

IBM VisualAge TeamConnection



Commands Reference

Version 3.0

IBM VisualAge TeamConnection



Commands Reference

Version 3.0

Third Edition (December 1996)

Note

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

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, 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	xvii
Trademarks	xix
About this book.	xxi
How this book is organized	xxi
Conventions	xxi
Tell us what you think	xxii
Chapter 1. General command information	1
TeamConnection commands	1
Flags	2
Action flags	2
Attribute flags	3
Flag arguments	3
Using standard input for arguments	6
Environment variables	6
Setting environment variables	11
Authority requirements	11
Base authority	11
Superuser privilege	11
Implicit authority	12
Explicit authority	12
Restricted authority	12
How to read syntax statements	12
Chapter 2. Access	15
Command summary	15
access -create	15
Attributes	15
Examples	16
access -delete	16
Attributes	16
Examples	17
access -restrict	18
Attributes	18
Examples	18
Related information	19
Chapter 3. Approval	21
Command Summary	21
approval -abstain	22
Attributes	22
approval -accept	22
Attributes	22
Examples	23
approval -assign	23
Attributes	23
Examples	24
approval -create	24
Attributes	24
Examples	25
approval -delete	25

Attributes.	25
Examples	26
approval -reject	26
Attributes.	26
Related information	27
Chapter 4. Approver	29
Command Summary	29
approver -create	29
Attributes.	29
Examples	30
approver -delete	30
Attributes.	30
Examples	31
Related information	31
Chapter 5. Builder	33
Command summary	33
builder -create Name	34
Attributes.	34
builder -delete Name	36
Attributes.	36
builder -extract Name	37
Attributes.	37
builder -modify Name	37
Attributes.	37
builder -view Name	39
Attributes.	39
Related information	40
Chapter 6. Collision	41
Command summary	41
collision -accept	42
Attributes.	42
Examples	43
collision -reconcile	43
Attributes.	43
Examples	44
collision -reject	45
Attributes.	45
Examples	46
Related Information	46
Chapter 7. Component	47
Command summary	47
component -create Name	48
Attributes.	48
Examples	49
component -delete Name	49
Attributes.	49
Examples	50
component -link Name	50
Attributes.	50
Examples	50
component -modify Name	50
Attributes.	51

Examples	51
component -recreate Name	52
Attributes.	52
Examples	52
component -unlink Name	52
Attributes.	52
component -view Name	53
Attributes.	53
Examples	53
Related information	54
 Chapter 8. Coreq	 55
Command summary	55
coreq -create	55
Attributes.	55
Examples	56
coreq -delete	56
Attributes.	56
Examples	57
Related information	57
 Chapter 9. Defect	 59
Command summary	59
defect -accept Name	60
Attributes.	60
Examples	61
defect -assign Name	61
Attributes.	61
Examples	62
defect -cancel Name	62
Attributes.	63
Examples	63
defect -configInfo.	63
Attributes.	63
Examples	64
defect -design Name	64
Attributes.	64
defect -modify Name	65
Attributes.	65
Examples	68
defect -note Name	68
Attributes.	68
defect -open	69
Attributes.	69
Examples	71
defect -reopen Name	71
Attributes.	71
Examples	72
defect -return Name	72
Attributes.	72
Examples	73
defect -review Name	73
Attributes.	73
defect -size Name	74
Attributes.	74
defect -verify Name	75

Attributes.	75
defect -view Name	75
Attributes.	75
Examples	76
Related information	76
Chapter 10. Driver	77
Command Summary	77
driver -assign Name	78
Attributes.	79
Examples	79
driver -check Name	79
Attributes.	79
Examples	80
driver -commit Name	80
Attributes.	80
driver -complete Name	81
Attributes.	81
driver -create Name	82
Attributes.	82
Examples	82
driver -delete Name	82
Attributes.	83
driver -export Name	83
Attributes.	83
Examples	84
driver -extract Name	84
Attributes.	84
Examples	85
driver -freeze Name	86
Attributes.	86
driver -modify Name	86
Attributes.	86
driver -refresh Name	87
Attributes.	87
driver -restrict Name	88
Attributes.	88
Examples	88
driver -view Name	88
Attributes.	89
Examples	89
Related information	89
Chapter 11. DriverMember	91
Command summary	91
driverMember -create	91
Attributes.	92
Examples	93
driverMember -delete	93
Attributes.	93
Examples	94
Related information	94
Chapter 12. Environment	95
Command summary	95
environment -create Name	95

Attributes.	95
Examples	96
environment -delete Name	96
Attributes.	96
Examples	97
environment -modify Name	97
Attributes.	97
Examples	97
Related information	98
Chapter 13. Feature	99
Command summary	99
feature -accept Name	100
Attributes.	100
Examples	101
feature -assign Name	101
Attributes.	101
Examples	102
feature -cancel Name	102
Attributes.	102
Examples	103
feature -configInfo	103
Attributes.	103
Examples	104
feature -design Name	104
Attributes.	104
Examples	105
feature -modify Name	105
Attributes.	105
feature -note Name	107
Attributes.	107
feature -open	107
Attributes.	108
Examples	109
feature -reopen Name	109
Attributes.	109
Examples	110
feature -return Name	110
Attributes.	110
feature -review Name	111
Attributes.	111
feature -size Name	111
Attributes.	111
Examples	112
feature -verify Name	112
Attributes.	112
feature -view Name	113
Attributes.	113
Examples	113
Related information	114
Chapter 14. Fix	115
Command summary	115
fix -activate	116
Attributes.	116
Examples	117

fix -assign	117
Attributes.	117
Examples	118
fix -complete	118
Attributes.	118
fix -create	119
Attributes.	119
Examples	120
fix -delete	120
Attributes.	120
Related information	121
 Chapter 15. Host	123
Command summary	123
host -create Name	124
Attributes.	124
Examples	124
host -delete Name	125
Attributes.	125
Examples	125
Related information	126
 Chapter 16. Notify	127
Command summary	127
notify -create	127
Attributes.	127
Examples	128
notify -delete	128
Attributes.	128
Examples	129
Related information	129
 Chapter 17. Parser	131
Command summary	131
parser -create Name	131
Attributes.	131
parser -delete Name	132
Attributes.	132
parser -modify Name	133
Attributes.	133
parser -view Name	134
Attributes.	134
Related information	134
 Chapter 18. Part	135
What you can do with parts	135
Working with parts	136
Specifying the full path name of a TeamConnection part	136
Working from your current working directory	137
Using the -relative flag.	138
Using the TC_TOP variable	138
Common parts in releases	139
Command summary	140
part -build Name	143
Attributes.	143
Examples	145

part -build Name -cancel	145
Attributes.	145
part -checkin Name	146
Attributes.	146
Examples	148
part -checkout Name	148
Attributes.	148
Examples	149
part -childInfoView Name	149
Attributes.	149
part -configInfo	150
Attributes.	150
Examples	151
part -connect Name	151
Attributes.	151
Examples	152
part -create Name	153
Attributes.	153
Examples	156
part -delete Name	156
Attributes.	156
part -disconnect Name	157
Attributes.	157
part -export Name	158
Attributes.	158
Examples	159
part -extract Name	159
Attributes.	159
Examples	161
part -link Name	161
Attributes.	161
Examples	162
part -lock Name	162
Attributes.	162
part -modify Name	163
Attributes.	164
Examples	165
part -recreate Name	165
Attributes.	166
part -refresh Name	167
Attributes.	167
Examples	168
part -rename Name	168
Attributes.	168
Examples	169
part -resolve Name	169
Attributes.	170
part -touch Name	170
Attributes.	170
part -undo Name	171
Attributes.	171
Examples	172
part -unlock Name	173
Attributes.	173
Examples	174
part -view Name	174

Attributes.	174
Examples	175
part -viewmsg Name	175
Attributes.	175
Related information	176
Chapter 19. Prereq.	177
Command summary	177
prereq -create Name	177
Attributes.	177
prereq -delete Name	178
Attributes.	178
Related information	179
Chapter 20. Release	181
Command Summary	182
release -create Name	183
Attributes.	183
Examples	185
release -delete Name	185
Attributes.	185
release -export Name	185
Attributes.	186
release -extract Name	186
Attributes.	186
Examples	188
release -link Name	188
Attributes.	188
Examples	189
release -modify Name	189
Attributes.	189
Examples	191
release -prune Name	191
Attributes.	191
release -recreate Name	192
Attributes.	192
release -view Name	193
Attributes.	193
Related information	193
Chapter 21. Report.	195
Command summary	195
report -help	196
Attributes.	196
report -testClient	196
Attributes.	197
report -testServer	197
Attributes.	197
report -userExitInfo	198
Attributes.	198
Examples	198
report -view Name	199
Attributes.	200
Examples	201
report -view partView	204
Attributes.	205

Related information	206
Chapter 22. Size.	207
Command summary	207
size -accept.	207
Attributes.	208
Examples	208
size -assign.	209
Attributes.	209
Examples	209
size -create	210
Attributes.	210
Examples	211
size -delete	211
Attributes.	211
size -reject	212
Attributes.	212
Examples	212
Related information	213
Chapter 23. Tclogin	215
Command summary	215
tclogin -login	215
Attributes.	215
Examples	216
tclogin -logout	216
Attributes.	216
Examples	216
tclogin -view	216
Examples	216
Related information	217
Chapter 24. Test.	219
Command summary	219
test -abstain	220
Attributes.	220
Examples	220
test -accept	221
Attributes.	221
Examples	222
test -assign	222
Attributes.	222
Examples	223
test -create	223
Attributes.	223
Examples	224
test -delete	224
Attributes.	224
Examples	225
test -reject	225
Attributes.	225
Related information	226
Chapter 25. User	227
Command summary	227
user -create.	227

Attributes.	228
Examples	228
user -configInfo	228
Attributes.	229
Examples	229
user -delete Name	229
Attributes.	229
Examples	230
user -modify Name	230
Attributes.	230
Examples	231
user -recreate Name	232
Attributes.	232
user -view Name	232
Attributes.	232
Examples	233
Related information	233
 Chapter 26. Verify	235
Command summary	235
verify -abstain	235
Attributes.	235
verify -accept	236
Attributes.	236
Examples	237
verify -assign	237
Attributes.	237
Examples	238
verify -reject	238
Attributes.	238
Examples	238
Related information	239
 Chapter 27. Workarea	241
Command summary	242
workarea -assign Name	243
Attributes.	243
Examples	244
workarea -cancel Name	244
Attributes.	244
workarea -check Name	244
Attributes.	245
Examples	245
workarea -commit Name	246
Attributes.	246
workarea -complete Name	247
Attributes.	247
workarea -create	247
Attributes.	247
Examples	248
workarea -create {-defect -feature}	248
Attributes.	249
Examples	249
workarea -create -defect -feature	250
Attributes.	250
workarea -export Name	251

Attributes.	251
Examples	252
workarea -fix Name	252
Attributes.	252
Examples	252
workarea -freeze Name	253
Attributes.	253
Examples	253
workarea -import Name	253
Attributes.	254
Examples	254
workarea -integrate Name	254
Attributes.	254
Examples	255
workarea -modify Name	255
Attributes.	255
workarea -refresh Name	256
Attributes.	256
Examples	256
workarea -test Name	256
Attributes.	257
workarea -undo Name	257
Attributes.	257
workarea -view Name	258
Attributes.	258
Examples	259
Related information	259
 Appendix. Querying the TeamConnection database	 261
Constructing queries	261
Rules for defining queries	261
Views and report output	262
AccessDownView**	263
AccessFastView	263
AccessNInheritView	263
AccessTable	264
AccessUpView	264
AccessView**	264
Approvals	265
ApprovalView**	265
Approvers	266
ApproverView**	266
Authority**	266
BchangeView**	266
BcompView**	267
BpartView**	267
Builders	269
BuilderView**	269
BuildView**	270
Cfgcomproc**	270
Cfgrelproc**	271
ChangeExtractView	271
ChangeView**	271
Changes	272
Collectors	272
Collisions	273

