

TeamConnection



Getting Started with TeamConnection Clients

Version 2.0

TeamConnection



Getting Started with TeamConnection Clients

Version 2.0

Fourth Edition (December 1997)

Note

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

This edition applies to Version 2.0 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, 1996, 1997. 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

Notices	vii
Trademarks	ix
About this book.	xi
How this book is organized	xi
Conventions	xi
Tell us what you think	xii
Chapter 1. An introduction to TeamConnection	1
TeamConnection definitions	2
TeamConnection's client/server architecture	2
TeamConnection database	3
Interfaces	3
Families	3
Users and host lists	3
Parts	4
Components	4
Releases	5
Work areas	6
Drivers	7
Defects and features	7
Processes	7
Build	8
Packaging	9
Roles people play	9
Chapter 2. Installing the TeamConnection for AIX client	11
Hardware requirements	11
Software requirements	11
Preparing to install TeamConnection for AIX	12
Preparing the SYSLOG file	14
Installing from a CD-ROM	15
Adding and mounting a CD-ROM file system	15
Important installation information about TeamConnection (all platforms)	16
Configure the environment variables in the .profile	16
Ensure that the TeamConnection client command is accessible	17
Use the TeamConnection Client	17
How to use the mnemonics in the TeamConnection GUI for UNIX:	18
Chapter 3. Installing the TeamConnection for HP-UX client	19
Hardware requirements	19
Software requirements	19
Preparing to install TeamConnection for HP-UX	19
Preparing the SYSLOG file	21
Installing from a CD-ROM	22
Adding and mounting a CD-ROM file system	23
Important installation information about TeamConnection (all platforms)	24
Configure the environment variables in the .profile	24
Ensure that the TeamConnection client command is accessible	24
Use the TeamConnection Client	25
How to use the mnemonics in the TeamConnection GUI for UNIX:	25

Chapter 4. Installing the TeamConnection for Solaris client.	27
Hardware requirements	27
Software requirements.	27
Preparing to install TeamConnection for Solaris	27
Preparing the SYSLOG file	29
Installing from a CD-ROM	30
Adding and mounting a CD-ROM file system	31
Important installation information about TeamConnection (all platforms).	32
Configure the environment variables in the .profile	32
Ensure that the TeamConnection client command is accessible.	32
Use the TeamConnection Client	33
How to use the mnemonics in the TeamConnection GUI for UNIX:	33
 Chapter 5. Installing the TeamConnection client for OS/2	35
Hardware requirements	35
Software requirements.	35
Preparing to install TeamConnection for OS/2	36
Installing from a LAN	39
Installing using the installation GUI	39
 Chapter 6. Installing the TeamConnection clients for Windows	43
Hardware requirements	43
Preparing to install TeamConnection for Windows.	44
Installing using the installation GUI	48
 Chapter 7. Getting familiar with the TeamConnection client interfaces	51
Using the GUI	51
Starting the GUI	51
Stopping the GUI.	52
Performing tasks with the GUI	52
Using the Settings notebook	53
Online help information	54
Using the command line interface	55
Using the web client	57
 Chapter 8. The basics of using TeamConnection	59
Laying the groundwork.	59
Authority to perform tasks	60
Finding objects within TeamConnection	61
Finding parts	61
Using work areas	62
Naming your work areas	62
Creating parts	63
Naming your parts	63
Preparing to build your parts	63
Working with parts	64
Working in serial or concurrent development mode	64
Working with common parts.	64
Getting parts from TeamConnection	65
Checking parts in to TeamConnection	66
Finding different versions of TeamConnection objects	67
Versioning releases	67
Versioning work areas	68
Versioning drivers	69
Versioning parts	70
Working with defects and features	70

Testing and verifying part changes	71
Chapter 9. The states of TeamConnection objects	73
Defects and features	73
The states of work areas	76
The states of drivers	78
Verification and test records.	80
Customer support	83
Bibliography	85
IBM VisualAge TeamConnection library	85
Tool Builder's Development Kit.	85
TeamConnection Technical reports	86
ObjectStore.	86
IBM Exchange library	87
Related publications	87
Glossary	89
Index	97

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 the IBM Corporation in the United States or other countries or both:

AIX	MVS/XA
BookManager	NetView
Common User Access	Operating System/2
C Set++	OS/2
CUA	TeamConnection
C/370	VisualAge
IBM	VisualAge Generator
DB/2	XGA
MVS/ESA	

The following terms are trademarks of other companies:

ObjectStore

ObjectStore Design, Inc.

Microsoft, Windows, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

Solaris and the Sun logo are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

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

About this book

This book is intended for TeamConnection client users. It explains the following:

- What TeamConnection is
- The hardware and software you need for it
- How to install the client
- How to get around in the client's graphical user interface (GUI) or commands

This book contains information about all the TeamConnection clients. If information is specific to a client, that information is either contained in its own section or chapter or preceded by a client-specific icon.

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 on the installation CD in directory path `softpubs\enu` (`softpubs/en_us` in UNIX). To view these files, you need a PDF reader such as Acrobat.

How this book is organized

This book contains the following information:

- An introduction to TeamConnection
- Installing the TeamConnection client for AIX
- Installing the TeamConnection client for HP
- Installing the TeamConnection client for Solaris
- Installing the TeamConnection client for OS/2
- Installing the TeamConnection clients for Windows
- Getting familiar with the TeamConnection client interfaces
- The basics of using TeamConnection
- The states of TeamConnection objects

Information on customer service, a glossary, and a bibliography 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 and HP-UX 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 for OS/2



IBM VisualAge TeamConnection for Windows 3.1



IBM VisualAge TeamConnection for Windows/NT



IBM VisualAge TeamConnection for Windows 95



IBM VisualAge TeamConnection for AIX



IBM VisualAge TeamConnection for HP-UX



IBM VisualAge TeamConnection 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.

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, and MVS.
- *Packaging support*: the preparation of your application for electronic distribution to other users.

TeamConnection provides an open information model for sharing data between a set of integrated tools using TeamConnection. This object-based information model enables an extensible architecture, thus ensuring continued support for new versions of existing tools, as well as new tools that are brought into the repository environment. This support includes the previously mentioned standard services across all objects stored in the information model. TeamConnection's information model and tool builder's functions are provided separately in the Tool Builder's Development Kit. See the bibliography at the back of this book for more information.

IBM Exchange for OS/2, which is a feature of TeamConnection, provides the functions necessary to migrate existing model information into TeamConnection. For more information about IBM Exchange, refer to the *IBM Exchange User's Guide*

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:

- “Chapter 5. Installing the TeamConnection client for OS/2” on page 35
- “Chapter 7. Getting familiar with the TeamConnection client interfaces” on page 51
- “Chapter 8. The basics of using TeamConnection” on page 59
- “Chapter 9. The states of TeamConnection objects” on page 73

TeamConnection definitions

The following definitions are in logical order rather than alphabetical. provides additional information about these terms.

TeamConnection's client/server architecture

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

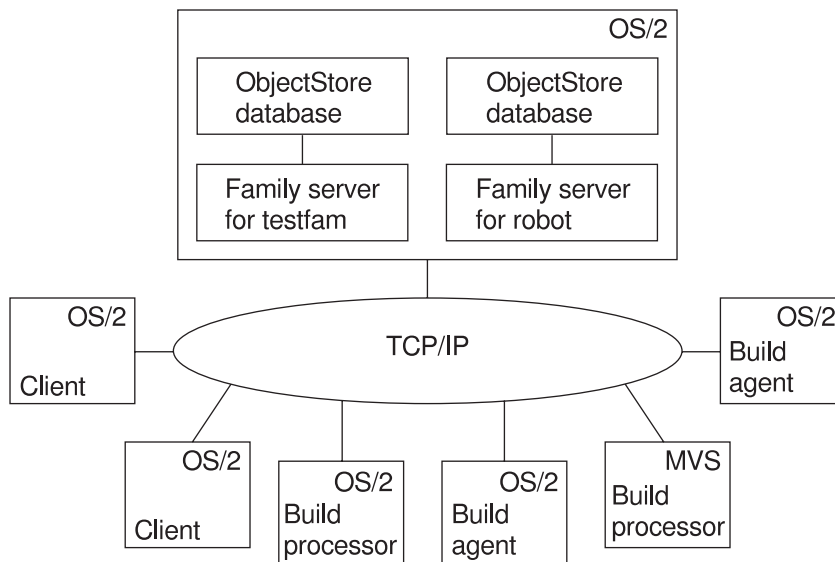


Figure 1. A sample TeamConnection client/server network

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

Build agents and *build processors* work together to perform product builds. We sometimes refer to the combination of a build agent and its connected build processor as a build server.

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

TeamConnection database

TeamConnection is built on Object Design, Inc.'s ObjectStore database. The TeamConnection server communicates with the ObjectStore database through an ObjectStore server.

Interfaces

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

- A graphical user interface based on Common User Access (CUA).
- 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.

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.

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 “Chapter 8. The basics of using TeamConnection” on page 59 .

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.

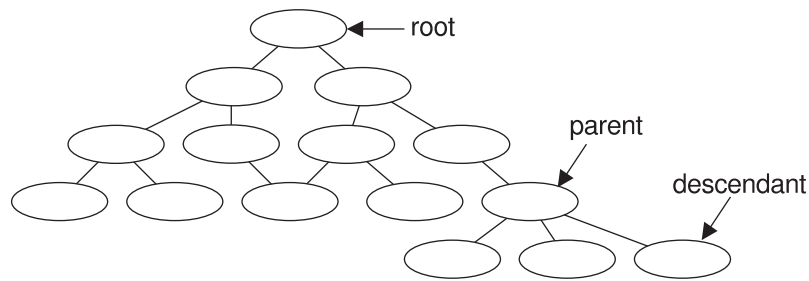


Figure 2. Sample of a component hierarchy

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.

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 on page 6 shows the relationships between parts, the releases that contain them, and the components that manage them.

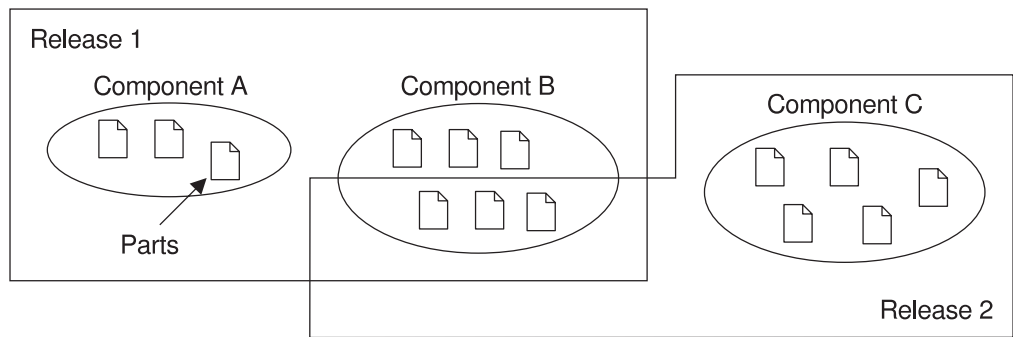


Figure 3. Parts, releases, and components

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.

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 are 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 "Chapter 9. The states of TeamConnection objects" on page 73 for an explanation of TeamConnection states.

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 dependants. 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

NVBridge

A bridge utility that automates the installation and distribution of software or data using IBM NetView Distribution Manager/2 as the distribution vehicle

NVBridge

A tool that supports automated distribution between a single NetView DM/2 CC server and its LAN-connected CC clients. It also supports remote distribution to APPC-connected NetView DM/2 servers and mainstream servers.

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

Roles people play

Because TeamConnection is extremely flexible, no two projects are likely to use it in the same way, and the jobs that people perform likewise vary. Still, TeamConnection tasks can be grouped into the following general categories:

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 server
- Maintaining and backing up the database used by TeamConnection

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

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 agents and processors
- Defining *pools*
- Monitoring build performance
- Creating driver members
- Committing and completing drivers
- Extracting releases
- Packaging and distributing applications

End user

End users, such as project leaders, programmers, and technical writers, use one or more TeamConnection families to control and maintain application development data.

Chapter 2. Installing the TeamConnection for AIX client

This chapter lists the hardware and software that are required before you can install and use the TeamConnection for AIX client. It also describes what you must do before installation, and provides detailed instructions for installing and configuring the TeamConnection client.

For more information about late breaking news about TeamConnection see the file:

`$TC_HOME/install/readme.txt`

Where `$TC_HOME` is the location where the TeamConnection code was installed.

Hardware requirements

The following are hardware requirements for the AIX client machine:

Table 1. Hardware Requirements for an AIX TeamConnection 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 processor.
Memory	32 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.

Software requirements

The following are the software requirements for the AIX client machine:

Table 2. Software Requirements for TeamConnection for AIX

Server	AIX version 4.2.1 (part number 5765-393) or a later release of version 4 that includes TCP/IP, and the xIC runtime
Client	<ul style="list-style-type: none">• AIX version 4.2.1 (part number 5765-393) or a later release of version 4 that includes TCP/IP• A Web browser such as Netscape Navigator to display the help panels for the GUI
Build server	AIX version 4.2.1 (part number 5765-393) or a later release of version 4 that includes TCP/IP, and the xIC runtime

Preparing to install TeamConnection for AIX

This section explains what you need to do before installing TeamConnection for AIX. Use the `smit` utility to create user accounts, to update the hosts and services files, and to add and mount a file system. The `smit` utility starts the System Management Interface Tool (`smit`). This tool presents a menu-driven environment for system administration tasks. For more information on `smit`, refer to your AIX operating system documentation.

It is assumed that TCP/IP for your operating system has already been installed on your client machine and that someone in your organization has already installed the TeamConnection family server.

Before you install the TeamConnection client code, ask your family administrator to provide you with the following information:

- IP address of the family server
- Family server name
- Name of the family you are to use
- Port address for your family

Ask your family administrator if you need to update your TCP/IP hosts and services files. If you do, follow the steps below:

You can update these files using SMIT. In many installations, a name server is used instead of `/etc/hosts`. Also, many installations distribute `/etc/services`. SMIT can be used to make the necessary updates.

1. Update the services file, which is located in `/etc/services`.

Include the family name and port address of the TeamConnection server.

The port address can be any 4-digit number, as long as it does not already exist in your services file. You might want to ask your TCP/IP administrator to assign you a number.

Type the following entries in your services file:

```
# TeamConnection servers
testfam ffff/tcp # port address for TeamConnection test family
bldsock bbbb/tcp # port address for build agent and processor
```

Note:

- a. For this installation, replace `ffff` and `bbbb` in the above example with appropriate port addresses.
 - b. Follow each line with a carriage return.
2. Update the TCP/IP hosts file, which is located in `/etc/hosts`.
Add the following:
 - IP address.
 - Server name.
 - Alias name of the TeamConnection family server, which is your family name. For this initial installation of TeamConnection, the family name is `testfam`.
 - Alias name for the build socket. For this initial installation of TeamConnection, use `bldsock`.

The following is an example of the entry you would type in your hosts file:

```
9.12.345.67    teamserv.company.com    testfam    bldsock
```

Note: Follow the line with a carriage return. You can use the hostname command to get the name of the server.

3. Do the following to verify that the hosts file is specified correctly:
 - Type "host family_name", where family_name is the name of your TeamConnection family. The information returned should match the number and name specified in your hosts file entry. For example, using the entry given in the previous step, the system response would be as follows:

```
teamserv.company.com = 9.12.345.67
```

- Type "host ip_address", where ip_address is the IP address of your machine. The information returned should match the number and name specified in your hosts file entry. For example, again using the entry given in the previous step, the system response would be as follows:

```
9.12.345.67 = teamserv.company.com
```

If you do not receive the expected response, contact your TCP/IP administrator to solve the problem.

Note: If the changes are done to the domain name server, then you can use the UNIX utility "nslookup" instead.

4. Do the following to verify that you can connect to your TeamConnection family:
At a prompt, type "ping testfam".

If you receive information that is similar to the following, you can successfully connect to your TeamConnection family:

```
PING teamserv.company.com: 56 data bytes
64 bytes from 1.23.457.78: icmp_seg:0. time=0. ms
64 bytes from 1.23.456.78: icmp_seg:1. time=0. ms
64 bytes from 1.23.456.78: icmp_seg:2. time=0. ms
```

5. Press Ctrl+C to end the command.

If you receive the message "unknown host testfam", you cannot connect to the family. Verify that the data you entered in the hosts and services files is correct, and then try the command again. If you still do not get the correct response, contact your TCP/IP administrator to solve the problem.

6. Do the following to verify that you can connect to your TeamConnection build server:
 - a. At a prompt, type "ping bldsock".
 - b. Press Ctrl+C to end the command.

If you receive information that is similar to the following, you can successfully connect to your TeamConnection build server:

```
PING teamserv.company.com: 56 data bytes
64 bytes from 1.23.457.78: icmp_seg:0. time=0. ms
64 bytes from 1.23.456.78: icmp_seg:1. time=0. ms
64 bytes from 1.23.456.78: icmp_seg:2. time=0. ms
```

If you receive the message "unknown host bldsock", you cannot connect to the build server. Verify that the data you entered in the hosts and services files is correct, and then try the command again. If you still do not get the correct response, contact your TCP/IP administrator to solve the problem.

Do not install the TeamConnection components until the ping commands successfully complete.

Preparing the SYSLOG file

TeamConnection and ObjectStore database errors are recorded in the syslog file. The information recorded in this file will help you resolve problems when they occur.

