfsconvert` is about 25 percent of the backing store file size.

Transition Mode Usage:

```
iwfsconvert [-h] [-v] [-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:

	<code>start</code>	starts the conversion process in transition mode
	<code>convert</code>	converts the backing store specified with <code>-d destination</code>
	<code>finish</code>	finishes the transition mode conversion
	<code>restart</code>	continues an interrupted conversion
	<code>fix</code>	fixes unconverted points
	<code>showbad</code>	shows a list of unconverted points
<code>-s</code>	<i>source</i>	Specifies the path to the original backing store.
<code>-d</code>	<i>destination</i>	Specifies the path to the copy of the backing store.

Examples:

The following command starts transition mode:

```
>iwfsconvert -t start -s C:\iw-store1 -d D:\iw-store1
```

The following command converts the copy of the backing store:

```
>iwfsconvert -t convert -d D:\iw-store1
```

The following command finishes the transition mode conversion:

```
>iwfsconvert -t finish -s C:\iw-store1 -d D:\iw-store1
```

The following command displays a list of unconverted points for both the source and destination copies of the backing store:

```
>iwfsconvert -t showbad -s C:\iw-store1 -d D:\iw-store1
```

iwfsfix.exe

The `iwfsfix` command repairs certain problems. If `iwfsck` (see page 29) finds problems 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 an NT platform, the ModList repair lines are in the format:

```
FIX echo x > junkfile; del /f 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. 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.

iwfsshrink.exe

Invokes a batch job in the server to find and remove duplicate file contents in the backing store. This command is used to improve space utilization in an existing backing store. It can be used as a maintenance tool on a regular, perhaps quarterly, basis. This operation results in no user-visible changes to the TeamXpress virtual file system; for example, file histories are unchanged.

Man Page Group:

Backing Store.

Usage:

```
iwfsshrink [-h|-v] [run|pause|abort|status]
```

-h	Displays usage message.
-v	Displays version.
run	Starts the iwfsshrink process.
pause	Temporarily stops the iwfsshrink process. It can be restarted with the run option. Because iwfsshrink takes a long time to run and it increases system load, you may want to start it during off-hours. When activity increases, you can pause it until the next period of inactivity.
abort	Terminates the iwfsshrink process.
status	Shows information about the iwfsshrink process.

Examples:

Issuing iwfsshrink status when iwfsshrink has finished running, results in status 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

Issuing the iwfs shrink run command, followed by the iwfs shrink status command, results in status similar to:

```
Current run started Tue Jul 11 14:19:20 2000
Not yet completed
In phase 1
Files examined: 980
Bytes examined: 67053122
Files found to be duplicates: 0
Files converted: 0
Bytes removed: 0
```

Issuing the iwfs shrink pause command, followed by the iwfs shrink status command, results in status similar to:

```
Current run started Tue Jul 11 14:19:20 2000
Paused
In phase 1
Files examined: 5694
Bytes examined: 1191581411
Files found to be duplicates: 0
Files converted: 0
Bytes removed: 0
```

iwfsstripe.exe

Initializes a multiple file system backing store and optionally populates it with the contents of an existing backing store. `iwfsconvert` must be run prior to running this command.

Man Page Group:

Backing Store.

Usage:

```
iwfsstripe [-h] [-v] [-V] [-w] [-s stripefile]  
[-b storepath [-t {start|convert|finish}]]
```

-h	Displays usage message.
-v	Displays version.
-V	Verbose mode.
-w	Overrides unique file system check.
-s <i>stripefile</i>	Initializes a multiple file system backing store using the pathnames listed in <i>stripefile</i> .
-b <i>storepath</i>	Copies the contents of the backing store located at <i>storepath</i> into the new backing store.
-t start	Phase 1 of transition mode; initiates logging of TeamXpress activity on the backing store specified by the -b option.
-t convert	Phase 2 of transition mode; copies the contents of the backing store specified by the -b option into the new backing store.
-f finish	Phase 3 of the transition mode; ends logging of TeamXpress activity on the backing store specified by the -b 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 C:\iw-store
```

iwgetelog.exe

Returns the contents of `/etc/defaultiwelog` (the TeamXpress events log) or a non-zero error code. The TeamXpress events log includes the time an operation was performed, the username and role of the user, the type of operation, the area or file affected, and additional information according to the type of operation performed (e.g., objids and comments attached to files and operations).

Man Page Group:

System Information.

Usage:

```
iwgetelog [-h|-o|-v]
```

-h	Displays usage message.
-v	Displays version.
-o	Returns original factory setting value.

Example:

```
>iwgetelog
```

returns:

```
C:\Program Files\Interwoven\TeamXpress\local\logs\iwevents.log
```


iwgethome.exe

Returns the location of the TeamXpress program files (the contents of /etc/default/iwhome).

Man Page Group:

System Information.

Usage:

```
iwgethome [-h|-v]
```

-h	Displays usage message.
-v	Displays version.

Example:

```
>iwgethome
```

returns

```
C:\Program Files\Interwoven\TeamXpress
```

iwgetlocation.exe

Returns the location of various TeamXpress configuration or log files.

Man Page Group:

System Information.

Usage:

```
iwgetlocation [-h|-v] [-a|-l|-c configfilekey|-g logfilekey|filekey
[filekey...]]
```

-h	Displays usage message.
-v	Displays version.
-a	Returns all locations of all configuration and log files.
-l	Lists all filekeys.
-c <i>configfilekey</i>	Specifies a configuration file.
-g <i>logfilekey</i>	Specifies a log file.
<i>filekey</i>	Specifies a particular configuration or log file.

Examples:

```
>iwgetlocation -l
```

returns:

```
iwhome
iwbin
iwconfig
iwstore
iwmount
iwcgimount
iwroles
iwlogs
iwconfigs
iweventlog
iwtracelog
iwdeploylog
```

```
>iwgetlocation -a
```

returns:

```
iwhome=C:\Program Files\Interwoven\TeamXpress
iwbin=C:\Program Files\Interwoven\TeamXpress\bin
iwconfig=C:\Program Files\Interwoven\TeamXpress\etc\iw.cfg
iwstore=C:\iw-store
iwmount=Y:
iwcgimount=Y:
iwroles=C:\Program Files\Interwoven\TeamXpress\conf\roles
iwlogs=C:\Program Files\Interwoven\TeamXpress\local\logs
iwconfigs=C:\Program Files\Interwoven\TeamXpress\local\config
iweventlog=C:\Program Files\Interwoven\TeamXpress\local\logs\iwevents.log
iwtracelog=C:\Program Files\Interwoven\TeamXpress\local\logs\iwtrace.log
iwdeploylog=C:\Program
Files\Interwoven\TeamXpress\local\logs\iwdeploy.log
```

>iwgetlocation iwmount

returns:

Y:

>iwgetlocation -c iwconfig

returns:

C:\Program Files\Interwoven\TeamXpress\etc\iw.cfg

>iwgetlocation iwmount iwconfig iweventlog

returns:

Y:

C:\Program Files\Interwoven\TeamXpress\etc\iw.cfg

C:\Program Files\Interwoven\TeamXpress\local\logs\iwevents.log

iwgetmount.exe

Returns the location of TeamXpress mount point (the contents of `/etc/defaultiwmount`).

Man Page Group:

System Information.

Usage:

```
iwgetmount [-h|-o|-v]
```

Returns the contents of `/etc/defaultiwmount` or a non-zero error code.

-h	Displays usage message.
-v	Displays version.
-o	Returns original factory setting value.

Example:

```
>iwgetmount
```

returns:

Y:

iwgetstore.exe

Returns the contents of /etc/defaultiwstore (the location of the TeamXpress backing store) or a non-zero error code.

Man Page Group:

System Information.

Usage:

```
iwgetstore [-h|-o|-v|-a]
```

-h	Displays usage message.
-v	Displays version.
-o	Returns original factory setting value.
-a	Displays all multiple file system backing store locations.

Example:

```
>iwgetstore
```

returns:

```
C:\iw-store
```

iwgettrace.exe

Returns the contents of `/etc/defaultiwtrace` (the location of the TeamXpress trace logs) or a non-zero error code. The trace log contains debug information for TeamXpress.

Man Page Group:

System Information.

Usage:

```
iwgettrace [-h|-o|-v]
```

-h	Displays usage message.
-v	Displays version.
-o	Returns original factory setting value.

Example:

```
>iwgettrace
```

returns:

```
C:\Program Files\Interwoven\TeamXpress\local\logs\iwtrace.log
```

iwproxy.exe

Used to invoke and debug the proxy server. The proxy server is automatically invoked at startup time.

Man Page Group:

System Services.

Usage:

```
iwproxy [-v|-h|-d]
```

-h	Displays usage message.
-v	Displays version.
-d	Debug mode (outputs client & server headers).

Example:

```
>iwproxy -d
```

will output client and server headers.

iwrecentusers.exe

Displays a list of everyone who has used TeamXpress since the last time the TeamXpress server was started. The list also contains a timestamp of each user's most recent TeamXpress operation.

Man Page Group:

System Information.

Usage:

```
iwrecentusers [-h|-v|-Gn]
```

-h	Displays usage message.
-v	Displays version information.
-G0	Output is tab-separated. If a field is longer than the tab, then alignment will be erratic.
-G1	Output is neatly formatted (default). Field width is adjusted for the longest value in the column and is aligned.

iwreset.exe

Tells the TeamXpress server to reread the TeamXpress configuration files (`iw.cfg`, `submit.cfg`, `autoprivat.cfg`, and the roles files) and resets `iwserver` based on the values in the configuration files.

Man Page Group:

System Services.

Usage:

```
iwreset [-h|-v]
```

-h	Displays usage message.
-v	Displays version.

Example:

```
>iwreset
```

tells the TeamXpress server to read the TeamXpress configuration files and reset `iwserver`.

Notes:

Exits with 0 on success, non-zero on failure.

All except the following `iw.cfg` settings are reset when `iwreset` executes. You must restart TeamXpress for the following settings to reset.

- Cache size (set in `[iwserver]`)
- File system thread count (set in `[iwserver]`)
- File system active area cache (set in `[iwserver]`)
- File locations (set in `[locations]`)

iwsi.ipl

Available only if the high availability version of TeamXpress is installed. Collects TeamXpress state data about the local environment, including core and configuration files. Files collected are not deleted from their original locations. Data is stored and compressed in an archive file named *si.timestamp* in either *iw-home* or a specified alternate location. The command exits after executing; it does not run as a daemon.

Man Page Group:

System Services.

Usage:

```
iwsi [-h|-v] [file]
```

-h	Displays usage message.
-v	Displays version. Has highest precedence if multiple options are specified.
<i>file</i>	Specifies the full directory path of an alternate archive location. Each archive file name is given a timestamp suffix to distinguish it from others with similar names.

Exit Status:

The following exit values are returned:

0	Successful options processing
-1	Invalid options; could not properly set the path; could not find all the necessary binaries
>1	Other processes returned an error

Notes:

The value of *timestamp* in the archive file name has the format *yyymmdd.hhmmss*.

The archive contains the following files:

Log Files:

```
iwevents.log  
iwserver.log
```

iwtrace.log
iwdeploy.log

Configuration Files:

iw.cfg
\$IW_HOME/local/config/submit.cfg
\$IW_HOME/local/config/autopriate.cfg

Core Files:

system.dmp
memory.dmp

Server and Process Output Files:

iwserver
information<**processid**> (a file containing output from iwversion command)

iwstat.exe

Returns current system activity and displays server state information about whether the server is running, frozen (with `iwfreeze`), flushing, or out of space.

Man Page Group:

System Information.

Usage:

```
iwstat
```

-h	Displays usage message.
-v	Displays version
-c	Displays cache statistics.
-G0	Output is tab-separated.
-G1	Output is neatly formatted (default).

Example:

While a Submit operation is underway,

```
>iwstat
```

returns:

ID	Thread	User	Duration	Operation
0x8738	0xe1	WEBSTER\andre	0.901	SubmitFSE 0x000021000000000000002134
0x873a	0xfa	Administrator	0.000	GetArchiveStatus

While the TeamXpress server is frozen as the result of `iwfreeze`,

```
>iwstat
```

returns:

*** SERVER FROZEN: 55 SECONDS REMAINING ***

ID	Thread	User	Duration	Operation
0x9d	0xf	root	0.000	GetArchiveStatus

The following output from `iwstat` was obtained while the server was running.

Status: Server running

ID	Thread	User	Duration	Operation
0x3f4e	0x17	mroot	0.000	GetArchiveStatus

Minutes	Thruput	Avg op	Load
1	27	0.0189	0.51
15	14	0.0219	0.30
29	9	0.0233	0.20

Explanation: In the most recent complete one-minute time duration, the TeamXpress server handled 27 operations per second. Each operation took an average of 0.0189 seconds per operation (19 ms/operation). The average load over this one minute was 51percent. By comparison, a single-CPU server is fully loaded at 100 percent. A two-cpu server is fully loaded at 200 percent load, etc. Similarly, for a recent complete 15-minute duration, TeamXpress handled 14 operations per second, with an average operation time of 22 ms. Over this time, the server was occupied at 30 percent load.

Comparing the short and long duration figures shows the load trends. For example, a high one-minute load, compared with a lower 60-minute load, tells you that there was a burst of activity. You can also observe that this is not a fast server, because 27 operations per second at 50 percent load is not much throughput. A fast server with fast disks can achieve operation throughput in the 200-500 operations per second range, with operation times in the 1-5 ms/operation range.

Finally, a load that is near or exceeds the number of CPUs on the server over a sustained number of hours may indicate an overloaded server. It may also indicate a need to obtain a more powerful server or a faster backing store volume.

iwteamsite.exe

Available only if the high availability version of TeamXpress is installed. Displays information about the TeamXpress iwtock watchdog daemon and lets you install or uninstall iwtock as a service.

Man Page Group:

System Services.

Usage:

```
iwteamsite [-h|-v|-install|-uninstall|-d]
    -h                Displays usage message.
    -v                Displays version. Has highest precedence if multiple
                    options are specified.
    -install          Installs iwtock as a service. Fails if iwtock is already
                    installed as a service.
    -uninstall        Uninstalls iwtock as a service. Fails if iwtock is not
                    already installed as a service.
    -d                Execute iwtock as a command line process (useful
                    for debugging).
```

Exit Status:

The following exit values are returned:

0	Successful options processing.
-1	Invalid options; could not properly set the path; could not find all the necessary binaries.
>1	Other processes returned an error.
127	Exit was requested.

Notes:

When you execute iwteamsite with an option, it exits upon completion (as opposed to running as a daemon).

See *TeamXpress Administration Guide* for more information about the high availability version of TeamXpress.

iwtestcfg.exe

Returns the submit filtering operation that will be performed upon a file at submission time. For more information about submit filtering, see the *TeamXpress Administration Guide*.

Man Page Group:

System Services.

Usage:

```
iwtestcfg [-h|-v] vpath
```

-h	Displays usage message.
-v	Displays version.
vpath	Specifies a vpath to a file (see page 13).

Example:

```
>iwtestcfg /default/main/WORKAREA/andre/cgi/test.sh
```

returns:

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

which means that TeamXpress is configured to set execute permissions on the specified file at the time that the file is submitted.

iwversion.exe

Returns the current TeamXpress release information. Often used as a test to see if the TeamXpress server is up.

Man Page Group:

System Information.

Usage:

```
iwversion [-h|-v]
```

-h	Displays usage message.
-v	Displays version.

Example:

```
>iwversion
```

returns:

```
iwserver: 4.5 Build 1833 Interwoven 20000812
```


Chapter 3

Development Tools

This chapter contains man pages for the development CLTs from the following groups. All man pages in this chapter are presented as one group, arranged alphabetically.

General Development CLTs let you perform operations on, or retrieve information about, the data in a specific file or TeamXpress object. CLTs in this category are as follows:

CLT	Description	See...
<code>iwattrib.exe</code>	Returns metadata information on any object in the TeamXpress server, including all branches, workareas, editions, staging areas, files, directories, and symlinks.	page 64
<code>iwckrole.exe</code>	Checks whether or not a user can log in with a particular role.	page 75
<code>iwdecode.exe</code>	Decodes HTML-encoded %xx lines to ASCII lines.	page 80
<code>iwencode.exe</code>	Encodes ASCII lines to HTML-encoded %xx lines.	page 89
<code>iwextattr.exe</code>	Sets extended attributes on a file.	page 94
<code>iwlist.exe</code>	Returns a list of all areas contained by a parent areas. For example, <code>iwlist</code> returns all editions, workareas, and branches contained on a parent branch.	page 107
<code>iwmenu.exe</code>	Modifies the Advanced menu on the SmartContext Editing tab.	page 116
<code>iwrename.exe</code>	Renames a file, directory, workarea, edition, or branch.	page 134
<code>iwvpath.exe</code>	Prints all or parts of the version path of a specified object.	page 151

Deployment CLTs let you deploy a TeamXpress edition to a production server. CLTs in this category are as follows:

CLT	Description	See...
iwdeploy.exe	Deploys website content to the production server.	page 83

Branch Operation CLTs let you create and manipulate TeamXpress branches. CLTs in this category are as follows:

CLT	Description	See...
iwmkbr.exe	Creates a new TeamXpress branch.	page 121
iwrnbr.exe	Deletes a TeamXpress branch and all contained areas (workareas, staging areas, and editions).	page 139

Edition Operation CLTs let you create and manipulate TeamXpress editions. CLTs in this category are as follows:

CLT	Description	See...
iwcompress.exe	Compresses and uncompresses editions.	page 78
iwlasted.exe	Returns the name of the last published edition on a branch.	page 106
iwnexted.exe	Returns the name that would be autogenerated for the next edition on a branch.	page 124
iwpublish.exe	Publishes the staging area.	page 127
iwrmed.exe	Deletes an edition.	page 140

Workarea Operation CLTs let you create and manipulate TeamXpress workareas. CLTs in this category are as follows:

CLT	Description	See...
iwevents.exe	Returns a history of submissions or updates of a workarea.	page 90
iwlistlocks.exe	Lists the locks and assignments in a workarea or branch.	page 110
iwlistmod.exe	Lists all modified files and directories in a specified area.	page 111
iwlock.exe	Locks any file or directory in a specified area.	page 113
iwlockinfo.exe	Provides detailed information on any lock.	page 114

iwmkwa.exe	Creates a new TeamXpress workarea.	page 123
iwprv.exe	Allows users to identify whether any file or directory in any workarea is marked private and automatically mark any file as either private or public.	page 125
iwrmw.exe	Deletes a TeamXpress workarea.	page 143
iwsubmit.exe	Submits a file, directory, or entire workarea to the staging area.	page 144
iwunlock.exe	Unlocks any file or directory.	page 149
iwupdate.exe	Updates a TeamXpress workarea with the version of a file or directory in the staging area.	page 150

Version Management CLTs return information about file revisions and versions, and let you manipulate specific versions of files and directories. CLTs in this category are as follows:

CLT	Description	See...
iwcat.exe	Displays any version of a (text) file in TeamXpress.	page 74
iwcmp.exe	Compares any two TeamXpress areas and returns a list of differences.	page 76
iwrevert.exe	Revert to a previous version of a file.	page 136
iwrlog.exe	Shows a revision log for a TeamXpress file.	page 137

Workflow/Job CLTs let you instantiate or destroy jobs, view the state of any object in the workflow system, add files to a particular task, and otherwise interact with running jobs. These tasks are typically performed by Administrators or Master users. CLTs in this category are as follows:

CLT	Description	See...
iwaddtaskfile.exe	Adds a file to a job task that is part of a job instance already created on the TeamXpress server.	page 63
iwcallback.exe	Tells the TeamXpress server that the program corresponding to an external or cgi task is finished. Passes a return code to the server.	page 73
iwgetwfobj.exe	Prints the state of any part of the server workflow subsystem in XML.	page 96
iwinvokejob.exe	Starts a job whose instance has already been created on the TeamXpress server.	page 103

<code>iwjobc.exe</code>	Creates a job instance (based on a job specification file) on the TeamXpress server.	page 104
<code>iwjobvariable.exe</code>	Manipulates workflow variables.	page 105
<code>iwqueryjobs.exe</code>	Prints a list of overall job information based on a query from stdin.	page 129
<code>iwquerytasks.exe</code>	Prints a list of job task information based on a query from stdin.	page 130
<code>iwretrywfop.exe</code>	Retries submission or task update after resolution of conflicts that originally prevented those operations.	page 135
<code>iwrmdir.exe</code>	Unconditionally removes an entire job instance from the TeamXpress server.	page 141
<code>iwrmtaskfile.exe</code>	Removes a file from a job task.	page 142
<code>iwtaketask.exe</code>	Assigns a shared task to a single user.	page 146
<code>iwtaskselect.exe</code>	Marks a user's job task finished and selects a successor task.	page 147
<code>iwundochoice.exe</code>	Reverses a user-chosen transition from a user task or group task.	page 148

iwaddtaskfile.exe

Adds a file to a specified task in an instantiated job.

Man Page Group:

Workflow/Job.

Usage:

```
iwaddtaskfile [-h|-v] [-s servername] taskID file [comment]
```

<i>-h</i>	Displays usage message.
<i>-v</i>	Displays version.
<i>-s servername</i>	Specifies the server on which the job is instantiated. Server names can be specified by name or IP address.
<i>taskID</i>	Specifies the task to which the file will be added. Task IDs are represented as integers.
<i>file</i>	Specifies the area-relative path of the file that will be added to the job task.
<i>comment</i>	Comment attached to the file.

Example:

The following command adds the file `/default/main/WORKAREA/eng/content.txt` to job 7734 on the server `production.example.com` (assuming the command is issued from the `eng` directory):

```
>iwaddtaskfile -s production.example.com 7734 content.txt
```

iwattrib.exe

Returns metadata information on any object in the TeamXpress server, including all branches, workareas, editions, staging areas, files, directories, and symlinks.

Man Page Group:

General Development.

Usage:

To return information on a specific attribute of an object:

```
iwattrib [-h|-v]
iwattrib objectvpath attribute
iwattrib -o objid attribute
```

To list available attributes for an object:

<code>iwattrib -l <i>objectvpath</i></code>	
<code>iwattrib -l -o <i>objid</i></code>	
<code>-h</code>	Displays usage message.
<code>-v</code>	Displays version.
<code><i>objectvpath</i></code>	Vpath of the object (see page 13).
<code><i>attribute</i></code>	Attribute to return information about.
<code>lockmodel</code>	Returns the locking model of a branch.
<code>-o <i>objid</i></code>	Objid of the object. An object can be specified by its full vpath or its object ID.
<code>-l</code>	Lists the attributes available for the specified object.

Examples:

```
>iwattrib -l main
```

returns the names of all the attributes available for the main branch:

```
cid
objid
```

crdate
creator
owner
group
name
label
moddate
busy
rootedition
firstedition
lastedition
parentbranch
archive
branchcount
editioncount
workareacount

>iwattrib main objid

returns the objid of the main branch:

0x000102500000000000000006d

>iwattrib main archive

returns the objid of the TeamXpress archive containing the main branch:

0x00010010000000000000000001

>iwattrib -n main archive

returns the name of the archive containing the main branch:

default

The values of *attribute* available for each type of object are:

Server

Name	Example	Description
cid	0x00000051	Class ID
objid	0x000000000000000000000000a	Object ID
crdate	Thu Aug 24 13:13:30 2000	Creation date
creator	smtp	Uid of the creator
owner	smtp	Uid of the owner
group	root	Gid of the group for sharing
name	IWSERVER	Name
hostname	chocolate	Host name of the TeamXpress server
ipaddress	0.0.0.0	IP address of the TeamXpress server
verstring	iwserver: 4.5.0 Build 1776 Interwoven 200713	Version of TeamXpress running on the TeamXpress server

Archive

Name	Example	Description
cid	0x00010010	Class ID
objid	0x000000000000000000000000a	Object ID
crdate	Thu Aug 24 13:13:30 2000	Creation date
creator	smtp	Uid of the creator
owner	smtp	Uid of the owner
group	root	Gid of the group for sharing
name	default	Name
label	Default archive	Comment attached at time of creation
moddate	Thu Aug 24 13:13:30 2000	Date modified
busy	TRUE or FALSE	Tells whether the archive is busy or not
mainbranch	0x000022500000000000000006d	Object id of the main branch

Branch

Name	Example	Description
cid	0x00010250	Class ID
objid	0x000022500000000000000006d	Object ID
crdate	Tue Aug 22 18:11:41 2000	Creation date
creator	andre	Uid of the creator
owner	andre	Uid of the owner
group	marketing	Gid of the group for sharing
name	dev	Name
label	Main branch	Comment attached at time of creation
moddate	Tue Aug 22 18:11:41 2000	Date modified
busy	TRUE or FALSE	Tells whether the branch is busy or not
rootedition	0x0000201000000000000000077 or INITIAL	Object ID of the edition that the branch is based on (-n returns the name of the edition)
firstedition	0x0000201000000000000000051e or INITIAL	Object ID of the first edition on the branch (-n returns the name of the edition)
lastedition	0x00002010000000000000000722 or ed_0001	Object ID of the latest edition on the branch (-n returns the name of the edition)
parentbranch	0x000022500000000000000006d or main	Object ID of the parent branch (-n returns the name of the branch)
archive	0x0000202000000000000000001 or default	Object ID of the archive (-n returns the name of the archive)
branchcount	1	Number of sub-branches present on the branch (does not recurse to sub-sub branches)
editioncount	2	Number of editions on the branch
workareacount	4	Number of workareas on the branch

Edition

Name	Example	Description
cid	0x00010020	Class ID
objid	0x0000201000000000000000722	Object ID
crdate	Mon Aug 28 13:45:57 2000	Creation date
creator	andre	Uid of the creator
owner	andre	Uid of the owner
group	marketing	Gid of the group for sharing
name	ed_0001	Name
label	baseline edition	Comment attached at time of creation
busy	TRUE or FALSE	Tells whether the edition is busy or not
rootdir	0x0000072200000000000000520 or /	Object ID of the root directory of the edition (-n returns the name of the directory)
nextedition	0x0000201000000000000000722 or ed_0002	Object ID of the edition after the specified edition (-n returns the name of the edition)
prevedition	0x000020100000000000000051e or INITIAL	Object ID of the edition before the specified edition (-n returns the name of the edition)
branch	0x0000225000000000000000514 or dev	Object ID of the edition's branch (-n returns the name of the branch)
sequenceno	1	Sequence number

Workarea

Name	Example	Description
cid	0x00010100	Class ID
objid	0x0000210000000000000000522	Object ID
crdate	Thu Aug 24 13:13:30 2000	Creation date
creator	andre	Uid of the creator
owner	andre	Uid of the owner
group	marketing	Gid of the group for sharing

Name	Example	Description
name	andre	Name
label	Andre's workarea	Comment attached at time of creation
moddate	Thu Aug 24 13:13:30 2000	Date modified
modified	TRUE or FALSE	Tells whether the object has been modified or not
busy	TRUE or FALSE	Tells whether the workarea is busy or not
rootdir	0x0000052200000000000000520 or /	Object ID of the root directory of the workarea (-n returns the name of the directory)
baseedition	0x000020100000000000000051e or INITIAL	Object ID of the edition that the workarea is based on (-n returns the name of the edition)
branch	0x0000225000000000000000514 or dev	Object ID of the workarea's branch (-n returns the name of the branch)

Submit Event

Name	Example	Description
cid	0x00002258	Class ID
objid	0x000022580000000000000006bb	Object ID of the submit event
workarea	0x0000210000000000000000522	Object ID of the submitting workarea
submitter	andre	Uid of the user who submitted
sdate	Mon Aug 28 13:45:43 2000	Date of submission
entries	5	Number of files and directories submitted
submit_cmt	first submit to staging area	Submit-level comment
submit_info	keyword	Info field

Update Event

Name	Example	Description
cid	0x0000225a	Class ID
objid	0x0000225a00000000000000a9a	Object ID of the update event
workarea	0x0000210000000000000000522	Object ID of the updated workarea
updater	andre	Uid of the user who updated the workarea
update	Mon Oct 2 16:57:12 2000	Date the workarea was updated
entries	59	Number of files and directories updated

Staging Area

Name	Example	Description
cid	0x00010255	Class ID
objid	0x0000225500000000000000521	Object ID
crdate	Thu Aug 24 13:13:06 2000	Creation date
creator	andre	Uid of the creator
owner	andre	Uid of the owner
group	marketing	Gid of the group for sharing

Name	Example	Description
name	STAGING	Name
label		Comment attached at time of creation
moddate	Thu Aug 24 13:13:06 2000	Date modified
modified	TRUE or FALSE	Tells whether the object has been modified or not
busy	TRUE or FALSE	Tells whether the staging area is busy or not
rootdir	0x0000052100000000000000520 or /	Object ID of the root directory of the staging area (-n returns the name of the directory)
branch	0x0000225000000000000000514 or dev	Object ID of the staging area's branch (-n returns the name of the branch)

File or Directory

Name	Example	Description
cid	0x08001000	Class ID
objid	0x000005220000058e0000080e	Object ID
crdate	Fri Sep 1 15:55:51 2000	Creation date
creator	andre	Uid of the creator
owner	andre	Uid of the owner
group	marketing	Gid of the group for sharing
name	index.html	Name of the file or directory
label	first submit	Latest comment attached
moddate	Fri Sep 1 16:24:13 2000	Date modified
modified	TRUE or FALSE	Tells whether the object has been modified or not
currdirectory	0x00000522000005200000058e	Object ID of the directory containing the specified item (-n returns the name of the directory)
area	0x000021000000000000000522	Object ID of the area containing the specified item (-n returns the name of the area)

Name	Example	Description
areacid	0x00002100	Class ID of the area containing the specified item
branch	0x00002250000000000000000514	Object ID of the branch containing the specified item (-n returns the name of the branch)
archive	0x00002020000000000000000001	Object ID of the archive containing the specified item (-n returns the name of the archive)
size	1949	Size of the file or directory (in bytes)
lastuser	andre	User who last submitted the file
istagged	TRUE or FALSE	File is locked or assigned
nfsmode	0x000001ff	NFS permission bits
isdir	TRUE or FALSE	TRUE if the specified item is a directory, FALSE if it is not
taglabel	Locked by launching edit	Comment attached to tag (e.g. a lock or assignment)
revisionnumber	3	Ordinal representing the version of the file submitted on a branch
revisionbranch	0x00002250000000000000000514	Object ID of the branch on which this version of the file has been submitted
submitevent	0x00002258000000000000000ab8	Object ID of the submit event for the most recent submission of this file.
history	first submit	Comments associated with file at time of submission (includes both submission comment and individual file comment)
prevversion	0x0000052100000aea00000815	Object ID of the previous version

iwcallback.exe

Notifies the server that the program corresponding to an <externaltask/> or <cgitask/> is finished and passes the server a return code.

Man Page Group:

Workflow/Job.

Usage:

```
iwcallback [-h|-v] [-s servername] taskID returnCode [comment]
```

<i>-h</i>	Displays usage message.
<i>-v</i>	Displays version.
<i>-s servername</i>	Specifies the server on which the job is instantiated. Server names can be specified by name or IP address.
<i>taskID</i>	Specifies the task to which the file will be added. Task IDs are represented as integers.
<i>returnCode</i>	Specifies the notification given to the server upon program completion. Represented as an integer from 0 to n.
<i>comment</i>	An optional comment.

Example:

The following command sends the return code 2 to the server `production.example.com` when the program named in task 7734 finishes running:

```
>iwcallback -s production.example.com 7734 2
```

iwcat.exe

iwcat is cat for TeamXpress files. If you specify a vpath to the file, you will see the current version of the file in the area you specify. You can view any previous version of a file by specifying the objid for that version.

Man Page Group:

Version Management.

Usage:

```
iwcat [-h|-v] filevpath
```

```
iwcat -o fileobjid directoryvpath
```

-h	Displays usage message.
-v	Displays version.
<i>filevpath</i>	Vpath of the file to display (used to display the current version in an area) (see page 13).
-o <i>fileobjid</i>	Objid of the file to display (used to display any version).

Examples:

```
>iwcat main/WORKAREA/andre/htdocs/about2.html
```

displays the version of about2.html in workarea andre.

```
>iwcat -o 0x0000007b0000007d000000bb main/STAGING
```

displays the version of the file that has the objid 0x0000007b0000007d000000bb (in this case, the second most recent version of about2.html). *directoryvpath* can point to any area containing the file.

iwckrole.exe

Checks whether the specified user has the specified role.

Man Page Group:

General Development.

Usage:

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

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

Exits with 0 on successful authorization, non-zero on failure.

Example:

```
>iwckrole admin andre
```

returns:

```
YES
```

iwcmp.exe

Compares any two TeamXpress areas and reports exhaustive list of all modifications. For a full description of TeamXpress comparison, see the *TeamXpress User's Guide*.

Man Page Group:

Version Management.

Usage:

```
iwcmp [-h|-v] area1vpath area2vpath
```

-h	Displays usage message.
-v	Displays version.
<i>area1vpath</i>	Vpath of area1 (see page 13).
<i>area2vpath</i>	Vpath of area2 (see page 13).

The output is in the form:

```
status-in-area1 status-in-area2 path-to-file-or-directory
```

where status is indicated by a combination of the following symbols:

-	file or directory does not exist in this area
f	file exists
d	directory exists
s	symlink exists
+	item has been modified
x	file or directory has been deleted
p	file or directory is marked private

Example:

```
>iwcmp main/branch1/WORKAREA/wa1 main/branch1/WORKAREA/wa2
```

returns:

x+	f	htdocs/about.html
f+	-	htdocs/about2.html

```
d+      -      htdocs/support/  
f+      -      htdocs/main.html  
f       fp+    htdocs/products.html
```

showing that the file `about.html` has been renamed to `about2.html` in the first area, so `about.html` appears as deleted and `about2.html` appears as new in the first workarea. A directory named `support` has been created in the first area and does not exist in the second. The file `main.html` only exists in the first area specified and has never existed in the second area, and the file `products.html` exists in both areas, but in the second area it has been marked private and modified.

iwcompress.exe

Allows system administrators to compress and uncompress any or all TeamXpress editions. For more information about TeamXpress compression, see the *TeamXpress User's Guide*.

Man Page Group:

Edition Operation.

Usage:

```
iwcompress [-h|-v] branchvpath edition [+|-]
```

-h	Displays usage message.
-v	Displays version.
<i>branchvpath</i>	Vpath of branch (see page 13).
<i>edition</i>	Name of edition.
+	Uncompress the specified edition.
-	Compress the specified edition.

Example:

```
>iwcompress main/branch1 ed_002 -
```

compresses the edition ed_002 on the sub-branch branch1.

```
>iwcompress main/branch1 ed_002 +
```

decompresses the edition ed_002 on the sub-branch branch1.

iwdecode.exe

Decodes HTML-encoded %xx lines to ASCII lines. Receives HTML-encoded lines through `stdin`.

Man Page Group:

General Development.

Usage:

```
iwdecode [-h|-v]
```

-h

Displays usage message.

-v

Displays version.

Example:

```
>iwdecode
```

```
Locked%20by%20launching%20edit
```

```
^D
```

returns:

```
Locked by launching edit
```

iwdeploy.exe

Deploys website content to the production server. OpenDeploy must be installed for this CLT to be available. For more information on deployment, see the *OpenDeploy Administration Guide*.

Man Page Group:

Deployment.

Usage:

```
iwdeploy [-h] [(-f|-fs) srcConfigFile] [-fd destConfigFile] [-r] [-v] [-d]
[-o package_name] [-t tempFilePath] deployment_name [param=value]+
```

-h	Displays usage message.
(-f -fs) <i>srcConfigFile</i>	Specifies source deployment configuration file (defaults to <code>/etc/iw.deploy.cfg</code>). For information on deployment configuration files, consult Interwoven Professional Services. -f is an option provided for backwards-compatibility.
-fd <i>destConfigFile</i>	Specifies destination configuration file (defaults to <code>/etc/iw.deployee.cfg</code>). For information on deployment configuration files, consult Interwoven Professional Services.
-r	Reverse deployment.
-v	Verbose logging.
-d	Debugging output.
-o <i>package_name</i>	Creates a deployment package that can be transferred to the production server by alternate means (e.g., via email, manually, etc.). This option can only be used with TeamXpress-based configurations.
-t <i>tempFilePath</i>	Specifies directory path for temporary file created during deployment (needs space for up to ~5 Mb).
<i>deployment_name</i>	Name of the section of <i>srcConfigFile</i> to invoke.

`[param=value] +`

Parameters that override configuration file parameters.

iwencode.exe

Encodes ASCII lines to HTML-encoded %0xx lines. Takes input from `stdin`.

Man Page Group:

General Development.

Usage:

```
iwencode [-h|-v]
```

-h	Displays usage message.
-v	Displays version.

Example:

```
>iwencode
```

```
Locked by launching edit
```

```
^D
```

```
returns:
```

```
Locked%20by%20launching%20edit
```


iwevents.exe

Returns a history of submissions or updates of a workarea.

Man Page Group:

Workarea Operation.

Usage:

```
iwevents [-h|-v] -s[ubmit] |-u[pdate] [-n[oheder]] [-i[ds]] vpath  
latest|all
```

```
iwevents -s[ubmit] |-u[pdate] [-n[oheder]] id
```

-h	Displays usage message.
-v	Displays version.
-s	Gets a history of Submit operations.
-u	Gets a history of Get Latest operations.
-i	Displays only the objids of the Submit or Get Latest events.
<i>vpath</i>	Specifies a vpath to a workarea, edition, or staging area (see page 13).
latest	Gets only the latest submit/update.
all	Gets all currently logged submits/updates.
<i>id</i>	Specifies the objid for the event.

When -s is specified, each entry will have ten fields:

TIME	time of submission
UPDATE FORCEMODIFY	type of update: ADD, MODIFY, DELETE, FORCEDELETE,
TYPE	DIRECTORY or FILE
PATH	path of the file or directory submitted

USER	user who submitted the file or directory
COMMENT	comment attached to the file at time of submission
APPROVE	TRUE or FALSE
EVENTID	objid of the submit event
FILEID	objid of the file
WORKAREAID	objid of the workarea

When `-u` is specified, each entry will have eight fields:

TIME	time of update
UPDATE	type of update: ADD, MODIFY, DELETE
TYPE	DIRECTORY or FILE
PATH	path of the file or directory updated
USER	user who performed the update
EVENTID	objid of the update event
FILEID	objid of the file
SOURCEAREAID	objid of the area files were updated from

Examples:

```
>iwevents -s main/WORKAREA/andre all
```

returns a history of all the submissions from workarea “andre” and includes the headers for each entry:

TIME	UPDATE	TYPE	PATH
USER	COMMENT	APPROVE	EVENTID
FILEID	WORKAREAID		
[Fri Apr 21 11:49:13 2000]	ADD	DIRECTORY	/htdocs/
andre	<none>	FALSE	0x0001025800000000000000e2
0x0000007a00000079000000e3			0x00010100000000000000007b
[Fri Apr 21 11:59:37 2000]	ADD	FILE	/htdocs/CorpGuide2.pdf
andre	<none>	FALSE	0x000102580000000000000011f
0x0000007b000000e400000117			0x00010100000000000000007b
[Fri Apr 21 11:59:37 2000]	ADD	FILE	/htdocs/logo.PSD
andre	<none>	FALSE	0x000102580000000000000011f
0x0000007b000000e400000119			0x00010100000000000000007b

```
>iwevents -s -n main/WORKAREA/andre latest
```

returns only the files submitted in the most recent **Submit** operation and strips the headers:

[Fri Apr 21 11:59:37 2000]	ADD	FILE	/htdocs/CorpGuide2.pdf
andre	<none>	FALSE	0x000102580000000000000011f
0x0000007b000000e400000117			0x00010100000000000000007b
[Fri Apr 21 11:59:37 2000]	ADD	FILE	/htdocs/logo.PSD
andre	<none>	FALSE	0x000102580000000000000011f
0x0000007b000000e400000119			0x00010100000000000000007b

```
>iwevents -u main/WORKAREA/andre all
```

returns a list of all files copied in to the workarea during all **Get Latest** operations and includes the headers:

TIME	UPDATE	TYPE	PATH
USER	EVENTID	FILEID	
SOURCEAREAID			



```
[Fri Apr 21 13:34:30 2000] MODIFYFILE/htdocs/index.html andre  
0x0001025a000000000000000013b 0x0000013000000007d00000136  
0x00010256000000000000000007a
```

iwextattr.exe

Allows you to set and look up extended attributes on a file. The file is specified with a full directory path, including the server.

Extended attributes can use any naming scheme, but one common method is to name attributes by general class, then by their specific description. For example:

```
description/language=English
description/keywords=marketing, software, enterprise
```

The attributes mentioned above both fall into the descriptive category, but one stores language information and the other stores keywords. You might also have attributes that describe the file's relationships to other files, such as its document of origin or the files it links to, e.g.:

```
/relationships/source
/relationships/links/outbound
/description/keywords
/description/language
```

Man Page Group:

General Development.

Usage:

```
iwextattr [-h|-v] [-s attribute=value | -g attr | -d attr | -l] [-f] file
```

Options:

-h	Displays usage message.
-v	Displays version.
-s <i>attribute=value</i>	Sets <i>attribute</i> to <i>value</i> .
-g <i>attribute</i>	Gets the value of <i>attribute</i> .
-d <i>attribute</i>	Deletes attribute <i>attribute</i> .
-f	Reads/writes <i>attribute</i> value from stdin/stout for -s and -g options.
-l	Lists attributes and values.
<i>file</i>	Full directory path to a file.

If no option is specified, attributes are listed as with `-l`.

When attribute values are printed, unprintable (non-ASCII) characters are output as `*`.

Examples:

```
>iwextattr -s /description/language=English  
Y:\default\main\dev\WORKAREA\andre\products\index.html
```

sets the `/description/language` attribute on the file `products\index.html` in `workarea andre`.