CollisionView**	273
CompMembers	273
CompMemberView	274
Components	274
CompView**	274
Config**	275
Coreqs	275
CoreqView**	275
DefectDownView**	276
Defects	277
DefectView**	278
DriverMembers	279
DriverMemberView**	279
Drivers	280
DriverView**	280
Environments	280
EnvView**	281
FeatureDownView**	281
FeatureView**	282
Files**	283
Fix	283
FixDownView	284
FixView**	284
History	285
HistoryView	285
Hosts	285
HostView**	286
Interest**	286
Notes	286
NoteView**	286
Notification	287
NotifyDownView**	287
NotifyUpView**	287
NotifyView**	288
OVersions	288
Parsers	288
ParserView**	289
PartFullView**	289
Parts	290
PartsOut	290
PartsOutView**	290
PartView**	291
Path	292
Prereqs	292
PrereqView**	293
Releases	293
ReleaseView**	294
Sequence	295
Sizes	295
SizeView**	295
Tests	296
TestView**	296
Tracks	297
Users**	297
Verify	297
VerifyView**	298

Versions	298
VersionView**	299
Workareas	299
WorkAreaView**	299
Customer support	301
Bibliography	303
IBM VisualAge TeamConnection library	303
Tool Builder's Development Kit.	303
TeamConnection Technical reports	304
DB/2	304
ObjectStore.	305
IBM Exchange library	305
Related publications	306
Glossary	307
Index	315

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
CUA	OS/2
C/370	TalkLink
IBM	TeamConnection
MVS	VisualAge
MVS/ESA	XGA

The following terms are trademarks of other companies:

ObjectStore

ObjectStore Design, Inc.

UNIX X/Open Company Limited

Solaris

Sun Microsystems, Inc.

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

About this book

This book is part of the documentation library supporting the IBM TeamConnection licensed program. It describes all of the TeamConnection commands you can use from the command line interface.

You should read the *TeamConnection User's Guide*, SC34-4499, before you use the TeamConnection product. It introduces the fundamentals of the *configuration management*, *version control*, *change control*, and *problem tracking* features in the TeamConnection licensed programs. It also defines the concepts that are the foundation of TeamConnection *actions* and establishes their interrelationships.

You should be familiar with your operating system because you access the TeamConnection licensed programs through that environment.

How this book is organized

Read "Chapter 1. General command information" on page 1 for an overview of the TeamConnection commands and an explanation of the command syntax.

The remaining chapters, which are in alphabetical order according to command name, describe the commands in detail. Each chapter describes one command and includes:

- Description of the command and an overview of its purpose
- Syntax statements of the command, one statement per action flag
- Action flags you can use with the command
- Attribute flags that apply to the command
- Examples of the command
- Related information

The appendix describes the fields for various TeamConnection views and tables, including the order that is output by the `-raw` option of the `report` command.

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. General command information

This chapter introduces the TeamConnection commands that you can issue from the command line. This chapter also does the following:

- Outlines the *authority* required to issue commands
- Explains how to use action and attribute flags with the commands
- Explains how to use TeamConnection environment variables
- Explains how to read the command syntax found in this book

TeamConnection commands

When you type TeamConnection commands, start with `teamc`, followed by the command, the action flag and argument, and the attribute flag and argument.

The purpose of each TeamConnection command is shown in the following table:

Table 1. TeamConnection commands

Command	Purpose
Access	Updates the <i>access lists</i> for a <i>component</i> , identifying the <i>user IDs</i> that have <i>explicit authority</i> to perform actions on it.
Approval	Marks <i>approval records</i> with <i>approvers'</i> opinions about proposed changes in a <i>release</i> .
Approver	Updates the list of approvers for a release.
Builder	Creates and maintains <i>builders</i> , which are used in the <i>build</i> 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 <i>family</i> . Defines a component hierarchy.
Coreq	Identifies <i>work areas</i> as <i>corequisites</i> , that is, work areas that must be included in the same <i>driver</i> or 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 <i>environment list</i> for a release, identifying the test <i>environments</i> and the names of <i>testers</i> .
Feature	Monitors the suggestion, evaluation, and implementation of design changes and enhancements.
Fix	Marks the <i>fix records</i> for a component identifying user IDs that receive notification of actions on the component.
Host	Identifies <i>client</i> access on the <i>host list</i> associated with a user ID.
Notify	Identifies notification interest for user IDs using component notification lists.
Parser	Creates and maintains <i>parsers</i> . Use this for the build function.
Part	<ul style="list-style-type: none">• Places <i>parts</i> in the TeamConnection environment and lets users work with them.• Starts or stops the build function.• Manages the <i>build tree</i>.
Release	Creates and maintains releases to group project-related parts.
Report	Searches database tables for information on TeamConnection objects.
Size	Updates the <i>sizing records</i> for <i>defects</i> and <i>features</i> .
Tclogin	Logs users in and off of TeamConnection families that use password security.
Test	Updates environment <i>test records</i> , identifying testers' opinions about test results.
User	Creates user IDs and maintains information about the owners.
Verify	Updates <i>verification records</i> , indicating the outcome of defects and features.

Introduction

Table 1. TeamConnection commands (continued)

Command	Purpose
Workarea	Creates, maintains, <i>freezes</i> , and <i>refreshes</i> work areas.

For information about commands used for the family and build servers, such as `teamcd`, `notifyd`, `teamcpak`, `teamagnt`, and `teamproc`, refer to the *User's Guide* and the *Administrator's Guide*

Flags

Two types of flags are associated with commands: action flags and attribute flags. You can type the names of flags in any order on the command line.

A flag is a negative (-) or a positive (+) symbol followed by a word, often in lowercase, on the command line. The positive or negative symbols associated with each flag cannot be interchanged. If a flag has a positive symbol, it can only have a positive symbol associated with it.

You can abbreviate both action and attribute flags; however, the number of letters required to make a flag unique within a command depends on the names of all of the other flags, both action and attribute flags, associated with that command.

For example, you can abbreviate the verbose flag by typing `verb`. However, you could not abbreviate it with `ver` because that would not differentiate it from the version flag, whose minimum abbreviation is `vers`.

Action flags

Every command has action flags associated with it. These action flags represent the actions that you can perform for a command. When you use the command line to perform a TeamConnection action, you must specify one command and only one action flag. You do not have to type the action flag directly after the command.

For example, you can perform six actions using the `user` command. Each of these tasks requires one of the following action flags:

teamc user -configInfo
Displays configurable field properties for users

teamc user -create
Creates a new user ID

teamc user -delete
Deletes an existing user ID

teamc user -recreate
Recreates a previously deleted user ID

teamc user -modify
Changes information related to a user ID

teamc user -view
Displays current information for a user ID

Attribute flags

Some action flags have required attribute flags associated with them; others have optional attribute flags.

For example, the `-login` and `-address` attribute flags are required when you use the `-create` action flag for the `user` command. The other attribute flags are optional.

```
teamc user -create -login billyb -address williams@vroom1.raleigh.ibm.com
-name "William Bronson" -area Dept450 +super
```

You get the same results if you rearrange the order of the flags and abbreviate some of them.

```
teamc user -login billyb -name "William Bronson" -ad
williams@vroom1.raleigh.ibm.com -create -ar Dept450 +super
```

Syntax indicates the attribute flags that are required.

Flag arguments

In most cases, you must type additional information for an action or attribute flag. This additional information is an argument. The seven types of arguments, their format and their restrictions, are listed in the following table:

Table 2. Flag arguments

Argument	Format	Example	Restrictions
Date	yyyy/mm/dd	1995/04/29	Only numbers separated by slashes are permitted.
Name	Alphanumeric string	42tool	You cannot use blanks, vertical bars (), or ASCII control characters.*
Name ...	One or more alphanumeric strings	prod1 prod2 prod3	You cannot use vertical bars () or ASCII control characters.* Blanks are permitted to separate unique strings.
Number	Numeric string ²	823	You must use numbers. Blanks are not permitted.
Number . . .	One or more numeric strings ³	411 1124 1 362	You must use numbers. Blanks are permitted to separate unique strings.
Octal_Number	Numeric string	750	You must use numbers from 0 to 7. Blanks are not permitted.
Text	Alphanumeric strings enclosed in quotation marks	"Not able to verify."	You cannot use vertical bars () or ASCII control characters.*

Note:

* For information on the ASCII control characters you use, refer to your operating system documentation.

² When used with a `-defect` or `-feature` flag, an alphanumeric string is acceptable and, consequently, the restrictions for *Name* apply.

Introduction

³ When used with a -defect or -feature flag, one or more alphanumeric strings are acceptable and, consequently, the restrictions for *Name* . . . apply.

If the positive (+) or negative (-) symbol is the first character of the argument then the symbol must be entered twice. For example: to display numeric string **+1234** as an argument, it must be entered as ++1234.

If you specify a list of arguments for more than one flag, the action is performed for every possible combination of arguments. For example:

```
teamc fix -create -workarea 1 2 -release one two
```

creates four fix records, as shown in the following table.

Work area	Release
1	one
1	two
2	one
2	two

Null keyword

You can use the keyword `null` on some of the TeamConnection -modify flags to reset attribute characters to zero. For example, you can reset the reference field for a defect as follows:

```
teamc defect -modify 247 -reference null
```

Using `null` in the preceding example changes the contents to zero characters; thus, the reference field is blank.

The following table lists the commands and attributes that accept the `null` keyword.

Table 3. TeamConnection attributes that accept null

Command	Attribute Flag(s)
Builder	-parameters
Component	-description
Defect	-driver -environment -reference -release
Feature	-reference
Parser	-include
Part	-builder -parameter -parser
User	-area -name
Workarea	-target

Octal numbers for directory and file permissions

The driver -extract Name . . ., part -extract Name . . ., and release -extract Name . . . actions use -dmask and -fmask attributes to enable you to set directory and file access permissions for parts that you extract from TeamConnection. These attributes take as their argument an octal number that sets three categories of permissions:

- User

- Group
- Others

For each category, there are three types of permission:

- Read
- Write
- Execute

Permissions are expressed in octal notation as shown in the following table:

Table 4. Values for file and directory permissions

Value	Read	Write	Execute
0	no	no	no
1	no	no	yes
2	no	yes	no
3	no	yes	yes
4	yes	no	no
5	yes	no	yes
6	yes	yes	no
7	yes	yes	yes

Permissions for all three categories are expressed using a 3-digit number in octal notation:

- The left number represents user access.
- The middle number represents group access.
- The right number represents access for others.

For example, parts that are created using TeamConnection have permissions 644. This octal number represents read/write permission for user, read permission for others in the user's group, and read permission for all other users. Permission 764 represents read/write/execute permission for user, read/write permission for others in the user's group, and read permission for all other users.



3.1

For parts that are extracted to a Windows or OS/2 system, only the file permissions specified for the user category are relevant. The execute file permission is ignored, as are all part permissions for the group and others categories.

Introduction

Using standard input for arguments

To specify an argument using standard input, use a negative symbol (-) as the argument type. You can specify only one flag per command in this way. The following example of standard input from a keyboard illustrates the remarks flag argument:

```
teamc defect -open -component debugr -sev 3 -remarks -
```

Press Enter to create additional lines on which to type the text. When you are finished entering the text, press Enter to create a new line and then press Ctrl Z to end standard input.