Before TeamConnection can record these errors, you must activate the syslog daemon. Do the following to activate the syslog daemon:

1. The syslog file is located in `/var/spool/syslog`. If the log file does not exist, type the following touch command to create it. When you create the log file, follow the directions for setting permissions for your operating system.

```
touch          /var/spool/syslog
chmod 666      /var/spool/syslog
chown root:system /var/spool/syslog
```

If you don't want to use `chmod 666`, you can use

```
"chmod a=rw  /var/spool/syslog" or
"chmod ugo=rw /var/spool/syslog".
```

The command `"chown root:system"` is a short way to do the following:

```
chown root  /var/spool/syslog
chgrp system /var/spool/syslog
```

2. Add the following lines to the `/etc/syslog.conf` file: Use a tab to separate the items.

```
*.warning      /var/spool/syslog
```

3. Type the following to stop and then restart the syslog daemon:

```
stopscr -s syslogd
startsrc -s syslogd
```

4. Type the following command to verify that the syslog daemon is running:

```
ps -ef | grep syslog
```

If the daemon is running, you will see one process for `syslogd`.

5. Do the following to verify that the syslog daemon is able to write to the syslog file:

- a. Log in as another user ID.
- b. Type `"su root"` and an incorrect password. This action will fail and a message will be logged in the syslog.

To quickly view the last 10 lines of the syslog, do:

```
tail <syslog-file-name>
***** => use the appropriate name
```

If the syslog file is configured properly, it will contain a message similar to the following:

```
Apr 19 10:21:45 hostname su: BAD SU from userid to root at /dev/pts/3
```

6. If you want to cleanup the syslog, you can use the following command:

```
cp /dev/null <syslog-file-name>
***** => use the appropriate name
```

7. Monitor the syslog file at regular intervals so that any required maintenance or problem resolution can be performed.

Installing from a CD-ROM

The target audience for this section is the system administrator who can login as user root and who will do the initial installation of TeamConnection for your organization.

The time required for installing this program varies depending on your workstation. In general, installation of the product takes about 30 minutes.

Note: Installing TeamConnection version 2 causes any previous version of all of the components of TeamConnection to be overwritten without any warning messages. You will not be able to go back to the version of TeamConnection that was previously installed without completely reinstalling that version.

If you already have an existing version of TeamConnection installed, ensure that none of the TeamConnection daemons and tools are running.

Adding and mounting a CD-ROM file system

QUICK START FOR EXPERTS:

- Mount the CD-ROM as /cdrom.
- Use the /cdrom/tcinst.ksh command to install the TeamConnection components you want.

To install the TeamConnection code from a CD-ROM, you must first add and mount a CD-ROM file system. Follow the instructions in this section to complete this task.

Do the following to add and mount a CD-ROM file system:

1. Insert the CD-ROM into the CD-ROM drive.
2. Log in as user root or do "su - root" to login as root and use the profile.
3. Create a directory for the CD-ROM /cdrom by entering "mkdir /cdrom"
4. Enter "smit" to add a CD-ROM file system.
5. Select "System Storage Management (Physical & Logical Storage)"
6. Select "File Systems"
7. Select "Add / Change / Show / Delete File Systems."
8. Select "CDROM File Systems"
9. Select "Add a CDROM File System"
10. Select a "Device name", such as "cd0". the CD-ROM file system device names must be unique. you may need to delete a previously defined CD-ROM file system if there is a conflict.
11. Type "/cdrom" for "Mount Point".
12. Select "OK" (or press enter if using the smit ascii interface).
13. Return to the previous smit level, "System Storage Management (Physical & Logical Storage)"
14. Select "File Systems".
15. Select "Mount a File System".
16. Select "/dev/cd0" for "File System name".
17. Select "/cdrom" for "Directory over which to mount".
18. Select "cdrfs" for "Type of file system".

19. Select "yes" for "Mount as a READ-ONLY system."
20. Select "OK" (or press enter if using the smit ascii interface).
21. Exit smit.

Now you are ready to install the appropriate TeamConnection components.

After you have added and mounted a CD-ROM, do the following to install the TeamConnection components:

1. Log in with root authority.
2. Change the directory to /cdrom.
3. Run the installation script by typing `./tcinst.ksh`
4. Follow the instructions in the installation script to install the TeamConnection client.
5. After the installation is complete, type the following commands:

```
cd ..
umount /cdrom
```
6. Eject the CD.

Important installation information about TeamConnection (all platforms)

After you install the TeamConnection code in its directory structure (represented by \$TC_HOME), please keep in mind the following guidelines:

- Do not remove directories, files or symbolic links from \$TC_HOME. Exception: To uninstall the product, you need to remove \$TC_HOME.
- Do not change the file permissions nor the ownership of the directories and/or files in \$TC_HOME.
- Use \$TC_HOME only to store the TeamConnection code:
 - Do not use \$TC_HOME as the home directory for users.
 - Do not use \$TC_HOME as the home directory for TeamConnection families.

Furthermore, if this is your first installation of TeamConnection it is recommended to accept the defaults in this installation script.

Configure the environment variables in the .profile

It is necessary to configure the environment variables in the .profile of the user account before using the TeamConnection client.

1. Log in as a user that will use TeamConnection.
2. To set environment variables for TeamConnection you can use the profile.user as a sample .profile:

```
mv $HOME/.profile $HOME/.profile.original
cp $TC_HOME/install/<language>/profile.user $HOME/.profile
```

Where \$TC_HOME is the location where the TeamConnection code was installed.

3. Customize all the environment variables in the .profile file.
Edit the new .profile and read the entire profile and modify the values as appropriate and uncomment the needed variables that depend on your operating system.

Note: If you are using the CDE environment, edit .dtprofile to enable the account to run .profile after .dtprofile. CDE stands for Common Desktop Environment and is the default for AIX 4 as well as many other versions of UNIX. CDE is the standard of the Open Group (c). For details on how to do it, see the header of the sample profile.

4. Exit the new user ID and login again to refresh the environment variables.

Ensure that the TeamConnection client command is accessible

To ensure that the TeamConnection client is accessible do the following:

1. Type the following TeamConnection command which does not require to connect yet to a working TeamConnection family:

```
teamc report -testClient
```

2. If the teamc command is not found, then ensure that the TC_HOME and PATH variables are properly set in the .profile.
3. If the teamc command is found, then verify that the top portion of the output is something like this, assuming that the LANG variable is en_US:

```
Connect to Family Name:      testfam
Server TCP/IP Name:         hostname.company.com
Server IP Address:          9.87.654.32
Server TCP/IP Port Number:  9000
```

```
Server Specific Information -----
Product Version:            3.0.0
Operating System:           AIX
Message catalog language:   English
Server Mode:                non-maintenance
Authentication Level:       PASSWORD_OR_HOST
TC_BECOME:                  your user ID
TC_FAMILY:                  your family name
```

- 4.

If you see that the entry for "Message catalog language" has a value of:

```
English (Internal)
```

Then, it means that the LANG or the NLSPATH variables are not properly set and the TeamConnection client is using its internal messages. It is strongly recommended that the external message catalog should be used.

Use the TeamConnection Client

You can run the TeamConnection line commands from the family account. For example, to see the list of all the users:

```
teamc report -view users -where 1=1
```

To run the TeamConnection GUI, you need to do the following:

1. Copy the sample initial tasks list for the main GUI window. This needs to be done only once.

```
cp $TC_HOME/nls/cfg/$LANG/teamc20.ini $HOME
chmod u+w $HOME/teamc20.ini
```

Where \$TC_HOME is the location where the TeamConnection code was installed.

2. Invoke the TeamConnection GUI by entering:

How to use the mnemonics in the TeamConnection GUI for UNIX:

To access the menu for an entry in the menu bar, use:

`Alt+MnemonicLetter` (such as `Alt+f` to display the File menu)

Even though some mnemonics will be shown in Uppercase, such as "F" for File, the lowercase letter is used. If you try to use the Shift operator with the Alt function, then it will not work:

`Alt+Shift+MnemonicLetter` (it will not work)

Once a menu is displayed, you can use either of the following:

`Action+MnemonicLetter` (such as `Action+r` for "Refresh now" in the File menu)

`Action+Shift+MnemonicLetter` (such as `Action+Shift+r` for "Refresh now" in the File menu)

You can use either lowercase or uppercase when using the Action key, even though some mnemonics will be shown in Uppercase in the menus (such as "R" for "Refresh now" in the File menu).

Chapter 3. Installing the TeamConnection for HP-UX client

This chapter lists the hardware and software that are required before you can install and use the TeamConnection for HP-UX client. It also describes what you must do before installation, and provides detailed instructions for installing and configuring the TeamConnection client.

For more information about late breaking news about TeamConnection see the file:

`$TC_HOME/install/readme.txt`

Where `$TC_HOME` is the location where the TeamConnection code was installed.

Hardware requirements

The following are hardware requirements for the HP-UX client machine:

Table 3. Hardware Requirements for an HP TeamConnection 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	32 MB memory minimum.
Disk space	500 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.

Software requirements

The following are the software requirements for the HP-UX client machine:

Table 4. Software Requirements for TeamConnection for HP

Server	<ul style="list-style-type: none">• HP-UX version 10.01, which includes TCP/IP• C++ runtime version B.004 or later
Client	HP-UX version 10.01, which includes TCP/IP
Build server	<ul style="list-style-type: none">• HP-UX version 10.01, which includes TCP/IP• C++ runtime version B.004 or later

Preparing to install TeamConnection for HP-UX

This section explains what you need to do before installing TeamConnection for HP-UX. Use the `smit` utility to create user accounts, to update the `hosts` and `services` files, and to add and mount a file system. The `smit` utility starts the System Management Interface Tool (`smit`). This tool presents a menu-driven environment for system administration tasks. For more information on `smit`, refer to your HP-UX operating system documentation.

It is assumed that TCP/IP for your operating system has already been installed on your client machine and that someone in your organization has already installed the TeamConnection family server.

Before you install the TeamConnection client code, ask your family administrator to provide you with the following information:

- IP address of the family server
- Family server name
- Name of the family you are to use
- Port address for your family

Ask your family administrator if you need to update your TCP/IP hosts and services files. If you do, follow the steps below:

You can update these files using SMIT. In many installations, a name server is used instead of /etc/hosts. Also, many installations distribute /etc/services. SMIT can be used to make the necessary updates.

1. Update the services file, which is located in /etc/services.

Include the family name and port address of the TeamConnection server.

The port address can be any 4-digit number, as long as it does not already exist in your services file. You might want to ask your TCP/IP administrator to assign you a number.

Type the following entries in your services file:

```
# TeamConnection servers
testfam ffff/tcp # port address for TeamConnection test family
bldsock bbbb/tcp # port address for build agent and processor
```

Note:

- a. For this installation, replace ffff and bbbb in the above example with appropriate port addresses.
 - b. Follow each line with a carriage return.
2. Update the TCP/IP hosts file, which is located in /etc/hosts.

Add the following:

- IP address.
- Server name.
- Alias name of the TeamConnection family server, which is your family name. For this initial installation of TeamConnection, the family name is testfam.
- Alias name for the build socket. For this initial installation of TeamConnection, use bldsock.

The following is an example of the entry you would type in your hosts file:

```
9.12.345.67 teamserv.company.com testfam bldsock
```

Note: Follow the line with a carriage return. You can use the hostname command to get the name of the server.

3. Do the following to verify that the hosts file is specified correctly:
 - Type "host family_name", where family_name is the name of your TeamConnection family. The information returned should match the number and name specified in your hosts file entry. For example, using the entry given in the previous step, the system response would be as follows:

```
teamserv.company.com = 9.12.345.67
```

- Type "host ip_address", where ip_address is the IP address of your machine. The information returned should match the number and name specified in your hosts file entry. For example, again using the entry given in the previous step, the system response would be as follows:

```
9.12.345.67 = teamserv.company.com
```

If you do not receive the expected response, contact your TCP/IP administrator to solve the problem.

Note: If the changes are done to the domain name server, then you can use the UNIX utility "nslookup" instead.

4. Do the following to verify that you can connect to your TeamConnection family:
At a prompt, type "ping testfam".

If you receive information that is similar to the following, you can successfully connect to your TeamConnection family:

```
PING teamserv.company.com: 56 data bytes
64 bytes from 1.23.457.78: icmp_seg:0. time=0. ms
64 bytes from 1.23.456.78: icmp_seg:1. time=0. ms
64 bytes from 1.23.456.78: icmp_seg:2. time=0. ms
```

5. Press Ctrl+C to end the command.

If you receive the message "unknown host testfam", you cannot connect to the family. Verify that the data you entered in the hosts and services files is correct, and then try the command again. If you still do not get the correct response, contact your TCP/IP administrator to solve the problem.

6. Do the following to verify that you can connect to your TeamConnection build server:
 - a. At a prompt, type "ping bldsock".
 - b. Press Ctrl+C to end the command.

If you receive information that is similar to the following, you can successfully connect to your TeamConnection build server:

```
PING teamserv.company.com: 56 data bytes
64 bytes from 1.23.457.78: icmp_seg:0. time=0. ms
64 bytes from 1.23.456.78: icmp_seg:1. time=0. ms
64 bytes from 1.23.456.78: icmp_seg:2. time=0. ms
```

If you receive the message "unknown host bldsock", you cannot connect to the build server. Verify that the data you entered in the hosts and services files is correct, and then try the command again. If you still do not get the correct response, contact your TCP/IP administrator to solve the problem.

Do not install the TeamConnection components until the ping commands successfully complete.

Preparing the SYSLOG file

TeamConnection and ObjectStore database errors are recorded in the syslog file. The information recorded in this file will help you resolve problems when they occur.

Before TeamConnection can record these errors, you must activate the syslog daemon. Do the following to activate the syslog daemon:

1. The syslog file is located in /var/spool/syslog. If the log file does not exist, type the following touch command to create it. When you create the log file, follow the directions for setting permissions for your operating system.

```
touch          /var/spool/syslog
chmod 666      /var/spool/syslog
chown root:system /var/spool/syslog
```

If you don't want to use `chmod 666`, you can use

```
"chmod a=rw /var/spool/syslog" or
"chmod ugo=rw /var/spool/syslog".
```

The command `"chown root:system"` is a short way to do the following:

```
chown root /var/spool/syslog
chgrp system /var/spool/syslog
```

2. Add the following lines to the `/etc/syslog.conf` file: Use a tab to separate the items.

```
*.warning      /var/spool/syslog
```

3. Type the following to stop and then restart the syslog daemon:

```
stopscr -s syslogd
startsrc -s syslogd
```

4. Type the following command to verify that the syslog daemon is running:

```
ps -ef | grep syslog
```

If the daemon is running, you will see one process for `syslogd`.

5. Do the following to verify that the syslog daemon is able to write to the syslog file:

- a. Log in as another user ID.
- b. Type `"su root"` and an incorrect password. This action will fail and a message will be logged in the syslog.

To quickly view the last 10 lines of the syslog, do:

```
tail <syslog-file-name>
***** => use the appropriate name
```

If the syslog file is configured properly, it will contain a message similar to the following:

```
Apr 19 10:21:45 hostname su: BAD SU from userid to root at /dev/pts/3
```

6. If you want to cleanup the syslog, you can use the following command:

```
cp /dev/null <syslog-file-name>
***** => use the appropriate name
```

7. Monitor the syslog file at regular intervals so that any required maintenance or problem resolution can be performed.

Installing from a CD-ROM

The target audience for this section is the system administrator who can login as user `root` and who will do the initial installation of TeamConnection for your organization.

The time required for installing this program varies depending on your workstation. In general, installation of the product takes about 30 minutes.

Note: Installing TeamConnection version 2 causes any previous version of all of the components of TeamConnection to be overwritten without any warning messages. You will not be able to go back to the version of TeamConnection that was previously installed without completely reinstalling that version.

If you already have an existing version of TeamConnection installed, ensure that none of the TeamConnection daemons and tools are running.

Adding and mounting a CD-ROM file system

QUICK START FOR EXPERTS:

- Mount the CD-ROM as /cdrom.
- Use the /cdrom/tcinst.ksh command to install the TeamConnection components you want.

To install the TeamConnection code from a CD-ROM, you must first add and mount a CD-ROM file system. Follow the instructions in this section to complete this task.

Do the following to add and mount a CD-ROM file system:

1. Insert the CD-ROM into the CD-ROM drive.
2. Log in as user root or do "su - root" to login as root and use the profile.
3. Create a directory for the CD-ROM /cdrom by entering "mkdir /cdrom"
4. Enter "smit" to add a CD-ROM file system.
5. Select "System Storage Management (Physical & Logical Storage)"
6. Select "File Systems"
7. Select "Add / Change / Show / Delete File Systems."
8. Select "CDROM File Systems"
9. Select "Add a CDROM File System"
10. Select a "Device name", such as "cd0". the CD-ROM file system device names must be unique. you may need to delete a previously defined CD-ROM file system if there is a conflict.
11. Type "/cdrom" for "Mount Point".
12. Select "OK" (or press enter if using the smit ascii interface).
13. Return to the previous smit level, "System Storage Management (Physical & Logical Storage)"
14. Select "File Systems".
15. Select "Mount a File System".
16. Select "/dev/cd0" for "File System name".
17. Select "/cdrom" for "Directory over which to mount".
18. Select "cdrfs" for "Type of file system".
19. Select "yes" for "Mount as a READ-ONLY system".
20. Select "OK" (or press enter if using the smit ascii interface).
21. Exit smit.

Now you are ready to install the appropriate TeamConnection components.

After you have added and mounted a CD-ROM, do the following to install the TeamConnection components:

1. Log in with root authority.
2. Change the directory to /cdrom.
3. Run the installation script by typing ./tcinst.ksh
4. Follow the instructions in the installation script to install the TeamConnection client.

