



IBM VisualAge TeamConnection Enterprise Server

# Administrator's Guide

*Version 3.0*





IBM VisualAge TeamConnection Enterprise Server

# Administrator's Guide

*Version 3.0*

**Second Edition (October 1998)**

**Note**

Before using this document, read the general information under "Notices" on page xi.

This edition applies to Version 3.0 of the licensed program IBM TeamConnection and to all subsequent releases and modifications until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product.

Order publications by phone or fax. The IBM Software Manufacturing Company takes publication orders between 8:30 a.m. and 7:00 p.m. eastern standard time (EST). The phone number is (800) 879-2755. The fax number is (800) 284-4721.

You can also order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address below.

A form for comments appears at the back of this publication. If the form has been removed, address your comments to:

IBM Corporation  
Attn: Information Development  
Department T99B/Building 062  
P.O. Box 12195  
Research Triangle Park, NC, USA 27709-2195

You can fax comments to (919) 254-0206.

If you have comments about the product, address them to:

IBM Corporation  
Attn: Department TH0/Building 062  
P.O. Box 12195  
Research Triangle Park, NC, USA 27709-2195

You can fax comments to (919) 254-4914.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© **Copyright International Business Machines Corporation 1992, 1995, 1996, 1997, 1998. All rights reserved.**  
Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

---

# Contents

<b>Figures</b> . . . . .	ix
<b>Notices</b> . . . . .	xi
<b>Trademarks</b> . . . . .	xiii
<b>About this book</b> . . . . .	xv
How this book is organized . . . . .	xv
Conventions . . . . .	xvi
Tell us what you think . . . . .	xvii

---

## Part 1. Introducing IBM VisualAge TeamConnection Enterprise Server . . . . . 1

<b>Chapter 1. An introduction to TeamConnection</b> . . . . .	3
TeamConnection definitions . . . . .	4
TeamConnection's client/server architecture . . . . .	4
TeamConnection database . . . . .	5
Interfaces . . . . .	5
Families . . . . .	5
Users and host lists . . . . .	5
Parts . . . . .	6
Components . . . . .	7
Releases . . . . .	7
Work areas . . . . .	8
Drivers . . . . .	9
Defects and features . . . . .	9
Processes . . . . .	9
Build . . . . .	11
Packaging . . . . .	11
Hardware and software requirements . . . . .	12
Requirements for TeamConnection for AIX . . . . .	12
Requirements for TeamConnection for HP-UX . . . . .	14
Requirements for TeamConnection for Solaris . . . . .	16
Requirements for TeamConnection for OS/2 . . . . .	18
Requirements for TeamConnection for Windows . . . . .	21
Requirements for MVS build servers . . . . .	22
<b>Chapter 2. Administrator Tasks</b> . . . . .	23
System Administrators . . . . .	24
Family Administrators . . . . .	25
Build Administrators . . . . .	25

---

## Part 2. Designing and creating your TeamConnection environment . . . . . 27

<b>Chapter 3. Creating your TeamConnection family</b> . . . . .	29
---	----

Planning your families . . . . .	29
DB2 considerations . . . . .	29
DB2 instances . . . . .	30
Database naming conventions . . . . .	30
Database configuration parameters . . . . .	31
Creating a family . . . . .	32
Using the family properties notebook . . . . .	34
Required . . . . .	35
Configurable fields . . . . .	37
Processes . . . . .	38
User exits . . . . .	38
Groups . . . . .	39
Adding an existing family to the Family Administrator window . . . . .	39
For further information . . . . .	40
<b>Chapter 4. Starting and stopping the servers . . . . .</b>	<b>41</b>
Specifying the number of daemons to start . . . . .	41
Setting up the mail facility . . . . .	42
Starting the servers . . . . .	42
Using the Family Administrator GUI . . . . .	42
Using teamcd . . . . .	44
Stopping the servers . . . . .	46
<b>Chapter 5. Setting up your family structure . . . . .</b>	<b>47</b>
Planning your components . . . . .	47
Organizing the component hierarchy . . . . .	47
Determining component ownership . . . . .	49
Naming the components . . . . .	50
Determining access to components . . . . .	50
Planning your releases . . . . .	50
Relating releases with components . . . . .	51
Selecting serial or concurrent development . . . . .	52
Controlling database growth . . . . .	52
Naming your releases . . . . .	53
Planning your processes . . . . .	53
Component processes . . . . .	54
Release processes . . . . .	55
How processes might change during development . . . . .	57
Using the driver subprocess . . . . .	58
Creating components and releases . . . . .	59
Creating components . . . . .	59
Creating releases . . . . .	60
Creating a new release from an old release . . . . .	61
<b>Chapter 6. Preparing for your users . . . . .</b>	<b>63</b>
Planning for user IDs . . . . .	63
Host only . . . . .	63
Password only . . . . .	64
Password or host . . . . .	64

None . . . . .	64
Login managers . . . . .	65
Creating user IDs . . . . .	66
Adding and modifying passwords . . . . .	67
Planning for host lists . . . . .	68
Creating host list entries . . . . .	69
Planning for user access to TeamConnection data.	70
What are the TeamConnection authority levels?	71
What are authority groups? . . . . .	72
Creating or modifying authority groups . . . . .	72
Granting authority to users . . . . .	74
Granting or restricting access . . . . .	76
Removing an entry from an access list . . . . .	76
Planning for user notification . . . . .	77
What are interest groups? . . . . .	78
Creating or modifying interest groups . . . . .	78
Working with notification lists . . . . .	80
Displaying interest groups . . . . .	80
Adding an entry to a notification list . . . . .	81
Removing an entry from a notification list. . . . .	82
<b>Chapter 7. Working with configurable fields . . . . .</b>	<b>83</b>
Defining configurable field types. . . . .	85
Defining dependent relationships between configurable field types . . . . .	87
Defining configurable fields . . . . .	88
Creating and modifying configurable fields . . . . .	89
Displaying configurable field properties . . . . .	92
Changing report formats . . . . .	93
The stanza report . . . . .	93
The table report . . . . .	96
<b>Chapter 8. Configuring family processes . . . . .</b>	<b>99</b>
Planning your changes. . . . .	99
Modifying or creating configurable processes . . . . .	99
<b>Chapter 9. Providing user exits . . . . .</b>	<b>103</b>
Writing user exit programs . . . . .	104
Environment file . . . . .	110
Setting up user exits . . . . .	111
Configuring user exit parameters . . . . .	113
Sample user exit programs . . . . .	115
<b>Chapter 10. TeamConnection shadows . . . . .</b>	<b>117</b>
Shadow types. . . . .	117
Shadow properties . . . . .	118
Shadow actions . . . . .	120
When does shadowing happen . . . . .	120
Writing shadowing programs . . . . .	121
Shadowing program interface . . . . .	121

Shadowing program requirements . . . . .	122
Shadowing program output . . . . .	123
Sample shadowing program . . . . .	123

---

**Part 3. Maintaining the TeamConnection server . . . . . 125**

<b>Chapter 11. Maintaining your TeamConnection environment . . . . .</b>	<b>127</b>
Changing the age of defects and features . . . . .	128
The age utility . . . . .	128
The resetAge utility . . . . .	129
Resolving TeamConnection errors . . . . .	129
Using the system error log (syslog.log) . . . . .	129
Using the audit log (audit.log) . . . . .	130
Using the trace facility . . . . .	136
Backing up the TeamConnection database . . . . .	137
<b>Chapter 12. Enhancing SQL performance . . . . .</b>	<b>139</b>
Collecting statistics using the RUNSTATS utility . . . . .	139
Analyzing statistics . . . . .	140
Reorganizing table data . . . . .	141
Applying these techniques to TeamConnection . . . . .	142
When REORG, RUNSTATS, and REBIND do not improve performance . . . . .	142
Table spaces and buffer pools . . . . .	143
Configuration and tuning . . . . .	143
<b>Chapter 13. Monitoring family use . . . . .</b>	<b>145</b>
Using the server daemon monitor . . . . .	145
Using the monitor on the Family Servers window . . . . .	145
Using the monitor command . . . . .	147
Monitoring the activity of the server daemons . . . . .	149
Detecting time-consuming requests . . . . .	149
Monitoring server daemon problems . . . . .	150
Using the license monitor . . . . .	150
How the license monitor counts users . . . . .	151
Using the tcllicmon command . . . . .	152
Reporting highest uses . . . . .	153
Displaying a full use report . . . . .	154
<b>Chapter 14. Server tools . . . . .</b>	<b>159</b>
Using tcqry . . . . .	159
Using tcupdb . . . . .	160

---

**Part 4. Appendixes . . . . . 161**

<b>Appendix A. Family administration commands . . . . .</b>	<b>163</b>
Creating a family database . . . . .	163
Creating an initial superuser for a family . . . . .	164
Creating or modifying authority groups . . . . .	165

Editing the <code>authorit.ld</code> file . . . . .	165
Reloading the authority table . . . . .	166
Creating or modifying interest groups . . . . .	167
Editing the <code>interest.ld</code> file . . . . .	167
Reloading the interest table . . . . .	167
Configuring component or release processes . . . . .	168
Editing the <code>comproc.ld</code> and <code>relproc.ld</code> files . . . . .	168
Reloading the configurable process tables . . . . .	170
Defining configurable field types . . . . .	170
Reloading the config table . . . . .	172
Updating database views with new configurable field information . . . . .	173
Changing report formats . . . . .	174
Updating TargetView and ConfigPartView Reports . . . . .	178
Setting up user exits . . . . .	178
Editing the <code>userExit</code> file . . . . .	178
Creating customized parameter lists . . . . .	180
Rebinding the family database . . . . .	180
<b>Appendix B. Configurable field types . . . . .</b>	<b>181</b>
Configurable field types . . . . .	181
<b>Appendix C. User exit parameters . . . . .</b>	<b>191</b>
Parameters passed to user exit programs . . . . .	191
User exit parameter definitions . . . . .	212
<b>Appendix D. Environment Variables . . . . .</b>	<b>225</b>
Setting environment variables . . . . .	231
<b>Appendix E. TeamConnection NLS and DBCS considerations . . . . .</b>	<b>233</b>
Overview of TeamConnection NLS and DBCS support . . . . .	233
Language and culture sensitive information in TeamConnection . . . . .	233
Supported locales (languages and code pages) . . . . .	235
Characteristics and limitations of NLS and DBCS support . . . . .	240
No conversion of code points when exchanging data . . . . .	240
Exceptions to the handling of characters in TeamConnection . . . . .	242
All clients in the same host must use the same language (Intel only) . . . . .	244
Untranslated strings that are visible to the users . . . . .	244
DBCS Limitations . . . . .	244
Installation, administration, and runtime issues . . . . .	245
Installation issues related to NLS and DBCS . . . . .	245
Family administration issues . . . . .	248
Client runtime issues . . . . .	249
<b>Appendix F. Worksheets . . . . .</b>	<b>251</b>
Authority groups worksheet . . . . .	251
Interest groups worksheet . . . . .	256
Configurable processes worksheets . . . . .	261
<b>Service and Support . . . . .</b>	<b>263</b>

VisualAge TeamConnection Services! . . . . .	263
VisualAge TeamConnection Support! . . . . .	263
IBM Lotus Passport Advantage Program . . . . .	263
DB2 Service Maintenance and Technical Library . . . . .	263
For North American Customers . . . . .	264
Support for Customers Outside North America . . . . .	265
<b>Bibliography</b> . . . . .	267
IBM VisualAge TeamConnection Enterprise Server library . . . . .	267
TeamConnection technical reports . . . . .	268
DB2 . . . . .	268
Related publications . . . . .	269
<b>Glossary</b> . . . . .	271
<b>Index</b> . . . . .	281
<b>Readers' Comments — We'd Like to Hear from You</b> . . . . .	287

## Figures

1. A sample TeamConnection client/server network . . . . .	4
2. Sample of a component hierarchy . . . . .	7
3. Parts, releases, and components. . . . .	8
4. Family Properties notebook . . . . .	35
5. Family Servers window . . . . .	43
6. A hierarchy representing product organization . . . . .	48
7. A hierarchy showing parallel components . . . . .	48
8. Components with more than one parent . . . . .	49
9. A hierarchy showing component ownership . . . . .	50
10. The release-component relationship. . . . .	51
11. Using the driver subprocess . . . . .	58
12. Create Components window . . . . .	59
13. Create Releases window . . . . .	60
14. Create User window . . . . .	66
15. Add Host window . . . . .	69
16. Granting authority to other users . . . . .	70
17. Authority Group Settings window. . . . .	73
18. Show Authority Actions window . . . . .	75
19. Restrict Access window . . . . .	76
20. Remove Access window . . . . .	77
21. Interest Group Settings window . . . . .	79
22. Show Interest Actions window. . . . .	81
23. Add Notification window. . . . .	81
24. Remove Notification window . . . . .	82
25. Field Type window . . . . .	86
26. Set Condition window . . . . .	88
27. Release Configurable Fields window . . . . .	90
28. New Field window. . . . .	91
29. Sample stanza report displayed after adding configurable fields . . . . .	94
30. Stanza View Format Settings . . . . .	95
31. Sample table report displayed after adding configurable fields . . . . .	96
32. Table View Format Settings . . . . .	97
33. Component Process Settings window . . . . .	100
34. User Exit Settings. . . . .	112
35. Pre-Check window . . . . .	113
36. Pre-Check window for PartAdd . . . . .	115
37. Sample of an audit log file . . . . .	131
38. Family Servers window . . . . .	146
39. Sample report format after adding configurable fields . . . . .	177



---

## Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Subject to IBM's valid intellectual property or other legally protectable rights, any functionally equivalent product, program, or service may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY, USA 10594.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact the Site Counsel, IBM Corporation, P.O. Box 12195, 3039 Cornwallis Road, Research Triangle Park, NC 27709-2195, USA. Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement.

This document is not intended for production use and is furnished as is without any warranty of any kind, and all warranties are hereby disclaimed including the warranties of merchantability and fitness for a particular purpose.

IBM may change this publication, the product described herein, or both. These changes will be incorporated in new editions of the publication.

This publication contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.



---

## Trademarks

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

AIX®	NetView®
C/370™	OpenEdition®
C Set ++®	Operating System/2®
DB2®	OS/2®
DB2 Universal Database®	SOM®
IBM®	SOMobjects™
MVS™	TeamConnection™
MVS/ESA™	VisualAge®
MVS/XA™	XGA

Lotus and Lotus Notes are registered trademarks and Domino is a trademark of Lotus Development Corporation.

Tivoli, Tivoli Management Environment, and TME 10 are trademarks of Tivoli Systems Inc. in the United States and/or other countries.

The following terms are trademarks of other companies:

HP-UX 9.\*, 10.0 and 10.01 for HP 9000 Series 700 and 800 computers are X/Open Company UNIX 93 branded products. HP-UX 10.10 and 10.20 for HP 9000 Series 700 and 800 computers are X/Open Company UNIX 95 branded products.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Intel and Pentium are registered trademarks of Intel Corporation.

Microsoft, Windows, Windows NT and the Windows logo are registered trademarks of Microsoft Corporation.

Java, HotJava, Network File System, NFS, Solaris and the Sun logo are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

Netscape Navigator is a U.S. trademark of Netscape Communications Corporation.

Adobe, the Adobe logo, Acrobat, the Acrobat logo, Acrobat Reader, and PostScript are trademarks of Adobe Systems Incorporated.

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



---

## About this book

This book is part of the documentation library supporting the IBM VisualAge TeamConnection Enterprise Server licensed programs. It is written for persons who need to perform the following tasks:

- Set up TeamConnection databases and administer TeamConnection families.
- Create and administer configurable fields and user exits.
- Maintain TeamConnection databases or migrate a database to TeamConnection version 3.

This book is available in PDF format. Because production time for printed manuals is longer than production time for PDF files, the PDF files may contain more up-to-date information. The PDF files are located in directory path nls\doc\enu (Intel) or softpubs/en\_US (UNIX). To view these files, you need a PDF reader such as Acrobat.

Before using this book, refer to the *TeamConnection Installation Guide*, available in softcopy format on the installation CD. The *Installation Guide* contains instructions for installing a TeamConnection server. The user who needs to install only a TeamConnection client should follow the installation instructions in *Getting Started with the TeamConnection Clients*.

IBM VisualAge TeamConnection Enterprise Server uses DB2 Universal Database, Enterprise Edition, version 5. Refer to the bibliography at the back of this book for a list of publications you can use to install and administer your DB2 database system.

**Note:** It is not recommended that you make changes to your database by issuing INSERT, UPDATE, or DELETE statements or by changing or deleting database tables or the columns defined in TeamConnection database tables. Changing your database in these ways, through the DB2 administrator tools, the DB2 command line processor, the TeamConnection migration tools, or the tcupdb tool can corrupt your TeamConnection database. Any such changes are made at your own risk. Please contact your IBM representative for information on the terms of IBM customer support.

---

## How this book is organized

This book contains the following sections:

**“Part 1. Introducing IBM VisualAge TeamConnection Enterprise Server” on page 1** presents an overview of the IBM VisualAge TeamConnection Enterprise Server product. The information in this section should be read and understood by everyone who is going to work with TeamConnection.

**“Part 2. Designing and creating your TeamConnection environment” on page 27** is intended for the family administrator who needs to plan for how the IBM VisualAge TeamConnection Enterprise Server product is going to be used in the company’s development environment. After the planning stage, the family

administrator will use this section to learn how to do TeamConnection administrative tasks. Before reading this section, you should be familiar with the TeamConnection terminology and concepts presented in “Part 1. Introducing IBM VisualAge TeamConnection Enterprise Server” on page 1.

“Part 3. Maintaining the TeamConnection server” on page 125 contains information on maintaining your TeamConnection database, monitoring family use, and migrating your family database to TeamConnection version 3.

This book also contains several appendixes providing more information for performing TeamConnection administrative tasks and worksheets that can help you plan your TeamConnection families.

Information on customer service, a bibliography, and a glossary are included at the back of this book.

---

## Conventions

This book uses the following highlighting conventions:

- *Italics* are used to indicate the first occurrence of a word or phrase that is defined in the glossary. They are also used for information that you must replace.
- **Bold** is used to indicate items on the GUI.
- Monospace font is used to indicate exactly how you type the information.
- File names follow Intel conventions: **mydir\myfile.txt**. AIX, HP-UX, and Solaris users should render this file name **mydir/myfile.txt**.

Tips or platform specific information is marked in this book as follows:



Shortcut techniques and other tips



IBM VisualAge TeamConnection Enterprise Server for OS/2



IBM VisualAge TeamConnection Enterprise Server for Windows/NT



IBM VisualAge TeamConnection Enterprise Server for Windows 95



IBM VisualAge TeamConnection Enterprise Server for AIX



IBM VisualAge TeamConnection Enterprise Server for HP-UX



IBM VisualAge TeamConnection Enterprise Server for Solaris

---

## Tell us what you think

In the back of this book is a comment form. Please take a few moments to tell us what you think about this book. The only way for us to know if you are satisfied with our books or if we can improve their quality is through feedback from customers like you.



---

## Part 1. Introducing IBM VisualAge TeamConnection Enterprise Server



---

## Chapter 1. An introduction to TeamConnection

TeamConnection provides an environment and tools to make software development run smoothly, whether your development team is small or large. Using TeamConnection, you can communicate with and share data among team members to keep up with the many tasks in the development life cycle, from planning through maintenance.

What does TeamConnection do for you? It takes care of the following:

- *Configuration management*: the process of identifying, organizing, managing, and controlling software modules as they change over time. This includes controlling access to your software modules and providing notification to team members as software modules change.
- *Release management*: the logical organization of objects that are related to an application. The release provides a logical view of objects that must be built, tested, and distributed together. Releases are versioned, built, and packaged.
- *Version control*: the tracking of relationships among the versions of the various parts that make up an application. Version control enables you to build your product using stable levels of code, even if the code is constantly changing. It provides control over which changes are available to everyone and, optionally, allows more than one developer at a time to update a part.
- *Change control*: the controlling of changes to parts that are stored in TeamConnection. TeamConnection keeps track of any part changes you make and the reasons you make them. Your development team can build releases with accuracy and efficiency, even as the parts evolve. The product ensures that the change process is followed and that the changes are authorized. After changes are made, it allows you to integrate the changes and build the application. TeamConnection tracks all changes to the parts across multiple products and environments.

The *change control process* is configurable. Your team can decide how strict the change control should be, from loose to very tight. You can also adjust the level of control as you move through a development cycle.

- *Build support*: the function that enables you to define the structure of your application and then to create it within TeamConnection from your input parts. Independent steps in a build can run in parallel on different servers, thus reducing your build time. You can build applications for platforms in addition to the one TeamConnection runs on—currently, you can use TeamConnection to build applications on AIX, HP-UX, OS/2, Windows NT, Windows 95, Solaris, MVS, and MVS OpenEdition.
- *Packaging support*: the preparation of your application for electronic distribution to other users.

This chapter defines the basic terms and concepts you need to make the most of TeamConnection. Read this chapter first; then decide which information you need next:

Topic and description	Page
Designing and creating your TeamConnection environment: <ul style="list-style-type: none"> <li>• Planning for and creating families</li> <li>• Preparing for users</li> <li>• Configuring fields</li> <li>• Configuring <i>processes</i></li> <li>• Providing user exits</li> </ul>	27
Maintaining TeamConnection: <ul style="list-style-type: none"> <li>• Setting up the mail facility</li> <li>• Changing the age of <i>defects</i> and <i>features</i></li> <li>• Resolving TeamConnection errors</li> <li>• Maintaining the database</li> </ul>	125

---

## TeamConnection definitions

The following definitions are in logical order rather than alphabetical. The *User's Guide* provides additional information about these terms.

### TeamConnection's client/server architecture

Figure 1 is an example of a network of TeamConnection clients and servers.

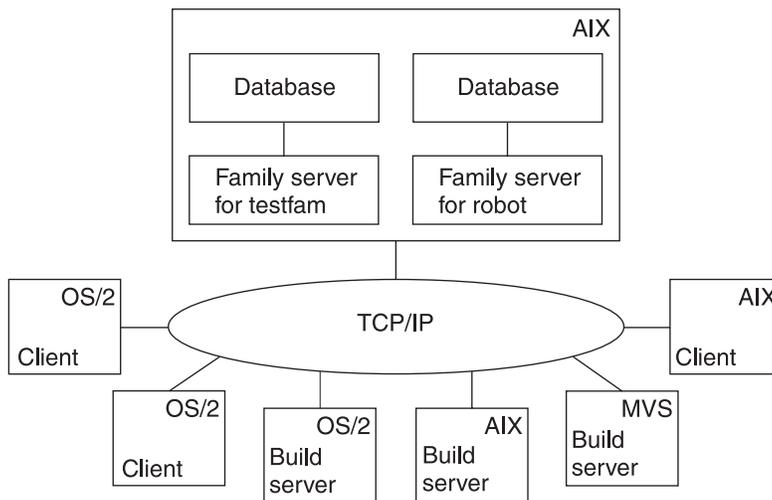


Figure 1. A sample TeamConnection client/server network

TeamConnection *family servers* control all data within the TeamConnection environment. Data stored in a family server's database includes:

- Text objects, such as source code and product documentation
- Binary objects, such as compiled code
- Modeled objects that are stored in the information model by tools such as VisualAge Generator
- Other TeamConnection objects that are metadata about the other objects

A TeamConnection *client* gives team members access to the development information and parts stored on the database server.

## TeamConnection database

TeamConnection is built on IBM's DB2 Universal Database. Please refer to the DB2 documentation referenced in this document's "Bibliography" on page 267 for detailed information on DB2 database configuration, administration, and utilities.

## Interfaces

TeamConnection provides the following interfaces that you can use to access data:

- A graphical user interface based on industry standards.
- A command line interface that lets you type TeamConnection commands from a prompt or from within TeamConnection
- A web client that you access through your web browser.

You can use any interface to do your TeamConnection work, or you can switch among them.

## Families

A *family* represents a complete and self-contained collection of TeamConnection users and development data. Data within a family is completely isolated from data in all other families. One family cannot share data with another.

See "Chapter 5. Setting up your family structure" on page 47 for more information about families.

## Users and host lists

Users are given access to the TeamConnection development data in a specific family through their *user IDs*. Each family has at least one *superuser*, who has privileged access to the family. The superuser gives other users the *authority* to perform some set of *actions* on particular data. Depending on the authority granted to a user, that user might in turn be able to grant some equal or lesser level of authority to other users. However, the ability to grant authority for some actions is reserved to the superuser. There are no actions which the superuser cannot perform.

For host-based authentication, each user ID is associated with a *host list*, which is a list of client machine addresses from which the user can access TeamConnection when using that ID.

A single user can access TeamConnection from multiple systems or logins. Likewise, a single system login can act on behalf of multiple users. The set of authorized logins for a TeamConnection user ID makes up the user's host list.

It is also possible to authenticate users through the use of passwords, either in place of host lists, or as an alternative form of authentication.

See "Chapter 6. Preparing for your users" on page 63 for more information.

## Parts

TeamConnection *parts* are objects that users and tools store in TeamConnection. They include text objects, binary objects, and modeled objects. These parts can be stored by the user or the tool, or they can be generated from other parts, such as when a linker generates an executable file. Parts can also be groupings of other TeamConnection objects for building and distribution, or simply for convenient reference. Common part actions include the following:

**Create** To store a part from your workstation on the server; from that time on, TeamConnection keeps track of all changes made to the part. Or, to create a part to use as a place holder to store the output of a build.

**Check out** To get a copy of a part so that you can make changes to it.

**Check in** To put the changed part back into TeamConnection.

**Extract** To get a copy of the part *without* making changes to the *current version* in TeamConnection.

**Edit** To change a part from within TeamConnection using a specified editor.

**Build** To construct an output part from parts that you have defined to TeamConnection as input to the output part.

These are simplified definitions of part actions; there is more about the actions you can perform against parts in the *Commands Reference*.

The current version of each part is stored in the TeamConnection database, along with previous versions of each part. You can return to previous versions if you need to.

## Components

Within each family, development data is organized into groups called *components*. The component hierarchy of each family includes a single top component, called *root*, and *descendants* of that root. Each *child component* has at least one parent component; a child can have multiple parents.

The following figure depicts a component hierarchy.

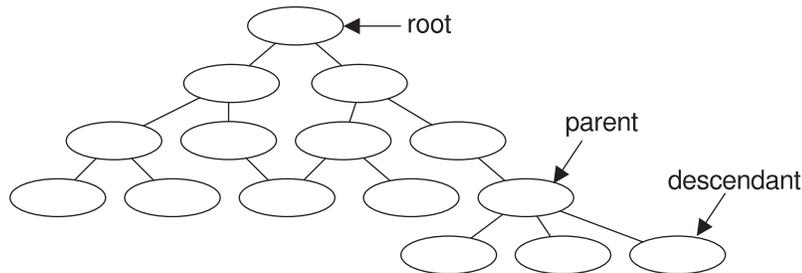


Figure 2. Sample of a component hierarchy

TeamConnection uses components to organize development data, control access to the data, and notify users when certain actions occur. Descendant components inherit access and notification information from ancestor components. Information about the components is stored in the database, including:

- The component's position in its family hierarchy.
- The user who owns the component. The component *owner* is responsible for managing data related to it, including defects or features.
- The users who have access to the component and the level of access each user has. This information makes up the component's *access list*.
- The users who are to be notified about changes to the component. This set of users is called the *notification list*.
- The *process* by which the component handles defects and features.

See "Planning your components" on page 47 for more information about components.

## Releases

An application is likely to contain parts from more than one component. Because you probably want to use some of the same parts in more than one application, or in more than one version of an application, TeamConnection also groups parts into *releases*. A release is a logical organization of all parts that are related to an application; that is, all parts that must be built, tested, and distributed together. Each time a release is changed, a new version of the release is created. Each version of the release points to the correct version of each part in the release.

Each part in TeamConnection is managed by at least one component and contained in at least one release. One release can contain parts from many components; a component can span several releases. Figure 3 shows the relationships between parts, the releases that contain them, and the components that manage them.

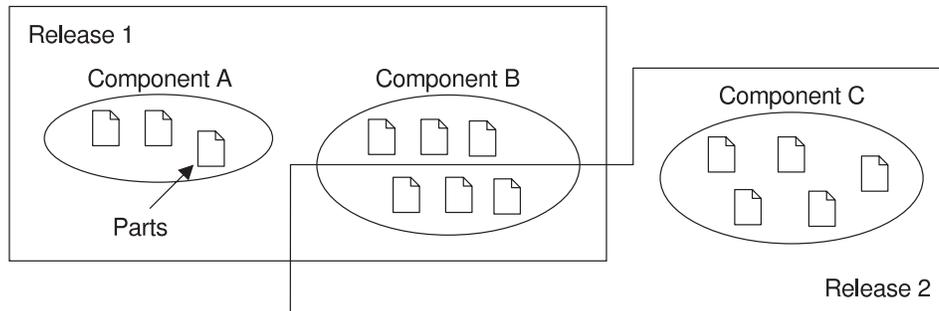


Figure 3. Parts, releases, and components

Each time a new development cycle begins, you can define a separate release. Each subsequent release of an application can share many of the same parts as its predecessor. Thus maintenance of an older release can progress at the same time as development of a newer one. Each release follows a process by which defects and features are handled.

See “Planning your releases” on page 50 for more information about releases.

## Work areas

A release contains the latest “official” version of each of its parts. As users check parts out of the releases, update them, and then check them back in, TeamConnection keeps track of all of these changes, even when more than one user updates the same part at the same time. To make this possible, TeamConnection uses something called a *work area*.

A work area is a logical temporary work space that enables you to isolate your work on the parts in a release from the official versions of the parts. You can check parts out to a work area, update them, and build them without affecting the official version of the parts in the release. After you are certain that your changes work, you *integrate* the work area with the release (or *commit* the driver that the work area is a member of, if you are using the driver subprocess). The integration makes the parts from your work area the new official parts in the release.

You can do the following with work areas:

- Check out parts from a release
- Update any or all of the checked-out parts

- Get the latest copies of the parts in the release, including any changes integrated by other users
- Get the latest copies of the parts in another work area
- *Freeze* the work area, making a snapshot of the parts as they exist at a particular instant in case you need to return to it later
- Build the parts in the work area
- Move all parts back into the release by integrating the work area

## Drivers

A driver is a collector for work areas. You create drivers associated with specific releases so that you can exercise greater control over which work areas are integrated into the release and commit the changes from multiple work areas simultaneously.

When a work area is added to a driver, it is called a *driver member*. A single work area can be a member of more than one driver. By making a work area part of a driver, you associate the parts changed in relation to that work area with the specified driver. These parts must be members of the release associated with the driver.

Drivers enable you to place the following controls over work area integrations:

- Define and monitor prerequisite and corequisite work areas to ensure that mutually dependent changes are integrated in proper order.
- Monitor and resolve conflicting changes to the same part (if you use concurrent development).
- Restrict access to driver members so that they can be changed only by users with proper authority.

## Defects and features

A defect is a record of a problem to be fixed. A feature is a record of a request for a functional addition or enhancement. Both may be associated with a work area, and both follow the processes defined for the component and release that are associated with the work area. TeamConnection tracks both objects through their life cycles as developers change and commit parts.

You can use defects and features to record problems and design changes for things other than the products you are developing under TeamConnection control. For example, you can use defects to record information about personnel problems, hardware problems, or process problems. You can use features to record proposals for process improvements and hardware design changes.

## Processes

An application changes over time as developers add features or correct defects. TeamConnection controls these changes according to the *processes* you choose for

your application's components and releases. A process enforces a specific level of control to part changes and ensures that actions occur in a specified order.

Two separate types of processes are defined: component processes, which can be different for each component within a family, and release processes, which apply to all activities associated with a given release. Component or release processes are built from a number of lower-level processes, or *subprocesses*, that are included with the TeamConnection product.

A defect or feature written against a component moves through successive *states* during its life cycle. The TeamConnection actions that you can perform against it depend on its current state. The component processes define these actions. You can require users to do some, all, or none of the following for tracking defects and features:

**dsrFeature**

Design, size, and review changes to be made for features

**verifyFeature**

Verify that the features have been implemented correctly

**dsrDefect**

Design, size, and review fixes to be made for defects

**verifyDefect**

Verify that the fixes work

At the release level you can require some, all, or none of the following subprocesses:

**track** This subprocess is TeamConnection's way of relating all part changes to a specific defect or feature and a specific release. Each work area gathers all the parts modified for the specified defect or feature in one release and records the status of the defect or feature. The work area moves through successive states during its life cycle. The TeamConnection actions that you can perform against a work area depend on its current state.

You must use the *track subprocess* if you want to use any of the other release subprocesses.

**approval**

This subprocess ensures that a designated *approver* agrees with the decision to incorporate changes into a particular release and electronically signs a record. As soon as approval is given, the changes can be made.

**fix** This subprocess ensures that as users check in parts associated with a work area, an action is taken to indicate that they have completed their portion. When everyone is done, the owner of the *fix record* (usually the component owner) can change the fix record to complete. The parts are then ready for integration.

**driver** A *driver* is a collection of all the work areas that are to be integrated with each other and with the unchanged parts in the release at a particular time. The driver subprocess allows you to include these changes incrementally so that

their impact can be evaluated and verified before additional changes are incorporated. Each work area that is included in a driver is called a *driver member*.

**test** The test subprocess guarantees that testing occurs prior to verifying that the fix is correct within the release.

TeamConnection is shipped with several predefined processes. If these do not apply to your organization, you can configure your own processes by defining different combinations of subprocesses.

See “Planning your processes” on page 53 for instructions for setting and changing processes.

## Build

The TeamConnection build function automates the process of building individual parts or entire applications, both in the work group LAN environment and on an enterprise server. This function enables you to reliably and repeatedly build the same output from the same inputs. You can also build different outputs from the same inputs for different environments.

You start a build against an output part that has an associated *builder*. A builder is an object that describes how to translate input parts to get the desired output, such as a linker or compiler. An input part might have an associated parser, which determines the dependencies for the input parts in a build.

The build function does the following:

- Tracks build times of inputs and outputs so that it builds only those parts that are out of date themselves or that have out of date dependents. You can also force a build regardless of the build times.
- Enables you to spread the build over multiple machines running at the same time or into multiple processes running on a single machine, such as on MVS.

## Packaging

*Packaging* is any of the steps necessary to distribute software and data onto the machines where they are to be used. TeamConnection includes two tools that you can use to automate the electronic distribution of TeamConnection-managed software and data:

**Gather** An automated data mover for server or file transfer-based distribution.

**Tivoli Software Distribution**

A bridge utility that automates the installation and distribution of software or data using Tivoli as the distribution vehicle.

For more information, refer to the *User's Guide*.

---

## Hardware and software requirements

The following sections list the hardware and software requirements for each platform supported by TeamConnection. Each section contains the following tables:

- Server hardware requirements
- Client hardware requirements
- Software requirements

The last section contains requirements for the MVS build server.

### Requirements for TeamConnection for AIX

*Table 1. Hardware Requirements for an AIX TeamConnection Server*

**Family server**

- **Processor:** IBM RS/6000 with PowerPC architecture (recommended), such as 43P. PowerServer architecture can be used.
  - **Pointing device:** A mouse or other pointing device
  - **Monitor:** Any X11 graphics display supported by the RS/6000 workstation
  - **Memory:** 128 MB minimum (256 MB recommended); more may be needed according to number of users and database size
  - **Disk space:**
    - 500 MB for operating system and prerequisites
    - 200 MB in the target file system for user data (1+ GB recommended)
    - 160 MB for TeamConnection server code
    - 65 MB for TeamConnection documentation
    - 200 MB for DB2
    - Amount recommended by operating system for swapper space
- Note:** Older buses, I/O controllers, and hard drives may not provide adequate I/O rates for a DB2 application such as TeamConnection to demonstrate minimally acceptable performance.
- **Communications support:** Any token-ring or Ethernet local area network (LAN) adapter card that supports TCP/IP and is supported by the workstation
  - **CD-ROM drive:** A CD-ROM drive (internal or external) for installation of the product

*Table 1. Hardware Requirements for an AIX TeamConnection Server (continued)*

**Build server**

- **Processor:** IBM RS/6000 with PowerPC architecture (recommended), such as 43P. PowerServer architecture can be used.
- **Pointing device:** A mouse or other pointing device
- **Monitor:** Any X11 graphics display supported by the RS/6000 workstation
- **Memory:** 64 MB memory minimum (128 MB recommended); more may be needed according to the compilers and linkers used
- **Disk space:**
  - 500 MB for operating system and prerequisites
  - 60 MB for TeamConnection server code
  - Amount recommended by operating system for swapper space
- **Communications support:** Any token-ring or Ethernet local area network (LAN) adapter card that supports TCP/IP and is supported by the workstation
- **CD-ROM drive:** A CD-ROM drive (internal or external) for installation of the product

*Table 2. Hardware Requirements for an AIX TeamConnection Client*

**Client**

- **Processor:** IBM RS/6000 with PowerPC architecture (recommended), such as 43P. PowerServer architecture can be used.
- **Pointing device:** A mouse or other pointing device.
- **Monitor:** Any X11 graphics display supported by the RS/6000 workstation.
- **Memory:** 64 MB memory minimum.
- **Disk space:**
  - 300 MB for operating system and prerequisites
  - 60 MB for TeamConnection client code
  - Amount recommended by operating system for swapper space
- **Communications support:** Any token-ring or Ethernet local area network (LAN) adapter card that supports TCP/IP and is supported by the workstation.
- **CD-ROM drive:** A CD-ROM drive (internal or external) for installation of the product.

*Table 3. Software Requirements for TeamConnection for AIX*

**Server**

- AIX version 4.2.1 or higher version that includes TCP/IP
- Java Development Kit 1.1 (1.1.2 recommended; you can also use 1.1.4 with the April 30, 1998 fixes applied ["JDK 1.1.4 IBM build a114-19980430"]
- A Web browser such as Netscape Navigator to display the help panels for the GUI

Table 3. Software Requirements for TeamConnection for AIX (continued)

<b>Client</b>	<ul style="list-style-type: none"> <li>• AIX version 4.2.1 or higher version that includes TCP/IP</li> <li>• Java Development Kit 1.1 (1.1.2 or higher recommended)</li> <li>• A Web browser such as Netscape Navigator to display the help panels for the GUI</li> </ul>
<b>Build server</b>	<ul style="list-style-type: none"> <li>• AIX version 4.2.1 or higher version that includes TCP/IP</li> </ul>
<b>For distribution function</b>	<ul style="list-style-type: none"> <li>• TME 10 Software Distribution Version 3.1. Revision A (LK2T-6047-03)</li> <li>• TME 10 Framework Upgrade to Version 3.1 Revision C (LK2t-6073-02)</li> </ul>
<b>For bridge between VisualAge Smalltalk and VisualAge TeamConnection</b>	<ul style="list-style-type: none"> <li>• TeamConnection client</li> <li>• VisualAge Smalltalk Version 4.0 (4231060)</li> </ul>

## Requirements for TeamConnection for HP-UX

Table 4. Hardware Requirements for an HP-UX TeamConnection Server

<b>Family server</b>	<ul style="list-style-type: none"> <li>• <b>Processor:</b> Any HP 9000 Series 700 or 800 workstation</li> <li>• <b>Pointing device:</b> A mouse or other pointing device</li> <li>• <b>Monitor:</b> Any X11 graphics display supported by the processor</li> <li>• <b>Memory:</b> 128 MB minimum (256 MB recommended); more may be needed according to number of users and database size</li> <li>• <b>Disk space:</b> <ul style="list-style-type: none"> <li>– 500 MB for operating system and prerequisites</li> <li>– 200 MB in the target file system for user data (1+ GB recommended)</li> <li>– 160 MB for TeamConnection server code</li> <li>– 200 MB for DB2</li> <li>– Amount recommended by operating system for swapper space</li> </ul> </li> </ul> <p><b>Note:</b> Older buses, I/O controllers, and hard drives may not provide adequate I/O rates for a DB2 application such as TeamConnection to demonstrate minimally acceptable performance.</p> <ul style="list-style-type: none"> <li>• <b>Communications support:</b> Any token-ring or Ethernet local area network (LAN) adapter card that supports TCP/IP and is supported by the workstation</li> <li>• <b>CD-ROM drive:</b> A CD-ROM drive (internal or external) for installation of the product</li> </ul>
----------------------	---

Table 4. Hardware Requirements for an HP-UX TeamConnection Server (continued)

**Build server**

- **Processor:** Any HP 9000 Series 700 or 800 workstation
- **Pointing device:** A mouse or other pointing device
- **Monitor:** Any X11 graphics display supported by the processor
- **Memory:** 64 MB memory minimum (128 MB recommended); more may be needed according to the compilers and linkers used
- **Disk space:**
  - 500 MB for operating system and prerequisites
  - 60 MB for TeamConnection build server code
  - Amount recommended by operating system for swapper space
- **Communications support:** Any token-ring or Ethernet local area network (LAN) adapter card that supports TCP/IP and is supported by the workstation
- **CD-ROM drive:** A CD-ROM drive (internal or external) for installation of the product

Table 5. Hardware Requirements for an HP-UX TeamConnection Client

**Client**

- **Processor:** Any HP 9000 Series 700 or 800 workstation.
- **Pointing device:** A mouse or other pointing device.
- **Monitor:** Any X11 graphics display supported by the processor.
- **Memory:** 64 MB memory minimum.
- **Disk space:**
  - 300 MB for operating system and prerequisites
  - 60 MB for TeamConnection client code
  - Amount recommended by operating system for swapper space
- **Communications support:** Any token-ring or Ethernet local area network (LAN) adapter card that supports TCP/IP and is supported by the workstation.
- **CD-ROM drive:** A CD-ROM drive (internal or external) for installation of the product.

Table 6. Software Requirements for TeamConnection for HP-UX

**Server**

- HP-UX version 10.20, which includes TCP/IP  
The following patches are required for DB2:
  - PHSS\_10556
  - PHSS\_10436
- Java Development Kit 1.1.3
- A Web browser such as Netscape Navigator to display the help panels for the GUI

---

**Client**

- HP-UX version 10.20, which includes TCP/IP
- Java Development Kit 1.1.3
- A Web browser such as Netscape Navigator to display the help panels for the GUI

Table 6. Software Requirements for TeamConnection for HP-UX (continued)

<b>Build server</b>	HP-UX version 10.20, which includes TCP/IP
<b>For distribution function</b>	<ul style="list-style-type: none"> <li>• TME 10 Software Distribution Version 3.1. Revision A (LK2T-6047-03)</li> <li>• TME 10 Framework Upgrade to Version 3.1 Revision C (LK2t-6073-02)</li> </ul>
<b>For bridge between VisualAge Smalltalk and VisualAge TeamConnection</b>	<ul style="list-style-type: none"> <li>• TeamConnection client</li> <li>• VisualAge Smalltalk Version 4.0 (4231060)</li> </ul>

## Requirements for TeamConnection for Solaris

Table 7. Hardware Requirements for a Solaris TeamConnection Server

<b>Family server</b>	<ul style="list-style-type: none"> <li>• <b>Processor:</b> SPARC or UltraSPARC workstation.</li> <li>• <b>Pointing device:</b> A mouse or other pointing device</li> <li>• <b>Monitor:</b> Any X11 graphics display supported by the processor</li> <li>• <b>Memory:</b> 128 MB memory minimum (256 MB recommended); more may be needed according to number of users and database size</li> <li>• <b>Disk space:</b> <ul style="list-style-type: none"> <li>– 500 MB for operating system and prerequisites</li> <li>– 200 MB in the target file system for user data (1+ GB recommended)</li> <li>– 200 MB for TeamConnection server code</li> <li>– 65 MB for TeamConnection documentation</li> <li>– 200 MB for DB2</li> <li>– 100 MB for temporary space</li> <li>– Amount recommended by operating system for swapper space</li> </ul> </li> </ul> <p><b>Note:</b> Older buses, I/O controllers, and hard drives may not provide adequate I/O rates for a DB2 application such as TeamConnection to demonstrate minimally acceptable performance.</p> <ul style="list-style-type: none"> <li>• <b>Communications support:</b> Any token-ring or Ethernet local area network (LAN) adapter card that supports TCP/IP and is supported by the workstation</li> <li>• <b>CD-ROM drive:</b> A CD-ROM drive (internal or external) for installation of the product</li> </ul>
----------------------	---

*Table 7. Hardware Requirements for a Solaris TeamConnection Server (continued)*

**Build server**

- **Processor:** SPARC or UltraSPARC workstation.
- **Pointing device:** A mouse or other pointing device
- **Monitor:** Any X11 graphics display supported by the processor
- **Memory:** 128 MB memory minimum (256 MB recommended); more may be needed according to the compilers and linkers used
- **Disk space:**
  - 500 MB for operating system and prerequisites
  - 200 MB for TeamConnection server code
  - 100 MB for temporary space
  - Amount recommended by operating system for swapper space
- **Communications support:** Any token-ring or Ethernet local area network (LAN) adapter card that supports TCP/IP and is supported by the workstation
- **CD-ROM drive:** A CD-ROM drive (internal or external) for installation of the product

*Table 8. Hardware Requirements for a Solaris TeamConnection Client*

**Client**

- **Processor:** SPARC and UltraSPARC workstation.
- **Pointing device:** A mouse or other pointing device.
- **Monitor:** Any X11 graphics display supported by the processor.
- **Memory:** 64 MB memory minimum.
- **Disk space:**
  - 300 MB for operating system and prerequisites
  - 60 MB for TeamConnection client code
  - Amount recommended by operating system for swapper space
- **Communications support:** Any token-ring or Ethernet local area network (LAN) adapter card that supports TCP/IP and is supported by the workstation.
- **CD-ROM drive:** A CD-ROM drive (internal or external) for installation of the product.

*Table 9. Software Requirements for TeamConnection for Solaris*

**Server**

- Sun Solaris 2.5.1 which includes TCP/IP  
The following patches are required:
  - 103663-11
  - 103600-13
  - 103640-08
- **Note:** Contact your Solaris service support representative for information on obtaining and applying these patches.
- Java Development Kit 1.1 (1.1.5 or higher recommended)
- A Web browser such as Netscape Navigator to display the help panels for the GUI

Table 9. Software Requirements for TeamConnection for Solaris (continued)

<b>Client</b>	<ul style="list-style-type: none"> <li>• Sun Solaris 2.5.1 which includes TCP/IP</li> <li>• Java Runtime Environment 1.1 (1.1.5 or higher recommended)</li> <li>• A Web browser such as Netscape Navigator to display the help panels for the GUI</li> </ul>
<b>Build server</b>	Solaris 2.5.1 which includes TCP/IP.
<b>For distribution function</b>	<ul style="list-style-type: none"> <li>• TME 10 Software Distribution Version 3.1. Revision A (LK2T-6047-03)</li> <li>• TME 10 Framework Upgrade to Version 3.1 Revision C (LK2t-6073-02)</li> </ul>
<b>For bridge between VisualAge Smalltalk and VisualAge TeamConnection</b>	<ul style="list-style-type: none"> <li>• TeamConnection client</li> <li>• VisualAge Smalltalk Version 4.0 (4231060)</li> </ul>

## Requirements for TeamConnection for OS/2

Table 10. Hardware Requirements for an OS/2 TeamConnection Server

<b>Family server</b>	<ul style="list-style-type: none"> <li>• <b>Processor:</b> 133 MHz Pentium-based processor or higher</li> <li>• <b>Pointing device:</b> A mouse or other pointing device</li> <li>• <b>Monitor:</b> VGA or higher resolution with the appropriate adapter</li> <li>• <b>Memory:</b> 64 MB memory minimum; more may be needed according to number of users and database size</li> <li>• <b>Disk space:</b> <ul style="list-style-type: none"> <li>– 200 MB for operating system and prerequisites</li> <li>– 200 MB for user data (1+ GB recommended)</li> <li>– 60 MB for TeamConnection server code</li> <li>– 200 MB for DB2</li> <li>– 128 MB for swapper space (150+ MB recommended)</li> </ul> </li> </ul> <p><b>Note:</b> Older buses, I/O controllers, and hard drives may not provide adequate I/O rates for a DB2 application such as TeamConnection to demonstrate minimally acceptable performance.</p> <ul style="list-style-type: none"> <li>• <b>Communications support:</b> Network card supported by TCP/IP for OS/2</li> <li>• <b>CD-ROM drive:</b> A CD-ROM drive (internal or external) for installation of the product</li> </ul>
----------------------	---

*Table 10. Hardware Requirements for an OS/2 TeamConnection Server (continued)*

- Build server**
- **Processor:** 133 MHz 486-based processor or higher
  - **Pointing device:** A mouse or other pointing device
  - **Monitor:** VGA or higher resolution with the appropriate adapter
  - **Memory:** 32 MB memory minimum; more may be needed according to compilers and linkers used
  - **Disk space:**
    - 75 MB for operating system and prerequisites
    - 15 MB for TeamConnection build server code
    - 45 MB for swapper space
  - **Communications support:** Network card supported by TCP/IP for OS/2
  - **CD-ROM drive:** A CD-ROM drive (internal or external) for installation of the product

*Table 11. Hardware Requirements for an OS/2 TeamConnection Client*

- Client**
- **Processor:** 66 MHz 486-based processor or higher
  - **Pointing device:** A mouse or other pointing device
  - **Monitor:** VGA or higher resolution with the appropriate adapter
  - **Memory:** 16 MB minimum
  - **Disk space:**
    - 75 MB for operating system and prerequisites
    - 25 MB for TeamConnection client code
    - 32 MB for swapper space
  - **Communications support:** Network card supported by TCP/IP for OS/2
  - **CD-ROM drive:** A CD-ROM drive (internal or external) for installation of the product

*Table 12. Software Requirements for TeamConnection for OS/2*

- Server**
- One of the following:
    - OS/2 Warp Version 4 (84H1426) and IBM TCP/IP Version 3.0 for OS/2 Warp (33H9749)
    - OS/2 Warp Server Version 4 (25H8002)
    - OS/2 Warp Server Advanced Version 4 (25H8030)
  - Java Development Kit 1.1 (1.1.4 recommended with the 1998 fixes applied ["JDK 1.1.4 IBM build o114-19980304"])
  - A Web browser such as Netscape Navigator to display the help panels for the GUI

Table 12. Software Requirements for TeamConnection for OS/2 (continued)

<b>Client</b>	<ul style="list-style-type: none"><li>• Java Development Kit 1.1 (1.1.4 recommended)</li><li>• A Web browser such as Netscape Navigator to display the help panels for the Web client and TeamConnection Merge</li><li>• One of the following:<ul style="list-style-type: none"><li>– OS/2 Warp Version 4 (84H1426)<ul style="list-style-type: none"><li>- IBM TCP/IP Version 3.0 for OS/2 Warp (33H9749)</li></ul></li><li>– OS/2 Warp Server Version 4 (25H8002)</li><li>– OS/2 Warp Server Advanced Version 4 (25H8030)</li></ul></li></ul>
<b>Build server</b>	<hr/> <p>One of the following:</p> <ul style="list-style-type: none"><li>• OS/2 Warp Version 4 (84H1426)<ul style="list-style-type: none"><li>– IBM TCP/IP Version 3.0 for OS/2 Warp (33H9749)</li></ul></li><li>• OS/2 Warp Server Version 4 (25H8002)</li><li>• OS/2 Warp Server Advanced Version 4 (25H8030)</li></ul> <hr/>
<b>For distribution function</b>	<ul style="list-style-type: none"><li>• TME 10 Software Distribution Version 3.1. Revision A (LK2T-6047-03)</li><li>• TME 10 Framework Upgrade to Version 3.1 Revision C (LK2t-6073-02)</li></ul> <hr/>
<b>For bridge between VisualAge Smalltalk and VisualAge TeamConnection</b>	<ul style="list-style-type: none"><li>• TeamConnection client</li><li>• VisualAge Smalltalk Version 4.0 (4231060)</li></ul>

## Requirements for TeamConnection for Windows

Table 13. Hardware Requirements for a Windows TeamConnection Server

### Family server

- **Processor:** 133 MHz Pentium-based processor or higher (however, the PowerPC is not supported)
  - **Pointing device:** A mouse or other pointing device
  - **Monitor:** VGA or higher resolution with the appropriate adapter
  - **Memory:** 64 MB memory minimum; more may be needed according to number of users and database size
  - **Disk space:**
    - 200 MB for operating system and prerequisites
    - 200 MB for user data (1+ GB recommended)
    - 60 MB for TeamConnection server code
    - 200 MB for DB2
    - 128 MB for swapper space (150+ MB recommended)
- Note:** Older buses, I/O controllers, and hard drives may not provide adequate I/O rates for a DB2 application such as TeamConnection to demonstrate minimally acceptable performance.
- **Communications support:** Network card supported by TCP/IP for Windows NT
  - **CD-ROM drive:** A CD-ROM drive (internal or external) for installation of the product

---

### Build server

- **Processor:** Any processor supported by the required version of Windows, except the PowerPC
- **Pointing device:** A mouse or other pointing device
- **Monitor:** VGA or higher resolution with the appropriate adapter
- **Memory:** 32 MB memory minimum; more may be needed according to compilers and linkers used
- **Disk space:**
  - 75 MB for operating system and prerequisites
  - 15 MB for TeamConnection build server code
  - 45 MB for swapper space
- **Communications support:** Network card supported by TCP/IP for Windows NT or Windows 95
- **CD-ROM drive:** A CD-ROM drive (internal or external) for installation of the product

Table 14. Hardware Requirements for a Windows TeamConnection Client

<b>Client</b>	<ul style="list-style-type: none"> <li>• <b>Processor:</b> Any personal workstation supported by the operating system, except the PowerPC</li> <li>• <b>Pointing device:</b> A mouse or other pointing device</li> <li>• <b>Monitor:</b> VGA or higher resolution with the appropriate adapter</li> <li>• <b>Memory:</b> 16 MB minimum</li> <li>• <b>Disk space:</b> <ul style="list-style-type: none"> <li>– 75 MB for operating system and prerequisites</li> <li>– 25 MB for TeamConnection client code</li> <li>– 32 MB for swapper space</li> </ul> </li> <li>• <b>Communications support:</b> Any network card supported by the above workstation that supports TCP/IP</li> <li>• <b>CD-ROM drive:</b> A CD-ROM drive (internal or external) for installation of the product</li> </ul>
---------------	---

Table 15. Software Requirements for TeamConnection for Windows

<b>Server</b>	<ul style="list-style-type: none"> <li>• Microsoft Windows NT 4.0 which includes TCP/IP</li> <li>• Java Development Kit 1.1 (1.1.6 recommended)</li> <li>• A Web browser such as Netscape Navigator to display the help panels for the GUI</li> </ul>
<b>Client</b>	<ul style="list-style-type: none"> <li>• Java Runtime Environment 1.1 (1.1.6 recommended)</li> <li>• A Web browser such as Netscape Navigator to display the help panels for the Web client and TeamConnection Merge</li> <li>• One of the following: <ul style="list-style-type: none"> <li>– Microsoft Windows 95</li> <li>– Microsoft Windows NT 4.0, which includes TCP/IP</li> </ul> </li> </ul>
<b>Windows NT build server</b>	Microsoft Windows NT 4.0 which includes TCP/IP
<b>For distribution function</b>	<ul style="list-style-type: none"> <li>• TME 10 Software Distribution Version 3.1. Revision A (LK2T-6047-03)</li> <li>• TME 10 Framework Upgrade to Version 3.1 Revision C (LK2t-6073-02)</li> </ul>
<b>For bridge between VisualAge Smalltalk and VisualAge TeamConnection</b>	<ul style="list-style-type: none"> <li>• TeamConnection client</li> <li>• VisualAge Smalltalk Version 4.0 (4231060)</li> </ul>

## Requirements for MVS build servers

The following are software requirements for the MVS build server:

- TCP/IP Version 3.2 for MVS
- OS/390 R3 LE

---

## Chapter 2. Administrator Tasks

This chapter briefly describes the tasks that a TeamConnection administrator performs. Administrators' responsibilities vary widely according to the needs of your development environment and the size and complexity of your network. The tasks explained in this book can be performed by a single system administrator or by two or more administrators. One way to distinguish administrators' roles is by function, as follows:

### System administrator

Has *superuser* access to the family server and database administration access to the database management system. This administrator is responsible for the following:

- Installing and maintaining the TeamConnection server
- Maintaining and backing up the database used by TeamConnection

**Note:** On UNIX systems, the system administrator must also have root access to the host machine.

### Family administrator

Has *superuser* access to the family server and database administration access to the database management system. This administrator is responsible for the following:

- Planning and configuring TeamConnection for one or more families
- Managing user access to one or more families
- Maintaining one or more families
- Creating and updating configurable fields
- Configuring release and component processes for a family
- Creating and updating user exits
- Monitoring the user activity of a family

### Build administrator

This administrator is responsible for the following:

- Setting up and maintaining build servers
- Planning for builds
- Creating builders and parsers
- Starting and stopping build servers
- Defining *pools*
- Monitoring build performance
- Creating driver members
- Committing and completing drivers
- Extracting releases
- Packaging and distributing applications

## System Administrators

System administrators are responsible for installing the TeamConnection server software, creating user accounts for TeamConnection families, updating network and services configuration files for TCP/IP and socket addresses used by families and build agents, preparing the TeamConnection client software for LAN installation (if your installation plans to make the client software available over a LAN), starting and stopping the servers, and maintaining the TeamConnection databases, and performing database backups.

These responsibilities span the boundaries between TeamConnection, the operating system, and the DB2 Universal database manager. The following table will help you determine where to find instructions for performing these tasks.

Environment	Tasks	For instructions, refer to:
TeamConnection tasks	<ul style="list-style-type: none"> <li>• Installing TeamConnection</li> <li>• Preparing the TeamConnection client software for LAN installation</li> <li>• Configuring TeamConnection families</li> <li>• Starting and stopping the TeamConnection servers</li> </ul>	For installation instructions, refer to the <i>TeamConnection Installation Guide</i> . For instructions on configuring TeamConnection families and starting and stopping the servers, see "Part 2. Designing and creating your TeamConnection environment" on page 27 in this book.
Operating system tasks	<ul style="list-style-type: none"> <li>• Creating user accounts for TeamConnection families (for multiuser operating systems)</li> <li>• Updating network and services configuration files for TCP/IP address and socket port numbers used by families and build servers</li> <li>• Enabling the syslog to capture system and database messages</li> </ul>	Your operating system user's or administrator's guide.
Database manager tasks	<ul style="list-style-type: none"> <li>• Installing the DB2 Universal Database</li> <li>• Starting and stopping the database manager</li> <li>• Maintaining the database</li> <li>• Database backup and recovery</li> </ul>	For installation instructions, refer to the <i>IBM DB2 Universal Database Quick Beginnings</i> manual appropriate to your platform. For database administration, refer to the <i>IBM DB2 Universal Database Administration Guide</i> .