In the following example of standard input from a part, the -remarks argument is equivalent to the contents of the part you specified:

```
teamc defect -open -component debugr -sev 3 -remarks - < \tmp\defect.des
```

Environment variables

You can set environment variables to describe the TeamConnection environment in which you are working. You are not required to set your TC_FAMILY environment variable for the TeamConnection client command line interface. However, if the TC_FAMILY environment variable is not set, the -family must be specified for every client command. See “Setting environment variables” on page 11 for more information about setting environment variables.

The names of the TeamConnection environment variables, the purpose they serve, the equivalent TeamConnection flag, the equivalent Settings notebook field, and the TeamConnection component that uses the environment variable are listed in the following table.

You can override the value you set for an environment variable by using the corresponding flag in a TeamConnection command. When an environment variable has a Settings notebook equivalent, TeamConnection uses the two as follows:

- The environment variable controls the command line interface.
- The Settings notebook controls the graphical user interface.

If there is no Settings notebook equivalent for the environment variable, then the environment variable takes effect regardless of the interface you are using.

To see a list of current environment variable settings, you can issue the following command from a command prompt:

```
teamc report -testServer
```

This command returns information like the following:

Connect to Family Name:	ptest
Server TCP/IP Name:	amachine.company.com
Server IP Address:	9.1.23.45
Server TCP/IP Port Number:	9999

Server Specific Information	-----
Product Version:	3.0.0
Operating System:	AIX
Message catalog language:	English

```

Server Mode: non-maintenance
Authentication Level: HOST_ONLY
TC_RELEASE|v300
TC_FAMILY|ptest@amachine.company.com@9999

```

Table 5. TeamConnection environment variables

Environment variable	Purpose	Flag	Setting	Used by
LANG	Specifies the language-specific message catalog.			Client, family server
NLSPATH	Specifies the search path for locating message files.		NLS path	Client, family server
OS_NETWORK	Specifies the networking protocols used by ObjectStore servers and clients. Set during installation.			ObjectStore
OS_ROOTDIR	Specifies the path where the TeamConnection server is installed. Set during installation.			ObjectStore
OS_TMPDIR	Specifies where ObjectStore is to place temporary files. Set during installation. This variable is used only in Windows 32s environments.			ObjectStore
PATH	Specifies where tcadmin is to search for the family create utilities.			Family server
TC_ALLOWTRACKFIX	Allows users to add works areas in fix state to drivers.			Family server
TC_BECOME	Identifies the user ID you want to issue TeamConnection commands from, if the user ID differs from your login. You assume the access authority of the user ID you specify.	-become	Become user	Client, build server
TC_BUILDENVIRONMENT	Specifies the build environment name, such as OS/2 or MVS. The value you specify here can be anything you like, but it must exactly match the environment specified for a builder in order for the builder to use this build agent. This value is case-sensitive.	-e		Client

Introduction

Table 5. TeamConnection environment variables (continued)

Environment variable	Purpose	Flag	Setting	Used by
TC_BUILDOPTS	Specifies build options for sending build log file messages to the screen, and setting the logging level. Possible values are TOSCREEN, and VERBOSE. If you do not specify any of these options, then the build server writes build messages to the build log file (teamcbld.log), and writes a minimum level of messages to the log file.	-C, -S, -I		Build server
TC_BUILDPOOL	Specifies the build pool name.	-pool	Pool	Build server
TC_BUILD_RSSBUILDS_FILE	Specifies the name of startup files to be used to provide information about build servers to the build process.			Build server
TC_BULKCHUNKSIZE	Identifies the size, in bytes, of the storage blocks used to store binary parts in ObjectStore. Do not use, unless you are trying to tune database performance with guidance from IBM service.			Family server
TC_CASESENSE	Changes the case of the arguments in commands, not in queries.		Case	Client
TC_CATALOG	Specifies a specific file for the TeamConnection message catalog. Sometimes, depending upon the operating system environment, the catalog open command will only look in a particular directory for the catalog. If the host is running multiple versions of TeamConnection, this variable may be used. To set this environment variable, specify the file path name of the message catalog as in the following example: TC_CATALOG="/family/msgcat/teamc.cat"			Family server
TC_COMPONENT	Specifies the default component.	-component	Component	Client, make import tool
TC_DBPATH	Specifies the database directory path.			Family server

Table 5. TeamConnection environment variables (continued)

Environment variable	Purpose	Flag	Setting	Used by
TC_FAMILY	Identifies the TeamConnection family you work with.	-family	Family	Build server, client, family server, make import tool
TC_MAKEIMPORTRULES	Specifies the name of the rules file that TeamConnection uses when importing the makefile data into TeamConnection. If you set this environment variable, then you do not have to use the /u option with the fhomigmk command. Specify the full path name of the rules file. If neither this environment variable nor the /u option is used, TeamConnection uses default rules.			Make import tool
TC_MAKEIMPORTTOP	Strips off the leading part of the directory name when importing parts into TeamConnection. For example, you have parts with the following directory structure: g:\octo\src\inc\. To create these parts without the g:\octo structure, you can set TC_MAKEIMPORTTOP=g:\octo before you invoke the make import tool. The parts created in TeamConnection have the directory structure of src\inc\.			Make import tool
TC_MAKEIMPORTVERBOSE	Causes the -verbose flag to be added to part commands created by fhomigmk.			Make import tool
TC_MIGRATERULES	Specifies the name of a file containing the rules to be applied for migration of makefiles if the name is not supplied on the fhomigmk command line as a parameter.			Client
TC_NOTIFY_DAEMON	An alternate way of starting notifyd with the teamcd command. If you set this environment variable, then you do not have to use the -n option with the teamcd command. Specify the full path name of the mail exit to use with notifyd.			Family server

Introduction

Table 5. TeamConnection environment variables (continued)

Environment variable	Purpose	Flag	Setting	Used by
TC_RELEASE	Specifies a release.	-release	Release	Client, make import tool
TC_SYSTEM_LOG	Specifies where syslog messages are to be written. Specify the full path name of the file to use for syslog messages. The default is syslog.log.			Family server
TC_TOP	Specifies the source directory.	-top	Top	Client
TC_TRACE	Specifies the variable that lets the user designate which parts should be traced. You should modify this only when directed to do so by an IBM service person. Otherwise it is set to null. To trace all parts, specify TC_TRACE=*.			Client, family server, build server
TC_TRACEATTEMPTS	Specifies maximum number of failed trace attempts accepted before giving up. You should modify this only when directed to do so by an IBM service person.			Client, family server
TC_TRACEDELAY	Specifies the amount of time, in seconds, that TeamConnection waits, when a trace attempt fails, before attempting another trace. The default is 1 second. You should modify this only when directed to do so by an IBM service person.			Client, family server
TC_TRACEFILE	Specifies the output (part path and name) of the trace that the user designates using TC_TRACE. The default trace file name is tctrace. For the MVS build server, the default trace file is stdout.			Client, family server
TC_TRACESIZE	Specifies the maximum size of the trace file in bytes. If the maximum is reached, wrapping occurs. The default is one million bytes.			Client, family server

Table 5. TeamConnection environment variables (continued)

Environment variable	Purpose	Flag	Setting	Used by
TC_USER	Specifies the user login ID for single-user environments OS/2, Windows 3.1, and Windows 95. This environment variable is not used in multiuser environments AIX, HP-UX, Solaris, and Windows NT.		User ID	Client, build server
TC_WORKAREA	Specifies the default work area name.	-workarea	Work area	Client, make import tool
TC_WWWPATH	Specifies the path for the HTML helps and image files for Web client.	-workarea	Work area	Client, family server

Setting environment variables

For methods of setting your environment variables, refer to your operating system documentation. For example, for OS/2 you can use the following command to set the TC_FAMILY environment variable:

```
SET TC_FAMILY=familyName@hostname@portnumber
```

Authority requirements

Different authority requirements are attached to each of the actions in the TeamConnection product. Five types of authority control the actions you can or cannot perform.

Base authority

If you have a valid TeamConnection user ID, you can perform these unrestricted *base authority* actions:

- Open defects and features
- Modify the information associated with your user ID
- View information associated with any user ID
- Add notes to existing defects or features
- Generate reports

Superuser privilege

If you have been granted TeamConnection *superuser privilege* by a *family administrator* or someone else with superuser privilege, you can perform all possible actions in your TeamConnection family. Only a superuser can do the following:

- Create a user ID
- Create the first host list entry for a user ID
- Delete a user ID
- Recreate a user ID
- Grant superuser privilege to a user ID

Introduction

Implicit authority

You have implicit authority to perform actions on the basis of ownership. For example, if you open a defect, you are the *originator* of the defect and have the implicit ability to perform certain actions, such as canceling the defect or verifying its outcome. Similarly, if you own a component, a release, or a feature, you have implicit authority related specifically to those roles. You have implicit authority until you relinquish ownership of the object in question.

Explicit authority

Explicit authority is specified for a user for a component. Granting explicit authority is a method for limiting who can perform certain actions for the component.

Your family administrator can group sets of actions according to access authority groups. You are assigned to one or more of these groups by a component owner. The authority groups you belong to for any given component are inherited for all descendant components unless the components are restricted.

For specific actions used to create access authority groups, refer to the *TeamConnection User's Guide*. For a list of the preconfigured authority groups shipped with this product, which your family administrator might use, refer to the *TeamConnection User's Guide*.

Restricted authority

A user can be restricted from performing certain actions in a specific component by someone with AccessRestrict authority who wants to control which users inherit authority from the *parent components*. The authority can be restricted for specific users in the authority group or for all users in the group. The TeamConnection product notifies you if your specific authority is restricted.

Note: *Restricted authority* is not inherited, and it does not affect implicit authority or superuser privilege.

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

How to read syntax statements

The style conventions listed in the following table apply to the command syntax:

Table 6. Style conventions for TeamConnection syntax statements

Style	Usage
First letter of an argument is capitalized.	You must supply these values. For example, Name or Text.
. . .	Parameter can be repeated on the command line. For example, <code>-login Name</code> . . . means that you can enter more than one argument for the <code>-login</code> flag.
[]	Optional parameters are enclosed in square brackets. For example, <code>[-description Text]</code> .

Table 6. Style conventions for TeamConnection syntax statements (continued)

Style	Usage
{ }	There is more than one parameter choice, but one is required.
	Choose one parameter only. [a b] indicates that you can choose a or b, or neither a nor b. {a b} indicates that you must choose either a or b.
–	Standard input.

Note: The syntax statements in this document use a single asterisk (*) to denote the default and a double asterisk (**) to denote a restriction. Do not type the asterisk as part of the command.

Introduction

Chapter 2. Access

Use the **access** command to create entries on the component access list, delete entries from this list, and restrict authority for entries (including those normally inherited from ancestor components) on the list. Each entry associates a user ID with an access authority group. The authority group specifies the set of actions a user ID has the authority to perform in relation to the component. A family administrator can modify existing authority groups and define new ones. For details of the access authority groups shipped with the TeamConnection product, refer to the *TeamConnection User's Guide*.

A user ID can have more than one entry on the access list for a given component. A user ID inherits explicit authority from all ascendant components for that component and has the accumulation or superset of all authority groups defined for those ascendant components, unless the authority is restricted.

You cannot grant another user any access authority that is not defined for your own user ID.

Command summary

In the following syntax statements, the first letter of each argument is capitalized, for example `-family Name`. Replace these arguments with a value that is valid for your TeamConnection database. If you want to issue a command for the family named **testfam**, for example, replace `-family Name` with `—family testfam`.