5. After the installation is complete, type the following commands:

```
cd ..  
umount /cdrom
```

6. Eject the CD.

Important installation information about TeamConnection (all platforms)

After you install the TeamConnection code in its directory structure (represented by \$TC_HOME), please keep in mind the following guidelines:

- Do not remove directories, files or symbolic links from \$TC_HOME. Exception: To uninstall the product, you need to remove \$TC_HOME.
- Do not change the file permissions nor the ownership of the directories and/or files in \$TC_HOME.
- Use \$TC_HOME only to store the TeamConnection code:
 - Do not use \$TC_HOME as the home directory for users.
 - Do not use \$TC_HOME as the home directory for TeamConnection families.

Furthermore, if this is your first installation of TeamConnection it is recommended to accept the defaults in this installation script.

Configure the environment variables in the .profile

It is necessary to configure the environment variables in the .profile of the user account before using the TeamConnection client.

1. Login as a user that will use TeamConnection.
2. To set environment variables for TeamConnection you can use the profile.user as a sample .profile:

```
mv $HOME/.profile $HOME/.profile.original  
cp $TC_HOME/install/<language>/profile.user $HOME/.profile
```

Where \$TC_HOME is the location where the TeamConnection code was installed.

3. Customize all the environment variables in the .profile file.

Edit the new .profile and read the entire profile and modify the values as appropriate and uncomment the needed variables that depend on your operating system.

Note: If you are using the CDE environment, edit .dtprofile to enable the account to run .profile after .dtprofile. CDE stands for Common Desktop Environment and is the default for AIX 4 as well as many other versions of UNIX. CDE is the standard of the Open Group (c). For details on how to do it, see the header of the sample profile.

4. Exit the new user ID and login again to refresh the environment variables.

Ensure that the TeamConnection client command is accessible

To ensure that the TeamConnection client is accessible do the following:

1. Type the following TeamConnection command which does not require to connect yet to a working TeamConnection family:

```
teamc report -testClient
```

2. If the teamc command is not found, then ensure that the TC_HOME and PATH variables are properly set in the .profile.
3. If the teamc command is found, then verify that the top portion of the output is something like this, assuming that the LANG variable is en_US:

```
Connect to Family Name:      testfam
Server TCP/IP Name:         hostname.company.com
Server IP Address:          9.87.654.32
Server TCP/IP Port Number:  9000
```

```
Server Specific Information -----
Product Version:            3.0.0
Operating System:           HP-UX
Message catalog language:   English
Server Mode:                non-maintenance
Authentication Level:        PASSWORD_OR_HOST
TC_BECOME:                  your user ID
TC_FAMILY:                   your family name
```

- 4.

If you see that the entry for "Message catalog language" has a value of:

```
English (Internal)
```

Then, it means that the LANG or the NLSPATH variables are not properly set and the TeamConnection client is using its internal messages. It is strongly recommended that the external message catalog should be used.

Use the TeamConnection Client

You can run the TeamConnection line commands from the family account. For example, to see the list of all the users:

```
teamc report -view users -where 1=1
```

To run the TeamConnection GUI, you need to do the following:

1. Copy the sample initial tasks list for the main GUI window. This needs to be done only once.

```
cp $TC_HOME/nls/cfg/$LANG/teamc20.ini $HOME
chmod u+w $HOME/teamc20.ini
```

Where \$TC_HOME is the location where the TeamConnection code was installed.

2. Invoke the TeamConnection GUI by entering:

```
teamcgui &
```

How to use the mnemonics in the TeamConnection GUI for UNIX:

To access the menu for an entry in the menu bar, use:

```
Alt+MnemonicLetter    (such as Alt+f to display the File menu)
```

Even though some mnemonics will be shown in Uppercase, such as "F" for File, the lowercase letter is used. If you try to use the Shift operator with the Alt function, then it will not work:

Alt+Shift+MnemonicLetter (it will not work)

Once a menu is displayed, you can use either of the following:

Action+MnemonicLetter	(such as Action+r for "Refresh now" in the File menu)
Action+Shift+MnemonicLetter	(such as Action+Shift+r for "Refresh now" in the File menu)

You can use either lowercase or uppercase when using the Action key, even though some mnemonics will be shown in Uppercase in the menus (such as "R" for "Refresh now" in the File menu).

Chapter 4. Installing the TeamConnection for Solaris client

This chapter lists the hardware and software that are required before you can install and use the TeamConnection for Solaris client. It also describes what you must do before installation, and provides detailed instructions for installing and configuring the TeamConnection client.

For more information about late breaking news about TeamConnection see the file:

`$TC_HOME/install/readme.txt`

Where `$TC_HOME` is the location where the TeamConnection code was installed.

Hardware requirements

The following are hardware requirements for the Solaris client machine:

Table 5. Hardware Requirements for a Solaris TeamConnection 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.

Software requirements

The following are the software requirements for the Solaris client machine:

Table 6. Software Requirements for TeamConnection for Solaris

Server	Solaris 2.5.1 or a later release of version 2 that includes TCP/IP.
Client	Solaris 2.5.1 or a later release of version 2 that includes TCP/IP.
Build server	Solaris 2.5.1 or a later release of version 2 that includes TCP/IP.

Preparing to install TeamConnection for Solaris

This section explains what you need to do before installing TeamConnection for Solaris. Use the `smit` utility to create user accounts, to update the `hosts` and `services` files, and to add and mount a file system. The `smit` utility starts the System Management Interface Tool (`smit`). This tool presents a menu-driven environment for system administration tasks. For more information on `smit`, refer to your Solaris operating system documentation.

It is assumed that TCP/IP for your operating system has already been installed on your client machine and that someone in your organization has already installed the TeamConnection family server.

Before you install the TeamConnection client code, ask your family administrator to provide you with the following information:

- IP address of the family server
- Family server name
- Name of the family you are to use
- Port address for your family

Ask your family administrator if you need to update your TCP/IP hosts and services files. If you do, follow the steps below:

You can update these files using SMIT. In many installations, a name server is used instead of /etc/hosts. Also, many installations distribute /etc/services. SMIT can be used to make the necessary updates.

1. Update the services file, which is located in /etc/services.

Include the family name and port address of the TeamConnection server.

The port address can be any 4-digit number, as long as it does not already exist in your services file. You might want to ask your TCP/IP administrator to assign you a number.

Type the following entries in your services file:

```
# TeamConnection servers
testfam ffff/tcp # port address for TeamConnection test family
bldsock bbbb/tcp # port address for build agent and processor
```

Note:

- a. For this installation, replace ffff and bbbb in the above example with appropriate port addresses.
 - b. Follow each line with a carriage return.
2. Update the TCP/IP hosts file, which is located in /etc/hosts.
Add the following:
 - IP address.
 - Server name.
 - Alias name of the TeamConnection family server, which is your family name. For this initial installation of TeamConnection, the family name is testfam.
 - Alias name for the build socket. For this initial installation of TeamConnection, use bldsock.

The following is an example of the entry you would type in your hosts file:

```
9.12.345.67 teamserv.company.com testfam bldsock
```

Note: Follow the line with a carriage return. You can use the hostname command to get the name of the server.

3. Do the following to verify that the hosts file is specified correctly:
 - Type "host family_name", where family_name is the name of your TeamConnection family. The information returned should match the number and name specified in your hosts file entry. For example, using the entry given in the previous step, the system response would be as follows:

```
teamserv.company.com = 9.12.345.67
```
 - Type "host ip_address", where ip_address is the IP address of your machine. The information returned should match the number and name specified in your hosts file entry. For example, again using the entry given in the previous step, the system response would be as follows:

9.12.345.67 = teamserv.company.com

If you do not receive the expected response, contact your TCP/IP administrator to solve the problem.

Note: If the changes are done to the domain name server, then you can use the UNIX utility "nslookup" instead.

4. Do the following to verify that you can connect to your TeamConnection family:

At a prompt, type "ping testfam".

If you receive information that is similar to the following, you can successfully connect to your TeamConnection family:

```
PING teamserv.company.com: 56 data bytes
64 bytes from 1.23.457.78: icmp_seg:0. time=0. ms
64 bytes from 1.23.456.78: icmp_seg:1. time=0. ms
64 bytes from 1.23.456.78: icmp_seg:2. time=0. ms
```

5. Press Ctrl+C to end the command.

If you receive the message "unknown host testfam", you cannot connect to the family. Verify that the data you entered in the hosts and services files is correct, and then try the command again. If you still do not get the correct response, contact your TCP/IP administrator to solve the problem.

6. Do the following to verify that you can connect to your TeamConnection build server:

- a. At a prompt, type "ping bldsock".
- b. Press Ctrl+C to end the command.

If you receive information that is similar to the following, you can successfully connect to your TeamConnection build server:

```
PING teamserv.company.com: 56 data bytes
64 bytes from 1.23.457.78: icmp_seg:0. time=0. ms
64 bytes from 1.23.456.78: icmp_seg:1. time=0. ms
64 bytes from 1.23.456.78: icmp_seg:2. time=0. ms
```

If you receive the message "unknown host bldsock", you cannot connect to the build server. Verify that the data you entered in the hosts and services files is correct, and then try the command again. If you still do not get the correct response, contact your TCP/IP administrator to solve the problem.

Do not install the TeamConnection components until the ping commands successfully complete.

Preparing the SYSLOG file

TeamConnection and ObjectStore database errors are recorded in the syslog file. The information recorded in this file will help you resolve problems when they occur.

Before TeamConnection can record these errors, you must activate the syslog daemon. Do the following to activate the syslog daemon:

1. The syslog file is located in /var/spool/syslog. If the log file does not exist, type the following touch command to create it. When you create the log file, follow the directions for setting permissions for your operating system.

```
touch /var/spool/syslog
chmod 666 /var/spool/syslog
chown root:system /var/spool/syslog
```

If you don't want to use `chmod 666`, you can use

```
"chmod a=rw /var/spool/syslog" or
"chmod ugo=rw /var/spool/syslog".
```

The command `"chown root:system"` is a short way to do the following:

```
chown root /var/spool/syslog
chgrp system /var/spool/syslog
```

2. Add the following lines to the `/etc/syslog.conf` file: Use a tab to separate the items.

```
*.warning /var/spool/syslog
```

3. Type the following to stop and then restart the syslog daemon:

```
stopscr -s syslogd
startsrc -s syslogd
```

4. Type the following command to verify that the syslog daemon is running:

```
ps -ef | grep syslog
```

If the daemon is running, you will see one process for `syslogd`.

5. Do the following to verify that the syslog daemon is able to write to the syslog file:

- a. Log in as another user ID.
- b. Type `"su root"` and an incorrect password. This action will fail and a message will be logged in the syslog.

To quickly view the last 10 lines of the syslog, do:

```
tail <syslog-file-name>
***** => use the appropriate name
```

If the syslog file is configured properly, it will contain a message similar to the following:

```
Apr 19 10:21:45 hostname su: BAD SU from userid to root at /dev/pts/3
```

6. If you want to cleanup the syslog, you can use the following command:

```
cp /dev/null <syslog-file-name>
***** => use the appropriate name
```

7. Monitor the syslog file at regular intervals so that any required maintenance or problem resolution can be performed.

Installing from a CD-ROM

The target audience for this section is the system administrator who can login as user `root` and who will do the initial installation of TeamConnection for your organization.

The time required for installing this program varies depending on your workstation. In general, installation of the product takes about 30 minutes.

Note: Installing TeamConnection version 2 causes any previous version of all of the components of TeamConnection to be overwritten without any warning messages. You will not be able to go back to the version of TeamConnection that was previously installed without completely reinstalling that version.

If you already have an existing version of TeamConnection installed, ensure that none of the TeamConnection daemons and tools are running.

Adding and mounting a CD-ROM file system

QUICK START FOR EXPERTS:

- Mount the CD-ROM as /cdrom.
- Use the /cdrom/tcinst.ksh command to install the TeamConnection components you want.

To install the TeamConnection code from a CD-ROM, you must first add and mount a CD-ROM file system. Follow the instructions in this section to complete this task.

Do the following to add and mount a CD-ROM file system:

1. Insert the CD-ROM into the CD-ROM drive.
2. Log in as user root or do "su - root" to login as root and use the profile.
3. Create a directory for the CD-ROM /cdrom by entering "mkdir /cdrom"
4. Enter "smit" to add a CD-ROM file system.
5. Select "System Storage Management (Physical & Logical Storage)"
6. Select "File Systems"
7. Select "Add / Change / Show / Delete File Systems."
8. Select "CDROM File Systems"
9. Select "Add a CDROM File System"
10. Select a "Device name", such as "cd0". the CD-ROM file system device names must be unique. you may need to delete a previously defined CD-ROM file system if there is a conflict.
11. Type "/cdrom" for "Mount Point".
12. Select "OK" (or press enter if using the smit ascii interface).
13. Return to the previous smit level, "System Storage Management (Physical & Logical Storage)"
14. Select "File Systems".
15. Select "Mount a File System".
16. Select "/dev/cd0" for "File System name".
17. Select "/cdrom" for "Directory over which to mount".
18. Select "cdrfs" for "Type of file system".
19. Select "yes" for "Mount as a READ-ONLY system".
20. Select "OK" (or press enter if using the smit ascii interface).
21. Exit smit.

Now you are ready to install the appropriate TeamConnection components.

After you have added and mounted a CD-ROM, do the following to install the TeamConnection components:

1. Log in with root authority.
2. Change the directory to /cdrom.
3. Run the installation script by typing ./tcinst.ksh
4. Follow the instructions in the installation script to install the TeamConnection client.
5. After the installation is complete, type the following commands:

```
cd ..  
umount /cdrom
```

6. Eject the CD.

Important installation information about TeamConnection (all platforms)

After you install the TeamConnection code in its directory structure (represented by \$TC_HOME), please keep in mind the following guidelines:

- Do not remove directories, files or symbolic links from \$TC_HOME. Exception: To uninstall the product, you need to remove \$TC_HOME.
- Do not change the file permissions nor the ownership of the directories and/or files in \$TC_HOME.
- Use \$TC_HOME only to store the TeamConnection code:
 - Do not use \$TC_HOME as the home directory for users.
 - Do not use \$TC_HOME as the home directory for TeamConnection families.

Furthermore, if this is your first installation of TeamConnection it is recommended to accept the defaults in this installation script.

Configure the environment variables in the .profile

It is necessary to configure the environment variables in the .profile of the user account before using the TeamConnection client.

1. Login as a user that will use TeamConnection.
2. To set environment variables for TeamConnection you can use the profile.user as a sample .profile:

```
mv $HOME/.profile $HOME/.profile.original
cp $TC_HOME/install/<language>/profile.user $HOME/.profile
```

Where \$TC_HOME is the location where the TeamConnection code was installed.

3. Customize all the environment variables in the .profile file.
Edit the new .profile and read the entire profile and modify the values as appropriate and uncomment the needed variables that depend on your operating system.

Note: If you are using the CDE environment, edit .dtprofile to enable the account to run .profile after .dtprofile. CDE stands for Common Desktop Environment and is the default for AIX 4 as well as many other versions of UNIX. CDE is the standard of the Open Group (c). For details on how to do it, see the header of the sample profile.

4. Exit the new user ID and login again to refresh the environment variables.

Ensure that the TeamConnection client command is accessible

To ensure that the TeamConnection client is accessible do the following:

1. Type the following TeamConnection command which does not require to connect yet to a working TeamConnection family:

```
teamc report -testClient
```

2. If the teamc command is not found, then ensure that the TC_HOME and PATH variables are properly set in the .profile.

3. If the `teamc` command is found, then verify that the top portion of the output is something like this, assuming that the `LANG` variable is `en_US`:

```
Connect to Family Name:      testfam
Server TCP/IP Name:         hostname.company.com
Server IP Address:          9.87.654.32
Server TCP/IP Port Number:  9000
```

```
Server Specific Information -----
Product Version:            3.0.0
Operating System:           Solaris
Message catalog language:   English
Server Mode:                non-maintenance
Authentication Level:        PASSWORD_OR_HOST
TC_BECOME:                  your user ID
TC_FAMILY:                  your family name
```

- 4.

If you see that the entry for "Message catalog language" has a value of:
English (Internal)

Then, it means that the `LANG` or the `NLSPATH` variables are not properly set and the TeamConnection client is using its internal messages. It is strongly recommended that the external message catalog should be used.

Use the TeamConnection Client

You can run the TeamConnection line commands from the family account. For example, to see the list of all the users:

```
teamc report -view users -where 1=1
```

To run the TeamConnection GUI, you need to do the following:

1. Copy the sample initial tasks list for the main GUI window. This needs to be done only once.

```
cp $TC_HOME/nls/cfg/$LANG/teamc20.ini $HOME
chmod u+w $HOME/teamc20.ini
```

Where `$TC_HOME` is the location where the TeamConnection code was installed.

2. Invoke the TeamConnection GUI by entering:

```
teamcgui &
```

How to use the mnemonics in the TeamConnection GUI for UNIX:

To access the menu for an entry in the menu bar, use:

```
Alt+MnemonicLetter    (such as Alt+f to display the File menu)
```

Even though some mnemonics will be shown in Uppercase, such as "F" for File, the lowercase letter is used. If you try to use the Shift operator with the Alt function, then it will not work:

```
Alt+Shift+MnemonicLetter  (it will not work)
```

Once a menu is displayed, you can use either of the following:

Action+MnemonicLetter	(such as Action+r for "Refresh now" in the File menu)
Action+Shift+MnemonicLetter	(such as Action+Shift+r for "Refresh now" in the File menu)

You can use either lowercase or uppercase when using the Action key, even though some mnemonics will be shown in Uppercase in the menus (such as "R" for "Refresh now" in the File menu).