One particularly important function of a system administrator is maintaining the TeamConnection databases. Your TeamConnection database needs to be backed up regularly using the DB2 backup utilities available from the DB2 Control Center or the command line processor. See “Backing up the TeamConnection database” on page 137 for instructions.

**Note:** It is not recommended that you make changes to your database by issuing INSERT, UPDATE, or DELETE statements or by changing or deleting database tables or the columns defined in TeamConnection database tables. Changing your database in these ways, through the DB2 administrator tools, the DB2 command line processor, the TeamConnection migration tools, or the tcupdb tool can corrupt your TeamConnection database. Any such changes are made at your own risk. Please contact your IBM representative for information on the terms of IBM customer support.

---

## Family Administrators

If your TeamConnection environment includes more than one family, you might consider assigning one family administrator to each family. Family administrators are responsible for planning and creating the component structure and releases to be used in your family, configuring the processes to be used for the components and releases, creating user IDs and managing their access to the family, and creating configurable fields and user exits.

“Part 2. Designing and creating your TeamConnection environment” on page 27 contains instructions for completing each of these tasks.

---

## Build Administrators

TeamConnection provides build environments for most of its platforms. If you have a large and complex project, or your development efforts require you to build on multiple platforms, it may be beneficial for you to assign a build administrator for TeamConnection or for each TeamConnection family. Build administrators are responsible for installing and maintaining the build servers, configuring your build environment, creating build scripts and parsers, monitoring build performance, and customizing packaging and distribution scripts.

The *TeamConnection User's Guide* explains how to create and maintain a build environment.



---

## **Part 2. Designing and creating your TeamConnection environment**

This section is intended for the family administrator who needs to plan for how the TeamConnection product is going to be used in the company's development environment. After the planning stage, the family administrator will use this section to learn how to do TeamConnection administrative tasks.

Before reading this section, you should be familiar with the TeamConnection terminology and concepts presented in "Part 1. Introducing IBM VisualAge TeamConnection Enterprise Server" on page 1.



---

## Chapter 3. Creating your TeamConnection family

This chapter contains information to help you plan for your TeamConnection family. It also provides instructions for creating the family, or families, that your development organization will use.

---

### Planning your families

Careful planning of the families that your organization will use is an important first step in preparing to use TeamConnection. First, decide how many families you will need. The following will help you decide:

- Data cannot be shared between families, so group all development projects that share source data within the same family.

For example, you have several applications under maintenance or development, and these applications share some source code, such as a utility subroutine library. If you create a family for each application, each family must maintain a copy of the source code for the library. If you create one family for all the applications, they can share a single copy of the source code.

When looking at the data your projects share, consider not just the data they share today, but what they might be sharing in the future. If your development projects are going to remain separate, create individual families.

- You can create new families as your needs evolve over time.
- The more families you have, the more administrative work you will have.

Keep in mind that these are merely guidelines. If it is not clear whether you need one or more than one family, consider starting with one. You can always create another family later.

You must also decide on a name for the family. You want the name to uniquely identify the purpose of the family. For example, you might use your product name or an abbreviation of your product name. Another consideration is what case to use—lower, upper, or mixed. Mixed case is not recommended because it is more difficult to remember. You might want to ask your users what case they prefer. See “Database naming conventions” on page 30 for more information about database names.

---

### DB2 considerations

Before you create a TeamConnection family, you need to be aware of certain DB2 considerations concerning DB2 instances, database naming conventions, and database configuration parameters

## DB2 instances

It is recommended that the databases for each TeamConnection family be placed in a separate DB2 instance. By following this recommendation, you can assure that if an instance is stopped, only one TeamConnection family is affected. It also enables you to tune the performance for one instance while affecting only one TeamConnection family.

You need to have at least 100 MB of free disk space in the file system where the family database is to be created.



On Intel platforms, it is recommended that you create only one DB2 instance per (physical) server. Because it is recommended that you have only one family per instance, you should have only one family per (physical) server.



On UNIX platforms, use the sample .profile for each family that you create. This .profile contains DB2 environment variables that you need to customize before you create a family. One of these variables, DB2INSTANCE, defines the DB2 instance in which the family is to be created. DB2INSTANCE must be set in the profile for the new family and should point to the new DB2 instance. Use a different name for the DB2 instance and the TeamConnection family. The sample profile is located in `$TC_HOME/install/$LANG/profile.family`.

If you already have a family running on an existing DB2 instance, it is necessary to create another DB2 instance for a new family. Refer to *DB2 Quick Beginnings* for information on how to create a new DB2 instance.

## Database naming conventions

DB2 places certain restrictions on database names. Because the TeamConnection family name corresponds to the name of the DB2 database, these DB2 restrictions apply to TeamConnection families as well.

The name you specify:

- Can contain 1 to 8 characters
- Cannot be any of the following:
  - USERS
  - ADMINS
  - GUESTS
  - PUBLIC
  - LOCAL
- Cannot begin with the following:
  - IBM
  - SQL

- SYS
- Cannot include accented characters.
- To avoid potential problems, do not use the special characters @, #, and \$ in a database name if you intend to have a client remotely connect to a host database. Also, because these characters are not common to all keyboards, do not use them if you plan to use the database in another country.
- Be aware of case-sensitivity on your platform:
  - On OS/2, use uppercase names.
  - On Windows 95 and Windows NT, use any case.
  - On UNIX, use lowercase names.

## Database configuration parameters

When you create a new family, TeamConnection creates a DB2 database and sets the following values for certain database configuration parameters. Use caution when modifying the values to which TeamConnection sets these parameters.

### **APPLHEAPSZ = 1280**

This parameter defines the number of private memory pages available to be used by the database manager on behalf of a specific agent or subagent.

### **BUFFPAGE = 12000**

This parameter controls the size of a buffer pool when the CREATE BUFFERPOOL or ALTER BUFFERPOOL statement is run.

### **DBHEAP=2400**

This parameter indicates the maximum amount of space that the catalog cache can use from the database heap (dbheap). The catalog cache is used to store table descriptor information that is used when a table, view or alias is referenced during the compilation of an SQL statement.

### **DLCHKTIME = 1000**

This parameter defines the frequency at which the database manager checks for deadlocks among all the applications connected to a database.

### **LOGFILSIZ = 4000**

This parameter determines the number of pages for each of the configured logs. A page is 4KB in size.

### **LOGPRIMARY = 5**

This parameter specifies the number of primary logs that will be created.

### **LOGSECOND = 30**

This parameter specifies the number of secondary log files that are created and used for recovery log files (only as needed).

On OS/2, the following additional database parameters are set when you create a family. The value for the DBHEAP parameter is set to a different value on OS/2 than on the remaining server platforms.

**APP\_CTL\_HEAP\_SZ=128**

This parameter determines the maximum size, in 4 KB pages, for the application control shared memory. Application control heaps are allocated from this shared memory.

**CATALOGCACHE\_SZ=32**

This parameter sets the catalog cache size. The catalog cache is used to store table descriptor information that is used when a table, view or alias is referenced during the compilation of an SQL statement.

**DBHEAP=600**

This parameter indicates the maximum amount of space that the catalog cache can use from the database heap (dbheap).

**LOCKLIST=50**

This parameter indicates the amount of storage that is allocated to the lock list. There is one lock list per database and it contains the locks held by all applications concurrently connected to the database.

**MAXAPPLS=32**

This parameter specifies the maximum number of concurrent applications that can be connected (both local and remote) to a database.



---

TeamConnection leaves all other DB2 database configuration parameters at their DB2 default values.

---

## Creating a family

An initial family, called testfam, is configured during the installation of the TeamConnection server. This family usually serves as a test family so that you can verify that TeamConnection is working properly. You can use this family to explore and learn about TeamConnection. Eventually, you will want to create another family for use during application development.

If you create more than one family, TeamConnection places each in a separate directory. Each family requires its own audit log, user exit, mail queue, and security and configurable field information, and therefore cannot share a directory with another family.

Each family has its own unique database. If you create more than one family on a single machine, they use the same database manager to access the database. If you want to install families on separate machines, you need a TeamConnection and DB2 server on each machine.

At a minimum, you will provide the following information when you create your family:

- The name of the family
- The fully qualified path name of the directory where you want the family configuration information stored
- The port address of the family server
- The level of security to use for the family
- The login and client host information for the first superuser for the family

TeamConnection provides a Family Administrator GUI for creating families and for performing other family administrative tasks. You can access this GUI only from a family server machine.

The family administrator GUI is capable of working with one or more families at a time. However, it is recommended that for novice UNIX users, you should work with only one family in family administrator GUI, the family ID from where this tool is started. Then, later on, for advanced users, you can create a separate user ID just to manage the family administrator GUI to control multiple families.

To create a TeamConnection family using the Family Administrator GUI, follow these steps:

1. Before you begin, define your family name in the TCP/IP hosts file and the port number for your database in the TCP/IP services file. See the *Installation Guide* for information on setting up TCP/IP files.
2. Do one of the following to display the TeamConnection Family Administrator window:
  - In OS/2, from the TeamConnection Group folder on the desktop, double-click on the **Family Administrator** icon.
  - In Windows NT 4.0, Select **TeamConnection Family Administrator** from the **Start** menu.
  - Type `tcadmin` from a command prompt.

This command has several optional parameters:

**-hide** Starts `tcadmin`, but does not open the Family Administrator GUI. You can use this parameter in conjunction with the `-start` parameter to start a family without opening the Family Administrator GUI.

**-start *family***  
Causes the family specified to be started automatically when the Family Administrator is started.

**-f *directory***  
Specifies the location of the configuration files needed to create a new family. The default location of these files is `\nls\cfg\%LANG%` from the directory where TeamConnection is installed. If you have configured the files differently and placed them in a different directory structure, you can use this parameter to point to the files you want to use to create a new family.

**-log [logfile]**

When -log is specified, a log file will be generated that will contain information on which commands are run, the output from those commands, and error messages and stack traces for debugging purposes. If no log file name is specified, then the file tcadmin.log will be created. If a log file name is specified, the error log will be written to that log file. The log file is overwritten if the same file name is specified again.

3. Select **Create Family** from the Family pull-down menu.
4. When the family properties notebook appears, complete the required information about the family. After you have set values for the family in the properties notebook, select the **OK** push button.

**Note:** After your family is created, you can access other pages in this notebook by selecting the family and then selecting **Family** → **Properties**.

---

## Using the family properties notebook

The following sections introduce each page of the family properties notebook. Many of these settings are discussed in greater detail in later chapters of this book.

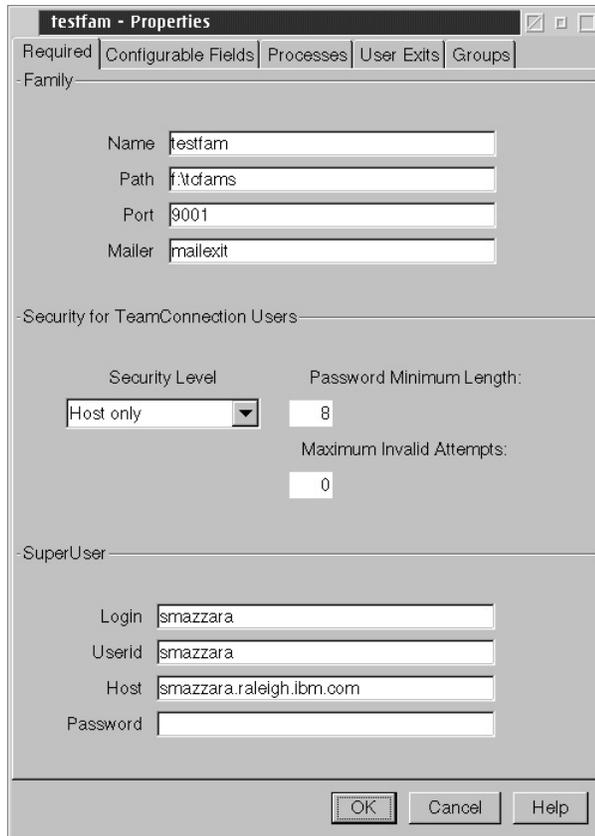


Figure 4. Family Properties notebook

## Required

Complete the fields on the **Required** page of the properties notebook as follows.

**Name** Type a name for your family.

**Path** Specify the fully-qualified path name of the directory where you want the database configuration information stored. TeamConnection places this information in a subdirectory of the path you specify. This subdirectory has the same name as the family. If you specify **c:\proddev** (for Intel) or **/proddev** (for UNIX) as the path name, for example, TeamConnection places all files related to the family in the directory path **c:\proddev\yourDBName** (Intel) or **/proddev/yourDBName** (UNIX).

**Note:** If a directory for the database name you specify already exists, you will need to delete it before you proceed. This procedure will fail if the directory already exists.

- Port** Specify the TCP/IP port address that you set in your TCP/IP services file.
- Mailer** Specify the name of the mail routine you want to use to notify users of actions they need to be informed of. See “Setting up the mail facility” on page 42 for information on setting up the mail routine for notification.

**Security level**

Select a level of security from the list box. Choose one of the following:

**Host-only**

A valid combination of the system login ID, TeamConnection user ID, and host name must be used to obtain access to the family. This is the default level of security.

**Password-only**

A user must log in to and log off of TeamConnection and supply a password in one of the following ways:

- Select **Login** from the **File** menu of the Tasks window.
- Issue the command `teamc tclogin` from a command prompt.

When the user logs in to the family, the family will send back a token associated with that user from that client. The server will check the attached token and, if valid, will proceed to perform the requested action.

If you specify the password-only option, you will need to specify a password for each TeamConnection user. See “Chapter 6. Preparing for your users” on page 63 for information on creating users.

**Password-or-host**

The user can use either the password-only function if he or she has a password or the host-only function if he or she has a valid host list entry. This level of security is useful for teams in which particular team members may be remote or mobile and have changing IP addresses. If the user supplies a valid password, then TeamConnection uses the password to admit access to the family. If the user either does not supply a password or supplies an incorrect password, then TeamConnection checks the user’s host list entry to admit access.

- None** Any user can access TeamConnection. Neither a password nor a valid host list entry is required.

See “Planning for user IDs” on page 63 for information on how to set up user IDs for the security level you select. See “Login managers” on page 65 for information on starting and stopping login managers for password-only and password-or-host security.

**Minimum password length**

Use this field to set the minimum number of characters to be used for passwords. The default password length is 8, the minimum is 1, and the maximum length is 32.

**Maximum invalid attempts**

Use this field to set the number of times users can attempt to log in before TeamConnection deactivates the user's ID. If this happens, a superuser must reactivate the ID before the user can attempt to log in again.

**Login** Specify a user ID for the superuser for the family. For Intel platforms, use the value set for the TC\_USER environment variable. To see this value, type the following from a command prompt and look for the TC\_USER variable:

```
set | more
```

For UNIX and Windows NT platforms, set this field to the login ID for the user.

**Userid** Specify the TeamConnection user ID for the superuser. If you omit this parameter, it defaults to the value specified in the **Login** field. It is a good idea to give the superuser an ID that is readily identifiable as a superuser. A good way to do this is to preface the user ID with su\_, such as su\_john.

**Host** Specify the TCP/IP host name for the family server machine, which was set in your TCP/IP hosts file.

To see this value, type the following from a command prompt:

```
hostname
```

**Note:** You do not need to use the fully-qualified host name. You can, for example, specify *myServer* instead of *myServer.myCompany.com*.

**Password**

If you want to use password security, you must specify the password to be used to verify the superuser's access to the TeamConnection server. If you do not specify the password for superuser access, then no one will be able to access the database. To use password security, you need to set the **Security level** field on the **Security** page of the family properties notebook to **password-only** or **password-or-host**.

The password must be a minimum of 1 character long and only include characters from the syntactic ASCII character set.

If you anticipate the need for security past the basic level of authentication (host-only) at any time in the future, it is recommended that you supply a user ID and password for the initial superuser when creating the family.

## Configurable fields

Use the **Configurable fields** page of the properties notebook to define special fields for defects, features, parts, releases, users, and work areas. See "Chapter 7. Working with configurable fields" on page 83 for more information on using this section of the properties notebook.

## Processes

This page of the family properties notebook provides access to two windows: one for defining release processes and one for defining component processes. To open one of these windows, select one of the **Settings** push buttons.

Use the **Release Process Settings** window and the **Component Process Settings** window to define processes and subprocesses for releases and components defined in your family. Complete the fields on these windows as follows. For more information about defining and using release processes, see “Chapter 8. Configuring family processes” on page 99.

- To see the default subprocesses defined for each release process, select a process name from the **Release Process** or **Component Process** list. The subprocesses included will appear highlighted in the **Subprocesses** list.
- To add or delete subprocesses for an existing process, follow these steps:
  1. Select a process from the **Release Process** or **Component Process** list.
  2. To add or delete a subprocess, select it from the **Subprocesses** list.
  3. To save your changes, select the **Apply** button.
- To create a new process, follow these steps:
  1. Select the **New** push button, type the name of the new process in the New Release Process or New Component Process window, and then select the **OK** push button.
  2. From the **Subprocesses** list, select the subprocesses you want to include in the new process.
  3. To save the new process, select the **Apply** push button.
- To delete a process, select it from the **Release Process** or **Component Process** list and then select the **Delete** push button. When the confirm delete window appears, select **Yes**.
- To rename a process, follow these steps:
  1. Select a process from the **Release Process** or **Component Process** list.
  2. Select the **Rename** push button.
  3. Type a new name in the **New name** field of the Rename Release Process or Rename Component Process window, and then select the **Apply** push button.

## User exits

Use the **User Exits** page of the properties notebook to define processes to be called at certain exit points for TeamConnection actions. See “Chapter 9. Providing user exits” on page 103 for more information on using this section of the properties notebook.

## Groups

This page of the family properties notebook provides access to two windows: one for defining authority groups and one for defining interest groups. To open one of these windows, select one of the **Settings** push buttons.

Use the **Authority Group Settings** window and the **Interest Group Settings** window to define authority and interest groups and actions for your family. Complete the fields on these windows as follows. For more information about defining and using authority and interest groups, see “Chapter 6. Preparing for your users” on page 63.

- To see the default actions defined for each authority or interest group, select a group name from the **Authority Group** or **Interest Group** list. The actions included will appear highlighted in the **Actions** list.
- To add or delete actions for an existing group, follow these steps:
  1. Select a group from the **Authority Group** or **Interest Group** list.
  2. To add or delete an action, select it from the **Actions** list.
  3. To save your changes, select the **Apply** push button.
- To create a new group, follow these steps:
  1. Select the **New** push button, type the name of the new group in New Authority Group or New Interest Group window, and then select the **OK** push button.
  2. From the **Actions** list, select the actions you want to include in the new group.
  3. To save the new group, select the **Apply** push button.
- To delete a group, select it from the **Authority Group** or **Interest Group** list and then select the **Delete** push button. When the confirm delete window appears, select **Yes**.
- To rename a group, follow these steps:
  1. Select a group from the **Authority Group** or **Interest Group** list.
  2. Select the **Rename** push button.
  3. Type a new name in the **New name** field of the Rename Authority Group or Rename Interest Group window, and then select the **Apply** push button.

---

### Adding an existing family to the Family Administrator window

You can add an icon to the Family Administrator window for an existing TeamConnectionfamily that was defined outside the GUI. To do this, follow these steps:

1. Select **Attach icon** from the Family pull-down menu. The Attach Icon to Family window appears.
2. Complete the fields on this window as follows:

- Name** Type the name of your family.
- Path** Specify the directory path where the family was created.
- Port** Specify the TCP/IP port address that you set in your TCP/IP services file.

- Mailer** Specify the name of the mail routine you want to use to notify users of actions they need to be informed of.
3. Select **OK**. An icon for that family appears.
  4. After you see your family icon in the Family Administrator window, you can:
    - Start and stop the family and notification servers. See “Using the Family Administrator GUI” on page 42 for instructions.
    - Change the default values in the database to better suit your needs. To change the default values, select the family icon and then select **Family** → **Properties**.

---

## For further information

The remaining chapters in Part 2. explain how to do the following tasks using the Family Administrator GUI.

For information about this task,	Go to this page.
Creating or modifying authority groups	72
Creating or modifying interest groups	78
Defining configurable field types	85
Creating configurable fields	88
Changing report formats	93
Configuring processes	99
Providing user exits	111

---

## Chapter 4. Starting and stopping the servers

This chapter explains how to start and stop TeamConnection servers using the family administrator GUI or the `teamcd` command.

- The family administrator GUI enables you start and stop the family server and the notification server.
- The `teamcd` command enables you to start and stop the family server, notification server, and build server.

TeamConnection also provides a `teamcbld` command for starting and stopping build servers. For more information about this command, refer to the *TeamConnection User's Guide*.

---

### Specifying the number of daemons to start

When you start the family server, you specify the number of daemons, one or more, that are to be started. A *daemon* is a process that runs as a background task and provides access to the TeamConnection database.

Because one daemon processes only one request at a time, the number of daemons you have running determines how quickly requests are processed. A daemon is not available until a request completes.

To determine the number of daemons to start, you need to understand the types of requests your users generally issue. For example, if many of the requests are for reports, which require longer processing, you will need more daemons than if most of the requests process quickly, such as checking files in and out.

Requests are queued and processed by the next available daemon. If the queue fills up, requests are not queued and the server refuses the connection. This is a signal that more daemons are needed.

If you do not explicitly specify the number of daemons when you start the family server, only one daemon is started. We recommend that you start with the cube root of the expected number of concurrent TeamConnection users. For example, start 3 daemons for 27 users, 4 daemons for 64 users. To change the number of daemons to start, you need to stop and restart the family.

Do one of the following to specify the number of daemons that you want to start:

- Use the Family Administrator GUI. For instructions, see “Using the Family Administrator GUI” on page 42.
- Use the `teamcd` command. For instructions, see “Using `teamcd`” on page 44.

---

## Setting up the mail facility

TeamConnection users can receive notification when certain events occur within TeamConnection. A user's mail address is specified when a TeamConnection user ID is created. TeamConnection uses this mail address to notify users when certain actions occur.

In order for users to receive notification, the notification server must be running. When you start the notification server, you specify an executable or command file that specifies the mail exit routine that processes mail requests.

The following mail exit routine samples are shipped with TeamConnection:

- mailexit.cmd (Intel platforms)
- mailexit.exe (Intel platforms)
- mailexit.ksh (UNIX platforms)

These samples are located in the directory where TeamConnection is installed. These samples use the sendmail command. You can either use one of these sample mail exits or you can use a different mail facility and write your own routine.

The sendmail command is part of TCP/IP, and is installed when TCP/IP is installed. If you use the sendmail function to send notification messages, you must configure it on your network in order for TeamConnection client workstations to receive notification messages from the server. Refer to your TCP/IP documentation for more information.

If you use a different mail facility, refer to the shipped mail exit routine sample, mailexit.c, to see how you can tailor TeamConnection to support your mail facility.

In order not to lose messages when the mail exit routine fails, you can have the exit routine return a code of 1041. This causes the notification daemon to exit and the mail that was being processed is not deleted. If the exit routine returns any other code, the mail that is being processed is deleted.

---

## Starting the servers

This section explains how to start the TeamConnection family server, the notification server, and the build server. You can start the family and notification servers using an icon in the Family Administrator GUI or the teamcd command. You can also use the teamcd command to start the build server. These processes can be started together when you start the family server, or individually.

### Using the Family Administrator GUI

You can follow these steps to start both the family and notification servers from the Family Administrator GUI:

1. Do one of the following to display the TeamConnection Family Administrator window:
  - From the TeamConnection Group folder on the desktop, double-click on the **Family Administrator** icon.
  - Type tcadmin from a prompt.
2. Double-click the family icon for the family you want to start. The Family Servers window appears.

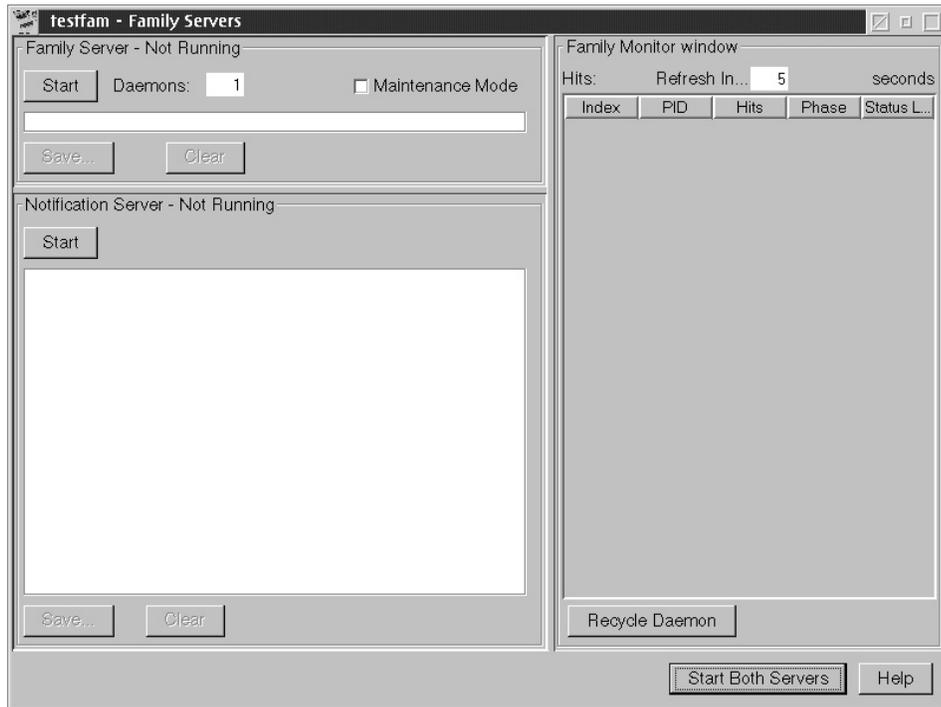


Figure 5. Family Servers window

3. When starting the family server, specify in the **Daemons** field the number of daemons you want started.
4. To start only one server, select the appropriate **Start** push button. To start both the family and notification servers, select the **Start Both Servers** push-button.  
When a server starts successfully, the message "Press CTRL-C to stop" appears in the list box and the **Start** push button changes to **Stop**.
5. Minimize the Family Servers window.

**Note:** Do not close the Family Servers window. Closing the Family Servers window stops the family server.

## Using teamcd

You can use the `teamcd` command to start the family server, notification server, and build servers together or to start any one of these by itself.

### Family server and, optionally, all servers

To start the family server from the command line, type the following from a prompt:

```
teamcd [-b bldsrvr -n mailexit -m] family n
```

Where:

- **-b** *bldsrvr* starts a build server and specifies the name of a file that describes the build servers that you want to start. Refer to the *TeamConnection User's Guide* for information about creating this file. You can also use the `TC_BUILD_RSSBUILDS_FILE` environment variable to set this value.
- **-n** *mailexit* starts a notification server and specifies the executable or command file to process mail requests. You can also use the `TC_NOTIFY_DAEMON` environment variable to set this value.
- **-m** starts the family in maintenance mode. While in maintenance mode, the family is locked into read-only mode and prevents users from updating the database while maintenance is being performed.

You can issue report queries and extract parts when the TeamConnection server is running in maintenance mode, but you cannot issue any commands that update the database. If you attempt a command that updates the database while the server is running in maintenance mode, you will receive an error message. You can supplement the text of this standard error message. In the server's `/config` directory, create a text file named **maintMsg** and place in it any appropriate text, such as "This TeamConnection family is down for backups from 2am to 4am daily."

- *family* is the name of the family you are starting.
- *n* is the number of daemons that you want to start. When starting the family server, if this value is not typed, the default is 1. When starting only build servers or the notification server, specify 0 for this parameter.

It is recommended that you use this command to start your build servers. However, you can start the build server separately as described in the *TeamConnection User's Guide*.

### Build server only

TeamConnection provides build servers on the following platforms: AIX, HP-UX, Solaris, OS/2, Windows NT, Windows 95, MVS, and MVS/OpenEdition.

Other than MVS and MVS/OE build servers, you can start build servers using either the `teamcd` or `teamcbld` command. We recommend you use the `teamcd` command as it provides better process and memory management of the build servers.

To use the `teamcd` command to start a build server apart from starting the family server (on the same machine), type the following from a prompt:

```
teamcd -b bldsvr family 0
```

Where:

- *bldsvr* is the name of a file that describes the build servers that you want to start. Refer to the *TeamConnection User's Guide* for information about creating this file. You can also use the `TC_BUILD_RSSBUILDS_FILE` environment variable to set this value.
- *family* is the name of the family for which you are starting the build server.
- 0 indicates that only the build server, and not the family server, is to be started. When you want to start only a build server, you must specify 0 as the number of daemons, otherwise TeamConnection will start one family daemon.

**Note:**

For information on starting the build sever using the `teamcbld` command, refer to the *TeamConnection User's Guide*.

An MVS or MVS/OE build server cannot be started using the `teamcd` command. Refer to the *TeamConnection User's Guide* for instructions on starting an MVS build server.

**Notification server only**

You can use the `teamcd` command to start the notification server apart from starting the family server by typing one of the following commands from a prompt:

```
notifyd family mailexit
```

```
teamcd -n mailexit family 0
```

Where:

- *family* is the name of your family.
- *mailexit* is the executable or command file that specifies the exit routine to process mail requests. You can also use the `TC_NOTIFY_DAEMON` environment variable to set this value.
- 0 on the `teamcd` command indicates that only the notification server, and not the family server, is to be started. When you want to start only a notification server, you must specify 0 as the number of daemons on the `teamcd` command, otherwise TeamConnection will start one family daemon.

---

## Stopping the servers

You can stop the family and notification servers from the Family Administrator GUI or the command line.

- **From the GUI:**

If you started the family or notification servers from the Family Administrator GUI, follow these steps to stop them:

1. From the Family Servers window, select the **Stop** push button for the appropriate server to stop only one server. To stop both the family and notification servers, select the **Stop Both Servers** push button.
2. Close the Family Servers window.

- **From a command line:**

1. If you started the family server from a command line, type the following at a prompt. Substitute the name of the family you want to stop for *familyName*.

```
tcstop familyName
```

---

## Chapter 5. Setting up your family structure

After you create your family, it is important that you think about the following:

- How to arrange your component structure
- How to organize your releases
- What processes you want to use

This chapter helps you determine how you want to organize your family and then explains how to do it.

You need to understand what families, components, releases, and processes are and what their purpose is within TeamConnection. If you have not already done so, read “Chapter 1. An introduction to TeamConnection” on page 3, before continuing.

The following table directs you to the task you need:

<b>For information about this task,</b>	<b>Go to this page.</b>
Planning your component structure	47
Planning your releases	50
Planning your processes	53
Creating your components and releases	59

---

### Planning your components

This section discusses how you can organize your component hierarchy to support your configuration management needs.

### Organizing the component hierarchy

You can organize your component hierarchy several ways. For example, one component hierarchy might mirror the application development organization hierarchy, such as department, section, team, or unit of development. Another hierarchy might reflect the software architecture of the applications under development, such as application, GUI, database.

When you set up your component hierarchy, consider that all defects and features are recorded by component, and the owner of a component becomes the default owner of the defects and features for that component. This is important because defect and feature owners automatically receive a considerable amount of authority over the defects and features they own. To see the actions that defect and feature owners can perform, refer to the authority and notification table in the *TeamConnection User's Guide*.

If you create your component hierarchy to store software or documentation source files, it is best to reflect the product organization at the top level. You can then create descendant components to reflect the development or maintenance responsibilities. Figure 6 gives an example of this type of structure.

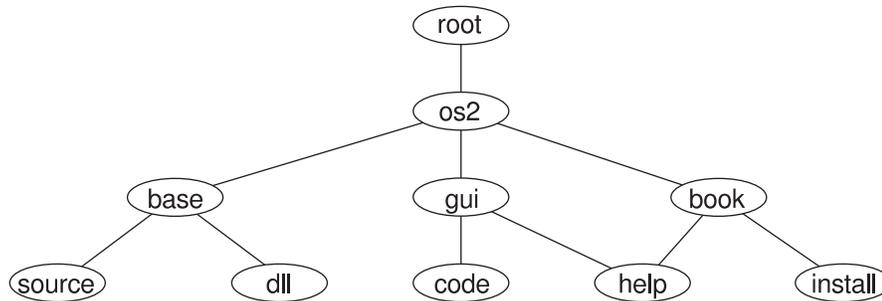


Figure 6. A hierarchy representing product organization

Your component hierarchy can consist of several parallel hierarchies so that you can easily restrict access to certain related components. For example, if you have vendors working on your development team, you might want to restrict their access to certain information. You can create a parallel hierarchy that contains only the information that they require. Figure 7 represents this type of structure.

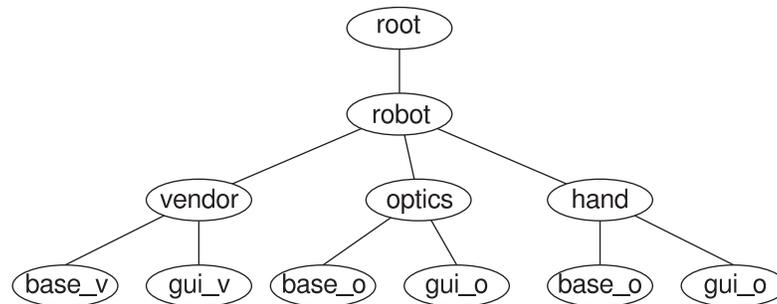


Figure 7. A hierarchy showing parallel components

Components can have more than one parent. A component that has more than one parent inherits authority and notification from both. In Figure 8 on page 49, the component `optics` groups both the `optics_v` and `optics_d` components for the development project, giving the `optics_v` components two parents. The `optics` component manages notification for the entire `optics` team. Access control is managed separately for the vendors and the internal users through the lower-level components, `optics_v` and `optics_d`.

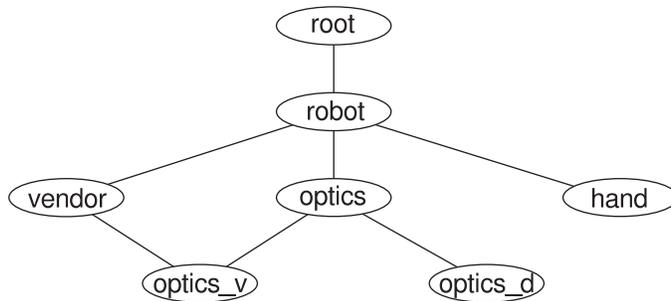


Figure 8. Components with more than one parent

Your initial component hierarchy is not necessarily going to be the same as your hierarchy a year from now. It will change as your organization grows and as your needs change. Remember that you can change the parents of a component as well as delete or rename the component.

When you plan your component hierarchy, you might find it helpful to first sketch it on paper. You can then use this sketch to help you make a table in which you note information about each component, such as the type of parts you want to control with the component, what releases a component will manage, and which processes each component and release will initially follow.

## Determining component ownership

Each component in the hierarchy has an owner. Initially that owner is the person who creates the component. After the component is created, the owner can, at any time, transfer ownership to another person.

Ownership of a component is critical. A component owner has authority to perform a wide variety of actions on that component and the parts contained in that component, as well as on all its descendant components and their parts. For example, the owner has authority to give other users access to the component and its parts and to delete the component.

You might create many of the initial components for your development organization, but you probably will not want to remain the owner of them all. As you are planning your component hierarchy, determine whom you want to own each component. The owner of the root component, the component at the top of the hierarchy, should be the person with overall responsibility for the project. If several other people have responsibility for various pieces of the development project, you might want those people to own the descendant components that relate to their piece of the project.

The owner of a parent component has the same level of authority for all of its descendant components.

Figure 9 shows a portion of a component hierarchy that Sam, a family administrator, created. Sam transferred ownership of many of the components to other members of the team. However, he kept ownership of the root component because he has overall responsibility for the project.

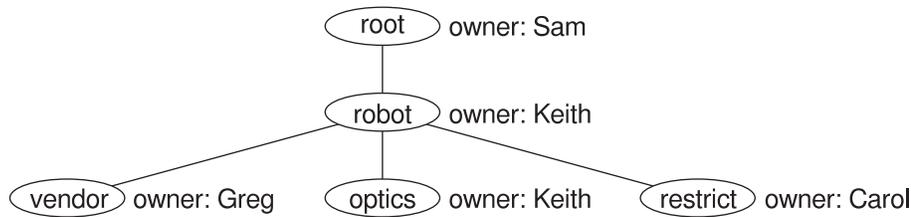


Figure 9. A hierarchy showing component ownership

## Naming the components

During the planning stage, it is helpful to decide on a component naming convention. Do you want each component name to reflect the type of data it is managing? If so, you need to understand the content, function, execution platform, or other characteristics of the parts the component will manage. For example, the name for a component that manages data for the graphical user interface of your application might begin with the characters *gui*. Do you want component names to be in all lowercase characters, all uppercase characters, or in mixed case? The database is case sensitive. Therefore, when you are consistent with how your components are named, your users will have less difficulty finding objects in the database. The names you use must be unique within the family.

Other TeamConnection users will be able to create components, so you will want to publicize your naming convention so that everyone can adhere to it. TeamConnection users need to access TeamConnection data, and that data can be difficult to find if they do not understand and follow your naming convention.

## Determining access to components

Each component has an access list that controls access to development data. Access authority is inherited for all descendant components, but can be explicitly restricted in the descendant components. See “Planning for user access to TeamConnection data” on page 70 for instructions. If you need to restrict access to several components for most users, you might need to redesign your component hierarchy.

---

## Planning your releases

After you decide how you are going to organize your components, determine the releases that you will initially create.

Basically, a release is a logical grouping of parts. This group of parts makes up a single version of a product, or part of a product, that is built separately, such as documentation or test cases. One release can group parts that are managed by many components.

## Relating releases with components

Every release is associated with a component that manages which users can access the parts in the release and which users are notified when certain actions occur. The only relationship between a release and the component from which it was created is to use the Access list of that component. You can create parts in the release that belong to another component.

For example, Figure 10 shows the component hierarchy for a development project. Keith owns the component robot. He creates the release robot\_control to contain all the parts that pertain to the first version of the application they are developing. When Keith created the release, he specified robot as the managing component, but he did not specify an owner. Therefore, Keith is the release owner by default.

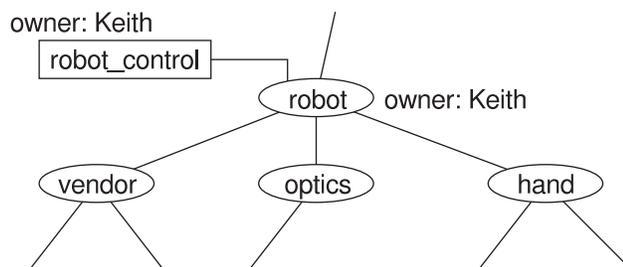


Figure 10. The release-component relationship

Keith decides that Doug should own release robot\_control. As owner of the release, Doug has authority to perform most actions against the release. However, access and notification for the release are managed by component robot and are controlled by Keith, the owner of robot. In this way, Keith can maintain access and notification control of the release, even though he has delegated the management responsibilities to Doug.

If Keith wants to give Doug the ability to control the access to release robot\_control, he can add Doug's user ID to the component's access list and specify an authority group, such as componentlead, that contains the authority to add users to access lists. Both Doug and Keith would then have the authority to add entries to the access list of component robot.

You can learn more about access lists and authority groups in "Planning for user access to TeamConnection data" on page 70.

## Selecting serial or concurrent development

The release can be set up for developing in serial development or in concurrent development mode. In *serial development*, a part is locked when a user checks it out, and no one else can update the part as long as it is checked out. In *concurrent development*, more than one user can simultaneously have the same part checked out. When TeamConnection detects that someone else has made changes to a part that another is checking in, it notifies the user that a collision has occurred. The user can reconcile the changes using the TeamConnection merge program.

You specify the mode in which your users will work when you create the release. Be aware, however, that after the mode is set to concurrent, you cannot change it back to serial. However, it is possible to change the mode from serial to concurrent.

## Controlling database growth

To optimize TeamConnection performance, you can control the size of your database in the following ways:

- Automatically by setting options when you create a release
- Manually by pruning a release

The following sections explain these methods of controlling database growth. The *IBM DB2 Universal Database Administration Guide* contains more information about managing the size and growth of your database.

### Controlling database size automatically

You can help control the size of your family database by requesting the following options when creating a release:

- Automatic pruning of work areas
- Maximum number of build output versions

These options let you reclaim database space without extra work.

**Automatic pruning of work areas:** To understand automatic pruning of work areas, it helps to understand the basics of work area versioning. Every time you freeze a work area, TeamConnection saves a revision level of the work area. When you freeze work area 123, for example, a version called 123:2 is created. This version contains information about each part in the work area and its current version at the time the work area was frozen. It may contain version 1 of part optics.c, for example. If you freeze the work area again later, a new version called 123:3 is created with information about the versions of the parts in the work area when it was frozen. This version may contain version 2 of part optics.c. Each of these work area versions is saved in the database and you can retrieve the versions of the parts they contain before you integrate the work area into the release.

Automatic pruning enables you to delete all versions of work areas after you have integrated the most current version of the work area into a release. You can indicate whether you want automatic pruning of work areas by doing one of the following:

- Select **Automatic version pruning** on the Create Releases window when you create the release
- Use the `+autopruning` flag with the release command

These options tell TeamConnection to destroy work area versions when a user integrates a work area or commits a driver to the release.

Be aware that when work areas are destroyed, most of their change tracking information is also destroyed and it will be more difficult for you to go back to previous versions.

**Maximum committed output versions:** You can also indicate the maximum number of committed build output versions you want kept. When that number is reached, TeamConnection discards the oldest one. Otherwise, all build outputs are saved. You can set the maximum committed build output versions by doing one of the following:

- Specify the number you want kept in the **Maximum number of output versions** field on the Create Releases window
- Use the `-outputVersions` flag in the release command

## Controlling database size manually

If you choose not to use autopruning to control database size, you can still prune your releases manually as follows:

1. From a Releases window, select the release you want to prune and then select **Prune** from the **Selected** menu.
2. In the **Version branch** field of the Prune Release window, type the name of the first version of the branch associated with the work area you want to prune and then select **OK**.

## Naming your releases

Next, decide how you are going to name the releases you create. You might want to name your releases according to the product or object you are building. For example, *prod1r1* for release 1.1 of your application, or *using1r1* for the book files for release 1.1 of your application. To make it easier on your users, continue using the basic naming convention that you are using for your components. The names you use must be unique within the family.

---

## Planning your processes

Before you create your family's components and releases, decide what processes you are going to use during initial development.

A TeamConnection process is used to enforce a specific level of control of components and releases. TeamConnection is shipped with a set of predefined processes for both components and releases, so you provide different processes for each to follow. You can use these processes, or you can configure your own processes using some of the predefined subprocesses. “Chapter 8. Configuring family processes” on page 99 explains how you configure TeamConnection processes.

You probably already have a process for tracking problems as well as a process for tracking suggested improvements to your applications. If you want to continue to use those processes, determine how you can best group the TeamConnection subprocesses to reflect your current process. If you do not have an existing method, decide how tightly you want to control part changes and track defects and features.

The poster, *Staying on Track with TeamConnection Processes*, explains the various states that different TeamConnection objects can go through depending on the process that is being followed. You might want to study this information before you determine how you want to use TeamConnection processes.

## Component processes

A component's process determines how much planning and designing is required before work on a defect or feature begins and whether the originator is required to verify that the work was done correctly.

When choosing a process for a component to follow, think about the type of data within the component. For example, the parts within one component might contain complex code that is time-consuming to fix. Before any defects or features are accepted, the work needs to be designed and sized, so the *preship* process is followed. Parts within another component contain code that is relatively easy to fix and test. The defects and features for this component do not need to be designed and sized, so the *prototype* process, which contains no subprocesses, is followed.

For components, you can require users to follow any, all, or none of the following predefined subprocesses:

### **dsrDefect**

Design, size and review fixes to be made for defects

### **verifyDefect**

Verify that the fixes work

### **dsrFeature**

Design, size, and review changes to be made for features

### **verifyFeature**

Verify that the features have been implemented correctly

The following table lists the component processes that are supplied by IBM. Each process combines a set of TeamConnection subprocesses. An X under the TeamConnection subprocess indicates that the corresponding process includes it.

Table 16. Shipped component processes

Shipped TeamConnection component process	TeamConnection subprocesses			
	dsrDefect	dsrFeature	verifyDefect	verifyFeature
default		x	x	x
development		x		
emergency_fix				
maintenance			x	x
preship	x	x	x	x
prototype				
test		x	x	x

## Release processes

A release's process determines to what extent part changes are tracked and the procedure for integrating changed parts into a build. Release processes control the day-to-day work that is involved in producing the product—fixing defects and implementing features, as well as building the product. The type of process control you want to enforce on a release is likely to change over time.

For releases, you can require any, all, or none of the following predefined subprocesses:

**track** This subprocess is TeamConnection's way of relating all part changes to a specific defect or feature and a specific release. Each work area gathers all the parts modified for the specified defect or feature in one release and records the status of the defect or feature. The work area moves through successive states during its life cycle. The TeamConnection actions that you can perform against a work area depend on its current state.

You must use the track subprocess if you want to use any of the other release subprocesses.

### approval

This subprocess ensures that a designated approver agrees with the decision to incorporate changes into a particular release and electronically signs a record. As soon as approval is given, the changes can be made.

**fix** This subprocess ensures that as users check in parts associated with a work area, an action is taken to indicate that they have completed their portion. When everyone finishes, the owner of the fix record (usually the component owner) can change the fix record to complete. The parts are then ready for integration.

**driver** A driver is a collection of all the work areas that are to be integrated with each other and with the unchanged parts in the release at a particular time. The driver subprocess allows you to include these changes incrementally so that

their impact can be evaluated and verified before additional changes are incorporated. Each work area that is included in a driver is called a driver member.

**test** The test subprocess guarantees that testing occurs prior to verifying that the fix is correct within the release.

Another level of control is to use release process attributes, which alter the automatic state changes applied to a work area.

#### **trackfixhold**

With the trackfixhold attribute and the fix subprocess a work area will remain in the fix state rather than moving to the integrate state when the final Fix -complete command has been issued. To move the work area to integrate state, issue a Workarea -integrate command.

#### **trackcommithold**

With the trackcommithold attribute a work area will remain in the commit state when

- a Driver -complete command is issued for a release with a driver subprocess.
- the final Fix -complete command is issued for a release without a driver subprocess and with the fix subprocess.
- the WorkArea -integrate command is issued for a release without a driver subprocess and without the fix subprocess.

To move the work area to test state, issue a Workarea -test command.

#### **tracktesthold**

With the tracktesthold attribute and the test subprocess a work area will remain in the test state rather than move to the complete state when the final test is marked. To move the work area to complete state, issue a Workarea -complete command.

To add these attributes to your release process, add the following to your relproc.ld file and then reload it.

```
track_test|trackfixhold
track_test|trackcommithold
track_test|tracktesthold
```

See “Configuring component or release processes” on page 168 for instructions on editing and reloading relproc.ld.

The following table lists the release processes that are supplied by IBM. Each process combines different sets of TeamConnection subprocesses. An X under the TeamConnection subprocess indicates that the corresponding process includes it.

Table 17. Shipped release processes

Shipped TeamConnection release processes	TeamConnection subprocesses				
	track	approval	fix	driver	test
prototype					
development	x		x		x
test	x		x	x	x
preship	x	x	x	x	x
maintenance	x		x	x	x
emergency_fix	x			x	
track_only	x				
track_driver	x		x	x	
track_approval	x	x	x	x	
track_test	x		x	x	x
track_full	x	x	x	x	x
no_track					

It is important that your users understand the meaning of each process and the type of control it enforces. For example, if a stringent release process such as `track_full` is selected, actions have to occur in a precise order. Compare this to the `no_track` process where users can freely check parts in and out of TeamConnection.

## How processes might change during development

TeamConnection provides different processes for components and releases. The processes you choose depend on how tightly you want to control changes and how you want to handle defects and features. Your choices, of course, will vary depending on where you are in your current development cycle. You can change your processes during a development effort to reflect different phases. For example, you might do the following:

- During the requirements gathering phase, you create a component that manages the requirements documentation. You want minimal defect or feature processing against the parts managed by this component, so you select a process, such as *prototype*, that is not strict. The release would also follow a relaxed process, such as *prototype*.
- After the requirements are settled and design work begins, you want to control changes to the requirements data but not to the rapidly evolving design data. For the requirements component, you change to a process that includes review and verification, such as the *default* process. You also create a new component to manage the design documentation and you select a process that is not strict. You continue to use the *prototype* process for the release.
- When coding begins, you change the process for the design component to one that includes review and verification, such as *development*. You also create a new component to manage the code files. Because you will be loading files into TeamConnection rapidly, you select a process that is not strict, such as *prototype*.

You also change to a release process, such as *development*, that tracks the resolution of defects and features.

- After all the code files that are managed by a given component pass unit test, you change that component's process to one that includes review and verification, such as *default*. You also change the release process to one with tight control, such as *track\_full*, so that you can carefully manage code changes.
- Ninety days before your delivery date, you change all the components to a very stringent process, such as *preship*, to ensure that all new features or defects are reviewed for impact to the delivery schedule.

## Using the driver subprocess

The driver subprocess is a way to better control the building and testing of your application code. As you develop your application program, you will probably have many drivers within a release, and you can have multiple overlapping releases during a development cycle.

For example, let's say you are developing a robot application and you send monthly updates to customers for their feedback. You do regular driver builds of your application. You use the driver subprocess to help you control the integration of changes that occur between builds. At some point during the month you cut off changes to the current release r9504 for system testing. You are then ready to create a new release called r9505. Using TeamConnection, you can link the parts in r9504 to the new release r9505. During the follow-on development work, release r9504 is still there. At the end of the month you send your final build and tested driver of release r9504 to your customers.

The following figure depicts this process graphically. In this illustration, releases are labeled *ryymm*, where *yy* represents the year (such as 95 for 1995) and *mm* represents the month (such as 04 for April). Drivers are labeled *dmd*, where *m* represents the month (such as 4 for April) and *dd* represents the day of the month.



Figure 11. Using the driver subprocess

---

## Creating components and releases

Now that you have gone through the planning phase and have your structure on paper, you are ready to create the components and releases that your organization will use.

### Creating components

For each family you create, TeamConnection creates the top component called root. Therefore, at least the first component you create has root as the parent. Do not change the name of the root component.

As with most TeamConnection tasks, you can use either the GUI or the command line. To create a component, do one of the following from a client machine:

- From the GUI:
  1. From the Tasks window, select **Components** → **Create** from the Actions pull-down menu. The Create Components window appears.

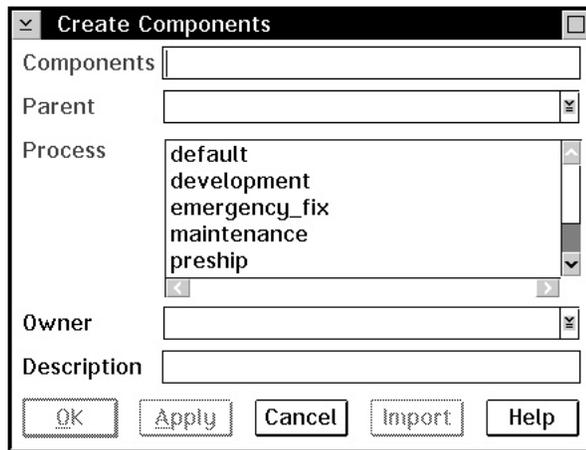


Figure 12. Create Components window

2. Type the component name, the parent name, and the name of the process you want the component to follow. Other information on this window is optional.
3. Select **OK** to create the component and close the window, or select **Apply** to create the component and leave the window open.

- From a command line, type:

```
teamc component -create componentName -parent parentName
                -process processName
```

For more information about the component command, refer to the *Commands Reference*.

## Creating releases

To create a release, do one of the following from a client machine. Before creating a release, read “Planning your releases” on page 50.

- From the GUI:
  1. From the Tasks window, select **Releases** → **Create** from the Actions pull-down menu. The Create Releases window appears.

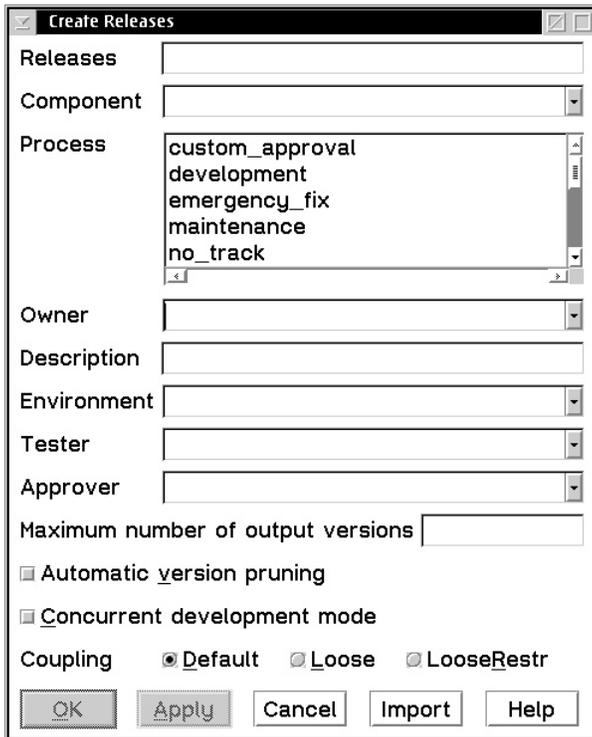


Figure 13. Create Releases window

2. Type the release name and the name of the managing component, and select the process you want the release to follow.

If you want your users to work on parts concurrently, check **Concurrent development mode**. For serial development, leave this box unchecked. Be aware that when you select a development mode, you cannot change it.

To reclaim database space without extra work, specify that you want automatic pruning of work area versions that have not been integrated with the release. Another way to save database space is to specify the maximum number of build output versions you want kept.

Use the **Coupling** field to control how TeamConnection handles common parts:

**Default** TeamConnection part commonality functions normally.

**Loose** Allows the release to exist without requiring a `-force` or `-common` attribute on Part actions.

**LooseRestr**

Specifies that the release cannot be kept common with any other release by the `-common` mechanism.

Refer to “Planning your releases” on page 50 for more information about these options.

3. Select **OK** to create the release and close the window, or select **Apply** to create the release and leave the window open.
- From a command line, type:

```
teamc release -create releaseName -component componentName  
              -process processName [-concurrent] [+autopruning]  
              [-outputVersions number] [-coupling default|loose|looseRestr]
```

For more information about the release command, refer to the *Commands Reference*.

## Creating a new release from an old release

A TeamConnection family contains the work of many individuals for one product or project. Within that family, several releases can be created.

For example, currently, a team is working in the release `robot_control`. After this team finishes with the current edition of the robot project, the next team might work on a follow-on release of the robot product. That team could create a new release called `robot_v2` in which to work. Another possibility is a team that wants to implement the `robot_control` program on a different type of robot (similar to developing the same application for a different operating system). The team could create a release called `robot_mk5` in which to work. These various releases in a family are used to isolate changes to a similar code base.

These examples illustrate that the various releases in a family will often share a code base from another release. Administrators can link releases in order to share code between the linked releases.

For example, to create a new release called `robot_v2` that links to release `robot_control`, do the following using either the GUI or command line interface:

1. Create the new release `robot_v2`.
2. Create a work area.
3. Link the existing `robot_control` release to the new release `robot_v2` using the work area that was just created.
4. Integrate the work area with the new release.



---

## Chapter 6. Preparing for your users

This chapter helps you determine how to set up user authentication for the security level in use by your family, how to identify your users to TeamConnection, and how to set up authority groups and interest groups.

Before you start defining users, make sure you have read “Chapter 3. Creating your TeamConnection family” on page 29 and understand how your TeamConnection installation implements security for families.

---

### Planning for user IDs

Each user must have a TeamConnection user ID that uniquely identifies the user to TeamConnection and gives the user access to TeamConnection objects. TeamConnection uses the following terms and objects to identify users and control their access to TeamConnection information:

#### **User ID**

The ID by which TeamConnection knows you and assigns access authority to you, and the ID under which you issue TeamConnection commands. This ID is the one specified by the TC\_BECOME environment variable or on the **Become user** field of the TeamConnection settings notebook.

#### **Login ID**

This term is most meaningful in a multiuser environment, such as AIX, HP-UX, Solaris, or Windows NT. The login ID is the ID that you use to log in to your workstation and is specified on the **User ID** field of the TeamConnection settings notebook. In single-user environments, such as OS/2 the login ID is the one specified by the TC\_USER environment variable. In Windows 95, the login ID is used if one is specified, otherwise, the TC\_USER environment variable is used.

Other information that TeamConnection uses to authenticate users varies according to the security level in use by your family.

### **Host only**

If your family uses host-only security, then each user requires a valid combination of the system login ID, TeamConnection user ID, and host name to access the family. This is the default level of security. To set up user authentication for host only security, you need to create a host list for each user ID. Host lists associate user IDs, login IDs, and host names. If you are using host-list security, each TeamConnection user has a host list that controls which user IDs, host names, and login IDs he or she can use to access TeamConnection.

If the hosts in your site use IP addresses that are assigned dynamically, then the host-only authentication level will not work. In this case, you can use the password-only authentication level.

To create user IDs, see “Creating user IDs” on page 66. To create host lists, see “Planning for host lists” on page 68.

## Password only

Password-only security requires a user to log in to and log off of TeamConnection and supply a password in one of the following ways:

- Select **Login** from the **File** menu of the Tasks window.
- Issue the command `teamc tclogin` from a command prompt.

When the user logs in to the family, the family sends back a token associated with that user from that client. The server checks the attached token and, if valid, proceeds to perform the requested action.

Use password-only security if the hosts in your site use IP addresses that are assigned dynamically.

To set up user authentication for password-only security, you need to create user IDs for each user according to the instructions in “Creating user IDs” on page 66 and then create a password for each user according to the instructions in “Adding and modifying passwords” on page 67.

For information on how TeamConnection’s login manager works, see “Login managers” on page 65.

## Password or host

If your family uses password-or-host security, users can either login to the family with a password or access the family with a valid host list entry. This level of security is useful for teams in which particular team members may be remote or mobile and have changing IP addresses. If the user supplies a valid password, then TeamConnection uses the password to admit access to the family. If the user either does not supply a password or supplies an incorrect password, then TeamConnection checks the user’s host list entry to admit access.

For information on how TeamConnection’s login manager works, see “Login managers” on page 65.

## None

If your family does not use any level of security (if you specified **None** for the security-level option), users can access TeamConnection from any client without

entering a password. Though all TeamConnection users need a user ID to access the database, when the security level is None, TeamConnection does not require the user ID to have a password or a host list entry.

Use this authentication level with caution and only when absolutely necessary. For example, if the superuser forgets his or her password and the authentication level is password only, then the superuser can stop the family, change the authentication level to None, restart the family, modify the password, stop the family, change the authentication level back to password only, and restart the family. Do not use this authentication level for normal operations.