```
>iwextattr -l Y:\default\main\dev\WORKAREA\andre\products\index.html
```

returns a list of all the attributes on `products\index.html`, e.g.:

```
/relationships/source=products.doc  
/relationships/links/outbound=/test.html;/mktg/index.html;/eng/index.html  
  
/description/keywords=marketing,software,enterprise  
/description/description=Overview of our suite of examples  
/description/launch_date=March 19th, 1999  
/description/expire_date=May 21st, 1999  
/description/language=English  
/description/contributors=Mark, Kevin, Stacy  
/description/audience=Marketing
```

```
>iwextattr -g /description/language  
Y:\default\main\dev\WORKAREA\andre\products\index.html
```

returns the value of the `/description/language` attribute on `products\index.html`, e.g.:

```
English
```

iwgetwfobj.exe

Displays the state of any part of the server workflow subsystem in XML. It can be called in three ways. First, by passing no arguments, in which case it displays the workflow registry. Second, by passing a single job ID, in which case it displays the state of that job. Third, by a task ID, in which case it displays the state of that particular task. Output format is described below in the “DTDs” section.

Man Page Group:

Workflow/Job.

Usage:

```
iwgetwfobj [-h|-v] [-r] [-s servername] [jobID|taskID]
```

<code>-h</code>	Displays usage message.
<code>-v</code>	Displays version.
<code>-r</code>	Prints all objects below the specified object recursively.
<code>-s <i>servername</i></code>	Specifies the TeamXpress server to which to connect. It can be specified by name or IP address. The default value is the current TeamXpress server.
<code><i>jobID</i></code>	Specifies the job to report information about. Job IDs are represented as integers.
<code><i>taskID</i></code>	Specifies the task to report information about. Task IDs are represented as integers.

Example:

The following command generates information about task 7734:

```
>iwgetwfobj 7734
```

DTDs:

Output from `iwgetwfobj` conforms to the following DTDs:

wfregistry

The `<wfregistry/>` element is the repository of all jobs instantiated in the server. The `<activeworkflows/>` element is a list of all jobs actually running.

```
<!ELEMENT wfregistry (workflows, activeworkflows, tasks,
    activetasks, workflow*)>
  <!ELEMENT workflows (id*)>
    <!ELEMENT id EMPTY>
    <!ATTLIST id v CDATA #REQUIRED>
  <!ELEMENT activeworkflows (id*)>
  <!ELEMENT tasks (id*)>
  <!ELEMENT activetasks (id*)>
```

workflow

When you run `iwgetwfobj` passing a single job ID, you get information about a particular job in the system. The `active` attribute indicates whether the job is running. The `<tasks/>` element contains the tasks owned by this job. `<starttasks/>` are those tasks that are active at job invocation. `<activetasks/>` are those tasks currently active.

```
<!ELEMENT workflow (description, tasks, starttasks, activetasks,
    locks, events, immediatetasks,
    variables, (updatetask|submittask|usertask
    |externaltask|cgitask|grouptask|endtask)*)>
  <!ATTLIST workflow active (t|f) #REQUIRED
    name NMTOKEN #REQUIRED
    owner CDATA #REQUIRED
    activationtime CDATA #REQUIRED
    creator CDATA #REQUIRED>
  <!ELEMENT description (#PCDATA)>
  <!ELEMENT starttasks (id*)>
  <!ELEMENT locks (lock*)>
    <!ELEMENT lock EMPTY>
    <!ATTLIST lock branch CDATA #REQUIRED
      owner CDATA #REQUIRED
      path CDATA #REQUIRED
      workarea CDATA #REQUIRED>
  <!ELEMENT events (event*)>
    <!ELEMENT event (arg*)>
    <!ELEMENT arg EMPTY>
```



```

    <!ATTLIST arg v CDATA #REQUIRED>
  <!ATTLIST event date CDATA #REQUIRED
    name CDATA #REQUIRED
    other CDATA #REQUIRED
    othername CDATA #REQUIRED
    role CDATA #REQUIRED
    task CDATA #REQUIRED
    taskname NMTOKEN #REQUIRED
    user CDATA #REQUIRED>
<!ELEMENT immediatetasks (id*)>
<!ELEMENT variables (variable*)>
  <!ELEMENT variable EMPTY>
    <!ATTLIST variable key CDATA #REQUIRED
      value CDATA #REQUIRED>

```

tasks

When you run `iwgetwfobj` passing a single task ID, you get information about that particular task in the job. The following are DTDs for the different task types. Most of the attributes and subelements correspond to the equivalent items in the configuration file.

```

<!ELEMENT updatetask (description, areavpath, activation,
  inactivate, resets, predecessors, files,
  comments, activationtime, unactivationtime,
  successorset, srcareavpath)>
  <!ATTLIST updatetask active (t|f) #REQUIRED
    name NMTOKEN #REQUIRED
    overwritemod (t|f) #REQUIRED
    owner CDATA #REQUIRED
    readonly (t|f) 'f'
    lock (t|f) 'f'
    updatedelete (t|f) #REQUIRED
    needsattention (t|f) 'f'
    owningworkflow CDATA #REQUIRED
    tryingtolock (t|f) #REQUIRED
    unactivatable (t|f) #REQUIRED>
<!ELEMENT areavpath EMPTY>
  <!ATTLIST areavpath v CDATA #REQUIRED>
<!ELEMENT activation (pred|or|and|not)?>
  <!ELEMENT or ((pred|or|and|not)*)>
  <!ELEMENT and ((pred|or|and|not)*)>

```



```
<!ELEMENT not (pred|or|and|not)>
<!ELEMENT pred EMPTY>
  <!ATTLIST pred v CDATA #REQUIRED>
<!ELEMENT inactivate (pred*)>
<!ELEMENT resets (reset*)>
  <!ELEMENT reset EMPTY>
  <!ATTLIST reset v CDATA #REQUIRED>
<!ELEMENT predecessors (predecessor*)>
  <!ELEMENT predecessor EMPTY>
  <!ATTLIST predecessor hasactivated (t|f) #REQUIRED>
  <!ATTLIST predecessor id CDATA #REQUIRED>
<!ELEMENT files (file*)>
  <!ELEMENT file (comments)>
  <!ATTLIST file deleted (t|f) #REQUIRED
    path CDATA #REQUIRED>
<!ELEMENT comments (comment*)>
  <!ELEMENT comment (#PCDATA)>
  <!ATTLIST comment date CDATA #REQUIRED
    task CDATA #REQUIRED
    user CDATA #REQUIRED>
<!ELEMENT activationtime EMPTY>
  <!ATTLIST activationtime v CDATA #REQUIRED>
<!ELEMENT unactivationtime EMPTY>
  <!ATTLIST unactivationtime v CDATA #REQUIRED>
<!ELEMENT successorset (succ*)>
  <!ATTLIST successorset description CDATA #REQUIRED>
  <!ELEMENT succ EMPTY>
  <!ATTLIST succ v NMTOKEN #REQUIRED>
<!ELEMENT srcareavpath EMPTY>
  <!ATTLIST srcareavpath v CDATA #REQUIRED>
```

The `needsattention` attribute indicates that the submit operation has been attempted but failed because of conflicts. It is up to the user interface to resolve conflicts, and then retry the operation.

Most of the following attributes and subelements are common to all tasks. The `active` attribute tells whether the task is active. The `<owningworkflow/>` element identifies the tasks owning the job. The `<predecessors/>` element shows all possible predecessor tasks and whether they have signalled this task.

```
<!ELEMENT submittask (description, areavpath, activation,
```

```

    inactivate, resets, predecessors,
    files, comments, activationtime,
    unactivationtime, successorset)>
<!ATTLIST submittask active (t|f) #REQUIRED
    name NMTOKEN #REQUIRED
    overrideconflicts (t|f) #REQUIRED
    owner CDATA #REQUIRED
    readonly (t|f) 'f'
    lock (t|f) 'f'
    skipconflicts (t|f) #REQUIRED
    skiplocked (t|f) #REQUIRED
    unlock (t|f) #REQUIRED
    needsattention (t|f) 'f'
    owningworkflow CDATA #REQUIRED
    tryingtolock (t|f) #REQUIRED
    unactivatable (t|f) #REQUIRED>

<!ELEMENT externaltask (description, areavpath, activation,
    inactivate, resets, predecessors, files,
    comments, activationtime, unactivationtime,
    successors, transitionmade, command)>
  <!ATTLIST externaltask active (t|f) #REQUIRED
    name NMTOKEN #REQUIRED
    owner CDATA #REQUIRED
    readonly (t|f) 'f'
    lock (t|f) 'f'
    owningworkflow CDATA #REQUIRED
    tryingtolock (t|f) #REQUIRED
    unactivatable (t|f) #REQUIRED
    undoable (t|f) #REQUIRED>
<!ELEMENT successors (successorset*)>
<!ELEMENT command EMPTY>
  <!ATTLIST command v CDATA #REQUIRED>

<!ELEMENT cgitask (description, areavpath, activation, inactivate,
    resets, predecessors, files, comments,
    activationtime, unactivationtime, successors,
    transitionmade, command)>
<!ATTLIST cgitask active (t|f) #REQUIRED
    name NMTOKEN #REQUIRED
    owner CDATA #REQUIRED

```



```
readonly (t|f) 'f'  
lock (t|f) 'f'  
owningworkflow CDATA #REQUIRED  
tryingtolock (t|f) #REQUIRED  
unactivatable (t|f) #REQUIRED  
undoable (t|f) #REQUIRED>
```

The `undoable` attribute in the following sections is turned on for those `<usertask/>` and `<grouptask/>` elements that have had a successor set chosen for them that can be revoked. Undoable tasks show up in the user interface. Using the CLT `iwundochoice` a user can take back the current choice.

```
<!ELEMENT usertask (description, areavpath, activation, inactivate,  
resets, predecessors, files, comments,  
activationtime, unactivationtime, successors,  
transitionmade)>
```

```
<!ATTLIST usertask active (t|f) #REQUIRED  
name NMTOKEN #REQUIRED  
owner CDATA #REQUIRED  
readonly (t|f) 'f'  
lock (t|f) 'f'  
undoable (t|f) #REQUIRED  
owningworkflow CDATA #REQUIRED  
tryingtolock (t|f) #REQUIRED  
unactivatable (t|f) #REQUIRED>
```

```
<!ELEMENT transitionmade EMPTY>
```

```
<!ATTLIST transitionmade v CDATA #REQUIRED>
```

```
<!ELEMENT grouptask (description, areavpath, activation,  
inactivate, resets, predecessors, files,  
comments, activationtime, unactivationtime,  
successors, transitionmade, users)>
```

```
<!ATTLIST grouptask active (t|f) #REQUIRED  
name NMTOKEN #REQUIRED  
owner CDATA #REQUIRED  
readonly (t|f) 'f'  
lock (t|f) 'f'  
undoable (t|f) #REQUIRED  
owningworkflow CDATA #REQUIRED  
tryingtolock (t|f) #REQUIRED
```

```
        unactivatable (t|f) #REQUIRED
        committed (t|f) #REQUIRED>
<!ELEMENT users (user|group)*>
  <!ELEMENT user EMPTY>
    <!ATTLIST user v CDATA #REQUIRED>
  <!ELEMENT group EMPTY>
    <!ATTLIST group v CDATA #REQUIRED>

<!ELEMENT endtask (description, areavpath, activation, inactivate,
  resets, predecessors, files, comments,
  activationtime, unactivationtime)>
<!ATTLIST endtask active (t|f) #REQUIRED
  name NMTOKEN #REQUIRED
  owner CDATA #REQUIRED
  owningworkflow CDATA #REQUIRED
  tryingtolock (t|f) #REQUIRED
  unactivatable (t|f) #REQUIRED>
```

iwinvokejob.exe

Turns on or runs a job that has been instantiated in the server.

Man Page Group:

Workflow/Job.

Usage:

```
iwinvokejob [-h|-v] [-s servername] jobID
```

<code>-h</code>	Displays usage message.
<code>-v</code>	Displays version.
<code>-s <i>servername</i></code>	Specifies the TeamXpress server to which to connect. It can be specified by name or IP address. The default value is the current TeamXpress server.
<code><i>jobID</i></code>	Specifies which job to run. Job IDs are represented as integers.

Example:

The following command starts job 7734 on the server `production.example.com`:

```
>iwinvokejob -s production.example.com 7734
```

iwjobc.exe

Instantiates a job into the server based on a specified job specification file and prints (in decimal) the ID of the job.

Man Page Group:

Workflow/Job.

Usage:

```
iwjobc [-h|-v] [-s servername] workflowfile
```

<code>-h</code>	Displays usage message.
<code>-v</code>	Displays version.
<code>-s <i>servername</i></code>	Specifies the TeamXpress server to which to connect. It can be specified by name or IP address. The default value is the current TeamXpress server.
<code><i>workflowfile</i></code>	Specifies the area-relative path name of the workflow specification file.

Example:

The following command instantiates the job defined in `config.xml`:

```
>iwjobc config.xml
```

iwjobvariable.exe

Manipulates workflow variables.

Man Page Group:

Workflow/Job.

Usage:

```
iwjobvariable [-h|-v] [-s servername]
               (-g key|-c key value|-t key value|-d key) workflowid
  -h           Print help.
  -v           Print version.
  -s servername Specifies the TeamXpress server to which to connect.
                It can be specified by name or IP address. The default
                value is the current TeamXpress server.
  -g key       Get value of workflow variable key. Prints value to
                standard output.
  -c key value Create variable key with value value. It is an error if
                key already exists.
  -t key value Set variable key to value value. If variable key does
                not exist, create it.
  -d key       Delete variable key.
  workflowid  Specifies the identifier of the workflow.
```

Example:

```
>iwjobvariable -g foo
```

gets the value of the *foo* job variable.

iwlasted.exe

Returns the name of the last published edition on a branch.

Man Page Group:

Edition Operation.

Usage:

```
iwlasted [-h|-v] branchvpath
```

-h

Displays usage message.

-v

Displays version.

branchvpath

Vpath of a branch (see page 13).

Example:

```
>iwlasted main
```

returns:

```
ed_0001
```

iwlist.exe

Returns list of all areas contained by a parent area.

Man Page Group:

General Development.

Usage:

```
iwlist [-h|-v] [-a|-l|-s] [-m [-u user]] vpath
```

-h	Displays usage message.
-v	Displays version.
-a	Lists all the components of a branch.
-l	Long listing (for iwbackup).
-s	Short listing.
-m	Displays all files modifiable by <i>user</i> .
-u <i>user</i>	Specifies the username of a TeamXpress user (defaults to the current user).
<i>vpath</i>	Specifies the vpath of the parent area (see page 13).

Examples:

```
>iwlist main
```

returns:

```
branch1  
branch2
```

Use the -a tag to include all editions and workareas on the branch.

```
>iwlist -a main
```

returns:

```
wa1
wa2
INITIAL
ed_001
branch1
branch2
```

```
>iwlist -a -l main
```

returns a full list of objids, comments, and base editions for all sub-branches, workareas, and editions on the branch:

```
wa1
andre's workarea
2205
INITIAL
-----
wa2
chris's workarea
2205
ed_0001
-----
INITIAL
Initial empty edition
0
0
-----
ed_001
2205
INITIAL
1
-----
branch1
test branch
2205
ed_002
-----
```

```
branch2  
redesign branch  
31001  
INITIAL  
-----
```

iwlistlocks.exe

Lists the locks and assignments in a workarea or branch. Displays the following information: lock date, file path, workarea, state, creator/assigner, owner/assignee, comments. To see if a file is locked, see the `istagged` attribute of `iwattrib` (page 72).

Man Page Group:

Workarea Operation.

Usage:

```
iwlistlocks [-h|-v] vpath
```

<i>vpath</i>	Vpath to workarea or branch (see page 13).
-h	Displays usage message.
-v	Displays version.

Example:

```
>iwlistlocks main
```

returns a list of all the locks (including assignments) in all workareas on the main branch. The information is in columns—date, path to the file relative to the workarea, workarea containing the lock, status, user who locked the file, owner of the lock, comments—as follows:

```
[Tue Aug 22 10:31:32 2000] /htdocs/about2.html    wa1 LOCKED    andre andre  
Locked%20by%20launching%20edit  
[Tue Aug 22 10:34:15 2000] /htdocs/banner.gif    wa1 ASSIGNED  andre pat  
add%20support%20section  
[Tue Aug 22 10:34:40 2000] /htdocs/index.html    wa1 DONE      andre pat  
added%20new%20links  
[Tue Aug 22 10:36:29 2000] /htdocs/products.html wa2 LOCKED    chris chris  
<none>
```

iwlistmod.exe

Returns a list of all modified files and directories in the specified area.

Man Page Group:

Workarea Operation.

Usage:

```
iwlistmod [-h|-v] vpath
```

<code>vpath</code>	Vpath to a workarea or directory (see page 13).
<code>-h</code>	Displays usage message.
<code>-v</code>	Displays version.

In the output of `iwlistmod`:

`x+` indicates a deleted file or directory

`f+` indicates a modified or new file

`d` indicates a new directory

Examples:

```
>iwlistmod /default/main/WORKAREA/andre
```

returns:

```
x+      assets/images/iwhead60.gif
x+      assets/images/iwlogo49.gif
f+      html/support/testing_form.html
f+      html/support/index.html
f+      html/support/instructions.html
d       html/newdir/
f+      index.html
```

```
>iwlistmod /default/main/WORKAREA/andre/html/support
```

returns:

f+ testing_form.html
f+ index.html
f+ instructions.html

iwlock.exe

Locks any file or directory in any TeamXpress workarea. For a full description of TeamXpress locking, see the *TeamXpress User's Guide*.

Man Page Group:

Workarea Operation

Usage:

```
iwlock [-f|-s] vpath comment [ownerid] [-f|-s]
```

where *vpath* specifies a vpath to a file.

-h	Displays usage message.
-v	Displays version.
<i>comment</i>	Comment attached to the file.
<i>ownerid</i>	Username of the lock owner.
-f	Force update.
-s	Suppress update.

Example:

```
iwlock main/branch1/WORKAREA/wa1/test.html 'need to use this for a while'
```

Locks the file `test.html` in the workarea `wa1` on the sub-branch `branch1`. The comment attached is “need to use this for a while”. Because no owner is specified, the lock owner is the user who locked it.

Forcing an update will take the version in the staging area to edit, regardless of whether it is the newer version. Suppressing an update will take the version in the workarea, regardless of whether it is newer than the version in the staging area.

iwlockinfo.exe

Provides detailed information on any lock held in TeamXpress, such as who holds the lock and in what workarea.

Man Page Group:

Workarea Operation

Usage:

```
iwlockinfo [-h|-v] vpath
```

where *vpath* specifies a vpath to a file.

-h	Displays usage message.
-v	Displays version.

Example:

```
>iwlockinfo main/branch1/WORKAREA/wal/test.html
```

returns:

```
Name: test.html  
Area: wal  
Assignor: andre  
Assignee: andre  
State: Locked  
Comments: fixing some links
```

showing test.html is locked in workarea wal by Andre, with the comment “fixing some links” attached.