Chapter 5. Installing the TeamConnection client for OS/2

This chapter lists the hardware and software that are required before you can install and use the TeamConnection OS/2 client. It also describes what you must do before installation, and provides detailed instructions for installing and configuring the TeamConnection client.

Hardware requirements

The following are hardware requirements for the OS/2 client machine:

Table 7. Hardware Requirements for an OS/2 TeamConnection 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	12 MB minimum
Disk space	<ul style="list-style-type: none">• 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

Software requirements

The following are software requirements for the OS/2 client machine:

Table 8. Software Requirements for TeamConnection for OS/2

Server	One of the following: <ul style="list-style-type: none">• OS/2 Warp Version 3 (part number 83G8100) or Version 4 (84H1428) and one of the following:<ul style="list-style-type: none">– TCP/IP Version 3.0 for OS/2 Warp (part number 33H9749)– Anynet/2 Version 2.0 (part number 87G7776)• OS/2 Warp Connect Version 3 (part number 10H9800)• OS/2 Warp Server Version 4 (part number 25H8002)• OS/2 Warp Server Advanced Version 4 (part number 25H8030)• OS/2 Warp Server Advanced Version 4 SMP Feature (part number 28H0150)
---------------	--

Table 8. Software Requirements for TeamConnection for OS/2 (continued)

Client	<p>One of the following:</p> <ul style="list-style-type: none"> • OS/2 Warp Version 3 (part number 83G8100) and one of the following: <ul style="list-style-type: none"> – TCP/IP Version 3.0 for OS/2 Warp (part number 33H9749) – Anynet/2 Version 2.0 (part number 87G7776) • OS/2 Warp Version 4 (part number 84H1426) and one of the following: <ul style="list-style-type: none"> – TCP/IP Version 3.0 for OS/2 Warp (part number 33H9749) – Anynet/2 Version 2.0 (part number 87G7776) • OS/2 Warp Connect Version 3 (part number 10H9800) • OS/2 Warp Server Version 4 (part number 25H8002) • OS/2 Warp Server Advanced Version 4 (part number 25H8030) • OS/2 Warp Server Advanced Version 4 SMP Feature (part number 28H0150) <p>Note: If double-byte character set (DBCS) support is required, one of the following DBCS versions of OS/2:</p> <ul style="list-style-type: none"> – IBM OS/2 J version 2.11 or later – IBM OS/2 H version 2.11 or later – IBM OS/2 T version 2.11 or later – IBM OS/2 P version 2.11 or later
Build server	<p>One of the following:</p> <ul style="list-style-type: none"> • OS/2 Warp Version 3 (part number 83G8100) and one of the following: <ul style="list-style-type: none"> – TCP/IP Version 3.0 for OS/2 Warp (part number 33H9749) – Anynet/2 Version 2.0 (part number 87G7776) • OS/2 Warp Connect Version 3 (part number 10H9800) • OS/2 Warp Server Version 4 (part number 25H8002) • OS/2 Warp Server Advanced Version 4 (part number 25H8030) • OS/2 Warp Server Advanced Version 4 SMP Feature (part number 28H0150)
Distribution function	NetView Distribution Manager/2 Version 2.0 or later (53G3924)

Preparing to install TeamConnection for OS/2

This section explains what you need to do before installing TeamConnection for OS/2.

It is assumed that TCP/IP for your operating system has already been installed on your client machine and that someone in your organization has already installed the TeamConnection family server.

Before you install the TeamConnection client code, ask your family administrator to provide you with the following information:

- IP address of the family server
- Family server name
- Name of the family you are to use
- Port address for your family

Ask your family administrator if you need to update your TCP/IP hosts and services files. If you do, follow the steps below:

1. Add the family name and port address of the TeamConnection server to the services file. If you have this file, it is located in the directory where TCP/IP is installed. To determine the directory name, type

```
echo %etc%
```

at a command line prompt. Typically, this file is located in the
\\ETC

subdirectory of the
TCPIP

directory.

The following is an example of the entry you would type in your services file:

```
# TeamConnection families
tcfamily 4321/tcp      # port address for the TeamConnection family
```

Note: Follow the line with a carriage return.

2. Add the following information to the hosts file. If you have this file, it is located in the directory where TCP/IP is installed. To determine the directory name, type

```
echo %etc%
```

at a command line prompt. Typically, this file is located in the
\\ETC

subdirectory of the
TCPIP

directory. If the hosts file does not exist, you must configure it using TCP/IP.

- IP address
- Server name
- Alias name of the TeamConnection family server

The following is an example of the entry you would type in your hosts file:

```
9.12.345.67    teamserv.company.com    tcfamily
```

Note: Follow the line with a carriage return.

3. Do the following to verify that the hosts file is specified correctly:

- a. Type

```
host family_name
```

, where

```
family_name
```

is the name of your TeamConnection family. The information returned must match the number and name specified in your hosts file entry. For example, using the entry given in the previous step, the system response would be as follows:

teamserv.company.com = 9.12.345.67

b. Type

host *ip_address*

, where

ip_address

is the IP address of your machine. The information returned must match the number and name specified in your hosts file entry. For example, again using the entry given in the previous step, the system response would be as follows:

9.12.345.67 = teamserv.company.com

If you do not receive the expected response, contact your TCP/IP administrator to solve the problem.

4. Do the following to verify that you can connect to your TeamConnection family:

a. At a prompt, type

ping *family*

, where

family

is the family name.

b. Press Ctrl+C to end the command.

If you receive information that is similar to the following, you can successfully connect to your TeamConnection family:

```
PING teamserv.company.com: 56 data bytes
64 bytes from 9.12.345.67: icmp_seg:0. time=0. ms
64 bytes from 9.12.345.67: icmp_seg:1. time=0. ms
64 bytes from 9.12.345.67: icmp_seg:2. time=0. ms
```

Note: The PING command will continue to send requests until you stop it by pressing Ctrl+C at the same time.

If you receive the message

unknown host *family*

, you cannot connect to the family. Verify that the data you entered in the hosts and services files is correct, and then try the command again. If you still do not get the correct response, contact your TCP/IP administrator to solve the problem.

Do not install the TeamConnection client until the ping command successfully completes.

Installing from a LAN

You can install the TeamConnection client onto your workstation from a LAN if a response file specifying your family name exists on the family server. If you are not sure, talk to your family administrator.

To install TeamConnection from a LAN, follow these steps:

1. Log on to your LAN.
2. Change to the LAN directory where TeamConnection is installed. By default this directory is named

`x:\teamc\clients`

where

`x`

is the drive.

3. Type the following and press Enter:

`install /x`

The

`/x`

indicates an unattended installation.

The TeamConnection client code is installed in the directory specified in the response file.

Installing using the installation GUI

In order to install the TeamConnection client code using the installation GUI, you must have access to a drive where the TeamConnection CD is accessible, either on your local system or as a mounted drive.

Do the following to install the client code using the GUI:

Step 1: Starting the installation

1. Type

`x:\install`

at a prompt, where `x` is the drive where the installation program is located.

The TeamConnection Installation window appears, followed by the Installation Instructions window.

2. Select **Continue**. The Install window appears.

Step 2: Updating the config.sys file

1. From the Install window, indicate that you want to have your config.sys file updated. This is the default. If you want existing installation files overwritten during the installation, select **Overwrite files**.

If you select not to have your config.sys file updated automatically, you will need to update the file manually after the installation program completes.

TeamConnection creates a file called config.add that contains your entire config.sys file with environment variables and changes to the PATH and LIBPATH statements.

2. Select **OK** or press Enter. The Install - directories window appears.

Step 3: Selecting the components for installation

Use the Install - directories window to select the components you want to install and where you want them installed.

1. Select the two components that you are going to install: TeamConnection Client and TeamConnection Online Documentation.
2. To verify that the default drive has sufficient space, or to see what other drives are available, select **Disk space**.

If you want to use a different drive, do the following from the Disk space window:

- a. Select the drive that you want to use.
 - b. Select **Change directories to selected drive**.
 - c. Select **OK** or press Enter. The Install - directories window appears.
3. Select **Install** from the Install - directories window to start the installation. The Install - progress window shows you the progress of the installation.

Step 4: Completing the installation

1. After the installation completes, the Installation and Maintenance window appears with an indication that TeamConnection successfully installed.
Select **OK** or press Enter to return to the TeamConnection Installation window.
2. Select **Exit** or press F3.
3. Verify that the installation program created a folder called **TeamConnection Group**.

Step 5: Customizing the TeamConnection client environment

The installation program updated your config.sys file with the TeamConnection client environment variables. These variables have the following values:

- TC_FAMILY=testfam
- TC_WWWPATH=x:\teamc\www
- TC_BECOME=user
- TC_USER=user
- LIBPATH=x:\teamc\dl
- NLSPATH=x:\teamc\bin\%N
- PATH=x:\teamc\bin
- TMP=x:\teamc\temp

Where:

- *user* is the USER environment variable you specified when TCP/IP was installed.
- *x* is the drive on which the TeamConnection client is installed.
- *teamc* is the default directory created for the client.

Note: If the TMP environment variable was previously set by another application, TeamConnection uses the current value.

Edit your config.sys file if you want to change the user ID in the TC_BECOME and TC_USER environment variables. Change the TeamConnection family name if you are going to connect to a family other than testfam.

Step 6: Verifying the installation

Shut down the workstation and then restart it to accept the changes made to the config.sys file.

Type the following command to verify that the environment variables in the config.sys file are correct:

```
teamc report -testClient
```

If everything is set correctly, TeamConnection displays information about your TeamConnection family.

Type the following command to verify that the client is connected to the TeamConnection family server:

```
teamc report -testServer
```

If everything is set correctly, TeamConnection displays information about the server.

Step 7: Starting the TeamConnection client

To start the TeamConnection client, select the **TeamConnection Client** icon from the TeamConnection Group folder on the desktop. The Tasks window appears.

Before you start to use TeamConnection, see “Chapter 7. Getting familiar with the TeamConnection client interfaces” on page 51.

Chapter 6. Installing the TeamConnection clients for Windows

This chapter lists the hardware and software that are required before you can install and use the TeamConnection Windows clients. It also describes what you must do before installation, and provides detailed instructions for installing and configuring the TeamConnection client.

Hardware requirements

The following are hardware requirements for the Windows 3.1 client machine:

Table 9. Hardware Requirements for a Windows TeamConnection Client

Processor	<ul style="list-style-type: none">• For Windows 3.1, any personal workstation with an Intel 80286 or higher processor• For Windows 95 and NT, any personal workstation supported by the operating system, except the PowerPC
Pointing device	A mouse or other pointing device
Monitor	VGA or higher resolution with the appropriate adapter
Memory	12 MB minimum
Disk space	<ul style="list-style-type: none">• 75 MB for operating system and prerequisites• 25 MB for TeamConnection client code• 32 MB for swapper space
Communications support	Any network card supported by the above workstation that supports TCP/IP
CD-ROM drive	A CD-ROM drive (internal or external) for installation of the product

The following are software requirements for the Windows client machines:

Table 10. Software Requirements for TeamConnection for Windows

Server	<ul style="list-style-type: none">• Microsoft Windows NT, Version 3.5.1 or later, which includes TCP/IP• To run the installation verification tool, you need IBM Object REXX for Windows (10J9372) or the Personal REXX for Windows product available from Quercus Systems (P.O. Box 2157, Saratoga, CA 95070, Phone: 408-867-REXX)
Clients	One of the following: <ul style="list-style-type: none">• Microsoft Windows 95, which includes TCP/IP• Microsoft Windows NT, Version 3.5.1 or Version 4.0, which includes TCP/IP• Microsoft Windows, Version 3.1 or later and the following:<ul style="list-style-type: none">– DOS Version 3.3 or later– IBM TCP/IP for DOS Version 2.1.1 (part number 87G7184)
Windows NT build server	Microsoft Windows NT, Version 3.5.1 or later, which includes TCP/IP
Windows 95 build server	Microsoft Windows 95

Preparing to install TeamConnection for Windows

This section explains what you need to do before installing TeamConnection for Windows.

It is assumed that TCP/IP for your operating system has already been installed on your client machine and that someone in your organization has already installed the TeamConnection family server.

Before you install the TeamConnection client code, ask your family administrator to provide you with the following information:

- IP address of the family server
- Family server name
- Name of the family you are to use
- Port address for your family

Ask your family administrator if you need to update your TCP/IP hosts and services files. If you do, follow the steps below:

1. Add the family name and port address of the TeamConnection server.

3.1

For Windows 3.1:

- a. Add the following TCP variables to your AUTOEXEC.BAT file.

```
SET USER=<your_cmvc_user_id_in_lowercase>  
SET HOSTNAME=<your_host_name>
```

- b. Add the TCP ports that you plan to use to the bottom of your C:\TCPDOS\ETC\SERVICES file. For example:

```
#  
# Sample Port Entries for the Services File  
#  
octo      1300/tcp  
toro      1411/tcp  
cmvctest  1410/tcp
```

Note: Follow the line with a carriage return.

- c. Restart your workstation to begin using the changed system files.
- 2.

For Windows NT:

- a. Edit the hosts and services files.

Use the Notepad accessory to open the hosts and services files. These files are located in the following directory:

c:\winnt35\system32\drivers\etc

Where

c:\

represents the drive where Windows is installed.

- b. Add the proper information in the hosts and services files.
The following are sample port entries for the services file.

```
#
# Sample Port Entries for the Services File
#
teamserv 1300/tcp
sandbox  1411/tcp
family1  1410/tcp
```

Note: Follow each line with a carriage return.

The following are sample port entries for the hosts file.

```
#
# Sample Port Entries For the Hosts File
#
9.12.345.67      teamserv.company.com
9.12.345.78      sandbox
9.12.345.89      family1
```

Note: Follow each line with a carriage return.

- c. Restart your workstation to begin using the changed system files.
- 3.

95

For Windows 95:

- a. Edit the hosts and services files.
Select **Start** then **Programs** and then **Windows Explorer**.
- b. Select the folder where you installed Windows 95, and then select **edit** to edit the hosts and services files.
- c. Add the proper information in the hosts file, if needed.
The following are sample port entries for the services file.

```
#
# Sample Port Entries for the Services File
#
teamserv 1300/tcp
sandbox  1411/tcp
family1  1410/tcp
```

Note: Follow each line with a carriage return.

The following are sample port entries for the hosts file.

```
#
# Sample Port Entries For the Hosts File
#
9.12.345.67      teamserv.company.com
9.12.345.78      sandbox
9.12.345.89      family1
```

Note: Follow each line with a carriage return.

- d. Restart your workstation to begin using the changed system files.
4. After rebooting your system, do the following to verify that you can connect to your TeamConnection family:

3.1

For Windows 3.1:

- a. At the DOS prompt, enter:

TCPCHECK

This command displays the status of each service that is running. For example:

```
Check to see if INET is up
Check to see if ADAPTER works
Check to see if GATEWAY is present
Check to see if NAMESERVER is present
Check to see if NAMESERVER works
Your basic configuration is correct; any exceptions are noted above.
```

If an entry is displayed for all of the services, your basic configuration is correct; any exceptions are noted above.

- b. At a prompt, type

ping family

, where

family

is the family name.

For example:

ping teamserv

If you receive information that is similar to the following, you can successfully connect to your TeamConnection family:

PING teamserv.company.com (9.12.345.67): 56 data bytes

64 bytes from 9.12.345.67: icmp_seq=0 ttl=59 time=0 ms ...

Note: The PING command will continue to send requests until you stop it by pressing Ctrl+C at the same time.



For Windows NT:

At a prompt, type

```
ping family
```

, where

```
family
```

is the family name.

For example:

```
ping teamserv
```

If you receive information that is similar to the following, you can successfully connect to your TeamConnection family:

```
PINGING teamserv.company.com (9.12.345.67): with 32 bytes of data:
```

```
Reply from 9.12.345.67: bytes=32 time=10ms TTL=255 ...
```

Note: The PING command will send four requests and then it will stop.



For Windows 95:

At a prompt, type

```
ping family
```

, where

```
family
```

is the family name.

For example:

```
ping teamserv
```

If you receive information that is similar to the following, you can successfully connect to your TeamConnection family:

```
PINGING teamserv.company.com (9.12.345.67): with 32 bytes of data:
```

```
Reply from 9.12.345.67: bytes=32 time=10ms TTL=255 ...
```

Note: The PING command will send four requests and then it will stop.

Installing using the installation GUI

In order to install the TeamConnection client code using the installation GUI, you must have access to a drive where the TeamConnection CD is accessible, either on your local system or as a mounted drive.

Do the following to install the client code using the GUI:

Step 1: Starting the installation for Windows 3.1 and Windows NT

To install the TeamConnection client on Windows 3.1 or Windows NT, do the following:

Note: If you are installing the TeamConnection client on Windows 3.1 you must be running in enhanced mode and have virtual memory enabled.

1. On the Program Manager File pull-down menu, select **Run**.
2. Type

`x:\setup`

, where *x* is the drive where the installation program is located.

The TeamConnection Welcome window appears.

3. Select **OK**.
4. Select **Next**.



If you are installing on Windows 3.1, the installation program checks to see if Win32s is installed and verifies the version. If Win32s is not installed or is an earlier version, you are asked if you want to install it. Select **Yes** to install Win32s or **No** to exit the TeamConnection installation.