The syntax statements for the **access** command are:

```
teamc access -create -login Name ... -authority Name -component Name
               -family Name [-become Name] [-verbose]
```

```
teamc access -delete { -login Name ... | -inherited } -authority Name
                    -component Name -family Name [-become Name] [-verbose]
```

```
teamc access -restrict { -login Name ... | -inherited } -authority Name
                    -component Name -family Name [-become Name] [-verbose]
```

access -create

Adds entries to a component access list.

Attributes

The `access -create` command takes the following attributes:

-authority Name

A preconfigured access authority group for the user ID. For more details about the access authority groups shipped with TeamConnection, refer to the *TeamConnection User's Guide*.

-become Name

Access

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-component Name

The component associated with the object. Different components can manage different versions of the same object.

(Environment variable: TC_COMPONENT.)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-login Name . . .

One or more TeamConnection user IDs. Specify only one of -inherited or -login.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

The following command gives **developer** access authority to TeamConnection user ID **barbara**, for the **graphix** component and all its descendants. This command creates an entry on the access list associated with the **graphix** component giving user ID **barbara** the authority to perform all actions included in the **developer** access authority group.

```
teamc access -create -login barbara -authority developer -component graphix
```

The following command gives **writer** access authority to TeamConnection user IDs **barbara** and **john** for the **graphix** component and all its descendants. This command creates two entries on the access list associated with the **graphix** component.

```
teamc access -create -login barbara john -authority writer -component graphix
```

access -delete

Deletes entries from a component list or deletes the restriction on authority for entries on a component access list.

Attributes

The access -delete command takes the following attributes:

-authority Name

A preconfigured access authority group for the user ID. For more details about the access authority groups shipped with TeamConnection, refer to the *TeamConnection User's Guide*.

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-component Name

The component associated with the object. Different components can manage different versions of the same object.

(Environment variable: TC_COMPONENT.)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-inherited

All users in the TeamConnection family who inherit the specified authority group from all parents of the specified component. Specify only one of -inherited or -login.

-login Name . . .

One or more TeamConnection user IDs. Specify only one of -inherited or -login.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

The following command removes **writer** access from user ID **barbara** for the **graphix** component. This command deletes the entry from the access list.

```
teamc access -delete -login barbara -authority writer -component graphix
```

The following command removes the restricted authority group **releaselead** for all users inheriting this access authority group from all parents of the **confidential** component. This command deletes the entry from the access list and permits users with the **releaselead** authority in parent components of the **confidential** component to perform all actions in the **releaselead** access authority group at the **confidential** component.

```
teamc access -delete -inherited -authority releaselead  
-component confidential
```

access -restrict

Restricts authority (including inherited authority) for entries on a component access list.

Attributes

The access -restrict command takes the following attributes:

-authority Name

A preconfigured access authority group for the user ID. For more details about the access authority groups shipped with TeamConnection, refer to the *TeamConnection User's Guide*.

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-component Name

The component associated with the object. Different components can manage different versions of the same object.

(Environment variable: TC_COMPONENT.)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-inherited

All users in the TeamConnection family who inherit the specified authority group from all parents of the specified component. Specify only one of -inherited or -login.

-login Name . . .

One or more TeamConnection user IDs. Specify only one of -inherited or -login.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

The following command restricts the actions in the access authority group **developer+** for the TeamConnection user ID **richard** in the **graphix** component. This command creates an entry on the access list associated with the **graphix** component restricting the user ID **richard** from performing all actions included in the **developer+** access authority group.

```
teamc access -restrict -login richard -authority developer+  
-component graphix
```

The following command restricts the actions in the access authority group **releaselead** for all users inheriting the **releaselead** access authority group from all parents of the **confidential** component. This command creates an entry on the access list associated with the **confidential** component restricting all users who inherited the **releaselead** access authority group from performing actions in the **releaselead** access authority group.

```
teamc access -restrict -inherited -authority releaselead -component  
confidential
```

Related information

See the following related commands:

Component
Report

Use the **report** command to obtain more information on existing authority groups:

- `teamc report -view authority`
- `teamc report -view authority -where "name ='developer' "`
- `teamc report -view authority -where "action ='PartCheckIn' "`

For a list of the access authority groups shipped with TeamConnection, refer to the *TeamConnection User's Guide*.

See your family administrator, or refer to the *Administrator's Guide*, for information about configuring new access authority groups and modifying existing ones.

Chapter 3. Approval

Use the **approval** command to record approvers' opinions on approval records about proposed changes to parts in a release. You can use this command only for a work area in the approve state. The **approval** command provides greater control over changes made to releases as final deadlines approach.

Approval records are created automatically every time a work area is created for a release that has an approver list. You can also create additional approval records for a work area, with the **approval** command, without changing the approver list associated with the release. For information on changing the approver list, see "Chapter 4. Approver" on page 29. You can also use the **approval** command to delete work area approval records or to assign them to other users.

Owners of an approval record must indicate on it whether they accept or reject the changes proposed by the work area. An abstain option is available.

The state of the approval record controls whether the associated work area can move to the fix state. If the release process includes the approval process, then when all approval records are in the accept or abstain state, the work area moves automatically to the fix state. If an approval record is in the reject state, the work area cannot move to the fix state. Refer to the *TeamConnection User's Guide* for descriptions of the various processes and states.

Command Summary

In the following syntax statements, the first letter of each argument is capitalized, for example `-family Name`. Replace these arguments with a value that is valid for your TeamConnection database. If you want to issue a command for the family named **testfam**, for example, replace `-family Name` with `—family testfam`.

The syntax statements for the `approval` command are:

```
teamc approval -abstain -release Name ... -family Name [-become Name]
               -workarea Name ... [-approver Name] [-verbose]
```

```
teamc approval -accept -release Name ... -family Name [-become Name]
               -workarea Name ... [-approver Name] [-verbose]
```

```
teamc approval -assign -to Name -release Name ... -family Name
               [-verbose] -workarea Name ... [-approver Name] [-become Name]
```

```
teamc approval -create -approver Name -release Name ... -family Name
               -workarea Name ... [-become Name] [-verbose]
```

```
teamc approval -delete -approver Name -release Name ... -family Name
               -workarea Name ... [-become Name] [-verbose]
```

```
teamc approval -reject -release Name ... -family Name [-become Name]
               -workarea Name ... [-approver Name] [-verbose]
```

approval -abstain

Lets users refrain from accepting or rejecting the proposed part changes for the specified work area.

Attributes

The approval -abstain command takes the following attributes:

-approver Name

The user ID of the approver.

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-verbose

TeamConnection displays a confirmation message after you issue the command.

-workarea Name . . .

The work area name.

(Environment variable: TC_WORKAREA)

approval -accept

Approves the proposed part changes for the specified work area.

Attributes

The approval -accept command takes the following attributes:

-approver Name

The user ID of the approver.

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-verbose

TeamConnection displays a confirmation message after you issue the command.

-workarea Name . . .

The work area name.

(Environment variable: TC_WORKAREA)

Examples

You own an approval record that is in the ready state. It refers to the changes proposed for a work area named **179** to monitor changes for feature **179** in the release specified in your TC_RELEASE environment variable. The following command indicates that you approve of the proposed changes for work area **179**. The approval record moves to the accept state. You do not have to specify the **-release** flag because the TC_RELEASE environment variable was set.

```
teamc approval -accept -workarea 179
```

approval -assign

Assigns an existing approval record to another user ID. The owner of that user ID becomes the owner of the approval record.

Attributes

The approval -assign command takes the following attributes:

-approver Name

The user ID of the approver.

-become Name

Approval

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-to Name

The user ID to which you want to reassign the object. The user ID you specify becomes the owner of the object.

-verbose

TeamConnection displays a confirmation message after you issue the command.

-workarea Name . . .

The work area name.

(Environment variable: TC_WORKAREA)

Examples

You own two approval records, one for the work required to resolve defect **9122** in release **10graphix** and the other for the work required to resolve the same defect in release **20graphix**. The name of the work area is **9122** in both releases. The following command assigns both of these approval records to **pam**, type:

```
teamc approval -assign -release 10graphix 20graphix -workarea 9122  
-to pam
```

approval -create

Creates an approval record for a work area. This action does not change the approver list.

For a user to perform this action, the associated release's process must include the approval subprocess.

Attributes

The approval -create command takes the following attributes:

-approver Name

The user ID of the approver.

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-verbose

TeamConnection displays a confirmation message after you issue the command.

-workarea Name . . .

The work area name.

(Environment variable: TC_WORKAREA)

Examples

Work area **147** in release **20graphix** is in the approve state. The following command creates an approval record so that **jack** must approve the proposed changes for that work area. An approval record is created and its owner is **jack**. When you specify a value for the **-release** flag, any existing value set for the TC_RELEASE environment variable is ignored. This action can be done only for releases that have an approver list, even though the action does not modify the list. For releases without approver lists, work areas are created with an initial state of fix; only when a work area is in the approve state can approval records be created or acted upon in such a release.

```
teamc approval -create -release 20graphix -workarea 147 -approver jack
```

approval -delete

Deletes an existing approval record for a specified user ID and work area.

Attributes

The approval -delete command takes the following attributes:

-approver Name

Approval

The user ID of the approver.

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-verbose

TeamConnection displays a confirmation message after you issue the command.

-workarea Name . . .

The work area name.

(Environment variable: TC_WORKAREA)

Examples

You have superuser privilege. The following command deletes the approval record owned by **maria** for a work area **2431B** addressing feature **2431** in release **10graphix**, type: The work area must still be in the approve state for a superuser to be able to delete the approval record.

```
teamc approval -delete -release 10graphix -workarea 2431B  
-approver maria
```

approval -reject

Lets users refuse to accept the proposed changes for the specified work area and keep the work area in the approve state. This prevents the work area from moving to the fix state. Work areas that are not approved can be canceled.

Attributes

The approval -reject command takes the following attributes:

-approver Name

The user ID of the approver.

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-verbose

TeamConnection displays a confirmation message after you issue the command.

-workarea Name . . .

The work area name.

(Environment variable: TC_WORKAREA)

Related information

See the following related commands:

- Approver
- Release
- Report
- Workarea

Refer to the *TeamConnection User's Guide* for descriptions of the various processes and states.

Chapter 4. Approver

Use the **approver** command to create entries on, and delete entries from, a release approver list. Each entry associates a user ID with a release, making the owner of the user ID an approver for any proposed changes to address defects or features in the specified release. The release approver list provides greater control over changes made to releases as final deadlines approach.

Every time a work area is created for a release to address a defect or a feature, approval records are created for each of the user IDs on the approver list associated with that release (providing that the release's process includes the approval subprocess). Each approval record refers to one defect or feature in one release and is owned by one approver. Approvers must use the **approval** command to accept or reject the proposed changes. Modifying an approver list does not change existing approval records.

Approval records that are accepted allow the work area to move to the fix state. If one or more approvers reject an approval record, the work area cannot move to the fix state.

Command Summary

In the following syntax statements, the first letter of each argument is capitalized, for example `-family Name`. Replace these arguments with a value that is valid for your TeamConnection database. If you want to issue a command for the family named **testfam**, for example, replace `-family Name` with `—family testfam`.

The syntax statements for the **approver** command are:

```
teamc approver -create -login Name ... -release Name -family Name  
[-become Name] [-verbose]
```

```
teamc approver -delete -login Name ... -release Name -family Name  
[-become Name] [-verbose]
```

approver -create

Adds user IDs to a release approver list.

Attributes

The `approver -create` command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

Approver

(Environment variable: TC_FAMILY)

-login Name . . .