```
iwlockinfo main/branch1/WORKAREA/wal/htdocs/index.html
```

returns:

```
Name: index.html  
Area: wal  
Assignor: andre  
Assignee: pat  
State: Done  
Comments: fix the marketing link.
```

fixed!

showing `index.html` in workarea `wa1` has been assigned by Andre to Pat, and that Pat has marked it Done. Comments attached to the file are “fix the marketing link. fixed!”

iwmenu.exe

Allows you to modify the **Advanced** menu on the SmartContext Editing tab.

Man Page Group:

General Development.

Usage:

```
iwmenu [options] -create [entity] entry parameter+
iwmenu [options] -delete [entity] entry
iwmenu [options] -modify [entity] entry parameter+
iwmenu [options] -query [entity] [entry] -f format
iwmenu [-h|-v]
```

Actions:

-create	Creates a new menu entry for user/group <i>entity</i> .
-delete	Deletes existing menu entry <i>entry</i> for user/group <i>entity</i> .
-modify	Modifies existing menu entry <i>entry</i> for user/group <i>entity</i> .
-query	Displays menu information for user/group <i>entity</i> .
-h	Show this help information (ignores all other options).
-v	Show version information (ignores all other options).

The default action is `-modify` if *parameter* is specified, or `-query` if *parameter* is not specified.

options may be any, all, or none of the following:

-n	Do not inherit from parent group's menu items.
-D	Use the local user database directly. Useful if the TeamXpress server is not up.
-F <i>path</i>	Use the local user database at <i>path</i> .

entity identifies the specific user or group that will be able to use the menu item.

- user *u* Allow user *u* to use the menu item.
- group *g* Allow group *g* to use the menu item.
- session *s* Determine user information from session string *s*.

The default *entity* is -group IWGLOBAL.

entry specifies a particular menu entry to operate on. It can be specified with some or all of the following options:

- menu *m* Operate on menu *m*. (default menu is SmartContext Edit).
- hierarchy *h* Operate on submenu at location *h* in menu hierarchy (required; default location is "", which refers to the top menu).
- position *p* Operate on entry at position *p* in the submenu (required). *p* is a numerical value. Lower values appear first on the submenu. Values do not need to be consecutive.

or the same data can be combined into a single string:

- entry *m/h/p* Operate on entry at menu *m*, submenu *h*, position *p*.

For -query only, *entry* can be omitted, in which case information about all menu entries for *entity* will be displayed.

parameter specifies an individual field of a single menu entry. To create or modify a submenu, *parameter* is:

- submenu *s* Create submenu *s* in specified hierarchy. Useful if you are editing a CGI file and only have access to CGI session strings.

To create a menu entry, *parameter* is some or all of the following. Note that -label, -program, and -roles are required for the -create action:

<code>-cgihook</code>	Run as a CGI program. If <code>-cgihook</code> is not specified, the menu item will be treated as a link.
<code>-confirm</code>	Post a confirmation alert box. The dialog will read “Your configuration has been updated.”
<code>-label <i>l</i></code>	Use <i>l</i> for the menu label (required).
<code>-name <i>n</i></code>	Run the program in a window with name <i>n</i> . If name is <code>no_win</code> , the program will be run with no new window created. If name is <code>new_win</code> , a new window will be opened every time a user selects this entry.
<code>-options <i>o</i></code>	<i>o</i> is a comma-separated <code>name=value</code> list of window options (ignored if <code>-name no_win</code> is specified).
<code>-program <i>p</i></code>	If <code>-cgihook</code> is specified, runs CGI program <i>p</i> when the menu item is selected. Required if <code>-cgihook</code> is specified. The program must be located in <code>iw-home/httpd/iw-bin</code> . Otherwise, it will be treated as a link.
<code>-refresh</code>	Refresh the page after executing the program.
<code>-roles <i>r</i></code>	<i>r</i> is a comma-separated list of roles that can see the item (required).
<code>-vars <i>v</i></code>	<i>v</i> is a comma-separated list of CGI variables to be passed to the CGI program.

For the `-modify` action, flags can be reset and fields from the menu entry’s *entry* can be changed using the following additional *parameters*:

<code>-new_menu <i>m</i></code>	Change the menu under which the menu item appears to <i>m</i> .
<code>-new_hierarchy <i>h</i></code>	Change the submenu under which the menu item appears to <i>h</i> .
<code>-new_position <i>p</i></code>	Change the menu item’s position to <i>p</i> .
<code>-no_confirm</code>	Turn off the <code>-confirm</code> flag.
<code>-no_cgihook</code>	Turn off the <code>-cgihook</code> flag.
<code>-no_refresh</code>	Turn off the <code>-refresh</code> flag.

format is the format of the output for the `-query` action. It may be one of the following values:

<code>text</code>	Human-readable text format (default).
<code>script</code>	Easily-parsed key-value pairs for scripts.
<code>raw</code>	Raw internal format.

Examples:

The following command creates a menu CGI entry using a string to define *entry* parameters. User is Chris. The entry label is `Light Blue`. The entry will reside in the **SmartContextEdit** menu in the file hierarchy **File > Shirts > New Colors**. Entry position is 1999. The entry runs the CGI program `shirt.cgi` as a CGI wrapper. The entry is accessible only to Master users. The page refreshes after the CGI program executes.

```
>iwmenu -create -user chris -entry "SmartContextEdit/File/Shirts/New
Colors/1999" -cgihook -label "Light Blue" -program "shirt.cgi" -roles
"Master" -refresh
```

The following command creates a menu CGI entry using the `-menu`, `-hierarchy`, and `-position` options instead of a single string to define *entry*. User is Chris. The entry label is `Light Blue`. The entry will reside in the **SmartContextEdit** menu in the file hierarchy **File > Shirts > New Colors**. Entry position is 2000. The entry runs the CGI program `shirt.cgi` in a new, resizable window named `new_window` measuring 200 by 300 pixels and containing a menu bar. The entry passes the parameters `size=large` and `collar=yes` to the CGI program. The entry is accessible only to Editors, Administrators, and Master users (note that Administrator is shortened to Admin as shown below). The page does not refresh after the CGI program executes.

```
>iwmenu -create -user Chris -menu SmartContextEdit -hierarchy "File/
Shirts/New Colors" -position "2000" -label "Light Blue" -
program "shirt.cgi" name "new_window" vars "size=large,collar=yes"-
options "width=200,height=300,menubar=yes,resizable" -roles
"Master,Admin,Editor"
```

The following command creates a menu entry that is a link to a web page. User is Chris. The entry label is `Shirt Page`. The entry will reside in the **SmartContextEdit** menu in the file hierarchy **File > Shirts**. Entry position is 2001. The page that is linked is `http://www.acmeshirts.com`. The entry is accessible only to Editors and Authors.

```
>iwmenu -create -user Chris -entry "SmartContextEdit/File/Shirts/2001" -  
label "Shirt Page" -program "http://www.acmeshirts.com" -roles  
"Editor,Author"
```

The following command modifies an existing menu entry for the Buyers group. The entry will now run as a CGI hook program. It resides in the **SmartContextEdit** menu in the file hierarchy **File > Shirts > Purchase Options** in position 2001.

```
>iwmenu -modify -group Buyers -entry SmartContextEdit/File/Shirts/  
Purchase Options/2001 -cghook
```

The following command deletes an existing menu entry for the user Chris. The entry resides in the file hierarchy **File > Shirts** in position 2001.

```
>iwmenu -delete -user Chris -hierarchy "File/Shirts" -position 2001
```

iwmkbr.exe

Creates a new TeamXpress branch. For more information on creating TeamXpress branches, see the *TeamXpress User's Guide*.

Man Page Group:

Branch Operation.

Usage:

```
iwmkbr [flags] parent brname comment rooted [ownerid [groupid]]
```

Creates a new branch in a TeamXpress archive.

<i>parent</i>	Vpath of the parent branch of the new branch (see page 13).
<i>brname</i>	Name of the new branch.
<i>comment</i>	Description of the new branch.
<i>rooted</i>	Name of edition in the parent branch to serve as a basis for the new branch.
<i>ownerid</i>	Username of the owner of the new branch.
<i>groupid</i>	Group name of the group of Administrators that has Administrator-level access to the new branch.

Flags:

-h	Displays usage message.
-v	Displays version.
-l [ock=] s	Create branch with submit locking [default].
-l [ock=] o	Create branch with optional write locking.
-l [ock=] m	Create branch with mandatory write locking.

Example:

```
>iwmkbr //IWSERVER/default/main branch1 'test branch' INITIAL andre
```


creates a sub-branch called `branch1` off of the main branch. The comment attached to this branch is “test branch”. It is based on the initial edition of the main branch, and it is owned by user `andre`. There is no group of Administrators for this branch (only `andre` has Administrator-level privileges for the branch), and the locking model used is the default, submit locking.

iwmkwa.exe

Creates a new TeamXpress workarea on a branch. For more information on creating TeamXpress workareas, see the *TeamXpress User's Guide*.

Man Page Group:

Workarea Operation.

Usage:

```
iwmkwa [-h|-v] branchvpath waname comment base_ed [ownerid [groupid]]
```

<i>-h</i>	Displays usage message.
<i>-v</i>	Displays version.
<i>branchvpath</i>	Vpath of the branch to create the new workarea on (see page 13).
<i>waname</i>	Name of the new workarea.
<i>comment</i>	Comment attached to the workarea.
<i>base_ed</i>	Specifies which edition on the branch to base the workarea on.
<i>ownerid</i>	Specifies who owns the workarea.
<i>groupid</i>	Specifies the group that has access to the workarea.

Example:

```
>iwmkwa main/branch1 wa1 '' INITIAL chrisc
```

This creates a workarea named `wa1` on the sub-branch `branch1`. No comment has been attached to this workarea, and it is based on the initial edition of the branch and owned by `chrisc`. This is a private workarea because no group has access to it.

iwnexted.exe

Returns the name that would be autogenerated by the `iwpublish -a` command for the next edition on a branch.

Man Page Group:

Edition Operation.

Usage:

```
iwnexted [-h|-v] branchvpath
```

<code>-h</code>	Displays usage message.
<code>-v</code>	Displays version.
<code><i>branchvpath</i></code>	Specifies the vpath of the branch (see page 13).

Example:

if the last edition on the main branch is named `ed_0001`,

```
>iwnexted main
```

returns:

```
ed_0002
```

If the name of the last edition ends with a number, `iwnexted` will increment it. If it does not, `iwnexted` will append a number to the name.

iwprv.exe

Allows users to identify whether any file or directory in any workarea is marked private and automatically mark any file as either private or public. For a full description of private and public files and directories, see the *TeamXpress User's Guide*.

Man Page Group:

Workarea Operation.

Usage:

```
iwprv [-h|-v] [-c|-s|-g] list-of-vpath
```

-h	Displays usage message.
-v	Displays version.
-c	Clear the private bit.
-g	Get the private bit (returns 0 if the item is not private, 1 if it is) (default). This option can only accept one vpath.
-s	Set the private bit.
<i>list-of-vpath</i>	List of vpaths of files and directories to mark private (see page 13). Entries must be separated by spaces.

Examples:

```
>iwprv -s main/br1/WORKAREA/wal/gifs main/br1/WORKAREA/wal/mktg main/br1/WORKAREA/wal/index.html main/br1/WORKAREA/wal/main.html
```

makes the directories `gifs` and `mktg` and the files `index.html` and `main.html` private.

```
>iwprv -g main/br1/WORKAREA/wal/gifs
```

returns:

```
1
```

indicating that the item is private.

```
>iwprv -c main/br1/WORKAREA/wa1/gifs main/br1/WORKAREA/wa1/mktg main/br1/  
WORKAREA/wa1/index.html main/br1/WORKAREA/wa1/main.html
```

makes these same files and directories no longer private.

```
>iwprv -g main/branch1/WORKAREA/wa1/test.html
```

returns

0

indicating that the item is not private.

iwpublish.exe

Publishes the staging area on a branch. For more information on publishing editions, see the *TeamXpress User's Guide*.

Man Page Group:

Edition Operation.

Usage:

```
iwpublish -g [-w|-s] [-f] vpath area comment [ownerid]
```

```
iwpublish [-h|-v] [-w|-s] [-f] vpath area edition comment [ownerid]
```

- g Generates an edition name based on the last edition name.
- w Overrides conflicts.
- s Skips conflicts and files locked in another area.
- f Fails publish if there have been no changes since the last publish.
- h Displays usage message.
- v Displays version.

vpath Vpath of the branch whose area you want to publish. Can be a full vpath (*//server/archive/branch*) or a relative path (as shown in the example below).

area The workarea or staging area that you want to publish.

edition Name of the new edition.

comment Comment attached to the edition.

Example:

```
>iwpublish main/branch1 STAGING ed_002 'checkpoint'
```

publishes the staging area on the sub-branch `branch1` as edition `ed_002` with the comment “checkpoint” attached.

iwqueryjobs.exe

Reads a query from `stdin` and prints to `stdout` a list of job IDs that match the query criteria. Query syntax is described below in the “DTD” section.

Man Page Group:

Workflow/Job.

Usage:

```
iwqueryjobs [-h|-v] [-s servername]
```

<code>-h</code>	Displays usage message.
<code>-v</code>	Displays version.
<code>-s <i>servername</i></code>	Specifies the server on which to check for matching query criteria. Server names can be specified by name or IP address.

Example:

The following command searches for matching job IDs on the server `production.example.com`:

```
>iwqueryjobs -s production.example.com
```

DTD:

```
<!ELEMENT wfquery (and|or|not|ownedby|active)>
  <!ELEMENT and ((and|or|not|ownedby|active)+)>
  <!ELEMENT or ((and|or|not|ownedby|active)+)>
  <!ELEMENT not (and|or|not|ownedby|active)>
  <!ELEMENT ownedby EMPTY>
    <!ATTLIST ownedby v CDATA #REQUIRED>
  <!ELEMENT active EMPTY>
```

The `ownedby` element returns the job owned by the specified user. The `active` element returns active jobs.

iwquerytasks.exe

Reads a query from `stdin` and prints to `stdout` a list of task IDs that match the query criteria. Query syntax is described below in the “DTD” section.

Man Page Group:

Workflow/Job.

Usage:

```
iwquerytasks [-h|-v] [-s servername]
```

<code>-h</code>	Displays usage message.
<code>-v</code>	Displays version.
<code>-s <i>servername</i></code>	Specifies the server on which to check for matching query criteria. Server names can be specified by name or IP address.

Example:

The following command searches for matching task IDs on the server `production.example.com`:

```
>iwquerytasks -s production.example.com
```

DTD:

All queries must be in XML format that matches the following DTD:

```
<!ELEMENT taskquery (and|or|not|ownedby|active|wfactive|wfownedby|
    workflow|needsattention|type|sharedby|
    undoableby) >
  <!ELEMENT and ((and|or|not|ownedby|active|wfactive|wfownedby|
    workflow|needsattention|type|sharedby|
    undoableby)+) >
  <!ELEMENT or ((and|or|not|ownedby|active|wfactive|wfownedby|
    workflow|needsattention|type|sharedby|
    undoableby)+) >
  <!ELEMENT not (and|or|not|ownedby|active|wfactive|wfownedby|
    workflow|needsattention|type|sharedby|
```



```

undoableby) >
<!ELEMENT ownedby EMPTY>
  <!ATTLIST ownedby v CDATA #REQUIRED>
<!ELEMENT active EMPTY>
<!ELEMENT wfactive EMPTY>
<!ELEMENT wfownedby EMPTY>
  <!ATTLIST wfownedby v CDATA #REQUIRED>
<!ELEMENT workflow EMPTY>
  <!ATTLIST workflow v CDATA #REQUIRED>
<!ELEMENT needsattention EMPTY>
<!ELEMENT type EMPTY>
  <!ATTLIST type v (usertask, grouptask, externaltask,
                    cgitask, submittask, updatetask,
                    endtask) #REQUIRED>
<!ELEMENT sharedby EMPTY>
  <!ATTLIST sharedby v CDATA #REQUIRED>
<!ELEMENT undoableby EMPTY>
  <!ATTLIST undoableby v CDATA #REQUIRED>

```

Expression and Element Descriptions:

Expression/Element	Description
active	Returns tasks which are active.
and, or, and not	Standard logical operations.
needsattention	Returns submit or update tasks that have stalled because of conflicts and cgi tasks that are ready to be run.
ownedby	Returns tasks owned by the specified user.
sharedby	Returns group tasks that can be grabbed by the specified user.
type	Returns tasks of the specified type.
undoable	Returns user or group tasks that can have their chosen transitions taken back.
wfactive	Returns tasks that belong to active jobs.
wfownedby	Returns tasks that belong to jobs owned by the specified user.
workflow	Returns tasks that belong to the specified job.

iwrename.exe

Renames any file, directory, workarea, edition, or branch (except main) in TeamXpress.

Man Page Group:

General Development.

Usage:

```
iwrename [-h|-v] vpath newname
```

-h	Displays usage message.
-v	Displays version.
vpath	Specifies a vpath to a file, directory, area, or branch (see page 13).
newname	Specifies the new name (not the full vpath).

Examples:

```
>iwrename main/branch1 test
```

renames the sub-branch branch1 to test.

```
>iwrename main/test/WORKAREA/wa2 andre
```

renames the workarea wa1 on test to andre.