## Login managers

To enable users to execute as many commands as possible without authenticating before each command, they must log into the family server under one of the following conditions:

- The server is running in password-only mode.
- The server is running in password-or-host mode, and the user is attempting to access the server from a host that is not defined in a host list for that user.

When the user logs in, the server generates a token for that user, and each TeamConnection command authenticates itself to the server by passing that token to the server. When the user logs out, the server discards the token and will no longer accept it. In order to "remember" the token across invocations of the client commands, the client machine automatically starts a login manager to perform the login operation and to remember the token.

When the user issues a TeamConnection command, the command requests the token from the login manager and then forwards the token to the server. When the user logs out of the family server, the login manager exits normally.

On single-user operating systems, only one login manager will ever be running, but on multiuser systems, such as AIX, HP-UX, Solaris, and Windows NT, by default, one login manager will run for each user logged into a family. Running multiple login managers can strain resources. To avoid this situation, the superuser can start a global login manager that will act as the login manager for every user.

The global login manager must be started by root or someone with system administrator authority and with TeamConnection superuser authority. To start the global login manager, execute the following command:

```
# teamc tclogin -START_TCLOGINMGR
```

To stop the global login manager, execute the following command:

```
# teamc tclogin -KILL_TCLOGINMGR
```

## Creating user IDs



On Intel platforms, TeamConnection provides a User Management Wizard that you can use to add, delete, and edit user IDs. The User Management Wizard guides you through these tasks and prompts you for the information you need to provide to create, delete, and modify TeamConnection user IDs.

You can use a number of methods to assign IDs to your users. For instance, you can have each user's TeamConnection ID match the user's login ID. This is easy to do because each user already has a login ID. This method can be ideal if your users use the same login ID across their different systems, but it's confusing if they do not. For example, if Chris Wright has access to two workstations and logs in to one as *chris* and the other as *wright*, then identifying all the objects that belong to Chris Wright is more complicated. If you use this method, when a user moves on to another project, you will have to transfer the ownership of objects from that user to another user. Using this method can make additional work for you on a project where people move around a lot.

Another method is to assign IDs according to the roles that people have, such as *proj\_lead*, *writer1*, *tester\_mvs*, and *manager*. When you use this method, ownership remains the same when people leave the project. However, it can be more difficult to identify the person who owns a particular ID. For example, it is easier to identify Chris Wright as the user of the ID *cwright* than it is to identify him as the owner of the ID *writer1*.

You must have superuser or admin authority to create user IDs. To create a new user ID in TeamConnection, do one of the following from a client machine:

- From the user interface:
  1. Select **Users** → **Create** from the Actions pull-down menu on the Tasks window. The Create User window appears.

The image shows a 'Create User' dialog box with the following fields and controls:

- User ID: [Text Input Field]
- Mail address: [Text Input Field]
- Full name: [Text Input Field]
- Area: [Text Input Field]
- Superuser
- Buttons: OK, Apply, Cancel, Help

Figure 14. Create User window

2. Type the user's ID and electronic mailing address. Other information on this window is optional.

If the mail address you enter in this window is unreachable from the server, you will receive an error message. For more information, select **Help** from the Create User window.

3. Select **OK** to exit the window, or select **Apply** when you want to add another user ID immediately.
- From a command line, type:  

```
teamc user -create -login userID -name name -address mailAddress
```

For more information about the user command, refer to the *Commands Reference* book.

Superuser privilege is granted to one user ID when TeamConnection is installed. This privilege is required so that at least one person has privileged access to the family to perform special tasks, such as creating and deleting other user IDs. The person with superuser privilege can perform all possible actions in your TeamConnection family. This is an authority level that you definitely want to limit to only a very few individuals. In fact, individuals who have a superuser ID should also have another ID that has less authority, which is the ID they will use when doing their normal work. The user can switch between the two IDs by using the TC\_BECOME environment variable.

To give a user superuser privilege, include the +super flag with the user command, or select the **Superuser** check box when using the GUI. When creating superuser IDs, you might want to begin the ID with su\_. This will make it obvious as to which ID has superuser privilege. You must have superuser privilege yourself in order to give this authority to someone else.

**Note:** If you are using password security, you must add a password to the user's ID. "Adding and modifying passwords" provides more information. If you are using host-list security, you must add at least one host address entry to the user's host list to enable TeamConnection to recognize a new user. "Planning for host lists" on page 68 provides more information.

---

## Adding and modifying passwords

If you are using password-only or password-or-host security, you will need to modify the user IDs you have created to add passwords to them. To add or modify a user ID's password, follow these steps:

- From the user interface:
  1. Select **Users** → **Modify** → **Password** from the Actions pull-down menu on the Tasks window. The Modify Password window appears.
  2. Type the user ID in the **User ID** field.
  3. Type the password in the **Password** and **Verify Password** fields.

**Note:** The default minimum password length is 1 character. To determine if the minimum password length for a family is something other than the default, check the **Required** page of the properties notebook for the family.

4. If you are modifying an existing password, type the old password in the **Old Password** field.
  5. To apply the changes and leave the Modify Password window open, select the **Apply** push button. To apply the changes and close the window, select the **OK** push button.
- From a command line, type:  
`teamc user -modify -login userID -password password`

For more information about the user command, refer to the *Commands Reference*.

---

## Planning for host lists

When host-list security is in effect, each user ID is associated with a host list, which is a list of client machine addresses from which the user can access TeamConnection when using that ID. Users must have at least one entry on their host lists so that TeamConnection will recognize them as valid users.

The following example illustrates how these terms and objects are put into use and why TeamConnection requires both a login ID and a user ID:

A user named Chris Wright has the following TeamConnection responsibilities:

- Develops code for a product
- Performs superuser tasks for the family in which the product is developed
- Supervises builds of the product

Chris has a workstation with the TCP/IP host name *cwright.company.com*. This workstation is used for Chris's daily programming activities and for superuser activities. For day-to-day programming work, Chris uses the TeamConnection user ID *cwright* from host name *cwright.company.com*. For superuser activities, Chris uses the TeamConnection user ID *su\_cwright* from host name *cwright.company.com*. For build activities, Chris has access to a workstation with the TCP/IP host name *build.company.com*. Chris logs into *build.company.com* as *cwright* and extracts files from TeamConnection as user *build*. To enable Chris to perform each of these activities with the proper TeamConnection authority, Chris needs the following TeamConnection host list entries:

*Table 18. Host list entries for a sample user*

<b>Login ID</b>	<b>Host name</b>	<b>User ID</b>
<i>cwright</i>	<i>cwright.company.com</i>	<i>cwright</i>
<i>cwright</i>	<i>cwright.company.com</i>	<i>su_cwright</i>
<i>cwright</i>	<i>build.company.com</i>	<i>build</i>

With these host list entries, Chris can do the following:

- Access TeamConnection as user ID cwright to perform daily programming tasks from workstation cwright.company.com.
- Access TeamConnection as user ID su\_cwright to perform superuser tasks from workstation cwright.company.com.
- Access TeamConnection as user ID build to perform build activities from workstation build.company.com.

A superuser or someone with admin authority must create the initial host entry for a user. After the initial entry is created, users can add host list entries for themselves. Additional entries in a host list lets a user access TeamConnection from other client machines.

---

## Creating host list entries

The initial host list entry for each user must be created by someone with superuser or admin authority. To create a host list entry in TeamConnection, do one of the following from a client machine:

- From the user interface:
  1. Select **Users** → **Add host** from the Actions pull-down menu on the Tasks window. The Add Host window appears.

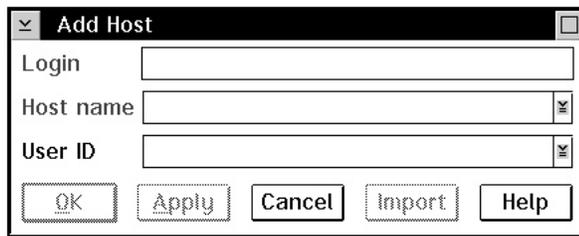


Figure 15. Add Host window

2. Type the login ID and the name of the client machine from which the user will access TeamConnection. The user ID is optional.  
For more information, select **Help** from the Add Host window.
  3. Select **OK** to exit the window or select **Apply** when you want to add another host list entry immediately.
- From a command line, type:

```
teamc host -create login@hostName -login userID
```

The login value following -create is the user ID (TC\_USER) with the host name appended to it, while the -login attribute flag is the TeamConnection user's ID

(TC\_BECOME) for the GUI (usually, these values are the same). The value of the user's ID is case sensitive, so type it exactly as it was typed when the user was created.

For more information about the host command, refer to the *Commands Reference*.

---

## Planning for user access to TeamConnection data

As soon as a TeamConnection user ID and a password (for password security) or host list entry (for host-list security) are created for a user, that user automatically has the authority to perform certain basic actions within the family. This authority is referred to as *base authority*. Beyond base authority, authority to access TeamConnection data is managed by the components that you create. Each component has an *access list* that controls access to development data. Authority granted in an access list is called *explicit authority*. Explicit authority is inherited by descendant components. So when a user has authority to perform actions within one component, that authority is inherited for all its descendant components. Explicit authority and how it is inherited is discussed in "Granting authority to users" on page 74.

TeamConnection users also get authority to perform additional actions when they own TeamConnection objects. Authority granted by ownership is called *implicit authority*. Because this authority is inherited, you need to be careful when assigning component ownership and when granting access authority to your users.

Figure 16 shows Doug as the owner of the optics component. As owner, Doug has the implicit authority to perform most TeamConnection actions against the objects that are managed by the optics component. Doug wants Greg to be able to perform many of the same actions, so he gives him explicit releaselead authority through the component's access list. Because authority is inherited by descendant components, Doug and Greg have releaselead authority in the optics\_v and base\_h components.

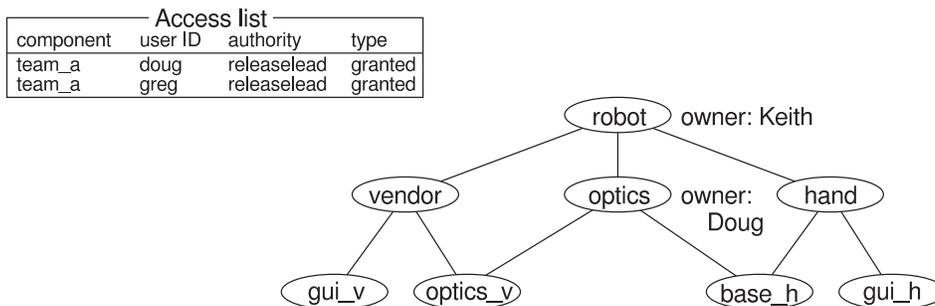


Figure 16. Granting authority to other users

Look at your component hierarchy before granting access authority. Do you want that user to have the authority to perform the same set of actions in all of the descendant

components? If the answer is no for only one or two of the components, you can restrict the user from inheriting authority for those components. See “Granting or restricting access” on page 76 for instructions on restricting access to a component. If the answer is no for many of the descendant components, you might not want to give the user that level of authority.

## What are the TeamConnection authority levels?

The authority to perform various TeamConnection actions is based on four types of authority levels: base authority, implicit authority, explicit authority, and superuser privilege. These types of authority are described as follows. For a summary of the authority required for performing TeamConnection actions, refer to the *TeamConnection User's Guide*.

### Base authority

All users defined to TeamConnection have authority to perform the following actions:

- Open defects and features
- Modify the information for their user ID
- Display information about any user ID
- Add notes to existing defects and features
- Search for information within TeamConnection to create reports (some information may be filtered out if you are not authorized to see it)

### Implicit authority

Many TeamConnection objects, such as a component, part, or defect, have an owner. The object owner automatically receives authority to perform certain actions. For example, when a defect is opened, the owner has the authority to accept the defect or reassign ownership of the defect. Similarly, the owner of a component, a release, or a feature has authority related specifically to those objects. Sometimes authority is given based on an action the user takes. For example, when someone checks a part out of TeamConnection, that person is given the authority to check it back in.

### Explicit authority

Some users need additional authority to perform actions against objects that they do not own. For example, users other than the component owner will need to check parts out of TeamConnection. The component owners give additional authority to users by adding their names to the component's access list.

When a user is assigned to be the owner of a component it is a good idea to give explicit componentlead authority to that user. In that way, the user can grant the componentlead authority to other users. Otherwise, the owner of a component cannot grant the componentlead authority to others.

See “Granting authority to users” on page 74 for more information about access lists.

### Superuser privilege

A user with TeamConnection superuser privilege can perform any TeamConnection action. Only an individual with superuser privilege can add, delete, or recreate a user ID, as well as grant superuser privilege to another user. Only a few users in your organization should have this privilege.

See page 67 for information about granting this privilege.

## What are authority groups?

There are many actions that users can perform against TeamConnection objects. It would be tedious to grant one action at a time to each of your users. Instead, you can grant a user the authority to perform a group of actions, called an *authority group*. For example, the managers of a project might want to view only the status of certain TeamConnection objects, while the developers need to view objects and also check in, check out, and extract parts. You can grant access to an authority group for either of these jobs.

The family administrator is responsible for the authority groups that your organization uses. IBM ships a set of default authority groups with TeamConnection. Determine whether these meet your needs or whether you need to change them. As your organization grows and as your needs change, you will probably want to revise your authority groups.

---

## Creating or modifying authority groups

When the database is initially created, the authority table contains the default values for the authority groups. If the default authority groups are not adequate for your development organization, you can create new authority groups or modify existing groups. Authority groups can be created or modified at any time during your development cycle.

First, decide what group of actions the intended users are required to perform. If there is a shipped authority group that closely matches your needs, you might want to modify that group. Otherwise, you will need to create a new group. To help you keep track of the authority groups that the family uses, add the name of each authority group you create to the worksheet provided in “Appendix F. Worksheets” on page 251.

We recommend you use the Family Administrator GUI to create or modify authority groups; however, if you prefer to do this manually, see page 165. Instructions for using the GUI follow.

Before you do this task, we recommend that you stop the family server (see page “Stopping the servers” on page 46 for instructions).

1. Do one of the following from a server machine to display the TeamConnection Family Administrator window:

- From the TeamConnection Group folder on the desktop, double-click on the **Family Administrator** icon.
  - Type tcadmin from a command prompt.
2. Display the family icon's pop-up menu; then select **Properties**. The properties notebook appears.
  3. Select the **Groups** page and then select the **Settings** push button under Authority to display the authority group settings.

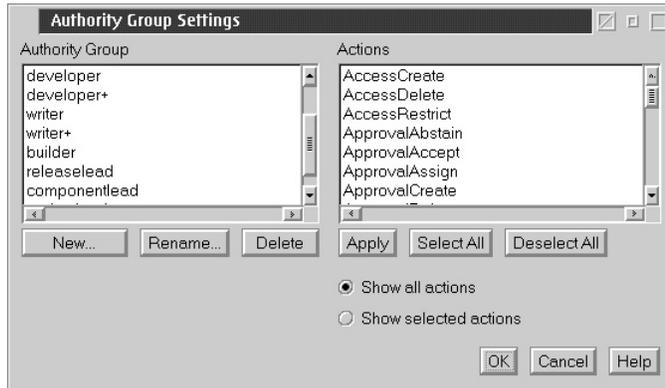


Figure 17. Authority Group Settings window

4. Do one of the following:
  - To change an existing group, highlight the group from the **Authority Group** list, and then select or deselect the appropriate actions from the **Actions** list.
  - To create a new group, follow these steps:
    - a. Select the **New** push button, type the name of the new group in New Authority Group window, and then select the **OK** push button.
    - b. From the **Actions** list, select the actions you want to include in the new group.
    - c. To save the new group, select the **Apply** push button.
  - To delete a group, select it from the **Authority Group** list and then select the **Delete** push button. When the conform delete window appears, select **Yes**.
  - To rename a group, follow these steps:
    - a. Select a group from the **Authority Group** list.
    - b. Select the **Rename** push button.
    - c. Type a new name in the **New name** field of the Rename Authority Group window, and then select the **Apply** push button.
5. When you finish making changes to your notebook pages, select **OK** to save your changes and exit from the notebook.

The changes will not take effect until you start the family server.

---

## Granting authority to users

Each component has an access list that controls access to development data. Each entry in an access list contains a user ID, the name of an authority group, and whether the authority is granted or restricted for that access group. A user whose user ID appears in the component's access list either has authority to perform any action or is restricted from performing any action listed in the specified authority group. A user ID can appear in a component's access list more than once.

There are three actions you can perform on the entries on a component's access list:

- Add a new entry to the access list. This action grants a user a certain level of authority to access the component.
- Add a new restricted entry to the access list. This action blocks a user from a certain level of authority to access the component.
- Remove an entry from the access list. This action removes a user's granted or restricted authority to access the component. This action can have one of the following effects:
  - If the user has only one entry in the component's access list and has no inherited access to the component, then all access is rescinded.
  - If the user has another entry in the access list or has inherited access to the component, then that user's access is controlled by his or her other entries in the list.

Restricting a user from an authority group is useful when a user has inherited access that you want to rescind. If user ID doug, for example, has releaselead access to component optics, and if component base\_h is a child of component optics, then doug also has inherited releaselead access to component base\_h. TeamConnection allows you to restrict doug's releaselead access to base\_h, so that doug no longer has that level of access to the component. You can also create another access list entry for doug to grant him a lower level of access, developer, for example, to base\_h. Such actions would result in an access list as follows:

Component	User ID	Authority	Type
base_h	doug	releaselead	restricted
base_h	doug	developer	granted

If you do not know what authority groups your organization uses, you can either display the groups on the Show Authority Actions window on the GUI, or ask your family administrator. Do the following from a client machine to display your authority groups:

1. Select **Lists** → **Access lists** → **Show authority actions** from the Actions pull-down menu on the Tasks window. The Show Authority Actions window appears.



Figure 18. Show Authority Actions window

2. To see a list of the actions that are contained in a group, highlight one or more group names, and then select **OK**.

Before you grant access authority to users, you should understand the following:

- Each component has only one access list.
- The authority groups in the access list must exist in the database.
- Each entry on an access list grants or restricts one user's authority to perform the actions in the specified authority group for the development data managed by that component.
- The authority granted on an access list also grants the specified authority to the user for any descendant components unless the authority has been explicitly restricted from any of those components.
- The total authority a user has is based on the combination of the different authority groups that are associated with the user. For example, a user that has been granted developer+ and writer authority can perform all the actions listed in those groups. You can create new authority groups that will build on existing groups. For example, you might create a group called creator that contains two actions: compCreate and releaseCreate. You could then grant certain users creator authority that would give them this additional authority without duplicating actions that are in their other groups.
- Only the following users can grant or restrict access authority:
  - A superuser
  - The component owner
  - Users with accessCreate or accessRestrict authority
- You cannot grant authority greater than the authority that you have for a component. For example, if you have releaselead authority for the component optics, you cannot grant componentlead authority to another user. However, if you had componentlead authority, you could grant that same authority to another user.
- A user with superuser authority can grant any authority to any user on any access list.

## Granting or restricting access

To grant or restrict access, do one of the following from a client machine:

- From the GUI:
  1. Select **Lists** → **Access lists** from the Actions pull-down menu on the Tasks window, and then select either **Add** or **Restrict**. The Add Access or Restrict Access window appears.

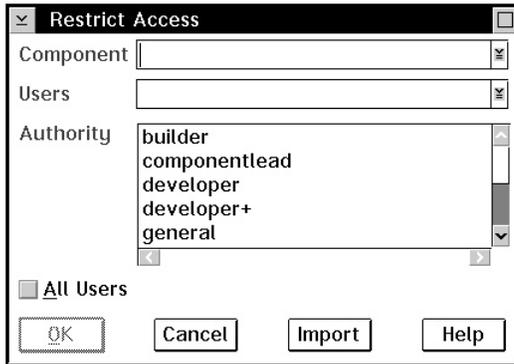


Figure 19. Restrict Access window

2. Type the name of the component and the IDs of the users, and then select the authority group that you want to add them to or restrict them from.  
For more information, select **Help** from the Add Access or Restrict Access window.
  3. Select **OK** to perform the action and close the window, or select **Apply** to perform the action and leave the window open (for the Add Access window).
- From the command line:
    - Use the access -create command to grant authority.
    - Use the access -restrict command to restrict authority.

For example, to give writer authority to a user with an ID of bruce for component robot\_dev, type the following command:

```
teamc access -create -login bruce -authority writer -component robot_dev
```

For more information about the access command, refer to the *Commands Reference* book.

## Removing an entry from an access list

To remove an entry from an access list, do one of the following from a client machine:

- From the GUI:

1. Select **Lists** → **Access lists** from the Actions pull-down menu on the Tasks window, and then select **Remove**. The Remove Access window appears.

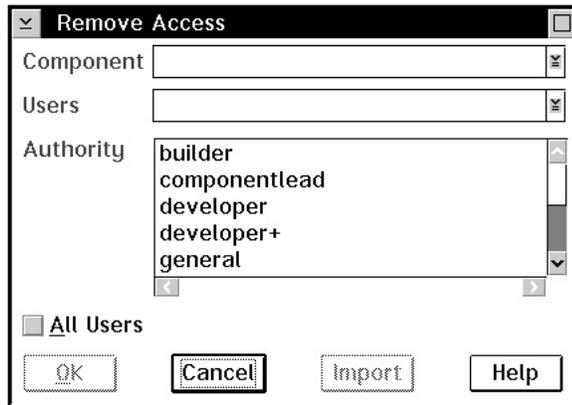


Figure 20. Remove Access window

2. Type the name of the component, the IDs of the users, and then select the authority group that you want to remove from the access list.  
For more information, select **Help** from the Remove Access window.
  3. Select **OK** to perform the action and close the window.
- From the command line:  
Use the access -delete command to remove an entry from an access list. For example, to remove the entry that grants writer authority to a user with an ID of bruce for component robot\_dev, type the following command:  

```
teamc access -delete -login bruce -authority writer -component robot_dev
```

---

## Planning for user notification

Upon request, TeamConnection notifies users when certain actions are performed on certain objects. Notification messages are sent to an electronic mailing address that is specified when the user's ID is created. (See "Setting up the mail facility" on page 42 for more information.)

Some notification is automatic. For example, a user receives notification when someone adds the user's ID to an access list.

Users can receive additional notification. For example, a manager might want to be notified whenever a defect is opened against a component. The component owner can explicitly request that TeamConnection send the manager notification.

Each component has a *notification list* that controls who is notified of what actions. Notification is inherited by descendant components. When a user is to be notified that a

specific action occurred within one component, that user will also be notified when that action occurs in any of the descendant components. Unlike authority, notification cannot be restricted for a specific component.

## What are interest groups?

There are many actions that users can be notified of. It would be tedious to request one action notification at a time for each of your users. Instead, you can request that a user receive notification for a group of actions, called an *interest group*. Each interest group is a group of actions that a certain type of user might want to be notified of.

For example, a developer might want to be notified when defects are opened or closed, while the lead developer needs to be notified not only when defects are opened, modified, or closed, but also when defects are sized or verified.

When planning for notification, you need to be familiar with what type of user is automatically notified when specific actions occur. This information is listed in a table in the *TeamConnection User's Guide*. Interest groups are composed of a subset of these TeamConnection actions.

You, the family administrator, are responsible for the interest groups that your organization uses. IBM ships a set of default interest groups with TeamConnection. Determine whether these meet your needs or whether you need to change them. As your organization grows and as your needs change, you will probably want to revise your interest groups.

---

## Creating or modifying interest groups

When the family database is initially created, the interest table contains the default values for the interest groups. If you find that the default interest groups are not adequate, you can create new interest groups or modify existing groups. Interest groups can be created or modified at any time during your development cycle.

First, decide what group of actions the intended users want to be notified of. See if there is a shipped interest group that closely matches your needs. If there is, you might want to modify that group. Otherwise, you will need to create a new group. To help you keep track of the interest groups that the family uses, add the name of each interest group you create to the worksheet provided in "Appendix F. Worksheets" on page 251.

We recommend you use the Family Administrator GUI to create or modify interest groups; however, if you prefer to do this manually, see page 167. Instructions for using the GUI follow.

Go to "Creating or modifying interest groups" on page 167 to complete this task.

Follow these steps to create or modify interest groups from the Family Administrator GUI. Before you do this task, we recommend that you stop the family server (see page “Stopping the servers” on page 46 for instructions).

1. Do one of the following from a server machine to display the TeamConnection Family Administrator window:
  - From the TeamConnection Group folder on the desktop, double-click on the **Family Administrator** icon.
  - Type `tcadmin` from a command prompt.
2. Display the family icon’s pop-up menu, then select **Properties**. The properties notebook appears.
3. Select the **Groups** page and then select the **Settings** push button under Interest to display the interest group settings.

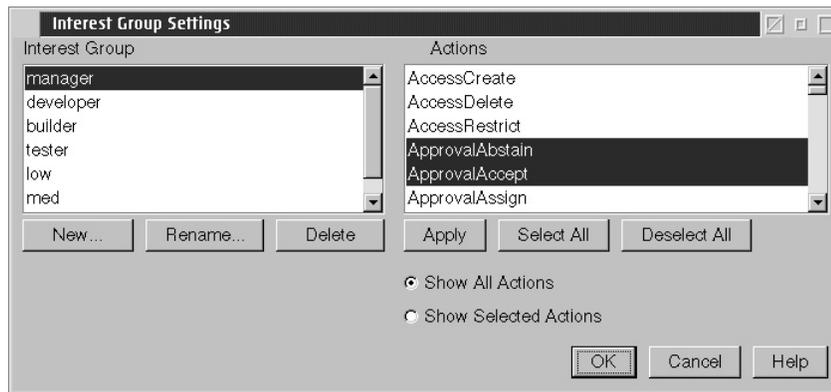


Figure 21. Interest Group Settings window

4. Do one of the following:
  - To change an existing group, highlight the group from the **Interest Group** list, and then select or deselect the appropriate actions from the **Actions** list.
  - To create a new group, follow these steps:
    - a. Select the **New** push button, type the name of the new group in New Interest Group window, and then select the **OK** push button.
    - b. From the **Actions** list, select the actions you want to include in the new group.
    - c. To save the new group, select the **Apply** push button.
  - To delete a group, select it from the **Interest Group** list and then select the **Delete** push button. When the confirm delete window appears, select **Yes**.
  - To rename a group, follow these steps:
    - a. Select a group from the **Interest Group** list.
    - b. Select the **Rename** push button.

- c. Type a new name in the **New name** field of the Rename Interest Group window, and then select the **Apply** push button.
5. When you finish making changes to your notebook pages, select **OK** to save your changes and exit from the notebook.  
The changes will not take effect until you start the family server.

---

## Working with notification lists

Each component has a notification list that controls who gets notified of what actions. Each entry in a notification list contains a user ID and the name of an interest group. An interest group defines the actions that each user in the group is to be notified of.

Before working with notification lists, you should understand the following:

- Each component has only one notification list.
- The interest groups listed in the notification list must exist in the database.
- The total notification a user has is based on the combination of the different interest groups that are associated with the user. For example, a user that has been granted med and builder notification will receive notification on actions listed only in those groups.

You can create new interest groups that build on existing groups. For example, you might create a group called size that contains two actions: defectSize and featureSize. You could then add certain users to the size group for a component to give them this additional notification without duplicating actions that are in their other groups.

- Each entry on a notification list ensures that the user will be notified when those actions in the specified interest group occur.
- The user receives notification when actions in the specified interest group occur in any descendant components.
- You cannot restrict notification as you can access authority.

## Displaying interest groups

Before adding users to notification lists, you need to know what interest groups your organization uses. To see a list of groups, do the following from a client machine to display your interest groups on the GUI:

1. Select **Lists** → **Notification lists** → **Show interest actions** from the Actions pull-down menu on the Tasks window. The Show Interest Actions window appears.



Figure 22. Show Interest Actions window

2. To see a list of the actions that are contained in a group, highlight one or more group names, and then select **OK**.

## Adding an entry to a notification list

To add an entry to a component's notification list, do one of the following from a client machine:

- From the user interface:
  1. Select **Lists** → **Notification lists** → **Add** from the Actions pull-down menu on the Tasks window. The Add Notification window appears.

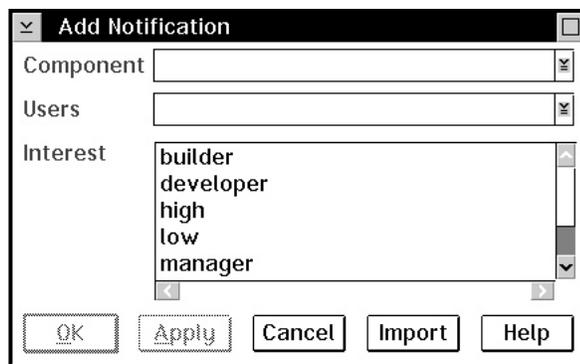


Figure 23. Add Notification window

2. Type the name of the component and the IDs of the users, and then select the interest group that you want to add them to.  
For more information, select **Help** from the Add Notification window.
  3. Select **OK** to exit the window or select **Apply** when you want to add the users to another interest group immediately.
- From a command line, use the notify -create command. For example, to add notification to the developer interest group in the robot\_dev component for the owners of user IDs korn and kotora, type the following:

```
teamc notify -create -login korn kotora -interest developer
-component robot_dev
```

For more information about the notify command, refer to the *Commands Reference* book.

## Removing an entry from a notification list

To remove an entry from a component's notification list, do one of the following from a client machine:

- From the user interface:
  1. Select **Lists** → **Notification lists** → **Remove** from the Actions pull-down menu on the Tasks window. The Remove Notification window appears.

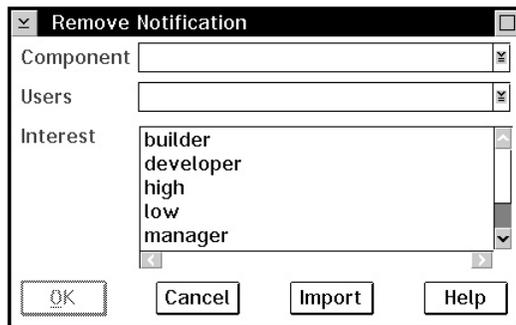


Figure 24. Remove Notification window

2. Type the name of the component and the IDs of the users, and then select the interest group that you want to remove them from.  
For more information, select **Help** from the Remove Notification window.
  3. Select **OK** to perform the action and close the window or select **Apply** to perform the action and leave the window open.
- From a command line, use the notify -delete command. For example, to remove notification from the developer interest group in the robot\_dev component for the owners of user IDs korn and kotora, type the following:

```
teamc notify -delete -login korn kotora -interest developer
-component robot_dev
```

## Chapter 7. Working with configurable fields

Many of the attributes for defects and features are configurable. TeamConnection allows you to customize them so that they more closely match the needs of your development environment. Some examples of attributes that you can customize include the following:

### Defects

prefix, phaseFound, phase inject, priority, symptom, target

### Features

prefix, priority, target

“Appendix B. Configurable field types” on page 181 contains complete lists of attributes that you can customize. These are referred to as configurable fields.

You can also create and add your own configurable fields to defects, features, parts, releases, work areas, and users. For example, you might want to add a field called PublImpact to defects and features. Programmers can then use this field to notify the writing team as to whether or not a defect or feature affects the accuracy of the product documentation.

TeamConnection uses two types of objects to make configurable fields work: configurable field types and configurable fields.

A *configurable field type* defines possible values for a field, the default value, and a description of each value. Configurable field types are required only when you want to specify a list of possible values for a configurable field or a default value. They are not required if you want to create a configurable field for text entry. The configurable field type, priority, for example, which is shipped with TeamConnection, is defined as follows. This configurable field type is used to define the values for the priority field for features and defects.

Table 19. Definition of configurable field type priority

Possible values	Description
mustfix	Defect or feature must be resolved in this release
candidate	Defect or feature is a candidate if time permits
deferred	Defect or feature deferred to next release
easy	Defect or feature is easy to solve or implement
moderate	Defect or feature is moderately difficult to resolve
difficult	Defect or feature is difficult to solve or implement
n/a	Priority does not apply to this defect or feature

A *configurable field* defines how the configurable field type is to be used by the object for which it is defined. A configurable field definition includes the following information:

- Whether or not the field is active
- Whether it is required or optional
- Whether you can set its value on Create or Open windows, or just modify it on Modify Property windows
- The configurable field type that defines the possible values for the field, if a field can have one value from multiple choices
- The attribute name to be used for the field on TeamConnection commands
- The label to be used for the field on Create, Open, and Modify Property windows
- The title to be used for the field in reports and in Features, Defects, Parts, Releases, Users, and Work Areas windows
- Whether the owner or originator of defects, features, parts, and users can modify the field
- Whether the field is included on Defect Accept or Feature Accept windows
- Dependent relationships between configurable fields

Features and defects use the priority field type differently, for example, and the difference between the two is determined by how the configurable field is defined. The following table shows how options set for priority differ for features and defects.

*Table 20. Definition of priority field in features and defects*

Option	Features	Defects
<b>Active</b>	Yes	Yes
<b>Required</b>	No	No
<b>Allow on Create/Open</b>	No	Yes
<b>Field Type</b>	priority	priority
<b>Attribute</b>	priority	priority
<b>Field Label</b>	Priority	Priority
<b>Title</b>	Priority	Priority

If you want to create a text-entry configurable field that does not require a list of possible values or a default value, then you can create a configurable field without specifying a configurable field type. If you do want to specify a list of possible values or a default value, then you must specify a configurable field type when you create the field. If you do not want to use an existing field type, you must create the type before you create the field.

For example, if you create a PubImpact field, you might want to create a new configurable field type called PubImpact (the configurable field type can have the same name as the field). If you assign the attributes of yes, no, and maybe to this field, writers can use the GUI interface or issue a TeamConnection command to see a list of all defects or features that affect the publications. If you also add a value of done, the writers can indicate when they have finished updating the documentation. The following is an example of a TeamConnection command to query such a field.

```
teamc report -view DefectView -where "PubImpact in ('done')"
```

---

## Defining configurable field types

For the defect and feature tables, TeamConnection ships configurable field types that have defined values. You can use the default field types as is or change their attributes. For example, TeamConnection defines a field named `Severity`. This field has valid values of 1, 2, 3, and 4 (see the table on page 181). You could add an additional value of 5, or you could change the description of what the value 2 represents.

**Note:** The maximum size for the value of a configurable field type is 15 characters (single-byte or double-byte).

You can also create new configurable field types for the defect, feature, part, release, work area, and user tables. This allows you to structure problem tracking information for your development environment. You can create new types or change the attributes of the existing types at any time during your development cycle. TeamConnection stores configurable field types in a file called `config.ld`. “Defining configurable field types” on page 170 describes the `config.ld` file.

It is recommended that you use the Family Administrator GUI to create or modify configurable field types; however, if you prefer to do this manually, see page 170. Always backup your database before adding or changing configurable fields or configurable field types. Instructions for using the GUI follow.

**Note:** The Family Administrator GUI does not support configurable fields for `TargetView` and `ConfigPartView`. To add configurable fields to these views, follow the instructions in “Updating `TargetView` and `ConfigPartView`” on page 174.

Follow these steps to create or modify configurable field types from the Family Administrator GUI. Before you do this task, stop the family server (refer to page “Stopping the servers” on page 46 for instructions).

1. Do one of the following from a server machine to display the TeamConnection Family Administrator window:
  - From the TeamConnection Group folder on the desktop, double-click on the **Family Administrator** icon.
  - Type `tcadmin` from a prompt.
2. Display the family icon’s pop-up menu; then select **Properties**. The properties notebook appears.
3. Select the **Configurable Fields** page.
4. Under the section labeled **Configurable Field Types**, select the **Settings** push button to open the Field Type window. The following figure shows the Field Type window.

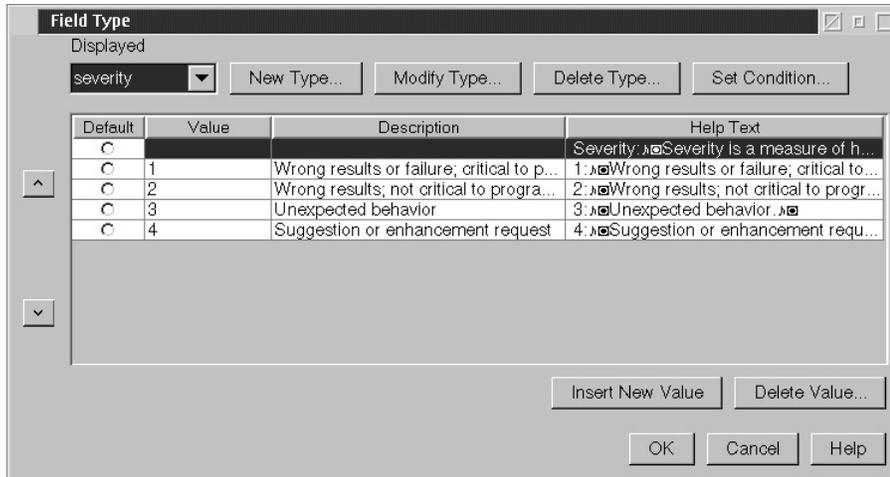


Figure 25. Field Type window

5. From this window, you can do the following:
  - To create a new field type, follow these steps:
    - a. Select the **New Type** push button beside the **Displayed** list to open the New Field Type window. Type a name for the new field type.
    - b. Use the **Resolve by Matching** radio buttons to determine the rules for specifying values for the field type. Select one of the following:
      - A unique name value**  
Only one value can be specified for the field.
      - Multiple name values**  
One or more values can be specified for the field.
      - An Expression**  
The value specified must match specific rules (6–digit numeric value, for example) defined for the configurable field type.
    - c. Use the **Help Text** field to enter help information for the configurable field type.
    - d. When you have entered the field name, resolution option, and help text select the **OK** push button.
  - To change an existing field type, select it from the **Displayed** list and then select the **Modify Type** push button. The Modify Type window opens. This window contains the same fields as the New Type window. Once you have changed the configurable field type, select the **OK** push button.
  - To delete an existing field type, select it from the **Displayed** list, and then select the **Delete** push button.

- To define values for a field type, select the field type from the **Displayed** list then select the **Insert New Value** push button. This push button adds a new line to the table. Complete the fields on this table as follows:
  - a. To set one of the values you have defined as the default, select the radio button in the **Default** field.
  - b. Type the value in the **Value** field.
  - c. Type a description for the value in the **Description** field.
  - d. Type help text for the value in the **Help Text** field.

Repeat these steps for each possible value for the configurable field. Configurable field values cannot exceed 15 characters and cannot contain spaces. If you want to enable users to set this configurable field type to the value null, include the value null in this list of possible values.

- To delete a value for a field type, select it from the table and then select the **Delete Value** push button.
  - To change the order of values in the table, select a value and then select the up and down arrow buttons to change its order.
6. When you finish making changes in the Field Type window, select **OK** to save your changes and close the window.

---

## Defining dependent relationships between configurable field types

You can set up dependencies between configurable field types. A dependency relationship between configurable field types enables you to use one configurable field type to constrain the values that can be set for another configurable field type. In this type of relationship, one configurable field type is the driver and the others are its dependents.

One example of using dependent configurable field types is for setting values for state and city. You can define a configurable field type called *state* as a driver. The range of values that can be set for its dependent field type, *city* is constrained to cities that are located in the state to which the field *state* has been set.

**Note:** You can define dependent relationships between any shipped configurable field type except the defect configurable fields, prefix and severity and the feature configurable field, prefix. These can be defined as driver types only.

To define a dependent relationship between two configurable field types, follow these steps:

1. Do one of the following from a server machine to display the TeamConnection Family Administrator window:
  - From the TeamConnection Group folder on the desktop, double-click on the **Family Administrator** icon.
  - Type tcadmin from a prompt.

2. Display the family icon's pop-up menu; then select **Properties**. The properties notebook appears.
3. Select the **Configurable Fields** page.
4. Under the section labeled **Configurable Field Types**, select the **Settings** push button to open the Field Type window.
5. On the Field Type window, select the **Set Condition** push button. The following figure shows the Set Condition window.

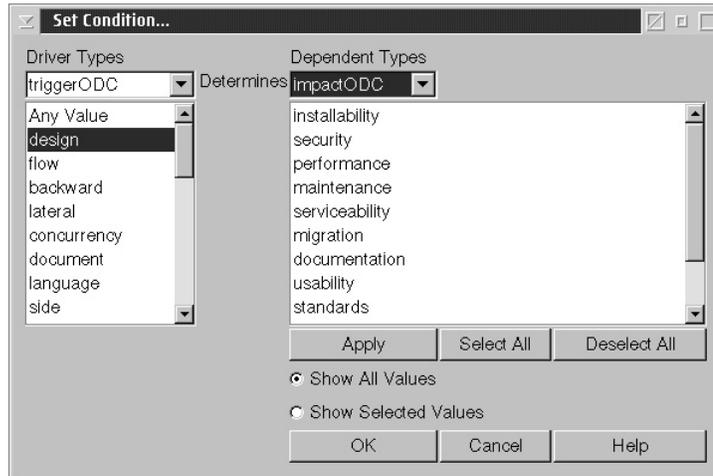


Figure 26. Set Condition window

6. From the Driver Types field, select a configurable field type to be the driver; then, in the list box below the Driver Types field, select the specific value of the field that will control the values of the dependent type.
7. From the Dependent Types field, select the dependent configurable field type; then, in the list box below the Dependent Types field, select the values that are valid for the driver value you selected. Select the **Apply** button to save your changes.
8. Repeat these last two steps until you have set up all the dependent relationships you need, then select the **OK** push button to save your changes.

---

## Defining configurable fields

Default configurable fields are shipped by IBM and are installed when the TeamConnections server is configured. If you do not want to use these defaults, you can change them at any time after the family database is created.

The following conditions apply to the use of configurable fields:

- Defect, feature, part, release, work area, and user objects can have up to 20 configurable fields each.

- Fields for defect and feature objects are effective on open, accept, and modify actions. Fields for part, release, work area and user objects are effective on create and modify actions.
- You can use the data from configurable fields to search the database and display information in reports, but TeamConnection does not use the data. For example, if you have a field called PubImpact, TeamConnection cannot change the state of a defect based on the value of this field, but users can sort all defects and features by whether or not they impact the publications.
- When you add fields, TeamConnection displays them on the GUI like any predefined field. However, the help information for configurable fields for the GUI and the commands do not reflect your new or changed fields.
- Whenever you create or modify a configurable field, your client users need to do one of the following to make the new field appear on the GUI:
  - Close the GUI and reopen it
  - Use the settings notebook to change the family and then change it back

Refreshing the GUI in this way is particularly important if the new field is required. Otherwise, your users will receive errors, but will have no way to enter information in the required fields.

- The data type for all configurable fields is character.

## Creating and modifying configurable fields

We recommend you use the Family Administrator GUI to create or modify configurable fields; however, if you prefer to do this manually, see “Appendix A. Family administration commands” on page 163. Instructions for using the GUI follow.

Follow these steps to create or modify configurable fields from the Family Administrator GUI. Before you do this task, stop the family server (refer to page “Stopping the servers” on page 46 for instructions).

1. Do one of the following from a server machine to display the TeamConnection Family Administrator window:
  - From the TeamConnection Group folder on the desktop, double-click on the **Family Administrator** icon.
  - Type `tcadmin` from a prompt.
2. Display the family icon’s pop-up menu; then select **Properties**. The properties notebook appears.
3. Select the **Configurable Fields** page.  
The Configurable Fields section of this page has a list box from which you can select the following objects: Defect, Feature, Part, Release, Work Area, and User.
4. Select one of these objects and then select the **Settings** push button to open the configurable field settings for that object.

This window has three notebook pages that you can use to do the following:

**Fields** Define configurable fields to be used for the object.

**Table** Define the table report for the object.

**Stanza** Define the stanza report for the object.

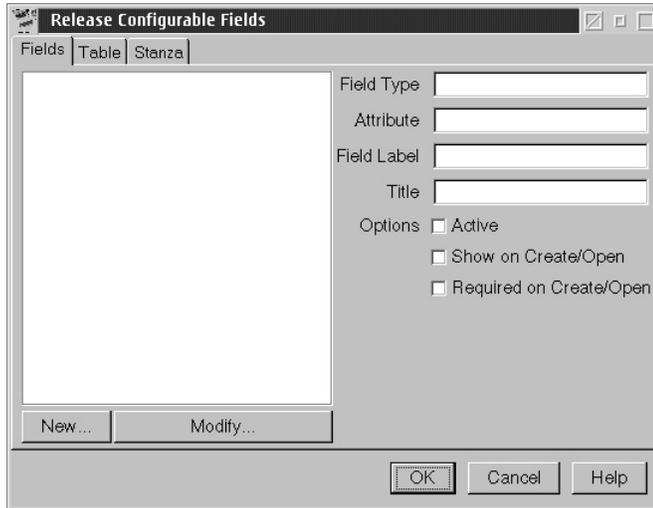


Figure 27. Release Configurable Fields window

5. Select the **Fields** page of the configurable fields window.  
From this page, you can do the following:
  - To create a new configurable field, follow these steps:
    - a. Select the **New** push button under the list of configurable fields to open the New Field window.

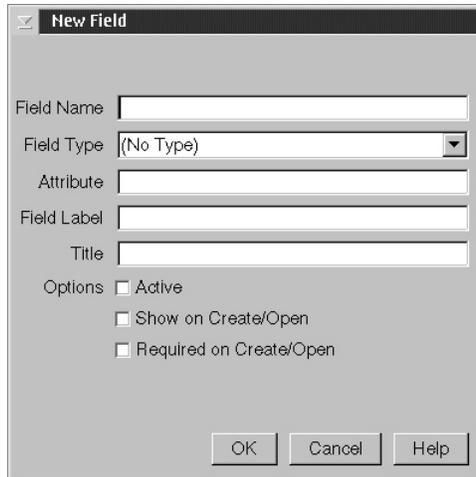


Figure 28. New Field window

- b. Type the new name in the **Field Name** entry field. The name you enter cannot exceed 15 characters and cannot contain spaces.
- c. Set options for the new configurable field by selecting or entering information in the remaining fields in this window.

#### Field Type

To define a text-entry field, select **(No Type)**. To use a predefined configurable field type, select one from this list. This field creates a pointer to the entries in config.Id that define the configurable field type you enter here. TeamConnection retrieves the possible values for the configurable field from config.Id. See “Defining configurable field types” on page 85 for instructions on creating a configurable field type.

#### Attribute

Type the name to be used for this field on TeamConnection line commands.

#### Field Label

Type the name to be used for this field on the Create, Open, Filter, or Modify Properties window.

**Title** Type the name to be used for this field on the Defects, Features, Parts, Releases, Users, or Work areas window.

The remaining fields appear on the window only if required or allowed by the object. For releases, for example, the New Field window contains the fields, Active, Show on Create/Open, and Required on Create/Open. For defects, the window contains all of the fields in these fields.

**Active** Select this checkbox to activate the new configurable field. If this checkbox is not selected, the field does not appear on the GUI.

**Required on Create/Open**

Select this checkbox to make the new field required on Create and Open windows. If this checkbox is not selected the field is optional.

**Note:** A configurable field that is required, and has a field type with a default value set behaves like an optional field because if no value is given, a default has already been specified.

**Required on Accept**

For defect or feature fields that are not required for open actions, select this checkbox to indicate the field is mandatory for accept. If a value is mandatory for accept and has been provided on the open command or on a modify command before the accept, it need not be respecified with the accept command. If a defect or feature is opened and, prior to being accepted, a field is added or modified to be mandatory on open, the field will be required for the accept action.

**Show on Create/Open**

Select this checkbox to add the field to the Create or Open windows for defects, features, parts, releases, work areas, or users. If this checkbox is not selected the field appears only on the Modify Properties windows.

**Owner modifiable**

Select this checkbox to enable the owner of the object to modify the field. For parts, the owner is the component owner; for users, the owner is the user. This field is not applicable to work areas and releases.

**Originator modifiable**

Select this checkbox to enable the originator of the object to modify the field. It is applicable to defects and features.

- To change information about a field, select it from the list and then select the **Modify** push button to open the Modify Field window. This window contains the same fields as the New Field window, described above.

Select **Help** for more information about using this window.

As noted earlier, the number of configurable fields that you can have is limited.

6. When you finish making changes to your notebook pages, select **OK** to save your changes and exit from the notebook.

The changes do not take effect until you start the family server.

## Displaying configurable field properties

To display the properties of the active configurable fields for an object, type one of the following commands from a prompt:

- `teamc defect -configInfo -family familyName [-raw]`
- `teamc feature -configInfo -family familyName [-raw]`

- teamc part -configInfo -family *familyName* [-raw]
- teamc release -configInfo -family *familyName* [-raw]
- teamc user -configInfo -family *familyName* [-raw]
- teamc workarea -configInfo -family *familyName* [-raw]

These commands show you exactly what has been defined and let you verify that the fields were loaded correctly.

If the -raw flag is used, the information is organized in a fixed ASCII table format as follows:

```
Field Label|Title|Attribute|DB Column Name|Create|Required|Field Type|
OwnerMod|OrigMod|ShowOnAccept|ReqOnAccept|Driver
```

**Note:** The properties of both the Prefix and Severity configurable fields are displayed for defects, whereas only the Prefix field is displayed for features.

---

## Changing report formats

TeamConnection users can view or print reports about an object. When you create a field, TeamConnection adds the new field to the report. You can choose the field information that you want to present to the user and the place on the report where the information appears.

Reports are displayed in two formats: stanza and table. The following sections describe these formats and explain how you can change them using the Family Administrator GUI.

**Note:** The Family Administrator GUI does not support modifying reports for TargetView and ConfigPartView. To modify these reports, follow the instructions in “Updating TargetView and ConfigPartView Reports” on page 178.

### The stanza report

Figure 29 on page 94 shows an example of a stanza report. Each line in the report represents one or more attributes of the object. To display the report from a command prompt, type the following command. This example displays information about defect 3168.

```
teamc report -view DefectView -where "name='3168'" -stanza
```

From the GUI, select **Defects** → **View** from the Actions pull-down menu. When the View Defect Information window appears, type the defect name.

The following is an example of a stanza report:

prefix	d		
name	3168		
reference	testcase_099		
abstract	Compilation error occurred.		
duplicate			
state	open	priority	
severity	2	target	driver_020
age	9		
compName	demoComponent	answer	
release	demoRelease	symptom	compile_failed
envName		phaseFound	prototyping
level	level_019	phaseInject	
addDate	93/04/01 11:32:47	assignDate	93/04/04 18:45:41
lastUpdate	93/04/13 11:54:15	responseDate	93/04/03 11:29:59
endDate			
ownerLogin	annHar	originLogin	martin
ownerName	Ann Harrison	originName	Martin Karland
ownerArea	Development	originArea	Testing
developer	johnDoe		

Figure 29. Sample stanza report displayed after adding configurable fields

We recommend you use the Family Administrator GUI to change the stanza report formats; however, if you prefer to do this manually, see page 174. Instructions for using the GUI follow.

Follow these steps to change the position of the field, or to change or delete the format specification from the Family Administrator GUI. Before you do this task, stop the family server (refer to page “Stopping the servers” on page 46 for instructions).

1. Do one of the following from a server machine to display the TeamConnection Family Administrator window:
  - From the TeamConnection Group folder on the desktop, double-click on the **Family Administrator** icon.
  - Type tcadmin from a prompt.
2. Display the family icon's pop-up menu; then select **Properties**. The properties notebook appears.
3. Select the Configurable Fields page.
4. Select the object whose report format you want to change and then select the **Settings** push button.
5. Select the **Stanza** page of the configurable fields window.

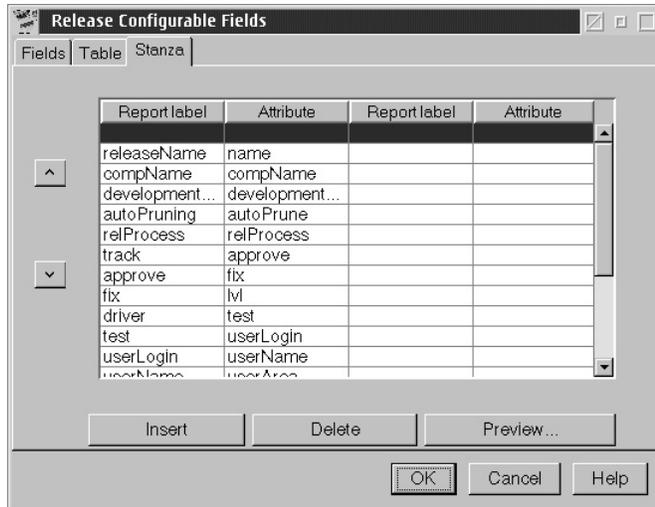


Figure 30. Stanza View Format Settings

6. Do one of the following:
  - The columns labeled **Report label** are input fields. Type in these fields to change the report labels.
  - The columns labeled **Attribute** specify the values you want displayed. To change the value for a specific field, do the following:
    - a. Select the field.
    - b. Select the drop-down button to display a list of acceptable values.
    - c. Select the value you want to use.
  - Use the **Insert** push button to add blank lines into which you can insert new fields.
  - Use the **Delete** push button to remove a field from the report. This button removes all fields on the selected line. If the line defines fields in columns one and three, then both fields are deleted.
7. Use the up and down arrow buttons to change the position of a line.
8. Use the **Preview** push button to see how the report will look when displayed in the TeamConnection GUI.
9. When you finish making changes to your notebook pages, select **OK** to save your changes and exit from the notebook.

**Note:** When changing the Part stanza format, only the partView table is changed, not the partFullView table. To manually change the partFullView table, see page 174.

Refer to Appendix A in the *Commands Reference* if you are not familiar with the differences between these two views.

The changes do not take effect until you start the family server.

## The table report

Figure 31 is an example of a table report. Each row of the table represents a different object, and the value of each attribute for that object is displayed in columns. The following is an example of the command to display the report.

```
teamc report -view DefectView -where "name='3168'" -table
```

The following is an example of the table format for defects. This example shows a table report to which a developer field has been added.

pref name	compName	state	originLo	ownerLog	sev	age	prio	abstract	developer
d	3168	demoComponent	open	martin	annHar	2	009	Compilation error	johnDoe

Figure 31. Sample table report displayed after adding configurable fields

It is recommended that you use the Family Administrator GUI to change the table formats; however, if you prefer to do this manually, see page 174. Instructions for using the GUI follow.

Follow these steps to change the columns you want displayed, their position in the table, or the width of the columns. Before you do this task, stop the family server (refer to page “Stopping the servers” on page 46 for instructions).

1. Do one of the following from a server machine to display the TeamConnection Family Administrator window:
  - From the TeamConnection Group folder on the desktop, double-click on the **Family Administrator** icon.
  - Type `tcadmin` from a prompt.
2. Display the family icon's pop-up menu; then select **Properties**. The properties notebook appears.
3. Select the Configurable Fields page.
4. Select the object whose report format you want to change and then select the **Settings** push button.
5. Select the **Table** page of the configurable fields window.

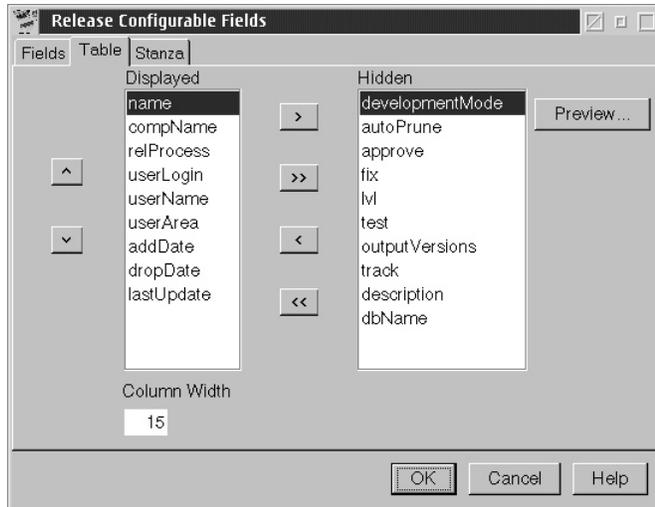


Figure 32. Table View Format Settings

6. Do one of the following:
  - To change the position of a column, select it and then select the up or down arrow repeatedly until the column name is in the correct position.
  - To remove a column from the table, select the column label from the **Displayed** list box and then select the right arrow button to move the column to the **Hidden** list box.
  - To add a column to the table, select the column label from the **Hidden** list box and then select the left arrow button to move the column to the **Displayed** list box. The label is placed at the bottom of the list; you can change its position.
  - To move all columns between the **Displayed** and **Hidden** list boxes, use the double-left and double-right arrow buttons.
  - To change the width of a column, select the column label and then select the up or down arrow in the **Column width** field.
7. When you finish making changes to your notebook pages, select **OK** to save your changes and exit from the notebook.

**Note:** When changing the Part table format, only the partView table is changed, not the partFullView table. To manually change the partFullView table, see page 174.

Refer to Appendix A in the *Commands Reference* if you are not familiar with the differences between these two views.

The changes do not take effect until you start the family server.



---

## Chapter 8. Configuring family processes

TeamConnection is shipped with several predefined processes for both components and releases. If these processes do not meet the needs of your development organization, you can create your own processes. You do this by combining some of the predefined subprocesses that IBM provides.

This chapter explains how you configure new processes or change existing processes. If you are not familiar with TeamConnection processes and how they are used, read “Planning your processes” on page 53.

---

### Planning your changes

Consider the following before configuring your processes:

- To avoid confusion for end users, do not modify processes after they are in use. If you must add or delete subprocesses in an existing process, consider instead creating a new process.
- Do not delete a process that is being used by any component or release in your family.
- Changes to processes do not take effect until one of the following occurs:
  - A component or release is modified using the process name
  - A component or release is created using the process name

The processes shipped by IBM are explained to your users in on-line help. Any processes you configure are not explained in on-line help. Therefore, you will need to provide this type of information to your users.

Tables are provided in “Configurable processes worksheets” on page 261 for you to record the processes you configure.

---

### Modifying or creating configurable processes

Your first step in configuring a process is to give the process a name. The name you choose can be up to 15 characters in length with no blanks, tabs, or vertical separators. To help you keep track of the processes that you configure for your family, add the name of each process you create to the worksheet provided in “Configurable processes worksheets” on page 261.

We recommend you use the Family Administrator GUI to modify or create configurable processes; however, if you prefer to do this manually, see page 168. Instructions for using the GUI follow.

Follow these steps from a server machine to configure component or release processes from the Family Administrator GUI. Before you do this task, we recommend that you stop the family server (refer to page “Stopping the servers” on page 46 for instructions).

1. Do one of the following to display the TeamConnection Family Administrator window:
  - From the TeamConnection Group folder on the desktop, double-click on the **Family Administrator** icon.
  - Type tcadmin from a command prompt.
2. Display the family icon’s pop-up menu, then select **Properties**. The properties notebook appears.
3. Select **Processes**.