One or more TeamConnection user IDs. Specify only one of -inherited or -login.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

The following command makes the owners of user IDs **jack** and **smitty** approvers for any changes that may be proposed for the **10debugr** release. Two entries are added to the approver list for the **10debugr** release. Approval records are created for **jack** and **smitty** when new work areas are created in reference to the **10debugr** release.

```
teamc approver -create -login jack smitty -release 10debugr
```

The following command adds user IDs **maria**, **john**, and **kevin** to the approver list associated with the release you have specified in your TC_RELEASE environment variable, type: Three entries are made to the approver list for the release set in your environment variable, one for each of the user IDs you specified.

```
teamc approver -create -login maria john kevin
```

approver -delete

Deletes user IDs from a release approver list.

You cannot delete the last entry in an approver list if it is associated with a release whose process includes the approval subprocess.

Attributes

The approver -delete Command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-login Name . . .

One or more TeamConnection user IDs. Specify only one of -inherited or -login.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

The following command deletes the approver list entry identifying the owner of user ID **maria** as an approver for the **tools** release.

```
teamc approver -delete -login maria -release tools
```

Related information

See the following related commands:

- Approval
- Defect
- Feature
- Release
- Report
- Workarea

Refer to the *TeamConnection User's Guide* for descriptions of the various processes and states.

Chapter 5. Builder

Use the **builder** command to create, view, modify, extract, or delete builders.

A builder is an object that can transform one set of TeamConnection parts into another by invoking tools such as compilers and linkers. For example, one builder might transform a COBOL source file into an object file. Another might transform a set of object files into an executable file. Builders use build scripts to invoke the tools that actually transform TeamConnection parts.

For more information about the **builder** command and build scripts, and for sample builder commands, refer to the *TeamConnection User's Guide*.

Command summary

In the following syntax statements, the first letter of each argument is capitalized, for example `-family Name`. Replace these arguments with a value that is valid for your TeamConnection database. If you want to issue a command for the family named **testfam**, for example, replace `-family Name` with `—family testfam`.

The syntax statements for the **builder** command are:

```
teamc builder -create Name -condition RCExpression
               -environment Name
               -from ScriptFilespec**
               -script Name
               -value RCValue -release Name
               -family Name
               [-text* | -binary | -none]
               [-parameters Parameters]
               [-timeout Number] [-become UserName]
               [-verbose]
```

```
teamc builder -delete Name -release Name -family Name
               [-become UserName] [-verbose]
```

```
teamc builder -extract Name -to ScriptFilespec
               -release Name -family Name
               [-become UserName] [-verbose]
```

```
teamc builder -modify Name
               -release Name -family Name
               {[-text | -binary | -none]
               -from ScriptFilespec**
               -condition RCExpression
               -environment Name
               -parameters Parameters
               -script Name
               -timeout Number -value RCValue}
               [-become UserName] [-verbose]
```

```
teamc builder -view Name -release Name -family Name
               [-become UserName] [-verbose]
```

Builder

Notes:

*text is the default

**from is required for file type of text or binary
and is prohibited for file type none

builder -create Name

The **builder -create Name** command creates a new builder.

When you create a builder, you must specify a build script. The build script actually invokes the transformation tool and passes it parameters defined in the **-parameter** attribute of the **builder** command.

Together with the contents of the build script and the tools you use (the compilers, linkers, and so on), a builder's attributes define how a transformation takes place.

- If the build script is simple enough to be expressed in one line, you can specify it in the **-script** attribute when you create the builder, and specify a file type of **-none**. At minimum, the script must specify the name of the transformation tool. For example, to invoke the VisualAge C++ compiler, you might specify these attributes:

```
-none -script icc
```

- If the build script is more complex, you must first create a separate file containing it. Refer to the *TeamConnection User's Guide* for more information about how to write a build script. Specify the fully qualified path name of your file as the source file, and specify the file type as **-text** or **-binary**. If you omit the **-text** or **-binary** attribute, TeamConnection can detect the file type and create the build script in the proper format.

When the builder is created, this source file is stored as part of the builder in the TeamConnection database; during a build, the build processor creates and runs a local version of this file. Specify the name you want for this local file using the **-script** attribute flag. For example, you might specify these attributes:

```
-text -script c_compile.cmd -source c:\src\c_compile.cmd
```

When this builder is created, the contents of c:\src\c_compile.cmd are stored in the builder. When this builder is invoked, TeamConnection creates a file named c_compile.cmd, writes the build script into this file, and then runs it.

Attributes

The builder -create Name command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-binary

Specifies that the object is a binary file. Specify only one of -binary, -none, or -text. If you specify none of these attributes, -text is the default.

-condition return condition

Together with the **-value** flag, makes up a Boolean expression that defines the criteria used to decide whether a specific build event was successfully accomplished, when evaluated against the value returned by the build script. The values allowed for this flag are as follows:

EQ or ==

Equals

LT or <

Less than

LE or <=

Less than or equals

GT or >

Greater than

GE or >=

Greater than or equals

NE or !=

Not equal to

-environment environment

Specifies the target environment this builder builds for. Note, the environment string should match the one specified on the build agent that you want to handle the build events performed by this builder.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-from script location

Specifies the location of the file contents.

-none

Specifies that the build script is not stored in the TeamConnection product. Specify only one of **-binary**, **-none**, or **-text**. If you specify none of these attributes, **-text** is the default.

-parameters

Specifies the parameters passed to the build script. For more information about parameters and build scripts, refer to the *TeamConnection User's Guide*.

In UNIX environments, you need to include an escape character before the **\$**: `\$(variable_name)`. The following is an example: `\$(TC_INPUT)`.

-release Name

The release for which this command is being issued.

Builder

(Environment variable: TC_RELEASE)

-script script name

Specifies the name of the build script. It can also identify the translator when the **-none** flag is specified. To specify no build script, use `-script " "` or `-script NULL` and set the file type to none.

-text

Specifies that the object is a text file. If you specify neither `-binary`, `-none`, nor `-text`, `-text` is the default.

-timeout minutes

Specifies the amount of time that the build processor waits for a build script to complete before assuming a failure has occurred. The default is 1440 minutes (24 hours).

-value return value

Together with the **-condition** flag, makes up a Boolean expression that defines the criteria used to decide whether a specific build event was successfully accomplished, when evaluated against the value returned by the build script. The **-value** can be any positive integer. An example of a Boolean expression formed from these two attributes is *return_value LE 4*, meaning that the build event is considered a success if the build script returns a value less than or equal to four.

-verbose

TeamConnection displays a confirmation message after you issue the command.

builder -delete Name

Deletes the specified builder from the database.

Attributes

The `builder -delete Name` command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-verbose

TeamConnection displays a confirmation message after you issue the command.

builder -extract Name

Extracts the build script of an existing builder so that you can modify it.

Attributes

The builder -extract Name command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-to filespec

Specifies the location (file name) to extract the contents to.

-verbose

TeamConnection displays a confirmation message after you issue the command.

builder -modify Name

Modifies an existing builder.

Attributes

The builder -modify Name command takes the following attributes:

-become Name

Builder

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-binary

Specifies that the object is a binary file. Specify only one of -binary, -none, or -text. If you specify none of these attributes, -text is the default.

-condition return condition

Together with the **-value** flag, makes up a Boolean expression that defines the criteria used to decide whether a specific build event was successfully accomplished, when evaluated against the value returned by the build script. The values allowed for this flag are as follows:

EQ or ==

Equals

LT or <

Less than

LE or <=

Less than or equals

GT or >

Greater than

GE or >=

Greater than or equals

NE or !=

Not equal to

-environment environment

Specifies the target environment this builder builds for. Note, the environment string should match the one specified on the build agent that you want to handle the build events performed by this builder.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-from script location

Specifies the location of the file contents.

-none

Specifies that the build script is not stored in the TeamConnection product. Specify only one of -binary, -none, or -text. If you specify none of these attributes, -text is the default.

-parameters

Specifies the parameters passed to the build script. For more information about parameters and build scripts, refer to the *TeamConnection User's Guide*.

In UNIX environments, you need to include an escape character before the **\$**: `\$(variable_name)`. The following is an example: `\$(TC_INPUT)`.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-script script name

Specifies the name of the build script. It can also identify the translator when the **-none** flag is specified. To specify no build script, use `-script ""` or `-script NULL` and set the file type to none.

-text

Specifies that the object is a text file. If you specify neither `-binary`, `-none`, nor `-text`, `-text` is the default.

-timeout minutes

Specifies the amount of time that the build processor waits for a build script to complete before assuming a failure has occurred. The default is 1440 minutes (24 hours).

-value return value

Together with the **-condition** flag, makes up a Boolean expression that defines the criteria used to decide whether a specific build event was successfully accomplished, when evaluated against the value returned by the build script. The **-value** can be any positive integer. An example of a Boolean expression formed from these two attributes is `return_value LE 4`, meaning that the build event is considered a success if the build script returns a value less than or equal to four.

-verbose

TeamConnection displays a confirmation message after you issue the command.

builder -view Name

Gets the settings for an existing builder.

Attributes

The `builder -view Name` command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

Builder

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-verbose

TeamConnection displays a confirmation message after you issue the command.

Related information

See the following related commands:

- Part

- Release

Chapter 6. Collision

Use the **collision** command to determine what should be done with a version of a part that is not integrated when another user has already integrated a new version of the same part in a concurrent release.

The TeamConnection product enables you to work with parts in a release in concurrent development mode. This means that multiple users can work on the same part at the same time. This is in contrast to serial mode, in which only one user can work on a part at a time.

Concurrent development mode is turned on when a release is created, and once it is set, it cannot be changed. After that, users can then check out parts to their work areas. From your work area, you can modify and build parts without the parts being visible to other users of the release. You make the parts visible to other users of the release when you integrate your work area.

If any other users in the release have integrated the same parts that you are working on and you refresh or try to integrate the parts in your work area, or you add a driverMember to a driver, the TeamConnection product generates collision records. The collision records identify the parts that are in conflict. You must compare the two files and determine how the collisions should be reconciled. The TeamConnection product provides a tool, called TeamConnection Merge, that enables you to selectively merge two files into one. However, the TeamConnection Merge tool is only valid for merging parts with the type of TCPart.

For example, assume that you have modified a part **x.c** in your work area **wa2** and in release **rel3v2**. Another user, Jon, who is also working in this concurrent release, has also modified part **x.c**. Jon integrates his work area and part **x.c**. You attempt to integrate your work area **wa2**, but you receive a message that you need to refresh your work area because the TeamConnection product has detected a collision. When you refresh your work area, the TeamConnection product generates a collision record for part **x.c**.

The **collision -reject** command rejects the newly integrated version (found in rel3v2:3) and says that the version in work area **wa2** should supersede the version in release **rel3v2**. You and Jon decide that your work area should supersede his. To integrate your work area **wa2** to the release, and to supersede Jon's version, type:

```
teamc collision -reject -path x.c -altversion rel3v2:3 -workarea wa2  
-release rel3v2 -verbose
```

Now you can try to integrate your part again.

Command summary

In the following syntax statements, the first letter of each argument is capitalized, for example **-family Name**. Replace these arguments with a value that is valid for your TeamConnection database. If you want to issue a command for the family named **testfam**, for example, replace **-family Name** with **-family testfam**.

The syntax statements for the **collision** command are:

```
teamc collision -accept -path Name -altversion Name -release Name  
-workarea Name ... -family Name [-top Name] [-type Name]
```

Collision

`[-become Name] [-verbose]`

```
teamc collision -reconcile -path Name -altversion Name -release Name  
-workArea Name ... -family Name [-top Name] [-type Name]  
[-become Name] [-verbose]
```

```
teamc collision -reject -path Name -altversion Name -release Name  
-workArea Name ... -family Name [-top Name] [-type Name]  
[-become Name] [-verbose]
```

Notes:

-type will default to TcPart if not specified.

collision -accept

The integrated part stays in the release, while the uncommitted part never gets in.