```
>iwrename main/test/WORKAREA/andre/htdocs/index.html index2.html
```

renames the file index.html to index2.html in workarea andre.

iwretryjobop.exe

Called when conflicts preventing successful completion of submit or update tasks have been resolved.

Man Page Group:

Workflow/Job.

Usage:

```
iwretryjobop [-h|-v|-o] [-s servername] taskID [comment]
```

<i>-h</i>	Displays usage message.
<i>-v</i>	Displays version.
<i>-o</i>	Force submit or update.
<i>-s servername</i>	Specifies the server on which the job is instantiated. Server names can be specified by name or IP address.
<i>taskID</i>	Specifies the task that will be retried. Task IDs are represented as integers.
<i>comment</i>	An optional comment.

Example:

The following command retries task 7734 on the server `production.example.com`:

```
>iwretrywfoop -s production.example.com 7734
```

iwrevert.exe

Gives command-line users the same **Revert** functionality that is available from the History screen in the TeamXpress GUI. That is, a user can now revert to any version of a file via the command line.

Man Page Group:

Version Management.

Usage:

```
iwrevert [-h|-v] vpath rev
```

<code>-h</code>	Displays usage message.
<code>-v</code>	Displays version.
<code>vpath</code>	Specifies a vpath to a file.
<code>rev</code>	Specifies the version of the file to revert to. <code>rev</code> can have any of these formats: A revision string, as displayed by <code>iwrlog</code> or the History screen in the TeamXpress GUI (e.g. <code>/main/br/52</code>) A number, indicating how many revisions back (e.g., <code>3</code> to revert to three versions ago) The object id of the version of the file (obtainable through <code>iwrlog</code>).

iwrlog.exe

Shows a revision log for the specified TeamXpress file. Provides functionality similar to RCS's `rlog` for TeamXpress files.

Man Page Group:

Version Management.

Usage:

```
iwrlog [-h|-v] [-c] [-m] vpath
```

<code>-h</code>	Displays usage message.
<code>-v</code>	Displays version.
<code>-c</code>	Omits comments/labels from listing.
<code>-m</code>	Displays log information in machine-parsable format. Paths and comments are URL-encoded; fields are separated by tabs.
<code><i>vpath</i></code>	Vpath to a file (see page 16).

Example:

```
>iwrlog -c /default/main/branch1/WORKAREA/andre/htdocs/index.html
```

returns the revision number, date, user, and area containing that version for each version of the file:

```
file /default/main/branch1/WORKAREA/andre/htdocs/index.html (locked)
working revision /main/branch1/6+
```

```
-----
name /htdocs/index.html
revision /main/branch1/6+
last modified Wed Jul 26 11:26:13 2000 by andre in area andre
size 16911
objid 0x0000478f0000479200004816
submit event objid N/A
-----
name /htdocs/index.html
revision /main/branch1/6
last modified Wed Jul 26 11:29:19 2000 by andre in area ed_0002
size 16919
```

```

objid 0x000048180000479d0000480f
submit event objid 0x00002258000000000000480e
-----
name /htdocs/index.html
revision /main/branch1/5
last modified Wed Jul 26 11:26:13 2000 by andre in area ed_0001
size 16911
objid 0x000048060000479d00004805
submit event objid 0x000022580000000000004802
-----
name index.html
revision /main/branch1/4
last modified Wed Jul 26 11:24:35 2000 by andre
size 16904
objid 0x0000480600004804000047fa
submit event objid 0x0000225800000000000047f7
-----
name index.html
revision /main/branch1/3
last modified Tue Jul 25 18:59:51 2000 by andre
size 1206
objid 0x00004806000047f9000047b7
submit event objid 0x0000225800000000000047b4
-----
name index.html
revision /main/branch1/2
last modified Tue Jul 25 18:58:46 2000 by andre
size 4
objid 0x00004806000047b6000047ac
submit event objid 0x0000225800000000000047a9
-----
name index.html
revision /main/branch1/1
last modified Tue Jul 25 18:39:17 2000 by andre
size 0
objid 0x00004806000047ab000047a1
submit event objid 0x00002258000000000000479b
=====
total revisions: 7
=====

```

iwrnbr.exe

Removes the specified branch and all of its contents from TeamXpress. For more information on deleting TeamXpress branches, see the *TeamXpress User's Guide*.

Man Page Group:

Branch Operation.

Usage:

```
iwrnbr [-h|-v] parentvpath brname
```

```
iwrnbr [-h|-v] branchvpath
```

<i>branchvpath</i>	Vpath of the branch to be deleted (see page 13).
<i>parentvpath</i>	Vpath of the parent branch (see page 13).
<i>brname</i>	Name of the branch to be deleted.
-h	Displays usage message.
-v	Displays version.

Example:

```
>iwrnbr main branch1
```

removes the sub-branch `branch1` from the main branch.

iwrmed.exe

Deletes any edition on a branch. For more information on deleting TeamXpress editions, see the *TeamXpress User's Guide*.

Man Page Group:

Edition Operation.

Usage:

```
iwrmed [-h|-v] branchvpath edition
```

<i>-h</i>	Displays usage message.
<i>-v</i>	Displays version.
<i>branchvpath</i>	Vpath of the branch on which the edition is located (see page 13).
<i>edition</i>	Name of the edition to be removed.

Example:

```
>iwrmed main/branch1 ed_001
```

removes the edition ed_001 from the sub-branch branch1.

iwrmdir.exe

Unconditionally removes a job instance from the server. Only use as a last resort.

Man Page Group:

Workflow/Job.

Usage:

```
iwrmdir [-h|-v] [-s servername] jobID
```

<code>-h</code>	Displays usage message.
<code>-v</code>	Displays version.
<code>-s <i>servername</i></code>	Specifies the server on which the job is instantiated. Server names can be specified by name or IP address.
<code><i>jobID</i></code>	Specifies the job that will be removed. Job IDs are represented as integers. Multiple job IDs may be specified.

Example:

The following command removes job 7734 from the server `production.example.com`:

```
>iwrmdir -s production.example.com 7734
```

iwrmtaskfile.exe

Removes a file from a specific task in an instantiated job.

Man Page Group:

Workflow/Job.

Usage:

```
iwrmtaskfile [-h|-v] [-s servername] taskID file
```

<i>-h</i>	Displays usage message.
<i>-v</i>	<i>Displays version.</i>
<i>-s servername</i>	Specifies the server on which the job is instantiated. Server names can be specified by name or IP address.
<i>taskID</i>	Specifies the task from which the file will be removed. Task IDs are represented as integers.
<i>file</i>	Specifies the area-relative path of the file that will be removed from the job task.

Example:

The following command removes the file `/default/main/WORKAREA/eng/content.txt` from job 7734 on the server `production.example.com` (assuming the command is issued from the `eng` directory):

```
>iwrmtaskfile -s production.example.com 7734 content.txt
```

iwormwa.exe

Removes the specified workarea. For more information on deleting workareas, see the *TeamXpress User's Guide*.

Man Page Group:

Workarea Operation.

Usage:

```
iwormwa [-h|-v] branchvpath workarea
```

<i>-h</i>	Displays usage message.
<i>-v</i>	Displays version.
<i>branchvpath</i>	Vpath of the branch on which the workarea is located (see page 13).
<i>workarea</i>	Name of the workarea to be removed.

Example:

```
>iwormwa main/branch1 wa1
```

removes the workarea wa1 from the sub-branch branch1.

iwsubmit.exe

Submits any TeamXpress workarea, file, or directory to the staging area. For a full description of submitting files to the staging area, see the *TeamXpress User's Guide*.

Man Page Group:

Workarea Operation.

Usage:

```
iwsubmit [flags] [-c global_comment] [-i info_comment] {vpath  
file_comment}+
```

```
iwsubmit [flags] [-c global_comment] [-i info_comment] -f file_name
```

-c <i>global_comment</i>	Comment to attach to the submit event.
-i <i>info_comment</i>	Second comment to attach to the submit event.
<i>vpath</i>	One or more vpaths to a file, directory, or workarea (see page 13).
<i>file_comment</i>	Comment to attach to the file, directory, or workarea specified.
-f <i>file_name</i>	File from which to read the vpath(s) and <i>file_comment</i> . If <i>file_name</i> is -, the information will be read from <code>stdin</code> .

Flags:

-h	Displays usage message.
-v	Displays version.
-s	Skips conflicts and locked files.
-w	Overrides conflicts.
-u	Unlocks locked files.
-r	Reports on submitted files.

Example:

```
>iwsubmit -w -u -c 'updated support info' -i 'keyword' main/WORKAREA/  
andre/htdocs/index.html 'added imagemap'
```

submits the file `index.html` to the staging area, with the comment “added imagemap” attached to the file. The comments for the submit operation are “updated support info” and “keyword”. If `index.html` is locked, this operation will unlock it. If the version in the staging area is newer, this operation will overwrite it.

iwtask.exe

Assigns a shared task to a specified user.

Man Page Group:

Workflow/Job.

Usage:

```
iwtask [-h|-v] [-s servername] taskID username
```

<i>-h</i>	Displays usage message.
<i>-v</i>	Displays version.
<i>-s servername</i>	Specifies the server on which the job is instantiated. Server names can be specified by name or IP address.
<i>taskID</i>	Specifies the task that will be assigned. Task IDs are represented as integers.
<i>username</i>	Specifies the user who will receive the task assignment.

Example:

The following command assigns task 7734 on the server `production.example.com` to user Andre:

```
>iwtask -s production.example.com 7734 Andre
```

iwtaskselect.exe

Marks a user task finished and chooses which <successorset/> element to signal.

Man Page Group:

Workflow/Job.

Usage:

```
iwtaskselect [-h|-v] [-s servername] taskID choice [comment]
```

<code>-h</code>	Displays usage message.
<code>-v</code>	Displays version.
<code>-s <i>servername</i></code>	Specifies the server on which the job is instantiated. Server names can be specified by name or IP address.
<code><i>taskID</i></code>	Specifies the task that will be marked finished. Task IDs are represented as integers.
<code><i>choice</i></code>	Specifies the index (starting from 0) of the <successorset/> element in the <successors/> section of the task's configuration.
<code><i>comment</i></code>	Optional comment.

Example:

The following command marks task 7734 done on the server `production.example.com` and signals the third <successorset/> element in that task's <successors/> section:

```
>iwtaskselect -s production.example.com 7734 2
```


iwundochoice.exe

When a user chooses a transition from a usertask or grouptask, that choice can be taken back. `iwundochoice` reverses one such task. The `iwundochoice` CLT cannot reverse a transition for a task with multiple successors.

Man Page Group:

Workflow/Job.

Usage:

```
iwundochoice [-h|-v] [-s servername] taskID
```

<code>-h</code>	Displays usage message.
<code>-v</code>	Displays version.
<code>-s <i>servername</i></code>	Specifies the server on which the job is instantiated. Server names can be specified by name or IP address.
<code><i>taskID</i></code>	Specifies the task that was active prior to the transition. Task IDs are represented as integers.

Example:

The following command reverses the transition from task 7734 on the server `production.example.com`:

```
>iwundochoice -s production.example.com 7734
```

iwunlock.exe

Unlocks any file or directory in any TeamXpress workarea. See the *TeamXpress User's Guide* for a full description of unlocking files.

Man Page Group:

Workarea Operation.

Usage:

```
iwunlock [-h|-v] vpath
```

-h	Displays usage message.
-v	Displays version.
<i>vpath</i>	Specifies a vpath to a file.

Example:

```
iwunlock main/branch1/WORKAREA/wa1/index.html
```

Unlocks the file `index.html` in the workarea `wa1` on the sub-branch `branch1`. Only one file may be unlocked at a time with this command.

iwupdate.exe

Updates any TeamXpress file, directory, or workarea with the latest version of any file system element. This command acts like the **Copy To** or (if *vpath_from* specifies the staging area on that branch) the **Get Latest** command in the GUI. See the *TeamXpress User's Guide* for more information about these commands.

Man Page Group:

Workarea Operation.

Usage:

```
iwupdate [-h|-v] [-w] [-r] [-o] {src_vpath}+ dst_vpath
```

```
iwupdate [-h|-v] [-w] [-r] [-o] [-f vpath_file] dst_vpath
```

-h	Displays usage message.
-v	Displays version.
-w	Override conflicts.
-r	Report on updated files.
-f	Read source vpaths from the file <i>vpath_file</i> (if <i>vpath_file</i> is -, STDIN will be read).
-o	Directory update conflicts return success.
<i>src_vpath</i>	Specifies a path to a file, directory, or area.
<i>dst_vpath</i>	Specifies a path to a workarea (<i>dst_vpath</i> may specify a path to a file or directory in a workarea if the area-relative portions of <i>dst_vpath</i> and <i>src_vpath</i> are the same).

Example:

```
>iwupdate -w main/branch1/WORKAREA/wa1 main/branch1/WORKAREA/wa2
```

copies all the files that are different in workarea wa1 to workarea wa2, overriding any conflicts.

iwvpath.exe

Prints all or parts of the version path of the specified object.

Man Page Group:

General Development.

Usage:

```
iwvpath [-h|-v] [-p|-b|-d|[-s] [-a]] objectvpath
```

```
iwvpath [-p|-b|-d|[-s] [-a]] -o objid
```

<i>objectvpath</i>	Vpath of any TeamXpress object (see page 13).
-o <i>objid</i>	Objid of any TeamXpress object (see page 17).
-a	Prints the area vpath (rooted at the archive) (see page 13).
-b	Prints the branch relative vpath only (see page 13).
-d	Prints the directory path only (see page 13).
-p	Prints the components of the version path in separate lines: <server> <archive> <branch> <area-type> <area-name> <directory-path> where <area-type> is WORKAREA, STAGING, or EDITION
-h	Displays usage message.
-v	Displays version.
-s	Includes the server as part of the vpath.

Examples:

```
>iwvpath -d main/WORKAREA/andre/htdocs/index.html
```

returns:

```
/htdocs/index.html
```

```
>iwvpath -p -c fse -o 0x0000007b0000007d00000093 //IWSERVER/default
```

returns:

```
chocolate  
default  
/main  
WORKAREA  
andre  
/htdocs/index.html
```

```
>iwvpath -b -c fse -o 0x0000007b0000007d00000093 //IWSERVER/default
```

returns:

```
/main
```

```
>iwvpath -d -c fse -o 0x0000007b0000007d00000093 //IWSERVER/default
```

returns:

```
/htdocs/index.html
```


Chapter 4

Command Triggers

This chapter explains how to:

- Configure command triggers to run as services.
- Start and stop command triggers that have been set up to run as services.

This chapter also contains man pages for all supported command triggers. All man pages in this chapter are presented as one group, arranged alphabetically:

Trigger	Description	See...
<code>iwat.ip1</code>	Registers an event handler for events that do not have a specific command trigger.	page 158
<code>iwatasgn.ip1</code>	Triggers a custom script each time a file is assigned, approved, rejected, unassigned, or marked done.	page 161
<code>iwatcreate.ip1</code>	Triggers a custom script each time a file or a directory is created.	page 162
<code>iwatlock.ip1</code>	Triggers a custom script each time a file or a directory is locked.	page 163
<code>iwatmkbr.ip1</code>	Triggers a custom script each time a new branch is created.	page 164
<code>iwatmkwa.ip1</code>	Triggers a custom script each time a new workarea is created.	page 165
<code>iwatpub.ip1</code>	Triggers a custom script each time a new edition is published.	page 166
<code>iwatrmbr.ip1</code>	Triggers a custom script each time a branch is deleted.	page 167
<code>iwatrmmed.ip1</code>	Triggers a custom script each time an edition is deleted.	page 168
<code>iwatrmwa.ip1</code>	Triggers a custom script each time a workarea is deleted.	page 169
<code>iwatserver.ip1</code>	Triggers a custom script each time a StartUp, ShutDown, Freeze, Thaw, or DiskLow event occurs.	page 170
<code>iwatsub.ip1</code>	Triggers a custom script each time a file or a directory is submitted.	page 171
<code>iwatunlock.ip1</code>	Triggers a custom script each time a file or a directory is unlocked.	page 172
<code>iwatupdate.ip1</code>	Triggers a custom script each time an update event occurs (Copy or Get Latest).	page 173

<code>iwlsat.ip1</code>	Lists registered <code>iwat</code> programs.	page 174
<code>iwrmat.ip1</code>	Removes registered <code>iwat</code> programs.	page 175

Starting Command Triggers

TeamXpress's command triggers run custom scripts whenever certain events occur in the TeamXpress system. For example, the `iwatasgn` trigger can be configured to execute an email notification script when a file is assigned. Each trigger that is invoked tails the event log and responds to different events. See Appendix B, "Sample Command Trigger Scripts," for script examples.

Because they are non-terminating, command triggers are usually included in a script that starts and stops the process and its tail.

By default, the `iwat` command triggers invoke user scripts synchronously. That is, they wait for the scripts to return before handling the next event. You can use the syntax shown in the following example to configure an `iwat` trigger to run a script asynchronously:

```
iwatserver "progrname progargs ... &"
```

Note that the ampersand is inside the quotation marks that encase the command. This prevents the shell from running `iwatserver` itself in the background. As coded above, *progrname* runs in the background when executed. The `iwatserver` command trigger is just used as an example here. You can use this syntax with any `iwat` command trigger.

To configure the automatic triggering of custom scripts, you will need to use `srvany.exe` and `instsrv.exe` (available as part of the Windows NT Resource Kit).

Notes:

- In the following steps, you must use MS-DOS naming conventions if the file or directory names that you specify contain spaces or more than eight characters. For example, instead of `C:\iw-home\Program Files\Interwoven`, you would specify `C:\iw-home\Progra~1\Interw~1`. You can use the `dir /x` command to display the long and short versions of the file names in the current directory.
- The String data type in `regedit` is the same as `REG_SZ` in `regedt32`.

- See the Windows NT Resource Kit document `srvany.doc` for more information about `srvany`.

Perform the following steps to set up a command trigger script as a service:

1. Create a service:

```
>instsrv.exe <servicename> "C:\iw-home\tools\srvany.exe"
```

For example:

```
>instsrv.exe iwatsub "C:\iw-home\tools\srvany.exe"
```

2. Start `regedt32.exe` (note that `regedt` does not contain the letter `i`). Go to `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\servicename`. Using the example from Item 1, `servicename` would be `iwatsub`.
3. Add a key called `Parameters`.
4. Add an `Application` value to the `Parameters` key:

```
Application: REG_SZ: script
```

where *script* specifies the full path to your executable, including extension. Continuing with the example from Item 1, this line would read as follows for a perl script named `iwperl.exe` residing in the `iw-perl/bin` directory:

```
Application: REG_SZ: C:\iw-home\iw-perl\bin\iwperl.exe
```

5. Add an `AppParameters` value to the `Parameters` key:

```
AppParameters: REG_SZ: C:\iw-home\bin\triggername.ip1 script  
[parameters]
```

where *triggername* is the command trigger you want to run when the submit event occurs and *parameters* are the parameters (if any) the script requires that can be known at service startup time. For example (assuming no parameters are necessary):

```
AppParameters: REG_SZ: C:\iw-home\bin\iwatsub.ip1 C:\iw-home\  
iw-perl\bin\iwperl.exe
```

6. Add an `AppDirectory` value to the `Parameters` key:

```
AppDirectory: REG_SZ: exec_dir
```

where *exec_dir* is the directory where you want the application to execute. For example:

```
C:\iw-home\iw-perl\bin
```

To remove a service:

```
>instsrv.exe servicename REMOVE
```

To start a service:

```
>net start servicename
```

To stop the service:

```
>net stop servicename
```

Environment Variables

When using command triggers, environment variables are set before the program *programe* is run. The events and their values are described under the appropriate command trigger.

The IW_EVENT environment variable has one of the following values, depending on the command trigger being used. All events, including those without an entry in the **Command Trigger** column, can be monitored using *iwat*.

IW_EVENT Environment Variable	Description	Command Trigger
ApproveFile	A file has been approved	iwatasgn
AssignFile	A file has been assigned	iwatasgn
CreateBranch	A branch has been created	iwatmkbr
CreateFSE	An FSE (file/directory/symlink) has been created	iwatcreate
CreateWorkarea	A workarea has been created	iwatmkwa
DeleteEA	An extended attribute (EA) has been deleted from a file	
DestroyBranch	A branch has been deleted	iwatrmbr
DestroyEdition	An edition has been deleted	iwatrmcd

IW_EVENT Environment Variable	Description	Command Trigger
DestroyFSE	An FSE has been deleted	
DestroyWorkarea	A workarea has been deleted	iwatrmwa
DiskFail	An I/O error has occurred on the disk(s) containing the backing store.	iwatserver
DiskLow	The disk(s) containing the backing store are running low on space.	iwatserver
Freeze	The server has been frozen (iwfreeze)	iwatserver
Lock	A file or directory has been locked	iwatlock
MarkDoneFile	A file has been marked as “done”	iwatasgn
ModifyFSE	An FSE has been modified	
PublishStagingArea	A new edition has been published from the staging area	iwatpub
ReassignFile	A file has been reassigned	iwatasgn
RejectFile	A file has been reject	iwatasgn
RenameFSE	An FSE has been renamed	
ResetConfig	TeamXpress has been asked to reload its configuration files (iwreset)	
SetEA	An EA has been added to/modified on a file	
ShutDown	The server has been shut down	iwatserver
Startup	The server has been started	iwatserver
Submit	A file or directory has been submitted	iwatsub
SyncDestroy	A file with EAs has been deleted	
SyncModify	A file with EAs has been modified	
SyncRevert	A file with EAs has reverted to an earlier version	
Thaw	The server has been unfrozen (thawed)	iwatserver
UnassignFile	A file has been unassigned	iwatasgn
Unlock	A file or directory has been unlocked	iwatunlock



IW_EVENT Environment Variable	Description	Command Trigger
Update	A file or directory has been updated	iwatupdate

iwat.ipl

This program is used to register an event handler for events that do not have a specific command trigger program.

Usage:

```
iwat [-h|-v] [-s servername] eventregex_program
```

<code>-h</code>	Displays usage message.
<code>-v</code>	Displays version.
<code>-s <i>servername</i></code>	Specifies the TeamXpress server to which to connect. It can be specified by name or IP address. The default value is the current TeamXpress server.
<code><i>eventregex_program</i></code>	A regular expression to match complete iwevents.log lines. When an event matches this regular expression, the program is run.

Example:

```
>iwat CreateWorkarea /usr/ucb/echo
```

iwatasgn.ipl

iwatasgn is a non-terminating script which executes the custom script *progrname* once each time a file is assigned, approved, rejected, unassigned, or marked done.

Usage:

```
iwatasgn [-h|-v] [-p] progrname progargs...
```

-h	Displays usage message.
-v	Displays version.
-p	Prints process ID before starting.
<i>progrname</i>	Custom script to be executed.
<i>progargs</i>	Arguments to <i>progrname</i> .

iwatasgn triggers on the following events:

AssignFile	ReassignFile
ApproveFile	RejectFile
MarkDoneFile	UnassignFile

The following environment variables are set (to the described values) before the program *progrname* is run:

IW_EVENT	one of the above list of events
IW_TIMESTAMP	the time when the event occurred
IW_EVENTUSER	username of person who performed the operation
IW_EVENTROLE	role IW_EVENTUSER was in during the operation
IW_AREA	vpath of workarea (rooted at the archive)
IW_FILEPATH	file path of the file affected (relative to its area)
IW_OBJID	objid of the file which was affected file
IW_ASSIGNEE	person who is/was assigned the file
IW_COMMENTS	comments associated with the event
IW_EDITOR	workarea owner

iwatcreate.ipl

`iwatcreate` is a non-terminating script which executes the custom script *progrname* once each time a file or a directory has been created.

Usage:

```
iwatcreate [-h|-v] [-p] progrname progargs...
```

<code>-h</code>	Displays usage message.
<code>-v</code>	Displays version.
<code>-p</code>	Prints process ID before starting.
<i>progrname</i>	Custom script to be executed.
<i>progargs</i>	Arguments to <i>progrname</i> .

The following environment variables are set (to the described values) before the program *progrname* is run:

<code>IW_EVENT</code>	“CreateFSE”
<code>IW_TIMESTAMP</code>	the time when the event occurred
<code>IW_EVENTUSER</code>	username of person who performed the operation
<code>IW_EVENTROLE</code>	role <code>IW_EVENTUSER</code> was in during the operation
<code>IW_FILE</code>	vpath of file (rooted at the archive)

iwatlock.ipl

`iwatlock` is a non-terminating script which executes the custom script *progrname* once each time a file or a directory is locked.

Usage:

```
iwatlock [-h|-v] [-p] progrname progargs...
```

<code>-h</code>	Displays usage message.
<code>-v</code>	Displays version.
<code>-p</code>	Prints process ID before starting.
<i>progrname</i>	Custom script to be executed.
<i>progargs</i>	Arguments to <i>progrname</i> .

The following environment variables are set (to the described values) before the program *progrname* is run:

<code>IW_EVENT</code>	“Lock”
<code>IW_TIMESTAMP</code>	the time when the event occurred
<code>IW_EVENTUSER</code>	username of person who performed the operation
<code>IW_EVENTROLE</code>	role <code>IW_EVENTUSER</code> was in during the operation
<code>IW_FILE</code>	vpath of file (rooted at the archive)
<code>IW_EDITOR</code>	workarea owner

iwatmkbr . ipl

iwatmkbr is a non-terminating script which executes the custom script *progrname* once each time a new branch is created.

Usage:

iwatmkbr [-h|-v] [-p] *progrname* *progargs*...

-h	Displays usage message.
-v	Displays version.
-p	Prints process ID before starting.
<i>progrname</i>	Custom script to be executed.
<i>progargs</i>	Arguments to <i>progrname</i> .

The following environment variables are set (to the described values) before the program *progrname* is run:

IW_EVENT	“CreateBranch”
IW_TIMESTAMP	the time when the event occurred
IW_EVENTUSER	username of person who performed the operation
IW_EVENTROLE	role IW_EVENTUSER was in during the operation
IW_BRANCH	vpath of branch (rooted at the archive)

iwatmkwa.ipl

iwatmkwa is a non-terminating script which executes the custom script *progrname* once each time a new workarea is created.

Usage:

```
iwatmkwa [-h|-v] [-p] progrname progargs...
```

<i>-h</i>	Displays usage message.
<i>-v</i>	Displays version.
<i>-p</i>	Prints process ID before starting.
<i>progrname</i>	Custom script to be executed.
<i>progargs</i>	Arguments to <i>progrname</i> .

The following environment variables are set (to the described values) before the program *progrname* is run:

<i>IW_EVENT</i>	“CreateWorkarea”
<i>IW_TIMESTAMP</i>	the time when the event occurred
<i>IW_EVENTUSER</i>	username of person who performed the operation
<i>IW_EVENTROLE</i>	role <i>IW_EVENTUSER</i> was in during the operation
<i>IW_WORKAREA</i>	vpath of workarea (rooted at the archive)

iwatpub.ipl

`iwatpub` is a non-terminating script which executes the custom script *progrname* once each time a new edition is published.

Usage:

```
iwatpub [-h|-v] [-p] progrname progargs...
```

<code>-h</code>	Displays usage message.
<code>-v</code>	Displays version.
<code>-p</code>	Prints process ID before starting.
<i>progrname</i>	Custom script to be executed.
<i>progargs</i>	Arguments to <i>progrname</i> .

The following environment variables are set (to the described values) before the program *progrname* is run:

<code>IW_EVENT</code>	“PublishStagingArea”
<code>IW_TIMESTAMP</code>	the time when the event occurred
<code>IW_EVENTUSER</code>	username of person who performed the operation
<code>IW_EVENTROLE</code>	role <code>IW_EVENTUSER</code> was in during the operation
<code>IW_EDITION</code>	vpath of edition (rooted at the archive)
<code>IW_STAGINGAREA</code>	version path of the staging area published

iwatrmbr.ipl

`iwatrmbr` is a non-terminating script which executes the custom script *progname* once each time a branch is deleted.

Triggers an event (custom script) upon removal of a branch.

Usage:

```
iwatrmbr [-h|-v] [-p] progname progargs...
```

<code>-h</code>	Displays usage message.
<code>-v</code>	Displays version.
<code>-p</code>	Prints process ID before starting.
<i>progname</i>	Custom script to be executed.
<i>progargs</i>	Arguments to <i>progname</i> .

The following environment variables are set (to the described values) before the program *progname* is run:

<code>IW_EVENT</code>	“DestroyBranch”
<code>IW_TIMESTAMP</code>	the time when the event occurred
<code>IW_EVENTUSER</code>	username of person who performed the operation
<code>IW_EVENTROLE</code>	role <code>IW_EVENTUSER</code> was in during the operation
<code>IW_BRANCH</code>	vpath of branch (rooted at the archive)

iwatrmcd.ipl

iwatrmcd is a non-terminating script which executes the custom script *progrname* once each time an edition is deleted.

Usage:

iwatrmcd [-h|-v] [-p] *progrname* *progargs*...

-h	Displays usage message.
-v	Displays version.
-p	Prints process ID before starting.
<i>progrname</i>	Custom script to be executed.
<i>progargs</i>	Arguments to <i>progrname</i> .

The following environment variables are set (to the described values) before the program *progrname* is run:

IW_EVENT	“DestroyEdition”
IW_TIMESTAMP	the time when the event occurred
IW_EVENTUSER	username of person who performed the operation
IW_EVENTROLE	role IW_EVENTUSER was in during the operation
IW_EDITION	vpath of edition (rooted at the archive)

iwatrmwa.ipl

`iwatrmwa` is a non-terminating script which executes the custom script *progrname* once each time a workarea is deleted.

Usage:

```
iwatrmwa [-h|-v] [-p] progrname progargs...
```

<code>-h</code>	Displays usage message.
<code>-v</code>	Displays version.
<code>-p</code>	Prints process ID before starting.
<i>progrname</i>	Custom script to be executed.
<i>progargs</i>	Arguments to <i>progrname</i> .

The following environment variables are set (to the described values) before the program *progrname* is run:

<code>IW_EVENT</code>	“DestroyWorkarea”
<code>IW_TIMESTAMP</code>	the time when the event occurred
<code>IW_EVENTUSER</code>	username of person who performed the operation
<code>IW_EVENTROLE</code>	role <code>IW_EVENTUSER</code> was in during the operation
<code>IW_WORKAREA</code>	vpath of workarea (rooted at the archive)

iwatserver.ipl

The `iwatserver` CLT is a non-terminating script that executes *progname* once each time after one of the following events has occurred: `StartUp`, `ShutDown`, `Freeze`, `Thaw`, `DiskLow`, or `DiskFail`.

Usage:

```
iwatserver [-h|-p|-v] progname progargs...
```

<code>-h</code>	Prints usage message.
<code>-p</code>	Prints the process ID before starting.
<code>-v</code>	Prints the version string.
<code>progargs...</code>	Arguments to <i>progname</i> .

The following environment variables are set (to the described values) before *progname* is run:

<code>IW_EVENT</code>	One of the events— <code>StartUp</code> , <code>ShutDown</code> , <code>Freeze</code> , <code>Thaw</code> , <code>DiskLow</code> , or <code>DiskFail</code> .
<code>IW_TIMESTAMP</code>	The time when the event occurred.
<code>IW_EVENTUSER</code>	Username of person who performed the operation.
<code>IW_EVENTROLE</code>	Role that <code>IW_EVENTUSER</code> was in during the operation.
<code>IW_COMMENTS</code>	Comments associated with the event.

iwatsub.ipl

`iwatsub` is a non-terminating script which executes the custom script *programe* once each time a file or a directory is submitted.

Triggers an event (custom script) upon submission of any file system element.

Usage:

```
iwatsub [-h|-v] [-p] programe progargs...
```

<code>-h</code>	Displays usage message.
<code>-v</code>	Displays version.
<code>-p</code>	Prints process ID before starting.
<i>programe</i>	Custom script to be executed.
<i>progargs</i>	Arguments to <i>programe</i> .

The following environment variables are set (to the described values) before the program *programe* is run:

<code>IW_EVENT</code>	“Submit”
<code>IW_TIMESTAMP</code>	the time when the event occurred
<code>IW_EVENTLINE</code>	Copy of the line from the event log that originally triggered the event
<code>IW_EVENTUSER</code>	username of person who performed the operation
<code>IW_EVENTROLE</code>	role <code>IW_EVENTUSER</code> was in during the operation
<code>IW_WORKAREA</code>	vpath of workarea (rooted at the archive) (same as <code>IW_AREA</code> and <code>IW_VPATH</code>)
<code>IW_SUBMITID</code>	objid associated with the submit event
<code>IW_EDITOR</code>	workarea owner

iwatunlock.ipl

`iwatunlock` is a non-terminating script which executes the custom script *progrname* once each time a file or a directory is unlocked.

Usage:

```
iwatunlock [-h|-v] [-p] progrname progargs...
```

<code>-h</code>	Displays usage message.
<code>-v</code>	Displays version.
<code>-p</code>	Prints process ID before starting.
<i>progrname</i>	Custom script to be executed.
<i>progargs</i>	Arguments to <i>progrname</i> .

The following environment variables are set (to the described values) before the program *progrname* is run:

<code>IW_EVENT</code>	“Unlock”
<code>IW_TIMESTAMP</code>	the time when the event occurred
<code>IW_EVENTUSER</code>	username of person who performed the operation
<code>IW_EVENTROLE</code>	role <code>IW_EVENTUSER</code> was in during the operation
<code>IW_FILE</code>	vpath of file (rooted at the archive)
<code>IW_EDITOR</code>	workarea owner

iwatupdate.ipl

iwatupdate is a non-terminating script which executes the custom script *programe* once each time a file or a directory is updated.

Usage:

```
iwatupdate [-h|-v] [-p] programe progargs...
```

-h	Displays usage message.
-v	Displays version.
-p	Prints process ID before starting.
<i>programe</i>	Custom script to be executed.
<i>progargs</i>	Arguments to <i>programe</i> .

The following environment variables are set (to the described values) before the program *programe* is run:

IW_EVENT	“Update”
IW_TIMESTAMP	the time when the event occurred
IW_EVENTUSER	username of person who performed the operation
IW_EVENTROLE	role IW_EVENTUSER was in during the operation
IW_WORKAREA	version path of workarea (rooted at the archive)
IW_UPDATEID	objid associated with the update event
IW_EDITOR	workarea owner

iwlsat.ipl

`iwlsat` lists registered `iwat` programs, their IDs, and their event regular expressions.

Usage:

```
iwlsat [-h|-v] [-s servername]
```

<code>-h</code>	Displays usage message.
<code>-v</code>	Displays version.
<code>-s <i>servername</i></code>	Specifies the TeamXpress server to which to connect. It can be specified by name or IP address. The default value is the current TeamXpress server.

Example:

```
>iwlsat
```

displays the following output:

```
ID: 1234 "CreateWorkarea" /usr/ucb/echo
```

iwrmat.ipl

iwrmat unregisters programs that have been added using the `iwat` command.

Usage:

```
iwrmat [-h|-v] [-s servername] ID
```

<code>-h</code>	Displays usage message.
<code>-v</code>	Displays version.
<code>-s <i>servername</i></code>	Specifies the TeamXpress server to which to connect. It can be specified by name or IP address. The default value is the current TeamXpress server.
<code><i>ID</i></code>	The ID number of the <code>iwat</code> program to unregister.

Example:

```
>iwrmat 1234
```

Appendix A

Master List

The following is an alphabetized master list of all the CLTs and command triggers described in this manual:

CLT/Trigger	Description	See...
<code>iwabort.exe</code>	Provides a method for terminating a long-running server operation.	page 21
<code>iwaddtaskfile.exe</code>	Adds a file to a job task that is part of a job instance already created on the TeamXpress server.	page 63
<code>iwat.ipl</code>	Registers an event handler for events that do not have a specific command trigger.	page 158
<code>iwatasgn.ipl</code>	Triggers a custom script each time a file is assigned, approved, rejected, unassigned, or marked done.	page 161
<code>iwatcreate.ipl</code>	Triggers a custom script each time a file or a directory is created.	page 162
<code>iwatlock.ipl</code>	Triggers a custom script each time a file or a directory is locked.	page 163
<code>iwatmkbr.ipl</code>	Triggers a custom script each time a new branch is created.	page 164
<code>iwatmkwa.ipl</code>	Triggers a custom script each time a new workarea is created.	page 165
<code>iwatpub.ipl</code>	Triggers a custom script each time a new edition is published.	page 166
<code>iwatrmbr.ipl</code>	Triggers a custom script each time a branch is deleted.	page 167
<code>iwatrmmed.ipl</code>	Triggers a custom script each time an edition is deleted.	page 168
<code>iwatrmwa.ipl</code>	Triggers a custom script each time a workarea is deleted.	page 169
<code>iwatserver.ipl</code>	Triggers a custom script each time a StartUp, ShutDown, Freeze, Thaw, or DiskLow event occurs.	page 170



iwatsub.ipl	Triggers a custom script each time a file or a directory is submitted.	page 171
iwattrib.exe	Returns metadata information on any object in the TeamXpress server, including all branches, workareas, editions, staging areas, files, directories, and symlinks.	page 64
iwatunlock.ipl	Triggers a custom script each time a file or a directory is unlocked.	page 172
iwatupdate.ipl	Triggers a custom script each time an update event occurs (Copy or Get Latest).	page 173
iwcallback.exe	Tells the TeamXpress server that the program corresponding to an external or cgi task is finished. Passes a return code to the server.	page 73
iwcat.exe	Displays any version of a (text) file in TeamXpress.	page 74
iwchgrp.exe	Changes the group of a TeamXpress workarea or branch.	
iwckrole.exe	Checks whether or not a user can log in with a particular role.	page 75
iwcmp.exe	Compares any two TeamXpress areas and returns a list of differences.	page 76
iwcompress.exe	Compresses and uncompresses editions.	page 78
iwconfig.exe	Reads or writes to TeamXpress's main configuration file.	page 23
iwdecode.exe	Decodes HTML-encoded %xx lines to ASCII lines.	page 80
iwdeploy.exe	Deploys website content to the production server.	page 83
iwencode.exe	Encodes ASCII lines to HTML-encoded %xx lines.	page 89
iwevents.exe	Returns a history of submissions or updates of a workarea.	page 90
iwextattr.exe	Sets extended attributes on a file.	page 94
iwfreeze.exe	Freezes and unfreezes all system writes.	page 27
iwfsck.exe	Diagnoses backing store problems.	page 29
iwfsconvert.exe	Merges multiple metadata files into a single file.	page 32
iwfsfix.exe	Fixes problems found by iwfsck.	page 34
iwfsshrink.exe	Finds and removes duplicate data in the backing store.	page 36

iwfsstripe.exe	Converts existing backing store to a multiple file systems.	page 38
iwgetelog.exe	Returns location of TeamXpress events log.	page 40
iwgethome.exe	Returns location of TeamXpress program files.	page 41
iwgetlocation.exe	Returns the locations of various TeamXpress log and configuration files.	page 42
iwgetmount.exe	Returns location of TeamXpress mount point	page 45
iwgetstore.exe	Returns location of TeamXpress backing store.	page 46
iwgettrace.exe	Returns location of TeamXpress trace logs.	page 47
iwgetwfobj.exe	Prints the state of any part of the server workflow subsystem in XML.	page 96
iwinvokejob.exe	Starts a job whose instance has already been created on the TeamXpress server.	page 103
iwjobc.exe	Creates a job instance (based on a job specification file) on the TeamXpress server.	page 104
iwjobvariable.exe	Manipulates workflow variables.	page 105
iwlasted.exe	Returns the name of the last published edition on a branch.	page 106
iwlist.exe	Returns a list of all areas contained by a parent areas. For example, iwlist returns all editions, workareas, and branches contained on a parent branch.	page 107
iwlistlocks.exe	Lists the locks and assignments in a workarea or branch.	page 110
iwlistmod.exe	Lists all modified files and directories in a specified area.	page 111
iwlock.exe	Locks any file or directory in a specified area.	page 113
iwlockinfo.exe	Provides detailed information on any lock.	page 114
iwlstat.ip1	Lists registered iwat programs.	page 174
iwmenu.exe	Modifies the Advanced menu on the SmartContext Editing tab.	page 116
iwmkbr.exe	Creates a new TeamXpress branch.	page 121
iwmkwa.exe	Creates a new TeamXpress workarea.	page 123

<code>iwnexted.exe</code>	Returns the name that would be autogenerated for the next edition on a branch.	page 124
<code>iwproxy.exe</code>	Invokes a proxy server.	page 48
<code>iwprv.exe</code>	Allows users to identify whether any file or directory in any workarea is marked private and automatically mark any file as either private or public.	page 125
<code>iwpublish.exe</code>	Publishes the staging area.	page 127
<code>iwqueryjobs.exe</code>	Prints a list of overall job information based on a query from stdin.	page 129
<code>iwquerytasks.exe</code>	Prints a list of job task information based on a query from stdin.	page 130
<code>iwrecentusers.exe</code>	Displays a list of everyone who has used TeamXpress since the last time the TeamXpress server was started, and a timestamp of each user's most recent TeamXpress operation.	page 49
<code>iwrename.exe</code>	Renames a file, directory, workarea, edition, or branch.	page 134
<code>iwreset.exe</code>	Rereads TeamXpress configuration files.	page 50
<code>iwretryjobop.exe</code>	Retries submission or task update after resolution of conflicts that originally prevented those operations.	page 135
<code>iwrevert.exe</code>	Revert to a previous version of a file.	page 136
<code>iwrlog.exe</code>	Shows a revision log for a TeamXpress file.	page 137
<code>iwrmat.ipl</code>	Removes registered <code>iwat</code> programs.	page 175
<code>iwrnbr.exe</code>	Deletes a TeamXpress branch and all contained areas (workareas, staging areas, and editions).	page 139
<code>iwrmed.exe</code>	Deletes an edition.	page 140
<code>iwrnmjob.exe</code>	Unconditionally removes an entire job instance from the TeamXpress server.	page 141
<code>iwrmtaskfile.exe</code>	Removes a file from a job task.	page 142
<code>iwrnwa.exe</code>	Deletes a TeamXpress workarea.	page 143
<code>iwsi.ipl</code>	Collects system state data.	page 52
<code>iwstat.exe</code>	Returns current system activity.	page 54
<code>iwsubmit.exe</code>	Submits any TeamXpress workarea, file, or directory to the staging area.	page 144

iwtasktask.exe	Assigns a shared task to a single user.	page 146
iwtaskselect.exe	Marks a user's job task finished and selects a successor task.	page 147
iwteamsite.exe	Installs or uninstalls the iwtock watchdog daemon.	page 54
iwtestcfg.exe	Returns the operation that will be performed upon a file at submission time.	page 56
iwundochoice.exe	Reverses a user-chosen transition from a user task or group task.	page 148
iwunlock.exe	Unlocks any file or directory.	page 149
iwupdate.exe	Updates a TeamXpress workarea with the version of a file or directory in the staging area.	page 150
iwversion.exe	Returns current TeamXpress release information.	page 58
iwvpath.exe	Prints all or parts of the version path of a specified object.	page 151

Sample Command Trigger Scripts

This appendix contains the following sample command trigger Perl scripts:

- `email_to.pl` – sends email to a list of users when a triggering event occurs.
- `replicate_tobranch.pl` – replicates submitted files and directories from a workarea to the staging area of a specified branch.

Email Notification Script

The following Perl script, `email_to.pl`, sends email to a specified list of users when an triggering event occurs. Usage and syntax are explained in comment lines.