This page of the family properties notebook provides access to two windows: one for defining release processes and one for defining component processes. To open one of these windows, select one of the **Settings** push buttons.

Use the **Release Process Settings** window and the **Component Process Settings** window to define processes and subprocesses for releases and components defined in your family. Complete the fields on these windows as follows.

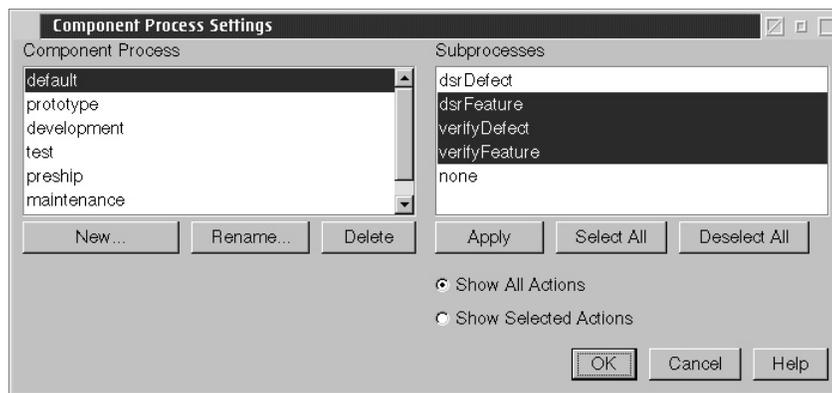


Figure 33. Component Process Settings window

- To see the default subprocesses defined for each release or component process, select a process name from the **Release Process** or **Component Process** list. The subprocesses included will appear highlighted in the **Subprocesses** list.
- To add or delete subprocesses for an existing process, follow these steps:
  - a. Select a process from the **Release Process** or **Component Process** list.
  - b. To add or delete a subprocess, select it from the **Subprocesses** list.
  - c. To save your changes, select the **Apply** button.
- To create a new process, follow these steps:

- a. Select the **New** push button, type the name of the new process in New Release Process or New Component Process window, and then select the **OK** push button.
- b. From the **Subprocesses** list, select the subprocesses you want to include in the new process.
- c. To save the new process, select the **Apply** push button.
- To delete a process, select it from the **Release Process** or **Component Process** list and then select the **Delete** push button. When the confirm delete window appears, select **Yes**.
- To rename a process, follow these steps:
  - a. Select a process from the **Release Process** or **Component Process** list.
  - b. Select the **Rename** push button.
  - c. Type a new name in this **New name** field of the Rename Release Process or Rename Component Process window, and then select the **Apply** push button.
4. When you finish making changes to your notebook pages, select **OK** to save your changes and exit from the notebook.

The changes do not take effect until you start the family server.



---

## Chapter 9. Providing user exits

TeamConnection provides a highly configurable set of processes so that you can adapt the tool to your specific needs. However, there are many cases where you might want to make further process adjustments or add automated steps. User exits allow you to enhance the TeamConnection processes to perform tasks like the following:

- Ensuring that code files meet formatting requirements, such as the inclusion of keywords that identify the name and version of the file
- Creating a new defect when a verification record is rejected on a current defect
- Analyzing a build failure and removing any work areas from a driver that include changes to files that failed to compile (if the build fails the driver remains in restrict state, however if it succeeds the driver can be committed)
- Ensuring that the right information is in a sizing record before it is accepted
- Automatically generating management reports when a driver is committed

This chapter describes user exits, how to use them, and how to implement them for each TeamConnection family. User exits are not necessary for the operation of TeamConnection; they are optional and can be configured for each family.

With user exits, you can specify additional actions that you want performed before completing or proceeding with a specific TeamConnection command action. A user exit enables the TeamConnection server to call a user-defined program during the processing of TeamConnection actions. The program can be an executable file or a command file. Thus, you can use TeamConnection as a trigger to start non-TeamConnection processing. You can also use user exits to restrict certain TeamConnection actions based on external considerations. For example, you might have a user exit scan C source files to ensure that the source code conforms to the standards defined by your development process.

The userExit file indicates the programs you want started for specific TeamConnection actions. You can add entries to this file using the Family Administrator GUI, which is the recommended method, or directly edit the file. This chapter explains how to set up user exits using the Family Administrator GUI. For instructions on editing the userExit file directly, see “Setting up user exits” on page 178.

**Note:** The userExit file is copied to your family database directory from a file located in the language subdirectory of the nls\cfg directory path in the TeamConnection installation directory, for example, teamc\nls\cfg\enu. The version of the userExit file in this location contains comments that are not copied when the family is created using the Family Administrator GUI.

User exits are provided for most TeamConnection actions. The actions that support user exits are listed in “Appendix C. User exit parameters” on page 191.

## Writing user exit programs

User exit programs have the following behavior:

- You can call an exit at the following times:

For the user exit program to...	Select this if using the GUI	Specify this if manually updating the userExit file
Start at the beginning of the action, before any initialization or access checking takes place.	Pre-check	Exit ID 0
Start after all checks are made and TeamConnection is ready to process the command.	Pre-action	Exit ID 1
Start after action is completed and all database or library updates have been committed.	Post-action	Exit ID 2
Start when a previous user exit with an exit ID of 0 or 1 is not successful, or when the action is not successful. This user exit program can clean up what the other user exit programs started.	Failure	Exit ID 3

- Each user exit program receives a unique list of parameters, defined as follows:

### Program (UEprogram)

Name of the calling user exit.

### User-Defined Parameter (UEparameters)

The user-defined parameter in the userExit file.

### Action parameters

The parameters passed to the user exit program from the TeamConnection command. The first parameter is the name of an environment file that contains the name and value of any action parameters specified by the family administrator.

- When a user does not enter a value in an optional GUI field or command line, the user exit puts "" in its place. This is also true for UEparameters.
- Positional parameters that pass true or false values, such as **Break common link** (force flag), return the following:

**True** 1

**False** 0

- A nonzero return code from a user exit program for exit ID 0 or 1 terminates the TeamConnection command. Nonzero return codes do not affect exit ID 2 or 3.
- The userExit file is read only when the family server is started. After a user exit is enabled, you can change the exit without restarting the family server. However, you must stop and restart the server before your changes are recognized.

Follow these guidelines when you write user exit programs:

- Not all TeamConnection commands can be used in a user exit. Some may cause a database deadlock to occur.
- User exit programs do not permit user interaction (for example, from a user exit program, you cannot prompt a user with a read command).
- Define only one user exit program for each TeamConnection action and exit ID combination. If you define more than one program, TeamConnection uses the last one you define.
- You are limited to a total of 40,000 bytes of total output from all user exits, plus warnings, for each TeamConnection action (except teamc report and -view actions).
- Limit the length of time that the user exit program runs.

The Family Administrator GUI guides you through the configuring of user exits, and the following examples give you a start on writing your own user exits:

- `viewexit.c`: A c program that displays the output of a user exit, identifying the user exit command, the environment file, the user-defined parameter, and each of the positional parameters. Also, parameters that are null or were truncated because they would make the command string too long are identified. There is also a function that was derived from `teamcenv` (below) to display the contents of the environment file (if one was specified).
- `teamcenv.c`: A c program that lets you read the entire environment file and display the contents of each variable stored, or extract the value of a specific variable based on the name. It can be called from `viewexit.c`.
- `viewexit.ksh`: A version of `viewexit.c` written in Korn shell.
- `viewexit.cmd`: A version of `viewexit.c` written in REXX.

The table in “Appendix C. User exit parameters” on page 191 lists the parameters that are passed by each TeamConnection action. The table also contains descriptions of many of the parameters.

The following table provides examples of the parameters passed to a user exit program for the PartAdd action. Each parameter has a numeric identifier. The first column shows how to identify the parameter in source for an executable (such as C). The second column shows how to identify the parameter in source for a shell program (such as an OS/2 `.cmd` file or an AIX `.ksh` file). The third column describes each parameter.

*Table 21. Parameters passed to a user exit for the PartAdd action*

<b>Argument number in an executable</b>	<b>Argument number in a shell program</b>	<b>Argument value</b>
Exit ID 0		
arg[0]	\$0	User exit program name

arg[1]	\$1	User defined parameter - required field containing a string passed to a user exit program. The string must be in quotes to ensure that all subsequent parameters are in the correct position in the argument list.
arg[2]	\$2	Environment file name - null if not used
arg[3]	\$3	File path name
arg[4]	\$4	'0' if the file will not be transmitted; otherwise the file will be transmitted.
arg[5]	\$5	Client location
arg[6]	\$6	Temp file on server
arg[7]	\$7	Release name
arg[8]	\$8	Component name
arg[9]	\$9	File type: set to '2' if binary is specified at file creation, to '1' if text, or, otherwise, to '0'
arg[10]	\$10	Work area name
arg[11]	\$11	FileMode
arg[12]	\$12	Parent name
arg[13]	\$13	Parser name
arg[14]	\$13	Builder name
arg[15]	\$15	Parent relation type
arg[16]	\$16	Builder parms
arg[17]	\$17	Part type
arg[18]	\$18	Parent type
arg[19]	\$19	Temporary to build
arg[20]	\$20	Part translation
arg[21]	\$21	Effective TeamConnection user ID (TC_BECOME)
arg[22]	\$22	Real TeamConnection user ID (TC_USER)
arg[23]	\$23	Verbose flag
Exit ID 1, 2		
arg[0]	\$0	User exit program name
arg[1]	\$1	User defined parameter - should be in quotes
arg[2]	\$2	Environment file name - null if not used
arg[3]	\$3	File path name
arg[4]	\$4	Temp file on server
arg[5]	\$5	Release name
arg[6]	\$6	Component name
arg[7]	\$7	File type: set to '2' if binary is specified at file creation, to '1' if text, or, otherwise, to '0'

arg[8]	\$8	Work area name
arg[9]	\$9	Remarks
arg[10]	\$10	FileMode
arg[11]	\$11	Parent name
arg[12]	\$12	Parser name
arg[13]	\$13	Builder name
arg[14]	\$14	Parent relation type
arg[15]	\$15	Builder parms
arg[16]	\$16	Part type
arg[17]	\$17	Parent type
arg[18]	\$18	Temporary to build
arg[19]	\$19	Part translation
arg[20]	\$20	Effective TeamConnection user ID
arg[22]	\$22	Verbose flag

---

Exit ID 3

---

Same as Exit ID 0 with an additional parameter as the last parameter to indicate the last user ExitID that has been executed successfully, for example, 0 or 1.

---

If you have a user exit program called viewexit, for example, that you want to execute when a user creates a part in TeamConnection, you would include the following line in the userExit file:

```
PartAdd 1 viewexit
"1991 1992" # copyright for PartAdd with '1991 1992'
```

This program runs when TeamConnection recognizes that the user is requesting a valid PartAdd action, but before the PartAdd action actually starts. If the content of the viewexit user exit program is:

```
#include <stdio.h>
#include <string.h>
#include <sys/types.h>

extern int errno;

/* This is based on a limit used for actions in TeamC */
#define maxParmName 40

/*-----*/
envGetFromEnvFile:
- Read each entry to environment file
- Read binary:
  size of parameter, parameter string, size of value, value string
- Minimal error checking to simplify example
Note: This routine is condensed from the sample program teamcenv.c
/*-----*/
```

```

int envGetFromEnvFile(char *envFileName)
{
    FILE *envFile;

    int nNameLength;
    int nValueLength;
    char parameterName[maxParmName+1]; /* allow for maximum in TeamC (15 + NULL) */
    char parameterValue[16001]; /* allow for max in TeamC 16000 for remarks + NULL */

    /* Open temporary file */
    envFile = fopen(envFileName, "rb");
    if (envFile == NULL)
    {
        fprintf(stderr,"teamcenv: Error, could not open file \"%s\"\n",
            envFileName);
        return 1;
    }

    /* Dump all attributes */
    printf("\
Parameter                                Value\n\
=====                                =====\n");
    fread(&nNameLength, sizeof(int), 1, envFile);
    fread(parameterName, sizeof(char), nNameLength, envFile);
    *(parameterName+nNameLength)='\0';
    fread(&nValueLength, sizeof(int), 1, envFile);
    fread(parameterValue, sizeof(char), nValueLength, envFile);
    *(parameterValue+nValueLength)='\0';

    while (!feof(envFile))
    {
        strncat(parameterName, "
",
            (maxParmName - strlen(parameterName)));
        *(parameterName+maxParmName) = '\0';
        printf("%s %s\n", parameterName, parameterValue);
        fread(&nNameLength, sizeof(int), 1, envFile);
        *(parameterName+nNameLength)='\0';
        fread(parameterName, sizeof(char), nNameLength, envFile);
        fread(&nValueLength, sizeof(int), 1, envFile);
        fread(parameterValue, sizeof(char), nValueLength, envFile);
        *(parameterValue+nValueLength)='\0';
    }

    fclose(envFile);
    return 0;
}

/*-----\
main:
- Print standard set of arguments at beginning of parameter list:
  UEprogram, UEparameter and EnvFile
- Print rest of arguments; the positional parameters
- If EnvFile is not NULL, print the contents of the EnvFile
\*/

```

```

\-----*/
int main(int argc, char *argv[])
{
    int i,n;

    /* Compute the number of action parameters passed at definition */
    int totalParms = argc - 3;

    /* Display the name of the command. */
    /* It is not necessary to parse name. */
    printf("UEProgram:          %s\n", argv[0]);

    /* Display parameter list. */
    if (strlen(argv[1]) != 0)
        printf("UEParameters string:    %s\n", argv[1]);
    else
        printf("UEParameters string is NULL\n");

    /* Display parameter env file. */
    if (strlen(argv[2]) != 0)
        printf("EnvFile name:          %s\n", argv[2]);
    else
        printf("No environment file; string is NULL\n");

    /* Display parameter list. */
    for (i = 3; i < argc; i++)
    {
        n = strlen(argv[i]);
        if (n == 0)
            printf("Action Parameter [%d] is NULL\n", i-2);
        else
        {
            printf("Action Parameter [%d]:    %s\n", i-2, argv[i]);

            /* Ellipses in data, checking for end (i.e. truncation) */
            /* - Each parameter limited to 400 bytes in OS/2, Windows */
            /*   and Windows NT, and 16000 bytes in Unix */
            /* - Total command string limited to 1024 bytes in OS/2, */
            /*   Windows and Windows NT, and 32000 bytes in Unix */
            /* - compute address of last 3 characters then compare */
            if (strcmp(((argv[i])+n-3), "...") == 0)
            {
                printf("                parameter %d string was \
truncated!\n", (i-1));
                /* If last parameter truncated, then entire list truncated */
                if (i == (argc - 1))
                    printf("Parameter list was truncated!\n");
            }
        }
    }
    printf("Total action parameters: %d\n", totalParms);

    /* Print contents of parameter env file. */
    if (strlen(argv[2]) != 0)

```

```

    {
        printf("Printing contents of Environment File\n");
        envGetFromEnvFile(argv[2]);
    }

    return (0);
}

```

and the command issued by the TeamConnection client is:

```

teamc part -create src\partX.c -component codeAcomp -release ToolAvRe1
          -family testfam -workarea waABC -parent partX.obj -input -parser Cparser

```

the output displayed in the client window is:

```

UEProgram:          C:\TESTFAM\BIN\viewexit.cmd
UEParameters string: 1991 1992

No environment file; string is NULL
Action Parameter {1}: src/partX.c
Action Parameter {2}: C:\TESTFAM\BIN\TCTMP\D373\RQOESU7J.C2T
Action Parameter {3}: ToolAvRe1
Action Parameter {4}: codeAcomp
Action Parameter {5}: 1
Action Parameter {6}: waABC
Action Parameter {7}: Initial Version of src/partX.c
Action Parameter {8}: 0600
Action Parameter {9}: partX.obj
Action Parameter {10}: Cparser
Action Parameter {11}: is NULL
Action Parameter {12}: 1
Action Parameter {13}: is NULL
Action Parameter {14}: TcPart
Action Parameter {15}: TcPart
Action Parameter {16}: no
Action Parameter {17}: no
Action Parameter {18}: cmvctest
Action Parameter {19}: 1
Total action parameters:19

```

In the preceding example, no remark was specified so TeamConnection provided Initial Version of src/partX.c.

---

## Environment file

The list of positional parameters in a user exit command string can get very long. Further, it is sometimes difficult to parse through such a command. For example, in OS/2, a carriage return in a parameter prematurely ends the command string. Therefore, it is not always certain that the command string will include the parameters after a remarks parameter.

The solution to this problem is to select the variables that you will need and have their values inserted into an environment file. The environment file is a binary file that contains the names of parameters and their values, including carriage returns and anything else that has been passed along. If the use of an environment file is specified, the name of the temporary file containing the environment variable names and values is the positional parameter after the user-defined parameter.

The `viewexit.c` and `teamcenv.c` files in the `samples` directory show how to read the environment file.

“Appendix C. User exit parameters” on page 191, containing the names of all of the parameters, is also the list of names used for the environment file.

The `tcadmin` tool guides you through selecting parameters to be passed in the environment file.

---

## Setting up user exits

When you want TeamConnection to start a user exit, you must associate the user exit program with TeamConnection actions. We recommend you use the Family Administrator GUI to associate the actions with the program; however, if you prefer to do this manually, see 178. Instructions for using the GUI follow.

Follow these steps to associate user exit programs with TeamConnection actions. Before you do this task, we recommend that you stop the family server (refer to page “Stopping the servers” on page 46 for instructions).

1. Do one of the following to display the TeamConnection Family Administrator window:
  - From the TeamConnection Group folder on the desktop, double-click on the **Family Administrator** icon.
  - Type `tcadmin` from a command prompt.
2. Display the family icon’s pop-up menu; then select **Properties**. The properties notebook appears.
3. Select the **User Exits** page and then select the **Settings** push button to open the User Exit Settings window.

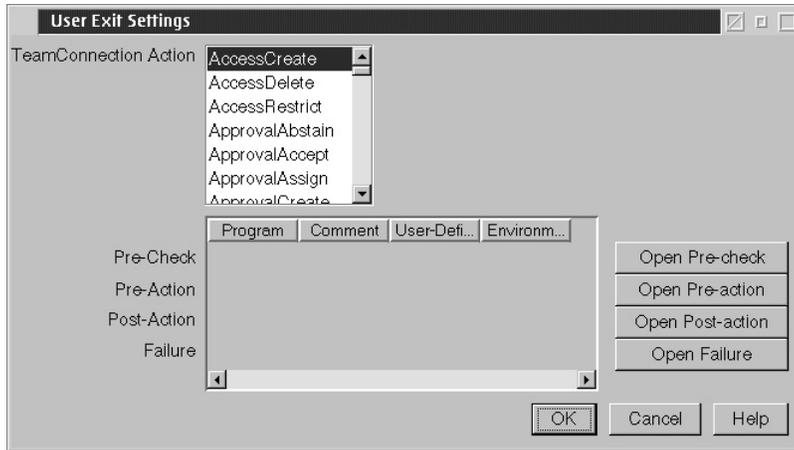


Figure 34. User Exit Settings

4. Highlight the action that you want to associate the user exit with, then select the push button at the bottom of the settings window to specify when you want the user exit to start.

**Pre-check (Exit ID 0)**

before Action checking

**Pre-action (Exit ID 1)**

before Actions processing

**Post-action (Exit ID 2)**

after Action completes

**Failure (Exit ID 3)**

if action or programs fail

Selecting one of these push buttons opens a window that you can use to specify the program and parameters to use for the user exit. The following is an example of the Pre-Check window.

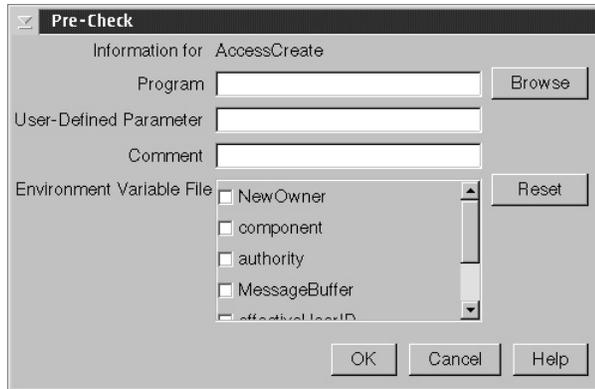


Figure 35. Pre-Check window

5. Type the name of the user exit program to be executed as well as any user-defined parameters and comments (see “Configuring user exit parameters” for more information on specifying parameters). Select the **OK** push button to save the information for each user exit you specify.
6. Repeat these steps for each user exit program that you want to start.
7. When you finish making changes to your notebook pages, select **OK** to save your changes and close the notebook.

Select **Help** for more information about using the User Exit Settings window.

The changes do not take effect until you start the family server.

---

## Configuring user exit parameters

You can configure the parameters to be passed to a user exit program for each user exit ID (0, 1, 2, or 3) as follows:

- Reorder the parameters
- Select which of the available parameters for an action are to be passed to the user exit and which are to be omitted
- Include configurable fields in the list of parameters to be passed to the user exit
- Include new fields that were previously not available

If you choose to configure the parameter list for a user exit ID, TeamConnection creates a new field in the userExit file that identifies the parameter list. Normally, a user exit definition in the userExit file appears as follows:

```
PartAdd 0 viewexit "1991 1992"
```

TeamConnection uses an ENV=( ) field to define a customized parameter list. If you want your user exit to be passed only the component and release parameters of the PartAdd action, for example, TeamConnection defines the user exit as follows:

```
PartAdd 0 viewexit "1991 1992" ENV=(component,release)
```

Any of the customizations listed at the beginning of this section will cause an ENV=( ) field to be generated for the user exit definition. The values for the customized parameters are stored in a temporary file so that they can be extracted into the user exit program.

To configure a parameter list for a user exit, follow these steps:

1. From the **User Exits** page of the Family Administrator properties notebook, set up the user exit program for the action and the exit ID as described in "Setting up user exits" on page 111.
2. Using the notebook tabs at the bottom of the User Exits page, select the tab corresponding to the user exit ID for which you defined the user exit:

**Exit ID 0**

**Pre-Check**

**Exit ID 1**

**Pre-Action**

**Exit ID 2**

**Post-Action**

**Exit ID 3**

**Failure**

To configure a parameter list for the user exit associated with the Pre-Check exit ID of the PartAdd action, for example, select **PartAdd** from the list of actions on the User Exit Settings window, then select the **Open Pre-check** button at the bottom of the page.

The following is an example of the Pre-Check window for the PartAdd action.

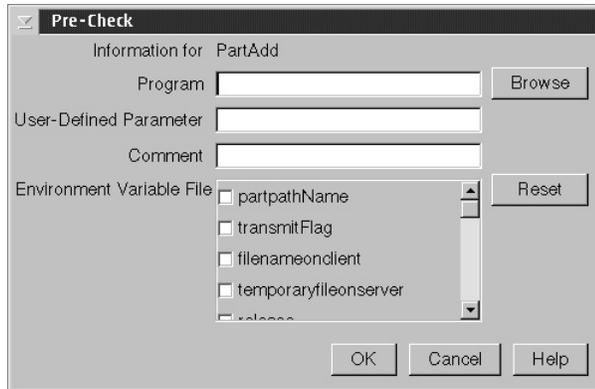


Figure 36. Pre-Check window for PartAdd

3. Set options for the Pre-Check exit ID for PartAdd as follows:

**Program**

Type the full path name of the user exit program.

**User-Defined Parameter**

Type any parameters that you want to pass to the user exit program for the action and exit ID.

**Comment**

Type a comment to be included in the userExit file with the user exit definition.

**Environment Variable File**

Select the parameters you want to use from the list. If any configurable fields have been defined, they appear at the bottom of this list. The values for these parameters will be stored in a temporary file for extracting into the user exit program.

4. Select the **OK** push button to save the options you have set.

The options that you set on this page are added to the userExit file. They take effect after you stop and restart the family server.

---

## Sample user exit programs

TeamConnection is shipped with the following sample user exit programs:

**samples/teamcenv.c**

Extracts the value of customized parameters from the temporary environment file so that they can be passed to a user exit program. You can incorporate this code into any user exit programs written in C.

**samples/viewexit.c, samples/viewexit.ksh, and samples/viewexit.cmd.**

Each of these samples shows the following information:

- Parameter 0: Executable name
- Parameter 1: User defined parameters
- Parameter 2: Environment variable file name (generated by TeamConnection and deleted automatically after the daemons are brought down)

Each sample also can output the contents of the environment variable file (viewexit.c has a simple function to dump the contents, while viewexit.cmd and viewexit.ksh have commented out lines that can call teamcenv).

The list of available user exits and their parameters can be viewed by a super user issuing the following command:

```
teamc report -userExitInfo
```

Adding the -long flag will also display the user exits currently configured and the environment variables to be written to the environment file.

---

## Chapter 10. TeamConnection shadows

A shadow is a collection of parts in a file system that reflects the contents of a work area, driver, or release. The shadow could be a simple directory structure on a network server, or a file system on a completely different computer platform. You can use shadows to build your product, or they can simply be a place where developers can go to search through code. Shadowing is similar to extracting in that the purpose of each is to provide a set of files that reflect the version of a TeamConnection object.

TeamConnection does not do shadowing all by itself. It implements a framework that requires you to provide the "shadowing program" to perform the actual file system updates. This shadowing program can be one you create yourself, or a sample that is shipped with TeamConnection. When commands are issued that change the contents or properties of a part, TeamConnection determines what needs to be updated in the shadows. TeamConnection then extracts the parts and calls the "shadowing program" to update the file system.

TeamConnection stores information about the shadowed parts in the TeamConnection database. Subsequent shadowing actions "remember" what is already in the shadow to avoid doing unnecessary updates. For example, assume a driver exists with many driver members. When a new work area is added to the driver, only those files that have been changed on the new work area are extracted and placed in the shadow.

This chapter describes shadows and how to implement them for your families. Shadow properties and shadow actions are described. The requirements and interfaces that shadowing programs must implement are also defined.

**Note:** Shadows are not necessary for the operation of TeamConnection; they are optional and can be configured for each family.

---

### Shadow types

A shadow type is an association between a name you define and a shadowing program. A shadow type has the following properties:

**name** The name you choose for the type. The name must be unique within the family.

**description** The description of the shadow type.

**program name** The program that TeamConnection will call to perform the shadowing actions. This name should be a fully qualified path to the program. The TeamConnection server must be able to access and run this program. This program can be a sample shipped with TeamConnection, or one that you write yourself.

---

## Shadow properties

Shadows have the following properties.

**name** The name of the shadow. The name must be unique within the release.

**type** A type that has been defined for the family.

**release**

The release for the shadow.

**mode** The mode of the shadow. The mode can have two possible values:

- **manual** - Shadows are updated only when explicitly requested by authorized users with the shadow **-synchronize** command.
- **synchronous** - Shadows are updated as the contents of work areas, drivers, and releases are changed.

**state** A shadow has two possible states:

- **disabled** - The shadow is not able to perform shadowing actions. A shadow can be disabled if, for example, the file system has run out of space. After the problem with the shadow has been corrected, the shadow can be enabled.
- **enabled** - The shadow is able to perform shadowing actions.

**location**

The location of the shadow. This is text that defines where each part should be placed in the shadow. The location text supports the following substitution variables.

**Note:** On UNIX platforms, to bypass the UNIX shell, use the escape character `\` with the `$` macros.

- `$N` - the name of the work area, driver, or release that is being shadowed.
- `$P` - the path name of the part being shadowed.
- `$B` - the base name of the part being shadowed.
- `$F` - the family name.
- `$R` - the release name of the part being shadowed.
- `$C` - the component name of the part being shadowed.
- `$S` - the name of the shadow.
- `$V` - the version SID of the part being shadowed.
- `$$` - used to define the literal "\$".

This property can contain any text. This information is passed to the shadowing program after the values for any variables are substituted. The contents of this property are defined, validated, and implemented by the shadowing program.

For example, the location could be specified as some directory that the TeamConnection family server can access, such as:

```
/home/tcparts/$R/$N/$P
```

or it could be a combination of machine name, port, and directory:

```
hostname.ibm.com 1300 /home/tcparts/$R/$N/$P
```

The location property should be defined so that parts from work area, driver, and release versions do not resolve to the same location. For example, if the location was specified as /tcparts/\$R/\$P, the same part in two different work areas would resolve to the same location. You should always include the name of the object (\$N) and the path (\$P) in the location to avoid this situation.

**drivers** Indicates whether the specified shadow contains drivers. Valid values are **yes** and **no**.

**work areas**

Indicates whether the specified shadow contains work areas. Valid values are **yes** and **no**.

**release**

Indicates whether the specified shadow contains release versions. Valid values are **yes** and **no**.

**crlf**

Indicates whether a crlf (carriage return / line feed) conversion should be performed. Valid values are **yes** and **no**. When this value is yes, both a "cr" and a "lf" character are used to indicate the end of a line in a file. Generally, a crlf value of yes means that the shadow is primarily used by Intel-based clients, and a value of no indicates AIX/UNIX users. This property only applies to text parts.

**keys**

Indicates whether TeamConnection keywords embedded in parts will be expanded. Valid values are **yes** and **no**. This property only applies to text parts.

**timestamp**

Indicates which timestamp should be used on files extracted from TeamConnection. This property can have the following values:

- **preserve** - The timestamp of the part will be set to the timestamp maintained by TeamConnection as set when the part was checked in. This is the same timestamp that you would see if you extracted the part.
- **current** - The timestamp of the part will be the current time. That is, the time when the part is shadowed.

**priority**

A positive integer number that indicates the shadow's priority within the release. Shadows are processed from the highest priority shadow to the lowest. This property applies only to synchronous shadows. For manual shadows, this is set to zero (0).

### parameters

Additional parameters to pass to the shadowing program. These are parameters that the shadowing program defines and knows how to implement. The format is any text. This property supports the same substitution variables that are supported by the location property. For example:

```
-component $C -version $V -language English
```

---

## Shadow actions

The following actions can be performed with the TeamConnection shadow command.

- **define** - Define a shadow type.
- **redefine** - Update the properties of an existing shadow type.
- **undefine** - Delete a shadow type.
- **create** - Create a shadow of a specific type for a release.
- **modify** - Modify the properties of a shadow.
- **disable** - Disable a shadow. This will temporarily turn a shadow off.
- **enable** - Enable a previously disabled shadow.
- **delete** - Delete a shadow. This action only affects the shadowing information in the TeamConnection database. It does not delete the files in the shadow.
- **view** - View the shadow properties.
- **synchronize** - Synchronize a shadow to the state of the TeamConnection database. A shadow is out of sync with TeamConnection when the contents of the parts in the shadow are not the same as the contents of the parts in TeamConnection. The shadow could be out of sync if a prior attempt at shadowing failed. Synchronizing the shadow will attempt to update the shadow to reflect the current state of the TeamConnection objects (release, drivers, work areas). The synchronize action has a report option that will only report the parts that are out of sync.
- **verify** - Verifying a shadow will synchronize a shadow. Additionally, TeamConnection will verify that the timestamps of the parts in the shadow match the timestamps that TeamConnection stored when the parts were placed in the shadow. This action is typically required when your shadow has been damaged from unexpected problems. For example, in the case of a disk crash, the shadow could be restored from backup. Then the verify action could be used to update any parts that have changed since the backup.

See the Command Reference for more details on each of these actions.

---

## When does shadowing happen

For **manual** mode shadows, shadowing occurs only when the shadows are explicitly synchronized with the shadow **-synchronize** command. This action will update the shadows to reflect the current state of the TeamConnection release, driver, or work area that you are synchronizing.

For **synchronous** mode shadows, shadowing occurs as the contents of releases, drivers, and work areas change. The following actions will trigger shadowing for each of the objects.

- Releases
  - Driver -commit
  - Workarea -integrate (if the driver subprocess is not included in the release process.)
- Drivers
  - DriverMember -create/-delete
- WorkAreas
  - Workarea -refresh/-undo/-cancel
  - Part -checkin/-create/-rename/-undo/-build/-delete/-link/-modify/-recreate/-refresh/-rename

The Driver -commit and WorkArea -integrate commands will trigger shadowing for all of the parts on the specified driver or work area. The DriverMember commands will trigger shadowing for all of the parts on the specified driver that are not current. The work area -refresh, -undo, and -cancel commands will trigger shadowing for all of the parts on the specified work area that are not current. A part command will only trigger shadowing for the specified parts in the specified work areas.

---

## Writing shadowing programs

When TeamConnection determines that a shadow needs to be updated, it will call a shadowing program to perform the actual updates to the file system. The shadowing program is called once for each file in the shadow that needs to be updated. For example, on a Part **-create** action, the shadowing program will be called once for the part. For a DriverMember **-create** action, the shadowing program will be called once for each part in the work area that was added to the driver.

## Shadowing program interface

The shadowing program must implement the following interface to copy, delete, and verify parts in the shadow:

```
-chmod -family Name -release Name -shadow Name -path Name -location Text
        -fmode Name -parameters Text

-copy -family Name -release Name -shadow Name -path Name
        -location Text -type [ text | binary ] -fmode Name
        -sourcefile Name -parameters Text

-delete -family Name -release Name -shadow Name -path Name
        -location Text -parameters Text

-verify -family Name -release Name -shadow Name -path Name
        -location Text -timestamp Timestamp -parameters Text
```

Attributes:

Attribute	Description
-family Name	The family for which this program is being called.
-release Name	The release for which this program is being called.
-shadow Name	The name of the shadow.
-path Name	The full pathname of the part being shadowed.
-location Text	The location as defined for the shadow with all variables replaced by their actual value.
-type [ text   binary ]	The type of the file.
-fmode Name	The filemode of the file.
-timestamp TimeStamp	The timestamp of the file in the form YYYY/MM/DD HH:MM:SS
-sourcefile Name	The full pathname of a file that contains the contents of the part. This is a temporary copy of the part that TeamConnection extracts. After the shadowing has finished, this file will be removed.
-parameters Text	The parameters defined for the shadow with all variables replaced by their actual value. This is always passed as the last parameter to the program.

Your shadowing program must also implement a validate function. Every time a shadow is created or modified, this function is called with the properties of the shadow. When a shadow is created, the shadow properties are determined by overriding the default property values with those explicitly specified on the command. When a shadow definition is modified, the shadow properties are determined by overriding the current property values with those specified on the command. The syntax for the validate action is:

```
-validate -shadow Name -type Name -release Name -location Text
        -contents [drivers] [work areas] [release]
        -mode { synchronous | manual } -crlf { yes | no }
        -keys { yes | no } -timestamp { preserve | current }
        -priority Number -parameters Parameters
```

The shadowing program should validate the values specified. In particular, the **-location** and **-parameters** value should be checked to see if they have been specified properly (since these parameters are implemented by the shadowing program, TeamConnection does not perform any validation on them). If the parameters are valid, this function should return a zero, and TeamConnection will store the updates. If the parameters are not valid, display an error message and return a nonzero return code. The updates will not be stored.

## Shadowing program requirements

This section defines the minimal requirements that each of the shadowing actions should implement. Your shadowing program may perform more than what is required here. It all depends on the needs and characteristics of your installation. Since shadowing programs will be run frequently on the TeamConnection family server, you should keep performance in mind when writing shadowing programs. Try to keep the time required to perform shadowing as short as possible. Use compiled programs rather than interpretive languages (command files, shells, or scripts).

The **-chmod** function of the program should modify the file mode of *location* to the *fmode value*. Note that when you create your shadowing program, you should only change the mode when you are shadowing to file systems that support file modes (AIX, HP-UX, and Solaris). If the function is successful, return zero, otherwise return a nonzero return code.

The **-copy** function of the program should copy the contents of the *sourcefile* to the *location*. The timestamp of the part in the shadow must be the same as the timestamp on the *sourcefile*. This enables the shadow to be verified at a later time. If the function is successful, return zero, otherwise return a nonzero return code.

The **-delete** function should delete the part from the *location*. If the function is successful, return zero, otherwise return a nonzero return code.

The **-verify** function should validate that the *timestamp* of the file in the *location* is the same as the timestamp parameter. If the timestamp is the same, a return code of zero should be returned. If the timestamp is not the same and you want the timestamp to be refreshed, return a one. For other errors, return any other nonzero return code.

**Note:** For the **-chmod**, **-copy**, and **-delete** functions, a shadowing failure does not cause the TeamConnection command to fail. For example, if on a part **-checkin**, the shadowing fails, the part is still checked in to TeamConnection and unlocked from your user ID. This results in the shadow being out of sync with TeamConnection. The shadow can be synchronized after the problem with the shadow has been corrected.

## Shadowing program output

Any error messages that are displayed from within the shadowing program are returned to the user as part of the command output. TeamConnection will display a warning message stating that the command was successful, but shadowing errors occurred.

## Sample shadowing program

A sample shadowing program named TCshadow.c is shipped with TeamConnection. It implements a simple shadow on a file system that the TeamConnection server can access directly. Included in the sample are routines for parsing and validating the parameters passed to the shadow program. You can compile this and use it as is, or use it as the basis for creating your own shadow programs.



---

## Part 3. Maintaining the TeamConnection server

This section contains information on maintaining your TeamConnection database, monitoring family use, and migrating your database to TeamConnection version 3.



---

## Chapter 11. Maintaining your TeamConnection environment

This chapter tells you how to use several TeamConnection tools for the following:

- Changing the age of defects and features
- Taking care of returned mail
- Resolving TeamConnection errors
- Backing up the TeamConnection database

As the family or database administrator for a TeamConnection family, you will also need to perform maintenance and tuning operations on the DB2 database that stores your TeamConnection family. For information on administering a DB2 database, refer to the following DB2 Universal Database administration manuals:

- *Administration Getting Started* (S10J-8154-00)  
An introductory guide to basic administration tasks and the DB2 administration tools.
- *SQL Getting Started* (S10J-8156-00)  
Discusses basic concepts of DB2 SQL.
- *Administration Guide* (S10J-8157-00)  
A complete guide to administration tasks and the DB2 administration tools.
- *SQL Reference* (S10J-8165-00)  
A reference to DB2 SQL for programmers and database administrators.
- *Troubleshooting Guide* (S10J-8169-00)  
A guide to identifying and solving problems with DB2 servers and clients and to using the DB2 diagnostic tools.
- *Messages Reference* (S10J-8168-00)  
Provides detailed information about DB2 messages.
- *Command Reference* (S10J-8166-00)  
Provides information about DB2 system commands and the command line processor.
- *Replication Guide* (S10J-0999-00)  
Describes how to plan, configure, administer, and operate IBM replication tools available with DB2.
- *System Monitor Guide and Reference* (S10J-8164-00)  
Describes how to monitor DB2 database activity and analyze system performance.
- *Glossary*  
A comprehensive glossary of DB2 terms.

More information on administering a TeamConnection DB2 family database may be available in technical reports on the IBM VisualAge TeamConnection Enterprise Server Library home page. To access this home page, select Library from the IBM VisualAge TeamConnection home page at URL <http://www.software.ibm.com/ad/teamcon>.

---

## Changing the age of defects and features

TeamConnection provides two aging utilities: `age` and `resetAge`. Use these utilities to update the age value of defects and features while work is in progress. If you do not use these utilities, the age value for each defect and feature remains at zero.

Before you use the age utilities, make sure you have set the `TC_DBPATH` and `TC_FAMILY` environment variables in your `config.sys` file as follows:

- Set `TC_DBPATH` to the directory where the family database is. Make sure that you do not include a semicolon ( ; ) or backslash ( \ ) at the end of this path.
- Set `TC_FAMILY` to the family name.

The following is an example for a family database named `testfam`:

```
SET TC_DBPATH=c:\teamc\testfam
SET TC_FAMILY=testfam
```

With these environment variables, the age utilities will change the defect and feature ages of the TeamConnection family database `testfam` in the directory `c:\teamc\testfam`.

### The age utility

Use the age utility to increment the age value by 1 for each defect or feature that is in a specified state.

The age utility is shipped in one of the following files:

**UNIX platforms**

`age`

**OS/2** `age.cmd`

**Windows**

`age.bat`

The file is located in `bin` subdirectory of the directory where the TeamConnections server is installed. Initially, the file is set up to update the age of defects that are in one of the following states:

- open
- working
- design
- size
- review

You can edit the file to delete one or more of these states or to add any of the following states:

- canceled

- returned
- closed
- verify

Run the age utility from a server machine using the following command:

```
age
```

## The resetAge utility

Use the resetAge utility to reset the age of defects and features based on their state (open, working, design, size, and review), the date they were opened, and the selected aging increment.

Run the resetAge utility from a server machine using the following command.

```
resetage ageIncr
```

Where *ageIncr* is one of the following:

fullweek

Ages the defects and features according to a 7-day schedule.

workweek

Ages the defects and features according to a 5-day work week schedule.

---

## Resolving TeamConnection errors

The TeamConnection error log and audit log can help you resolve TeamConnection error messages. This section explains how to use these two logs and briefly explains the trace facility.

### Using the system error log (syslog.log)

Severe errors that are encountered by the family server are recorded in the syslog.log file. Use this file to help you better understand and resolve the error. This log usually provides more information than what is found in the initial message. There is only one syslog.log file, so if you have multiple families, error information for each family is recorded in the same file.

Because the syslog facility is not a native application for OS/2 and Windows NT, TeamConnection provides an application specific syslog. The syslog file resides in the same location where the TeamConnection Family Server (teamcd.exe) resides, and it is called syslog.log.

Refer to the *Installation Guide* for instructions on activating the syslog (on UNIX platforms).

## Using the audit log (audit.log)

For each family, TeamConnection provides an audit log that contains an entry for every action performed since the family was created. The audit.log file is located in the directory where your family database is installed (your TC\_DBPATH).

The audit log file contains information about both successful and unsuccessful transactions, making it useful for determining the source of a problem. It also includes an entry whenever an unauthorized attempt is made to access the TeamConnection server. This can help you audit your system's security.

The following information is recorded in the audit log for each transaction:

- For authorized transactions:
  - Process ID number of the family server
  - TeamConnection action
  - Whether the transaction was successful or not
  - Date of the transaction
  - Time the transaction started
  - Time the transaction ended
  - For failure transactions, status phase information showing the C++ method being executed by the TeamConnection action name
  - User ID of the person who requested the action
  - The name of the host system from which the user is accessing TeamConnection
  - Additional information for successful transactions, or error messages for unsuccessful transactions. See page 131 for the additional information about each TeamConnection action.
- For unauthorized transactions:
  - Process ID number of the family server
  - User ID of the person who requested the action
  - The name of the host system from which the user is accessing TeamConnection
  - Notification that the request was unauthorized
  - Date and time of the transaction request
  - For failure transactions, status phase information showing the C++ method being executed by the TeamConnection action name
  - Error message

The following is an example of information as it appears in the audit.log file.

```

31436,ReleaseCreate,SUCCESS,1998/03/17,11:32:50,11:33:00,tcserv,tcserv,
alexm.ral.ibm.com,robot_v2
31449,PartLink,SUCCESS,1998/03/17,11:33:41,11:33:42,tcserv,tcserv,
alexm.ral.ibm.com,FILEH1.bin,relH1,robot_v2,1.2
31249,PartCheckOut,SUCCESS,1998/03/17,11:35:08,11:35:08,tcserv,tcserv,
alexm.ral.ibm.com,FILEH1.bin,robot_v2,1.3
31259,PartCheckIn,SUCCESS,1998/03/17,11:35:18,11:35:18,tcserv,tcserv,5
alexm.ral.ibm.com,FILEH1.bin,relI1,1.4
24942,Transaction from joe/tcserv@tcserv.ral.ibm.com was UNAUTHORIZED,03/18/95,09:43:11,
0010-100 User joe was not found.
68256,PartExtract,FAILURE,1998/02/06,10:35:45,10:35:46,beville,beville,ausaix18.austin.ibm.com,
statusphase = getListByBaseName
6021-140 There is no committed version of part junk.c visible to release r2.
To view the part, specify a work area that has a visible version of the part.
Recovery:
- Verify that the correct release name and part name were specified.
- Specify a valid work area which has a visible version of the part.

```

Figure 37. Sample of an audit log file

The following table lists the additional information that is provided for each TeamConnection action.

TeamConnectionAction	Additional information
AccessCreate	TeamConnection user ID, component name, authority group name
AccessDelete	TeamConnection user ID, component name
AccessRestrict	TeamConnection user ID, component name, authority group name
ApprovalAbstain	Release name, defect or feature name, approver's name
ApprovalAccept	Release name, defect or feature name, approver's name
ApprovalAssign	Release name, defect or feature name, new approver's name
ApprovalCreate	Defect or feature name, release name, approver's name
ApprovalDelete	Defect or feature name, release name, approver's name
ApprovalReject	Release name, defect or feature name, approver's name
ApproverCreate	TeamConnection user ID, release name
ApproverDelete	TeamConnection user ID, release name
CompCreate	New component name
CompDelete	Component name
CompLink	Component name, new parent component name
CompModify	Component name
CompRecreate	Component name
CompUnlink	Component name, parent component name
CompView	Component name

<b>TeamConnectionAction</b>	<b>Additional information</b>
<b>CoreqCreate</b>	Release name, first defect or feature name, second defect or feature name
<b>CoreqDelete</b>	Release name, defect or feature name
<b>DefectAccept</b>	Defect name
<b>DefectAssign</b>	Defect name
<b>DefectCancel</b>	Defect name
<b>DefectClose</b>	**This action is not audited**
<b>DefectComment</b>	Defect name
<b>DefectDesign</b>	Defect name
<b>DefectModify</b>	Defect name
<b>DefectOpen</b>	Defect name
<b>DefectReopen</b>	Defect name
<b>DefectReturn</b>	Defect name
<b>DefectReview</b>	Defect name
<b>DefectSize</b>	Defect name
<b>DefectVerify</b>	Defect name
<b>DefectView</b>	Defect name
<b>DriverAssign</b>	Driver name, release name, new driver owner's TeamConnection user ID
<b>DriverCheck</b>	Driver name, release name
<b>DriverCommit</b>	Driver name, release name
<b>DriverComplete</b>	Driver name, release name
<b>DriverCreate</b>	Driver name, release name
<b>DriverDelete</b>	Driver name, release name
<b>DriverExtract</b>	Driver name, release name
<b>DriverModify</b>	Driver name, release name
<b>DriverView</b>	Driver name, release name
<b>EnvCreate</b>	Tester's TeamConnection user ID, release name, environment name
<b>EnvDelete</b>	Environment name, release name
<b>EnvModify</b>	Tester's TeamConnection user ID, release name, environment name
<b>FeatureAccept</b>	Feature name
<b>FeatureAssign</b>	Feature name
<b>FeatureCancel</b>	Feature name
<b>FeatureClose</b>	**This action is not audited**
<b>FeatureComment</b>	Feature name
<b>FeatureDesign</b>	Feature name
<b>FeatureModify</b>	Feature name
<b>FeatureOpen</b>	Feature name

<b>TeamConnectionAction</b>	<b>Additional information</b>
<b>FeatureReopen</b>	Feature name
<b>FeatureReturn</b>	Feature name
<b>FeatureReview</b>	Feature name
<b>FeatureSize</b>	Feature name
<b>FeatureVerify</b>	Feature name
<b>FeatureView</b>	Feature name
<b>FixActive</b>	Defect or feature name, release name, component name
<b>FixAssign</b>	Defect or feature name, release name, component name
<b>FixComplete</b>	Defect or feature name, release name, component name
<b>FixCreate</b>	Defect or feature name, release name, component name
<b>FixDelete</b>	Defect or feature name, release name, component name
<b>HostCreate</b>	TeamConnection user ID, host name, user login on host
<b>HostDelete</b>	TeamConnection user ID, user login on host, host name
<b>MemberCreate</b>	Driver name, defect or feature name, release name
<b>MemberDelete</b>	Driver name, defect or feature name, release name
<b>NotifyCreate</b>	TeamConnection user ID, component name, interest group
<b>NotifyDelete</b>	TeamConnection user ID, component name
<b>PartAdd</b>	Path name, release name, SID
<b>PartCheckIn</b>	Path name, release name, SID
<b>PartCheckOut</b>	Path name, release name, SID
<b>PartDelete</b>	Path name, release name
<b>PartExtract</b>	Path name, release name, SID
<b>PartForceIn</b>	**Audited via PartCheckIn**
<b>PartForceOut</b>	**Audited via PartCheckOut**
<b>PartLink</b>	Path name, release name, new release name, SID
<b>PartLock</b>	Path name, release name, SID
<b>PartLockForce</b>	**Audited via PartLock**
<b>PartMark</b>	Path name, release name, SID
<b>PartMerge</b>	Path name, release name, work area name, source release name, **Also audited via underlying PartCheckOut, PartCheckIn, and PartExtract actions**
<b>PartModify</b>	Path name, release name
<b>PartOverrideR</b>	Path name, release name, cancel flag, workarea name, User ID
<b>PartReconcile</b>	Path name, release name, work area name, **Also audited via underlying PartCheckOut, PartCheckIn, and PartExtract actions**
<b>PartRecreate</b>	Path name, release name
<b>PartRecreaForce</b>	**Audited via PartRecreate**

<b>TeamConnectionAction</b>	<b>Additional information</b>
<b>PartRename</b>	Path name, new path name, release name
<b>PartRenameForce</b>	**Audited via PartRename**
<b>PartResolve</b>	Base name, release name
<b>PartRestrict</b>	Path name, release name, cancel flag
<b>PartUndo</b>	Path name, release name, undo type, SID
<b>PartUndoForce</b>	**Audited via PartUndo**
<b>PartUnlock</b>	Path name, release name
<b>PartView</b>	Path name, release name
<b>ReleaseCreate</b>	New release name
<b>ReleaseDelete</b>	Release name, new release name
<b>ReleaseExtract</b>	Release name, new release name
<b>ReleaseLink</b>	Release name, new release name
<b>ReleaseMerge</b>	Release name, work area name, source release name, **Also audited via underlying PartCheckOut, PartCheckIn, and PartExtract actions**
<b>ReleaseModify</b>	Release name, new release name
<b>ReleaseRecreate</b>	Release name, new release name
<b>ReleaseView</b>	Release name, new release name
<b>Report</b>	**With -where flag: view name, criteria **With -help flag: help, none **With -testClient flag: test, none **With -testServer flag: test, none
<b>ShadowCreate</b>	Shadow name, release name
<b>ShadowDefine</b>	Type name
<b>ShadowDelete</b>	Shadow name, release name
<b>ShadowDisable</b>	Shadow name, release name
<b>ShadowEnable</b>	Shadow name, release name
<b>ShadowModify</b>	Shadow name, release name
<b>ShadowRedefine</b>	Type name
<b>ShadowSync</b>	Shadow name, release name, workarea or driver name
<b>ShadowUndefine</b>	Type name
<b>ShadowVerify</b>	Shadow name, release name, workarea or driver name
<b>ShadowView</b>	Shadow name, release name
<b>SizeAssign</b>	Defect or feature name, component name, release name
<b>SizeAccept</b>	Defect or feature name, component name, release name
<b>SizeCreate</b>	Defect or feature name, component name, release name
<b>SizeDelete</b>	Defect or feature name, component name, release name
<b>SizeReject</b>	Defect or feature name, component name, release name
<b>TestAbstain</b>	Defect name, release name, environment name, tester's TeamConnectionuser ID

<b>TeamConnectionAction</b>	<b>Additional information</b>
<b>TestAccept</b>	Defect name, release name, environment name, tester's TeamConnectionuser ID
<b>TestAssign</b>	Defect name, environment name, new tester's TeamConnection user ID
<b>TestCreate</b>	Defect name, environment name, release name, tester's TeamConnectionuser ID
<b>TestDelete</b>	Defect name, environment name, release name
<b>TestReject</b>	Defect name, release name, environment name, tester's TeamConnectionuser ID
<b>WorkareaAssign</b>	Defect or feature name, release name, new work area owner's TeamConnectionuser ID
<b>WorkareaCancel</b>	Defect or feature name, release name
<b>WorkareaCheck</b>	Defect or feature name, release name, driver name
<b>WorkareaCommit</b>	Defect or feature name, release name
<b>WorkareaComple</b>	Defect or feature name, release name
<b>WorkareaCreate</b>	Defect or feature name, release name
<b>WorkareaExtract</b>	Work area name, release name
<b>WorkareaFix</b>	Defect or feature name, release name
<b>WorkareaIntegra</b>	Defect or feature name, release name
<b>WorkareaModify</b>	Defect or feature name, release name, target
<b>WorkareaReconcile</b>	Release name, work area name, <b>**Also audited via underlying PartCheckOut, PartCheckIn, and PartExtract actions**</b>
<b>WorkareaTest</b>	Defect or feature name, release name
<b>WorkAreaView</b>	Defect or feature name, release name
<b>UserCreate</b>	New user ID
<b>UserDelete</b>	User ID
<b>UserRecreate</b>	User ID
<b>UserModify</b>	User ID
<b>UserView</b>	<b>**No additional information is audited**</b>
<b>VerifyAbstain</b>	Defect or feature name, TeamConnection user ID
<b>VerifyAccept</b>	Defect or feature name, TeamConnection user ID
<b>VerifyAssign</b>	Defect or feature name, TeamConnection user ID of the new verification record owner
<b>VerifyReject</b>	Defect or feature name, TeamConnection user ID

## Cleaning up the audit log



The tccleanu utility is not available on Windows NT.

TeamConnection continually appends information to the end of the audit log. To keep this file from growing too large, type the following from a command line in the directory containing your TeamConnection audit log. If you need to maintain the audit.log for more than one TeamConnection family, then type this command from the directory where each audit log is located. Before issuing this command, stop the family server (refer to page “Stopping the servers” on page 46).

```
tccleanu fileSize
```

Where *fileSize* is the size of the specified file in bytes. If you do not specify the size, the default is 256000.

TeamConnection creates a backup file called audit1.log. It places this file in the directory where the audit log is located and from which you issue the tccleanu command. You can rename this file to any name you want for archive purposes. If you do not rename the file, TeamConnection keeps three backup logs in addition to the current log: audit2.log, audit3.log, and audit4.log. Each time you run the tccleanu program, TeamConnection moves the contents of each log file as follows:

1. audit3.log information is moved to audit4.log.
2. audit2.log information is moved to audit3.log.
3. audit1.log information is moved to audit2.log.
4. audit.log information is moved to audit1.log.

After this command is issued, the audit.log file is empty and ready to log new information.

## Using the trace facility

TeamConnection provides environment variables for trace. Modify the trace environment variables *only* when directed to do so by an IBM service representative.

The names of the TeamConnection trace environment variables, the purpose they serve, and the TeamConnection component that uses the environment variable are listed in the following table:

Environment variable	Purpose	Used by
TC_TRACE	Specifies the variable that lets the user designate which parts should be traced. You should modify this only when directed to do so by an IBM service representative. Otherwise it is set to null.	Client and family and build servers
TC_TRACEATTEMPTS	Specifies the maximum number of failed trace attempts accepted before giving up. You should modify this only when directed to do so by an IBM service representative.	Client and family and build servers

Environment variable	Purpose	Used by
TC_TRACEDELAY	Specifies the amount of time in seconds that TeamConnection waits, when a trace attempt fails, before attempting another trace. The default is 1 second. You should modify this only when directed to do so by an IBM service representative.	Client and family and build servers
TC_TRACEFILE	Specifies the output (part path and name) of the trace that the user designates using TC_TRACE.	Client and family and build servers
TC_TRACESIZE	Specifies the maximum size of the trace file in bytes. If this size is reached, wrapping occurs. The default is one million bytes.	Client and family and build servers

## Backing up the TeamConnection database

Your TeamConnection database needs to be backed up regularly using the DB2 backup utilities available from the DB2 Control Center or the following command from the command line processor:

```
db2 backup database family_name to backup_directory
```

Substitute your family name for *family\_name* and a directory path for your backed up database for *backup\_directory*. The DB2 backup utility will place a compressed version of the database in the backup directory path. Be sure to set file permissions for the backup directory such that the compressed backup file is accessible. It is recommended that you copy this backup file to an external backup media (i.e. tape) to protect against file system failures. Refer to the *IBM DB2 Universal Database Administration Guide* for details on this process.

**Note:** It is not recommended that you make changes to your database by issuing INSERT, UPDATE, or DELETE statements or by changing or deleting database tables or the columns defined in TeamConnection database tables. Changing your database in these ways, through the DB2 administrator tools, the DB2 command line processor, the TeamConnection migration tools, or the tcupdb tool can corrupt your TeamConnection database. Any such changes are made at your own risk. Please contact your IBM representative for information on the terms of IBM customer support.



---

## Chapter 12. Enhancing SQL performance

The performance of SQL applications can be impaired after many updates, deletes, or inserts have been made. Generally, newly inserted rows cannot be placed in a physical sequence that is the same as the logical sequence defined by the index. This means that the database manager must perform additional physical reads to access the data, because logically sequential data may be on different data pages.

In general, reorganizing a table takes more time than running statistics. Performance may be improved sufficiently by obtaining the current statistics for your data and rebinding your applications, so try this first. If this does not improve performance, the data in the tables and indexes may not be arranged efficiently, so reorganization may help.

For more details on using RUNSTATS and reorganizing table data, see the DB2 Universal Database *Administration Guide* and *Command Reference*.

---

### Collecting statistics using the RUNSTATS utility

The RUNSTATS utility updates statistics in the system catalog tables to help with the query optimization process. Without these statistics, the database manager could make a decision that would adversely affect the performance of an SQL statement. The RUNSTATS utility allows you to collect statistics on the data contained in the tables, indexes, or both tables and indexes.

Use the RUNSTATS utility to collect statistics based on both the table and the index data to provide accurate information to the access plan selection process in the following situations:

- When a table has been loaded with data, and the appropriate indexes have been created.
- When a table has been reorganized with the REORG utility.
- When there have been extensive updates, deletions, and insertions that affect a table and its indexes. (Extensive in this case may mean that 10 to 20 percent of the table and index data has been affected.)
- Before binding application programs whose performance is critical
- When comparison with previous statistics is desired. Running statistics on a periodic basis permits the discovery of performance problems at an early stage, as described below.
- When the prefetch quantity is changed.
- When you have used the REDISTRIBUTE NODEGROUP utility.

---

## Analyzing statistics

Analyzing the statistics can indicate when reorganization is necessary. Some of these indications are:

- Clustering of indexes

Index scans that are **not** index-only accesses might perform better with higher cluster ratios. A low cluster ratio leads to more I/O for this type of scan, since after the first access of each data page, it is less likely that the page is still in the buffer pool the next time it is accessed. Increasing the buffer size can improve the performance of an unclustered index. (See for information about how the database manager can improve index scan performance for indexes with low cluster ratios and the optimizer uses index statistics.)

If the table data was initially clustered with respect to a certain index, and the above clustering information indicates that the data is now poorly clustered for that same index, you may wish to reorganize the table to re-cluster the data with respect to that index.