If you choose to install Win32s, you may see a severe warning message like the following:

Win32s Setup: SHARE.EXE is not loaded. File sharing must be enabled.
Run SHARE.EXE before starting Windows, or add SHARE.EXE to your
AUTOEXEC.BAT file.

If you do see this message, do the following:

1. Select **OK** to exit the installation.
2. Run SHARE.EXE or add SHARE.EXE to your AUTOEXEC.BAT file.
3. Start the installation again.

Note: If you choose to run SHARE.EXE, you **must** exit Windows first. If you choose to update your AUTOEXEC.BAT, you must reboot your machine before restarting the installation.

Starting the installation for Windows 95

To install the TeamConnection client on Windows 95, do the following:

1. On the Windows 95 desktop, select **Start**.
2. Select **Run**.
3. Type
`x:\setup`

, where *x* is the drive where the installation program is located.

The TeamConnection Welcome window appears.

4. Select **Next**.

Step 2: Selecting the components for installation

Use the Select Components window to select the components you want to install and specify where you want them installed.

1. Select the components that you want to install.
2. Select **Next**. The Select Destination Directory window appears.
3. On the Select Destination Directory window, you can verify that the default drive has sufficient space, or see what other drives are available. To see what drives are available, select **Browse**.

If you want to use a different drive, do the following from the Choose directory window:

- a. Choose the directory for installation.
- b. Select **OK**. The Select Destination Directory window appears with the new destination directory.
4. Select **Next**. The Select Program Folder window appears.

Step 3: Adding program icons to the Program Folder

Use the Select Program Folder Window to specify the name of the program folder in which the TeamConnection icons will be created.

Step 4: Copying the files

Use the Start Copying Files window to verify that you are installing the desired components into the desired Program Directory and Program Folder.

Select **Next** to start copying the files.

Step 5: Completing the installation

1. After the installation completes, the Setup Complete window appears. You can choose to restart your system immediately or restart your system later.
 Select **Finish** or press Enter to restart your system or return to the Windows desktop.
2. Once you restart your system, verify that the installation program created a folder called **TeamConnection Group**.

Step 6: Customizing the TeamConnection client environment

The installation program updated your environment with the TeamConnection client environment variables. These variables have the following values:

- TC_FAMILY=testfam
- TC_BECOME=user
- TC_USER=user

Where:

- *user* is the USER environment variable you specified when TCP/IP was installed.

Edit your environment file if you want to change the user ID in the TC_BECOME and TC_USER environment variables. Change the TeamConnection family name if you are going to connect to a family other than testfam.

Step 7: Verifying the installation

Shut down the workstation and then restart it to accept the changes made to the environment.

Type the following command to verify that the environment variables in the config.sys file are correct:

```
teamc report -testClient
```



If you are using Windows 3.1, you must type this command in the entry field at the bottom of the TeamConnection Tasks window just above the user and family names, or use the TeamConnection Command window.

If everything is set correctly, TeamConnection displays information about your TeamConnection family.

Type the following command to verify that the client is connected to the TeamConnection family server:

```
teamc report -testServer
```

If everything is set correctly, TeamConnection displays information about the server.

Step 8: Starting the TeamConnection client

To start the TeamConnection client, select the **TeamConnection Client** icon from the TeamConnection Group folder on the desktop. The Tasks window appears.

Before you start to use TeamConnection, see “Chapter 7. Getting familiar with the TeamConnection client interfaces” on page 51.

Chapter 7. Getting familiar with the TeamConnection client interfaces

TeamConnection provides several interfaces that you can use to access data:

- A graphical user interface based on the Common User Access (CUA) architecture
- A command line interface that lets you type TeamConnection commands from a prompt or from within TeamConnection

If you are using Windows 3.1, you must use the entry field at the bottom of the TeamConnection Tasks window just above the user and family names, or use the TeamConnection Command window to enter TeamConnection commands.

- A web client, that you access through your web browser.

You can use any of the interfaces to do all of your TeamConnection work, or you can switch back and forth between the them. You might find that some tasks are easier to do from the GUI or through the web, while others are easier to do from the command line.

This chapter helps you to begin using the TeamConnection client interfaces. It describes the following:

- Using the GUI
 - Starting and stopping the GUI
 - Getting around in the GUI
 - Using the Settings notebook
 - Using the online help that is provided with TeamConnection
- Using the command line interface
- Using the web client

Before you can use TeamConnection, someone in your organization with superuser authority, such as your family administrator, must create for you a unique user ID and a host list entry for the workstation where you installed the client.

Using the GUI

TeamConnection provides a GUI that you can use to do all of your TeamConnection work. To use the GUI efficiently, set your default values in your Settings notebook to suit your working environment, and then become familiar with the Tasks window and how you can save time by adding your most common tasks to it.

Starting the GUI

You can start the TeamConnection client GUI in one of the following ways:

- Select the **TeamConnection Client** icon from the TeamConnection Groupfolder on the desktop.
- Type `teamcgui` from a prompt in the directory where TeamConnection was installed.

If you are using AIX or HP-UX, type `teamcgui` from a prompt in your home directory.

The Tasks window appears.

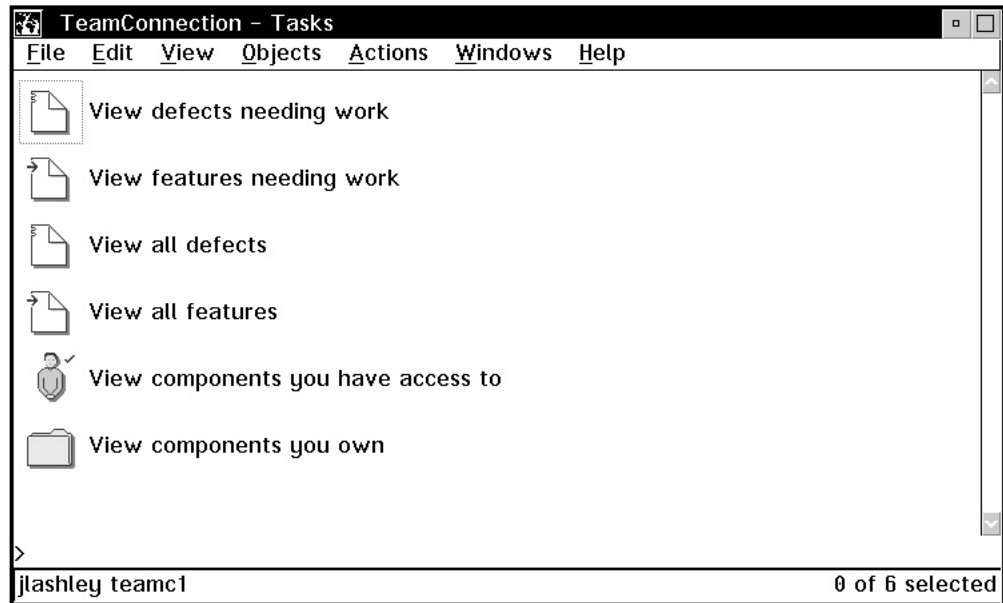


Figure 4. Tasks window

Initially, a set of default tasks appears in your Tasks window. As you become more familiar with TeamConnection and see what tasks you do most often, you can change, delete from, and add new tasks to this list. To learn how to do this, select **How do I** from the Help pull-down menu, and then select **Update tasks on the Tasks** window.

From the Tasks window, you can either select actions from the menu bar or select a task.

Stopping the GUI

To stop the TeamConnection client GUI, do one of the following:

- Select **Close** from the System menu in the Tasks window.
- Select **Exit** from the File pull-down menu of a TeamConnection window.

Performing tasks with the GUI

There are several ways you can perform TeamConnection tasks with the GUI. You can:

- Select an action from the Actions pull-down menu and then select the object you want to work with. For example, if you want to view a specific defect, select **Defects → View** from the Actions pull-down menu; then type the name of the defect in the View Defects window.

This method is useful when you know the exact names of the objects you want to work with.

- Select the type of object you want to work with, such as **Defects**, from the Objects pull-down menu. A Filter window appears in which you can specify search criteria. You then get a list of objects that match the search criteria. Online help provides information about using the Filter window.

This method is useful when you do not know the exact name of the object you want to work with, or you want to view a list of objects.

After you have a list of objects, and if you are going to use this list at other times, you can keep the window open. Leave the window in the background as you do your other work, or minimize it. This way, you can quickly retrieve the list when you want to perform another action.

- Select a task from the Tasks window.

This method provides a fast path within the GUI. When you select a task, TeamConnection performs the underlying query or command and then displays the requested information.

- Select an object from an object window (such as the Parts, Defects, or Features window) and then select an action to perform on the selected part from the **Selected** menu. You can also display a pop-up menu listing valid actions for a specific object by placing the mouse pointer over the object and pressing mouse button 2.

Using the Settings notebook

The TeamConnection GUI provides a Settings notebook in which you can set default values for your working environment. To open the Settings notebook, select **Settings** from the Windows pull-down menu. You can set the following values; for more information about them, refer to the online help. The notebook has five pages.

On the Environment page:	<ul style="list-style-type: none">• Family• Release• Component• Work area• Become user• User ID• Top• Relative directory• Working directory	The environment variables you specify on this page are relevant for the GUI only, not the command line.
On the Setup page:	<ul style="list-style-type: none">• NLS path• Log file• Case• Print command• Compare command• Edit command	

On the GUI page:	<ul style="list-style-type: none"> • Verbose commands • Auto refresh • Multiple object windows • Show query line • Sort pre-defined list values • Use small icons in icon views (not available in Windows clients) • Use small icons in tree views (not available in Windows clients) • Font for object windows (not available in Windows clients) • Font for output windows (not available in Windows clients) • Required field label color • Modified field label color 	
On the Extract page:	<ul style="list-style-type: none"> • Destination directory • Read-only • Expand keywords 	
On the Pool page:	<ul style="list-style-type: none"> • Pool 	

Online help information

Online help information is available from anywhere in the TeamConnection GUI. Use the online help when you need more information about a topic or task.

TeamConnection offers two types of help:

- General help

This is help for a specific window. General help provides an overview of the task and describes the objects on the window, such as menu-bar items, icons, fields, and push buttons. Do one of the following to access general help:

- Select **Help** from a menu bar.
- Select the **Help** push button.
- Press F1.

- How do I

This is where you find step-by-step instructions for doing a specific task. How you do a task depends on the component or release process that is being followed, and this help information takes that into consideration. To access this help, select **How do I** from the Help pull-down menu. Double-click on one of the task items.

At the bottom of each Help window is a **Diagram** push button. Select this push button to view a graphical process diagram. Step your way through the diagram to better understand the processes that TeamConnection components and releases can follow. The processes that your components and releases follow depend on

how the processes are configured for your organization. The defined processes determine the actions that must occur before a defect or feature can move toward completion.

Using the command line interface

To use the command line interface effectively, you must be familiar with the actions that you can perform using TeamConnection commands. A complete description of each command, including examples for each, is available in the *Commands Reference*

To view the syntax of a TeamConnection command online, type the following at a prompt:

```
teamc commandName
```

Where *commandName* is the name of the TeamConnection command.

The *Quick Commands Reference* is a booklet that lists the syntax of each TeamConnection command.

You can also become familiar with the commands by looking at the contents of the log file where TeamConnection stores the commands that are issued as you use the GUI. This file is specified in the **Log file** field on the Setup page of the Settings notebook. The default name is teamc.log; it is stored in the directory where the client is installed, unless you specify a different location in the Settings notebook.

You can type TeamConnection commands from a prompt within any directory; the TeamConnection GUI does not need to be started. Or you can type a command on the command line in the Tasks window.

Before you start to use the command line interface, you might want to set the most used environment variables, such as TC_FAMILY or TC_COMPONENT. You are not required to set these environment variables, but if you do not, you will need to specify them in the command when required.

You set environment variables differently for different platforms: AIX and HP-UX users set environment variables in the .profile (sh, ksh environment), .dtprofile (cde environment), or .cshrc (csh environment). OS/2 and Windows 3.1 users set environment variables in the config.sys file or from a command line prompt. Some environment variables are set in your config.sys file during installation.

Windows 95 and Windows NT users set environment variables in the Windows Control Panel.

You can override the value you set for an environment variable by using the corresponding flag in the command. For example, you have the TC_FAMILY environment variable set to robot, but you need to extract a file from another family named octo, so you issue the following command:

```
teamc part -extract hello.c -family octo -release 9501
```

provides a complete list of the

The following list provides a brief description of each command.

Command	Purpose
Access	Updates the access lists for a component, identifying the user IDs that have explicit authority to perform actions on it.
Approval	Marks <i>approval records</i> with approvers' opinions about proposed changes in a release.
Approver	Updates the list of approvers for a release.
Builder	Creates and maintains builders, which are used in the build function.
Collision	Enables you to accept, reject, or reconcile <i>collision records</i> during <i>concurrent development</i> .
Component	Creates and maintains components in a family. Defines a component hierarchy.
Coreq	Identifies work areas as corequisites, that is, work areas that must be included in the same driver. If the release is not following a tracking process, the work areas must be integrated into the release at the same time.
Defect	Monitors the reporting, evaluation, and resolution of problems.
Driver	Defines and works with a collection of work area changes within a release.
DriverMember	Adds work areas to or deletes work areas from a driver.
Environment	Updates the environment list for a release, identifying the test environments and the names of testers.
Feature	Monitors the suggestion, evaluation, and implementation of design changes and enhancements.
Fix	Marks the fix records for a component identifying user IDs that receive notification of actions on the component.
Host	Identifies client access on the host list associated with a user ID.
Notify	Identifies notification interest for user IDs using component notification lists.
Parser	Creates and maintains parsers. Use this for the build function.
Part	<ul style="list-style-type: none"> Places parts in the TeamConnection environment and lets users work with them. Starts or stops the build function. Manages the <i>build tree</i>.
Prereq	Identifies work areas as prerequisites, that is, work areas that must be included in the same driver or a previously committed driver. If the release is not following a tracking process, the work areas must be integrated into the release previously or at the same time.
Release	Creates and maintains releases to group project-related parts.
Report	Searches database tables for information on TeamConnection objects.
Size	Updates the sizing records for defects and features.
Tclogin	Log into or out of, and display a list of user login IDs.
Test	Updates environment test records, identifying testers' opinions about test results.
User	Creates user IDs and maintains information about the owners.
Verify	Updates verification records, indicating the outcome of defects and features.
Workarea	Creates, maintains, freezes, and refreshes work areas.

Using the web client

The TeamConnection Web Client provides family server connectivity and great deal of the functionality provided by a standard TeamConnection client without the overhead required by a standard client installation. Using a web browser, anyone in the organization can access server data (provided the server is configured appropriately) by addressing a machine and port number. Although file input/output functions are not currently available, most other familiar TeamConnection functions are available through the Web client.

To begin using the TeamConnection web client you must point your web browser to the correct URL. The syntax of the URL is: *http://host name of the server:port number of your family*. For example, if your server host name is *bldproc1* and your port number is *7890*, the URL would look like: **http://bldproc1:7890**

Your organization might require that you log in to the TeamConnection family server before you can access TeamConnection objects. If you are accessing TeamConnection through the host, you will not be able to go through a proxy unless you are:

- Logging in using a password.
- Using a smart proxy.

Using the web client is much like using the TeamConnection GUI. The following are some differences you might find:

- For the BuilderView filter the following are available in the TeamConnection GUI but not in the TeamConnection web client:
 - Source File
 - TIME-OUT
 - setupOptions
- A filter for Corequisites does not exist in the TeamConnection GUI.
- For the DefectModify action, *originLogin* is available for use with the TeamConnection web client (but not the TeamConnection GUI).
- For the DriverCheck action, *Dependencies* is available in the TeamConnection GUI but not in the TeamConnection web client.
- For the FeatureModify action, the following are available with the TeamConnection web client (but not the TeamConnection GUI):
 - *newName*
 - *originLogin*.
- For the DriverMemberView action, the following are available with the TeamConnection web client (but not the TeamConnection GUI):
 - *state*
 - *defectName*
 - *defectAbstract*
 - *committedVersion*.
- For the ChangeView action, the following are available with the TeamConnection web client (but not the TeamConnection GUI):
 - *ChangeView*
 - *workAreaState*.
- For the PartBuild action, the following are available with the TeamConnection web client (but not the TeamConnection GUI):

- cancel
- partType.
- The PartChildInfoView action does not exist in the TeamConnection GUI.
- For the PartDelete action, force is available for use with the TeamConnection web client (but not the TeamConnection GUI).
- For the PartDisconnect action, parentType is available for use with the TeamConnection web client (but not the TeamConnection GUI.)
- For the PartModify action, fileType is available for use with the TeamConnection web client (but not the TeamConnection GUI).
- For the PartUnlock action, Source Directory is available in the TeamConnection GUI but not in the TeamConnection web client.
- For the PartViewContents, Expand Keywords is available in the TeamConnection GUI but not in the TeamConnection web client.
- For the UserView Filter action, the following are available with the TeamConnection web client (but not the TeamConnection GUI):
 - pswStatus
 - pswModifyTime
 - pswCreateTime.

Chapter 8. The basics of using TeamConnection

All users of TeamConnection perform a number of basic tasks, such as checking parts out of TeamConnection and then back in, and testing and verifying part changes. Before you start doing these tasks, you need to understand the basic concepts behind them; that is what this chapter explains.

This chapter assumes that you have read “Chapter 1. An introduction to TeamConnection” on page 1 and are familiar with the different objects, such as components and releases.