```
#!/usr/local/bin/perl
# email_to.pl
#
# Usage:
#     iwatasgn email_to.pl tolist
#     iwatlock email_to.pl tolist
#     iwatmkbr email_to.pl tolist
#     iwatmkwa email_to.pl tolist
#     iwatpub email_to.pl tolist
#     iwatrmbr email_to.pl tolist
#     iwatrmmed email_to.pl tolist
#     iwatrmwa email_to.pl tolist
#     iwatsub email_to.pl tolist
#     iwatunlock email_to.pl tolist
#
# (Unix)
# Example : iwatpub email_to.pl 'mktg,staff,$IW_EVENTUSER'
# Example : iwatasgn email_to.pl '$IW_ASSIGNEE,$IW_EDITOR'
# Example : iwatsub email_to.pl '$IW_EVENTUSER,mktg'
```

```
#
# (NT)
# Example : perl iwatpub perl email_to.pl mktg,staff,$IW_EVENTUSER
# Example : perl iwatasgn perl email_to.pl $IW_ASSIGNEE,$IW_EDITOR
# Example : perl iwatsub perl email_to.pl $IW_EVENTUSER,mktg
#

$| = 1;

use Config;
if ($Config{osname} eq "MSWin32") {
    $MAILER = 'blat.exe - -t';
} else {
    $MAILER = '/usr/lib/sendmail';
}

# Change $DEFDOMAIN to be default domain suffix for user email
# $DEFDOMAIN = '@YOURDOMAIN.NET';

# $FROMUSER is added to From: line in emails
$FROMUSER = 'teamsite (TeamSite notifier)';

# $SIGNATURE is added to end of messages - signature string
$SIGNATURE = "\n\nTeamXpress is brought to you by Interwoven.";

# Preprocess path to include iw-home in case script is called from
# environment without path initialization

$IW_HOME = `iwgethome`; chop $IW_HOME;
if ( $ENV{'PATH'} !~ m|$IW_HOME/bin| ) {
    $ENV{'PATH'} .= ":$IW_HOME/bin";
}

# Add $IW_HOME/lib/perl5 to Perl module include path
push @INC, "$IW_HOME/iw-perl/lib";

use File::Basename;
$MYNAME = basename($0, ".pl" );

use Env qw(PATH IWDEBUGERR IWDEBUGMAIL IW_EVENT IW_TIMESTAMP
          IW_EVENTUSER IW_EVENTROLE IW_AREA IW_FILEPATH IW_OBJID
```

```

        IW_ASSIGNEE IW_COMMENTS);

# $IWDEBUGERR true allows system calls to print errors to STDERR
# $IWDEBUGERR=1;

# $IWDEBUGMAIL true redirects output to STDOUT instead of mail
# program
# $IWDEBUGMAIL=1 unless defined($IWDEBUGMAIL);

# Redirect STDERR and set up signal
# handler to restore STDERR for final error message

$IWDEBUGERR=0 unless defined($IWDEBUGERR);
if (! $IWDEBUGERR) {
    open(SAVESTDERR, ">&STDERR") || die "Can't save STDERR, $!\n";
    open(STDERR, ">/dev/null") || die "Can't open STDERR to /dev/
        null\n";

    $SIG{__DIE__} = sub {
        open(STDERR, ">&SAVESTDERR");
        die "$MYNAME: " . $_[0];
    };
}

# Mail programs - uses reference to hash with keys being email
# users

sub MailTolist {
    my $maillist_p = $_[0];
    my $maillist = join ",", map "$_&DEFDOMAIN", keys %$maillist_p;
    return $maillist;
}

sub MailTo {
    my $maillist = MailTolist($_[0]);
    my $oldfh;

    return unless $maillist;

    unless ($IWDEBUGMAIL) {
        open (MAIL, "| $MAILER $maillist")

```



```
        or die "Couldn't open pipe to $MAILER $maillist\n";
    $oldfh = select MAIL;
}
else {
    print "Pipe to $MAILER $maillist\n";
}

# Insert mail header
print "From: $FROMUSER\n";
print "To: $maillist\n";
print "Reply-To: $FROMUSER\n";

if ( $IW_EVENT eq "AssignFile" ) { &MailAssign; }
elsif ( $IW_EVENT eq "ApproveFile" ) { &MailApprove; }
elsif ( $IW_EVENT eq "RejectFile" ) { &MailReject ; }
elsif ( $IW_EVENT eq "UnassignFile" ) { &MailUnassign; }
elsif ( $IW_EVENT eq "MarkDoneFile" ) { &MailMarkDone; } ;

print "$IW_COMMENTS\n";

&DumpAll;

print "$SIGNATURE\n";

unless ($IWDEBUGMAIL) {
    close (MAIL);
    select $oldfh;
}
}

sub MailAssign {
    print "Subject: File assigned to $IW_ASSIGNEE: $IW_FILEPATH\n\n";
    print "Timestamp:          $IW_TIMESTAMP\n";
    print "Branch:                $IW_BRANCH\n";
    print "Workarea:                $IW_AREA\n";
    print "Workarea owner:         $IW_EDITOR\n";
    print "Assigned by user:       $IW_EVENTUSER\n";
    print "File path:              $IW_FILEPATH\n";
    print "\nAssignment comments:\n" . "-" x 60 . "\n";
}
}
```

```
sub MailApprove {
    print "Subject: File approved by $IW_EVENTUSER: $IW_FILEPATH\n\n";
    print "Timestamp:          $IW_TIMESTAMP\n";
    print "Branch:              $IW_BRANCH\n";
    print "Workkarea:            $IW_AREA\n";
    print "Workkarea owner:      $IW_EDITOR\n";
    print "Approved by:          $IW_EVENTUSER\n";
    print "File path:           $IW_FILEPATH\n";
    print "\nApproval comments:\n" . "-" x 60 . "\n";
}

sub MailReject {
    print "Subject: File rejected by $IW_EVENTUSER: $IW_FILEPATH\n\n";
    print "Timestamp:          $IW_TIMESTAMP\n";
    print "Branch:              $IW_BRANCH\n";
    print "Workkarea:            $IW_AREA\n";
    print "Workkarea owner:      $IW_EDITOR\n";
    print "Rejected by:          $IW_EVENTUSER\n";
    print "File path:           $IW_FILEPATH\n";
    print "\nRejection comments:\n" . "-" x 60 . "\n";
}

sub MailUnassign {
    print "Subject: File unassigned by $IW_EVENTUSER:
        $IW_FILEPATH\n\n";
    print "Timestamp:          $IW_TIMESTAMP\n";
    print "Branch:              $IW_BRANCH\n";
    print "Workkarea:            $IW_AREA\n";
    print "Workkarea owner:      $IW_EDITOR\n";
    print "Unassigned by:        $IW_EVENTUSER\n";
    print "File path:           $IW_FILEPATH\n";
    print "\nUnassignment comments:\n" . "-" x 60 . "\n";
}

sub MailMarkDone {
    print "Subject: File marked done by $IW_EVENTUSER:
        $IW_FILEPATH\n\n";
    print "Timestamp:          $IW_TIMESTAMP\n";
    print "Branch:              $IW_BRANCH\n";
    print "Workkarea:            $IW_AREA\n";
}
```



```
print "Workarea owner:      $IW_EDITOR\n";
print "Marked done by:     $IW_EVENTUSER\n";
print "File path:         $IW_FILEPATH\n";
print "\nMark Done comments:\n" . "-" x 60 . "\n";
}

sub DumpAll {
    @vars = qw(
        IW_TIMESTAMP IW_EVENTUSER IW_EVENTROLE IW_EVENT
        IW_VPATH IW_EVENTLINE IW_BRANCH IW_EDITION
        IW_STAGINGAREA IW_WORKAREA IW_AREA IW_FILE
        IW_FILEPATH IW_OBJID IW_ASSIGNEE IW_EDITOR
        IW_COMMENTS IW_SUBMITID);
    for ($i=0; $i<=$#vars; $i++) {
        $v = $vars[$i];
        if (! $ENV{$v}) { next; }
        print "$v : $ENV{$v}\n";
    }
}

# MAIN code
#

# Grab the mail targets...
#
foreach $mailDst (split(",", $ARGV[0])) {
    ##print $mailDst;
    if ($mailDst =~ /^\/$/) {
        $mailDst =~ s/^\/$/;
        $mailDst = $ENV{$mailDst};
        ##print "changed to $mailDst\n";
    }
    $emaillist{$mailDst} = 1;
}

MailTo(\%emaillist);

# -w mode can be a bit noisy for testing - eliminate unused var
# warnings
sub fakewarnings {
    my $il = \$_DEFDOMAIN;
```



```

    my $i2 = \$SAVESTDERR;
    my $i3 = \$objid;
}

```

Replication Script

The following Perl script, `replicate_tobbranch.pl`, replicates files and directories to the staging area of a specified branch. Replication is triggered whenever the files or directories are submitted to their original workarea. This script is designed for use with the `iwat.sub` command trigger and is typically used to integrate content from one or more sub-branches back to a parent branch. Syntax and an example are included in comment lines. Due to this document's formatting constraints, some individual lines are depicted below as two or lines. These lines are indicated by a large bold-face exclamation point (!). A line ending with this character should be joined with the next line to form a single line of code.

```

#!/usr/local/bin/perl
#replicate_tobbranch.pl
##This program replicates submitted files/dirs from a workarea
##to the staging area of a specified branch. This can be useful
##when integrating content from subbranches back to parent
##branches.
##Example:
##Given TeamXpress directory layout:
##main->dev1
##      EDITION
##      STAGING
##      WORKAREA
##      ->dev2
##      EDITION
##      STAGING
##      WORKAREA
##      user1
##      file1
##
##> iwat.sub replicate_tobbranch.pl dev1
##

```



```
##When file1 is submitted to dev2 STAGING, a submit event
##will be generated, and file1 will also get submitted to
##STAGING in dev1.
push @INC, "/usr/local/lib/perl5";

#get argument as the pathname for download directory

if ($#ARGV >=0) {
    $submitto_branch="//IWSERVER/default/main/" . @ARGV[0];
} else {
    $submitto_branch="//IWSERVER/default/main/acmetest/A"; #default
}

#Assume the replicated workarea is called wal
$replicate_workarea="$submitto_branch/WORKAREA/wal";

chomp ($iwhome = `usr/bin/iwgethome`);
$iwhomebin=$iwhome . "/bin";

# environment variables available
$event = $ENV{'IW_EVENT'};          # should always be "Submit"
$timestamp = $ENV{'IW_TIMESTAMP'};
$user = $ENV{'IW_EVENTUSER'};
$role = $ENV{'IW_EVENTROLE'};
$objid = $ENV{'IW_SUBMITID'};
$path = $ENV{'IW_WORKAREA'};

# testing only, get the last event, assume it's submit for now

if ($event eq "") {
    $eventlog=`$iwhomebin/iwgetlog`;
    chop($eventlog);
    $tailevent=`grep Submit $eventlog |tail -1`;

    @tailevent=split("\t", $tailevent);

    $event=@tailevent[3];
    $timestamp = @tailevent[0];
    $user = @tailevent[1];
    $role = @tailevent[2];
    $objid = @tailevent[5];
}
```

```

    chop($objid);
    $path = @tailevent[4];
}

#
#iwlist makes sure the branch is valid
#

$existbranch=`$iwhomebin/iwlist $submitto_branch 2>&1`;
chop($existbranch);

if ($existbranch =~ /ERROR/) { #branch does not exist
    print STDERR ("$submitto_branch is not a valid branch.\n");
    exit(0);
}

#
#iwlist makes sure the replicated workarea exists in
#submitto_branch
#
$existreparea=`$iwhomebin/iwlist $replicate_workarea 2>&1`;
chop($existreparea);

if ($existreparea =~ /ERROR/) { #reparea does not exist
    print STDERR ("$replicate_workarea need to exist.\n");
    exit(0);
}

## get submit comment info and submit info field
print "GETTING SUBMIT COMMENTS\n";

$submit_cmt = `$iwhomebin/iwattrib -c submitevent -o $objid $path!
submit_cmt`;
$submit_info = `$iwhomebin/iwattrib -c submitevent -o $objid $path!
submit_info`;
$submit_cmt =~ s/\s+$/;/;
$submit_info =~ s/\s+$/;/;

## print info for debugging
print "

```

```
For Debugging...
User   : $user ($role)
Area   : $path
Time   : $timestamp
Submit Info: $submit_info
Submit Cmt : $submit_cmt
ObjId  : $objid

";

#
#find parentbranch from the path in the submit event
#

($parentbranch,$pathend) = ($path =~ /(.*?)\./WORKAREA\/(.*)/ );

print "$parentbranch\n";

#get the list of files/dirs submitted from iwevents CLT

unless(open(SUBLOG, "$iwhomebin/iwevents -s -n $objid |")) {
    print STDERR "Can't open iwevents: $!\n";
    return;
}

#can only update to a workarea, so in order to submit to a
#specific branch, first
#copy it to a workarea under that branch, then submit

    #get individual submitted file/dir name and description

    $submit_objs="";
    while (<SUBLOG>) {
        @result = split('\t', $_);
        $action = $result[1];
        $obj_type = $result[2];
        ($filename) = &URLDecode($result[3]);
        ($description) = &URLDecode($result[5]);
        $description =~ s/(\s)+/ /g;          # loose newlines/tabs
        if ($description eq "<none>") {
            $description = "[no comment specified]";
        }
    }
}
```

```

    }
    print "DEBUG:filename : $filename\n";
    print "DEBUG:obj_type : $obj_type $action\n";
    print "DEBUG:Individual Comment: $description\n\n";

    ##construct full path name for the updated file and
    ##to-replicate workarea

    $updateobj = $parentbranch . "\/STAGING" . $filename;
    $replicateobj=$replicate_workarea;

    print "DEBUG:iwupdate -r $updateobj
          $replicateobj...\n";

    # update obj to the to-replicate workarea

    @updatertn=`iwupdate -r $updateobj $replicateobj`;
    print ("${iwhomebin/iwupdate return: @updatertn\n");

    # construct submit strings <obj comment> pairs to be
    #used in iwsubmit

    $submit_objs = $submit_objs. " " . $replicateobj .
                  $filename . " ". "$description" . "'";
}

print ("DEBUG: submit_objs = $submit_objs\n");
#Submit the newly copied objs
$submitcmd ="${iwhomebin/iwsubmit -r -c \''$submit_cmt\' !
-i \''$submit_info\' '$submit_objs";
print "$submitcmd\n";

$submitrtn=`$submitcmd`;
print "DEBUG: submit rtn=$submitrtn\n";

sub URLDecode
{
#Decode a URL encoded string or array of strings
#+ is converted to space
#%NN is converted from hex
    foreach (@_) {

```