- Overflow of rows

The overflow number indicates the number of rows that do not fit on their original pages. This can occur when VARCHAR columns are updated with longer values. In such cases, a pointer is kept at the row's original location. This can hurt performance, because the database manager must follow the pointer to find the row's contents, which increases the processing time and may also increase the number of I/Os.

As the number of overflow rows grows higher, the potential benefit of reorganizing your table data also increases. Reorganizing the table data will eliminate the overflowing of rows.

- Comparison of file pages

The number of pages with rows can be compared with the total number of pages that a table contains. Empty pages will be read for a table scan. Empty pages can occur when entire ranges of rows are deleted.

As the number of empty pages grows higher, so does the need for a table reorganization. Reorganizing the table can compress the amount of space used by a table, by reclaiming these empty pages. In addition to more efficient use of disk space, reclaiming unused pages can also improve the performance of table scan, since fewer pages will be read into the buffer pool.

- Number of leaf pages

The number of leaf pages predicts how many index page I/Os are needed for a complete scan of an index.

Random update activity can cause page splits to occur that increase the size of the index beyond the minimum amount of space required. When indexes are rebuilt during the reorganization of a table, it is possible to build each index with the minimum amount of space possible.

**Note:** A default of ten percent free space is left on each index page when the indexes are rebuilt. The environment variable DB2\_INDEX\_FREE can be used to

establish a value other than the default for the amount of free space for each index page. The maximum amount of free space for each index page is sixty percent.

RUNSTATS can also help you determine how performance is related to changes in your database. The statistics show the data distribution within a table. When used routinely, RUNSTATS provides data about tables and indexes over a period of time, thereby allowing performance trends to be identified for your data model as it evolves over time.

Ideally, you should rebind application programs after running statistics, because the query optimizer may choose a different access plan given the new statistics. See “Rebinding the family database” on page 180 for instructions on rebinding the TeamConnection database.

If you do not have enough time available to collect all of the statistics at one time, you may choose to periodically run RUNSTATS to update only a portion of the statistics that could be gathered.

However, you should periodically use RUNSTATS to gather both table and index statistics at once, to ensure that the index statistics are synchronized with the table statistics. Index statistics retain most of the table and column statistics collected from the last run of RUNSTATS. If the table has been modified extensively since the last time its table statistics were gathered, gathering only the index statistics for that table will leave the two sets of statistics out of synchronization.

You may wish to collect statistics based only on index data in the following situations:

- A new index has been created since the utility was performed and you do not want to re-collect statistics on the table data.
- There have been a lot of changes to the data that affect the first column of an index.

The RUNSTATS utility allows you to collect varying levels of statistics. For tables, you can collect basic level statistics or you can also collect distribution statistics for the column values within a table. For indexes, you can collect basic level statistics or you can also collect detailed statistics which can help the optimizer better estimate the I/O cost of an index scan.

**Note:** Statistics are not collected for LONG or large object (LOB) columns.

---

## Reorganizing table data

The REORGCHK command returns information about the physical characteristics of a table, and whether or not it would be beneficial to reorganize that table. This command can be used through the command line processor. See the *Command Reference* for more information, including how to interpret the command output.

The REORG utility optionally rearranges data into a physical sequence according to a specified index. REORG has an option to specify the order of rows in a table with an

index, thereby clustering the table data according to the index and improving the CLUSTERRATIO or CLUSTERFACTOR statistics collected by the RUNSTATS utility. As a result, SQL statements requiring rows in the indexed order can be processed more efficiently. REORG also stores the tables more compactly by removing unused, empty space.

You may wish to consider the following factors to determine when to reorganize your table data:

- The volume of insert, update, and delete activity
- Any significant change to the performance of queries which use an index with a high cluster ratio
- Running statistics (RUNSTATS) does not improve the performance of queries
- The REORGCHK command indicates a need to reorganize your table
- The cost of reorganizing your table, including the CPU time, the elapsed time, and the reduced concurrency resulting from the REORG utility locking the table until the reorganization is complete.

To execute the REORG utility, you must have SYSADM, SYSMANT, SYSCTRL or DBADM authority, or CONTROL privilege on the table.

---

## Applying these techniques to TeamConnection

TeamConnection is a diverse SQL application whose performance characteristics can be very sensitive to the statistics available to DB2 at the time the access plan for a given SQL statement is built. TeamConnection uses both dynamic SQL (as in a report) and static SQL, which means that some access plans are built dynamically when queries are encountered, while others are statically bound at bind time.

When you encounter a TeamConnection performance problem, the first approach should be to determine how recently RUNSTATS was performed against your family, and whether TeamConnection was then re-bound to refresh the access plans. If the performance problems persists after refreshing the statistics and access plans, use REORGCHK to determine which tables would benefit from reorganization, and then reorganize (using the REORG utility) those tables.

TeamConnection is designed such that the primary key index is the preferred index to organize a table. Primary key indexes are those with an index name that begins with PK. Refer to the product softcopy documentation and readme.txt file for any exceptions to this guideline.

## When REORG, RUNSTATS, and REBIND do not improve performance

If a performance problem persists, DB2 provides numerous tuning parameters that an administrator can update. Caution should be used in modifying any of these parameters. It is recommended that you modify a single parameter (or a small, related set of parameters) at a time, and then run a representative workload to determine the

impact of the modification. Many of these changes are not applied immediately, so it is advisable to stop and restart the DB2 instance after changing the DB2 configuration.

See the DB2 Universal Database *Administration Guide*, particularly the sections that discuss getting and updating the database manager and database configurations, for details about configuration and tuning opportunities.

## Table spaces and buffer pools

The Data Definition Language (DDL) used to define the TeamConnection database schema describes a number of table spaces for the tables that contain your TeamConnection family's data. If you assign those tables to separate I/O devices and separate I/O cards, you can improve the degree of I/O parallelism that DB2 provides TeamConnection.

By assigning these tablespaces to separate buffer pools and tuning the buffer pools for your system configuration, you can also improve the overall performance of your TeamConnection family.

---

## Configuration and tuning

The optimal values for the DB2 configuration and tuning parameters will be unique to each TeamConnection family and system.

When you create a new family, TeamConnection creates a DB2 database and sets the following values for certain database configuration parameters. Use caution when modifying the values to which TeamConnection sets these parameters.

### **APPLHEAPSZ = 1280**

This parameter defines the number of private memory pages available to be used by the database manager on behalf of a specific agent or subagent.

### **BUFFPAGE = 12000**

This parameter controls the size of a buffer pool when the CREATE BUFFERPOOL or ALTER BUFFERPOOL statement is run.

### **DBHEAP=2400**

This parameter indicates the maximum amount of space that the catalog cache can use from the database heap (dbheap). The catalog cache is used to store table descriptor information that is used when a table, view or alias is referenced during the compilation of an SQL statement.

### **DLCHKTIME = 1000**

This parameter defines the frequency at which the database manager checks for deadlocks among all the applications connected to a database.

### **LOGFILSIZ = 4000**

This parameter determines the number of pages for each of the configured logs. A page is 4KB in size.

**LOGPRIMARY = 5**

This parameter specifies the number of primary logs that will be created.

**LOGSECOND = 30**

This parameter specifies the number of secondary log files that are created and used for recovery log files (only as needed).

On OS/2, the following additional database parameters are set when you create a family. The value for the DBHEAP parameter is set to a different value on OS/2 than on the remaining server platforms.

**APP\_CTL\_HEAP\_SZ=128**

This parameter determines the maximum size, in 4 KB pages, for the application control shared memory. Application control heaps are allocated from this shared memory.

**CATALOGCACHE\_SZ=32**

This parameter sets the catalog cache size. The catalog cache is used to store table descriptor information that is used when a table, view or alias is referenced during the compilation of an SQL statement.

**DBHEAP=600**

This parameter indicates the maximum amount of space that the catalog cache can use from the database heap (dbheap).

**LOCKLIST=50**

This parameter indicates the amount of storage that is allocated to the lock list. There is one lock list per database and it contains the locks held by all applications concurrently connected to the database.

**MAXAPPLS=32**

This parameter specifies the maximum number of concurrent applications that can be connected (both local and remote) to a database.

---

TeamConnection leaves all other DB2 database configuration parameters at their DB2 default values.

---

## Chapter 13. Monitoring family use

TeamConnection provides monitoring tools that enable you to keep track of how family servers are being used:

- A daemon monitor accessible from the family administrator GUI or a line command (monitor) for monitoring the activity of the TeamConnection server daemons in real time.
- A license monitor command (tclicmon) for gathering information from the audit log concerning the number of users who have contacted a TeamConnection family in a given time interval.

---

### Using the server daemon monitor

The TeamConnection server daemon monitor permits you to monitor the activity of the TeamConnection server daemons. It makes use of the server's shared memory space. Each TeamConnection daemon, as well as the monitor itself, attaches to the same shared memory segment. Each time a TeamConnection server daemon services a request, the shared memory segment for that particular server daemon is updated with information regarding the user who has requested the work and the nature of the request.

You can use the server daemon monitor in a number of ways:

- To determine the activity of the server
- To determine which users issue time-consuming requests
- To determine the total number of requests serviced by the TeamConnection server and the number serviced by each server daemon since it was started
- To determine if there is a problem with one or more of the server daemons

### Using the monitor on the Family Servers window

The Family Servers window provides a family monitor area that you can use to monitor the TeamConnection family daemons. To open this window, follow these steps:

1. Do one of the following to display the TeamConnection Family Administrator window:
  - From the TeamConnection Group folder on the desktop, double-click on the **Family Administrator** icon.
  - Type `tcadmin` from a prompt.
2. Double-click the family icon for the family you want to start. The Family Servers window appears.

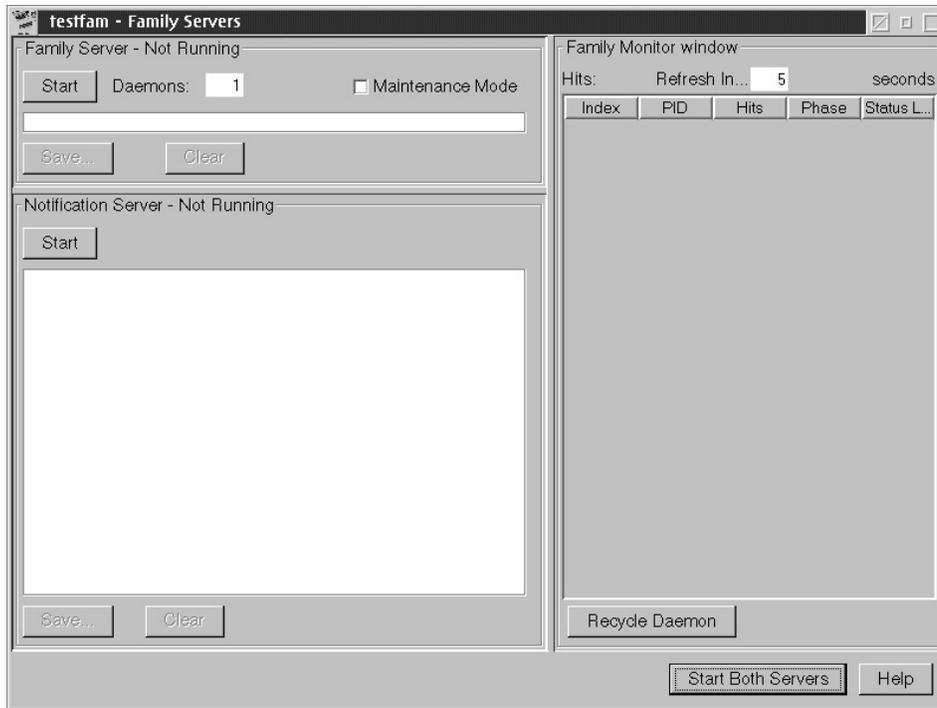


Figure 38. Family Servers window

To start the server daemon monitor, you must start the family you want to monitor on the TeamConnection server machine.

The monitor section of this window has the following fields:

**Hits** This text area displays the total number of requests processed by all family daemons.

**Refresh Interval**

This text box displays the interval in seconds at which the information in the Family Monitor window is updated. Use the up and down buttons to increase or decrease the refresh interval.

**Daemon information**

This table displays the following status information for each family daemon.

**Hits** The number of requests processed by the daemon.

**Index** The position of the daemon. If you have five daemons running, for example, each one is numbered 01 through 05.

**PID** The operating system process ID for the daemon.

## Status Line

The current command being executed by the daemon.

## Using the monitor command

To start the server daemon monitor, you must start the family you want to monitor on the TeamConnection server machine. The server daemon monitor program is located in the *teamcInstallPath\bin* directory. To start the server daemon monitor, issue the following command:

```
monitor refreshInterval [width] [-raw]
```

### Where:

- **refreshInterval** indicates the time in seconds between successive screen updates. If you start the monitor with a refresh interval of 2, for example, then the activity monitor screen is updated with new information every 2 seconds. A refresh interval of 0 displays the monitor information once and then terminates.

Set the refresh interval to a number low enough to capture requests as they are issued and processed. If you set the refresh interval too high, you may never see any activity occurring because the server daemon would have received and processed the request before the screen is updated. A refresh interval of 1 or 2 seconds is usually sufficient.

- **width** indicates the number of characters of status information to display for each TeamConnection server daemon. The default is 132 characters. The maximum is also 132 characters.
- **-raw** causes the monitor to display information in a raw format. The **-raw** option also causes the width parameter to be ignored. The raw format is separated by | characters, as in the following example:

```
*|25|1|1|
01|01234|00025|
.....
```

The asterisk in the first column marks the start of information for the family. The next three columns represent the total hits that the family has received, the daemon count, and the number of active daemons. The subsequent lines show information for each daemon: the index, process ID, hits for each daemon, phase (60 characters), and status line (256 characters). The final line of output for each family is a line of periods. This output will be displayed every refresh interval.

After you issue this command, an activity monitor screen displays, showing which server daemons are running and which are servicing requests. To exit the server daemon monitor, press any key.

The following is an example of a TeamConnection server daemon monitor screen showing 3 TeamConnection server daemons running. Two of these daemons are servicing requests. This example shows formatted output. See the description of the **-raw** parameter, above, for an example of raw output.

```
3 of 3 teamcd daemons running. Shared mem size is 1088.
Press any key to quit.
Total hits = 441
```

```
01,16504,00143,
```

```
02,15454,00152,statusphase = getListByBaseName,1998/02/26,08:56:54,
    Report,ksloop,ksloop,ksloop1.raleigh,DefectVi
    ew,state not in ('returned','canceled','verify','cl
```

```
03,12364,00146,
```

- The first line shows that all three of the TeamConnection server daemons are running and that the monitor and the server daemons are using 1088 bytes of shared memory.
- The second line indicates the total number of requests serviced by the server daemons since it was last started.
- The remaining lines show one status line for each server daemon. The status lines consist of comma-separated columns showing the following information for each daemon. If a daemon is not currently servicing requests, then only the first three columns of information are displayed. The amount of information displayed is also controlled by the *width* option specified with the `monitor` command. If you issue the `monitor` command without the *width* option, 132 characters of information are shown.

#### Column number

##### Information displayed

- 1** Daemon index. The index number of the TeamConnection server daemon in the shared memory segment. If a server daemon is stopped normally while the server daemon monitor is running, then `--` appears in this column instead of a process ID. If a server daemon is stopped abruptly or abnormally while the server daemon monitor is running, then `>>` appears in this column. In either case, the information about the request that was being processed when the daemon was stopped remains on the screen. After a daemon is started again, its process ID appears in this column.
- 2** Daemon process ID. The process ID of the TeamConnection server daemon.
- 3** The number of requests serviced by the daemon since it was started.
- 4** Status phase information, indicating the C++ method being executed.
- 5** The date the last request to the server daemon was issued. The format is `mm/dd/yy`.
- 6** The time the last request to the server daemon was issued. The format is `hh:mm:ss`.
- 7** The TeamConnection request that is being serviced.
- 8** The TeamConnection user ID that issued the request.
- 9** The login ID of the TeamConnection user who issued the request.

- 10 The hostname of the machine from which the request was issued.
- 11 Additional information about the request being serviced. This can include, for example, details about a TeamConnection query.

## Monitoring the activity of the server daemons

You can use the TeamConnection server daemon monitor to determine if you have enough server daemons running for a family:

- If you find that all daemons are constantly in use, then you may need to increase the number of daemons you start when you start the family server. Each TeamConnection family server daemon can process only one request at a time. If all daemons are busy processing requests, new requests are rejected. Users whose requests are rejected receive a message like the following:

```
0010-250 A connection cannot be established with family or port testfam at
node testfam on port 9001.
```

```
An error occurred while processing the connect system function on the
TeamConnection server. The connection request has been rejected by
the TeamConnection server.
```

Recovery:

- Verify that the connection information displayed in the message is correct and that the TeamConnection server is running.
- If the error occurs frequently, the TeamConnection server daemons may be overloaded by incoming requests. Increase the number of TeamConnection server daemons to alleviate this problem.
- If the problem persists, contact the system administrator or the family administrator.

Usually a request can be processed very quickly, but some requests can take several seconds to complete if the information being requested is lengthy or the action is complex, as in a driver -commit request. If your server is having trouble processing requests, you may want to stop the server and then restart it with more daemons, provided your license agreement permits you to do so.

- If you find that one or more daemons are rarely used, then you may need to decrease the number of daemons you start when you start the family server.

## Detecting time-consuming requests

If you notice that a specific request of a server daemon takes a long time to complete, then you can cancel the request by recycling the daemon. To recycle a server daemon, issue one of the following commands, replacing *pid* with the process ID of the daemon.



`kill pid`



Refer to your Windows NT documentation.



`kill -1 pid`



`kill -1 pid`



`kill -1 pid`

See “Stopping the servers” on page 46 for more information on recycling server daemons.

## Monitoring server daemon problems

Column 3 of the server daemon monitor screen displays the number of requests serviced by each server daemon. Requests should be nearly evenly distributed among the daemons. If one or more daemons shows an unusually low number of requests processed, then there may be a problem with that daemon. There can be one or more reasons for a low processing rate for a daemon:

- A request can take a long time to process. Actions such as `driver -commit`, `driver -check -long`, `driver -extract`, `release -extract`, and `report` can be time consuming.
- A request may be held pending the release of a lock on a database table. Certain actions, such as `driver -commit`, need to lock some of the database tables so that other users do not damage the data integrity before the request completes. If a database table is locked and an update request for that table is received, then the request will be held until the database table is unlocked. An update request is any request that alters the contents of the information in a database table. Requests that query the contents of a locked database table can still be completed.

---

## Using the license monitor

Use the TeamConnection license monitor to obtain a snapshot of the number of users who have contacted a TeamConnection family in a given time interval. The license monitor obtains family use information from the audit log. By default, only the `audit.log` for the current family is processed, but you can request information for another family on the same server.

**Note:** The license monitor needs to use an audit log file that is not currently in use by a family server. If the audit log is in use, stop the family server before running the license monitor.

The license monitor is a command that allows TeamConnection family administrators to monitor compliance with the terms of your license agreement by showing the number of concurrent uses of TeamConnection for a given time period. It is assumed that the family administrators know how many licenses the company obtained for TeamConnection.

The number of concurrent users is defined as the number of users who have contacted a TeamConnection family in a given amount of time. The default is 15 minutes. If, for example, you have 30 licenses and a total pool of 100 users, then up to 30 users can work with a TeamConnection family for any given period of 15 minutes.

The license monitor command does not enforce the limit of the number of licenses. Even if the number of actual users exceeds the number of licenses for TeamConnection, no attempt is made to limit access to a TeamConnection family. It is the responsibility of the family administrator to monitor the license usage and if the number of concurrent users exceeds the number of licenses for TeamConnection, then the family administrator should contact IBM to obtain more licenses. The number of licenses and the highest actual number of concurrent users should match.

The license monitor command is invoked from the directory of the family you want to monitor. It uses the contents of the audit.log to determine how many users (defined by each unique combination of user ID, login ID, and host name) have contacted the TeamConnection family in a given date and time interval, according to periods of a given duration (also called histograms). If, for example, use is to be monitored for two hours from 08:00 to 10:00, then the license monitor checks the audit log for users each 15 minutes (the default): four times per hour or eight times in the two-hour interval.

The following is an example of how family use might be reported for this two-hour period:

```
From 08:00:01 to 08:15:00, actual users: 1
From 08:15:01 to 08:30:00, actual users: 5
From 08:30:01 to 08:45:00, actual users: 5
From 08:45:01 to 09:00:00, actual users: 10
From 09:00:01 to 09:15:00, actual users: 8
From 09:15:01 to 09:30:00, actual users: 5
From 09:30:01 to 09:45:00, actual users: 3
From 09:45:01 to 10:00:00, actual users: 4
```

The highest amount of users in a given period is ten.

## How the license monitor counts users

A user is any unique combination of user ID, login ID, and host name. If, a user accesses the family using two user IDs from a single host name, for example, then that is counted as two separate users.

If more than one period in the interval being monitored has the same highest number of users, then only the first occurrence of that number is reported. If, for example, you monitor family use for three hours and the highest number of uses reaches twenty for two separate fifteen-minute periods, only the first occurrence is reported.

Because most TeamConnection transactions have a short duration, only the starting time for the transaction is considered by the license monitor command. When a long transaction starts in one time period and ends in another time period, the license monitor counts that use only once. It ignores the user's transaction for the second time period. A transaction in the audit log is processed only for those entries with a status of SUCCESS.

## Using the `tclicmon` command

You can issue the license monitor command any time, but it is recommended that you issue it at least once a day, especially before or after the daily backup of the family.

When you issue the command, you specify values for the dates and times that mark the interval you want to monitor. Use the following format for dates and times in the license monitor command:

```
yyyy/mm/dd, hh:mm:ss
```

The comma between the date and the time is required. The default value for the begin date and time is today at 00:00:01, and the default value for the end date and time is today at the current time.

The license monitor command has three action flags:

### **`tclicmon -highest`**

Displays only a summary of the report of concurrent users. The main element of the report is the time period that had the maximum use.

### **`tclicmon -report`**

Displays a full report of use for all time periods between the `-begin` date and the `-end` date for the duration specified in the `-timePeriod` attribute. You can request the report in several different formats.

### **`tclicmon -help`**

Displays a summary of the command, showing some examples and the defaults. To see help for the syntax, enter the command without any arguments.

**Note:** The order of the arguments for the `tclicmon` command needs to follow the sequence described in the syntax. For example, if you want to use a long report format with a begin date, then the order is:

```
-report -long -begin
```

If you change the order of the command arguments as follows:

```
-report -begin -long
```

the command will not be executed and a usage message will appear.

## Reporting highest uses

To display the highest use for an interval, issue the following command from the directory containing the family you want to monitor:

```
tclicmon -highest
        [-begin yyyy/mm/dd,hh:mm:ss]
        [-end   yyyy/mm/dd,hh:mm:ss]
        [-timePeriod minutes]
        [-input fileName]
```

Where:

- `-begin yyyy/mm/dd,hh:mm:ss` is the date and time of the beginning of the interval. The default is today at 00:00:01.
- `-end yyyy/mm/dd,hh:mm:ss` is the date and time of the end of the interval. The default is today at the current time.
- `-timePeriod minutes` is the duration of each time period, in minutes. The minimum value is five minutes. The default is 15 minutes.
- `-input fileName` is the full path name of the file that contains the audit log. The default is the file name "\$TC\_DBPATH/audit.log" (for AIX, HP-UX, and Solaris) or "d:\%TC\_DBPATH%\audit.log" (for OS/2 or Windows NT), where `TC_DBPATH` is the top directory for the family.

To obtain the default report of only the highest use, type the following,

```
tclicmon -highest
```

If the current date and time when the command is issued is July 31, 1998, 08:15:59 and the current directory for the family is `k:\testfam`, then the result shown in the standard output might be as follows:

```
*** TeamConnection License Monitor ***
```

```
Begin date:
  1998/07/31,00:00:01
End date:
  1998/07/31,08:15:59
Length of each time period, in minutes:
  15
Audit file:
  K:\testfam\audit.log
```

The period that has the highest number of concurrent users is:

```
beginDate          endDate          concurrentUsers
-----
1998/07/31,07:00:00 1998/07/31,07:15:00 3
```

## Displaying a full use report

To display a full use report for an interval, issue the following command from the directory containing the family you want to monitor:

```
tclicmon -report [-outputFormat]
             [-begin yyyy/mm/dd,hh:mm:ss]
             [-end   yyyy/mm/dd,hh:mm:ss]
             [-timePeriod minutes]
             [-input fileName]
```

Where:

- *-outputFormat* is one of the following:

**-csv** Produces an output in comma-separated-values (CSV) format. This format can be used to prepare charts with software that can import data in CSV format.

- Each row corresponds to one time period.
- The fields are separated by a comma, and the dates are enclosed between quotes, for example:  
"1998/10/01,00:00:01", "1998/10/01,14:30:15", 10

**-long** This is the default format for the *-report* action. Produces an output with a header and a footer, and the time periods are shown in the following table format.

- Each field is displayed as a column heading.
- Field values appear under respective column heading.
- Each row corresponds to one time period.

**-raw** Produces an output in raw format:

- Each row corresponds to one time period.
- The fields are separated by a vertical bar, for example:  
1998/10/01,00:00:01|1998/10/01,14:30:15|10

**-stanza** Produces an output that is equivalent to the long format.

**-table** Produces an output without a header or a footer, and the time periods are shown in the following table format:

- Each field is displayed as a column heading.
- Field values appear under respective column heading.
- Each row corresponds to one time period.

- *-begin yyyy/mm/dd,hh:mm:mm* is the date and time of the beginning of the interval. The default is today at 00:00:01.
- *-end yyyy/mm/dd,hh:mm:ss* is the date and time of the end of the interval. The default is today at the current time.
- *-timePeriod minutes* is the duration of each time period, in minutes. The minimum value is five minutes. The default is 15 minutes.

- -input *fileName* is the full path name of the file that contains the audit log. The default is the file name "\$TC\_DBPATH/audit.log" (for AIX, HP-UX, and Solaris) or "d:\%TC\_DBPATH%\audit.log" (for OS/2 or Windows NT), where *TC\_DBPATH*: is the top directory for the family.

## Examples

- To obtain a default detailed report (-long) on family use, type the following command:  

```
tcllicmon -report -begin 1998/07/31,04:00:01
```

If the current date and time is July 31, 1998, 08:15:59, the starting time is 04:00 and the current directory for the family is k:\testfam, the result shown in the standard output might be as follows:

\*\*\* TeamConnection License Monitor \*\*\*

```
Begin date:
  1998/07/31,04:00:01
End date:
  1998/07/31,08:15:59
Length of each time period, in minutes:
  15
Audit file:
  K:\testfam\audit.log
```

beginDate	endDate	concurrentUsers
1998/07/31,04:00:01	1998/07/31,04:15:00	2
1998/07/31,04:15:00	1998/07/31,04:30:00	1
1998/07/31,04:30:00	1998/07/31,04:45:00	0
1998/07/31,04:45:00	1998/07/31,05:00:00	0
1998/07/31,05:00:00	1998/07/31,05:15:00	0
1998/07/31,05:15:00	1998/07/31,05:30:00	0
1998/07/31,05:30:00	1998/07/31,05:45:00	0
1998/07/31,05:45:00	1998/07/31,06:00:00	0
1998/07/31,06:00:00	1998/07/31,06:15:00	2
1998/07/31,06:15:00	1998/07/31,06:30:00	1
1998/07/31,06:30:00	1998/07/31,06:45:00	0
1998/07/31,06:45:00	1998/07/31,07:00:00	0
1998/07/31,07:00:00	1998/07/31,07:15:00	3
1998/07/31,07:15:00	1998/07/31,07:30:00	0
1998/07/31,07:30:00	1998/07/31,07:45:00	0
1998/07/31,07:45:00	1998/07/31,08:00:00	1
1998/07/31,08:00:00	1998/07/31,08:15:00	0

The period that has the highest number of concurrent users is:

beginDate	endDate	concurrentUsers
1998/07/31,07:00:00	1998/07/31,07:15:00	3

- To obtain a detailed report on family use in table format (without the header and footer), type the following command:  

```
tcllicmon -report -table -begin 1998/07/31,04:00:01
```

If the current date and time is July 31, 1998, 08:15:59, the starting time is 04:00, and the current directory for the family is k:\testfam, the result shown in the standard output might be as follows:

beginDate	endDate	concurrentUsers
1998/07/31,04:00:01	1998/07/31,04:15:00	2
1998/07/31,04:15:00	1998/07/31,04:30:00	1
1998/07/31,04:30:00	1998/07/31,04:45:00	0
1998/07/31,04:45:00	1998/07/31,05:00:00	0
1998/07/31,05:00:00	1998/07/31,05:15:00	0
1998/07/31,05:15:00	1998/07/31,05:30:00	0
1998/07/31,05:30:00	1998/07/31,05:45:00	0
1998/07/31,05:45:00	1998/07/31,06:00:00	0
1998/07/31,06:00:00	1998/07/31,06:15:00	2
1998/07/31,06:15:00	1998/07/31,06:30:00	1
1998/07/31,06:30:00	1998/07/31,06:45:00	0
1998/07/31,06:45:00	1998/07/31,07:00:00	0
1998/07/31,07:00:00	1998/07/31,07:15:00	3
1998/07/31,07:15:00	1998/07/31,07:30:00	0
1998/07/31,07:30:00	1998/07/31,07:45:00	0
1998/07/31,07:45:00	1998/07/31,08:00:00	1
1998/07/31,08:00:00	1998/07/31,08:15:00	0

- To obtain a detailed report on family use in raw format (without the header and footer), type the following command:

```
tclicmon -report -raw -begin 1998/07/31,04:00:01
```

If the current date and time is July 31, 1998, 08:15:59, the starting time is 04:00, and the current directory for the family is k:\testfam, the result shown in the standard output might be as follows:

1998/07/31,04:00:01	1998/07/31,04:15:00	2
1998/07/31,04:15:00	1998/07/31,04:30:00	1
1998/07/31,04:30:00	1998/07/31,04:45:00	0
1998/07/31,04:45:00	1998/07/31,05:00:00	0
1998/07/31,05:00:00	1998/07/31,05:15:00	0
1998/07/31,05:15:00	1998/07/31,05:30:00	0
1998/07/31,05:30:00	1998/07/31,05:45:00	0
1998/07/31,05:45:00	1998/07/31,06:00:00	0
1998/07/31,06:00:00	1998/07/31,06:15:00	2
1998/07/31,06:15:00	1998/07/31,06:30:00	1
1998/07/31,06:30:00	1998/07/31,06:45:00	0
1998/07/31,06:45:00	1998/07/31,07:00:00	0
1998/07/31,07:00:00	1998/07/31,07:15:00	3
1998/07/31,07:15:00	1998/07/31,07:30:00	0
1998/07/31,07:30:00	1998/07/31,07:45:00	0
1998/07/31,07:45:00	1998/07/31,08:00:00	1
1998/07/31,08:00:00	1998/07/31,08:15:00	0

- To obtain a detailed report on family use in comma-separated-values format (without the header and footer), type the following command:

```
tclicmon -report -csv -begin 1998/07/31,04:00:01
```

If the current date and time is July 31, 1998, 08:15:59, the starting time is 04:00, and the current directory for the family is k:\testfam, the result shown in the standard output might be as follows:

```
"1998/07/31,04:00:01", "1998/07/31,04:15:00", 2
"1998/07/31,04:15:00", "1998/07/31,04:30:00", 1
"1998/07/31,04:30:00", "1998/07/31,04:45:00", 0
"1998/07/31,04:45:00", "1998/07/31,05:00:00", 0
"1998/07/31,05:00:00", "1998/07/31,05:15:00", 0
"1998/07/31,05:15:00", "1998/07/31,05:30:00", 0
"1998/07/31,05:30:00", "1998/07/31,05:45:00", 0
"1998/07/31,05:45:00", "1998/07/31,06:00:00", 0
"1998/07/31,06:00:00", "1998/07/31,06:15:00", 2
"1998/07/31,06:15:00", "1998/07/31,06:30:00", 1
"1998/07/31,06:30:00", "1998/07/31,06:45:00", 0
"1998/07/31,06:45:00", "1998/07/31,07:00:00", 0
"1998/07/31,07:00:00", "1998/07/31,07:15:00", 3
"1998/07/31,07:15:00", "1998/07/31,07:30:00", 0
"1998/07/31,07:30:00", "1998/07/31,07:45:00", 0
"1998/07/31,07:45:00", "1998/07/31,08:00:00", 1
"1998/07/31,08:00:00", "1998/07/31,08:15:00", 0
```



---

## Chapter 14. Server tools

The following tools are provided on the TeamConnection server. Generally, they are run from the family directory and expect that all environment variables needed to run the family server are set. The PATH environment variable should include the path containing the tools since some tools will use another.

**tcqry** A standalone version of the teamc report command.

**tcupdb** A standalone routine to update a non-TeamConnection table in the TeamConnection database.

**Note:** It is not recommended that you make changes to your database by issuing INSERT, UPDATE, or DELETE statements or by changing or deleting database tables or the columns defined in TeamConnection database tables. Changing your database in these ways, through the DB2 administrator tools, the DB2 command line processor, the TeamConnection migration tools, or the tcupdb tool can corrupt your TeamConnection database. Any such changes are made at your own risk. Please contact your IBM representative for information on the terms of IBM customer support.

---

### Using tcqry

The tcqry tool is a standalone routine that issues a TeamConnection database query. It is essentially the teamc report -general command, but bypasses the client/server interface. The following is the syntax for the tcqry command:

```
tcqry -g tabspec [-s selspec] [-w whereClause] [-c colspec]
```

Where:

- **-g** *tabspec* is the table specification.
- **-s** *selspec* specifies the columns to select. If omitted, "select \*" is assumed.
- **-w** *whereClause* is the where clause criteria.
- **-c** *colspec* is a series of numbers giving the minimum column widths for displaying the selected columns in a tabular format. The last number will be propagated if there are more columns than numbers. If omitted, a "raw" format is used with the "|" character separating the data columns.

The following example lists the user id, login and name of users defined to be superusers in TeamConnection.

```
tcqry-g users -s id,login,name -w "superuser='yes'" -c 5,12
```

---

## Using tcupdb

The tcupdb is a standalone routine that issues a database command to modify a non-TeamConnection table in the TeamConnection database. The following are options for the syntax of the tcupdb command:

```
tcupdb -g tabspec -s setClause [-w whereClause]
```

```
tcupdb -g tabspec -d [-w whereClause]
```

```
tcupdb -g tabspec -i insertClause
```

Where:

- **-g** *tabspec* is the table specification.
- **-s** *setClause* modifies the table with the given criteria.
- **-w** *whereClause* is the where clause criteria.
- **-d** [**w** *whereClause*] deletes the selected rows from the table.
- **-i** *insertClause* inserts rows into the table.

The following example deletes all rows of table mytab where the column col1 has a value less than 1.

```
tcupdb -g mytab -d -w "col1<1"
```

The following example inserts a new row into table mytab with col1 = 5 and col2 = 7.

```
tcupdb -g mytab -i "(col1, col2) values (5, 7)"
```

The following example adds one to col2 of table mytab for rows with col1 = 5.

```
tcupdb -g mytab -s "col2 = col2 + 1" -w "col1 = 5"
```

---

## Part 4. Appendixes



---

## Appendix A. Family administration commands

You can use either the Family Administrator GUI or the command line to configure TeamConnection families. This appendix explains how to do these tasks using the various family administrator commands.

Doing these tasks from the command line sometimes requires extra steps and is more prone to error. For these reasons, we recommend that you use the GUI when you can. See “Part 2. Designing and creating your TeamConnection environment” on page 27 for instructions on using the GUI.

This appendix explains how to do the following tasks from a command line:

To do this task,	Go to this page.
Creating a family database	163
Creating an initial superuser for a family	164
Creating or modifying authority groups	165
Creating or modifying interest groups	167
Configuring component or release processes	168
Defining configurable field types	170
Updating configurable field tables	173
Changing report formats	174
Configuring user exits	178
Rebinding the family database	180

---

### Creating a family database

You can create a family database from a command line prompt using the `fhcirt` command, as follows:

```
fhcirt -c -d databaseLocation loadfiles
```

Where:

- **-c** causes the DB2 database to be dropped (if it already exists) and created.
- **-d *databaseLocation*** specifies the location where the database should be created. On Intel platforms, specify a drive, such as **e:**. On UNIX platforms, specify a directory path, such as **/disk2/database**. If you omit this parameter, the database will be created where DB2 is installed.
- ***loadfiles*** are the DB2 files to be loaded into the database. On Intel platforms, specify the drive and directory path where the TeamConnection DB2 table definitions, view definitions, and bind files are installed. The default directory path is `\teamc\nls\cfg`. On UNIX platforms, specify the directory path where the TeamConnection DB2 table

definitions, view definitions, and bind files are installed. The default directory path is \$TC\_HOME/nls/cfg. For both platforms, the files you need are \*.ddl (table definitions), \*.ddv (view definitions), and \*.bnd (bind files).

The family name is specified by the TC\_FAMILY environment variable.

To create a family database named family1 on drive d:, set TC\_FAMILY=family1 and issue the following command:

```
fhcirt -c -d d: f:\teamc\nls\cfg\*.ddl f:\teamc\nls\cfg\*.ddv
      f:\teamc\nls\cfg\*.bnd
```

**Note:** The directory path \teamc\nls\cfg is the default installation path on Intel for the TeamConnection DB2 files needed to create tables and views and bind the family database. If you specify an installation path other than the default, make sure the path you specify for the *loadfiles* parameter contains the TeamConnection DB2 files.

---

## Creating an initial superuser for a family

Before you can define users, components, and releases for a family, you need to create a user ID with superuser access to the family. From a command line, you can do this using the fhchdf command. Before you can use this command, you need to create the family database and make sure the environment variable TC\_DBPATH is set. You can issue this command only once for each family. After the initial superuser ID has been created, use the TeamConnection GUI or line commands to modify or create additional users. If the family database has a component called "root," then the fhchdf command will not execute.

```
fhchdf -create
-user Name
-login Name
-address Name
-family Name
[-name Text]
[-area Name]
[-password Name]
```

Where:

- -user *Name* is the TeamConnection user ID for the superuser. If you omit this parameter, it defaults to the value specified for the -login parameter. It is a good idea to give the superuser an ID that is readily identifiable as a superuser. A good way to do this is to preface the user ID with su\_, such as -user su\_john.

**Note:** This parameter is used only in single-user environments, such as OS/2.

- -login *Name* is the login ID for the superuser. This parameter is used in multiuser environments, such as AIX, HP-UX, Solaris, and Windows NT, to identify the user account to which the TeamConnection superuser ID is assigned. It is a good idea to give the superuser an ID that is readily identifiable as a superuser. A good way to do

this is to preface the ID with `su_`, such as `-login su_john`. Single-user environments, such as OS/2, do not define a separate login ID. If you omit the `-user` parameter, it defaults to the value specified for the `-login` parameter.

- `-address Name` is the hostname of the family server from which the superuser will be authorized to access the family, such as `-address tcserver`.
- `-family Name` is the name of the family for which you are defining the superuser. The family must have already been created. An example is `-family testfam`.
- `-name Text` is the real name of the superuser, such as `-name "John Smith"`. This attribute is optional.
- `-area Name` is the development area in which the superuser works, such as `-area "User interface"`. This attribute is optional.
- `-password Name` is the password that must be used by the initial superuser. A password is required only if you created the family with the `password-only` or `password-or-host` level of security. Under these security levels, if a password is not created, then no one will have access to the database.

The following example creates a superuser ID for John Smith on the family server `tcserver` for the family named `robot`:

```
fhchdf -create -user su_john -login jsmith -address tcserver
        -family robot -name "John Smith" -area "User interface"
        -password f5asdfjk
```

---

## Creating or modifying authority groups

When a TeamConnection family is created, the authority table is primed with default information contained in the `authorit.ld` file. This section provides instructions for manually editing the `authorit.ld` file to create new authority groups or change information about existing authority groups. When you change the `authorit.ld` file, you must also reload the contents of the authority table in the TeamConnection database.

Instructions for using the Family Administrator GUI to create or modify authority groups are on page 72.

Before you create or modify authority groups, you should be familiar with the information in "Planning for user access to TeamConnection data" on page 70.

## Editing the `authorit.ld` file

To add new authority groups or to add actions to an existing authority group, edit the `authorit.ld` file. It is recommended that you keep the `authorit.ld` file in the family directory. If you want to maintain common authority group definitions for more than one family, however, you can store this file in a common directory; but you will need to specify the fully-qualified path name for the `authorit.ld` file when you load it using the `fhclauth` command.

Add entries to the file using the following format:

AuthorityGroup|ActionName

### **AuthorityGroup**

This is the name of an existing authority group or the name of a group that you are creating. The name can be 15 characters long; it cannot contain blanks, tabs, or vertical separators. For an existing authority group, type the name exactly as it appears in the database table. The default names provided by IBM use all lowercase characters.

### **ActionName**

This is the name of an existing TeamConnection action. Specify only one action per entry. You must type the name exactly as it appears in the database table. Refer to the list of actions in the *TeamConnection User's Guide* for the correct spelling and capitalization. Certain actions cannot be included in an authority group. These actions are noted in the table found in "Appendix F. Worksheets" on page 251.

## **Reloading the authority table**

Whenever you change the `authorit.ld` file, you must reload the contents of the authority table before your users can use the new and changed authority groups.

You can reload the authority table as often as necessary. We recommend that you stop the family server before you reload the authority table (see page "Stopping the servers" on page 46 for instructions).

To reload the authority table, issue the following command from the server machine. Before issuing this command, ensure that the `TC_FAMILY` environment variable is set to the correct family name and `TC_DBPATH` is set to the correct database path name.

```
fhclauth path\authorit.ld
```

Where:

- `path\authorit.ld` is the path name of the `authorit.ld` file. If you specify a fully-qualified path name, TeamConnection looks for the `authorit.ld` file in the path you specify. If you specify only **authorit.ld** with no directory path, TeamConnection looks for the file in the directory specified by the `TC_DBPATH` environment variable.

To verify that the authority table loaded correctly, use the report command to generate a report on the authority table. For example, to verify that a new authority group named `general` was added to the table, type the following from a client machine on an OS/2 or TeamConnection command line:

```
teamc report -view authority -where "name='general'"
```

If the table loaded correctly, information about the `general` authority group appears. If the authority table did not load correctly, make the necessary changes to the `authorit.ld` file and run the `fhclauth` command again.

For more information about the report -view command, refer to the *Commands Reference*.

---

## Creating or modifying interest groups

This section provides instructions for manually editing the interest.ld file to create or modify interest groups. When you change the interest.ld file, you must also reload the contents of the interest table.

Instructions for using the Family Administrator GUI to create or modify interest groups are on page 78.

Before you create or modify interest groups, you should be familiar with the information in “Planning for user notification” on page 77.

## Editing the interest.ld file

To add new interest groups or to add actions to an existing interest group, edit the interest.ld file. It is recommended that you keep the interest.ld file in the family directory. If you want to maintain common interest group definitions for more than one family, however, you can store this file in a common directory; but you will need to specify the fully-qualified path name for the interest.ld file when you load it using the fhclintr command.

Add entries to the file using the following format:

```
InterestGroup|ActionName
```

### InterestGroup

This is the name of an existing interest group or the name of a group that you are creating. The name can be up to 15 characters; it cannot contain blanks, tabs, or vertical separators. For an existing interest group, type the name exactly as it appears in the database table. The default names provided by IBM use all lowercase characters.

### ActionName

This is the name of an existing TeamConnection action. Specify only one action per entry. You must type the name exactly as it appears in the database table. Refer to the list of actions in the *TeamConnection User's Guide* for the correct spelling and capitalization. Certain actions cannot be included in an interest group. These actions are noted in the table found in “Appendix F. Worksheets” on page 251.

## Reloading the interest table

Whenever you change the interest.ld file, you must reload the contents of the interest table before your users can use the new and changed interest groups.

You can reload the interest table as often as necessary. We recommend that you stop the family server before you reload the interest table (see page “Stopping the servers” on page 46 for instructions).

To reload the interest table, issue the following command from the server machine in the directory where the interest.ld file is stored. Before issuing this command, ensure that the TC\_FAMILY environment variable is set to the correct family name and TC\_DBPATH is set to the correct database path name.

```
fhc\intr path\interest.ld
```

Where:

- *path*\interest.ld is the path name of the interest.ld file. If you specify a fully-qualified path name, TeamConnection looks for the interest.ld file in the path you specify. If you specify only **interest.ld** with no directory path, TeamConnection looks for the file in the directory specified by the TC\_DBPATH environment variable.

To verify that the interest table loaded correctly, use the report command to generate a report on the interest table. For example, to verify that a new interest group named general was added to the table, type the following from a command line on a client machine:

```
teamc report -view interest -where "name='general'"
```

If the interest table loaded correctly, information about the general interest group appears. If the interest table did not load correctly, make the necessary changes to the interest.ld file and run the command again.

For more information about the report -view command, refer to the *Commands Reference*.

---

## Configuring component or release processes

This section provides instructions for manually editing the comproc.ld and relproc.ld files to configure processes. When you change the .ld files, you must also reload the contents of the configurable process tables.

Instructions for using the Family Administrator GUI to configure processes are on page 99.

Before you configure processes, you should be familiar with the information in “Chapter 8. Configuring family processes” on page 99.

## Editing the comproc.ld and relproc.ld files

Information about configurable processes for components is stored in the comproc.ld file. Information about configurable processes for releases is stored in the relproc.ld file. When the family is created, the configurable process tables are created, based on the

settings in the `comproc.ld` and `relproc.ld` files. If you modify the configurable process tables after the family is created, edit the `.ld` files and then run the `fhclproc` command.

To add new processes or change existing processes, edit the `comproc.ld` file for component processes or the `relproc.ld` file for release processes. It is recommended that you keep the `comproc.ld` and `relproc.ld` files in the family directory. If you want to maintain common process definitions for more than one family, however, you can store these files in a common directory; but you will need to specify the fully-qualified path name for them when you load them using the `fhclproc` command.

Add entries to the file using the following format:

```
ProcessName|SubprocessName
```

**ProcessName**

The name of the process you are creating. The name can be up to 15 characters in length; it cannot contain blanks, tabs, or vertical separators.

**SubprocessName**

The name of a TeamConnection subprocess. You can specify only one of the following subprocesses for each entry. If you want to include more than one subprocess, you must have an entry for each subprocess. Type the name exactly as it appears in the database.

The following are the subprocesses for components:

- none
- dsrDefect
- dsrFeature
- verifyDefect
- verifyFeature

The following are the subprocesses for releases:

- none
- approval
- fix
- driver
- test
- track
- trackfixhold
- tracktesthold
- trackcommithold

See “Release processes” on page 55 for an explanation of these subprocesses.

## Reloading the configurable process tables

After you edit an .ld file, use the fhclproc command to reload the contents of the configurable component or release process tables with the changed values. Before issuing this command, ensure that the TC\_FAMILY environment variable is set to the correct family name and TC\_DBPATH is set to the correct database path name. The format of the fhclproc command when reloading the component process table is:

```
fhclproc path\comproc.ld c
```

The format of the fhclproc command when reloading the release process table is:

```
fhclproc path\relproc.ld r
```

Where:

- *path\comproc.ld* or *path\relproc.ld* is the path name of the process definition file. If you specify a fully-qualified path name, TeamConnection looks for the .ld file in the path you specify. If you specify only the file name with no directory path, TeamConnection looks for the file in the directory specified by the TC\_DBPATH environment variable.
- *c* indicates that you are reloading the component process table.
- *r* indicates that you are reloading the release process table.

To verify that the command successfully modified the tables, use the report command to generate a report. To do this, type one of the following commands from an OS/2 or TeamConnection command line:

```
teamc report -view Cfgcomproc  
teamc report -view Cfgrelproc
```

If the table did not load correctly, make the necessary changes to the comproc.ld or relproc.ld file and run the command again.

For more information about the report -view command, refer to the *Commands Reference*.

---

## Defining configurable field types

This section provides instructions for manually editing the config.ld file to define configurable field types. When you change the config.ld file, you must also reload the contents of the config table.

Instructions for using the Family Administrator GUI to define field types are on page 85.

Before you define field types, you should be familiar with the information in "Defining configurable field types" on page 85.

It is recommended that you keep the config.ld file in the family directory. If you want to maintain common configurable field definitions for more than one family, however, you

can store this file in a common directory; but you will need to specify the fully-qualified path name for it when you load it using the `fhclcnfg` command.

When adding entries to the file, follow the existing format of the file:

```
fieldType|value|default|kind|driver|driverSeq|dependent|dependSeq|choiceOrder|  
description|helpText
```

Information about configurable field types is stored in the config table. After you modify the config table, you must reload it (see “Reloading the config table” on page 172).

The config table consists of the following information:

*fieldType*

Identifies the types of configurable fields that are defined for your family. You specify one of these types when you configure a new field. You can create new types, and you can configure the acceptable values for each type. You must have at least one value for each type. The type field can have up to 15 characters, but it cannot contain blank spaces or tabs.

“Appendix B. Configurable field types” on page 181 describes the configurable field types that are shipped by TeamConnection.

*value*

This field represents the choices the user has for the configurable field. You can add choices to the default fields shipped by TeamConnection and to the fields created specifically for your family. The value can have up to 85 characters (single-byte or double-byte); but it cannot contain spaces or tabs. If you want to enable users to set this configurable field type to the value null, include the value null among the possible values.

**Note:** Because a user can abbreviate these values from the command line, you cannot define a value that can be an abbreviation of another value of the same type. For example, you cannot add a value of `build` to the phase type, because a value of `building` already exists. Also, if a value of 1 exists for the severity type, you cannot add a severity value of 12.

*default*

This field indicates whether the defined name is used as the default when the user does not enter a value for the configuration type. Valid values are either `yes` or `no`, and only one name for each configuration type can have the default field set to `yes`.

*kind*

This field defines the method of resolving the configured value. A kind of **0** is resolved by matching the (abbreviated) input value to a unique name value for the type. For a kind of **1**, the input may contain a list of any of the values in the *value* field separated by blanks. No abbreviations may be used. For a kind of **2**, the input must match specific rules (6-digit numeric value, for example) defined in the **value** field.

The next four fields are used to define dependencies between different parameters. One parameter for a configurable object (Defect, Feature, etc.) can be defined to be a “driver.” Other parameters may be defined to be dependent on the driver. For example,

you can define a driver type called "state" and a dependent type called "city." The possible values for "city" depend on the value selected for "state."

This dependency is set up by using non-zero values in the driver and dependent fields. The values that can be selected for the dependent field are restricted by the value selected for the driver. The values for the driver and dependent fields must be the same for all rows of a given *fieldType*. The type with a given non-zero value in the dependent field is dependent on the parameter that has that same value in its driver field. The *value* that is selected for a driver field has some number in its *driverSeq* field. If the *driverSeq* is zero, any name values of a dependent field can be selected. If the *driverSeq* is not zero, then values that can be selected for the dependent parameter must have a *dependSeq* value that is the same as the *driverSeq* value or zero.

*driver* The driver field for a dependent field.

*driverSeq*

The driver sequence number that associates a driver field with its dependent fields.

*dependent*

The dependent field.

*dependSeq*

The dependent sequence number that associates a dependent field with its driver field.

*choiceOrder*

The order in which the *value* is shown in the GUI.

*description*

This field contains the description of each value. The description field cannot contain more than 63 characters, but it can be set to blank. The description with the defined values appears on the GUI window when the field is displayed.

*helpText*

A long description of the *value*. This description is displayed if the user requests help for the value requested. A row in the config table with a null value for the name may supply general help text for the config type. The help text cannot contain newline characters if the field is enclosed in quotes. The preferred (by the GUI) format of help text is:

"xyz: This is help for the value xyz."

The default configuration field types, along with their attributes, that IBM ships are listed starting on page 181.

## Reloading the config table

When you edit the config.ld file and change any values, you must reload the contents of the config table so that TeamConnection recognizes the changes.

You can reload the config table as often as necessary. It is recommended that you stop the family server before you reload the table (see page “Stopping the servers” on page 46 for instructions).

To reload the config table, issue the following command from the server machine. Before issuing this command, ensure that the TC\_FAMILY environment variable is set to the correct family name and TC\_DBPATH is set to the correct database path name.

```
fhclcnfg path\config.ld
```

Where:

- *path*\config.ld is the path name of the configurable fields definition file. If you specify a fully-qualified path name, TeamConnection looks for the file in the path you specify. If you specify only the file name with no directory path, TeamConnection looks for the file in the directory specified by the TC\_DBPATH environment variable.

Changing values in the config table does not change any values that are already in the database for existing records.

To verify that the command successfully modified the config table, use the report command to generate a report. To do this, type the following from an OS/2 or TeamConnection command line:

```
teamc report -view config
```

If the config table did not load correctly, make the necessary changes to the config.ld file and run the command again.

For more information, about the report -view command, refer to the *Commands Reference*.

## Updating database views with new configurable field information

After you reload the contents of the config table (update the .tbl files), you must also update the database views so that the new configurable fields appear in the GUI.

It is recommended that you stop the family server before you update the database views (see “Stopping the servers” on page 46 for instructions).

To update the database views, issue the following command from the server machine. Before issuing this command, ensure that the TC\_FAMILY environment variable is set to the correct family name and TC\_DBPATH is set to the correct database path name.

```
fhcfupdv configFile view
```

Where:

- *configFile* is the name of the configurable field table (.tbl file) with which the view is to be updated. TeamConnection looks for the file in the directory specified by the TC\_DBPATH environment variable.

- *view* is the database view to be updated. The following are the views you can specify with this command. Refer to the *TeamConnection Commands Reference* for a full description of each of these views.
  - DefectView
  - Feature View
  - DefectDownView
  - FeatureDownView
  - Users
  - PartView
  - PartFullView
  - WorkAreaView
  - ReleaseView

## Updating TargetView and ConfigPartView

The TeamConnection Family Administrator GUI does not support adding configurable fields to TargetView and ConfigPartView. To add configurable fields to these views, follow these steps. You can perform this procedure any time after the database is created.

1. Copy `tcsource.tbl` from the `samples` directory to the `cfgField` directory.
2. Edit `tcsource.tbl` for any new fields to be added. By default, these fields contain definitions for only one configurable field: `externalVersion`.
3. To update TargetView with the configurable field information, issue the following command from a command line prompt:
 

```
fhcfupdv tcsource.tbl TargetView
```
4. To update ConfigPartView with the configurable field information, issue the following command from a command line prompt:
 

```
fhcfupdv tcsource.tbl ConfigPartView
```

---

## Changing report formats

This section explains how you can manually change the position of report fields on the reports TeamConnection generates for the user, defect, feature, partFullView, and partView objects.

Instructions for using the Family Administrator GUI to change the reports are on page 93.

You can use the system editor to edit the following files. Before you change the report formats, you might want to make backup copies of these files.

- `cfgfield\Defect.fmt`
- `cfgfield\Feature.fmt`

- cfgfield\Part.fmt
- chgfield\Partview.fmt
- cfgfield\Release.fmt
- cfgfield\User.fmt
- cfgfield\Workarea.fmt

Each .fmt file is divided into five sections, separated by colons. The sections are:

- StanzaViewFormat
- StanzaViewColumn
- TableViewFormat
- TableViewColumn
- TableViewHeader

The column sections describe the column name of each of the labels specified in the format sections. The header section specifies how the columns appear in the table format.

The format sections specify the layout of the report. For example, a format specification of %3\$-25.25s indicates the following:

**%** Start of format specification.

**3** The sequence number of the field that is generated by TeamConnection. The dollar sign must appear after the sequence number.

**Note:** If you add a new field to the report, you must adjust all sequence numbers for fields that appear after the new field. If you create a new configurable field and place it in position 3, for example, then you must increase the sequence number of the field that was previously defined in position 3 to 4 and increase the sequence number for all remaining fields.

**-** The output is left-justified. If you do not include this character, the output is right-justified.

**25** The minimum number of characters (bytes) of output.

**.25** The maximum number of characters (bytes) printed for all or part of the output field, or minimum number of digits printed for integer values.

If you do not want the field displayed, type 0.0. For example, you have three sequence fields: 1, 2, and 3. If you do not want sequence 2 displayed, you type:

```
%1$-4.4s %2$-0.0s %3$-15.15s
```

**s** Type of data:

s for strings

ld for integers

You can specify only a data type of `s` for configurable fields. Use `1d` to display existing values, such as defect age.

You can also change or delete the format specification. Before you change a format specification, be aware of the following:

- A format specification in the stanza view does not have to match the format specification for the same field in the table view.
- Information in a stanza report appears in columns. When you specify the identical minimum and maximum number of characters for all fields appearing in a column, the report columns are left-justified. For example, Figure 39 on page 177 shows all the fields in the first column defined as `25.25`.
- When you change a format specification in the table view, adjust the matching heading length in the table view header section. Otherwise, information will not appear correctly under the headings when users display the table.

Figure 39 on page 177 shows a sample report format for the defect table after configurable fields have been added. The changes are noted in bold font and are described following the figure.

```

# StanzaViewFormat
prefix          %01$s
name            %02$s
reference       %03$s
abstract        %04$s
duplicate       %05$s

state           %06$-25.25s  priority      %07$-20.20s
severity        %08$-25.25s  target        %09$-20.20s
age             %10$s

compName        %11$-25.25s  answer        %12$-20.20s
release         %13$-25.25s  symptom       %14$-20.20s
envName         %15$-25.25s  phaseFound    %16$-20.20s
driver          %17$-25.25s  phaseInject   %18$-20.20s

addDate         %19$-25.25s  assignDate    %20$-20.20s
lastUpdate      %21$-25.25s  responseDate  %22$-20.20s
endDate         %23$-25.25s

ownerLogin      %24$-25.25s  originLogin   %25$-20.20s
ownerName       %26$-25.25s  originName    %27$-20.20s
ownerArea       %28$-25.25s  originArea    %29$-20.20s

developer      %30$-25.25s

:
# StanzaViewColumn
# NOTE: please leave this section in English
prefix,name,reference,abstract,duplicate,state,priority,severity,
target,age,compName,answer,releaseName,symptom,envName,phaseFound,
driverName,phaseInject,addDate,assignDate,lastUpdate,responseDate,
endDate,ownerLogin,originLogin,ownerName,originName,ownerArea,
originArea,developer
:
# TableViewFormat
%-4.4s %-15.15s %-15.15s %-8.8s %-8.8s %-8.8s %-3.3s %-3.3s %-4.4s %-55.55s %30$-9.9s
:
# TableViewColumn
# NOTE: please leave this section in English
prefix,name,compName,state,originLogin,ownerLogin,severity,age,
priority,abstract,developer
:
# TableViewHeader
pref name          compName          state      originLo ownerLog sev age prio abstract developer
:

```

Figure 39. Sample report format after adding configurable fields

In Figure 39, the format of the shipped defect report was modified as follows:

- Added a new label, **developer**, at the end of the StanzaViewFormat section, and the format specification **%30\$-25.25s**
- Added the column name, **developer**, as the last entry in the StanzaViewColumn section

**Note:** When you edit the StanzaViewColumn, you must maintain a continuous line of text. Control characters are ignored and appear as output in the report.