Attributes

The collision -accept command takes the following attributes:

-altversion Name

An ID of the view of the system containing the conflicting version of the part.

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-path Name

The path name of the part. Part names, consisting of the base name and the path name, must be unique within a release.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-top Name

Specifies the leading portion of the path name that is a subset of the current working directory on the client machine.

(Environment variable: TC_TOP)

-type Name

The type of the parts. The default is TcPart.

-verbose

TeamConnection displays a confirmation message after you issue the command.

-workarea Name . . .

The work area name.

(Environment variable: TC_WORKAREA)

Examples

In these examples, assume that you have refreshed work area **w1** from the release **R1**. You have collision records for part **x1.c**, part **x2.c**, and part **x3.c**. Also assume that the altversion for the collisions is **R1:3**.

The following command replaces your version of part **x1.c** with the committed version of the part, type:

```
teamc collision -accept -path x1.c -altversion R1:3 -workarea w1
-release R1 -verbose
```

collision -reconcile

The latest part that is not integrated takes precedence over the committed part, only if a new version of the part has been checked in since the TeamConnection product detected the collision.

When you reconcile parts using the TeamConnection command line, you must go through the following steps:

1. Check out the part in the area that has the collision.
2. Extract the part specified at the alternate version with another name.
3. Run the merge program against the two parts.
4. Check in the resultant file.
5. Mark the collision record as reconciled.

However, on the graphical user interface (GUI), the reconcile command automatically does the preceding steps for you.

Attributes

The collision -reconcile command takes the following attributes:

-altversion Name

An ID of the view of the system containing the conflicting version of the part.

-become Name

Collision

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-path Name

The path name of the part. Part names, consisting of the base name and the path name, must be unique within a release.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-top Name

Specifies the leading portion of the path name that is a subset of the current working directory on the client machine.

(Environment variable: TC_TOP)

-type Name

The type of the parts. The default is TcPart.

-verbose

TeamConnection displays a confirmation message after you issue the command.

-workarea Name . . .

The work area name.

(Environment variable: TC_WORKAREA)

Examples

In these examples, assume that you have refreshed work area **w1** from the release **R1**. You have collision records for part **x1.c**, part **x2.c**, and part **x3.c**. Also assume that the altversion for the collisions is **R1:3**.

To merge your current version of part **x3.c** in work area **w1** with the committed version of **x3.c**, follow these steps:

1. Check out part **x3.c** from work area **w1**, using the following command:

```
teamc part -checkout x3.c -workarea w1 -release R1 -verbose
```
2. Extract the committed version of part **x3.c** as part **x3.tmp**, using the following command:

```
teamc part -extract x3.c -version R1:3 -release R1 -fmask 777 -verbose
rename x3.c x3.tmp
```

3. Merge part **x3.tmp** with part **x3.c**, using the TeamConnection Merge tool. (You can use another merge tool if you choose.) To merge the parts, type:

```
tcmerge x3.tmp x3.c
```

4. Check part **x3.c** into work area **w1**.

5. Mark the collision as reconciled, using the following command:

```
teamc collision -reconcile -path x3.c -altversion R1:3 -workarea w1
-release R1 -verbose
```

This example assumes that you store the merged file as **x3.c**.

collision -reject

The part that is not integrated takes precedence over the integrated part.

The integrated version of the part will be overwritten by the version that is not integrated when the work area or driver is integrated into the release.

Attributes

The collision -reject command takes the following attributes:

-altversion Name

An ID of the view of the system containing the conflicting version of the part.

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-path Name

The path name of the part. Part names, consisting of the base name and the path name, must be unique within a release.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-top Name

Collision

Specifies the leading portion of the path name that is a subset of the current working directory on the client machine.

(Environment variable: TC_TOP)

-type Name

The type of the parts. The default is TcPart.

-verbose

TeamConnection displays a confirmation message after you issue the command.

-workarea Name . . .

The work area name.

(Environment variable: TC_WORKAREA)

Examples

In these examples, assume that you have refreshed work area **w1** from the release **R1**. You have collision records for part **x1.c**, part **x2.c**, and part **x3.c**. Also assume that the altversion for the collisions is **R1:3**.

The following command keeps your current version of part **x2.c** in work area **w1**, type:

```
teamc collision -reject -path x2.c -altversion R1:3 -workarea w1  
-release R1 -verbose
```

Related Information

See the following related commands:

Part

Release

Workarea

Chapter 7. Component

Use the **component** command to create and maintain a component structure for project control and management. The component structure or hierarchy consists of a top-level component called root. Every component below root is linked to one or more parent components and zero or more *child components*. Use **-link** and **-unlink** to redefine an existing component structure.

You can create, delete, and recreate components, modify their properties, or view information about them.

When you create a component, you become its owner and have implicit authority to define the access list and the notification list for that component. Although you have implicit authority to define the access list, you cannot add to that list until you have some level of authority defined on the access list. Therefore, when you first become the owner of a component, someone with enough authority must give you authority to create access for additional users. When you become the owner of a component, you might want to ask the component creator to give you access authority so that you can add other users to the access list. The access and notification list entries for a component apply to all descendant components via inheritance, unless access has been specifically restricted. As component owner, you also have implicit authority to manage that component and other objects relating to it.

When creating a component, you must specify a process for the component using the **-process** flag. A process groups different combinations of TeamConnection subprocesses. TeamConnection subprocesses determine the states that apply to the defects and features associated with a component. For component processes, the design, size, review (DSR) and verify subprocesses can be specified for defects, features, or both. Processes are configured by your family administrator, who can modify current processes or define new ones. For a list of the valid component processes and the TeamConnection subprocesses they include, use the **report -view cfgcomproc** command. You can change the process for an existing component using the **-modify** flag. For more information on how TeamConnection subprocesses relate to the states of TeamConnection objects, refer to the *TeamConnection User's Guide*.

You can delete a component only if there are no parts, child components, releases, active features, active defects, or active sizing records associated with it. The component's access and notification lists are deleted when it is deleted, and the component is detached from its parents. You cannot reuse the name of a deleted component to create another component; however, you can recreate a deleted component.

When you recreate a deleted component, you have to create new access and notification lists for it. The original access and notification lists are not recreated; however, the recreated component does inherit the access and notification information from all of the components above it in the hierarchy.

Command summary

In the following syntax statements, the first letter of each argument is capitalized, for example `-family Name`. Replace these arguments with a value that is valid for your TeamConnection database. If you want to issue a command for the family named **testfam**, for example, replace `-family Name` with **—family testfam**.

Component

The syntax statements for the **component** command are:

```
teamc component -create Name ... -parent Name -process Name  
-family Name [-owner Name] [-description Text]  
[-become Name] [-verbose]
```

```
teamc component -delete Name ... -family Name [-become Name] [-verbose]
```

```
teamc component -link Name ... -parent Name -family Name [-become Name]  
[-verbose]
```

```
teamc component -modify Name ... -family Name { -process Name  
-owner Name -name Name -description Text }  
[-become Name] [-verbose]
```

```
teamc component -recreate Name ... -parent Name -family Name  
[-become Name] [-verbose]
```

```
teamc component -unlink Name ... -parent Name -family Name  
[-become Name] [-verbose]
```

```
teamc component -view Name ... -family Name [-processInfo | -long]  
[-become Name] [-verbose]
```

component -create Name . . .

Creates components with the specified names. Component names must be unique within a family.

Attributes

The component -create Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-description Text

Specifies a description of the object.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-owner Name

Specifies the user ID of the owner of the object.

-parent Name

Specifies the parent of the object.

-process Name

Specifies a process setting and current subprocesses for the component. Your family administrator configures processes. For a list of the valid component processes and the TeamConnection subprocesses they include, use the **report -view cfgcomproc** command.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

The following command creates a new child component called **docs** for the existing **graphix** component, and assigns the IBM shipped process **preship** to it. The **docs** component inherits the access and notification defined at the **graphix** component and at all components above it in the hierarchy provided access has not been restricted.

Note: The access and notification lists for the **docs** component do not show the inherited access and notification information. Additional access and notification can be defined by creating access and notification lists for the **docs** component.

```
teamc component -create docs -parent graphix -description "Technical Info"
-process preship
```

component -delete Name . . .

Deletes the specified components. You cannot delete the root component..

Attributes

The component -delete Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-verbose

TeamConnection displays a confirmation message after you issue the command.

Component Examples

The following command deletes a component called **archive01**. The component called **archive01** is deleted only if it has no child components, no releases, no associated parts, no active defects, no active features, or no active sizing records referencing it.

```
teamc component -delete archive01
```

component -link Name . . .

Attaches components to an existing component. The components you list with this flag become child components of the component you specify with the **-parent** attribute flag.

Attributes

The component -link Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-parent Name

Specifies the parent of the object.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

The following command links two existing components, **docs** and **etc**, so that **etc** is the parent component to the **docs** component. The component **docs** becomes a child component of the component **etc**. The **docs** component inherits access and notification information from the **etc** component. It does not lose existing access and notification information from its own access and notification lists.

```
teamc component -link docs -parent etc
```

component -modify Name . . .

Modifies properties of the specified components.

Attributes

The component `-modify Name . . .` command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-description Text

Specifies a description of the object. Specify either `-description` or one of `-name`, `-owner`, or `-process`.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-name Name

Specifies a name for the object. Specify either `-description` or one of `-name`, `-owner`, or `-process`.

-owner Name

Specifies the user ID of the owner of the object. Specify either `-description` or one of `-name`, `-owner`, or `-process`.

-process Name

Specifies a process setting and current subprocesses for the component. Your family administrator configures processes. For a list of the valid component processes and the TeamConnection subprocesses they include, use the **report -view cfgcomproc** command. Specify either `-description` or one of `-name`, `-owner`, or `-process`.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

The following command gives **pam** ownership of the **graphix** component.

```
teamc component -modify graphix -owner pam
```

The following command changes the name, description, and process of the existing component called **graphix**, type: The component **graphix** is renamed to **graphix00**. The description indicates that this component refers to version **00 graphix** files. The process for the component is changed to **prototype**.

```
teamc component -modify graphix -name graphix00 -description
"Version 00 of graphix files" -process prototype
```

Component

component -recreate Name . . .

Recreates components as child components of the parent component. Use the **-parent** flag to specify the parent.

Attributes

The component -recreate Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-parent Name

Specifies the parent of the object.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

The following command recreates the deleted **tools** component so that it exists as a child component of the **graphix00** component. It inherits the access and notification information from the **graphix00** component.

```
teamc component -recreate tools -parent graphix00
```

component -unlink Name . . .

Detaches components from a parent component. The components being unlinked must still be linked to at least one parent component.

Attributes

The component -unlink Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-parent Name

Specifies the parent of the object.

-verbose

TeamConnection displays a confirmation message after you issue the command.

component -view Name . . .

Shows all current information for the specified components.

Attributes

The component -view Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-long

Displays detailed information for the specified objects. Specify only one of -long or -processInfo.

-processInfo

Displays the current process setting and associated TeamConnection subprocesses for the specified components when used with the **-view** action flag. Specify only one of -long or -processInfo.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

The following command displays information about an existing component called **debugr**, type:

Component

`teamc component -view debugr`

Related information

See the following related commands:

Access
Notify
Part
Report

Use the **report** command to obtain more information on existing components:

- `teamc report -view CompView`
- `teamc report -view CompView -where "name ='myComponent' "`

For a list of the valid component processes and the TeamConnection subprocesses they include, use the **report -view cfgcomproc** command.