Laying the groundwork

Someone has already created your family’s component structure, and those components manage your parts and control access to the data. Your TeamConnection family also contains releases. A release identifies a version of all the parts that comprise an application at a given point in time. When you create a release, you specify the component that will manage it. One component manages a release, but many components can manage the individual parts associated with that release.

A single part can be associated with more than one release, but it is managed by one component. When you create a part, you specify the release that you want to associate with the part and the component that you want to manage it. At any time, you can link the created part to other releases so that the part can be shared, or you can change its managing component.

Before you start working with parts, you need to be familiar with your family’s component structure. This will help you when trying to locate parts within TeamConnection and when writing defects and features. You can do the following to display your family’s component structure from the GUI:

1. Select **Components** → **Components** from the Objects pull-down menu on the Tasks window. The Component Filter window appears.
2. Type the name of the component that is at the top of your component hierarchy in the **Component** field, and select **OK**. Initially this component is called root. The Components window appears, listing the component.
3. Verify that the component is displayed in tree view (a plus sign (+) appears before the component name). If not, select **Tree** from the View pull-down menu.
4. Select **Expand fully** from the Selected pull-down menu.

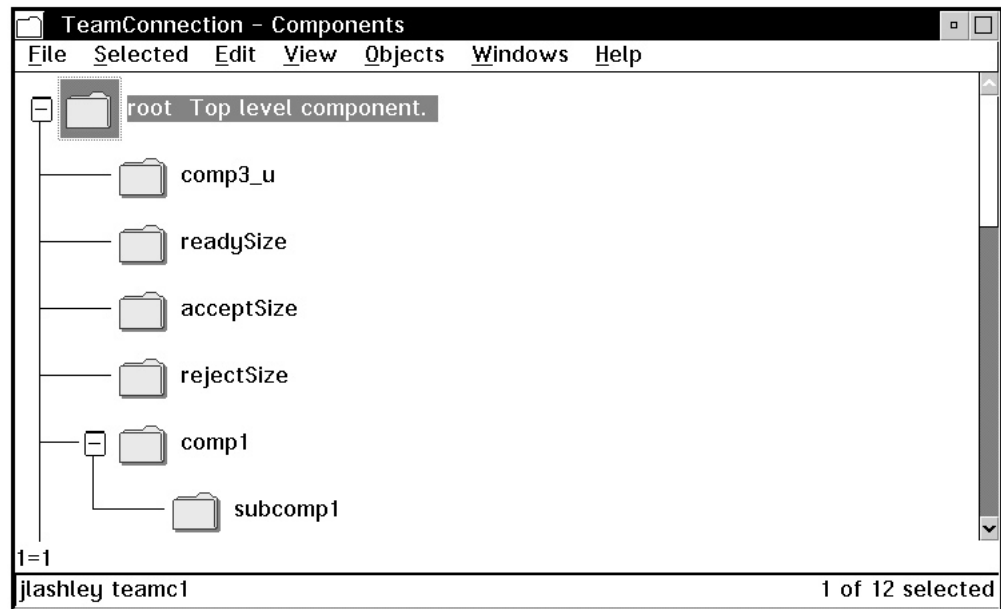


Figure 5. Components window

From a command prompt, you can issue the following command to view the component structure.

```
teamc report -view -raw bCompView -where "name='root'"
```

Authority to perform tasks

As a TeamConnection user, you are automatically given the authority to perform some basic tasks. You can:

- Open defects and features
- Add notes to existing defects and features
- Modify the information for your user ID
- Display information about any user ID
- *Search* for information within TeamConnection to create reports

You receive authority to perform additional actions when you become the owner of a TeamConnection object, such as a component or a part, or when authority is explicitly given to you by the component owners.

If you attempt an action that you do not have authority to do, TeamConnection tells you so. When this happens, you can ask the component owner, the family administrator, or a user with superuser authority to grant you the necessary authority.

Note: You can issue queries to generate reports of data from tables and views using the `— view` action flag. If you do not specify selection criteria, such as the fields and the search conditions you want to use, the report query selects all entries for the table or view indicated that the user has authority to access. This command does not show any objects in components that you are not authorized to access

Finding objects within TeamConnection

All TeamConnection objects are stored on a server in a database. To find one or more of these objects within a family, do one of the following:

- Use the report command with the -view action flag from a command line or a command line within TeamConnection.

Command usage is explained in the *Commands Reference*

- Use a Filter window in the GUI.

Online helps explain how to use the Filter windows.

For now, you need to understand that the database is case-sensitive. You need to refer to and search for objects in the correct case. For example, if a component is stored in the database as hand, you would not find it if you typed Hand or HAND. This is why it is important that your organization sets a naming convention, and that everyone follows that decision when creating objects. If you do not know what naming convention has been established for your organization, talk to your family administrator.

Note: It is recommended that you use lowercase as much as possible.

Finding parts

There are three Filter windows that you can use to find parts within TeamConnection:

Parts Use when you want to limit your search to a particular context of a work area or driver in a release, or a particular version of a release. This is generally the view users will use most often.

If you specify only a release, TeamConnection lists the committed parts for that release. However, if you want a list of all parts in a specified work area and release, TeamConnection displays all the parts visible to the work area. This includes parts that are committed to the release as well as changed parts that are visible only to the work area.

BuildView

Use when you want to search for information related to building your application, such as viewing a build tree, or when you want to do build actions.

PartFull

Use when you want to search for parts across releases, components, or work areas. For example, you want a list of all the optics.c parts. Unlike the Parts Filter, you can specify one or more release or work area names.

You can also use this filter to display only parts that have been changed in a work area. For example, you check out robot.c to work area 310:1, and that is the only part that you have changed. If you use the PartFull Filter to query for all the parts in work area 310:1, only one record is returned.

You cannot use this filter to search for build information.

Refer to the online help, in particular **How do I**, for more information on how to use the Filter windows. Select **How do I** from the Help pull-down menu to access the information.

Using work areas

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. You must create a work area before you can create, check out, or check in parts. If your component's process includes a design, size, review subprocess for defects or features and the release follows a tracking subprocess, a work area is automatically created when sizing records exist and the associated defect or feature is accepted. TeamConnection associates these work areas with the appropriate defect or feature.

The parts in a work area do not become available in the release until the work area is integrated. Also, if your release follows a driver subprocess, parts that have been changed do not become available in the release until the associated driver is committed. However, users who have the authority to access the work area can view and work with the parts in it.

You can save intermediate versions of the parts in your work area by *freezing* your work area. Every time you freeze a work area, TeamConnection saves a revision level of the work area. When you freeze work area 123:1, 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. Therefore, you should freeze a work area whenever there is a possibility that you will want to return to that version of the work area. For example, you might be adding a major feature to the code, and you want to be able to return to something that works in case the application no longer builds. When you integrate a work area or commit a driver, the work area is frozen automatically.

Naming your work areas

When TeamConnection automatically creates a work area, the work area is given the same name as the defect or feature it was created for plus the initial version number, :1. When you create a work area, you can also give it the same name as the defect or feature, or you could give it any other name. Where possible, we recommend that you name it after a defect or feature, or relate the name to the change that is being made.

Here are some things you should know before you name a work area:

- Work area names must be unique within the context of a release.
- After you create a work area, you cannot delete it. You can, however, cancel the work area in the following situations:
 - No part changes were made.
 - You undo the changes you made.
- With the proper authority, other users in your organization will be able to access your work area and make changes to the parts. This means that you need to make it easy for them to locate the work area. Following your local naming conventions will help.

- After the work area is integrated with the release, you cannot reuse the work area. If the defect is still in the working state, you can create another work area with a different name after the initial work area is integrated with the release.

Creating parts

A TeamConnection part is controlled by the TeamConnection server. A TeamConnection part is uniquely identified by the *path name* of the part, the *part type*, and the name of the release in which it is contained. You must specify both the release name and the path name whenever you perform a TeamConnection action on a part. Multiple releases can share the same part.

When you create a part, you do one of the following:

- Take an existing text or binary file that is on your workstation and place it into TeamConnection.
- Create an empty part that has no content. Empty parts are used as place holders until an application is built. For example, you can create a place holder for an executable part that will be created by a build.

After you put a part under TeamConnection control, the official copy of the part resides in the database. The copy on your workstation is changed to read-only mode. You can then change the part by checking it out to your workstation or editing it within the GUI.

Use the online help facility if you need assistance when creating parts.

Naming your parts

If your organization has a naming convention, be sure to follow it when naming your parts. When the naming convention is not followed, everybody in your organization can have trouble locating parts. Part names created on the server are case-sensitive; they must be retrieved using the same case in which they were created.

When you name TeamConnection parts, you can specify only the base name, such as `hand.c`, or you can specify the directory path in addition to the base name, such as `robot\hand\hand.c`. Specifying the path name as part of the name lets you have several identical base name parts included in the same release—for example, `robot\hand\msg.h` and `robot\optics\msg.h`.

You can also have identical part names within the context of a release as long as their part types are different, such as `TCPart` and `vgdata`.

Note: It is recommended that you use lowercase letters to name your parts.

Preparing to build your parts

If you are going to use the TeamConnection build function, you must provide certain information about each part that participates in a build. You can provide this information when you create the parts or wait until later. You can also change the information at any time.

To associate a part with a build, you must specify the following information:

- The parent part that you want to associate the part with.

- The type of relationship the part has to the parent, such as:

Input The part will be used as input to building its parent. An example of an input part is a C language source file, `x.c`, which is compiled to create its parent, `x.obj`.

Output The part will be a generated output from the same build that creates its parent part. In other words, both the parent part and this child part are outputs when the parent part is built.

Dependent The part will be needed for the build operation of its parent to complete, but it will not be passed directly to the builder. An example of this is an include file.

If you do not provide this information when you create the part, you can provide it later using the `connect` function.

You can also specify the builder or parser that a part will use, as well as any build parameters.

Working with parts

After the parts are created in TeamConnection, you will be working with these parts—getting them to your workstation so you can change them and then getting them back in to TeamConnection. This section gives a brief overview of these tasks, and go into more detail about these

Working in serial or concurrent development mode

A release is set up for either serial development or concurrent development mode. Once the development mode is set you can change from serial mode to concurrent mode, but not from concurrent mode to serial 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 two users have the same part checked out in concurrent development, both can change it. The first user integrates the work area, which contains the changed part, with the release. When the second user does the same, TeamConnection recognizes that the parts differ and notifies the user. It is up to this user to resolve the differences, using the TeamConnection merge program or some other merge program.

Before getting parts from TeamConnection, you might want to find out if the development mode for the release is concurrent or serial. To determine the mode, view the information about the specific release. To do this, select **View** from the Selected pull-down menu on the Releases window.

Working with common parts

A *common* part is a part with identical content that is shared by two or more releases or two or more work areas. For example, when an identical part is needed

in two separate releases, you can link the part from one release to the other (if you have the proper authority). Both releases would then have a link to the current version of that part.

When a common part is checked out of a release, TeamConnection locks the current version of the part in all releases if one of them uses serial development. When putting the part back into the release, one of the following actions reflects the change in all releases in which the part is common:

- You integrate the work area when the driver subprocess is not followed, or
- You commit the driver when the driver subprocess is followed.

You can break the common link if you make changes to a common part and you do not want these changes reflected in other releases or work areas that link to the part. You can break the common link when you check out, check in, rename, delete, re-create, connect, or *disconnect* parts. When a part is common to more than two releases, you can maintain the common link with some of the releases while breaking the link with other releases. When a link is broken, the parts still share the same name, but the information contained in the parts is different.

Parts can also be linked between two or more work areas in the same or different releases, making the parts common to those work areas. For example, a user working in one work area can link to the latest version of a part in another work area of the same release (the part has yet to be integrated with the release). The part is then common to the two work areas within the same release. If you want to maintain the common link to all work areas, you must specify the names of the common work areas when you check in, rename, delete, or re-create the parts. As with common parts in releases, you can break the common link.

You can also link all the parts within a release to another release. This function is especially helpful when development begins on a new release of a product, and you want the parts in the new release to initially be the same as the parts in the current release. As development of the two releases continues, the common link between the parts can be broken to separate development of the new release from maintenance of the current release.

For more information about how to link parts, refer to the *Commands Reference* and online help.

Getting parts from TeamConnection

Checking out a part implies that you intend to modify it; extracting a part merely gives you a copy of the part. Normally, when you extract a part, you do not plan to change the current version in TeamConnection.

You must have the necessary authority to a component before you can check out or extract parts from that component. You need *PartExtract* authority to extract a part from TeamConnection; you need *PartCheckOut* authority to check a part out.

Parts are checked out to work areas. The work area is where you store updated parts and do builds without affecting the version of the parts in the release. When a part is checked out of the release to the specified work area, TeamConnection locks the part in the release if you are working in serial development. If you are working in concurrent development, the part is never locked. TeamConnection also puts a copy of the part on your workstation. It is here where you update the part. If a read-only copy of the part exists on your workstation, the first character of its file

extension changes as follows: It becomes `$` for OS/2 and Windows platforms. It becomes a `.` for AIX and HP-UX platforms. This copy is a backup copy. If a backup copy already exists, it is deleted. When you are finished updating the part, you check it back in to the work area. A work area is optional when extracting a part.

When you extract a part, TeamConnection copies the part to your workstation, and the part is not locked. In other words, other users can still check out the same version of the part and make changes to it, even in serial development mode. By default, TeamConnection sets the extracted part to read-only access. This is done to keep you from inadvertently changing the part on your workstation when the part in TeamConnection is not locked. You can, however, change this in the Settings window or when you are extracting the part. When you do this, be aware that someone else can change the official part in TeamConnection, making your workstation copy back level.

Where TeamConnection places a checked-out or extracted part on your workstation depends on the following:

- Your workstation's *current working directory*
- Whether you use the `-relative` flag on the command line or the **Destination directory** field on the GUI
- Whether the `TC_TOP` environment variable is set

For more information about how these interact, refer to the part command examples in the *Commands Reference*

When you want to make changes to a part, you can do one of the following:

- Check out one or more parts and edit the parts on your workstation. When you finish making changes to the parts, you check them back in.
- Edit a part from within the TeamConnection GUI using a specified editor. When you exit the editor, the Check In Parts window appears and you can check the part back in to TeamConnection.

In either case, if you are working in concurrent development and someone else changed a part while you had it checked out, you are asked to resolve the differences when you try to integrate your work area.

Checking parts in to TeamConnection

After you have verified the accuracy of your part changes, you are ready to check them in to TeamConnection. Any parts that you have checked out, you have the authority to check back in.

As mentioned earlier, you check parts out to a work area so you can work on them. Therefore, when you check in a part, you must specify the work area where that part is checked out. In other words, you check the part back in to the same work area. When the part is checked in, the copy on your workstation is flagged read-only.

At this time, the changed part is visible in only the named work area; it is not visible at the release or to any other work area. This lets you test your changes by building the version of the code that is in your work area.

When you are satisfied with your changes, you can integrate the parts into the release by integrating your work area. This action makes the work area visible to all the users in the release.

If you are working in concurrent development mode, TeamConnection generates a *collision record* when a changed part conflicts with a previously committed part. For example, both you and Keith have `hand.c` checked out. Keith makes changes to the part and then integrates the work area that contains that part. (Depending on the process being followed, Keith might have to commit the work area rather than integrate it.) Later, after making changes to `hand.c`, you attempt to integrate the work area that contains the part. Because the part was already integrated by Keith, you are notified of a collision and asked to refresh your work area. After the refresh, you can view the collision record and decide how you want to resolve the conflicts. [Explains in more detail how this works.](#)

Finding different versions of TeamConnection objects

TeamConnection version control maintains different versions of the following objects:

- Releases
- Work areas (and driver members)
- Drivers
- Parts

When you want to find and retrieve previous versions of these objects, it is helpful to know how TeamConnection creates and deletes previous versions of each object.

Some basics of TeamConnection versioning will help you understand how TeamConnection identifies unique versions of objects:

- When you first create an object, the initial version name is the object name suffixed with `:1`. When you create a new work area called *myWorkArea*, for example, its version is *myWorkArea:1*. Subsequent versions are identified in numerical order: *myWorkArea:2*, *myWorkArea:3*, *myWorkArea:4*, and so on. Versions of releases and drivers are identified similarly: *myRelease:1*, *myRelease:2*, *myRelease:3*; *myDriver:1*, *myDriver:2*, *myDriver:3*; and so on.
- Unique versions of parts are identified by association with a specific version of a release, work area, or driver. Your TeamConnection family may have three different versions of a part called *myPart*, for example: one associated with release *myRelease:2*, one associated with work area *myWorkArea:1*, and one associated with work area *myWorkArea:2*.

Versioning releases

TeamConnection creates new versions of releases whenever you do the following:

- Create a release

This is the initial version of a release and contains no parts. When you create *myRelease*, for example, its version name is *myRelease:1* and it contains no parts.

- Commit a work area to the release

Committing a work area to a new release creates a new version of the release and adds the parts in the work area to the release. When you commit work area

myWorkArea:1, for example, to myRelease:1, TeamConnection creates a version of myRelease called myRelease:2. It also associates the parts in myWorkArea:1 with myRelease:2.

- Commit a driver to a release

Because drivers are simply collections of work areas, committing a driver to a release has the same effect as committing a work area: TeamConnection creates a new version of the release. When you commit myDriver:2 to myRelease:2, for example, TeamConnection creates a version of myRelease called myRelease:3.

TeamConnection deletes versions of releases whenever you prune the release. Refer to the *Administrator's Guide* for an explanation of pruning.

Versioning work areas

TeamConnection creates new versions of work areas whenever you do the following:

- Create a work area

This is the initial version of a work area. When you create myWorkArea, for example, its version name is myWorkArea:1.