- Added %30\$-9.9s in the corresponding position for the developer entry in the TableViewFormat section
- Added the column name developer in the TableViewColumn section
- Added a new label, developer, in the TableViewHeader section and added the corresponding dashes in the next line

If the developer field had been added to the middle of the reports instead of to the end, its sequence number and the sequence number of all remaining fields would need to be adjusted.

## Updating TargetView and ConfigPartView Reports

The TeamConnection Family Administrator GUI does not support modifying reports for TargetView and ConfigPartView. To modify the reports for these views, follow these steps. You can perform this procedure any time after the database is created.

1. Copy target.fmt from the samples directory to the cfgField directory.
2. Edit target.fmt for any new fields to be added. By default, these fields contain definitions for only one configurable field: externalVersion.

---

## Setting up user exits

This section provides instructions for manually updating the userExit file to add entries that call user-defined programs during the processing of TeamConnection actions.

Instructions for using the Family Administrator GUI to set up user exits are on page 111.

Before you edit the userExit file, you should be familiar with the information in “Chapter 9. Providing user exits” on page 103.

**Note:** The userExit file is copied to your family database directory from a file located in the language subdirectory of the nls\cfg directory path in the TeamConnection installation directory, for example, teamc\nls\cfg\enu. The version of the userExit file in this location contains comments that are not copied when the family is created using the Family Administrator GUI.

## Editing the userExit file

The userExit file has no defined actions until you add entries for the user exits that your organization will use. The entries you add specify the programs that you want started for specific TeamConnection actions. For each user exit, add an entry using the following format:

```
Action ExitID UEprogram UEparameter ENV=( ) #Comments
```

Use one or more blank spaces to separate each field in the entry. A line that begins with a # sign is a comment. You can have blank lines in the file.

The userExit file is located in the config subdirectory of the directory where your family's database is installed.

A description of each field in the entry follows:

**Action** The name of the TeamConnection action that causes the user exit to start. You must type the name exactly as it appears in the database. See the list of actions in "Appendix F. Worksheets" on page 251 for the correct spelling and capitalization. For a list of actions that support user exits, see the User Exits page of the Settings notebook for your family.

**ExitID** Identifies when the user exit program is started during the course of the TeamConnection action. Valid values are 0, 1, 2, and 3. The value indicates that the user exit program does one of the following:

- 0** Starts at the beginning of the TeamConnection action, before any initialization or access checking takes place.
- 1** Starts after all TeamConnection checks are made and TeamConnection is ready to process the command.
- 2** Starts after the TeamConnection action is completed. At this point, the action has been submitted to TeamConnection, and all database or library updates have been committed.
- 3** Starts when a previous user exit with an exit ID of 0 or 1 is not successful, or when the TeamConnection action is not successful. This exit ID allows the user exit program to clean up what the other user exit programs started.

**UEprogram**

The name of the user exit program. The program must exist in a directory defined in the PATH statement of your config.sys file (for OS/2 or Windows platforms).

**UEparameter**

A variable-length list of character string parameters provided to the user exit program.

**ENV=( )**

The customized parameter list for the user exit. See "Creating customized parameter lists" on page 180 for more information on passing a customized parameter list to a user exit program.

**#Comments**

A comment about the user exit program. This field is optional.

TeamConnection does not recognize the updates to the userExit file until you stop and restart the TeamConnection server.

## Creating customized parameter lists

To create a customized parameter list for a user exit program, include the ENV=( ) field in the definition for the user exit in the userExit file. The ENV=( ) field consists of a comma-separated list of the parameters or configurable fields to be passed to the user exit program. To pass the component and release parameters of a PartAdd action to a user exit, for example, include the ENV=( ) field as follows:

```
ENV=(component,release)
```

See “Appendix C. User exit parameters” on page 191 for a list of parameters that can be passed to a user exit program for each action’s exit IDs.

To include a configurable field in a customized parameter list, identify it by its attribute name.

---

## Rebinding the family database

After doing certain administrative tasks with the family database, such as installing patches for TeamConnection or for DB2 and after performing the DB2 action of REORG, it will be necessary to rebind the DB2 plans to the family database in order to resynchronize the consistency token and avoid a runtime error SQL -818. You can rebind the DB2 plans to the family database by issuing the following command:

```
fhcirt $TC_HOME/nls/cfg/*.bnd
```

---

## Appendix B. Configurable field types

This appendix describes the configurable field types shipped with TeamConnection. It also contains information that may help you determine how options that define configurable field types and configurable fields in the TeamConnection GUI, command line interface, and SQL interface correspond.

---

### Configurable field types

The following tables show the configuration table values under the following column headings:

#### Field type

Configuration field types that are supported by TeamConnection.

**Value** Values for the various configuration field types that are shipped with TeamConnection.

**Note:** Your TeamConnection family administrator can add names for each configuration field type.

#### Description

A description of each value shipped with TeamConnection.

**Note:** Your TeamConnection family administrator can add descriptions for fields.

There are no default values specified for most of the field types that IBM ships. However, your TeamConnection family administrator can set defaults for your family. For information on setting defaults, see "Defining configurable field types" on page 85.

#### Priority levels for defects and features

Field Type	Value	Description
priority	mustfix	Defect or feature must be resolved in this release
priority	candidate	Defect or feature is a candidate if time permits
priority	deferred	Defect or feature deferred to next release
priority	easy	Defect or feature is easy to solve or implement
priority	moderate	Defect or feature is moderately difficult to resolve
priority	difficult	Defect or feature is difficult to solve or implement
priority	n/a	Priority does not apply to this defect or feature

### The type of driver

Field Type	Value	Description
drivertype	development	Development driver
drivertype	production	Production driver
drivertype	integration	Integration driver
drivertype	prototype	Prototype driver
drivertype	other	Other type of driver

### The severity of the problem that a defect was opened to resolve

Field Type	Value	Description
severity	1	Wrong results or failure; critical to program execution
severity	2	Wrong results; not critical to program execution
severity	3	Unexpected behavior
severity	4	Suggestion or enhancement request

### The type of defect or feature

Field Type	Value	Description
defectPrefix	c	Defect reported by a customer
defectPrefix	d	Defect reported by internal users
featurePrefix	s	Suggestion made by customer
featurePrefix	f	Feature requested by internal users

### The symptom of the problem a defect was opened to resolve

Field Type	Value	Description
symptom	incorrect_i/o	Incorrect or unexpected input or output
symptom	program_defect	Program defect
symptom	design_wrong	Original design is incorrect; redesign required
symptom	function_needed	Additional function is required
symptom	plans_incorrect	Plans need to be changed or enhanced
symptom	docs_incorrect	Documentation is incorrect
symptom	prog_suspended	Program suspended during normal operation
symptom	core_dump	Core dump occurred during normal operation
symptom	lost_data	Data loss occurred during normal operation
symptom	usability	Program or application is not usable as is
symptom	test_failed	Test failed
symptom	build_failed	Build, compile, or module integration failed

Field Type	Value	Description
symptom	install_failed	Installation failed
symptom	obsolete_code	Remove obsolete code
symptom	intgr_problem	Integration problems with other applications
symptom	performance	Performance problems; code needs to be optimized
symptom	reliability	Reliability problems; code needs more work
symptom	non-standard	Coding practices or program execution is non-standard
symptom	not_to_spec	Program or application does not function as specified

#### The development phase in progress when a defect was found or injected

Field Type	Value	Description
phase	design	Design Phase
phase	planning	Planning Phase
phase	strategy	Strategic Planning Phase
phase	prototyping	Prototyping Phase
phase	development	Development Phase
phase	documenting	Documentation or Publication Phase
phase	inspections	Inspection Phase
phase	maintenance	Maintenance Phase
phase	building	Building, Compiling or Module Integration Phase
phase	unit_test	Unit Test
phase	functional_test	Functional Test
phase	regression_test	Regression Test
phase	install_test	Installation Test
phase	config_test	Configuration Test
phase	integrate_test	Integration Test
phase	quality_test	Quality Assurance Test
phase	usability_test	Usability Test
phase	ship_test	Ship Test
phase	beta_test	Beta Test
phase	n/a	Not applicable to any particular phase

#### The reason a defect or feature is being accepted

Field Type	Value	Description
answerAccept	program_defect	The problem was due to a program error

Field Type	Value	Description
answerAccept	docs_defect	Documentation needs to be changed
answerAccept	docs_change	Documentation needs to address new features
answerAccept	plans_change	Plans or schedules need to be changed
answerAccept	new_function	New function will be added
answerAccept	redesign	Current function needs to be redesigned
answerAccept	fix_testcase	Testcase needs to be fixed
answerAccept	remove_code	Obsolete code needs to be removed
answerAccept	remove_support	Nonsupported functions need to be removed
answerAccept	comply_with	Coding practices and operation needs to comply with standards

#### The reason a defect or feature is being returned

Field Type	Value	Description
answerReturn	fixed	The problem is already fixed
answerReturn	future	Future releases or versions will address the defect or feature
answerReturn	duplicate	This is a duplicate of an existing defect or feature
answerReturn	usage_error	The problem is caused by incorrect usage
answerReturn	hardware_error	The problem is caused by a hardware error
answerReturn	info_needed	More information is required
answerReturn	limitation	This problem is a current limitation
answerReturn	suggestion	This problem is a suggestion, not an error
answerReturn	unrecreatable	The problem cannot be re-created
answerReturn	as_designed	The program works as designed
answerReturn	deviation	Code or documentation will deviate from the standards

#### The relationship of a part to the translation process

Field Type	Value	Description
translation	no	Part is not involved in translation
translation	yes	Part is translated into other languages
translation	related	Part is not translated but is related to translation process

#### Reasons for returning a feature

Field Type	Value	Description
featureReturn	fixed	The feature is already implemented

Field Type	Value	Description
featureReturn	future	Future releases or versions will address the feature
featureReturn	duplicate	This is a duplicate of an existing feature
featureReturn	info_needed	More information is required
featureReturn	deviation	Code or documentation will deviate from the standards
featureReturn	null	Null

#### Reasons for accepting a feature

Field Type	Value	Description
featureAccept	docs_change	Documentation needs to address new features
featureAccept	new_function	New function will be added
featureAccept	redesign	Current function needs to be redesigned
featureAccept	null	Null

#### Fields for specifying expressions

Field Type	Value	Description
serial	^NULL\$	Null value
serial	^[0-9]{6}\$	Six-numeral serial number

#### Phone numbers

Field Type	Value	Description
phone	^NULL\$	Null value
phone	^TL-[0-9]{3}-[0-9]{4}\$	Phone format TL-PPP-NNNN, P and N are numerals

#### A list from which more than one item can be selected

Field Type	Value	Description
list	item1	Item to be selected from a list
list	item2	Item to be selected from a list
list	item3	Item to be selected from a list
list	item4	Item to be selected from a list
list	item5	Item to be selected from a list

#### Field for noting code development iterations

Field Type	Value	Description
iteration	base_code	Base Code/Prior Release

<b>Field Type</b>	<b>Value</b>	<b>Description</b>
iteration	01	First iteration
iteration	02	Second iteration
iteration	03	Third iteration
iteration	04	Fourth iteration
iteration	05	Fifth iteration
iteration	06	Sixth iteration
iteration	07	Seventh iteration
iteration	08	Eighth iteration
iteration	09	Ninth iteration
iteration	10	Tenth iteration
iteration	11	Eleventh iteration
iteration	12	Twelfth iteration
iteration	13	Thirteenth iteration
iteration	14	Fourteenth iteration
iteration	15	Fifteenth iteration

**Describes the activity in progress when a defect was discovered**

<b>Field Type</b>	<b>Value</b>	<b>Description</b>
activityODC	review	Review or inspection
activityODC	ut/ft	Unit Test or Functional Test
activityODC	st	System Test
activityODC	id	Information Development
activityODC	customer	Customer use

**Identify specific intents or purposes for which an activity that triggered a defect was being performed**

<b>Field Type</b>	<b>Value</b>	<b>Description</b>
triggerODC	design	Design Nonconformance
triggerODC	flow	Understanding Flow
triggerODC	backward	Backward Compatibility
triggerODC	lateral	Lateral Compatibility
triggerODC	concurrency	Concurrency
triggerODC	document	Internal Document Consistency/Completeness
triggerODC	language	Language Dependencies
triggerODC	side	Side Effects
triggerODC	rare	Rare Situation
triggerODC	simple	Simple Path

Field Type	Value	Description
triggerODC	complex	Complex Path
triggerODC	coverage	Test Coverage
triggerODC	variation	Test Variation
triggerODC	sequencing	Test Sequencing
triggerODC	interaction	Test Interaction
triggerODC	workload	Workload Volume/Stress
triggerODC	recover	Recovery/Exception
triggerODC	startup	Startup/Restart
triggerODC	hw	Hardware Configuration
triggerODC	sw	Software Configuration
triggerODC	normal	Normal Mode
triggerODC	accuracy	The information does not describe the product correctly
triggerODC	clarity	The information is confusing or difficult to understand
triggerODC	completeness	Necessary information is missing
triggerODC	organization	The relationship between parts or between a part and the whole is not conveyed
triggerODC	retrievability	The information is difficult to find
triggerODC	style	The manner of expression is inappropriate or difficult to understand
triggerODC	task	The presentation of why and how to perform a task is inappropriate
triggerODC	aesthetics	The appearance and layout of the information is inappropriate

#### The impact a defect might have on customers if not fixed

Field Type	Value	Description
impactODC	installability	The ability of the customer to prepare and place the software in position for use
impactODC	security	The protection of systems, programs, and data from inadvertent or malicious destruction, alteration, or disclosure
impactODC	performance	The speed of the software as perceived by the customer and the customer's end users, in terms of their ability to perform their tasks
impactODC	maintenance	The ease of applying preventive or corrective fixes to the software
impactODC	serviceability	The ability to diagnose failures easily and quickly, with minimal impact to the customer

Field Type	Value	Description
impactODC	migration	The ease of upgrading to a current release
impactODC	documentation	The degree to which the publication aids provided for understanding the structure and intended uses of the software are correct and complete
impactODC	usability	The degree to which the software and publication aids enable the product to be easily understood and conveniently employed by its end user
impactODC	standards	The degree to which the software complies with established pertinent standards
impactODC	reliability	The ability of the software to consistently perform its intended function without unplanned interruption
impactODC	requirements	A customer expectation, with regard to capability, which was not known, understood, or prioritized as a requirement for the current product or release
impactODC	capability	The ability of the software to perform its intended functions, and satisfy known requirements

#### The aspect of the product that a defect is intended to address

Field Type	Value	Description
targetODC	requirements	Customer, market, or technical requirements
targetODC	design	Product design
targetODC	code	Product code
targetODC	build	Problems encountered during the driver build process, in library systems, or with management of change or version control
targetODC	information	Information/User Documentation
targetODC	ui	User Interface
targetODC	nls	National Language Support

#### Represents the actual correction that was made

Field Type	Value	Description
defTypeODC	assignment	Value(s) assigned incorrectly or not assigned at all
defTypeODC	checking	Errors caused by missing or incorrect validation of parameters or data in conditional statements

Field Type	Value	Description
defTypeODC	algorithm	Efficiency or correctness problems that affect the task and can be fixed by (re)implementing an algorithm or local data structure without the need for requesting a design change
defTypeODC	function	The error should require a formal design change
defTypeODC	timing	Necessary serialization of shared resource was missing
defTypeODC	interface	Communication problem between product components
defTypeODC	relationship	Problems related to associations among procedures, data structures and objects
defTypeODC	editorial	Defects relates to grammar, spelling, punctuation, organization, etc.
defTypeODC	technical	Defects related to the description of a product and its interfaces
defTypeODC	navigational	Defects that prevent users from finding needed information about a product
defTypeODC	GUI	Graphical User Interface
defTypeODC	cmdline	Command Line Interface
defTypeODC	panels	Panels
defTypeODC	na	Not Available

#### Indication of whether the defect was an omission, a commission, or extraneous

Field Type	Value	Description
qualifierODC	missing	The defect was to due to an error of omission
qualifierODC	incorrect	The defect was to due to an error of commission
qualifierODC	extraneous	The defect was to due to something not relevant or pertinent to the document or code

#### The source of the code or information that was fixed

Field Type	Value	Description
sourceODC	here	A defect is in code which was developed in house
sourceODC	reused	A defect is encountered using a part of a standard reuse library
sourceODC	outsourced	A defect is in a part provided by a vendor
sourceODC	reference	Defect contained in detailed descriptive information
sourceODC	tasks	Defect contained in guidance information

Field Type	Value	Description
sourceODC	presentation	Defect contained in graphical and other elements used to present the information
sourceODC	concepts	Defect contained in high level overview and conceptual information
sourceODC	examples	Examples
sourceODC	na	Not available or not applicable

**The history of the code or information that was fixed**

Field Type	Value	Description
srcHistoryODC	base	The defect is in part of the product which has not been modified by the current project and is not part of a standard reuse library
srcHistoryODC	new	The defect is in a function which was created by and for the current project and which introduces new function
srcHistoryODC	rewritten	The defect was introduced as a direct result of redesign and/or rewrite of old function in an attempt to improve its design or quality
srcHistoryODC	refixed	The defect was introduced by the solution provided to fix a previous defect

---

## Appendix C. User exit parameters

The following table shows the parameters passed to each user exit program defined for a specific TeamConnection action and ExitID. A description of the parameters follows the table on page 212.

**Note:** Parameters are not shown for exit ID 3. The parameters for exit ID 3 are the same as those passed to exit ID 0, with an additional parameter at the end to indicate the last user exit ID that has been executed successfully, for example, 0 or 1. The msgBuff parameter will always be null for exit ID 0, but will probably not be null for exit ID 3.

A parameter name followed by *not used* indicates that TeamConnection passes an empty string.

See “Chapter 9. Providing user exits” on page 103 for more information on user exits.

---

### Parameters passed to user exit programs

The figure that follows shows the parameters passed to each user exit program defined for a specific TeamConnection action and exit ID.

TeamConnection action	Exit ID	Parameters passed to the user exit program (see page 212 for definitions)
<b>Access</b>		
AccessCreate	0	NewOwner, component, authority, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	NewOwner, component, authority, effectiveUserID, VerboseFlag
	2	NewOwner, component, authority, effectiveUserID, VerboseFlag
AccessDelete	0	OldOwner, component, authority, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	OldOwner, component, authority, effectiveUserID, VerboseFlag
	2	OldOwner, component, authority, effectiveUserID, VerboseFlag
AccessRestrict	0	NewOwner, component, authority, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	NewOwner, component, authority, effectiveUserID, VerboseFlag
	2	NewOwner, component, authority, effectiveUserID, VerboseFlag
ApprovalAbstain	0	release, WorkAreaName, ApproverName, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	release, WorkAreaName, DefectOrFeatureName, ApproverName, workareaType, effectiveUserID, VerboseFlag

<b>TeamConnection action</b>	<b>Exit ID</b>	<b>Parameters passed to the user exit program (see page 212 for definitions)</b>
	2	release, WorkAreaName, DefectOrFeatureName, ApproverName, workareaType, effectiveUserID, VerboseFlag
ApprovalAccept	0	release, WorkAreaName, ApproverName, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	release, WorkAreaName, DefectOrFeatureName, ApproverName, workareaType, effectiveUserID, VerboseFlag
	2	release, WorkAreaName, DefectOrFeatureName, ApproverName, workareaType, effectiveUserID, VerboseFlag
ApprovalAssign	0	release, WorkAreaName, OldOwner, NewOwner, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	release, WorkAreaName, DefectOrFeatureName, OldOwner, NewOwner, workareaType, effectiveUserID, VerboseFlag
	2	release, WorkAreaName, DefectOrFeatureName, OldOwner, NewOwner, workareaType, effectiveUserID, VerboseFlag
ApprovalCreate	0	release, WorkAreaName, ApproverName, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	release, WorkAreaName, DefectOrFeatureName, ApproverName, workareaType, effectiveUserID, VerboseFlag
	2	release, WorkAreaName, DefectOrFeatureName, ApproverName, workareaType, effectiveUserID, VerboseFlag
ApprovalDelete	0	release, WorkAreaName, ApproverName, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	release, WorkAreaName, DefectOrFeatureName, ApproverName, workareaType, effectiveUserID, VerboseFlag
	2	release, WorkAreaName, DefectOrFeatureName, ApproverName, workareaType, effectiveUserID, VerboseFlag
ApprovalReject	0	release, WorkAreaName, ApproverName, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	release, WorkAreaName, DefectOrFeatureName, ApproverName, workareaType, effectiveUserID, VerboseFlag
	2	release, WorkAreaName, DefectOrFeatureName, ApproverName, workareaType, effectiveUserID, VerboseFlag
ApproverCreate	0	NewOwner, release, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	NewOwner, release, effectiveUserID, VerboseFlag
	2	NewOwner, release, effectiveUserID, VerboseFlag
ApproverDelete	0	OldOwner, release, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	OldOwner, release, effectiveUserID, VerboseFlag
	2	OldOwner, release, effectiveUserID, VerboseFlag
BecomeCreate	0	login, becomeLogin, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	login, becomeLogin, effectiveUserID, VerboseFlag

<b>TeamConnection action</b>	<b>Exit ID</b>	<b>Parameters passed to the user exit program (see page 212 for definitions)</b>
	2	login, becomeLogin, effectiveUserID, VerboseFlag
BecomeDelete	0	login, becomeLogin, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	login, becomeLogin, effectiveUserID, VerboseFlag
	2	login, becomeLogin, effectiveUserID, VerboseFlag
BuilderCreate	0	name, transmitFlag, temporaryfileonserver, release, condition, value, script, filetype, buildparameters, targetenvironment, timeout, processoroptions, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	name, temporaryfileonserver, release, condition, value, script, filetype, buildparameters, targetenvironment, timeout, processoroptions, effectiveUserID, VerboseFlag
	2	name, temporaryfileonserver, release, condition, value, script, filetype, buildparameters, targetenvironment, timeout, processoroptions, effectiveUserID, VerboseFlag
BuilderDelete	0	name, release, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	name, release, effectiveUserID, VerboseFlag
	2	name, release, effectiveUserID, VerboseFlag
BuilderExtract	0	name, release, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	name, release, effectiveUserID, VerboseFlag
	2	name, temporaryfileonserver, release, condition, value, script, filetype, buildparameters, targetenvironment, timeout, processoroptions, effectiveUserID, VerboseFlag
BuilderModify	0	name, transmitFlag, temporaryfileonserver, release, condition, value, script, filetype, buildparameters, targetenvironment, timeout, processoroptions, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	name, temporaryfileonserver, release, condition, value, script, filetype, buildparameters, targetenvironment, timeout, processoroptions, effectiveUserID, VerboseFlag
	2	name, temporaryfileonserver, release, condition, value, script, filetype, buildparameters, targetenvironment, timeout, processoroptions, effectiveUserID, VerboseFlag
BuilderView	0	name, release, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	name, release, effectiveUserID, VerboseFlag
	2	name, release, effectiveUserID, VerboseFlag
CollisionAccept	0	pathName, WorkAreaName, release, state, alternateversion, workareaType, typename, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	pathName, WorkAreaName, release, typename, effectiveUserID, VerboseFlag
	2	pathName, WorkAreaName, release, typename, effectiveUserID, VerboseFlag
CollisionReconc	0	pathName, WorkAreaName, release, state, alternateversion, workareaType, typename, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag

<b>TeamConnection action</b>	<b>Exit ID</b>	<b>Parameters passed to the user exit program (see page 212 for definitions)</b>
	1	pathName, WorkAreaName, release, typename, effectiveUserID, VerboseFlag
	2	pathName, WorkAreaName, release, typename, effectiveUserID, VerboseFlag
CollisionReject	0	pathName, WorkAreaName, release, state, alternateversion, workareaType, typename, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	pathName, WorkAreaName, release, typename, effectiveUserID, VerboseFlag
	2	pathName, WorkAreaName, release, typename, effectiveUserID, VerboseFlag
CompCreate	0	component, parentcomponent, owner, componentprocess, description, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	newcomponent, parentcomponent, owner, newcomponentprocess, description, effectiveUserID, VerboseFlag
	2	newcomponent, parentcomponent, owner, newcomponentprocess, description, effectiveUserID, VerboseFlag
CompDelete	0	component, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	component, effectiveUserID, VerboseFlag
	2	component, effectiveUserID, VerboseFlag
CompLink	0	component, parentcomponent, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	component, parentcomponent, effectiveUserID, VerboseFlag
	2	component, parentcomponent, effectiveUserID, VerboseFlag
CompModify	0	component, newcomponent, NewOwner, newdescription, newcomponentprocess, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	component, newcomponent, OldOwner, NewOwner, olddescription, newdescription, oldcomponentprocess, newcomponentprocess, dateoflastupdate, effectiveUserID, VerboseFlag
	2	name, newcomponent, NewOwner, description, process, effectiveUserID, VerboseFlag
CompRecreate	0	component, parentcomponent, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	component, parentcomponent, olddropDate, effectiveUserID, VerboseFlag
	2	component, parentcomponent, olddropDate, effectiveUserID, VerboseFlag
CompUnlink	0	component, parentcomponent, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	component, parentcomponent, effectiveUserID, VerboseFlag
	2	component, parentcomponent, effectiveUserID, VerboseFlag
CompView	0	component, displaytype, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	component, displaytype, effectiveUserID, VerboseFlag
	2	component, displaytype, effectiveUserID, VerboseFlag

<b>TeamConnection action</b>	<b>Exit ID</b>	<b>Parameters passed to the user exit program (see page 212 for definitions)</b>
CoreqCreate	0	release, primeworkareaname, secondworkareaname, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	release, primeworkareaname, secondworkareaname, primeworkareatype, secondworkareatype, effectiveUserID, VerboseFlag
	2	release, primeworkareaname, secondworkareaname, primeworkareatype, secondworkareatype, effectiveUserID, VerboseFlag
CoreqDelete	0	release, WorkAreaName, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	release, WorkAreaName, workareaType, effectiveUserID, VerboseFlag
	2	release, WorkAreaName, workareaType, effectiveUserID, VerboseFlag
DefectAccept	0	defectname, originaldefectname, answer, remarks, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	defectname, answer, remarks, configFields, effectiveUserID, VerboseFlag
	2	defectname, originaldefectname, answer, remarks, effectiveUserID, VerboseFlag
DefectAssign	0	defectname, newcomponent, NewOwner, remarks, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	defectname, newcomponent, NewOwner, remarks, effectiveUserID, VerboseFlag
	2	defectname, newcomponent, NewOwner, remarks, effectiveUserID, VerboseFlag
DefectCancel	0	defectname, originaldefectname, answer, remarks, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	defectname, remarks, effectiveUserID, VerboseFlag
	2	defectname, originaldefectname, answer, remarks, effectiveUserID, VerboseFlag
DefectComment	0	defectname, remarks, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	defectname, remarks, effectiveUserID, VerboseFlag
	2	defectname, remarks, effectiveUserID, VerboseFlag
DefectDesign	0	defectname, originaldefectname, answer, remarks, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	defectname, remarks, effectiveUserID, VerboseFlag
	2	defectname, originaldefectname, answer, remarks, effectiveUserID, VerboseFlag
DefectModify	0	defectname, newdefectname, severity, environmentname, prefix, reference, drivername, abstract, originator, answer, remarks, release, configFields, MessageBuffer, notesDB, notesID, effectiveUserID, TeamcUserID, VerboseFlag
	1	defectname, newdefectname, severity, environmentname, prefix, reference, drivername, abstract, originator, answer, remarks, release, configFields, dateoflastupdate, notesDB, notesID, effectiveUserID, VerboseFlag
	2	defectname, newdefectname, severity, environmentname, prefix, reference, drivername, abstract, originator, answer, remarks, release, configFields, notesDB, notesID, effectiveUserID, VerboseFlag

<b>TeamConnection action</b>	<b>Exit ID</b>	<b>Parameters passed to the user exit program (see page 212 for definitions)</b>
DefectOpen	0	component, prefix, severity, reference, environmentname, remarks, drivename, abstract, release, configFields, defectname, MessageBuffer, notesDB, notesID, effectiveUserID, TeamcUserID, VerboseFlag
	1	component, prefix, severity, reference, environmentname, remarks, drivename, abstract, release, configFields, defectname, effectiveuserarea, notesDB, notesID, effectiveUserID, VerboseFlag
	2	component, prefix, severity, reference, environmentname, remarks, drivename, abstract, release, configFields, defectname, effectiveuserarea, notesDB, notesID, effectiveUserID, VerboseFlag
DefectReopen	0	defectname, originaldefectname, answer, remarks, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	defectname, remarks, effectiveUserID, VerboseFlag
	2	defectname, originaldefectname, answer, remarks, effectiveUserID, VerboseFlag
DefectReturn	0	defectname, originaldefectname, answer, remarks, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	defectname, originaldefectname, answer, remarks, effectiveUserID, VerboseFlag
	2	defectname, originaldefectname, answer, remarks, effectiveUserID, VerboseFlag
DefectReview	0	defectname, originaldefectname, answer, remarks, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	defectname, remarks, effectiveUserID, VerboseFlag
	2	defectname, originaldefectname, answer, remarks, effectiveUserID, VerboseFlag
DefectSize	0	defectname, originaldefectname, answer, remarks, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	defectname, remarks, effectiveUserID, VerboseFlag
	2	defectname, originaldefectname, answer, remarks, effectiveUserID, VerboseFlag
DefectVerify	0	defectname, originaldefectname, answer, remarks, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	defectname, remarks, effectiveUserID, VerboseFlag
	2	defectname, originaldefectname, answer, remarks, effectiveUserID, VerboseFlag
DefectView	0	defectname, displaytype, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	defectname, displaytype, effectiveUserID, VerboseFlag
	2	defectname, displaytype, effectiveUserID, VerboseFlag
DriverAssign	0	drivename, release, NewOwner, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	drivename, release, NewOwner, driverstate, drivertype, effectiveUserID, VerboseFlag
	2	drivename, release, NewOwner, driverstate, drivertype, effectiveUserID, VerboseFlag

<b>TeamConnection action</b>	<b>Exit ID</b>	<b>Parameters passed to the user exit program (see page 212 for definitions)</b>
DriverCheck	0	drivername, release, longFlag, basename, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	drivername, release, longFlag, driverstate, drivertype, basename, effectiveUserID, VerboseFlag
	2	drivername, release, longFlag, driverstate, drivertype, basename, effectiveUserID, VerboseFlag
DriverCommit	0	drivername, release, forceFlag, ignoreFlag, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	drivername, release, drivertype, effectiveUserID, VerboseFlag
	2	drivername, release, drivertype, effectiveUserID, VerboseFlag
DriverComplete	0	drivername, release, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	drivername, release, drivertype, effectiveUserID, VerboseFlag
	2	drivername, release, drivertype, effectiveUserID, VerboseFlag
DriverCreate	0	drivername, release, drivertype, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	drivername, release, drivertype, effectiveUserID, VerboseFlag
	2	drivername, release, drivertype, effectiveUserID, VerboseFlag
DriverDelete	0	drivername, release, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	drivername, release, driverstate, drivertype, effectiveUserID, VerboseFlag
	2	drivername, release, driverstate, drivertype, effectiveUserID, VerboseFlag
DriverExtract	0	drivername, release, root, nokeysFlag, ExtractType, fmask, dmask, crlfFlag, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	drivername, release, root, nokeysFlag, ExtractType, fmask, dmask, crlfFlag, driverstate, drivertype, effectiveUserID, VerboseFlag
	2	drivername, release, root, nokeysFlag, ExtractType, fmask, dmask, crlfFlag, driverstate, drivertype, effectiveUserID, VerboseFlag
DriverFreeze	0	drivername, release, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	drivername, release, driverstate, drivertype, effectiveUserID, VerboseFlag
	2	drivername, release, driverstate, drivertype, effectiveUserID, VerboseFlag
DriverModify	0	drivername, newdrivername, release, newdrivertype, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	drivername, newdrivername, release, oldtype, newtype, driverstate, dateoflastupdate, effectiveUserID, VerboseFlag
	2	drivername, newdrivername, release, oldtype, newtype, driverstate, effectiveUserID, VerboseFlag
DriverRefresh	0	drivername, release, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	drivername, release, driverstate, drivertype, effectiveUserID, VerboseFlag
	2	drivername, release, driverstate, drivertype, effectiveUserID, VerboseFlag

<b>TeamConnection action</b>	<b>Exit ID</b>	<b>Parameters passed to the user exit program (see page 212 for definitions)</b>
DriverRestrict	0	drivename, release, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	drivename, release, drivertype, effectiveUserID, VerboseFlag
	2	drivename, release, drivertype, effectiveUserID, VerboseFlag
DriverView	0	drivename, release, displaytype, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	drivename, release, displaytype, driverstate, drivertype, effectiveUserID, VerboseFlag
	2	drivename, release, displaytype, driverstate, drivertype, effectiveUserID, VerboseFlag
MemberCreate	0	drivename, release, WorkAreaName, forceFlag, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	drivename, release, WorkAreaName, DefectOrFeatureName, workareastate, workareaType, drivertype, effectiveUserID, VerboseFlag
	2	drivename, release, WorkAreaName, DefectOrFeatureName, workareastate, workareaType, drivertype, effectiveUserID, VerboseFlag
MemberDelete	0	drivename, release, numberofworkareas, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	drivename, release, drivertype, effectiveUserID, VerboseFlag
	2	drivename, release, drivertype, effectiveUserID, VerboseFlag
EnvCreate	0	environmentname, release, testersname, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	environmentname, release, testersname, effectiveUserID, VerboseFlag
	2	environmentname, release, testersname, effectiveUserID, VerboseFlag
EnvDelete	0	environmentname, release, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	environmentname, release, effectiveUserID, VerboseFlag
	2	environmentname, release, effectiveUserID, VerboseFlag
EnvModify	0	environmentname, release, newtestersname, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	environmentname, release, newtestersname, effectiveUserID, VerboseFlag
	2	environmentname, release, newtestersname, effectiveUserID, VerboseFlag
FeatureAccept	0	featurename, originalfeaturename, answer, remarks, StandardFields, configFields, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	featurename, remarks, configFields, answer, effectiveUserID, VerboseFlag
	2	featurename, originalfeaturename, remarks, answer, effectiveUserID, VerboseFlag
FeatureAssign	0	featurename, newcomponent, NewOwner, remarks, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	featurename, newcomponent, NewOwner, remarks, effectiveUserID, VerboseFlag

<b>TeamConnection action</b>	<b>Exit ID</b>	<b>Parameters passed to the user exit program (see page 212 for definitions)</b>
	2	featurename, newcomponent, NewOwner, remarks, effectiveUserID, VerboseFlag
FeatureCancel	0	featurename, originalfeaturename, answer, remarks, StandardFields, configFields, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	featurename, remarks, effectiveUserID, VerboseFlag
	2	featurename, originalfeaturename, remarks, answer, effectiveUserID, VerboseFlag
FeatureComment	0	featurename, remarks, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	featurename, remarks, effectiveUserID, VerboseFlag
	2	featurename, remarks, effectiveUserID, VerboseFlag
FeatureDesign	0	featurename, originalfeaturename, answer, remarks, StandardFields, configFields, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	featurename, remarks, effectiveUserID, VerboseFlag
	2	featurename, originalfeaturename, remarks, answer, effectiveUserID, VerboseFlag
FeatureModify	0	featurename, newfeaturename, prefix, reference, abstract, originator, remarks, configFields, MessageBuffer, answer, release, notesDB, notesID, effectiveUserID, TeamcUserID, VerboseFlag
	1	featurename, newfeaturename, prefix, reference, abstract, originator, remarks, configFields, dateoflastupdate, answer, release, notesDB, notesID, effectiveUserID, VerboseFlag
	2	featurename, newfeaturename, prefix, reference, abstract, originator, remarks, configFields, answer, release, notesDB, notesID, effectiveUserID, VerboseFlag
FeatureOpen	0	component, prefix, reference, remarks, abstract, configFields, featurename, MessageBuffer, release, notesDB, notesID, effectiveUserID, TeamcUserID, VerboseFlag
	1	component, prefix, reference, remarks, abstract, configFields, featurename, release, notesDB, notesID, effectiveUserID, VerboseFlag
	2	component, prefix, reference, remarks, abstract, configFields, featurename, release, notesDB, notesID, effectiveUserID, VerboseFlag
FeatureReopen	0	featurename, originalfeaturename, answer, remarks, StandardFields, configFields, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	featurename, remarks, effectiveUserID, VerboseFlag
	2	featurename, originalfeaturename, remarks, answer, effectiveUserID, VerboseFlag
FeatureReturn	0	featurename, originalfeaturename, answer, remarks, StandardFields, configFields, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	featurename, originalfeaturename, remarks, answer, effectiveUserID, VerboseFlag
	2	featurename, originalfeaturename, remarks, answer, effectiveUserID, VerboseFlag
FeatureReview	0	featurename, originalfeaturename, answer, remarks, StandardFields, configFields, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	featurename, remarks, effectiveUserID, VerboseFlag

<b>TeamConnection action</b>	<b>Exit ID</b>	<b>Parameters passed to the user exit program (see page 212 for definitions)</b>
	2	featurename, originalfeaturename, remarks, answer, effectiveUserID, VerboseFlag
FeatureSize	0	featurename, originalfeaturename, answer, remarks, StandardFields, configFields, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	featurename, remarks, effectiveUserID, VerboseFlag
	2	featurename, originalfeaturename, remarks, answer, effectiveUserID, VerboseFlag
FeatureVerify	0	featurename, originalfeaturename, answer, remarks, StandardFields, configFields, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	featurename, remarks, effectiveUserID, VerboseFlag
	2	featurename, originalfeaturename, remarks, answer, effectiveUserID, VerboseFlag
FeatureView	0	featurename, displaytype, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	featurename, displaytype, effectiveUserID, VerboseFlag
	2	featurename, displaytype, effectiveUserID, VerboseFlag
FixActive	0	WorkAreaName, release, component, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	WorkAreaName, DefectOrFeatureName, release, component, type, effectiveUserID, VerboseFlag
	2	WorkAreaName, DefectOrFeatureName, release, component, type, effectiveUserID, VerboseFlag
FixAssign	0	WorkAreaName, release, component, NewOwner, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	WorkAreaName, release, component, NewOwner, type, effectiveUserID, VerboseFlag
	2	WorkAreaName, release, component, NewOwner, type, effectiveUserID, VerboseFlag
FixComplete	0	WorkAreaName, release, component, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	WorkAreaName, DefectOrFeatureName, release, component, type, effectiveUserID, VerboseFlag
	2	WorkAreaName, DefectOrFeatureName, release, component, type, effectiveUserID, VerboseFlag
FixCreate	0	WorkAreaName, release, component, NewOwner, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	WorkAreaName, DefectOrFeatureName, release, component, NewOwner, type, effectiveUserID, VerboseFlag
	2	WorkAreaName, DefectOrFeatureName, release, component, NewOwner, type, effectiveUserID, VerboseFlag
FixDelete	0	WorkAreaName, release, component, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag

<b>TeamConnection action</b>	<b>Exit ID</b>	<b>Parameters passed to the user exit program (see page 212 for definitions)</b>
	1	WorkAreaName, release, component, type, effectiveUserID, VerboseFlag
	2	WorkAreaName, release, component, type, effectiveUserID, VerboseFlag
HostCreate	0	NewOwner, login@hostname, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	NewOwner, login@hostname, effectiveUserID, VerboseFlag
	2	NewOwner, login@hostname, effectiveUserID, VerboseFlag
HostDelete	0	OldOwner, login@hostname, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	OldOwner, login@hostname, effectiveUserID, VerboseFlag
	2	OldOwner, login@hostname, effectiveUserID, VerboseFlag
NotifyCreate	0	NewOwner, component, interestgroupname, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	NewOwner, component, interestgroupname, effectiveUserID, VerboseFlag
	2	NewOwner, component, interestgroupname, effectiveUserID, VerboseFlag
NotifyDelete	0	OldOwner, component, interestgroupname, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	OldOwner, component, interestgroupname, effectiveUserID, VerboseFlag
	2	OldOwner, component, interestgroupname, effectiveUserID, VerboseFlag
ParserCreate	0	description, release, parsercommand, paths, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	description, release, parsercommand, paths, effectiveUserID, VerboseFlag
	2	description, release, parsercommand, paths, effectiveUserID, VerboseFlag
ParserDelete	0	description, release, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	description, release, effectiveUserID, VerboseFlag
	2	description, release, effectiveUserID, VerboseFlag
ParserModify	0	description, release, parsercommand, paths, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	description, release, parsercommand, paths, effectiveUserID, VerboseFlag
	2	description, release, parsercommand, paths, effectiveUserID, VerboseFlag
ParserView	0	description, release, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	description, release, effectiveUserID, VerboseFlag
	2	description, release, effectiveUserID, VerboseFlag

<b>TeamConnection action</b>	<b>Exit ID</b>	<b>Parameters passed to the user exit program (see page 212 for definitions)</b>
PartAdd	0	partpathName, transmitFlag, filenameonclient, temporaryfileonserver, release, component, filetype, WorkAreaName, fMode, parentname, parsername, buildername, relationtoparent, buildparameters, parttype, parenttype, temporaryFlag, StandardFields, configFields, translation, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	partpathName, temporaryfileonserver, release, component, filetype, WorkAreaName, remarks, fMode, parentname, parsername, buildername, relationtoparent, buildparameters, parttype, parenttype, temporaryFlag, configFields, translation, effectiveUserID, VerboseFlag
	2	partpathName, temporaryfileonserver, release, component, filetype, WorkAreaName, remarks, fMode, parentname, parsername, buildername, relationtoparent, buildparameters, parttype, parenttype, temporaryFlag, configFields, translation, effectiveUserID, VerboseFlag
PartBuild	0	partpathName, release, WorkAreaName, buildmode, poolname, buildparameters, cancelFlag, detailfilename, clientportname, parttype, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	partpathName, WorkAreaName, release, component, buildmode, poolname, buildparameters, cancelFlag, detailfilename, clienthostname, clientportname, parttype, effectiveUserID, VerboseFlag
	2	partpathName, WorkAreaName, release, component, buildmode, poolname, buildparameters, cancelFlag, detailfilename, clienthostname, clientportname, parttype, effectiveUserID, VerboseFlag
PartCheckIn	0	partpathName, transmitFlag, filenameonclient, temporaryfileonserver, release, forceFlag, WorkAreaName, commonFlag, filetype, parttype, retainlockFlag, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	partpathName, temporaryfileonserver, release, component, forceFlag, WorkAreaName, remarks, commonreleases, filetype, parttype, retainlockFlag, configFields, effectiveUserID, VerboseFlag
	2	partpathName, temporaryfileonserver, release, component, versionname, forceFlag, WorkAreaName, remarks, commonreleases, filetype, parttype, retainlockFlag, configFields, effectiveUserID, VerboseFlag
PartCheckOut	0	partpathName, temporaryfileonserver, release, forceFlag, WorkAreaName, parttype, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	partpathName, temporaryfileonserver, release, filetype, component, versionname, forceFlag, workareaname, parttype, configFields, effectiveUserID, VerboseFlag
	2	partpathName, temporaryfileonserver, release, filetype, component, versionname, forceFlag, workareaname, parttype, configFields, effectiveUserID, VerboseFlag
PartChildInfo	0	partpathName, release, versionname, WorkAreaName, displaytype, relationtoparent, parttype, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	partpathName, release, versionname, WorkAreaName, displaytype, relationtoparent, parttype, effectiveUserID, VerboseFlag
	2	partpathName, release, versionname, WorkAreaName, displaytype, relationtoparent, parttype, effectiveUserID, VerboseFlag

<b>TeamConnection action</b>	<b>Exit ID</b>	<b>Parameters passed to the user exit program (see page 212 for definitions)</b>
PartConnect	0	partpathName, release, WorkAreaName, parentname, relationtoparent, parttype, parenttype, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	partpathName, release, WorkAreaName, component, parentname, relationtoparent, parttype, parenttype, effectiveUserID, VerboseFlag
	2	partpathName, release, WorkAreaName, component, parentname, relationtoparent, parttype, parenttype, effectiveUserID, VerboseFlag
PartDelete	0	partpathName, release, forceFlag, WorkAreaName, commonFlag, parttype, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	partpathName, release, forceFlag, WorkAreaName, commonFlag, component, parttype, commonRelBuffer, effectiveUserID, VerboseFlag
	2	partpathName, release, forceFlag, WorkAreaName, commonFlag, component, parttype, commonRelBuffer, effectiveUserID, VerboseFlag
PartDisconnect	0	partpathName, release, WorkAreaName, parentname, parttype, parenttype, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	partpathName, release, WorkAreaName, component, parentname, parttype, parenttype, effectiveUserID, VerboseFlag
	2	partpathName, release, WorkAreaName, component, parentname, parttype, parenttype, effectiveUserID, VerboseFlag
PartExtract	0	partpathName, temporaryfileonserver, release, nokeysFlag, WorkAreaName, versionname, parttype, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	partpathName, temporaryfileonserver, release, nokeysFlag, WorkAreaName, versionname, component, parttype, configFields, effectiveUserID, VerboseFlag
	2	partpathName, temporaryfileonserver, release, nokeysFlag, WorkAreaName, versionname, component, parttype, configFields, effectiveUserID, VerboseFlag
PartLink	0	partpathName, sourcecrelease, release, sourceworkareaname, sourceversion, parttype, targetworkareaname, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	partpathName, sourceworkareaname, sourcecrelease, targetrelease, sourceversion, component, parttype, targetworkareaname, effectiveUserID, VerboseFlag
	2	partpathName, sourceworkareaname, sourcecrelease, targetrelease, sourceversion, component, parttype, targetworkareaname, effectiveUserID, VerboseFlag
PartLock	0	partpathName, temporaryfileonserver, release, forceFlag, WorkAreaName, parttype, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	partpathName, release, forceFlag, WorkAreaName, filetype, component, versionname, parttype, configFields, effectiveUserID, VerboseFlag
	2	partpathName, release, forceFlag, WorkAreaName, filetype, component, versionname, parttype, configFields, effectiveUserID, VerboseFlag
PartMark	0	partpathName, release, versionname, WorkAreaName, translationstate, parttype, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	partpathName, release, versionname, WorkAreaName, translationstate, component, parttype, configFields, effectiveUserID, VerboseFlag

<b>TeamConnection action</b>	<b>Exit ID</b>	<b>Parameters passed to the user exit program (see page 212 for definitions)</b>
	2	partpathName, release, versionname, WorkAreaName, translationstate, component, parttype, configFields, effectiveUserID, VerboseFlag
PartMerge	0	partpathName, release, WorkAreaName, FromRelease, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	partpathName, release, WorkAreaName, FromRelease, effectiveUserID, VerboseFlag
	2	partpathName, release, WorkAreaName, FromRelease, effectiveUserID, VerboseFlag
PartModify	0	partpathName, release, newcomponent, newfMode, configFields, WorkAreaName, filetype, parsername, buildername, buildparameters, parttype, temporaryfilename, translation, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	partpathName, release, oldcomponent, newcomponent, oldfMode, newfMode, configFields, WorkAreaName, dateoflastupdate, filetype, parsername, buildername, buildparameters, parttype, temporaryFlag, translation, effectiveUserID, VerboseFlag
	2	partpathName, release, oldcomponent, newcomponent, oldfMode, newfMode, configFields, WorkAreaName, dateoflastupdate, filetype, parsername, buildername, buildparameters, parttype, temporaryFlag, translation, effectiveUserID, VerboseFlag
PartReconcile	0	partpathName, release, WorkAreaName, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	partpathName, release, WorkAreaName, effectiveUserID, VerboseFlag
	2	partpathName, release, WorkAreaName, effectiveUserID, VerboseFlag
PartRecreate	0	partpathName, release, forceFlag, WorkAreaName, commonFlag, parttype, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	partpathName, release, forceFlag, WorkAreaName, commonFlag, component, olddropDate, parttype, commonRelBuffer, effectiveUserID, VerboseFlag
	2	partpathName, release, forceFlag, WorkAreaName, commonFlag, component, olddropDate, parttype, commonRelBuffer, effectiveUserID, VerboseFlag
PartRefresh	0	partpathName, sourcecrelease, release, sourceworkkareaname, sourceversion, parttype, targetworkkareaname, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	partpathName, sourceworkkareaname, sourcecrelease, targetrelease, sourceversion, component, parttype, targetworkkareaname, effectiveUserID, VerboseFlag
	2	partpathName, sourceworkkareaname, sourcecrelease, targetrelease, sourceversion, component, parttype, targetworkkareaname, effectiveUserID, VerboseFlag
PartRename	0	partpathName, release, nuPartPathName, forceFlag, WorkAreaName, commonFlag, parttype, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	partpathName, release, nuPartPathName, forceFlag, WorkAreaName, commonFlag, component, parttype, commonRelBuffer, effectiveUserID, VerboseFlag
	2	partpathName, release, nuPartPathName, forceFlag, WorkAreaName, commonFlag, component, parttype, commonRelBuffer, effectiveUserID, VerboseFlag
PartTouch	0	partpathName, release, forceFlag, WorkAreaName, commonFlag, parttype, creatChangeFlag, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag

<b>TeamConnection action</b>	<b>Exit ID</b>	<b>Parameters passed to the user exit program (see page 212 for definitions)</b>
	1	partpathName, release, forceFlag, WorkAreaName, commonFlag, component, parttype, effectiveUserID, VerboseFlag
	2	partpathName, release, forceFlag, WorkAreaName, commonFlag, component, parttype, effectiveUserID, VerboseFlag
PartUndo	0	partpathName, release, forceFlag, WorkAreaName, commonFlag, parttype, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	partpathName, release, forceFlag, WorkAreaName, commonFlag, component, versionname, parttype, commonRelBuffer, timeNow, effectiveUserID, VerboseFlag
	2	partpathName, release, forceFlag, WorkAreaName, commonFlag, component, versionname, parttype, commonRelBuffer, timeNow, effectiveUserID, VerboseFlag
PartUnlock	0	partpathName, WorkAreaName, release, parttype, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	partpathName, WorkAreaName, release, component, parttype, configFields, effectiveUserID, VerboseFlag
	2	partpathName, WorkAreaName, release, component, parttype, configFields, effectiveUserID, VerboseFlag
PartView	0	partpathName, release, versionname, WorkAreaName, displaytype, parttype, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	partpathName, release, versionname, WorkAreaName, displaytype, parttype, effectiveUserID, VerboseFlag
	2	partpathName, release, versionname, WorkAreaName, displaytype, parttype, effectiveUserID, VerboseFlag
PartViewmsg	0	partpathName, release, versionname, WorkAreaName, parttype, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	partpathName, release, component, versionname, WorkAreaName, parttype, effectiveUserID, VerboseFlag
	2	partpathName, release, component, versionname, WorkAreaName, parttype, effectiveUserID, VerboseFlag
PartRestrict	0	partpathName, release, cancelFlag, parttype, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	partpathName, release, cancelFlag, component, parttype, effectiveUserID, VerboseFlag
	2	partpathName, release, cancelFlag, component, parttype, effectiveUserID, VerboseFlag
PartOverrideR	0	partpathName, release, WorkAreaName, login, cancelFlag, parttype, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	partpathName, release, WorkAreaName, login, cancelFlag, component, parttype, effectiveUserID, VerboseFlag
	2	partpathName, release, WorkAreaName, login, cancelFlag, component, parttype, effectiveUserID, VerboseFlag
PrereqCreate	0	release, primeworkareaname, secondworkareaname, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag

<b>TeamConnection action</b>	<b>Exit ID</b>	<b>Parameters passed to the user exit program (see page 212 for definitions)</b>
	1	release, primeworkareaname, secondworkareaname, primeworkareatype, secondworkareatype, effectiveUserID, VerboseFlag
	2	release, primeworkareaname, secondworkareaname, primeworkareatype, secondworkareatype, effectiveUserID, VerboseFlag
PrereqDelete	0	release, primeworkareaname, secondworkareaname, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	release, primeworkareaname, secondworkareaname, primeworkareatype, secondworkareatype, effectiveUserID, VerboseFlag
	2	release, primeworkareaname, secondworkareaname, primeworkareatype, secondworkareatype, effectiveUserID, VerboseFlag
ReleaseCreate	0	release, component, newreleaseprocess, environmentname, testersname, ApproverName, description, releaseowner, autoprune, coupling, developmentmode, releasedatabasename, outputversions, StandardFields, configFields, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	release, component, newreleaseprocess, environmentname, testersname, ApproverName, description, releaseowner, coupling, effectiveUserID, VerboseFlag
	2	release, component, newreleaseprocess, environmentname, testersname, ApproverName, description, releaseowner, coupling, effectiveUserID, VerboseFlag
ReleaseDelete	0	release, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	release, effectiveUserID, VerboseFlag
	2	release, effectiveUserID, VerboseFlag
ReleaseExtract	0	release, root, nokeysFlag, committedFlag, date, fmask, dmask, complist, crffFlag, versionname, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	release, root, nokeysFlag, committedFlag, date, fmask, dmask, crffFlag, complist, effectiveUserID, VerboseFlag
	2	release, root, nokeysFlag, committedFlag, date, fmask, dmask, crffFlag, complist, effectiveUserID, VerboseFlag
ReleaseLink	0	release, FromRelease, WorkAreaName, newworkareaname, fromversionname, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	release, FromRelease, WorkAreaName, fromworkareaname, fromversionname, effectiveUserID, VerboseFlag
	2	release, FromRelease, WorkAreaName, fromworkareaname, fromversionname, effectiveUserID, VerboseFlag
ReleaseMerge	0	release, WorkAreaName, FromRelease, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	release, WorkAreaName, FromRelease, effectiveUserID, VerboseFlag
	2	release, WorkAreaName, FromRelease, effectiveUserID, VerboseFlag
ReleaseModify	0	release, newrelease, component, description, newreleaseprocess, environmentname, testersname, ApproverName, NewOwner, autoprune, coupling, outputversions, StandardFields, configFields, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag

<b>TeamConnection action</b>	<b>Exit ID</b>	<b>Parameters passed to the user exit program (see page 212 for definitions)</b>
	1	release, newrelease, oldcomponent, newcomponent, olddescription, newdescription, oldreleaseprocess, newreleaseprocess, environmentname, testersname, ApproverName, OldOwnerName, NewOwner, dateoflastupdate, coupling, effectiveUserID, VerboseFlag
	2	release, newrelease, component, description, newreleaseprocess, environmentname, testersname, ApproverName, NewOwner, coupling, effectiveUserID, VerboseFlag
ReleasePrune	0	release, versionname, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	release, versionname, effectiveUserID, VerboseFlag
	2	release, versionname, effectiveUserID, VerboseFlag
ReleaseRecreate	0	release, environmentname, testersname, ApproverName, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	release, lastdropdate, environment, testersname, ApproverName, effectiveUserID, VerboseFlag
	2	release, lastdropdate, environment, testersname, ApproverName, effectiveUserID, VerboseFlag
ReleaseView	0	release, reporttype, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	release, reporttype, effectiveUserID, VerboseFlag
	2	release, reporttype, effectiveUserID, VerboseFlag
Report	0	viewname, reportcriteria, parent, release, WorkAreaName, versionname, reporttype, parenttype, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	viewname, reportcriteria, parent, effectiveUserID, VerboseFlag
	2	viewname, reportcriteria, parent, effectiveUserID, VerboseFlag
Report	0	dbobjnames, selspec, reportcriteria, colspec, queryopt, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	viewname, reportcriteria, selspec, colspec, queryChar, effectiveUserID, VerboseFlag
	2	viewname, reportcriteria, selspec, colspec, queryChar, effectiveUserID, VerboseFlag
SizeAccept	0	WorkAreaName, component, release, sizetext, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	WorkAreaName, component, release, sizetext, sizetype, effectiveUserID, VerboseFlag
	2	WorkAreaName, component, release, sizetext, sizetype, effectiveUserID, VerboseFlag
SizeAssign	0	WorkAreaName, component, release, NewOwner, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	WorkAreaName, component, release, NewOwner, sizetype, effectiveUserID, VerboseFlag
	2	WorkAreaName, component, release, NewOwner, sizetype, effectiveUserID, VerboseFlag

<b>TeamConnection action</b>	<b>Exit ID</b>	<b>Parameters passed to the user exit program (see page 212 for definitions)</b>
SizeCreate	0	WorkAreaName, component, release, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	WorkAreaName, component, release, sizetype, effectiveUserID, VerboseFlag
	2	WorkAreaName, component, release, sizetype, effectiveUserID, VerboseFlag
SizeDelete	0	WorkAreaName, component, release, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	WorkAreaName, component, release, sizetype, effectiveUserID, VerboseFlag
	2	WorkAreaName, component, release, sizetype, effectiveUserID, VerboseFlag
SizeReject	0	WorkAreaName, component, release, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	WorkAreaName, component, release, sizetype, effectiveUserID, VerboseFlag
	2	WorkAreaName, component, release, sizetype, effectiveUserID, VerboseFlag
TestAbstain	0	WorkAreaName, TesterName, release, environmentname, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	WorkAreaName, DefectOrFeatureName, TesterName, release, environmentname, type, effectiveUserID, VerboseFlag
	2	WorkAreaName, DefectOrFeatureName, TesterName, release, environmentname, type, effectiveUserID, VerboseFlag
TestAccept	0	WorkAreaName, TesterName, release, environmentname, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	WorkAreaName, DefectOrFeatureName, TesterName, release, environmentname, type, effectiveUserID, VerboseFlag
	2	WorkAreaName, DefectOrFeatureName, TesterName, release, environmentname, type, effectiveUserID, VerboseFlag
TestAssign	0	WorkAreaName, OldOwner, NewOwner, release, environmentname, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	WorkAreaName, OldOwner, NewOwner, release, environmentname, workareaType, effectiveUserID, VerboseFlag
	2	WorkAreaName, OldOwner, NewOwner, release, environmentname, workareaType, effectiveUserID, VerboseFlag
TestCreate	0	DefectOrFeatureName, TesterName, release, environmentname, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	DefectOrFeatureName, TesterName, release, environmentname, type, effectiveUserID, VerboseFlag
	2	DefectOrFeatureName, TesterName, release, environmentname, type, effectiveUserID, VerboseFlag
TestDelete	0	DefectOrFeatureName, TesterName, release, environmentname, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	DefectOrFeatureName, TesterName, release, environmentname, type, effectiveUserID, VerboseFlag

<b>TeamConnection action</b>	<b>Exit ID</b>	<b>Parameters passed to the user exit program (see page 212 for definitions)</b>
	2	DefectOrFeatureName, TesterName, release, environmentname, type, effectiveUserID, VerboseFlag
TestReject	0	WorkAreaName, TesterName, release, environmentname, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	WorkAreaName, DefectOrFeatureName, TesterName, release, environmentname, type, effectiveUserID, VerboseFlag
	2	WorkAreaName, DefectOrFeatureName, TesterName, release, environmentname, type, effectiveUserID, VerboseFlag
UserCreate	0	login, usersfullname, area, sendmailaddress, superuserprivilegeFlag, configFields, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	login, usersfullname, area, sendmailaddress, superuserprivilegeFlag, configFields, effectiveUserID, VerboseFlag
	2	login, usersfullname, area, sendmailaddress, superuserprivilegeFlag, configFields, effectiveUserID, VerboseFlag
UserDelete	0	login, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	login, usersfullname, effectiveUserID, VerboseFlag
	2	login, usersfullname, effectiveUserID, VerboseFlag
UserModify	0	login, newlogin, newusersfullname, newarea, newuserssendmailaddress, newsuperuserprivilegeFlag, configFields, passwordlength, oldpassword, newpassword, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	login, newlogin, oldusersfullname, newusersfullname, oldarea, newarea, oldsendmailaddress, newsendmailaddress, oldsuperuserprivilegeFlag, newsuperuserprivilegeFlag, configFields, dateoflastupdate, effectiveUserID, VerboseFlag
	2	login, newlogin, newusersfullname, newarea, newuserssendmailaddress, newsuperuserprivilegeFlag, configFields, effectiveUserID, VerboseFlag
UserRecreate	0	login, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	login, usersfullname, olddropDate, effectiveUserID, VerboseFlag
	2	login, usersfullname, olddropDate, effectiveUserID, VerboseFlag
UserView	0	login, displaytype, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	login, displaytype, effectiveUserID, VerboseFlag
	2	login, displaytype, effectiveUserID, VerboseFlag
VerifyAbstain	0	WorkAreaName, TesterName, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	WorkAreaName, TesterName, type, effectiveUserID, VerboseFlag
	2	WorkAreaName, TesterName, type, effectiveUserID, VerboseFlag
VerifyAccept	0	WorkAreaName, TesterName, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	WorkAreaName, TesterName, type, effectiveUserID, VerboseFlag

<b>TeamConnection action</b>	<b>Exit ID</b>	<b>Parameters passed to the user exit program (see page 212 for definitions)</b>
	2	WorkAreaName, TesterName, type, effectiveUserID, VerboseFlag
VerifyAssign	0	WorkAreaName, OldOwner, NewOwner, effectiveUserID, TesterName, VerboseFlag
	1	WorkAreaName, OldOwner, NewOwner, type, effectiveUserID, VerboseFlag
	2	WorkAreaName, OldOwner, NewOwner, type, effectiveUserID, VerboseFlag
VerifyReject	0	WorkAreaName, TesterName, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	WorkAreaName, TesterName, type, effectiveUserID, VerboseFlag
	2	WorkAreaName, TesterName, type, effectiveUserID, VerboseFlag
WorkAreaAssign	0	release, WorkAreaName, NewOwner, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	WorkAreaName, release, DefectOrFeatureName, NewOwner, workareaType, effectiveUserID, VerboseFlag
	2	WorkAreaName, release, DefectOrFeatureName, NewOwner, workareaType, effectiveUserID, VerboseFlag
WorkAreaCancel	0	release, WorkAreaName, forceFlag, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	WorkAreaName, release, DefectOrFeatureName, workareaType, effectiveUserID, VerboseFlag
	2	WorkAreaName, release, DefectOrFeatureName, workareaType, effectiveUserID, VerboseFlag
WorkAreaCheck	0	release, WorkAreaName, driver, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	release, WorkAreaName, drivername, workareaType, effectiveUserID, VerboseFlag
	2	release, WorkAreaName, drivername, workareaType, effectiveUserID, VerboseFlag
WorkAreaCommit	0	release, WorkAreaName, forceFlag, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	WorkAreaName, release, DefectOrFeatureName, workareaType, effectiveUserID, VerboseFlag
	2	WorkAreaName, release, DefectOrFeatureName, workareaType, effectiveUserID, VerboseFlag
WorkAreaCompleat	0	release, WorkAreaName, forceFlag, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	WorkAreaName, release, DefectOrFeatureName, workareaType, effectiveUserID, VerboseFlag
	2	WorkAreaName, release, DefectOrFeatureName, workareaType, effectiveUserID, VerboseFlag
WorkAreaCreate	0	release, WorkAreaName, DefectOrFeatureName, target, workareaOwner, StandardFields, configFields, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag

<b>TeamConnection action</b>	<b>Exit ID</b>	<b>Parameters passed to the user exit program (see page 212 for definitions)</b>
	1	WorkAreaName, release, DefectOrFeatureName, target, workareaOwner, workareaType, effectiveUserID, VerboseFlag
	2	WorkAreaName, release, DefectOrFeatureName, target, workareaOwner, workareaType, effectiveUserID, VerboseFlag
WorkAreaExtract	0	WorkAreaName, release, root, nokeysFlag, fmask, dmask, crlfFlag, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	WorkAreaName, release, root, nokeysFlag, fmask, dmask, crlfFlag, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	2	WorkAreaName, release, root, nokeysFlag, fmask, dmask, crlfFlag, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
WorkAreaFix	0	release, WorkAreaName, forceFlag, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	WorkAreaName, release, DefectOrFeatureName, workareaType, effectiveUserID, VerboseFlag
	2	WorkAreaName, release, DefectOrFeatureName, workareaType, effectiveUserID, VerboseFlag
WorkAreaFreeze	0	release, WorkAreaName, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	release, WorkAreaName, workareatarget, workareaType, effectiveUserID, VerboseFlag
	2	release, WorkAreaName, workareatarget, workareaType, effectiveUserID, VerboseFlag
WorkAreaIntegra	0	release, WorkAreaName, forceFlag, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	WorkAreaName, release, DefectOrFeatureName, workareaType, effectiveUserID, VerboseFlag
	2	WorkAreaName, release, DefectOrFeatureName, workareaType, effectiveUserID, VerboseFlag
WorkAreaModify	0	release, WorkAreaName, newtarget, StandardFields, configFields, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	WorkAreaName, release, oldtarget, newtarget, DefectOrFeatureName, workareaType, effectiveUserID, VerboseFlag
	2	WorkAreaName, release, oldtarget, newtarget, DefectOrFeatureName, workareaType, effectiveUserID, VerboseFlag
WorkAreaReconci	0	release, WorkAreaName, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	release, WorkAreaName, effectiveUserID, VerboseFlag
	2	release, WorkAreaName, effectiveUserID, VerboseFlag
WorkAreaRefresh	0	release, WorkAreaName, source, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	release, WorkAreaName, source, workareaType, effectiveUserID, VerboseFlag

<b>TeamConnection action</b>	<b>Exit ID</b>	<b>Parameters passed to the user exit program (see page 212 for definitions)</b>
	2	release, WorkAreaName, source, workareaType, effectiveUserID, VerboseFlag
WorkAreaTest	0	release, WorkAreaName, forceFlag, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	WorkAreaName, release, DefectOrFeatureName, workareaType, effectiveUserID, VerboseFlag
	2	WorkAreaName, release, DefectOrFeatureName, workareaType, effectiveUserID, VerboseFlag
WorkAreaUndo	0	release, WorkAreaName, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	release, WorkAreaName, target, workareaType, effectiveUserID, VerboseFlag
	2	release, WorkAreaName, target, workareaType, effectiveUserID, VerboseFlag
WorkAreaView	0	release, WorkAreaName, reporttype, MessageBuffer, effectiveUserID, TeamcUserID, VerboseFlag
	1	release, WorkAreaName, reporttype, workareaType, effectiveUserID, VerboseFlag
	2	release, WorkAreaName, reporttype, workareaType, effectiveUserID, VerboseFlag

## User exit parameter definitions

The following list provides definitions for most of the parameters passed to user exit programs. Parameters are listed in alphabetical order. Parameter names are in lowercase, except where they are the name of a field in a TeamConnection database table. For more information on these and other parameters, refer to the *Commands Reference*.