See your family administrator, or refer to the *Administrator's Guide*, for information about creating new components and modifying existing ones.

Chapter 8. Coreq

Use the **coreq** command to create and delete corequisite relationships between two or more work areas that are in the fix or integrate state. The work areas you identify as corequisites must all apply to the same release to be built together. Work areas defined as prerequisites by the TeamConnection product must also be built and committed together. For a discussion of *prerequisite work areas*, refer to the *TeamConnection User's Guide*.

Identify corequisite relationships between work areas to indicate that work being done in one or more work areas is dependent on changes to parts associated with changes in another work area. These work areas must therefore be built together (committed together) so that the resulting code works correctly. This action ensures that a driver that includes one or more groups of *corequisite work areas* cannot be committed unless all the work areas in the corequisite group are included in the driver.

After you identify two or more work areas as corequisites, you can add additional work areas to that corequisite group without identifying all of the work areas already in the group. You have to specify only one work area from the existing group and the new work area or work areas you want to add to the group. If you specify one work area from each of two or more groups of corequisites, the associated groups are merged into one corequisite group.

When you delete one work area from a corequisite group containing only two work areas, no corequisite group remains. You must have at least two work areas to create a corequisite group of work areas.

Command summary

In the following syntax statements, the first letter of each argument is capitalized, for example `-family Name`. Replace these arguments with a value that is valid for your TeamConnection database. If you want to issue a command for the family named **testfam**, for example, replace `-family Name` with **—family testfam**.

The syntax statements for the **coreq** command are:

```
teamc coreq -create -workarea Name ... -release Name  
-family Name [-become Name] [-verbose]
```

```
teamc coreq -delete -workarea Name ... -release Name  
-family Name [-become Name] [-verbose]
```

coreq -create

Creates a corequisite relationship between the specified work areas.

Attributes

The `coreq -create` command takes the following attributes:

-become Name

Coreq

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-verbose

TeamConnection displays a confirmation message after you issue the command.

-workarea Name . . .

The work area name.

(Environment variable: TC_WORKAREA)

Examples

Work areas, with the same names as the features and defects, exist for feature **318**, defect **A329**, and defect **B312** in reference to the **graphix11** release. The following command establishes these three work areas as corequisites.

```
teamc coreq -create -workarea 318 A329 B312 -release graphix11
```

The following command adds work area **322** to the group of corequisite work areas created in the previous example. By naming one of the work areas from the existing corequisite group along with a new work area, you identify the new work area as a corequisite of each of the work areas in the existing group.

```
teamc coreq -create -workarea A329 322 -release graphix11
```

coreq -delete

Deletes the specified work areas from an existing group of corequisite work areas.

Attributes

The coreq -delete command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-verbose

TeamConnection displays a confirmation message after you issue the command.

-workarea Name . . .

The work area name.

(Environment variable: TC_WORKAREA)

Examples

The following command deletes the work area **318** in the release defined by the TC_RELEASE environment variable from its corequisite group.

```
teamc coreq -delete -workarea 318
```

Related information

See the following related commands:

- Prereq
- Report
- Workarea

For more information about corequisite and prerequisite relationships, refer to the *TeamConnection User's Guide*.

Chapter 9. Defect

Use the **defect** command to report problems by opening defects. Also use this command to modify properties of defects, change the state of defects, and view information about defects.

When you open a defect, you become the originator of the reported defect. You must describe the problem you think needs to be resolved and the primary component affected by the problem. By default the component owner is the owner of the defect if it is not assigned. That person must respond to the defect by accepting it, returning it, or assigning it to a different component or user ID. If the design, size, and review (dsrDefect) subprocess is included in the managing component's process, then the defect owner must respond to it by designing it, returning it, or assigning it to a different component or user ID.

As the originator of the defect, you can cancel or reopen it if it is returned by the defect owner, and you can modify selected properties of a defect.

Originators of duplicate defects are also notified when the corresponding active defect or feature is closed or canceled. They can either cancel or reopen the duplicate defect, as appropriate.

The states a defect moves through depends on the TeamConnection subprocesses included in its associated component process. A component process can include the dsrDefect or verifyDefect subprocesses, or none at all. For more information on the defect states and their relationship to TeamConnection subprocesses, refer to the *TeamConnection User's Guide*.

Command summary

In the following syntax statements, the first letter of each argument is capitalized, for example `-family Name`. Replace these arguments with a value that is valid for your TeamConnection database. If you want to issue a command for the family named **testfam**, for example, replace `-family Name` with `—family testfam`.

The syntax statements for the **defect** command are:

```
teamc defect -accept Name ... -family Name [-configField] [-answer Name]*
          [-remarks Text] [-become Name] [-verbose]

teamc defect -assign Name ... -family Name
          { -component Name -owner Name }
          [-remarks Text] [-become Name] [-verbose]

teamc defect -cancel Name ... -family Name [-remarks Text]
          [-become Name] [-verbose]

teamc defect -configInfo -family Name [-become Name] [-raw]

teamc defect -design Name ... -family Name [-remarks Text]
          [-become Name] [-verbose]

teamc defect -modify Name ... -family Name
          { -severity Name -answer Name
```

Defect

```
-environment Name -reference Name -priority Name -symptom Name  
-release Name -originator Name -target Name -driver Name  
-abstract Text -phaseFound Name -phaseInject Name  
-prefix Name -name Name} [-remarks Text] [-become Name]  
[-verbose]
```

```
teamc defect -note Name ... -remarks Text -family Name [-become Name]  
[-verbose]
```

```
teamc defect -open -remarks Text -component Name -family Name  
[-name Name] [-prefix Name]* [-environment Name]  
[-severity Name]* [-reference Name] [-symptom Name]*  
[-phaseFound Name]* [-driver Name] [-abstract Text] [-release Name]  
[-become Name] [-verbose]
```

```
teamc defect -reopen Name ... -family Name [-remarks Text]  
[-become Name] [-verbose]
```

```
teamc defect -return Name ... -family Name  
[-answer Name | -duplicate Name]*  
[-remarks Text] [-become Name] [-verbose]
```

```
teamc defect -review Name ... -family Name [-remarks Text]  
[-become Name] [-verbose]
```

```
teamc defect -size Name ... -family Name [-remarks Text]  
[-become Name] [-verbose]
```

```
teamc defect -verify Name ... -family Name [-remarks Text]  
[-become Name] [-verbose]
```

```
teamc defect -view Name ... -family Name [-processInfo | -long]  
[-become Name] [-verbose]
```

Notes:

* required when no default value is set for the TeamConnection family

defect -accept Name . . .

Accepts defects, in the open or review state, depending on the subprocess configuration of the component so that problems can be resolved. You can specify fields to be user-configurable using the **-configField** attribute overriding field defaults.

Attributes

The defect -accept Name . . . command takes the following attributes:

-answer Name

Specifies an answer code when accepting, modifying, or returning a defect. If no default has been set for this attribute, then you must include it with the command. This attribute is configurable.

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-configField Name

Specify a field to be user-configurable, overriding the field default. NULL is a valid value.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-remarks Text

Describes the change being requested, the actual design for the defect or feature, or the reason for modifying or changing the state of the defect or feature. Remarks are limited to 29,999 characters. If you need to include more than this limit, break your remarks into smaller files and load them from these files. After you issue a command that adds remarks, you cannot change the remarks (that is, you cannot use the **-modify** action to change the remarks).

Note: To move a defect or feature to the size state, you must have entered some design text using the **-remarks** flag with the **-design** action.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

Assume that you own a component against which someone opened defect **4312**. The following command accepts defect **4312** and associates it with the answer code your family administrator has configured to represent an enhancement (**enh**). Defect **4312** moves to the working state with an answer code for enhancement. Defect answer codes are defined by the family administrator.

```
teamc defect -accept 4312 -answer enh
```

defect -assign Name . . .

Reassigns defects to another owner or another component. The owner of the user ID or component becomes the new defect owner.

Attributes

The defect -assign Name . . . command takes the following attributes:

-become Name

Defect

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-component Name

The component associated with the object. Different components can manage different versions of the same object.

(Environment variable: TC_COMPONENT.)

-component Name

The component associated with the object. Different components can manage different versions of the same object. Either -component or -owner is required.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-owner Name

Specifies the user ID of the owner of the object. Either -component or -owner is required.

-remarks Text

Describes the change being requested, the actual design for the defect or feature, or the reason for modifying or changing the state of the defect or feature. Remarks are limited to 29,999 characters. If you need to include more than this limit, break your remarks into smaller files and load them from these files. After you issue a command that adds remarks, you cannot change the remarks (that is, you cannot use the **-modify** action to change the remarks).

Note: To move a defect or feature to the size state, you must have entered some design text using the **-remarks** flag with the **-design** action.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

The following command assigns defect **4312** to the **graphix** component. The owner of the **graphix** component becomes the owner of defect **4312**.

```
teamc defect -assign 4312 -component graphix
```

defect -cancel Name . . .

Cancels defects that are in the open or returned state.

Attributes

The defect `-cancel Name . . .` command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-remarks Text

Describes the change being requested, the actual design for the defect or feature, or the reason for modifying or changing the state of the defect or feature. Remarks are limited to 29,999 characters. If you need to include more than this limit, break your remarks into smaller files and load them from these files. After you issue a command that adds remarks, you cannot change the remarks (that is, you cannot use the **-modify** action to change the remarks).

Note: To move a defect or feature to the size state, you must have entered some design text using the **-remarks** flag with the **-design** action.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

Assume that you are the originator of defect **4298** and this defect is currently in the returned state. The following command cancels this defect. As the originator, you could have also canceled this defect if it was in the open state.

```
teamc defect -cancel 4298 -remarks "This was a user error."
```

defect -configInfo

The defect `-configInfo` action shows configurable field properties for defects in the specified family. The information is returned in a fixed ASCII table format.

Attributes

The defect `-configInfo` command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

Defect

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-raw

Displays configurable field information in raw format.

Examples

- The following command displays the configurable fields defined for the defects in family **rdev**.

```
teamc defect -configInfo -family rdev
```

The following is an example of the output provided for this command. This example assumes that the only configurable field defined is called **defTest**.

Attribute Name	DB Column Name	Create/Required	Type
defTest	defTest	yes/no	test

- The following command displays the configurable fields defined for users in family **rdev** in raw format.

```
teamc defect -configInfo -family rdev -raw
```

The following is an example of the output provided for this command. This example assumes that the only configurable field defined is called **defTest**.

```
Defect Test|Defect Test|defTest|defTest|yes|no|test
```

defect -design Name . . .

Moves defects to the design state or specifies design text. Defects can move to the design state from the open, returned, size, or review state.

Attributes

The defect -design Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-remarks Text

Describes the change being requested, the actual design for the defect or feature, or the reason for modifying or changing the state of the defect or feature. Remarks are limited to 29,999 characters. If you need to include more than this limit, break your remarks into smaller files and load them from these files. After you issue a command that adds remarks, you cannot change the remarks (that is, you cannot use the **-modify** action to change the remarks).

Note: To move a defect or feature to the size state, you must have entered some design text using the **-remarks** flag with the **-design** action.

-verbose

TeamConnection displays a confirmation message after you issue the command.

defect -modify Name . . .

The defect -modify Name . . . action modifies the following defect properties. At least one of these attributes is required with the defect -modify Name . . . action.

- abstract
- answer
- driver
- environment
- name
- originator
- phaseFound*
- phaseInject*
- prefix
- priority*
- reference
- release
- severity
- symptom*
- target*

You cannot modify existing remarks in a defect.