- Refresh a work area

Refreshing a work area updates it with any new versions of parts that have been integrated with the release. When a workarea is refreshed, two versions of the workarea are created. One of the contents before the refresh and one with the contents after the refresh.

- Freeze a work area

Freezing a work area is like taking a snapshot of the work area. It preserves the parts as they are at a given point in time. If you create work area myWorkArea:1, add three new parts to it—called part1, part2, and part3—and then freeze it, your family contains a work area called myWorkArea:2, with part1, part2, and part3. The version name of each of these parts is myWorkArea:1. If you then alter part2 and freeze the work area again, your family contains the following:

- myWorkArea:1, with nothing in it
- myWorkArea:2 contains part1, part2, and part3 at version myWorkArea:1
- myWorkArea:3 contains part1 and part3 at version myWorkArea:1, and part2 at version myWorkArea:2

- Commit a work area

Committing a work area adds the parts in the latest version of the work area to the release. It also does the following:

- Creates a new version of the release
- Creates new versions of the parts in the release
- Deletes any intermediate versions of the work area

Using the previous example, if you commit myWorkArea:3 to myRelease:1, the following happens:

- TeamConnection creates a new version of myRelease called myRelease:2.
- TeamConnection creates new versions of the parts in myRelease:2.
- TeamConnection deletes myWorkArea:1, myWorkArea:2, and myWorkArea:3.

TeamConnection deletes versions of work areas whenever you commit them to a release. Once a work area has been committed, you can no longer use it for making part changes and you cannot create a new work area with the same name.

Deleting work area versions is controlled by the autopruning option of the release associated with the work area. By default, TeamConnection always deletes work area versions on commit, but you can change this option. Refer to the *Administrator's Guide* for an explanation of autopruning.

Versioning drivers

TeamConnection creates new versions of drivers whenever you do the following:

- Create a driver

When you create a new driver, TeamConnection makes two versions of it: myDriver:1, for example and myDriver:2.

- Add a work area (driver member) to a driver

If you add myWorkArea:1 to myDriver:2, for example, TeamConnection creates a new version of myDriver called myDriver:3.

- Freeze a driver

Freezing a driver is like taking a snapshot of the driver. It preserves the parts as they are when the driver is frozen. If you freeze myDriver:3, for example, TeamConnection creates a new version called myDriver:4.

- Refresh a driver

Refreshing a driver updates the driver with all changes that have been made in all of its driver members. Refreshing a driver actually creates two new versions of the driver, as follows:

1. Freezes the driver (so that TeamConnection can have a point to roll back to if an error occurs during the refresh operation).
2. Updates the driver with any changes from the driver members
3. Freezes the driver again, thus preserving a copy of the updated driver.

If the current version of myDriver is myDriver:2, for example, and the parts in its driver members have been changed, then TeamConnection does the following when it refreshes the driver:

1. Freezes myDriver, creating myDriver:3.
2. Updates myDriver with changes from its driver members.
3. Freezes myDriver again, creating myDriver:4.

The result of refreshing myDriver (version myDriver:2) is two new versions: myDriver:3, containing a snapshot of the driver before the refresh, and myDriver:4, containing a snapshot of the driver after the refresh.

TeamConnection deletes versions of drivers whenever you remove driver members or commit a driver to a release.

- If you have a driver version myDriver:4 with driver members myWorkArea, yourWorkArea, and ourWorkArea, and you remove myWorkArea, then TeamConnection deletes driver versions myDriver:2, myDriver:3, and myDriver:4 and creates a new driver version called myDriver:5 containing members yourWorkArea and ourWorkArea. As a result, the family contains two versions of the driver, myDriver:1 and myDriver:5.
- When you commit a driver to a release, all intermediate versions of the driver (resulting from driver member add, driver freeze, driver refresh, or driver member remove operations) are deleted.

Versioning parts

TeamConnection versions parts in association with other TeamConnection objects, such as work areas. If, for example, you create part1 in myWorkArea:1, the current version of part1 is myWorkArea:1. If part1 is in release myRelease:2 and work area myWorkArea:2, then you can view the version of the part for either the release or the work area. The version label for part1 in myRelease:2 is **myRelease:2** and in myWorkArea:2 is **myWorkArea:2**.

TeamConnection deletes part versions whenever it deletes versions of the object that the part is associated with. In addition to versioning in association with other TeamConnection objects, TeamConnection maintains versions of build output parts (parts that are created as the result of a build, such as an .exe file or a .hlp file). When you create a release, you can set the maximum number of versions of build output parts to maintain. If you set this maximum to 10, for example, then TeamConnection saves only 10 versions of build output parts and discards the oldest version each time a new version is created.

Working with defects and features

Defects are used to report problem information; features are used to record information about proposed design changes. After a defect or feature is opened, TeamConnection tracks the progress of the defect or feature through various states. To what degree defects and features are tracked depends on the processes followed by the release and component to which they are assigned. The following describes actions that your defined processes might require:

Analyzing defects and features

The owner is responsible for analyzing a defect or feature after it is opened. The owner can then return it if it is not valid or feasible, reassign it to another user or component, or accept it for resolution.

Designing the resolution

After a defect or feature has been accepted, the actual resolution needs to be designed so that an informed evaluation can be made. This resolution needs to be designed by users who are familiar with the product or area affected by the defect or feature.

Identifying the required resources

Sizing records are created by the owner to identify the components and releases that might be affected by the defect or feature. Each owner of a component that is referenced in a sizing record needs to evaluate the impact of the defect or feature on the parts managed by the component. If the defect or feature requires changes to parts, the sizing record is accepted and sizing information is added.

When sizing records exist and the associated defect or feature is accepted, TeamConnection automatically creates a work area.

Reviewing the design and resource estimates

After the resolution has been designed and the resources have been identified, the proposal needs to be reviewed. If the review indicates that work should continue on the defect or feature, it is accepted.

Resolving defects and implementing features

Resolving one defect or implementing one feature in one release can involve one or more users changing many parts. To change a part, a user must check out the part, make the changes required to resolve the problem

or implement the design change, and check the part back in. If the release follows a tracking process, all defects or features must be associated with a work area. Parts that are checked out refer to the work areas that are monitoring the defect or feature.

Resolving a defect or implementing a feature also involves integrating the changed parts with changes made for other defects and features in that release. All changed parts are eventually integrated with the unchanged parts within the release.

Verifying the resolution of the defect or feature

The originator uses a verification record to acknowledge that the defect or feature was satisfactorily resolved or not. Accepting a verification record formally closes the defect or feature. Rejecting a verification returns the defect or feature to working state.

“Chapter 9. The states of TeamConnection objects” on page 73 explains in more detail the various states that different TeamConnection objects can go through depending on the process that is being followed. A diagram in this chapter shows the flow of these states. You might want to study this information before you start to work with defects and features.

Testing and verifying part changes

You can use TeamConnection’s build function to build your program. Before you check in updated parts, you will probably want to verify the accuracy of your changes.

Chapter 9. The states of TeamConnection objects

The actions that you can perform on certain TeamConnection objects are controlled by two factors:

- The process followed by the component and by the release
- The current state of the object

Certain TeamConnection objects can follow certain states through their life cycle. An instance of an object might not go through all the possible states for that object—it moves through the states that are defined in the process followed by the component and by the release. The following table briefly lists the component and release subprocesses. For more information on component and release subprocesses, refer to the *Administrator's Guide*

Component subprocesses

dssDefect

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

verifyDefect

Verify that defect fixes work

dssFeature

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

verifyFeature

Verify that feature changes work

Release subprocesses

track Relate all part changes to a specific defect or feature and a specific release

approval

Require all changes to be approved before incorporating them into a release

fix Use fix records to ensure that all required changes are made

driver Use drivers to integrate changes into a release

test Require all changes to be tested before they are integrated into the release

This chapter explains the possible states of certain TeamConnection objects and how objects are moved from one state to the next. It also explains how component and release subprocesses affect the flow of states. For a diagram showing the flow of states, refer to the poster *Staying on Track with TeamConnection Processes*.

Defects and features

Use defects to report problem information; use features to record information about proposed design changes. After you open defect or feature, TeamConnection tracks the progress of the defect or feature through various states. Defects and features are tracked according to the processes followed by the release and component that they are assigned to. The possible states for defects and features are:

Open state

When you open a defect or feature, it is in the open state and you are considered the originator.

You assign the defect or feature to a component. The owner of this component becomes the feature or defect owner and is responsible for managing its resolution. The component you open a defect or feature against should be one that manages the parts affected by the enhancement or problem. Use the component descriptions and the structure of your family's hierarchy to find the most appropriate component. If you open a defect or feature in an inappropriate component, the component owner can reassign it.

While the defect or feature owner is responsible for implementation, the originator is responsible for verifying that the defect or feature is resolved correctly.

Returned state

A defect or feature owner can return a defect or feature to its originator. You can return a feature or defect from the open, design, size, or review state if you decide that the defect or feature is not feasible or not valid. You can return a defect or feature back to the working state only if it has no associated work areas. If there are associated work areas, you must cancel or undo them before you can return the defect or feature. When you return a defect or feature, add your reason for returning it so that the originator and any other users can evaluate why you believe it is not feasible or not valid.

Canceled state

A feature or defect in the open or returned state can be canceled only by its originator or by a superuser. A canceled defect or feature remains inactive unless it is reopened by the originator.

Design state

If the component to which a defect or feature is assigned includes the `dsrDefect` or `dsrFeature` subprocess, you move defects or features in the open or returned state to the design state.

In this state, the proposed change is designed, and a description of the design change is entered. The owner must describe the design change before the defect or feature can move to the next state.

If the release includes the fix subprocess, fix records are automatically created when a defect or feature is designed.

Size state

Defects or features move to this state after the owner enters design information.

In this state, users can create a *sizing record* for each release that contains parts affected by the enhancement or problem. A sizing record identifies the work that is required for and the resources affected by the defect or feature. The owner of the component that is referenced in the sizing record is the owner of the sizing record. The owner is responsible for entering information about the amount of work that is required to implement the feature or resolve the problem.

The sizing record owner can reject the sizing record if it does not affect the specified component. After all sizing records are either accepted or rejected, the defect or feature moves to the review state or returns to the design state if more design information is needed.

Review state

Defects or features move to this state after they have been sized. In this

state, the design text and sizing records are reviewed to determine the feasibility of the proposal. The owner can do one of the following:

- Accept the defect or feature if all design and sizing records are acceptable. This moves the defect or feature to the working state.
- Return the defect or feature to the originator if all design and sizing records are not acceptable. If necessary, the originator can reopen a defect or feature.
- Move the defect or feature back to the design state if design modifications are needed.

Working state

Defects or features move to this state when the owner accepts the defect or feature when it is in the:

- Review state, if the component includes the `dsrDefect` or `dsrFeature` subprocess
- Open state, if the component does not include the `dsrDefect` or `dsrFeature` subprocess

When you accept a defect or feature, you accept the responsibility of resolving it. A defect or feature might require changes in more than one release.

What happens after a defect or feature is accepted varies according to the subprocesses in effect:

Component subprocesses

dsrDefect or dsrFeature

TeamConnection creates a work area in the approve state for each release identified in the accepted sizing records for the defect or feature.

verifyDefect or verifyFeature

TeamConnection creates verification records in the notReady state.

Release subprocesses

fix TeamConnection creates fix records in the notReady state based on the sizing records.

approval

TeamConnection creates approval records for each user on the release's approver list.

If the component does not include the `dsrDefect` or `dsrFeature` subprocess, then you must manually create a work area before you can check out or create parts to address the defect or feature.

Verify state

Defects and features go through the verify state only if their component includes the `verifyDefect` or `verifyFeature` subprocess. Defects and features are automatically moved to this state when one of the following happens:

- All work areas (there can be multiple work areas for the defect or feature) for the release are integrated, if a release is specified when the defect or feature is created

When a defect or feature is accepted, TeamConnection creates a *verification record*. This record lets the originator:

- Accept the fix if the resolution was satisfactory
- Reject the fix if not satisfied with the resolution
- Abstain if unable to assess the resolution

Once all verification records have been accepted or abstained, the defect or feature moves to the closed state. If a verification record is rejected, the defect or feature returns to the working state. The defect or feature cannot be closed until the verification records are accepted.

A defect or feature can have more than one verification record. For example, if defect 123 is returned because it is a duplicate of defect 122, a second verification record is created for defect 122. The originator of defect 123 is the owner of the second verification record for defect 122. If the originator is the same for both defects, only one verification record is created.

Note: For a discussion of verification records and test records, see “Verification and test records” on page 80.

Closed state

The closed state is the final state of a defect or feature.

If the defect is associated with multiple work areas, the defect will remain in the working state until all of the work areas are integrated.

If the component includes the `verifyDefect` or `verifyFeature` subprocess, the defect or feature automatically moves to the closed state after all verification records are in the accept or abstain state and all work areas are in the complete state. If a verification record is rejected, the defect or feature moves back to the working state. Otherwise, the defect or feature moves directly from the working state to the closed state when the first work area moves to the complete state.

You cannot re-open a defect or feature that is in the closed state. If the defect or feature was not resolved correctly, you must open a new defect or feature to address the necessary changes.

The states of work areas

A work area is a storage area where you can work on the parts in a release without affecting the “official” versions of those parts. A work area can be associated with a specific defect or feature, but it does not have to be.

Approve state

When a work area is created, it goes to this state if the release includes the approval subprocess. TeamConnection creates an approval record for each user on the release’s *approver list*. Each approver indicates their evaluation of the changes in their approval record:

- Accept that work should continue
- Abstain if unable to assess if work should continue
- Reject if work should not continue

When all approval records are marked as abstain or accept, the work area goes automatically to the fix state. If any approval record is marked as reject, the state of the work area remains at approve. You can change rejected approval records to accept or abstain.

Fix state

If the release does not include the approval subprocess, work areas for the release begin in the fix state.

While the work area is in this state, parts are checked out to the work area, changes are made to these parts, and builds are done to verify the accuracy of the changes.

If the release includes the fix subprocess, any fix records created for a defect or feature move to the active state when a part change is associated with the work area for the defect or feature. A fix record monitors the part changes within a single component. Fix records provide a mechanism for reviewing all part changes that apply to components before integrating those changes with changes made for other defects and features.

How fix records are handled varies according to the subprocesses in effect:

Component subprocesses

dsrDefect or dsrFeature

TeamConnection creates fix records for features or defects when existing sizing records are accepted.

Release subprocesses

fix If a fix record does not already exist for the component, TeamConnection creates one when a part managed by that component is checked in to the database.

If neither of these subprocesses are in place and a defect or feature owner needs to create a work area manually, he or she can create fix records at the same time. Existing fix records go to the active state when a part is checked in to the work area.

Fix records provide a way of ensuring that all necessary part changes within the specified component have been made and are reviewed or inspected. The fix record owner is responsible for this review. When the fix record owner is satisfied that the part changes made within that component are complete and ready for integration with other parts in the release, the owner marks the fix record as complete. When all existing fix records for a work area are complete, the work area automatically moves to the integrate state.

Integrate state

Work areas can be moved to the integrate state as follows. If the release includes the fix and driver subprocesses, the work area automatically moves to the integrate state when all fix records are complete. If all fix records are not complete, you can force a work area to the integrate state, provided that no part changes are associated with the work area. If the release does not include the fix and driver subprocesses, you must move the work area to the integrate state manually.

While a work area is in integrate state, you must add it to an existing driver as a driver member if the release includes the driver subprocess. All work areas in the integrate state do not have to be added to the same driver. The work area stays in the integrate state until the driver in which it is a member is committed.

You can move work areas from the integrate to the following states, according the subprocesses in effect:

Release subprocesses

driver A work area moves to the commit state when the driver it is a member of is committed, or to the restrict state when the driver is restricted. You can also force a work area to the commit state, provided that no part changes are associated with the work area.

test A work area moves to the test state so that test records can be approved or rejected.

If the release does not include these subprocesses, you can manually complete a work area in the integrate state.

Restrict state

Work areas can be moved to the restrict state only when the release includes the driver subprocess. The work area moves automatically to the restrict state when the driver to which it belongs is restricted. If a work area in this state is removed from the driver, it returns to the integrate state. Otherwise, the work area remains in the restrict state until the driver to which it belongs is committed.

Commit state

Work areas can be moved to the commit state only when the release includes the driver subprocess. The work area moves automatically to the commit state when the driver to which it belongs is committed. At this point, all parts that were changed in this release to resolve the feature or defect are committed. The work area remains in the commit state until the driver to which it belongs is completed.

Test state

Work areas can be moved to the test state only when the release includes the test subprocess. When the associated driver moves to the complete state or when a work area is committed without a driver, the work area moves to the test state. The driver is then ready for formal testing in the specified environments. Test records for the work area are created in the ready state when the work area moves to the test state. The work area stays in the test state until all test records are accepted, rejected, or abstained.

Complete state

The complete state is the final state of a work area; the work area can no longer be used. If the test subprocess is not included in the release process, the work area moves directly to this state when the associated driver is completed or when the work area is explicitly integrated.

When a work area is completed, the feature or defect associated with that work area automatically moves to the verify or complete state. The defect does not leave the working state until the work area for that release is completed.

The states of drivers

Drivers monitor and implement the integration of part changes within a release. Those part changes are included in a driver by adding the work areas containing the changed parts to the driver as driver members.

Working state

The working state is the initial state of a driver. While the driver is in this state, it is not associated with any work areas and, therefore, contains no part changes.

If the release includes the driver subprocess, drivers can be explicitly created at any time.