```
s/%0D%0A/ /g;  
tr/+/ /;  
s/%(..)/pack("c", hex($1))/ge;  
}  
@_;  
}
```

Appendix C

Error Codes

This table lists all the possible error codes that can be returned by a TeamXpress command-line tool. Most of them are standard errors, but there are also many TeamXpress-specific error codes. The list of TeamXpress-specific error codes starts on page 203.

#	Error	Description
General OS and NFS errors		
-1	psx_EOF	
0	psx_ENFS_OK (NFS error)	
0	psx_SUCCESS	
0	psx_YES	
1	psx_FAILURE	Generic failure.
2	psx_ENFS_NOENT (NFS error)	No such file or directory.
2	psx_ENOENT	No such file or directory.
3	psx_ESRCH	No such process.
4	psx_EINTR	Interrupted system call.
5	psx_EIO	I/O error.
5	psx_ENFS_IO (NFS error)	A hard error occurred when the operation was in progress.
6	psx_ENFS_NXIO (NFS error)	No such device or address.
6	psx_ENXIO	No such device or address.
7	psx_E2BIG	Argument list too long.
8	psx_ENOEXEC	Executable file format error.
9	psx_EBADF	Bad file number.
10	psx_ECHILD	No children.



#	Error	Description
11	psx_EAGAIN	Resource temporarily unavailable.
12	psx_ENOMEM	Not enough core memory.
13	psx_EACCES	Permission denied.
13	psx_ENFS_ACCES (NFS error)	Permission denied.
14	psx_EFAULT	Bad address.
15	psx_ENOTBLK	Block device required.
16	psx_EBUSY	Mount device busy.
17	psx_EEXIST	Item already exists.
17	psx_ENFS_EXIST (NFS error)	Item already exists.
18	psx_ENFS_XDEV (NFS error)	Attempt to do a cross-device hard link.
18	psx_EXDEV	Cross-device link.
19	psx_ENFS_NODEV (NFS error)	No such device.
19	psx_ENODEV	No such device.
20	psx_ENFS_NOTDIR (NFS error)	Directory expected, but wasn't one.
20	psx_ENOTDIR	Directory expected, but wasn't one.
21	psx_EISDIR	Is a directory.
21	psx_ENFS_ISDIR (NFS error)	Is a directory.
22	psx_EINVAL	Invalid argument.
22	psx_ENFS_INVALID (NFS error)	Invalid argument or unsupported argument for an operation.
23	psx_ENFILE	File table overflow.
24	psx_EMFILE	Too many open files.
25	psx_ENOTTY	Inappropriate I/O control for device.
26	psx_ETXTBSY	Text file busy.
27	psx_EFBIG	File too large.
27	psx_ENFS_FBIG (NFS error)	File too large.

#	Error	Description
28	psx_ENFS_NOSPC (NFS error)	Insufficient space to complete operation.
28	psx_ENOSPC	Insufficient space to complete operation.
29	psx_ESPIPE	Illegal seek.
30	psx_ENFS_ROFS (NFS error)	File system is read-only or archive is frozen.
30	psx_EROFS	File system is read-only or archive is frozen.
31	psx_EMLINK	Too many links.
31	psx_ENFS_MLINK (NFS error)	Too many hard links.
32	psx_EPIPE	Broken pipe.
33	psx_EDOM	Math argument out of domain of function.
34	psx_ERANGE	Math result not representable.
35	psx_ENOMSG	No message of desired type.
36	psx_EIDRM	Identifier removed.
37	psx_ECHRNG	Channel number out of range.
38	psx_EL2NSYNC	UNIX boot level 2 not synchronized.
39	psx_EL3HLT	UNIX boot level 3 halted.
40	psx_EL3RST	UNIX boot level 3 reset.
41	psx_ELN RNG	Link number out of range.
42	psx_EUNATCH	Protocol driver not attached.
43	psx_ENOCSI	No CSI structure available.
44	psx_EL2HLT	UNIX boot level 2 halted.
45	psx_EDEADLK	Resource deadlock condition.
46	psx_ENOLCK	No record locks available.
47	psx_ECANCELED	Operation canceled.
48	psx_ENOTSUP	Operation not supported.
Filesystem Quotas		
49	psx_EDQUOT	Disk quota exceeded.



#	Error	Description
Convergent Error Returns		
50	psx_EBADE	Invalid exchange.
51	psx_EBADR	Invalid request descriptor.
52	psx_EXFULL	Exchange full.
53	psx_ENOANO	No anode.
54	psx_EBADRQC	Invalid request code.
55	psx_EBADSLT	Invalid slot.
56	psx_EDEADLOCK	File locking deadlock error.
57	psx_EBFONT	Bad font file format.
Stream Problems		
60	psx_ENOSTR	Device not a stream.
61	psx_ENODATA	No data (for no delay I/O).
62	psx_ETIME	Timer expired.
63	psx_ENOSR	Out of streams resources.
64	psx_ENONET	Machine is not on the network.
65	psx_ENOPKG	Package not installed.
66	psx_EREMOTE	The object is remote.
67	psx_ENOLINK	The link has been severed.
68	psx_EADV	Advertise error.
69	psx_ESRMNT	Srmount error.
70	psx_ECOMM	Communication error on send.
71	psx_EPROTO	Protocol error.
72	psx_EBADRPC	RPC struct is bad; error checks failed.
73	psx_EDOTDOT	RFS specific error.
74	psx_EMULTIHOP	Multihop attempted.
77	psx_EBADMSG	Trying to read unreadable message.

#	Error	Description
78	psx_ENAMETOOLONG	Path name is too long.
79	psx_EOVERFLOW	Value too large to be stored in data type.
80	psx_ENOTUNIQ	Given log. name not unique.
81	psx_EBADFD	File descriptor invalid for this operation.
82	psx_EREMCHG	Remote address changed.
Shared Library Problems		
83	psx_ELIBACC	Can't access a needed shared library.
84	psx_ELIBBAD	Accessing a corrupted shared library.
85	psx_ELIBSCN	.lib section in a.out corrupted.
86	psx_ELIBMAX	Attempting to link in too many libraries.
87	psx_ELIBEXEC	Attempting to exec a shared library.
88	psx_EILSEQ	Illegal byte sequence.
89	psx_ENOSYS	Unsupported file system operation.
90	psx_ELOOP	Symbolic link loop.
91	psx_ERESTART	Restartable system call.
92	psx ESTRPIPE	Pipe or FIFO doesn't sleep in stream head.
93	psx_ENOTEMPTY	Directory not empty.
94	psx_EUSERS	Too many users (for UFS).



#	Error	Description
BSD Networking Software		
argument errors		
95	psx_ENOTSOCK	Socket operation on non-socket.
96	psx_EDESTADDRREQ	Destination address required.
97	psx EMSGSIZE	Message too long.
98	psx_EPROTOTYPE	Protocol wrong type for socket.
99	psx_ENOPROTOOPT	Protocol not available.
Linux-specific		
117	psx_EUCLEAN	Structure needs cleaning.
118	psx_ENOTNAM	Not a XENIX named type file.
119	psx_ENAVAIL	No XENIX semaphores available.
120	psx_EISNAM	Is a named type file.
121	psx_EREMOTEIO	Remote I/O error.
BSD Networking Software		
120	psx_EPROTONOSUPPORT	Protocol not supported.
122	psx_EOPNOTSUPP	Operation not supported on socket.
123	psx_EPFNOSUPPORT	Protocol family not supported.
124	psx_EAFNOSUPPORT	Address family not supported by protocol family.
125	psx_EADDRINUSE	Address already in use.
126	psx_EADDRNOTAVAIL	Can't assign requested address.
Operational Errors		
127	psx_ENETDOWN	Network is down.
128	psx_ENETUNREACH	Network is unreachable.
129	psx_ENETRESET	Network dropped connection because of reset.
130	psx_ECONNABORTED	Software caused connection abort.

#	Error	Description
131	psx_ECONNRESET	Connection reset by peer.
132	psx_ENOBUFS	No buffers available.
133	psx_EISCONN	Socket is already connected.
134	psx_ENOTCONN	Socket is not connected.
143	psx_ESHUTDOWN	Can't send after socket shutdown.
144	psx_ETOOMANYREFS	Too many references: can't splice.
145	psx_ETIMEDOUT	Connection timed out.
146	psx_ECONNREFUSED	Connection refused.
147	psx_EHOSTDOWN	Host is down.
148	psx_EHOSTUNREACH	No route to host.
149	psx_EALREADY	Operation already in progress/done.
150	psx_EINPROGRESS	Operation now in progress.
151	psx_ESTALE	Stale NFS file handle.
301	psx_EPERM	No permission; or not super-user.
301	psx_ENFSPERM (NFS error)	Not owner.
311	psx_EWOULDBLOCK	Berkeley sockets error - operation would block till completion.
FreeBSD-specific		
398	psx_EJUSTRETURN	(400-2) don't modify registration, just return.
473	psx_ERPCMISMATCH	RPC version wrong.
474	psx_EPROGUNAVAIL	RPC program not available.
475	psx_EPROGMISMATCH	Program version wrong.
476	psx_EPROCUNAVAIL	Bad procedure for program.
479	psx_EFTYPE	Inappropriate file type or format.
480	psx_EAUTH	Authentication error.
481	psx_ENEEDAUTH	Need authenticator.

#	Error	Description
Linux-specific		
512	psx_ERESTARTSYS	Linux system startup error.
513	psx_ERESTARTNOINTR	Linux system startup error.
514	psx_ERESTARTNOHAND	Restart if no handler.
515	psx_ENOIOCTLCMD	No I/O control command.
SunOS4-specific		
567	psx_EPROCLIM	Too many processes.
581	psx_ERREMOTE	Object is remote.
OSF/AXP-specific		
688	psx_ECLONEME	Tells open to clone the device.
689	psx_EDIRTY	Mounting a dirty (unchecked) file system.
690	psx_EDUPPKG	Duplicate package name on install.
691	psx_EVERSION	Version number mismatch.
693	psx_ENOSYM	Unresolved symbol name.
695	psx_EFAIL	Cannot start operation.
697	psx_EINPROG	Operation (now) in progress.
698	psx_EMTIMERS	Too many timers.
700	psx_EAIO	Internal AIO operation complete.
723	psx_ESOFT	Indicates a correctable error.
724	psx_EMEDIA	Returned by a disk driver to indicate a hard ECC error or similar disk media failure.
725	psx_ERELOCATED	“Success” code indicating that a defect relocation request was performed successfully.
Misc. generic error numbers		
901	psx_EEXITING	Exiting process/thread.
905	psx_EIMPLEMENT	Function not yet implemented.
910	psx_EBUFTOOSMALL	Buffer is too small for output.

#	Error	Description
911	psx_ESTRTOOLONG	String argument is too long.
920	psx_ENOTFOUND	Object being looked up was not found.
925	psx_ENOUSER	No such user.
930	psx_ERPCFAILURE	RPC connection failure.
980	psx_ENOAUTHENDONE	No authentication was done.
981	psx_ENOAUTHORDONE	No authorization was done.
999	psx_NO	“No” is the response.
Interwoven-specific error codes		
2000	psx_EITERENDED	Iteration ended.
2001	psx_EITERNOENT	No iteration entry.
2003	psx_ENOBASEED	Invalid base edition for operation.
2004	psx_ENOSE	No server found.
2005	psx_ENOAR	No archive found.
2006	psx_ENOBR	No branch found.
2007	psx_ENOWA	No workarea found.
2008	psx_ENOED	No edition found.
2009	psx_ENOFSE	No file or directory found.
2010	psx_ENOPREVFSE	No previous file or directory found.
2011	psx_ENOWADIR	Workarea directory not found.
2012	psx_ENOEDDIR	Edition directory not found.
2013	psx_ENOSA	No staging area found.
2014	psx_ENOOP	No operation found.
2015	psx_ENOTAG	No tag found.
2016	psx_ECONFLICTS	Conflicts prevent operation.
2017	psx_ECHECKEDOUT	Checked out files prevent submitting.
2018	psx_EWABRBAD	Workarea does not match branch.



#	Error	Description
2019	psx_EONLYED	Cannot destroy last remaining edition on branch.
2100	psx_EBADNAME	Illegal name given.
2102	psx_EBADVPATH	Bad version path.
2150	psx_EPRIVATE	Illegal operation on private file or directory.
2200	psx_ENOROLE	No such role.
2300	psx_EILOP	Illegal operator.
2301	psx_ENOATTRIB	No such attribute.
2400	psx_ERESERVED	Object is being reserved by someone else.
2401	psx_ESAMEFSE	File or directory is the same as the other version.
2402	psx_ENEWERFSE	File or directory is newer than the other version.
2403	psx_ENOUPDATE	No update performed.
2404	psx_ENOSUBMIT	No submit performed.
2405	psx_ENOCMPRS	Compressing workarea, staging area, or latest edition is not allowed.
2406	psx_ECMPRED	Edition is already compressed.
2407	psx_EUNCMPRED	Edition is already uncompressed.
2408	psx_EMULTIBR	More than one branch found.
2409	psx_EMULTIWA	More than one workarea found.
2410	psx_ENOGROUP	No such group.
2411	psx_EDELETED	File had been deleted.
2412	psx_EASSIGNED	Assigned files prevent submitting.
2413	psx_ENOTAUTHOR	Must be Author.
2414	psx_EOLDERFSE	File is older than the other version.
2415	psx_ENOTWAMBR	Not owner or member of workarea.

#	Error	Description
2416	psx_EABORTED	Operation aborted.
2417	psx_ENEWERDELETION	File or directory is a deletion of the other file or directory.
2418	psx_ENOROLEAUTH	Role is not authorized for this user.
2419	psx_EWRONGUSERPWD	Invalid username/password combination.
2420	psx_ENODOMAIN	Domain does not exist.
2421	psx_ENOSADIR	Staging area directory not found.
2422	psx_ENOTSUSER	No such TeamXpress user.
ODBC errors		
21098	psx_ECONN_INVALID_HANDLE	ODBC code: invalid handle.
21099	psx_ECONN_ERROR	ODBC code: generic error.
21101	psx_ECONN_SUCCESS_WITH_INFO	ODBC code: success; more information is coming.
21199	psx_ECONN_NEED_DATA	ODBC code: need more data.
21200	psx_ECONN_NO_DATA_FOUND	ODBC code: no data found.
Interwoven Object Transfer Protocol		
21501	psx_ECOP_ILLEXPORT	Illegal transport.
21502	psx_ECOP_XPORTNOTAVAIL	Transport not available.
21503	psx_ECOP_NOSERVER	Server not found.
21504	psx_ECOP_BUFTOOSMALL	Buffer too small.
21505	psx_ECOP_NOTENOUGHDATA	Not enough bytes.
21506	psx_ECOP_BADBYTECOUNT	Incorrect byte count.
21507	psx_ECOP_ILLLDATA	Illegal data.
21508	psx_ECOP_METHODNOTAVAIL	Specified method not available.
21510	psx_ECOP_VERNOTSUP	COP version not supported.
21512	psx_ECOP_BADMSGINDEX	Incorrect message index.
22001	psx_ESC_FAILURE	General Special Channel failure.



#	Error	Description
22004	psx_ESC_BADKEY	Bad Special Channel key.
22006	psx_ESC_NOCHANNEL	No Special Channel.
22008	psx_ESC_NODIR	No Special Channel directory.
22010	psx_ESC_CANTCREATE	Cannot create Special Channel.
NFS errors		
100063	psx_ENFS_NAMETOOLONG	File name too long.
100066	psx_ENFS_NOTEMPTY	Directory not empty.
100069	psx_ENFS_DQUOT	Disk quota exceeded.
100070	psx_ENFS_STALE	Invalid file handle.
100071	psx_ENFS_REMOTE	The file handle given in the arguments referred to a file on a non-local file system on the server.
100099	psx_ENFS_WFLUSH	The server's write cache used in the "WRITECACHE" call got flushed to disk.
110001	psx_ENFS_BADHANDLE	Illegal NFS file handle. The file handle failed internal consistency checks.
110002	psx_ENFS_NOT_SYNC	Update synchronization mismatch was detected during a SETATTR operation.
110003	psx_ENFS_BAD_COOKIE	READDIR or READDIRPLUS cookie is stale.
110004	psx_ENFS_NOTSUPP	Operation is not supported.
110005	psx_ENFS_TOOSMALL	Buffer or request is too small.
110006	psx_ENFS_SERVERFAULT	An error occurred on the server which does not map to any of the legal NFS version 3 protocol error values
110007	psx_ENFS_BADTYPE	An attempt was made to create an object of a type not supported by the server.
110008	psx_ENFS_JUKEBOX	The server initiated the request, but was not able to complete it in a timely fashion.

Index

- A
 - areas
 - comparison 76
 - list of 107
 - modified files in 111
 - attributes
 - extended 94
 - of servers 66
 - autopivate.cfg 50
- B
 - backing store
 - conversion 32
 - diagnose 29
 - fixing problems 34
 - hot backups 27
 - iwfsck 29
 - iwfsconvert 32
 - iwfsfix 34
 - iwfsshink 36
 - iwfsstripe 38
 - location 46
 - multiple file systems 38
 - recovery 51
 - removing duplicate file contents 36
 - backup 22
 - branch
 - creating 121, 164
 - deleting 167
 - locks 110
 - removing 139
 - renaming 134
 - branch operation 60
 - iwmkbr 121
 - iwrnbr 139
 - BSD networking software error
 - codes 200
- C
 - CGI task 73
 - command trigger 168
 - iwatasgn 161
 - iwatcreate 162
 - iwat-env 159
 - iwatlock 163
 - iwatmkbr 164
 - iwatmkwa 165
 - iwatpub 166
 - iwatrmbr 167
 - iwatrmwa 169
 - iwatserver 170
 - iwatsub 171
 - iwatunlock 172
 - iwatupdate 173
 - replication script 189
 - sample scripts 183
 - starting 154
 - testing 159
 - compress edition 78
 - configuration files
 - interfacingwith 23
 - location 42
 - rereading 50
 - conflicts
 - resolving 135
 - convergent error returns 198
- D
 - decoding %xx lines 80
 - deleting editions 140
 - deleting workarea 143
 - deploy website content 83
- E
 - deployment 60
 - iwdelcp 81
 - iwdeploy 83
 - diagnose backing store 29
 - difference list 85
 - differences
 - files versions 132
 - directories
 - modified 111
 - synchronizing 85
 - directory
 - creation 162
 - directory paths 16
- E
 - edition
 - compressing 78
 - deleting 140, 168
 - name of last 106
 - name of next 124
 - propagate differences 81
 - publishing 166
 - renaming 134
 - edition operation 60, 127
 - iwcompress 78
 - iwlasted 106
 - iwnexted 124
 - iwrmed 140
 - email notification script 183
 - encoding HTML %xx lines 89
 - environment data
 - collecting 52
 - environment variables 159
 - error codes 195
 - argument errors 200
 - BSD networking software 200



- convergent error returns 198
- file system quotas 197
- free BSD 201
- general OS and NFS 195
- generic 202
- Interwoven 203
- Interwoven Object Transfer Protocol 205
- Linux-specific 200, 202
- NFS 206
- ODBC 205
- operational 200
- OSF/AXP 202
- shared library 199
- stream problems 198
- SunOS4 202
- event
 - occurring 170
 - triggering 171
- events log
 - contents 40
- extended attributes 94
- external task 73
- F**
- file
 - action on 161
 - adding to task 63
 - modified 111
 - removing duplicate contents 36
 - removing from task 142
 - renaming 134
 - reverting 136
 - submitting 144, 171
 - unlocking 172
 - update 150
 - updating 173
 - version differences 132
 - versions 74
- file difference list 87
- file system quota error codes 197
- filtering 56
- finishing task 147
- free BSD specific error codes 201
- freeze 27, 54
- full backup 22
- G**
- general development 59
 - iwattrib 64
 - iwckrole 75
 - iwdecode 80
 - iwencode 89
 - iwextattr 94
 - iwlist 107
 - iwmenu 116
 - iwrename 134
 - iwvpath 151
- general OS and NFS errors 195
- generic error numbers 202
- Get Latest 150
- H**
- history of workarea 90
- HTML %xx lines
 - decoding 80
 - encoding 89
- I**
- incremental backup 22
- incremental file changes 87
- Interwoven error codes 203
- Interwoven Object Transfer Protocol error codes 205
- iw.cfg 23, 50
- iwabort 21
- iwaddtaskfile 63
- iwat 158
- iwatasn 161
- iwat-env 159
- iwatlock 162, 163
- iwatmkbr 164
- iwatmkwa 165
- iwatpub 166
- iwatrmbr 167
- iwatrmmed 168
- iwatrmwa 169
- iwatsub 171
- iwattrib 64
- iwatunlock 172
- iwatupdate 173
- iwbackup 22
- iwcallback 73
- iwcat 74
- iwchgrp 75
- iwcmp 76
- iwcompress 78
- iwconfig 23, 78
- iwdecode 80
- iwdelcp 81
- iwdeploy 83
- iwdiffapply 85
- iwdiffdir 87
- iwencode 89
- iwevents 90
- iwextattr 94
- iwfailsafe 25
- iwfreeze 27
- iwfsck 29
- iwfsconvert 32
- iwfsfix 34
- iwfsshrink 36
- iwfsstripe 38
- iwgetelog 40
- iwgethome 41
- iwgetlocation 42
- iwgetlog 44
- iwgetmount 45
- iwgetstore 46
- iwgettrace 47
- iwgetwfobj 96
- iwinvokejob 103
- iwjobc 104
- iwjobvariable 105
- iwlasted 106
- iwlist 107
- iwlistlocks 110
- iwlistmod 111
- iwlock 113
- iwlockinfo 114
- iwlsat 174
- iwmenu 116
- iwmkbr 121

- iwmkwa 123
- iwnexted 124
- iwproxy 48
- iwprv 125
- iwpublish 127
- iwqueryjobs 129
- iwquerytasks 130
- iwrscdiff 132
- iwrecentusers 49
- iwrename 134
- iwreset 50
- iwrestore 51
- iwretryjobop 135
- iwrevert 136
- iwrlog 137
- iwrmat 175
- iwrnbr 139
- iwrmed 140
- iwrnjob 141
- iwrntaskfile 142
- iwrnwa 143
- iwsi 52
- iwstat 54
- iwsubmit 56, 144
- iwtaketask 146
- iwtaskselect 147
- iwttestcfg 56
- iwuidname 57
- iwundochoice 148
- iwunlock 149
- iwupdate 150
- iwversion 58
- iwvpath 151

J

- job
 - instantiating 104
 - querying 129
 - removing from server 141
 - running 103

L

- Linux-specific error codes 200, 202
- location 45

- locks 110
 - directory 163
- log
 - file locations 42
 - revision 137
 - server 44
 - trace 47

M

- metadata
 - obtaining 64
- metadata files
 - merging 32
- modifications
 - list of 76
- mount point 45
- multiple file system backing store 38

N

- NSF error codes 206

O

- object IDs 17
- objids 17
- ODBC error codes 205
- operational error codes 200
- operations
 - deleting from server 21
- OSF/AXP error codes 202

P

- paths
 - directory 16
 - version 13, 15
- print
 - version path 151
- private workareas 125
- production server
 - deploy to 83
- program
 - executing 73
- proxy server 48
- publishing staging area 127

Q

- querying jobs 129
- querying tasks 130

R

- relative vpaths 15
- release information 58
- removing branch 139
- reverse task transition 148
- revert file to previous version 136
- revision log 137
- roles
 - user 75

S

- script
 - email notification 183
 - replication 189
- server
 - log 44
 - proxy 48
 - removing job 141
- server workflow
 - displaying 96
- shared library error codes 199
- shared task 146
- SmartContext Editing tab
 - modifying 116
- staging
 - publishing 127
 - submitting to 144
- stream problem error codes 198
- submit
 - resolving conflicts 135
- submit filtering 56
- submit to staging 144
- submit.cfg 50
- SunOS4 error codes 202
- synchronizing directories 85
- system activity 54
- system information 19
 - iwgetlog 40
 - iwgethome 41
 - iwgetlocation 42



- iwgetlog 44
- iwgetmount 45
- iwgetstore 46
- iwgettrace 47
- iwrecentusers 49
- iwstat 54
- iwversion 58
- system services 20
- iwabort 21
- iwbackup 22
- iwconfig 23
- iwfreeze 27
- iwproxy 48
- iwreset 50
- iwrestore 51
- iwsi 52
- iwtestcfg 56
- iwuidname 57

T

- task
 - adding file 63
 - assigning 146
 - finishing 147
 - querying 130
 - removing a file from 142
- tasks
 - reverse transition 148
- TeamXpress
 - high availability 52
- TeamXpress program files
 - location 41
- tracelogs
 - location 47

U

- uncompress edition 78
- unfreeze 27
- unlock 172
- update tasks
 - resolving conflicts 135
- users
 - list of 49
 - login name 57

- role 75

V

- version management 61
 - iwcat 74
 - iwcmp 76
 - iwdiffapply 85
 - iwdiffdir 87
 - iwrscdiff 132
 - iwrevert 136
 - iwrlog 137
- version paths 13
 - printing 151
 - relative 15
- vpaths 13, 74
 - relative 15

W

- workarea
 - creating 123, 165
 - deleting 169
 - history 90
 - locks 110
 - private 125
 - removing 143
 - renaming 134
 - submitting 144
 - update 150
- workarea operation 60
 - iwevents 90
 - iwlistlocks 110
 - iwlistmod 111
 - iwmkwa 123
 - iwprv 125
 - iwrnw 143
 - iwsubmit 144
 - iwupdate 150
- workflow variables
 - manipulating 105
- workflow/job 61
 - iwaddtaskfile 63
 - iwcallback 73
 - iwgetwfobj 96
 - iwinvokejob 103

- iwjobc 104
- iwjobvariable 105
- iwqueryjobs 129
- iwquerytasks 130
- iwretryjobop 135
- iwrnwjob 141
- iwrnwtaskfile 142
- iwtaketask 146
- iwtaskselect 147
- iwundochoice 148