### **abstract**

Defect or feature abstract.

### **alternateversion**

Specifies the name of a version of a driver, release, or work area where the conflicting version of a part is visible.

### **answer**

Specifies the reason for an action taken on a defect.

### **ApproverName**

Approver's TeamConnection user ID.

**area** Department or area in which the user works.

### **authority**

Specifies a user's authority group.

### **autoprun**

Whether or not to automatically prune work areas that have not been integrated with the release. Valid values are yes and no.

**buildername**

The name of the builder used to create an output part.

**buildmode**

The mode in which the build runs. The following values are valid:

- 1 force
- 2 normal
- 3 unconditional
- 4 report

**buildparameters**

Specifies the parameters passed to the build script.

**cancelFlag**

This flag is used with PartBuild actions to cancel a build request.

**clienthostname**

The name of the system where the client command originated.

**clientportname**

The port number used for the build pool.

**committedFlag**

This flag is used with ReleaseExtract and ReleaseLink actions to specify whether the user wants the last committed (as opposed to the current) versions of parts in the release. A value of 0 means to use the current version; 1 means to use the last committed version.

**commonFlag**

Indicates whether the part is common with other releases or not. A value of 0 indicates no, 1 indicates yes.

**commonreleases**

For a common part, this parameter specifies the other releases the part is common with (on the partCheckIn action). Release names are separated by blanks.

**component**

Specifies the name of a component.

**componentprocess**

Specifies the process to be used for a component in CompCreate actions.

**condition**

This parameter is used with the value parameter to determine if a build event was successful.

**configFields**

This parameter has the format:

```
attribute name    content
attribute name    content
```

⋮  
null string

For exit ID 0, the attribute name can appear in abbreviated form, as it is not processed by TeamConnection.

**creatChangeFlag**

This flag is used by PartTouch to specify the write permissions of the part: 0200 permits write by the owner; 0000 does not allow write. This parameter is defined numerically, in octal notation. The fMode code is constructed by combing the logical OR of the following values:

- 4000** setuid
- 2000** setgid
- 0400** Permits read by owner
- 0200** Permits write by owner
- 0100** Permits execute or search by owner
- 0040** Permits read by group
- 0020** Permits write by group
- 0010** Permits execute or search by group
- 0004** Permits read by all others
- 0002** Permits write by all others
- 0001** Permits execute or search by all others

For example, 0755 would permit read, write, and execute for the owner and read and execute for all others.

**crLfFlag**

This flag is used by DriverExtract and ReleaseExtract to handle crlf conversions when extracting Intel-based files to a UNIX-based platform.

**date** Enables you to extract only files from a release that are older than the specified date.

**dateoflastupdate**

Specifies the date in modify actions.

**defectname**

Indicates the name of the defect.

**DefectOrFeatureName**

Indicates the name of the defect or feature for approval record, fix record, test record, driver member, or work area actions.

**description**

Specifies a description of an object.

**detailfilename**

Specifies the file in which all build messages for a part are collected.

**developmentmode**

Valid values are serial and concurrent.

**displaytype**

This parameter is used on all view actions. The type of view format requested, where:

0	stanza
1	raw
2	table
3	long
4	process

**dmask** Specifies the read, write, and execute directory permissions for the extracted parts in octal notation.

**driver, drivename**

Specifies the name of the driver for defect, driver, driver member, and work area actions.

**driverstate**

Values can be working, integrate, commit, or complete.

**drivertype**

Specified by the user when a driver is created, for example, development, production, or prototype.

**effectiveUserID**

The TeamConnection user ID that initiated the transaction. This is the value of the TC\_BECOME environment variable or the -become attribute flag. In OS/2, Windows 3.1, and Windows 95 environments, if this variable is not set and the -become attribute is not specified, it is the value of the TC\_USER environment variable.

**environment, environmentname**

Specifies the environment in which the testing is to be done if the test subprocess is included in the release process. (The tester/environment name combination becomes an entry on the environment list for the release.)

**ExtractType**

Indicates whether this is a full or delta driver extract. 0 indicates delta; 1 indicates full.

**featurename**

Specifies the name of the feature for feature actions.

**filenameonclient**

The name of the source file from which a TeamConnection part is created using the PartAdd or PartCheckin action.

**filetype**

Specifies one of the following file types with the PartAdd action:

- 0** none
- 1** text
- 2** binary

Part type for the other part actions is text for text parts, and binary for binary parts.

**fmask** Specifies the read, write, and execute file permissions for the extracted parts in octal notation. Refer to the *Commands Reference* for details.

**fMode** Specifies the write permissions of the part: 0200 permits write by the owner; 0000 does not allow write. This parameter is defined numerically, in octal notation. The fMode code is constructed by combing the logical OR of the following values:

- 4000** setuid
- 2000** setgid
- 0400** Permits read by owner
- 0200** Permits write by owner
- 0100** Permits execute or search by owner
- 0040** Permits read by group
- 0020** Permits write by group
- 0010** Permits execute or search by group
- 0004** Permits read by all others
- 0002** Permits write by all others
- 0001** Permits execute or search by all others

For example, 0755 would permit read, write, and execute for the owner and read and execute for all others.

**forceFlag**

Indicates whether the force option was chosen on part actions (0 indicates no and 1 indicates yes). The force option is used to force a break between common parts when using PartLock, PartCheckOut, PartCheckIn, PartDelete, PartRecreate, PartRename, and PartUndo actions.

**FromRelease**

Specifies the name of the release to be linked from in ReleaseLink actions.

**fromversionname**

Specifies the version of the release to be linked from in ReleaseLink actions.

- fromworkareaname**  
Specifies the name of the work area to be linked from in ReleaseLink actions.
- gid**  
Specifies ownership of extracted parts by identifying the internal number that uniquely identifies the group to the system.
- interestgroupname**  
Specifies the name of an interest group in a NotifyCreate or NotifyDelete action.
- lastdropdate**  
Specifies the date on which a release to be recreated using the ReleaseRecreate action was deleted.
- login**  
The system login ID for a user. In single-user environments, such as OS/2, Windows 3.1 and Windows 95, this parameter is the TC\_USER environment variable.
- login@hostname**  
Specifies a host list entry in HostCreate and HostDelete actions.
- longFlag**  
Indicates whether the -long flag is specified or not specified. A value of 0 indicates not specified; 1 indicates specified. The -long flag is available on some of the view actions and is used to display more detailed information of the object being viewed. The -long flag on the driverCheck action displays details about prerequisites and corequisites.
- name**  
Specifies the name of a builder in Builder actions. In the CompModify action, this parameter specifies the current component name.
- newarea**  
Specifies a new area or department in which a user works.
- newcomponent**  
Specifies a new name for a component.
- newcomponentprocess**  
Specifies a new process to be used for a component.
- newdefectname**  
Specifies a new name for a defect in a DefectModify action.
- newdescription**  
Specifies a new description for an object.
- newdrivertype**  
Specifies a new name for a driver in a DriverModify action.
- newdrivertype**  
Specifies a new type for the driver in DriverModify actions. Valid types include development, production, or prototype.
- newfeaturename**  
Specifies a new name for a feature in a FeatureModify action.

**newfMode**

Specifies a new write permission for a PartModify action. See **fmode** for a list of values.

**newlogin**

Specifies a new login ID for a user.

**NewOwner**

Specifies the new owner of an object in actions that create, modify, or assign owners to objects.

**newrelease**

Specifies a new release name for ReleaseModify actions.

**newreleaseprocess**

Specifies the release process to be used for ReleaseCreate or ReleaseModify actions.

**newsendmailaddress**

Specifies a new email address for a user's notification messages.

**newsuperuserprivilegeflag**

Specifies the user's superuser status for UserModify actions. Specify 0 to deny superuser status or 1 to grant superuser status.

**newtarget**

Specifies a new target for work areas.

**newtestersname**

Specifies the full name of a new tester in an EnvModify action.

**newtype**

Specifies a new driver type for DriverModify actions. Valid types include development, production, or prototype.

**newusersfullname**

Specifies the new full name for a user in UserModify actions.

**newuserssendmailaddress**

Specifies the new mail address for a user in UserModify actions.

**newworkareaname**

Specifies the work area to be linked to in ReleaseLink actions.

**node** Specifies a remote host on which to place the extracted part tree.

**nokeysFlag**

For extract actions, indicates whether you want to substitute assigned values in place of keywords imbedded in the extracted parts. 0 means not to substitute assigned values; 1 means to substitute assigned values.

**numberofworkareas**

Specifies the number of work areas to be deleted in a MemberDelete action.

**nuPartPathName**

Specifies the new path name for PartRename actions.

- oldcomponentprocess**  
Specifies the process of a component to be modified.
- olddescription**  
Specifies the description of an object to be modified.
- olddropDate**  
Specifies the date on which a component, part, or user to be recreated using the CompRecreate, PartRecreate, or UserRecreate action was deleted.
- oldfMode**  
Specifies the old write permission for a PartModify action. See **fmode** for a list of values.
- OldOwner**  
Specifies the old owner of an object in actions that modify or assign owners to objects.
- oldreleaseprocess**  
Specifies the old release process to be changed by a ReleaseModify action.
- oldsendmailaddress**  
Specifies an email address to be changed for a user's notification messages.
- oldsuperuserprivilegeflag**  
Specifies the user's superuser status to be changed by a UserModify action. Specify 0 if the user currently does not have superuser status or 1 if he or she currently does have superuser status.
- oldtarget**  
Specifies a target to be changed for work areas.
- oldtype**  
Specifies the driver type to be changed by a DriverModify action. Valid types include development, production, or prototype.
- oldusersfullname**  
Specifies the full name for the user to be changed by a UserModify action.
- originaldefectname**  
The name of a defect for which the current defect is a duplicate.
- originalfeaturename**  
The name of a feature for which the current feature is a duplicate.
- originator**  
The TeamConnection user ID of the user who opens the defect or feature.
- outputversions**  
Specifies the number of versions of build output parts to be retained in ReleaseCreate or ReleaseModify actions.
- owner** Specifies the component owner in CompCreate actions.
- parent** Specifies the parent of the part to generate a report on using the Report -view PartView action.

**parentcomponent**

Specifies the parent component for Component actions.

**parentname**

Specifies the parent of a part in a build tree in PartAdd, PartConnect, and PartDisconnect actions.

**parenttype**

Specifies the part type of the parent of a part in a build tree in PartAdd, PartConnect, and PartDisconnect actions. In Report actions, this parameter specifies the part type of the parent of the part to generate a report on using the Report -view PartView action.

**parsername**

Specifies the name of the parser used to create an output part.

**parsercommand**

Specifies the command file you want to associate with the parser. This can be an .exe, a .com, a .cmd, or a .bat file. The executable file needs to be in the execution path of the TeamConnection family server.

**partpathName**

Specifies the path name of a part in Part actions.

**parttype**

Specifies the type of a part, such as TcPart or vgdata.

**pathName**

Specifies the path name of parts in Collision actions.

**paths** Specifies a concatenated set of paths that define where the parser looks for parts when processing the set of dependencies returned from the command file. These dependencies come in two types:

- A dependency in which the file is stored in the TeamConnection database. For example, hello.c includes hello.h, and both files are stored in the TeamConnection database. During a build, these dependencies must be extracted to a path accessible by the build processor.
- A dependency on a file that is not stored in the TeamConnection database. An example of such a dependency is stdio.h, which is typically stored in a compiler's include path and not in the TeamConnection database.

**poolname**

Specifies the build pool used to build a part.

**prefix** Defect or feature prefix.

**primeworkareaname**

Prime corequisite work area name.

**process**

Specifies the component process in CompModify actions.

**processoroptions**

Parameters specified for passing to a builder upon builder -create.

**reference**

Defect or feature reference.

**relationtoparent**

How a part is related to its parent in the build tree, where:

- 1 input
- 2 output
- 3 dependent

**release**

Name of the release.

**releasedatabasename**

Name of a separate database for the part data (the contents of each part) in a release.

**releaseowner**

Specifies the owner of a release.

**remarks**

For defect or feature actions, this is defect remarks or feature remarks. For part actions, this is part remarks added when a new version is created.

**reportcriteria**

The criteria entered as the -where clause for a Report action.

**reporttype**

The type of report format requested, where:

- 0 stanza
- 1 raw
- 2 table
- 3 long
- 4 process
- 5 html

User exit messages are not displayed if the -raw format is selected.

**retainlockFlag**

Specifies that a part is to remain locked after is it checked in.

**root**

This is the specified directory on the designated host where the extracted part tree is to be placed.

**script**

Specifies the name of the build script.

**secondworkareaname**

Second corequisite work area name.

**sendmailaddress**

The e-mail address to which a user's notification messages are sent.

- severity**  
Defect severity driver.
- sizetext**  
The sizing information for a defect or feature.
- sizetype**  
Specifies whether the sizing record is associated with a defect or a feature.
- source** Specifies the name of a work area with which another work area is refreshed.
- sourcerelease**  
Specifies the original release of a part to be linked using the PartLink action.
- sourceversion**  
Specifies the original version of a part to be linked using the PartLink action.
- sourceworkareaname**  
Specifies the original work area of a part to be linked using the PartLink action.
- StandardFields**  
Contains any fields abbreviated by users, requiring interpretation and verification on the TeamConnection server. Actions with configurable fields allow for the ambiguity in parameter names that requires intervention by the server.
- state** Specifies the state of parts in Collision actions.
- superuserprivilegeflag**  
A value of yes indicates on; a value of no indicates off.
- target** Specifies the value of the work area target field for a WorkareaUndo action.
- targetenvironment**  
Specifies the environment for which build output is generated.
- targetrelease**  
Specifies the new release of a part to be linked using the PartLink action.
- targetworkareaname**  
Specifies the new work area of a part to be linked using the PartLink action.
- TeamcUserID**  
The user's TeamConnection user ID on the client workstation. In AIX, HP-UX, Solaris, and Windows NT environments, this is the login ID. In OS/2, Windows 3.1, and Windows 95 environments, this is the value of the TC\_USER environment variable.
- temporaryfilename**  
Indicates if the -temporary flag is used on a Part -modify command.
- temporaryfileonserver**  
For some part actions, the contents of the file on the client are copied to a temporary file on the server. This parameter is the name for the temporary file on the server.

- temporaryFlag**  
Indicates the part is a temporary part on PartAdd and PartModify actions.
- testersname, TesterName**  
Specifies the full name of the person responsible for testing an object.
- timeout**  
Specifies the amount of time that the build processor waits for a build script to complete before assuming a failure has occurred. The default is 1440 minutes (24 hours).
- translation**  
Specifies how the part is related to the translation process. For example, a part might be translated into another language, used while translating other parts, or completely unrelated to translation.
- translation state**  
Specifies the translation state of the part. Valid values are notReady and ready.
- transmitFlag**  
Indicates whether the builder part is to be copied from the client to the server or not. Specify 0 or 1.
- type** Specifies whether the sizing record is associated with a defect or a feature on Test actions.
- typename**  
Specifies the type of parts being handled by Collision actions.
- uid** Specifies ownership of extracted parts by identifying the internal number that uniquely identifies the user to the system.
- usersfullname**  
Specifies the full name of a user.
- value** This parameter is used with the condition parameter to determine if a build event was successful.
- VerboseFlag**  
Specifies that you want to see a confirmation message after you issue this action. 0 indicates off; 1 indicates on. The user exit program can use this flag to include confirmation or status messages only when the -verbose flag is on.
- versionname**  
Part version name.
- viewname**  
The name of the view (for example, partView) that is being reported on.
- WorkAreaName**  
Specifies the name of a work area.
- workareaOwner**  
Specifies the owner of a work area.

**workareastate**

The value can be approve, fix, integrate, commit, test, or complete.

**workareatarget**

Specifies the target field used when creating or modifying a workarea in WorkAreaFreeze actions.

**workareaType**

The value can be defect or feature.

## Appendix D. Environment Variables

You can set environment variables to describe the TeamConnection environment in which you are working. You are not required to set your TC\_FAMILY environment variable for the TeamConnection client command line interface. However, if the TC\_FAMILY environment variable is not set, the -family must be specified for every client command. See "Setting environment variables" on page 231 for more information about setting environment variables.

The names of the TeamConnection environment variables, the purpose they serve, the equivalent TeamConnection flag, the equivalent Settings notebook field, and the TeamConnection component that uses the environment variable are listed in the following table.

You can override the value you set for an environment variable by using the corresponding flag in a TeamConnection command. When an environment variable has a Settings notebook equivalent, TeamConnection uses the two as follows:

- The environment variable controls the command line interface.
- The Settings notebook controls the graphical user interface.

If there is no Settings notebook equivalent for the environment variable, then the environment variable takes effect regardless of the interface you are using.

To see some of your client settings, you can issue the following command from a command prompt:

```
teamc report -testServer
```

This command returns information like the following:

```
Connect to Family Name:      ptest
Server TCP/IP Name:         amachine.company.com
Server IP Address:          9.1.23.45
Server TCP/IP Port Number:  9999
```

```
Server Specific Information -----
Product Version:            3.0.0
Operating System:           AIX
Message catalog language:   English
Server Mode:                non-maintenance
Authentication Level:       HOST_ONLY
```

Table 22. TeamConnection environment variables

Environment variable	Purpose	Flag	Setting	Used by
LANG	Specifies the language-specific message catalog.			Client, family server

Table 22. TeamConnection environment variables (continued)

Environment variable	Purpose	Flag	Setting	Used by
NLSPATH	Specifies the search path for locating message files.		NLS path	Client, family server
PATH	Specifies where tcadmin is to search for the family create utilities.			Client, build server, family server
TC_BACKUP	Controls whether or not the following commands create backup files. If this environment variable is set to off or OFF, the commands do not create backup files. <ul style="list-style-type: none"> <li>• builder -extract</li> <li>• part -checkout</li> <li>• part -extract</li> <li>• part -merge</li> <li>• part -reconcile</li> </ul>			Family server
TC_BECOME	Identifies the user ID you want to issue TeamConnection commands from, if the user ID differs from your login. You assume the access authority of the user ID you specify.	-become	Become user	Client, build server (except mvs)
TC_BUILDENVIRONMENT	Specifies the build environment name, such as OS/2 or MVS. The value you specify here can be anything you like, but it must exactly match the environment specified for a builder in order for the builder to use this build agent. This value is case-sensitive.	-e		Build server
TC_BUILDMINWAIT	Minimum amount of time to wait (in seconds) between queries for new jobs. Default setting is 5, minimum setting is 3.			Build server
TC_BUILDMAXWAIT	Maximum amount of time to wait (in seconds) between queries for new jobs. Default setting is 15, maximum setting is 300.			Build server

Table 22. TeamConnection environment variables (continued)

Environment variable	Purpose	Flag	Setting	Used by
TC_BUILD_OPTS	<p>Specifies build options for sending build log file messages to the screen, and setting the logging level. If you do not specify any of these options, then the build server writes build messages to the build log file (teamcbld.log), and writes a minimum level of messages to the log file. Some possible values are:</p> <ul style="list-style-type: none"> <li>• TOSCREEN (-s) sends the <b>teamcbld.log</b> file to the screen in addition to sending it to a file.</li> <li>• USEENVFILE (-n) <ul style="list-style-type: none"> <li>– writes the changed environment variables to a file called <b>tcbldev.lst</b> instead of setting them in program's environment. The format of the file is variable=value.</li> <li>– writes the list of input files to a file called <b>tcblidin.lst</b>. One file per line, format is pathName type.</li> <li>– writes the list of output files to a file called <b>tcblidout.lst</b>. One file per line, format is pathName type.</li> </ul> </li> </ul>	-s, -n		Build server
TC_BUILDPOOL	Specifies the build pool name.	-p	Pool	Build server
TC_BUILD_RSSBUILDS_FILE	Specifies the name of startup files to be used to provide information about build servers to the build process.			Build server
TC_CASESENSE	Changes the case of the arguments in commands, not in queries.		Case	Client

Table 22. TeamConnection environment variables (continued)

Environment variable	Purpose	Flag	Setting	Used by
TC_CATALOG	Specifies a specific file for the TeamConnection message catalog. Sometimes, depending upon the operating system environment, the catalog open command will only look in a particular directory for the catalog. If the host is running multiple versions of TeamConnection, this variable may be used. To set this environment variable, specify the file path name of the message catalog as in the following example:  TC_CATALOG= "/family/msgcat/teamc.cat"			Family server, oe build server
TC_COMPONENT	Specifies the default component.	-component	Component	Client, make import tool
TC_DBPATH	Specifies the database directory path. Family specific database files reside here.			Family server
TC_FAMILY	Identifies the TeamConnection family you work with.	-family	Family	Build server, client, family server, make import tool
TC_MAKEIMPORTRULES	Specifies the name of the rules file that TeamConnection uses when importing the makefile data into TeamConnection. If you set this environment variable, then you do not have to use the /u option with the fhomigmk command. Specify the full path name of the rules file. If neither this environment variable nor the /u option is used, TeamConnection uses default rules.			Make import tool

Table 22. TeamConnection environment variables (continued)

Environment variable	Purpose	Flag	Setting	Used by
TC_MAKEIMPORTTOP	Strips off the leading part of the directory name when importing parts into TeamConnection. For example, you have parts with the following directory structure: g:\octo\src\incl. To create these parts without the g:\octo structure, you can set TC_MAKEIMPORTTOP=g:\octo before you invoke the make import tool. The parts created in TeamConnection have the directory structure of src\incl.			Make import tool
TC_MAKEIMPORTVERBOSE	Causes the -verbose flag to be added to part commands created by fhomigmk.			Make import tool
TC_MIGRATERULES	Specifies the name of a file containing the rules to be applied for migration of makefiles if the name is not supplied on the fhomigmk command line as a parameter.			Client
TC_MODPERM	Controls whether or not the read-only attribute is set after a part is created, checked in or unlocked in TeamConnection. To cause the read-only attribute to be set, specify TC_MODPERM=ON. To prevent the read-only attribute from being set, specify TC_MODPERM=OFF. The default is TC_MODPERM=ON.			Client
TC_NOTIFY_DAEMON	An alternate way of starting notifyd with the teamcd command. If you set this environment variable, then you do not have to use the -n option with the teamcd command. Specify the full path name of the mail exit to use with notifyd.			Family server
TC_RELEASE	Specifies a release.	-release	Release	Client, make import tool
TC_TOP	Specifies the source directory.	-top	Top	Client

Table 22. TeamConnection environment variables (continued)

Environment variable	Purpose	Flag	Setting	Used by
TC_TRACE	Specifies the variable that lets the user designate which parts should be traced. You should modify this only when directed to do so by an IBM service person. Otherwise it is set to null. To trace all parts, specify TC_TRACE=*			Client, family server, build server
TC_TRACEFILE	Specifies the output (part path and name) of the trace that the user designates using TC_TRACE. The default trace file name is tctrace. For the MVS build server, the default trace file is stdout.			Client, family server, build server
TC_TRACESIZE	Specifies the maximum size of the trace file in bytes. If the maximum is reached, wrapping occurs. The default is one million bytes.			Client, family server, build server
TC_USER	Specifies the user login ID for single-user environments OS/2 and Windows 95 (if not using the login facility). This environment variable is not used in multiuser environments AIX, HP-UX, Solaris, MVS, MVS/OE, and Windows NT. If a user is using the Windows 95 login facility, this environment variable is not used.		User ID	Client, build server
TC_WORKAREA	Specifies the default work area name.	-workarea	Work area	Client, make import tool
TC_WWWPATH	Specifies the path for the HTML helps and image files for Web client.			Client, family server
TC_WWWDISABLED	Disables the Web client.			Family server

The following environment variables are dynamically set by the teamcblid command processing before the build script is invoked:

Table 23. TeamConnection dynamically set build environment variables

Environment variable	Purpose	Flag	Setting	Used by
TC_BUILD_USER	Login of user who initiated the part -build command.			Build server

Table 23. TeamConnection dynamically set build environment variables (continued)

Environment variable	Purpose	Flag	Setting	Used by
TC_INPUT	List of input files (separated by spaces).			Build server
TC_INPUTTYPE	List of input file types (such as TCPart).			Build server
TC_OUTPUT	List of output files.			Build server
TC_OUTPUTTYPE	List of output file types.			Build server
TC_LOCATION	Directory where build script is invoked.			Build server (except MVS build server)

---

## Setting environment variables

For methods of setting your environment variables, refer to your operating system documentation. For example, you can use the following command to set the TC\_FAMILY environment variable:

- OS/2 - SET TC\_FAMILY=familyName@hostname@portnumber
- UNIX - export TC\_FAMILY=familyName@hostName@portNumber



---

## Appendix E. TeamConnection NLS and DBCS considerations

This appendix describes how to use IBM VisualAge TeamConnection Enterprise Server Version 3 in situations that require National Language Support (NLS) and Double-Byte Character Sets (DBCS) in all the supported platforms.

The following topics are addressed in this appendix:

- The overview of the NLS and DBCS support provided by IBM VisualAge TeamConnection Enterprise Server Version 3, such as usage of locale XPG/4 I18N programming model, and the supported locales and platforms.  
The locale support is already provided by the UNIX operating systems. However, in OS/2 and Windows 32-bit, the locale support is not provided by these operating system, instead it is provided by TeamConnection during the installation process.
- The main characteristics and limitations related to NLS/DBCS, such as the interoperability between clients and server, and special cases for the manipulation of data by TeamConnection.
- The main issues related to installation, administration and runtime, such as directory structure of the installed code, and why you should not change the code page of an existing TeamConnection family.

More information on NLS and DBCS considerations for TeamConnection may be available in technical reports on the IBM VisualAge TeamConnection Enterprise Server Library home page. To access this home page, select Library from the IBM VisualAge TeamConnection home page at URL <http://www.software.ibm.com/ad/teamcon>.

---

### Overview of TeamConnection NLS and DBCS support

#### Language and culture sensitive information in TeamConnection

VisualAge TeamConnection supports the I18N (Internationalization) locale model proposed by XPG/4 (X/Open Portability Guide, issue 4) in which the language and culture sensitive information are not hard coded in the executable files; instead, they are provided as system resources by means of a "locale" that the user can specify at run-time.

One of the components of a locale is the code page in which the characters will be handled. For example, in AIX 4, the default locale is "en\_US" which is for the English language used in the USA and the associated code page is ISO8859-1, which is different than the default code page used for English in OS/2 (code page IBM-850) but the ISO8859-1 code page is similar to the one used for English in Windows (code page MS-1252 Latin 1).

The locale model for XPG/4 establishes several environment variables that can be used for controlling the culture sensitive information. Table 24 on page 234 describes these environment variables, their function and how TeamConnection deals with them.

Table 24. How TeamConnection handles the locale environment variables

Locale environment variable	Function	How TeamConnection uses it
LANG	Specifies the installation default locale.	It is an identifier that is used to resolve the complete path where the message catalogs and other language-sensitive files are located in the system. The specification for TeamConnection is shown in note (3).
NLSPATH	Specifies the full path for the message catalog file.	It is the specification of the full path where the message catalog file is located. The specification for TeamConnection is shown in note (4).
LC_ALL	Overrides the value of other LC_* environment variables.	It is not explicitly exploited by TeamConnection.
LC_COLLATE	Determines the character-collation or string-collation rules.	It is ignored by TeamConnection. (See Note 1).
LC_CTYPE	Determines the character handling rules governing the interpretation of sequences of bytes of text data characters and classification of characters.	It is ignored by TeamConnection. However, if LANG is not defined, then the value of LC_CTYPE is used by the UNIX operating system. The default value is the C locale.
LC_MESSAGES	Determines the rules governing affirmative and negative responses, and the locale for messages and menus.	It is ignored by TeamConnection.
LC_MONETARY	Determine the rules governing monetary-related formatting.	It is ignored by TeamConnection, because it does not handle this kind of information.
LC_NUMERIC	Determine the rules governing non-monetary numeric formatting.	It is ignored by TeamConnection, because it handles only integer numbers with no separation for thousands.
LC_TIME	Determine the rules governing date and time formatting.	It is ignored by TeamConnection. There is no special processing for the date and time information. (See Note 2).

**Notes:**

1. TeamConnection itself does not perform any sorting of data. Instead, the sorting is performed by the database.
2. The date and time in TeamConnection is represented as YYYY/mm/dd hh:mm:ss, where YYYY is the year, mm is the month, dd is the day, hh is the hour, mm is the minute, and ss is the second. Because four digits are used to represent a year, TeamConnection is compliant with the Year 2000 specifications.
3. The specification for LANG for TeamConnection is (Korn shell):  

```
export LANG=en_US
```
4. The specification for NLSPATH for TeamConnection is (Korn shell):  

```
export NLSPATH=/usr/teamc/nls/msg/%L/%N
```

Specify %L and %N as shown in uppercase. The placeholder %L is for all practical purposes a synonym for the value of the variable LANG, and the placeholder %N is used by the TeamConnection code to specify during run-time the name of the file that has the messages to be displayed (message catalog).

## Supported locales (languages and code pages)

VisualAge TeamConnection Version 3 provides support for the following locales (which include the translated message catalogs):

- Single-Byte Character Set (SBCS) locales; see "Supported Single-Byte Character Set (SBCS) locales" on page 236.
- Double-Byte Character Set (DBCS) locales; see "Supported Double-Byte Character Set (DBCS) locales" on page 236.

The majority of the locale names follow a format similar to en\_US, where the first 2 characters represent the abbreviation for language names defined in ISO 639 (such as "en" for English) and the last 2 characters represent the abbreviations for country names defined in ISO 3166 (such as "US" for the United States of America).

In some cases, the locale names may have a suffix which represents a special identification, such as the HP-UX locale "zh\_TW.big5".

## Locales supported by DB2 Universal Database (UDB) Version 5

TeamConnection uses DB2 Universal Database (UDB) Version 5 which is enabled to handle DBCS, regardless of the locale. The installation of TeamConnection also includes the installation of DB2 UDB V5 and its corresponding locales. Most of the information in the following tables was obtained from Table 101 "Supported Languages and Code Sets" from Appendix M, "National Language Support," in the *DB2 UDB V5 Administration Guide*.

## Supported Single-Byte Character Set (SBCS) locales

VisualAge TeamConnection supports the following Single-Byte Character Set (SBCS) locales. It is important to emphasize that the locales "En\_US" (code page 850) and "en\_US" (code page ISO8859-1) are different. For example, if you use a TeamConnection family in AIX with the ISO locale en\_US (code page ISO8859-1), and a TeamConnection client in OS/2 with the En\_US locale (code page IBM-850), then you will see "code page incompatibility" problems (in which some characters will NOT be shown or will not look OK).

**United States of America:** The relevant codes for United States of America are:

**Country Name**

United States of America

**Country Codes**

1, US

**Language Codes**

enu, en

The supported code pages and locales for United States of America are shown below.

*Table 25. United States of America - supported code pages and locales*

Code Page	Code Set	Locale	Operating System	Notes
819	ISO8859-1	en_US	AIX	
850	IBM-850	En_US	AIX	
819	iso8859-1	en_US.iso88591	HP-UX	
1051	roman8	en_US.roman8	HP-UX	
819	ISO8859-1	en_US	Solaris	
437	IBM-437	En_US	OS/2	
850	IBM-850	En_US	OS/2	
1252	1252	en_US	Win32	(1)
37	IBM-037	-	OS/390	

**Notes:**

1. The Microsoft Latin code page 1252 is very similar to ISO8859-1 (Latin 1). This code page is only used in the TeamConnection GUI tools. For details on the conversion of code pages in Windows, see "No conversion of code points when exchanging data" on page 240.

## Supported Double-Byte Character Set (DBCS) locales

VisualAge TeamConnection supports the following Double-Byte Character Set (DBCS) locales. It is important to emphasize that the locales "Ja\_JP" and "ja\_JP", and "Zh\_TW" and "zh\_TW" are different. For example, if you use a TeamConnection family in AIX with the EUC locale ja\_JP (code page IBM-eucJP), and a TeamConnection client

in OS/2 with the Ja\_JP locale (code page IBM-932) then you will see "code page incompatibility" problems (in which some characters will NOT be shown or will not look OK).

**Japan:** The relevant codes for Japan are:

**Country Name**  
Japan

**Country Codes**  
81, JP

**Language Codes**  
jap, ja

The supported code pages and locales for Japan are shown below.

*Table 26. Japan - supported code pages and locales*

Code Page	Code Set	Locale	Operating System	Notes
954	IBM-eucJP	ja_JP	AIX	
932	IBM-932	Ja_JP	AIX	
954	eucJP	ja_JP.eucJP	HP-UX	
5039	SJIS	ja_JP.SJIS	HP-UX	
954	eucJP	ja	Solaris	
932	IBM-932	Ja_JP	OS/2	(1)
942	IBM-942	Ja_JP	OS/2	(1)
943	IBM-943	Ja_JP	OS/2	(1)
943	IBM-943	Ja_JP	Win32	(1)
930	IBM-930	-	OS/390	
939	IBM-939	-	OS/390	
5026	IBM-5026	-	OS/390	
5035	IBM-5035	-	OS/390	

**Notes:**

1. The Japanese IBM-932, IBM-942 and IBM-943 code pages have very small differences between them, but generally speaking, they are compatible with each other.

**South Korea:** The relevant codes for South Korea are:

**Country Name**  
South Korea

**Country Codes**  
82, KR

## Language Codes

kor, ko

The supported code pages and locales for South Korea are shown below.

Table 27. South Korea - supported code pages and locales

Code Page	Code Set	Locale	Operating System	Notes
970	IBM-eucKR	ko_KR	AIX	
970	eucKR	ko_KR.eucKR	HP-UX	
970	eucKR	ko_KR	Solaris	
949	IBM-949	ko_KR	OS/2	(1)
1363	1363	ko_KR	Win32	(1)
933	IBM-933	-	OS/390	

### Notes:

1. The Korean code page for Windows NT/95 is called UHC (Unified Hangeul Code). The 1363 code page extends IBM-949 by adding missing Hangeul characters with no change of assignments in code points for IBM-949.

**People's Republic of China (PRC):** The relevant codes for People's Republic of China (PRC) are:

### Country Name

People's Republic of China (PRC)

### Country Codes

86, CN

### Language Codes

chs (Simplified), zh

The supported code pages and locales for People's Republic of China (PRC) are shown below.

Table 28. People's Republic of China (PRC) - supported code pages and locales

Code Page	Code Set	Locale	Operating System	Notes
1383	IBM-eucCN	zh_CN	AIX	
1386	GBK	Zh_CN.GBK	AIX	
1383	eucCN	zh_CN.hp15CN	HP-UX	
1383	eucCN	zh	Solaris	
1381	IBM-1381	Zh_CN	Win32	
1386	GBK	Zh_CN	Win32	(1)
935	IBM-935	-	OS/390	
1381	IBM-1381	Zh_CN	OS/2	

Table 28. People's Republic of China (PRC) - supported code pages and locales (continued)

Code Page	Code Set	Locale	Operating System	Notes
1386	GBK	Zh_CN	OS/2	(1)

**Notes:**

1. The code page for Simplified Chinese for Windows NT/95 is called GBK (Guo Biao Kuo). The IBM-1386 code page is equivalent to Microsoft 936. The IBM-1386 code page extends IBM-1381 by adding missing Unicode characters with no change of assignments in code points for IBM-1381.
2. The EUC code page for Traditional Chinese (IBM-eucTW) for AIX 4.1 has been enhanced with respect to AIX 3.2, but it keeps the same locale name (zh\_TW). This means that if the user in AIX 4.1 exploits the new characters in the enhanced locale version, there could be compatibility problems when the user uses the old locale version.

**Taiwan, Republic of China (ROC):** The relevant codes for Taiwan, Republic of China (ROC) are:

**Country Name**

Taiwan, Republic of China (ROC)

**Country Codes**

886, TW

**Language Codes**

cht (Traditional), zh

The supported code pages and locales for Taiwan, Republic of China (ROC) are shown below.

Table 29. Taiwan, Republic of China (ROC) - supported code pages and locales

Code Page	Code Set	Locale	Operating System	Notes
938	IBM-938	-	OS/2	old?
948	IBM-948	-	OS/2	old?
950	big5	Zh_TW	OS/2	(1)
950	big5	Zh_TW	AIX	(1)
964	IBM-eucTW	zh_TW	AIX	(2)
950	big5	zh_TW.big5	HP-UX	(1)
964	eucTW	zh_TW.eucTW	HP-UX	
950	big5	big5	Solaris	(1,3)
964	eucTW	zh_TW	Solaris	
950	big5	Zh_TW	Win32	(1)
937	IBM-937	-	OS/390	

**Notes:**

1. The PC code page for Traditional Chinese is called Big-5.
2. The EUC code page for Traditional Chinese (IBM-eucTW) for AIX 4.1 has been enhanced with respect to AIX 3.2, but it keeps the same locale name (zh\_TW). This means that if the user in AIX 4.1 exploits the new characters in the enhanced locale version, there could be compatibility problems when the user uses the old locale version.
3. The Solaris code page 950 (Taiwan) does not support certain characters from the IBM-850 code page.

---

## Characteristics and limitations of NLS and DBCS support

### No conversion of code points when exchanging data

The TeamConnection clients and servers do not alter the code points of the data. This means that the data is NOT converted from one code page to another when entered by the user, when stored in the database used by the family or when exchanged between a client and the server. The information about the code page in which the data was entered is not stored with TeamConnection objects; furthermore, there is no exchange of information between the client and the server to indicate which code page is being used by each of them.

### No impact if using English characters

Because most code pages have the same code points for the first 128 characters, which includes all the characters used in the English alphabet, then in practice there is no effect in using different code pages between clients and servers, if using only English characters.

As an example, the default multilingual code page for OS/2 is IBM-850, for Windows is MS-1252 Latin 1, and for AIX Version 4 is ISO8859-1. In these code pages the first 128 characters are the same, and thus, there is no impact in code points the English characters are used when remarks are entered for a defect in the OS/2 client, stored in the AIX server and retrieved by the Windows client.

For example, the code point value of 100 is the lower case letter "d" which has the same graphic representation in most of the code pages, as exemplified in the following table.

*Table 30. Graphical representation of code point 100 in several code pages*

Platform	Locale	Code Page	Representation
OS/2	English	IBM-437	lower case 'd'
OS/2	English	IBM-850	lower case 'd'
Windows, DOS mode	English	MS-437	lower case 'd'
Windows, DOS mode	English	MS-850	lower case 'd'

Table 30. Graphical representation of code point 100 in several code pages (continued)

Platform	Locale	Code Page	Representation
Windows, Graphical	English	MS-1252	lower case 'd'
AIX	En_US	IBM-850	lower case 'd'
AIX	en_US	ISO8859-1	lower case 'd'
OS/2	Japanese	IBM-932	lower case 'd'
Windows	Japanese	MS-932	lower case 'd'
AIX	ja_JP	IBM-eucJP	lower case 'd'

## Impact if using non-English characters

However, if the customer wants to use non-English characters, which are characters with code points greater than 128, such as accented characters, umlauts, double-byte characters, then the code pages differ greatly in this respect.

For example, the character with code point value of 252 (which can be entered by pressing ALT and typing 2, 5 and 2 from the numeric keypad in most systems) has the following different representations, as shown in the following table.

Table 31. Graphical representation of code point 100 in several code pages

Platform	Locale	Code Page	Representation
OS/2	English	IBM-437	superscript 'n'
OS/2	English	IBM-850	superscript '3'
Windows, DOS mode	English	MS-437	superscript 'n'
Windows, DOS mode	English	MS-850	superscript '3'
Windows, Graphical	English	MS-1252	lower case 'u' with dieresis
AIX	En_US	IBM-850	superscript '3'
AIX	en_US	ISO8859-1	lower case 'u' with dieresis
OS/2	Japanese	IBM-932	First byte of DBCS character
Windows	Japanese	MS-932	First byte of DBCS character
AIX	ja_JP	IBM-eucJP	First byte of DBCS character

In the above case, a German customer using Windows in Graphical Mode, with code page MS-1252 may enter a string that contains the u with umlaut and store it in TeamConnection, but the same customer when retrieving the data from OS/2 using IBM-850 code page, the character in the string will be shown as the number 3 in superscript.

## **To maximize compatibility, use same/similar code page**

As shown in “Impact if using non-English characters” on page 241, it is important that the customers who are using multiple platforms with TeamConnection, must understand the implications of using different code pages when dealing with non-English characters.

If possible, the customer should use the same (or similar) code page in the TeamConnection client and in the server.

## **Once a family is created, do not change the code page**

To avoid compatibility problems, if a family is created and used with a given code page, then this code page should not be changed later on.

For example, if a family is created with the Japanese IBM-932 code page in OS/2 and then migrated to the Japanese IBM-eucJP code page in AIX, then there might be several DBCS characters that are valid in the IBM-932 code page that will not be displayed properly when using the IBM-eucJP code page.

## **Using UNICODE in the future to solve incompatibilities**

In the future, once the support for the UNICODE code page is widespread and available in all the platforms supported by TeamConnection, then the customer could choose to use the UNICODE code page for the clients and the server, and in this way, avoid the current incompatibility between different code pages.

Another alternative that we studied to solve to this incompatibility problem between code pages was to add an extra field for EVERY SINGLE piece of data that is handled by TeamConnection in order to identify the code page that was used when the data was originated; then, this would require that the TeamConnection server should get the code page used by each client that is requesting a service, and then do the necessary conversions when exchanging the data. Because this alternative is very expensive to implement and has a lot of ramifications, and because UNICODE is the right way for the long term, we are not implementing this alternative to tag each piece of data.

## **Exceptions to the handling of characters in TeamConnection**

### **The | (split vertical bar) character could be changed**

The | (split vertical bar) character is used to separate the fields in the "teamc report -raw" command. Thus, if this character is found in a field that is shown by this command, such as in the abstract of a defect, then the character is changed to "!" (exclamation point) by the TeamConnection client. Thus, the server does not see these split vertical bar characters.

The reason for this change is to avoid confusion during the parsing of the -raw output because the split vertical bar is used to separate the fields. If in the output to be parsed

there is a split vertical bar character that is NOT intended to be a separator of a field, then the parsing routine will not be able to guess that this particular split vertical bar should not be considered as a field separator. In other words, ALL split vertical separator bars are considered to be field separators, and thus, any such characters in the abstract will not be parsed appropriately.

For example, when opening a defect, if the abstract field is left blank then the first 63 characters of the remarks field will be placed in the abstract. The abstract is a field that is shown with the "teamc report -raw" command, but the remarks field is not shown with this command. Thus, if the first 63 characters of the remarks have split vertical bar characters they will be left untouched in the actual remarks, but they will be converted to "!" in the abstract.

## **Keyword expansion**

TeamConnection supports the expansion of certain keywords embedded in the text during the extraction of text files. The routines that handle the expansion are NLS and DBCS enabled.

The important characteristic to remember is that the expansion is done by the TeamConnection family server and not by the client.

## **CR (carriage return) and LF (line feed)**

Although this is not an NLS issue, this is another topic that is worth including in this technical report, because some users may think, incorrectly, that this could be caused by code conversion processing done by TeamConnection.

The end of a line of text in OS/2 or in Windows is represented by the character pair CR-LF (carriage return and line feed), whereas in UNIX is represented simply by the character LF (line feed).

In TeamConnection, the model of what-you-see-is-what-you-get is used. This means that if a user creates a file in TeamConnection, regardless of the platform of the server, then TeamConnection will NOT do any conversion of LF or CRLF on that file. There are choices in the -extract action to allow for more fine tuning of these on-the-fly-conversions. For example, an AIX user may wish to extract with only LF a file that was stored originally from OS/2 that has CRLF.

The following file will be the source file to be used in the rest of the examples in this section:

```
This is line 1  
This is line 2  
This is line 3
```

If the source file is created from an OS/2 client and later on is extracted into a UNIX client without CRLF conversion, then the resulting file will have the CR character at the end of each line and the file would look like:

```
This is line 1^M
This is line 2^M
This is line 3^M
```

If the source file is created from a UNIX client and later on is extracted into an OS/2 client without CRLF conversion, then the resulting file will not have the CR character at the end of each line and the file would look like:

```
This is line 1
        This is line 2
                This is line 3
```

## All clients in the same host must use the same language (Intel only)

For OS/2 and Windows NT clients, all the TeamConnection clients that execute from one single host must use the same language if they run at the same time.

This limitation is due to the inherent limitation of these platforms in which ONLY ONE version of given DLL can be loaded at the same time, and because these platforms are not fully compliant with the XPG/4 model that allows usage of multiple locales. If there are different versions of some DLLs for each language, and if the English version of the DLL is loaded, the Japanese one cannot be loaded at the same time. This precludes having clients that have different languages to run at the same time.

## Untranslated strings that are visible to the users

There are certain kinds of strings that are visible to the user that are not translatable:

- command, action and flag names
- state names
- database table and view names
- database table column headings
- action name used in the audit log, the mail notifications, and the authority and interest tables
- the type field in the config database table

## DBCS Limitations

The following limitations apply to DBCS character sets:

1. The administration tools for the TeamConnection Server expect SBCS characters as the reply for Yes (y) and No (n).
2. The administration tools for the TeamConnection Server have the following limitations for DBCS:
  - a. The \*.ld files (authority, interest, cfgcomproc and cfgrelproc) in the family account can accept DBCS characters in the first field for each entry. The maximum size for this field is 15 bytes.

- b. The config.ld file in the family account can accept DBCS characters in the following fields (the positions are defined from left to right):
  - Field position 1 ("Field Type"): limit is 15 bytes
  - Field position 2 ("Value"): limit is 15 bytes
  - Field position 6 ("Description"): limit is 63 bytes
- c. The tcadmin program can accept only SBCS characters in the following fields related to configurable fields:
  - CMD attribute
  - DB Column Name
- d. The tcadmin program can accept DBCS characters in the following fields related to configurable fields:
  - Field label: limit is 15 bytes
  - Title label: limit is 15 bytes
  - Type: must be a valid type defined in config.ld (limit is 15 bytes).
3. The TeamConnection Commands Reference manual, in Appendix A, "Querying the TeamConnection database", shows the datatype and the size limit for the attributes of the TeamConnection objects; however, the actual size limit for many of the character attributes is smaller than the specified limit. For example, the field "login" in the "Users" table shows that the limit is 31 bytes, but in reality only 15 characters (SBCS or DBCS) can be stored in that field. The fields affected are usually related to names, such as the User login, the Component name, etc.

If you specify a string that has DBCS characters and that the size of the string goes beyond the limit, then the following error message will be displayed by the TeamConnection server:

```
0010-149 Your request cannot be completed.  
The attribute flag argument xxx is not valid.
```

4. Warning on the use of 0x7C as a second byte in a DBCS character  
The 0x7C character corresponds to the vertical bar ('|') which in TeamConnection is interpreted as a field separator when dealing with reports and with handling windows and fields in the GUI.  
You can use this value as the 2nd byte of a DBCS character, however, when the data that contains this 2nd byte is handled in a TeamConnection client that has an SBCS code page (and not a DBCS code page), then, the output shown by the client may be displaced, that is, the 0x7C value will be interpreted as the field separator. Moreover, this situation will apply for any string in the \*.ld files and in the configurable fields.

---

## Installation, administration, and runtime issues

### Installation issues related to NLS and DBCS

The installation process for TeamConnection is similar in UNIX, in OS/2 and Windows with respect to NLS. The similarities and the differences are explained in the following sections.

After the installation process, the executable code and the language related files will be installed in separate directories that are system wide, that is, they are not exclusive to one account.

When a TeamConnection family is created, several files are copied into the directory for the TeamConnection family; several of these files contain language sensitive information (such as the config.ld file and the files in the chfField directory). The family administrator can modify these files for the specific family; these files are not shared with other families.

## Using a similar directory structure across all the platforms

Even though there are differences in the NLS facilities that are available from the UNIX and the Intel (OS/2 and Windows) platforms the installation of TeamConnection in these platforms creates a similar directory structure whose top directory is shown below (using the default directory):

**AIX 4** /usr/teamc

**HP-UX 10**

/opt/teamc

**Solaris** /opt/teamc

**OS/2** c:\teamc

**Windows NT and 95**

c:\Program Files\TeamConnection

**Storing the language-independent files:** The language-independent files for the TeamConnection code are stored in similar directories, as shown in the following example. The teamc server daemon (teamcd) is located in the subdirectory "bin", from the TeamConnection top directory as shown below:

**AIX 4** /usr/teamc/bin

**HP-UX 10**

/opt/teamc/bin

**Solaris** /opt/teamc/bin

**OS/2** c:\teamc\bin

**Windows NT and 95**

c:\Program Files\TeamConnection\bin

**Storing the language-dependent files:** In a similar way, the language-dependent files for the TeamConnection code are stored in a similar subdirectory structure, which is the subdirectory "nls" as parent and then the "msg" for messages and "cfg" for configuration items.

For example, the ISO US English message catalog will be stored as shown below, using the default location:

**AIX 4** /usr/teamc/nls/msg/en\_US

**HP-UX 10**

/opt/teamc/nls/msg/C (which really is a symbolic link to /usr/lib/nls/msg/C)

**Solaris** /opt/teamc/nls/msg/C (which really is a symbolic link to /usr/lib/nls/msg/C)

**OS/2** c:\teamc\nls\msg\enu

**Windows NT and 95**

c:\Program Files\TeamConnection\nls\msg\enu

**List of language-dependent files:** The "nls" directory (see previous section for the complete path) contains the following subdirectories and files:

**nls/msg/<locale>/**

All message catalog files, such as teamcv3.cat; all help files; all resource DLLs for the GUI that are specific to a language.

**nls/doc/<locale>/**

All documentation: PDF, HTML, etc.

**nls/cfg/<locale>/**

All configuration files, such as config.ld, and files for the configurable fields; the original teamcv3.ini file.

## Installation issues for UNIX

During the installation process for TeamConnection in UNIX, it is necessary to select the appropriate language version to install. The code is not bundled together with the language sensitive information. That is, there is an individual installable package just for the language sensitive information that could be installed independently.

Because AIX 4 and HP-UX 10 operating systems already include the explicit support for the XPG/4 I18N locale model, the TeamConnection installation process will not install additional files for this matter (as in OS/2 and Windows).

The message catalog that contains the language sensitive information is located by the executable code by means of the combination of the NLSPATH and LANG environment variable. By default, this variable is set as follows:

```
set NLSPATH=/usr/lib/nls/msg/%L/%N
```

Where:

- %L is a variable that at runtime represents the value of the LANG environment variable; it must be in uppercase.
- %N is a variable that at runtime represents the name of the message catalog to be used; it must be in uppercase.

## Installation issues with OS/2 and Windows

During the installation process for TeamConnection in OS/2 and Windows, it is necessary to select the appropriate language version to install. The code and the language sensitive information is bundled together in a package and it is installed appropriately. That is, there is not an individual package just for the language sensitive information that can be installed independently.

Because the OS/2 and Windows operating systems do not include at this moment explicit support for the XPG/4 I18N locale model, the TeamConnection installation process will install any necessary support for this model.