Integrate state

Each driver automatically moves to the integrate state when the first work area is added to it as a driver member. If all work areas are removed from the driver, the driver automatically returns to the working state.

Work areas can be added to drivers as driver members when the driver is in the working, integrate, or restrict state and the work area is in the fix state. Adding driver members to a driver in restrict state requires proper authority.

You can extract the driver when it is in the integrate state; however, only those parts that were changed in reference to driver members are extracted. This is referred to as extracting a *delta part tree*.

Restrict state

Before a driver is committed, you can move it to the restrict state. While a driver is in this state, work areas in the integrate state can be created for or deleted from the driver by only a superuser or an individual with the special authority of memberCreateR or memberDeleteR. This allows a build administrator to have better control over what is being built. The build administrator can delete driver members that are causing build errors or add driver members to fix build errors. You can then commit an error-free driver.

When a driver moves to the restrict state, all work areas that are included as driver members also move to the restrict state.

Commit state

Committing a driver commits all work areas included as driver members and all parts that were changed in reference to those work areas.

TeamConnection commits only a successfully built driver. Committing a driver changes it to the commit state. You can, however, manually commit the driver. You can also commit an unsuccessful driver by using the force option.

When a driver moves to the commit state, all work areas that are included as driver members also move to the commit state. When a work area is in the commit state, all part changes associated with the work area become the "official" versions of the parts in the release and are visible to all users of the release.

A committed driver can be extracted as a full part tree as well as a delta part tree. A full part tree is the part structure of all the parts within the release.

Complete state

The complete state is the final state of a driver. In this state, the driver is ready for formal testing in the specified environments.

If the release includes the test subprocess, the work areas that are included as driver members move to the test state. Any existing test records for the

work area move to the ready state when the work area moves to the test state. The work area stays in the test state until all test records are accepted, rejected, or abstained.

Test records are used to record the outcome of environment tests for changes implemented in a driver. This record lets the owner:

- Accept the record if the test was satisfactory
- Reject the record if not satisfied with the test results
- Abstain if unable to assess the results

Once all test records have been accepted or abstained, the states of other objects change as follows:

Work areas

Go to complete state

Defects and features

Go to verify state if the component includes the `verifyDefect` or `verifyFeature` subprocess; otherwise they go to the closed state.

Verification records

Go to ready state and are sent to the defect or feature originators

If the test subprocess is not configured, then work areas move to the complete state and any defects or features move to the verify state.

If the component includes a `verifyFeature` or `verifyDefect` subprocess, verification records move to the ready state and notification is sent to the originators of any defects or features that were addressed in the completed driver.

The commit and complete states of drivers differ as follows:

- When a driver is committed, all work areas are committed, but no changes occur in the states of defects or features associated with the work areas.
- When a driver is completed, then the states of other associated objects (such as test records, work areas, verification records, defects, and features) change according to the other subprocesses in effect:

test Work areas go to the test state and test records are created in the ready state for each environment in the release's environment list.

verify Verification records go to the ready state.

If the release includes neither of these subprocesses, then the work area goes to the complete state and all features and defects associated with the work area are closed.

Verification and test records

If you use both the verify component subprocess (`verifyDefect` or `verifyFeature`) and the test release subprocess, then TeamConnection creates both verification records for features or defects and test records for each environment defined in the release's environment list. These records serve different purposes:

- Verification records provide a means of accepting or rejecting the product changes made in response to defects or features and are thus specific in nature.

- Test records provide a means of accepting or rejecting the results of a build and are more global in nature.

These records are handled by different people and enable you to monitor your development progress in different ways. The sequence of creating and handling verification and test records is as follows:

1. Verification records are created in the notReady state when a defect or feature is accepted. This indicates that someone on the development team has begun implementing the changes warranted by the defect or feature, but the changes are not yet ready to be verified. A work area is also created for the part changes.
2. When a driver is committed all part changes associated with the driver members are integrated into the release.
3. To create test records, the driver is completed. This action creates one test record for each environment on the release's environment list. The testers on your development team use the test records to accept or reject the results of their tests on the part changes.
4. After all test records have been accepted or abstained, the verification records are moved to the ready state. This indicates that the part changes have been tested in the context of the build and each individual defect or feature is ready to be accepted or rejected by the person who opened it.
5. The defect or feature originator accepts or abstains the verification record to close the defect or feature. The originator can also reject the verification record to move the defect or feature back to working state.

Customer support

Your options for IBM VisualAge TeamConnection support, as described in your License Information and Licensed Program Specifications, include electronic forums. You can use the electronic forums to access IBM VisualAge TeamConnection technical information, exchange messages with other TeamConnection users, and receive information regarding the availability of fixes. The following forums are available.

- **IBM Talklink**

Use the TEAMC CFORUM. For additional information about TalkLink, call

- United States 1-800-547-1283
- Canada 1-800-465-7999 ext. 228

- **CompuServe**

From any ! prompt, type GO SOFSOL, then select TeamConnection. For additional information, call 1-800-848-8199 and ask for representative 239.

- **Internet**

Go to the IBM homepage, <http://www.ibm.com>. Use the search function with keyword TeamConnection to go to the TeamConnection area.

If you cannot access these forums, contact your IBM representative.

There are several other support offerings available after purchasing IBM VisualAge TeamConnection. For a list of these offerings, please contact your IBM representative.

Bibliography

IBM VisualAge TeamConnection library

The following is a list of the TeamConnection publications.

- **License Information (GC34-4497):**
Contains license, service, and warranty information.
- **Administrator's Guide (GC34-4551):**
Lists the hardware and software that are required before you can install and use the IBM VisualAge TeamConnection product, provides detailed instructions for installing and configuring the TeamConnection family and build servers, and provides instructions for administering a TeamConnection family.
- **Getting Started (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.
- **TeamConnection 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**
 - TeamConnection User's Guide**
 - Commands Reference**
 - Quick Commands Reference**
 - Staying on Track with TeamConnection Processes**

Tool Builder's Development Kit

The following publications are part of the Tool Builder's Development Kit feature:

- **Tool Builder's Development Guide (SC34-4553):**
Explains how to create and extend tools for accessing objects in the TeamConnection database. It contains guidance and reference information.
- **Information Model Reference (SC34-4554):**
Details the TeamConnection information model. This publication is available in softcopy only.

TeamConnection Technical reports

- 29.2147**
SCLM Guide to TeamConnection Terminology
- 29.2196**
Using REXX command files with TeamConnection MVS Build Scripts
- 29.2231**
TeamConnection Interoperability with MVS and SCLM
- 29.2235**
Using REXX command files with TeamConnection MVS Build Scripts for PL/I programs
- 29.2253**
Comparison between CMVC 2.3 and TeamConnection 2
- 29.2254**
Migrating from CMVC 2.3 to TeamConnection 2
- 29.2267**
TeamConnection frequently asked questions: how to do routine operating system tasks

ObjectStore

The following publications are part of the ObjectStore library of documents and are available for order from Object Design, Inc. To order these documents call (617) 674-5000, Monday through Friday from 8:30 AM to 5:30 PM Eastern Time.

- **ObjectStore C++ Installation:**
Contains step-by-step procedures for installing the latest release of ObjectStore on a specific platform:
 - 310-100-40 I**
UNIX
 - 310-310-40 I**
Windows
 - 310-320-40 I**
OS/2
- **ObjectStore C++ API User Guide (310-000-40 U):**
Provides information about the application programming interface for application programmers.
- **ObjectStore C++ API Reference (310-000-40 R):**
Describes the API to the features provided by ObjectStore for application programmers.
- **ObjectStore C++ Building Applications (310-000-40 B):**
Provides information and instructions for compiling code, generating schemas, and linking files using all supported compilers; and provides instructions for developing ObjectStore client applications for use on multiple platforms.
- **ObjectStore Management (310-000-40 M):**
Provides information and instructions for performing management tasks on ObjectStore server and client systems. It includes server parameters, environment variables, and database utilities.
- **ObjectStore C++ Performance (310-000-40 P):**

Explains the fundamentals of designing and tuning ObjectStore applications for optimal performance.

IBM Exchange library

The publications listed below can be ordered as a set (SBOF-6098) or separately as indicated below. IBM Exchange will be available at a later date.

- *Licensed Programming Specification (GC34-4525):*
- *Installation Guide (SC34-4509):*
- *Bridge Builder's Guide (SC34-4508):*
- *User's Guide 1 (SC34-4506):*
- *User's Guide 2 (SC34-4507):*

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 name of a 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 agent. A program that handles access to persistent data on behalf of the build processor. Each build agent is connected to one and only one build processor, through a TCP/IP connection.

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 processor. A program that invokes the tools, such as compilers and linkers, that construct an application. Each build processor is connected to one and only one build agent, through a TCP/IP connection. See also *build agent* and *build cache*.

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. The combination of a build processor and a build agent. See also *build agent* and *build processor*.

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), and = (equal to).

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 agent 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- and OS/2-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 an AIX 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

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. In OS/2 and Windows, 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.

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.

Note: Parts created through an API can have other specified types. Refer to the *Commands Programming Reference* for more information.

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 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

description 11, 19, 27

access command 56

AIX

hardware requirements for AIX client 11

software requirements for AIX client 11

approval command

purpose of 56

approve state 76

approver command 56

authority

basic 60

for checking in parts 66

for checking out parts 65

for extracting parts 65

B

build action, definition of 4

build administrator, responsibilities 10

build function

definition of 8

builder command

purpose of 56

building an application

preparing your parts 63

buildView action, using 61

C

canceled state 75

change control, definition of 1

check-in action, definition of 4

check-out action, definition of 4

checking in parts

authority needed to 66

explanation of 66

checking out parts

authority needed to 65

explanation of 65

client machine

AIX

hardware requirements 11

software requirements 11

definition 3

HP

hardware requirements 19

software requirements 19

installing 39

OS/2

hardware requirements 35

software requirements 35

Solaris

hardware requirements 27

software requirements 27

starting 41, 50, 51

stopping 52

client machine (*continued*)

Windows 3.1

hardware requirements 43

software requirements 43

Windows 95

hardware requirements 43

software requirements 43

Windows NT

hardware requirements 43

software requirements 43

closed state 75

collision command

purpose of 56

collision record

example of 67

command line interface

using 55

viewing syntax online 55

commands

becoming familiar with 55

client

description of 56

teamc.log file 55

verify connection to server 41, 50

verify environment variables 41, 50

viewing syntax online 55

teamcgui 51

commit state

of drivers 79

of work areas 78

common parts

between releases 65

between work areas 65

breaking link 65

definition of 64

locking 65

complete state

of drivers 79

of work areas 78

component command 56

components

definition of 4

displaying structure of 59

example of hierarchy 4

information stored about 5

Components window 59

concepts of

TeamConnection 2

concurrent development

example of 67

how to work in 64

configuration management, definition of 1

connect function 64

coreq command, purpose of 56

create action, definition of 4

D

- defect command
 - purpose of 56
- defects
 - definition of 7
 - states of
 - canceled state 75
 - closed state 75
 - design state 75
 - open state 75
 - return state 75
 - review state 75
 - size state 75
 - verify state 75
 - working state 75
 - working with 70
- delta part tree, definition of 79
- dependent part, description of 64
- design changes, reporting 70
- design state 75
- development mode 64
- disk space, checking for 40, 49
- driver command
 - purpose of 56
- driverMember command
 - purpose of 56
- drivers
 - definition of 7
 - states of
 - commit state 79
 - complete state 79
 - integrate state 79
 - restrict state 79
 - working state 78
 - versioning 69

E

- edit action, definition of 4
- editing parts 66
- environment command, purpose of 57
- environment variables
 - manually updating for client 40, 50
 - setting for command line usage 55
 - setting for GUI usage 53
- examples of
 - client/server network 2
 - component hierarchy 4
 - showing part/release/component relationship 5
- expand keywords 53
- extract action, definition of 4
- extracting parts
 - authority needed 65
 - versus checking out 65

F

- family 3
- family administrator, responsibilities 9
- feature command 57

features

- definition of 7
- states of
 - canceled state 75
 - closed state 75
 - design state 75
 - open state 75
 - return state 75
 - review state 75
 - size state 75
 - verify state 75
 - working state 75
- working with 70

files

- teamc.log 55
- updating hosts file for client 37
- updating services file for client 37

filter windows for parts 61

finding objects 61

fix command

- purpose of 57

fix state 76

freezing work areas

- explanation of 62

full part tree, definition of 79

G

GUI

- client
 - accessing online help 54
 - fast path 53
 - Settings notebook 53
 - starting 51
 - stopping 52
 - Tasks window 51
 - using 51

H

hardware requirements

- for AIX client 11
- for HP-UX client 19
- for OS/2 client 35
- for Solaris client 27
- for Windows 3.1 client 43
- for Windows 95 client 43
- for Windows NT client 43

help, online

- diagram push button 54
- how do I 54
- how to access 54

hierarchy

- component example 4
- displaying component structure 59

host command 57

hosts file

- updating for client 37

how do I help, description of 54

HP

- hardware requirements for HP-UX client 19
- software requirements for HP-UX client 19

I

- information model 1
- input part, description of 64
- installation of client
 - installing 39
 - instructions 39
 - preparing for 12, 19, 27, 36, 44
 - verify connection to server 38
 - verify installation 41
- integrate state
 - of drivers 79
 - of work areas 77
- interfaces
 - becoming familiar with 51
 - description of 3

L

- LAN installation
 - of client 39

N

- naming
 - parts 63
 - work areas 62
- network, example of client/server 2
- NLSPATH
 - setting for client 40, 50
- notify command 57

O

- online help
 - diagram push button 54
 - how to access 54
- open state 75
- OS/2
 - hardware requirements for OS/2 client 35
 - software requirements 35
- output part, description of 64

P

- packaging 9
- parser command
 - purpose of 57
- part command
 - purpose of 57
- partFull action, using 61
- parts
 - authority needed to check in 66
 - authority needed to check out and extract 65
 - checking in 66
 - checking in versus integrating 66
 - checking out versus extracting 65
 - common
 - between releases 64, 65
 - between work areas 65
 - breaking link 65
 - definition of 64

parts (*continued*)

- creating 63
- definition of 4
- dependent 64
- editing parts 66
- empty 63
- extracting 66
- finding
 - using BuildView action 61
 - using Filter window 61
 - using PartFull action 61
 - using Parts action 61
 - using report command 61
- input 64
- linking 64
- linking between releases 65
- linking between work areas 65
- locked 64
- naming 63
- providing build information 63
- searching for 61
- versioning 70
- where placed on workstation 66

- parts action, using 61
- placeholder parts 63
- prereq command 56
- problem information, reporting 70

- processes
 - definition of 7
 - relating to defects and features 70

R

- relative flag 66
- release command 57
- release management, definition of 1
- releases
 - common parts 65
 - definition of 5
 - example of relationship with other objects 5
 - linking parts between releases 65
 - parts common to more than one 64
 - versioning 67
- report command
 - purpose of 57
 - to find parts 61
- restrict state
 - of drivers 79
 - of work areas 78
- retrieving past versions of objects 67
- return state 75
- review state 75

S

- searching for objects 61
- serial development
 - how to work in 64
- servers
 - definition 2
 - family server
 - definition 2

- services file
 - updating for client 37
- Settings notebook
 - for client 53
 - list of variables 53
- size command, purpose of 57
- size state 75
- software requirements
 - for AIX client 11
 - for HP-UX client 19
 - for OS/2 client 35
 - for Solaris client 27
 - for Windows 3.1 client 43
 - for Windows 95 client 43
 - for Windows NT client 43
- Solaris
 - hardware requirements for Solaris client 27
 - software requirements for Solaris client 27
- starting
 - GUI client 51
- states of objects
 - defects 73
 - drivers 78
 - features 73
 - work areas 76
- superuser 3
- syntax
 - how to view online 55
- system administrator, responsibilities 9

T

- tasks
 - authority to perform 60
 - understanding the basics 59
- Tasks window 51
- TC_BECOME
 - setting for client 40, 50
- TC_COMPONENT 55
- TC_FAMILY 55
 - setting for client 40, 50
- TC_TOP 66
- TC_USER
 - setting for client 40, 50
- tlogin command 55
- TCP/IP
 - updating files 12, 20, 28, 37, 44
- teamcgui command 51
- TeamConnection
 - concepts of 2
 - introducing 1
 - the basics of using 59
- test command
 - purpose of 57
- test state
 - of work areas 78
- testClient command 41, 50
- testServer command 41, 50

U

- user command 57

V

- verification record, when created 75
- verify command 57
- verify state 75
- version control, definition of 1, 67
- versions 67
 - of drivers 69
 - of parts 70
 - of releases 67
 - of work areas 68

W

- window examples
 - BuildView 61
 - Components window 59
 - PartFull 61
 - Parts Filter 61
 - Tasks 51
- Windows 3.1
 - hardware requirements 43
 - software requirements 43
- Windows 95
 - hardware requirements 43
 - software requirements 43
- Windows NT
 - hardware requirements 43
 - software requirements 43
- work area
 - automatic creation of 62
 - canceling 62
 - definition of 6
 - freezing 62
 - linking parts between work areas 65
 - naming 62
 - reusing 62
 - states of
 - approve state 76
 - commit state 78
 - complete state 78
 - fix state 76
 - integrate state 77
 - restrict state 78
 - test state 78
 - things you can do with 6
 - using 62
 - versioning 68
 - when parts become visible to others 62
 - when to create one 62
- workarea command
 - purpose of 57
- working state
 - of defects and features 75
 - of drivers 78



Part Number:



Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC34-4552-02