The message catalog that contains the language sensitive information is located by the executable code by means of the NLSPATH environment variable. By default, this variable is set as follows:

```
set NLSPATH=:\teamc\nls\%N
```

Where:

- `:\teamc` represents the appropriate drive and top directory where the TeamConnection code is installed in your system
- `nls` is the directory that contains the NLS related files
- `%N` is a variable that at runtime represents the name of the message catalog to be used; it must be in uppercase.

## Family administration issues

### A family should use the same language all the time

Although technically it could be possible for a family to be created using the `en_US` locale and then change it later on to another language, we consider that this process has the potential to cause a lot of confusion with the users, especially for the mapping of code points.

Therefore, this is treated as a limitation and if the customers try it, it is at their own risk and we will not help them. However, the customer may decide to delete the family, change the language by reinstalling the code for Intel and specify the new language, or to install the new language message catalogs for UNIX and change the `LANG` variable, and then create a new family to use the new setting.

This decision affects the arrangement of the subdirectories of a family: there is no provision in either UNIX or Intel to have language dependent directories inside the family directory.

The following sections contain examples that will clarify this point.

**UNIX:** An AIX customer installed the TeamConnection server, using the `en_US` locale. The `config.ld` file (which is language dependent) resides in `/usr/lib/nls/ctg/enu/config.ld`.

The "testfam" TeamConnection family is created, and the config.ld file is copied from the system directory to the top directory of the family, /home/testfam/config.ld There is no "/home/testfam/enu/config.ld" path.

**Intel:** An OS/2 customer installed the TeamConnection server, using the en\_US locale. The config.ld file (which is language dependent) resides in c:\teamc\nls\cfg\enu\config.ld. The "testfam" TeamConnection family is created, and the config.ld file is copied from the system directory mentioned above into the top directory of the family, c:\testfam\config.ld. Notice that there is no "c:\testfam\enu\config.ld" path.

## Client runtime issues

### A client should use the same language all the time

Although technically it could be possible for a TeamConnection client to be installed with one language (such as the IBM-850 code page in OS/2 or the en\_US locale in AIX) and then change the language in the middle, this process has the potential to cause a lot of confusion with the users, specially for the mapping of code points with the teamcv3.ini file, as explained below.

In the UNIX platforms, thanks to the use of the LANG variable, it would be possible to install additional message catalogs for other languages and the user could setup the language to use by setting the variable LANG. However, the teamcv3.ini file for the GUI will NOT be changed, and this file may contain characters that were valid in the original setup but that cannot be displayed in the new setup.

Because the Intel platforms do not provide the LANG variable, then it is not possible to have message catalogs for multiple languages for TeamConnection. This means that if the customer decides to change the language then it is necessary to reinstall the code specifying the new language.

The following sections contain examples that will clarify this point.

#### **UNIX:**

1. A Japanese AIX customer installs the TeamConnection client using the ja\_JP and en\_US message catalogs. The teamcv3.ini files (which are language dependent) reside in /usr/lib/nls/cfg/ja\_JP/teamcv3.ini and /usr/lib/nls/cfg/enu/teamcv3.ini.
2. The customer uses the Japanese GUI for the first time (LANG=ja\_JP), and the GUI detects that the following file does not exist: :\$HOME/teamcv3.ini.
3. The customer uses the GUI and creates several entries written in Japanese in the task list which are stored in the teamcv3.ini file.
4. The customer exits the GUI.
5. The user invokes the GUI again, and the GUI detects that the teamcv3.ini file exists in \$HOME and therefore the GUI uses it, and does not try to overwrite it with the file in the directory /usr/lib/nls/cfg/ja\_JP.
6. The customer decides then to switch the locale to en\_US, by setting LANG=en\_US, exits and logs in again.

7. The user brings up the English TeamConnection GUI and now the task list shows entries that may not be legible because their original code points were set with the ja\_JP locale.

**Intel:**

1. A Japanese OS/2 customer installs the TeamConnection client and specifies the jpn language only, because she cannot install multiple languages. The code page is IBM-932. The PATH and the NLSPATH variables point to the jpn directories. The teamcv3.ini file (which is language dependent) resides in c:\teamc\nls\cfg\jpn\teamcv3.ini.
2. The customer uses the Japanese GUI for the first time (LANG=jpn), and the GUI detects that the following file does not exist: c:\os2\teamcv3.ini. The GUI copies the original teamcv3.ini file from the appropriate jpn directory, c:\teamc\nls\cfg\jpn\teamcv3.ini, into c:\os2\teamcv3.ini.
3. The customer uses the GUI and creates several entries written in Japanese in the task list, and this list is stored in the teamcv3.ini file.
4. The customer exits the GUI.
5. The user invokes the GUI again, and the GUI detects that the teamcv3.ini file exists in c:\os2 and therefore the GUI uses it, and does not try to overwrite it with the file in the directory c:\teamc\nls\cfg\jpn.
6. The customer decides then to switch the locale to enu, by uninstalling the TeamConnection client and reinstalling it again specifying now the language enu, and reboots. The PATH and the NLSPATH variables are updated and they do not point to the non-existing jpn directories, but point to the new enu directories.
7. If the customer keeps the same code page, IBM-932, then when the user brings up the English TeamConnection GUI, the task list shows entries that are legible because their original code points were set with the IBM-932 code page. If the customer changes the code page, let's say to IBM-850, then when the user brings up the English TeamConnection GUI the task list shows entries that may not be legible because their original code points were set with the IBM-932 code page.

## Appendix F. Worksheets

### Authority groups worksheet

The following table lists TeamConnection actions. Use this table to record the authority groups for your family if you are using groups other than those supplied by IBM. Those TeamConnection actions that cannot be included in an authority group are marked with information about how they can be performed.

TeamConnection action	Authority groups for family: _____									
AccessCreate										
AccessDelete										
AccessRestrict										
ApprovalAbstain										
ApprovalAccept										
ApprovalAssign										
ApprovalCreate										
ApprovalDelete										
ApprovalReject										
ApproverCreate										
ApproverDelete										
BuilderCreate										
BuilderDelete										
BuilderExtract										
BuilderModify										
BuilderView										
CollisionAccept										
CollisionReconc										
CollisionReject										
CompCreate										
CompDelete										
CompLink										
CompModify										
CompRecreate										
CompUnlink										
CompView										
CoreqCreate										

TeamConnection action	Authority groups for family: _____									
CoreqDelete										
DefectAccept										
DefectAssign										
DefectCancel										
DefectClose	Automatic action									
DefectComment	Base authority									
DefectDesign										
DefectModify										
DefectOpen	Base authority									
DefectReopen										
DefectReturn										
DefectReview										
DefectSize										
DefectVerify										
DefectView										
DriverAssign										
DriverCheck										
DriverCommit										
DriverComplete										
DriverCreate										
DriverDelete										
DriverExtract										
DriverFreeze										
DriverModify										
DriverRefresh										
DriverRestrict										
DriverView										
EnvCreate										
EnvDelete										
EnvModify										
FeatureAccept										
FeatureAssign										
FeatureCancel										
FeatureClose	Automatic action									
FeatureComment	Base authority									

TeamConnection action	Authority groups for family: _____									
FeatureDesign										
FeatureModify										
FeatureOpen	Base authority									
FeatureReopen										
FeatureReturn										
FeatureReview										
FeatureSize										
FeatureVerify										
FeatureView										
FixActive										
FixAssign										
FixComplete										
FixCreate										
FixDelete										
HostCreate	Superuser, admin, or owner implicit authority									
HostDelete	Superuser, admin, or owner implicit authority									
MemberCreate										
MemberCreateR										
MemberDelete										
MemberDeleteR										
NotifyCreate										
NotifyDelete										
ParserCreate										
ParserDelete										
ParserModify										
ParserView										
PartAdd										
PartBuild										
PartCheckIn										
PartCheckOut										
PartChildInfo										
PartConnect										
PartDelete										
PartDeleteForce										
PartDisconnect										

TeamConnection action	Authority groups for family: _____									
PartExtract										
PartForceIn										
PartForceOut										
PartLink										
PartLock										
PartLockForce										
PartMark										
PartModify										
PartOverrideR										
PartRecreate										
PartRecreaForce										
PartRename										
PartRenameForce										
PartResolve	Base authority									
PartRestrict										
PartTouch										
PartUndo										
PartUndoForce										
PartUnlock										
PartView										
PartViewmsg										
PrereqCreate										
PrereqDelete										
ReleaseCreate										
ReleaseDelete										
ReleaseExtract										
ReleaseLink										
ReleaseModify										
ReleasePrune										
ReleaseRecreate										
ReleaseView										
Report	Base authority									
ShadowCreate										
ShadowDefine	Superuser									
ShadowDelete										

TeamConnection action	Authority groups for family: _____									
ShadowDisable										
ShadowEnable										
ShadowModify										
ShadowRedefine	Superuser									
ShadowSync										
ShadowUndefine	Superuser									
ShadowVerify										
ShadowView										
SizeAccept										
SizeAssign										
SizeCreate										
SizeDelete										
SizeReject										
TestAbstain										
TestAccept										
TestAssign										
TestReady	Automatic action									
TestReject										
UserCreate	Superuser implicit or admin authority									
UserDelete	Superuser implicit or admin authority									
UserModify	Superuser, admin, or owner implicit authority									
UserRecreate	Superuser implicit or admin authority									
UserView										
VerifyAbstain										
VerifyAccept										
VerifyAssign										
VerifyReady	Automatic action									
VerifyReject										
WorkAreaAssign										
WorkAreaCancel										
WorkAreaCheck										
WorkAreaCommit										
WorkAreaComple										
WorkAreaCreate										
WorkAreaFix										

TeamConnection action	Authority groups for family: _____									
WorkAreaFreeze										
WorkAreaIntegra										
WorkAreaModify										
WorkAreaRefresh										
WorkAreaTest										
WorkAreaView										

## Interest groups worksheet

The following table lists the TeamConnection actions. Use this table to record the interest groups for your family if you are using groups other than those supplied by IBM. Those TeamConnection actions that cannot be included in an interest group are marked with information about how users are notified.

TeamConnection actions	Interest groups for family: _____									
AccessCreate										
AccessDelete										
AccessRestrict										
ApprovalAbstain										
ApprovalAccept										
ApprovalAssign										
ApprovalCreate										
ApprovalDelete										
ApprovalReject										
ApproverCreate										
ApproverDelete										
BuilderCreate										
BuilderDelete										
BuilderExtract	No notification									
BuilderModify										
BuilderView	No notification									
CollisionAccept										
CollisionReconc										
CollisionReject										
CompCreate	New owner implicit notification									
CompDelete										

TeamConnection actions	Interest groups for family: _____						
CompLink							
CompModify							
CompRecreate							
CompUnlink							
CompView	No notification						
CoreqCreate	No notification						
CoreqDelete	No notification						
DefectAccept							
DefectAssign							
DefectCancel							
DefectClose							
DefectComment							
DefectDesign							
DefectModify							
DefectOpen							
DefectReopen							
DefectReturn							
DefectReview							
DefectSize							
DefectVerify							
DefectView	No notification						
DriverAssign							
DriverCheck	No notification						
DriverCommit							
DriverComplete							
DriverCreate							
DriverDelete							
DriverExtract	No notification						
DriverFreeze							
DriverModify							
DriverRefresh	No notification						
DriverRestrict	Owner implicit notification						
DriverView	No notification						
EnvCreate							
EnvDelete							

TeamConnection actions	Interest groups for family: _____						
EnvModify							
FeatureAccept							
FeatureAssign							
FeatureCancel							
FeatureClose							
FeatureComment							
FeatureDesign							
FeatureModify							
FeatureOpen							
FeatureReopen							
FeatureReturn							
FeatureReview							
FeatureSize							
FeatureVerify							
FeatureView	No notification						
FixActive							
FixAssign							
FixComplete							
FixCreate							
FixDelete							
HostCreate	No notification						
HostDelete	No notification						
MemberCreate							
MemberCreateR	New owner implicit notification						
MemberDelete							
MemberDeleteR	Owner implicit notification						
NotifyCreate	No notification						
NotifyDelete	No notification						
ParserCreate							
ParserDelete							
ParserModify							
ParserView	No notification						
PartAdd							
PartBuild	owner implicit notification						
PartCheckIn							

TeamConnection actions	Interest groups for family: _____						
PartCheckOut							
PartConnect							
PartDelete							
PartDisconnect							
PartExtract	No notification						
PartForceIn							
PartForceOut							
PartLink							
PartLock							
PartLockForce	PartLock subscribers						
PartMark							
PartModify							
PartOverrideR							
PartRecreaForce	PartRecreate subscribers						
PartRecreate							
PartRename							
PartRenameForce	PartRename subscribers						
PartResolve	No notification						
PartRestrict							
PartTouch	No notification						
PartUndo							
PartUndoForce	PartUndo subscribers						
PartUnlock							
PartView	No notification						
PartViewmsg	No notification						
PrereqCreate	New owner implicit notification						
PrereqDelete	Owner implicit notification						
ReleaseCreate							
ReleaseDelete							
ReleaseExtract	No notification						
ReleaseLink							
ReleaseModify							
ReleaseRecreate							
ReleaseView	No notification						
Report	No notification						

TeamConnection actions	Interest groups for family: _____						
ShadowCreate	No notification						
ShadowDefine	No notification						
ShadowDelete	No notification						
ShadowDisable	No notification						
ShadowEnable	No notification						
ShadowModify	No notification						
ShadowRedefine	No notification						
ShadowSync	No notification						
ShadowUndefine	No notification						
ShadowVerify	No notification						
ShadowView	No notification						
SizeAccept							
SizeAssign							
SizeCreate							
SizeDelete							
SizeReject							
TestAbstain							
TestAccept							
TestAssign							
TestCreate							
TestDelete							
TestReady	Owner implicit notification						
TestReject							
UserCreate	New user implicit notification						
UserDelete	No notification						
UserModify	No notification						
UserUnDelete	No notification						
UserView	No notification						
VerifyAbstain							
VerifyAccept							
VerifyAssign							
VerifyReady	Owner implicit notification						
VerifyReject							
WorkAreaAssign							
WorkAreaCancel							

TeamConnection actions	Interest groups for family: _____					
WorkAreaCheck	No notification					
WorkAreaCommit						
WorkAreaCompleat						
WorkAreaCreate						
WorkAreaExtract						
WorkAreaFix						
WorkAreaFreeze	Owner implicit notification					
WorkAreaIntegra						
WorkAreaModify						
WorkAreaRefresh						
WorkAreaTest						
WorkAreaView	No notification					

## Configurable processes worksheets

The following worksheets list the TeamConnection subprocesses. Use these worksheets to record the processes that you have created for your family. Separate worksheets are provided for component and release processes. For more information on configuring processes, see “Chapter 8. Configuring family processes” on page 99.

TeamConnection component subprocesses	Component processes for _____					
dsrDefect						
dsrFeature						
verifyDefect						
verifyFeature						
none						

TeamConnection release subprocesses	Release processes for _____					
track						
approval						
fix						
driver						

TeamConnection release subprocesses	Release processes for _____					
test						
trackfixhold						
trackcommithold						
tracktesthold						
none						

---

## Service and Support

---

### VisualAge TeamConnection Services!

VisualAge TeamConnection services offerings will provide customers with the tools to quickly establish a more productive and efficient development environment. These services offerings focus on the LAN library component of VisualAge TeamConnection and on the Repository function of VisualAge TeamConnection.

If you are interested in the VisualAge TeamConnection Services, select the **Support** item at:

<http://www.software.ibm.com/software/ad/teamcon>

---

### VisualAge TeamConnection Support!

If you have a question or a problem, please take a moment to review the Customer Support section from any of the manuals for the VisualAge TeamConnection product. Your options for VisualAge TeamConnection support, as described in your License Information, include the following information (subject to availability).

### IBM Lotus Passport Advantage Program

For more information on the IBM Lotus Passport Advantage volume licensing program that provides customers with a series of contract offerings under which they can acquire licenses, software subscriptions, and support, go to:

<http://www.lotus.com/passportadvantage>

### DB2 Service Maintenance and Technical Library

To download the latest service maintenance for DB2, use the DB2 Service and Support on the World Wide Web at:

<http://www.software.ibm.com/data/db2/db2tech>

**Note:** Even though DB2 is bundled with VisualAge TeamConnection you should contact VisualAge TeamConnection Support to report DB2 problems. The licensing for VisualAge TeamConnection does not entitle you to contact DB2 Support directly. For a complete and up-to-date source of DB2 information, use the DB2 Product and Service Technical Library, in English only, on the World Wide Web at:

<http://www.software.ibm.com/data/db2/library>

## For North American Customers

### Electronic Forums

So you may electronically access VisualAge TeamConnection technical information, exchange messages with other VisualAge TeamConnection users, and receive information regarding the availability of FixPaks.

#### IBM TalkLink

Use the TEAMC CFORUM. In the United States, call 1-800-547-1283. For TalkLink information available via the Internet, go to:

<http://www.ibmmlink.ibm.com/talklink>

#### CompuServe

From any ! prompt, type GO SOFSOL, then select TeamConnection. Refer to the enclosed CompuServe brochure for additional information, or call 1-800-848-8199. For CompuServe information available via the Internet, refer to:

<http://www.compuserve.com>

#### Internet

Go to the IBM homepage, <http://www.ibm.com>. Use the search function with keyword TeamConnection to go to the VisualAge TeamConnection area. Access the TeamConnection directory in our ftp site. Use ftp and login as anonymous to <ftp://software.ibm.com>. In the directory [ps/products/teamconnection](ftp://software.ibm.com/ps/products/teamconnection) you can find fixes and information related to VisualAge TeamConnection.

### Telephone Support

Direct customer support is provided by the Personal Systems Support Line and by the AIX Support Line. These fee services enhance customers' productivity by providing voice and electronic access to the IBM support organization. They will help answer questions pertaining to usage, "how to," and suspected software defects for eligible products.

The following are phone numbers for software support in the US:

- Personal Systems Support Line: 1-800-237-5511
- AIX Support Line: 1-800-CALL-AIX (1-800-225-5249)

In Canada, call 1-800-IBM-SERV (1-800-426-7378).

**Note:** In the US, 1-800-237-5511 is the Software Support phone number for all IBM software (390, OS/400, AIX, Personal Systems, etc.). You may call this number and take the option for OS/2 - DOS support, which then transfers you to 1-800-992-4777 for the Personal Systems (workstation) products, or you may call 1-800-992-4777 directly.

Obtaining product information packages:

- United States: 1-800-IBM-CALL (1-800-426-2255)
- Canada: 1-800-IBM-CALL (1-800-426-2255)

Ordering TeamConnection products:

- United States: 1-800-IBM-CALL (1-800-426-2255)
- Canada: 1-800-IBM-CALL (1-800-426-2255)

TeamConnection education:

- United States: 1-800-IBM-TEACH (1-800-426-8322) Canada 1-800-IBM-TEACH (1-800-426-8322)

While we may not be able to respond to or resolve all problems and questions, your satisfaction with our products and support is important to us. If you cannot access these forums, contact your IBM representative. There are several other support offerings available after purchasing IBM VisualAge TeamConnection.

If you live within the U.S.A., call any of the following numbers:

- 1-800-237-5511 to learn about available service options
- 1-800-IBM-CALL (1-800-426-2255) to order products or get general information
- 1-800-879-2755 to order publications.

## Support for Customers Outside North America

For information on how to contact IBM outside of the United States, see Appendix A of the *IBM Software Support Handbook*, which can be located by selecting the **Service Offering** item at:

<http://www.ibm.com/support>:

**Note:** In some countries, IBM-authorized dealers should contact their dealer support structure instead of the IBM Support Center.



---

## Bibliography

---

### IBM VisualAge TeamConnection Enterprise Server library

The following is a list of the TeamConnection publications. For a list of other publications about TeamConnection, including white papers, technical reports, a product fact sheet, and the product announcement letter, refer to the IBM VisualAge TeamConnection Enterprise Server Library home page. To access this home page, select **Library** from the IBM VisualAge TeamConnection Enterprise Server home page at URL <http://www.software.ibm.com/ad/teamcon>.

- **License Information (GC34-4497):**  
Contains license, service, and warranty information.
- **Installation Guide (GC34-4742):**  
Lists the hardware and software that are required before you can install and use the IBM VisualAge TeamConnection Enterprise Server product, provides detailed instructions for installing the TeamConnection server and client.
- **Administrator's Guide (GC34-4551):**  
Provides instructions for configuring the TeamConnection family server and administering a TeamConnection family.
- **Getting Started with the TeamConnection Clients (SC34-4552):**  
Tells first-time users how to install the TeamConnection clients on their workstations, and familiarizes them with the command line and graphical user interfaces.
- **User's Guide (SC34-4499):**  
A comprehensive guide for TeamConnection administrators and client users that helps them install and use TeamConnection.
- **Commands Reference (SC34-4501):**  
Describes the TeamConnection commands, their syntax, and the authority required to issue each command. This book also provides examples of how to use the various commands.
- **Quick Commands Reference (GC34-4500):**  
Lists the TeamConnection commands along with their syntax.
- **Staying on Track with TeamConnection Processes (83H9677):**  
Poster showing how objects flow through the states defined for each TeamConnection process.
- The following publications can be ordered as a set (SBOF-8560):
  - Administrator's Guide**
  - Getting Started with the TeamConnection Clients**
  - User's Guide**
  - Commands Reference**
  - Quick Commands Reference**
  - Staying on Track with TeamConnection Processes**

---

## TeamConnection technical reports

The following is a list of technical reports available for TeamConnection. Refer to the IBM VisualAge TeamConnection Enterprise Server Library home page for the most up-to-date list of technical reports. To access this home page, select **Library** from the IBM VisualAge TeamConnection Enterprise Server home page at URL <http://www.software.ibm.com/ad/teamcon>.

<b>29.2147</b>	SCLM Guide to TeamConnection Terminology
<b>29.2196</b>	Using REXX command files with TeamConnection MVS Build Scripts
<b>29.2231</b>	TeamConnection Interoperability with MVS and SCLM
<b>29.2235</b>	Using REXX command files with TeamConnection MVS Build Scripts for PL/I programs
<b>29.2253</b>	Comparison between CMVC 2.3 and TeamConnection 2
<b>29.2254</b>	Migrating from CMVC 2.3 to TeamConnection 2
<b>29.2267</b>	TeamConnection frequently asked questions: how to do routine operating system tasks

---

## DB2

The following publications are part of the IBM DB2 Universal Database library of documents for DB2 administration. DB2 publications are available in HTML format from the DB2 Product and Service Technical Library at the following URL:

<http://www.software.ibm.com/data/db2/library/>

- *Administration Getting Started* (S10J-8154-00)  
An introductory guide to basic administration tasks and the DB2 administration tools.
- *SQL Getting Started* (S10J-8156-00)  
Discusses basic concepts of DB2 SQL.
- *Administration Guide* (S10J-8157-00)  
A complete guide to administration tasks and the DB2 administration tools.
- *SQL Reference* (S10J-8165-00)  
A reference to DB2 SQL for programmers and database administrators.
- *Troubleshooting Guide* (S10J-8169-00)  
A guide to identifying and solving problems with DB2 servers and clients and to using the DB2 diagnostic tools.
- *Messages Reference* (S10J-8168-00)  
Provides detailed information about DB2 messages.
- *Command Reference* (S10J-8166-00)  
Provides information about DB2 system commands and the command line processor.
- *Replication Guide* (S10J-0999-00)  
Describes how to plan, configure, administer, and operate IBM replication tools available with DB2.
- *System Monitor Guide and Reference* (S10J-8164-00)

Describes how to monitor DB2 database activity and analyze system performance.

- *Glossary*  
A comprehensive glossary of DB2 terms.

---

## Related publications

- Transmission Control Protocol/Internet Protocol (TCP/IP)
  - *TCP/IP 2.0 for OS/2: Installation and Administration* (SC31-6075)
  - *TCP/IP for MVS Planning and Customization* (SC31-6085)
- MVS
  - *MVS/XA JCL User's Guide* (GC28-1351)
  - *MVS/XA JCL Reference* (GC28-1352)
  - *MVS/ESA JCL User's Guide* (GC28-1830)
  - *MVS/ESA JCL Reference* (GC28-1829)
- NLS and DBCS
  - *AIX 4, General Programming Concepts: Writing and Debugging Programs*. (SC23-2533-02). See chapter 16 "National Language Support" for an updated contents of the AIX 3 material (see below).
  - *AIX 4, System Management Guide: Operating System and Devices* (SC23-2525-03). See chapter 10, "National Language Support" for system tasks.
  - *AIX Version 3.2 for RISC System/6000, National Language Support* (GG24-3850).
  - *Internationalization of AIX Software, A Programmer's Guide* (SC23-2431).
  - *National Language Design Guide Volume 1* (SE09-8001-02). This manual contains very good information on how to enable an application for NLS.
  - *National Language Design Guide Volume 2* (SE09-8002-02). This manual provides information on the IBM language codes (consult the "Language codes" chapter).



---

## Glossary

This glossary includes terms and definitions from the *IBM Dictionary of Computing*, 10th edition (New York: McGraw-Hill, 1993). If you do not find the term you are looking for, refer to this document's index or to the *IBM Dictionary of Computing*.

This glossary uses the following cross-references:

### Compare to

Indicates a term or terms that have a similar but not identical meaning.

### Contrast with

Indicates a term or terms that have an opposed or substantially different meaning.

### See also

Refers to a term whose meaning bears a relationship to the current term.

## A

**absolute path name.** A directory or a part expressed as a sequence of directories followed by a part name beginning from the root directory.

**access list.** A set of objects that controls access to data. Each object consists of a component, a user, and the authority that the user is granted or is restricted from in that component. See also *authority*, *granted authority*, and *restricted authority*.

**action.** A task performed by the TeamConnection server and requested by a TeamConnection client. A TeamConnection action is the same as issuing one TeamConnection command.

**agent.** See *build agent*.

**alternate version ID.** In collision records, the database ID of the version of a driver, release, or work area where the conflicting version of a part is visible.

**approval record.** A status record on which an approver must give an opinion of the proposed part changes required to resolve a defect or implement a feature in a release.

**approver.** A user who has the authority to mark an approval record with accept, reject, or abstain within a specific release.

**approver list.** A list of user IDs attached to a release, representing the users who must review part changes that are required to resolve a defect or implement a feature in that release.

**attribute.** Information contained in a field that is accessible to the user. TeamConnection enables family administrators to customize defect, feature, user, and part tables by adding new attributes.

**authority.** The right to access development objects and perform TeamConnection commands. See also *access list*, *base authority*, *explicit authority*, *granted authority*, *implicit authority*, *restricted authority*, and *superuser privilege*.

## B

**base authority.** The set of actions granted to a user when a user ID is created within a TeamConnection family. See also *authority*. Contrast with *implicit authority* and *explicit authority*.

**base name.** The name assigned to the part outside of the TeamConnection server environment, excluding any directory names. See also *path name*.

**base part tree.** The base set of parts associated with a release, to which changes are applied over time. Each committed driver or work area for a release updates the base part tree for that release.

**build.** The process used to create applications within TeamConnection.

**build associate.** A TeamConnection part that is not an input to or an output from a build. An example of such a part is a read.me file.

**build cache.** A directory that the build processor uses to enhance performance.

**build dependent.** A TeamConnection part that is needed for the compile operation to complete, but it will not be passed directly to the compiler. An example of this is an include file. See also *dependencies*.

**builder.** An object that can transform one set of TeamConnection parts into another by invoking tools such as compilers and linkers.

**build event.** An individual step in the build of an application, such as the compiling of hello.c into hello.obj.

**build input.** A TeamConnection part that will be used as input to the object being built.

**build output.** A TeamConnection part that will be generated output from a build, such as an .obj or .exe file.

**build pool.** A group of build servers that resides in an environment. The environment in which several build servers operate. Typically, several servers are set up for each environment that the enterprise develops applications for.

**build scope.** A collection of build events that implement a specific build request. See also *build event*.

**build script.** An executable or command file that specifies the steps that should occur during a build operation. This file can be a compiler, a linker, or the name of a .cmd file you have written.

**build server.** A program that invokes the tools, such as compilers and linkers, that construct an application.

**build target.** The name of the part at the top of the build tree which is the final output of a build.

TeamConnection uses the build target to determine the scope of the build. See also *build tree*.

**build tree.** A graphical representation of the dependencies that the parts in an application have on one another. If you change the relationship of one part to another, the build tree changes accordingly.

## C

**change control process.** The process of limiting and auditing changes to parts through the mechanism of checking parts in and out of a central, controlled, storage location. Change control for individual releases can be integrated with problem tracking by specifying a process for the release that includes the tracking subprocess.

**check in.** The return of a TeamConnection part to version control.

**check out.** The retrieval of a version of a part under TeamConnection control. In non-concurrent releases, the check out operation does not allow a second user to check out a part until the first user has checked it back in.

**child component.** Any component in a TeamConnection family, except the root component, that is created in reference to an existing component. The existing component is the parent component, and the new component is the child component. A parent component can have more than one child component, and a child component can have more than one parent component. See also *component* and *parent component*.

**child part.** Any part in a build tree that has a parent defined. A child part can be input, output, or dependent. See also *part* and *parent part*.

**client.** A functional unit that receives shared services from a server. Contrast with *server*.

**collision record.** A status record associated with a work area or driver, a part, and one of the following:

- The work area or driver's release
- Another work area

TeamConnection generates a collision record when a user attempts to replace an older version of a part with a modified version, another user has already modified that part, and the first user's modification is not based on this latest version of the part.

**command.** A request to perform an operation or run a program from the command line interface. In TeamConnection, a command consists of the command name, one action flag, and zero or more attribute flags.

**command line.** (1) An area on the Tasks window or in the TeamConnection Commands window where a user can type TeamConnection commands. (2) An area on an operating system window where you can type TeamConnection commands.

**committed version.** The revision of a part that is visible from the release.

**common part.** A part that is shared by two or more releases, and the same version of the part is the current version for those releases.

**comparison operator.** An operator used in comparison expressions. Comparison operators used in TeamConnection are > (greater than), < (less than), >= (greater than or equal to), <= (less than or equal to), = (equal to), and <> (different from).

**component.** A TeamConnection object that organizes project data into structured groups, and controls configuration management properties. Component owners can control access to data and notification of TeamConnection actions. Components exist in a parent-child hierarchy, with descendant components inheriting access and notification information from ancestor components. See also *access list* and *notification list*.

**concurrent development.** Several users can work on the same part at the same time. TeamConnection requires these users to reconcile

their changes when they commit or integrate their work areas and drivers with the release. Contrast with *serial development*. See also *work area*.

**configuration management.** The process of identifying, managing, and controlling software modules as they change over time.

**connecting parts.** The process of linking parts so that they are included in a build.

**context.** The current work area or driver used for part operations.

**corequisite work areas.** Two or more work areas designated as corequisites by a user so that all work areas in the corequisite group must be included as members in the same driver, before that driver can be committed. If the driver process is not used in the release, then all corequisite work areas must be integrated by the same command. See also *prerequisite work areas*.

**current version.** The last visible modification of a part in a driver, release, or work area.

**current working directory.** (1) The directory that is the starting point for relative path names. (2) The directory in which you are working.

## D

**daemon.** A program that runs unattended to perform a standard service. Some daemons are triggered automatically to perform their task; others operate periodically.

**database.** A collection of data that can be accessed and operated upon by a data processing system for a specific purpose.

**default.** A value that is used when an alternative is not specified by the user.

**default query.** A database search, defined for a specific TeamConnection window, that is issued each time that TeamConnection window is opened. See also *search*.

**defect.** A TeamConnection object used to formally report a problem. The user who opens a defect is the defect originator.

**delete.** If you delete a development object, such as a part or a user ID, any reference to that object is removed from TeamConnection. Certain objects can be deleted only if certain criteria are met. Most objects that are deleted can be re-created.

**delta part tree.** A directory structure representing only the parts that were changed in a specified place.

**dependencies.** In TeamConnection builds there are two types of dependencies:

- **automatic.** These are build dependencies that a parser identifies.
- **manual.** These are build dependencies that a user explicitly identifies in a build tree.

See also *build dependent*.

**descendant.** If you descendant a development object, such as, a part or a user ID, any reference to that object is removed from TeamConnection. Certain objects can be descendant only if certain criteria are met. Most objects that are descendants can be re-created.

**disconnecting parts.** The process of unlinking parts so that they are not included in a build.

**driver.** A collection of work areas that represent a set of changed parts within a release. Drivers are only associated with releases whose processes include the track and driver subprocesses.

**driver member.** A work area that is added to a driver.

## E

**end user.** See *user*.

**environment.** (1) A user-defined testing domain for a particular release. (2) A defect field, in which

case it is the environment where the problem occurred. (3) The string that matches a build server with a build event.

**environment list.** A TeamConnection object used to specify environments in which a release should be tested. A list of environment-user ID pairs attached to a release, representing the user responsible for testing each environment. Only one tester can be identified for an environment.

**explicit authority.** The ability to perform an action against a TeamConnection object because you have been granted the authority to perform that action. Contrast with *base authority* and *implicit authority*.

**extract.** A TeamConnection action you can perform on a builder, part, driver or release builder. An extraction results in copying the specified builder, part, or parts contained in the driver or release to a client workstation.

## F

**family.** A logical organization of related data. A single TeamConnection server can support multiple families. The data in one family cannot be accessed from another family.

**family administrator.** A user who is responsible for all nonsystem-related tasks for one or more TeamConnection families, such as planning, configuring, and maintaining the TeamConnection environment and managing user access to those families.

**family server.** A workstation running the TeamConnection server software.

**FAT.** See *file allocation table*.

**feature.** A TeamConnection object used to formally request and record information about a functional addition or enhancement. The user who opens a feature is the feature originator.

**file.** A collection of data that is stored by the TeamConnection server and retrieved by a path name. Any text or binary file used in a

development project can be created as a TeamConnection file. Examples include source code, executable programs, documentation, and test cases.

**file allocation table (FAT).** The DOS-, OS/2-, Windows 95-, and Windows NT-compatible file system that manages input, output, and storage of files on your system. File names can be up to 8 characters long, followed by a file extension that can be up to 3 characters.

**fix record.** A status record that is associated with a work area and that is used to monitor the phases of change within each component that is affected by a defect or feature for a specific release.

**freeze.** The freeze action saves changed parts to the work area. Thus, TeamConnection takes a snapshot of the work area, including all of the current versions of parts visible from that work area, and saves this image of the system. The user can always come back to this stage of development in the work area. Note, however, that a freeze action does not make the changes visible to the other people working in the release. Compare with *refresh*.

**full part tree.** A directory structure representing a complete set of active parts associated with the release.

## G

**Gather.** A tool to organize files for distribution into a specified directory structure. This tool can be used as a prelude to further distribution, such as using CD-ROM or through electronic means like NetView DM/2. It can also be used by itself for distributing file copies to network-attached file systems.

**GID.** A number which uniquely identifies a file's group to a UNIX system.

**granted authority.** If an authority is granted on an access list, then it applies for all objects managed by this component and any of its descendants for which the authority is not

restricted. See also *access list*, *authority*, and *inheritance*. Contrast with *restricted authority*.

**graphical user interface (GUI).** A type of computer interface consisting of a visual metaphor of a real-world scene, often as a desktop. Within that scene are icons, representing actual objects, that the user can access and manipulate with a pointing device.

**GUI.** Graphical user interface.

## H

**high-performance file system (HPFS).** In the OS/2 operating system, an installable file system that uses high-speed buffer storage, known as a cache, to provide fast access to large disk volumes. The file system also supports the existence of multiple, active file systems on a single personal computer, with the capacity of multiple and different storage devices. File names used with HPFS can have as many as 254 characters.

**host.** A host node, host computer, or host system.

**host list.** A list associated with each TeamConnection user ID that indicates the client machine that can access the family server and act on behalf of the user. The family server uses the list to authenticate the identity of a client machine when the family server receives a command. Each entry consists of a login, a host name, and a TeamConnection user ID.

**host name.** The identifier associated with the host computer.

**HPFS.** See *high-performance file system*.

## I

**implicit authority.** The ability to perform an action on a TeamConnection object without being granted explicit authority. This authority is

automatically granted through inheritance or object ownership. Contrast with *base authority* and *explicit authority*.

**import.** To bring in data. In TeamConnection, to bring selected items into a field from a matching TeamConnection object window.

**inheritance.** The passing of configuration management properties from parent to child component. The configuration management properties that are inherited are access and notification. Inheritance within each TeamConnection family or component hierarchy is cumulative.

**integrated problem tracking.** The process of integrating problem tracking with change control to track all reported defects, all proposed features, and all subsequent changes to parts. See also *change control*.

**interest group.** The list of actions that trigger notification to the user IDs associated with those actions listed in the notification list.

## J

**job queue.** A queue of build scopes. One job queue exists for each TeamConnection family.

## L

**local version ID.** In collision records, the database ID of the version of the current work area.

**lock.** An action that prevents editing access to a part stored in the TeamConnection development environment so that only one user can change a part at a time.

**login.** The name that identifies a user on a multi-user system, such as AIX or HP-UX, Solaris, or Windows NT. In OS/2 and Windows 95, the login value is obtained from the TC\_USER environment variable.

## M

**map.** The process of reassigning the meaning of an object.

**metadata.** In databases, data that describe data objects.

## N

**name server.** In TCP/IP, a server program that supplies name-to-address translation by mapping domain names to Internet addresses.

**National Language Support (NLS).** The modification or conversion of a United States English product to conform to the requirements of another language or country. This can include the enabling or retrofitting of a product and the translation of nomenclature, MRI, or documentation of a product.

**Network File System (NFS).** The Network File System is a program that enables you to share files with other computers in networks over a variety of machine types and operating systems.

**notification list.** An object that enables component owners to configure notification. A list attached to a component that pairs a list of user IDs and a list of interest groups. It designates the users and the corresponding notification interest that they are being granted for all objects managed by this component or any of its descendants.

**notification server.** A server that sends notification messages to the client.

**NTFS.** NT file system.

**NVBridge.** A tool for automatic electronic distribution of TeamConnection software deliverables within a NetView DM/2 network.

## O

**operator.** A symbol that represents an operation to be done. See also *comparison operators*.

**originator.** The user who opens a defect or feature and is responsible for verifying the outcome of the defect or feature on a verification record. This responsibility can be reassigned.

**owner.** The user who is responsible for a TeamConnection object within a TeamConnection family, either because the user created the object or was assigned ownership of the object.

## P

**parent component.** All components in each TeamConnection family, except the root component, are created in reference to an existing component. The existing component is the parent component. See also *child component* and *component*.

**parent part.** Any part in a build tree that has a child defined. See also *part* and *child part*.

**parser.** A tool that can read a source file and report back a list of dependencies of that source file. It frees a developer from knowing the dependencies one part has on other parts to ensure a complete build is performed.

**part.** A collection of data that is stored by the family server and retrieved by a path name. They include text objects, binary objects, and modeled objects. These parts can be stored by the user or the tool, or they can be generated from other parts, such as when a linker generates an executable file.

**path name.** The name of the part under TeamConnection control. A path name can be a directory structure and a base name or just a base name. It must be unique within each release. See also *base name*.

**pool.** See *build pool*.

**pop-up menu.** A menu that, when requested, appears next to the object it is associated with.

**prerequisite work areas.** If a part is changed to resolve more than one defect or feature, the work area referenced by the first change is a

prerequisite of the work area referenced by later changes. A work area is a prerequisite to another work area if:

- Part changes are checked in, but not committed, for the first work area.
- One or more of the same parts are checked out, changed, and checked in again for the second work area.

**problem tracking.** The process of tracking all reported defects through to resolution and all proposed features through to implementation.

**process.** A combination of TeamConnection subprocesses, configured by the family administrator, that controls the general movement of TeamConnection objects (defects, features, work areas, and drivers) from state to state within a component or release. See also *subprocess* and *state*.

## Q

**query.** A request for information from a database, for example, a search for all defects that are in the open state. See also *default query* and *search*.

## R

**raw format.** Information retrieved on the report command that has the vertical bar delimiter separating field information, and each line of output corresponds to one database record.

**refresh.** This TeamConnection action updates a work area with any changes from the release, and it also freezes the work area, if it is not already frozen.

**relative path name.** The name of a directory or a part expressed as a sequence of directories followed by a part name, beginning from the current directory.

**release.** A TeamConnection object defined by a user that contains all the parts that must be built, tested, and distributed as a single entity.

**restricted authority.** The limitation on a user's ability to perform certain actions at a specific component. Authority can be restricted by the superuser, the component owner, or a user with AccessRestrict authority. See also *authority*.

**root component.** The initial component that is created when a TeamConnection family is configured. All components in a TeamConnection family are descendants of the root component. Only the root component has no parent component. See also *component*, *child component*, and *parent component*.

## S

**search.** To scan one or more data elements of a set in a database to find elements that have certain properties.

**serial development.** While a user has parts checked out from a work area, no one else on the team can check out the part. The user develops new material without interacting with other developers on the project. TeamConnection provides the opportunity to hold the part until the user is sure that it integrates with the rest of the application. Thus, the lock is not released until the work area as a whole is committed. Contrast with *concurrent development*. See also *work area*.

**server.** A workstation that performs a service for another workstation.

**shadow.** A collection of parts in a filesystem that reflects the contents of a TeamConnection workarea, driver, or release.

**shared part.** A part that is contained in two or more releases.

**shell script.** A series of commands combined in a file that carry out a function when the file is run.

**SID.** The name of a version of a driver, release, or work area.

**sizing record.** A status record created for each component-release pair affected by a proposed defect or feature. The sizing record owner must

indicate whether the defect or feature affects the specified component-release pair and the approximate amount of work needed to resolve the defect or implement the feature within the specified component-release pair.

**stanza format.** Data output generated by the Report command in which each database record is a stanza. Each stanza line consists of a field and its corresponding values.

**state.** Work areas, drivers, features, and defects move through various states during their life cycles. The state of an object determines the actions that can be performed on it. See also *process* and *subprocess*.

**subprocess.** TeamConnection subprocesses govern the state changes for TeamConnection objects. The design, size, review (DSR) and verify subprocesses are configured for component processes. The track, approve, fix, driver, and test subprocesses are configured for release processes. See also *process* and *state*.

**superuser.** This privilege lets a user perform any action available in the TeamConnection family.

**system administrator.** A user who is responsible for all system-related tasks involving the TeamConnection server, such as installing, maintaining, and backing up the TeamConnection server and the database it uses.

## T

**task list.** The list of tasks displayed in the Tasks window. The user can customize this list to issue requests for information from the server. Tasks can be added, modified, or deleted from the lists.

**TCP/IP.** Transmission Control Protocol/Internet Protocol.

**TeamConnection client.** A workstation that connects to the TeamConnection server by a TCP/IP connection and that is running the TeamConnection client software.

**TeamConnection part.** A part that is stored by the TeamConnection server and retrieved by a path name, release, type, and work area. See also *part*, *common part*, and *type*.

**TeamConnection superuser.** See *superuser*.

**tester.** A user responsible for testing the resolution of a defect or the implementation of a feature for a specific driver of a release and recording the results on a test record.

**test record.** A status record used to record the outcome of an environment test performed for a resolved defect or an implemented feature in a specific driver of a release.

**track subprocess.** An attribute of a TeamConnection release process that specifies that the change control process for that release will be integrated with the problem tracking process.

**Transmission Control Protocol/Internet Protocol (TCP/IP).** A set of communications protocols that support peer-to-peer connectivity functions for both local and wide area networks.

**type.** All parts that are created through the TeamConnection GUI or on the command line will show up in reports with the type of TcPart as the part type. The TeamConnection GUI and command line can only check in, check out, and extract parts of the type TcPart.

## U

**user exit.** A user exit allows TeamConnection to call a user-defined program during the processing of TeamConnection transactions. User exits provide a means by which users can specify additional actions that should be performed before completing or proceeding with a TeamConnection action.

**user ID.** The identifier assigned by the system administrator to each TeamConnection user.

## V

**verification record.** A status record that the originator of a defect or a feature must mark before the defect or feature can move to the closed state. Originators use verification records to verify the resolution or implementation of the defect or feature they opened.

**version.** (1) A specific view of a driver, release, or work area. (2) A revision of a part.

**version control.** The storage of multiple versions of a single part along with information about each version.

**view.** An alternative and temporary representation of data from one or more tables.

## W

**work area.** An object in TeamConnection that you create and associate with a release. When the work area is created, you see the most current view of the release and all the parts that it contains. You can check out the parts in the work area, make modifications, and check them back into the work area. You can also test the modifications without integrating them. Other users are not aware of the changes that you make in the work area until you integrate the work area to the release. While you work on files in a work area, you do not see subsequent part changes in the release until you integrate or refresh your work area.

**working part.** The checked-out version of a TeamConnection part.

## Y

**year 2000 ready.** IBM VisualAge TeamConnection Enterprise Server is Year 2000 ready. When used in accordance with its associated documentation, TeamConnection is capable of correctly processing, providing and/or receiving date data within and between the twentieth and twenty-first centuries, provided that all products (for example, hardware, software and

firmware) used with the product properly exchange accurate date data with it.

# Index

## A

- access command
  - creating example 76
- access list 74
- actions
  - Show Authority Actions window 74
  - Show Interest Actions window 80
  - user exit parameters for 191
- Add Host window 69
- Add Notification window 81
- age utility, for defects and features 128
- APP\_CTL\_HEAP\_SZ 32, 144
- APPLHEAPSZ 31
- approval subprocess 55
- audit log
  - cleaning up 136
  - description of 130
  - example of 130
  - information contained in 131
- authorit.ld 166
- authority
  - basic
    - reloading 166
    - verify loading of 166
  - example of granting 70
  - granting to users 74
  - inheritance of 75
  - instructions for granting and restricting 76
  - planning for 70
  - Remove Access window 76
  - restricting 71, 76
  - types of 71
- authority groups
  - Authority Group Settings window 73
  - creating from command line 165
  - creating or changing 72
  - definition of 72
  - display list of 74
  - worksheet 251
- Authority Groups Settings page 73
- authority table 165
- automatic pruning of work areas 52

## B

- backing up the database 137
- base authority 71
- BUFFPAGE 31
- build action 6
- build administrator 23
- build function
  - definition of 11

- build function (*continued*)
  - keeping build output versions 52

## C

- CATALOGCACHE\_SZ 32, 144
- change control 3
- check-in action 6
- check-out action 6
- client
  - definition 5
- command line interface
  - configuring processes 168
  - creating authority groups 165
  - creating interest groups 167
  - starting family server 44
  - starting notification server 45
- commands
  - fhchdf 164
  - fhcirt 163
  - fhclauth 166
  - fhclcnfg 172, 173
  - fhclintr 168
  - fhclproc 170
  - notifyd 45
  - sendmail 42
  - tcadmin 33
  - tcleanu 136
  - tclicmon 150
  - teamcd 44
- component command
  - example of creating 59
- Component Process Settings window 100
- components
  - creating 59
  - definition of 7
  - example of hierarchy 7
  - example of using processes 57
  - information stored about 7
  - list of processes 54
  - list of subprocesses 54
  - naming 50
  - organizing hierarchy 47
  - ownership 49
  - planning 47
  - planning processes for 54
  - processes, configuring 168
- comproc.ld 168
- concepts of
  - TeamConnection 3
- concurrent development
  - difference from serial 52

- config.ld 170
- config table
  - column descriptions 171
  - editing the config.ld file 170
  - modifying 170
  - reloading 172, 173
  - verify loading of 173
- ConfigPartView 85, 93
- configurable field dependencies 87
- configurable processes, worksheet 261
- configuration management 3
- configuring
  - fields
    - changing field types 85
    - changing field values 85
    - creating or changing 89
    - deleting 88, 92
    - displaying properties of 92
  - processes
    - about 99
    - editing the .ld files 168
    - reloading the config table 170
    - using the GUI 100
  - user exits 111
  - worksheet 261
- create action 6
- Create Components window 59
- Create Releases window 60
- Create User window 66

**D**

- daemon
  - number of 41
- database
  - backing up 137
  - controlling size of 52
  - creating with fhcirt 163
  - size restriction 52
- DB2 configuration parameters 31
- DB2 database administration tasks 24
- DB2 database maintenance 127
- DB2 instances 30
- DB2 naming conventions 30
- DBHEAP 31, 32, 144
- default component process 54
- defect.fmt 174
- defects
  - changing age of 128
  - definition of 9
- dependent configurable field 87
- development component process 54
- development mode
  - selecting serial or concurrent 52
- DLCHKTIME 31
- driver configurable field 87

- driver member
  - definition of 56
- driver subprocess 55
  - definition of 55
  - example of 58
  - how to use 58
- drivers
  - definition of 9
- dsrDefect subprocess 54
- dsrFeature subprocess 54

## E

- edit action 6
- emergency\_fix component process 54
- ENV=( ) 180
- environment variables 225
  - setting 231
  - used for trace 136
- error log 129
- errors 129, 130
- examples of
  - audit log 130
  - changing processes 57
  - client/server network 4
  - component hierarchy 7, 48, 50
  - driver subprocess 58
  - granting authority to users 70
  - linking releases 61
  - release-component relationship 51
  - report formats 176
  - showing part/release/component relationship 8
  - stanza report 93
  - table format 96
  - user exit program 107
- explicit authority 71
- extract action 6

## F

- family
  - creating
    - using fhcirt command 163
    - using GUI 32
  - definition of 5
  - planning 29
- family administrator
  - responsibilities 23
  - tasks 163
    - changing report formats 174
    - configuring processes 168
    - creating a family 163
    - creating an initial superuser 164
    - creating or modifying authority groups 165
    - creating or modifying interest groups 167
    - defining configurable field types 170
    - editing authorit.ld 165

- family administrator (*continued*)
  - tasks (*continued*)
    - editing comproc.ld 168
    - editing interest.ld 167
    - editing relproc.ld 168
    - editing userExit 178
    - license monitoring 150
    - reloading configurable process tables 170
    - reloading the authority table 166
    - reloading the config table 172, 173
    - reloading the interest table 167
    - setting up user exits 178
- family server
  - specifying daemons 41
  - starting 42
  - stopping 46
- Family Servers window 145
- Family Settings page 34
- feature.fmt, editing 174
- features
  - changing age of 128
  - definition of 9
- fhcfupdv 173
- fhchdf command 164
- fhcirt command 163
- fhclauth command 166
- fhclcnfg command 172, 173
- fhclintr command 168
- fhclproc command 168, 170
- Field Type window 86
- field types
  - creating or changing 85
- fields
  - configurable 83
    - changing field types 85
    - changing field values 85
    - conditions of 88
    - creating, using GUI 89
    - deleting 88, 92
    - displaying properties of 92
    - modifying, using GUI 89
- files
  - authorit.ld 165
  - comproc.ld 168
  - config.ld 170
  - defect.fmt 174
  - feature.fmt 174
  - interest.ld 167
  - part.fmt 174
  - relproc.ld 168
  - user.fmt 174
  - userExit 178
- fix subprocess 55

## G

- GUI
  - family administrator
    - to add user exit programs 111
    - to change authority groups 72
    - to change configurable field types 85
    - to change configurable processes 100
    - to change interest groups 79
    - to change report formats 94
    - to change table formats 96
    - to create family 32
    - to create or change configurable fields 89
    - to start family server 42
    - to start notification server 42
    - to stop family server 46
    - to stop notification server 46

## H

- hierarchy
  - component example 7, 48
  - component ownership example 50
  - release-component relationship 51
- host command
  - creating example 69
- host lists
  - Add Host window 69
  - creating entries 69
  - planning for 68
- host-only security 36

## I

- implicit authority 71
- Interest Group Settings window 79
- interest groups
  - creating
    - using command line 167
    - using GUI 78
  - definition of 78
  - display list of 80
  - Interest Group Settings window 79
  - worksheet 256
- interest.ld 167
- interest table
  - reloading 167
  - verify loading of 168
- interfaces
  - description of 5

## L

- LANG 225
- license monitoring 150
- LOCKLIST 32, 144
- LOGFILSIZ 31

- login manager
  - global 65
  - individual 65
- LOGPRIMARY 31
- LOGSECOND 31

## M

- mail exit routines 42
- mail facility 36, 40, 42
- maintaining the DB2 database 25
- maintenance component process 54
- manual shadowing 120
- MAXAPPLS 32, 144
- monitor command 147
- monitoring server daemons 145

## N

- naming
  - components 50
  - releases 53
- network 4
- NLSPATH 225
- notification
  - Add Notification window 81
  - adding for users 80
  - instructions for adding 81
  - planning for 77
  - restricting 80
  - setting up mail facility 36, 40, 42
- notification list 80
- notification server
  - starting 42
  - stopping 46
- notify command
  - example of 81
- notifyd command 45

## P

- packaging
  - definition of 11
- part.fmt 174
- parts
  - definition 6
- password-or-host security 36
- passwords 36, 37, 67, 165
- PATH 225
- planning
  - component hierarchy 47
  - for authority to access data 70
  - for host lists 68
  - for notification 77
  - for user access 70
  - for user IDs 50, 63
  - planning 53
- preship component process 54

- processes
  - configuring 99
  - definition of 9
  - example of using 57
  - for components
    - definition of 54
    - shipped 54
    - subprocesses 54
  - for releases
    - definition of 55
    - shipped 56
    - subprocesses 55
  - planning 53
  - worksheet 261
- properties of shadows 118
- prototype component process 54
- pruning 52

## R

- release command
  - creating example 61
- Release Configurable Fields window 90
- release management 3
- release process attributes 56
- releases
  - creating 60
  - creating from an old release 61
  - definition of 7
  - example of linking releases 61
  - example of relationship with other objects 8
  - example of using processes 57
  - list of processes 56
  - list of subprocesses 55
  - naming 53
  - planning 50
  - planning processes for 55
  - processes, configuring 168
  - selecting serial or concurrent development 52
- relproc.ld 168
- Remove Access window 76
- reports
  - changing format of 93
    - editing .fmt files 174
    - using GUI 94, 96
  - description of format sections 175
  - example of format 176
  - table format example 96
- resetAge utility 129
- return codes
  - from user exit program 104
- root component
  - owner 49

## S

- sample files shipped
  - mail exit routines 42
- security 37, 67, 165
- sendmail command 42
- serial development
  - difference from concurrent 52
- server daemon monitor 145
- servers
  - definition 5
  - family server
    - definition 5
    - specifying daemons 41
    - starting 42
  - notification server 42
- Settings notebook
  - Family Properties notebook 34
- shadow actions 120
- shadow types 117
- shadows 117
- Show Authority Actions window 74
- Show Interest Actions window 80
- stanza report
  - changing format of 93
  - example of 93
- Stanza View Format Settings page 94
- starting
  - family server 42
  - notification server 42
- subprocesses
  - for components 54
  - for releases 55
  - using driver subprocess 58
- superuser
  - creating others 67
  - creating using fhchdf 164
  - definition of 5, 71
- synchronous shadowing 120
- system administrator, responsibilities 23

## T

- table report
  - changing format of 96
  - displaying 96
  - example of 96
- Table View Format Settings page 96
- TargetView 85, 93
- tasks
  - family administrator 40
- TC\_BECOME 225
  - setting for superuser access 67
- TC\_BUILDPOOL 225
- TC\_CASESENSE 225
- TC\_COMPONENT 225

- TC\_DBPATH 45, 164, 225
- TC\_FAMILY 225, 231
- TC\_MAKEIMPORTRULES 225
- TC\_MAKEIMPORTTOP 225
- TC\_MAKEIMPORTVERBOSE 225
- TC\_MIGRATERULES 225
- TC\_NOTIFY\_DAEMON 44, 45, 225
- TC\_RELEASE 225
- TC\_TOP 225
- TC\_TRACE 225
- TC\_TRACEFILE 225
- TC\_TRACESIZE 225
- TC\_USER 225
- TC\_WORKAREA 225
- tcadmin command 33
- tccleanu command 136
- tclicmon command 150
- TCP/IP
  - sendmail command 42
- tcqry 159
- tcupdb 160
- teamcd command 44
- TeamConnection
  - concepts of 3
  - introducing 3
- test component process 54
- test subprocess 56
- trace 136
- track subprocess 55
- trackcommithold 56
- trackfixhold 56
- tracktesthold 56

## U

- user command
  - creating example 67
- User Exit parameters settings page 114
- User Exit Settings page 111
- user exits
  - configuring 111
  - customizing parameters for 180
  - editing userExit file 178
  - example of 107
  - nonzero return codes 104
  - parameters of 191
  - tips for writing 104
- user.fmt 174
- user IDs
  - creating 66
  - initial superuser 164
  - planning for 63
- user management wizard 66
- userExit file 103
- users
  - notification 77

users (*continued*)  
  preparing for 63  
  to have data access 70

## V

verifyDefect subprocess 54  
verifyFeature subprocess 54  
version control 3

## W

window examples  
  Add Host 69  
  Add Notification 81  
  Authority Group Settings window 73  
  Component Process Settings 100  
  Configurable Fields for Defects Settings 90  
  Configurable Fields Settings 86  
  Create Components 59  
  Create Releases 60  
  Create User 66  
  Family Settings 34  
  Interest Groups 79  
  Remove Access 76  
  Show Authority Actions 74  
  Show Interest Actions 80  
  Stanza View Format Settings 94  
  Table View Format Settings 96  
  User Exit parameters settings page 114  
  User Exit Settings 111  
wizard, user management 66  
work area  
  automatic pruning of 52  
  definition of 8  
  things you can do with 8

---

# Readers' Comments — We'd Like to Hear from You

**IBM VisualAge TeamConnection Enterprise Server  
Administrator's Guide  
Version 3.0**

**Publication No. SC34-4551-01**

**Overall, how satisfied are you with the information in this book?**

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	<input type="checkbox"/>				

**How satisfied are you that the information in this book is:**

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	<input type="checkbox"/>				
Complete	<input type="checkbox"/>				
Easy to find	<input type="checkbox"/>				
Easy to understand	<input type="checkbox"/>				
Well organized	<input type="checkbox"/>				
Applicable to your tasks	<input type="checkbox"/>				

**Please tell us how we can improve this book:**

Thank you for your responses. May we contact you?  Yes  No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

\_\_\_\_\_

Name

\_\_\_\_\_

Address

\_\_\_\_\_

Company or Organization

\_\_\_\_\_

Phone No.



Cut or Fold  
Along Line

Fold and Tape

Please do not staple

Fold and Tape



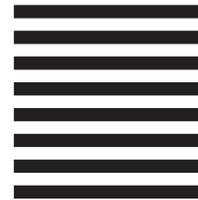
NO POSTAGE  
NECESSARY  
IF MAILED IN THE  
UNITED STATES

# BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation  
Information Development  
Department G71A / Bldg 062  
P.O. Box 12195  
Research Triangle Park, NC  
27709-2195



Fold and Tape

Please do not staple

Fold and Tape

Cut or Fold  
Along Line





Part Number: 30L9312  
Program Number: 5622-717



Printed in the United States of America  
on recycled paper containing 10%  
recovered post-consumer fiber.

SC34-4551-01



30L9312