*Because your family administrator can modify or delete certain configurable fields and create new fields, the attributes for the **-modify** action might be different from those in your family or might not appear at all. Those listed here represent the shipped default fields only. For a list of the field properties and flags in use in your family, use the **defect -configInfo** command or see your family administrator. For more information on configurable fields, refer to the *Administrator's Guide*.

Attributes

The defect -modify Name . . . command takes the following attributes:

-abstract Text

Lets users enter concise text to summarize a defect or feature. Up to 63 characters are allowed. This text appears in reports and notification messages. If this flag is

Defect

not specified when you are opening a defect or feature, the first 63 characters or the text up to the first new-line character of the **-remarks** flag serves as the abstract.

-answer Name

Specifies an answer code when accepting, modifying, or returning a defect. If no default has been set for this attribute, then you must include it with the command. This attribute is configurable.

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-driver Name

Specifies the driver for which the command is issued.

- For defects this is the driver in which the defect was discovered.
- For driver members and work areas this is the driver in which the member is created or modified.

-environment environment

Specifies the environment where a defect was discovered, for example, the OS/2 environment.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-name Name

Specifies the defect or feature identifier. Up to 15 alphanumeric characters are allowed for user-generated IDs. The TeamConnection product checks the uniqueness of the ID. If the ID already exists in the TeamConnection product, the action fails and you receive a message indicating that the identifier is not unique. You must then enter a new identifier or let the TeamConnection product generate one.

-originator Name

Specifies the user ID of the new originator when you modify a defect or feature. The previous originator's verification record is automatically reassigned to the new originator.

-phaseFound Name

When opening or modifying a defect, specifies the development phase in progress when the defect was discovered. If no default has been set for this attribute, then you must include it with the command. This attribute is configurable.

-phaseInject Name

When modifying a defect, specifies the development phase in progress when the defect was injected in the code. This attribute is configurable.

-prefix Name

Identifies a prefix that categorizes the defect or feature by type. This value precedes the identifier in report output. If no default has been set for this attribute, then you must include it with the command. This attribute is configurable.

-priority Name

When modifying a defect, specifies the timing or scheduling requirements for resolving a defect. This attribute is configurable.

-reference Name

Assigns a value, name, or keyword to a defect or feature.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-release Name

The release to trigger the defect verification process when the work area for this release moves to the complete state. This attribute can take the **null** keyword.

(Environment variable: TC_RELEASE)

-remarks Text

Describes the change being requested, the actual design for the defect or feature, or the reason for modifying or changing the state of the defect or feature. Remarks are limited to 29,999 characters. If you need to include more than this limit, break your remarks into smaller files and load them from these files. After you issue a command that adds remarks, you cannot change the remarks (that is, you cannot use the **-modify** action to change the remarks).

Note: To move a defect or feature to the size state, you must have entered some design text using the **-remarks** flag with the **-design** action.

-severity Name

Specifies the severity of the problem that the defect was opened to resolve. If no default has been set for this attribute, then you must include it with the command. This attribute is configurable.

-symptom Name

Specifies the symptom associated with the defect. If no default has been set for this attribute, then you must include it with the command. This attribute is configurable.

Defect

-target Name

Specifies a target (such as, a driver or a date) for defect resolution or availability.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

The following command changes the severity rating for defect **4312** and to change the existing value in the reference field, type: The severity of the defect **4312** is changed to **3**, and the reference is changed to **BADMSG**.

```
teamc defect -modify 4312 -sev 3 -reference BADMSG
```

defect -note Name . . .

Adds remarks to defects. These notes cannot be modified or deleted once they are in the system.

Attributes

The defect -note Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-remarks Text

Describes the change being requested, the actual design for the defect or feature, or the reason for modifying or changing the state of the defect or feature. Remarks are limited to 29,999 characters. If you need to include more than this limit, break your remarks into smaller files and load them from these files. After you issue a command that adds remarks, you cannot change the remarks (that is, you cannot use the **-modify** action to change the remarks).

Note: To move a defect or feature to the size state, you must have entered some design text using the **-remarks** flag with the **-design** action.

-verbose

TeamConnection displays a confirmation message after you issue the command.

defect -open

Opens a defect. A unique identifier is generated by the TeamConnection product to identify the new defect, unless you specify an identifier using the optional **-name** flag.

Because your family administrator can modify or delete certain configurable defect fields and create new fields, the attributes for the **-open** action might be different from those in your family or might not appear at all. Those listed here represent the shipped default fields only. For a list of the field properties and flags in use in your family, use the **defect -configInfo** command or see your family administrator. For more information on configurable fields, refer to the *Administrator's Guide*.

Attributes

The defect -open command takes the following attributes:

-abstract Text

Lets users enter concise text to summarize a defect or feature. Up to 63 characters are allowed. This text appears in reports and notification messages. If this flag is not specified when you are opening a defect or feature, the first 63 characters or the text up to the first new-line character of the **-remarks** flag serves as the abstract.

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-component Name

The component associated with the object. Different components can manage different versions of the same object.

(Environment variable: TC_COMPONENT.)

-driver Name

Specifies the driver for which the command is issued.

- For defects this is the driver in which the defect was discovered.
- For driver members and work areas this is the driver in which the member is created or modified.

-environment environment

Specifies the environment where a defect was discovered, for example, the OS/2 environment.

-family Name

The family for which this command is being issued.

Defect

(Environment variable: TC_FAMILY)

-name Name

Specifies the defect or feature identifier. Up to 15 alphanumeric characters are allowed for user-generated IDs. The TeamConnection product checks the uniqueness of the ID. If the ID already exists in the TeamConnection product, the action fails and you receive a message indicating that the identifier is not unique. You must then enter a new identifier or let the TeamConnection product generate one.

-phaseFound Name

When opening or modifying a defect, specifies the development phase in progress when the defect was discovered. If no default has been set for this attribute, then you must include it with the command. This attribute is configurable.

-prefix Name

Identifies a prefix that categorizes the defect or feature by type. This value precedes the identifier in report output. If no default has been set for this attribute, then you must include it with the command. This attribute is configurable.

-reference Name

Assigns a value, name, or keyword to a defect or feature.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-release Name

The release to trigger the defect verification process when the work area for this release moves to the complete state. This attribute can take the **null** keyword.

(Environment variable: TC_RELEASE)

-remarks Text

Describes the change being requested, the actual design for the defect or feature, or the reason for modifying or changing the state of the defect or feature. Remarks are limited to 29,999 characters. If you need to include more than this limit, break your remarks into smaller files and load them from these files. After you issue a command that adds remarks, you cannot change the remarks (that is, you cannot use the **-modify** action to change the remarks).

Note: To move a defect or feature to the size state, you must have entered some design text using the **-remarks** flag with the **-design** action.

-severity Name

Specifies the severity of the problem that the defect was opened to resolve. If no default has been set for this attribute, then you must include it with the command. This attribute is configurable.

-symptom Name

Specifies the symptom associated with the defect. If no default has been set for this attribute, then you must include it with the command. This attribute is configurable.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

Assume default values are set for phaseFound, symptom, and prefix. The following command opens a defect with a severity rating of **3** against the **debugr** component, using the text from an existing file to describe the defect, type: The negative symbol (-) after the **-remarks** flag indicates the location of the redirected input. The redirection symbol (<) indicates that the file **\tmp\defect.descr** contains the remarks, that is, the description of the problem. The first 63 characters are used as the abstract.

The defect identifier is displayed on the screen when the command is completed successfully.

The person issuing the command is the originator of this defect, and the component owner is the owner of the defect.

```
teamc defect -open -component debugr -sev 3
-remarks - < \tmp\defect.descr
```

defect -reopen Name . . .

Reopens defects that are in the returned or the canceled state.

Attributes

The defect -reopen Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-remarks Text

Defect

Describes the change being requested, the actual design for the defect or feature, or the reason for modifying or changing the state of the defect or feature. Remarks are limited to 29,999 characters. If you need to include more than this limit, break your remarks into smaller files and load them from these files. After you issue a command that adds remarks, you cannot change the remarks (that is, you cannot use the **-modify** action to change the remarks).

Note: To move a defect or feature to the size state, you must have entered some design text using the **-remarks** flag with the **-design** action.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

Assume that you are the originator of defect **1424**. It is returned to you by the defect owner. The following command reopens defect **1424**. Defect **1424** moves to the open state.

```
teamc defect -reopen 1424  
-remarks "Disagree with restriction classification."
```

defect -return Name . . .

Returns defects that are in the open, design, size, review, or working states. A working defect can be returned only if it does not have work areas associated with it.

Attributes

The defect -return Name . . . command takes the following attributes:

-answer Name

Specifies an answer code when accepting, modifying, or returning a defect. Specify either -answer or -duplicate. If no default has been set, then one of these attributes is required. This attribute is configurable.

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-duplicate Name

Specifies that another defect (that is not canceled, returned, or closed) already exists to address the defect being returned. Specify either -answer or -duplicate. If no default has been set, then one of these attributes is required.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-remarks Text

Describes the change being requested, the actual design for the defect or feature, or the reason for modifying or changing the state of the defect or feature. Remarks are limited to 29,999 characters. If you need to include more than this limit, break your remarks into smaller files and load them from these files. After you issue a command that adds remarks, you cannot change the remarks (that is, you cannot use the **-modify** action to change the remarks).

Note: To move a defect or feature to the size state, you must have entered some design text using the **-remarks** flag with the **-design** action.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

Assume that you are a defect owner. The following command returns a defect someone opened against your component because it is a duplicate of a defect that is currently in the working state. Defect **4245** is associated with defect **4197** as a duplicate. Defect **4245** is moved to the returned state, and its answer code becomes **duplicate**. A verification record is created for the originator of defect **4245** and it exists in reference to defect **4197**. Originators of all duplicate defects and features must complete verification records when the active defect is in the verify state.

```
teamc defect -return 4245 -duplicate 4197
```

defect -review Name . . .

Moves defects from the size state to the review state so that the proposed design implementation and sizing information can be reviewed.

Attributes

The defect -review Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-remarks Text

Defect

Describes the change being requested, the actual design for the defect or feature, or the reason for modifying or changing the state of the defect or feature. Remarks are limited to 29,999 characters. If you need to include more than this limit, break your remarks into smaller files and load them from these files. After you issue a command that adds remarks, you cannot change the remarks (that is, you cannot use the **-modify** action to change the remarks).

Note: To move a defect or feature to the size state, you must have entered some design text using the **-remarks** flag with the **-design** action.

-verbose

TeamConnection displays a confirmation message after you issue the command.

defect -size Name . . .

Moves defects from the design state to the size state for sizing. Design text must first be entered using **defect -design -remarks**.

Attributes

The defect -size Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-remarks Text

Describes the change being requested, the actual design for the defect or feature, or the reason for modifying or changing the state of the defect or feature. Remarks are limited to 29,999 characters. If you need to include more than this limit, break your remarks into smaller files and load them from these files. After you issue a command that adds remarks, you cannot change the remarks (that is, you cannot use the **-modify** action to change the remarks).

Note: To move a defect or feature to the size state, you must have entered some design text using the **-remarks** flag with the **-design** action.

-verbose

TeamConnection displays a confirmation message after you issue the command.

defect -verify Name . . .

Moves defects from the working state to the verify state. If the verify subprocess is not on, this moves the defect to the closed state.

Attributes

The defect -verify Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-remarks Text

Describes the change being requested, the actual design for the defect or feature, or the reason for modifying or changing the state of the defect or feature. Remarks are limited to 29,999 characters. If you need to include more than this limit, break your remarks into smaller files and load them from these files. After you issue a command that adds remarks, you cannot change the remarks (that is, you cannot use the **-modify** action to change the remarks).

Note: To move a defect or feature to the size state, you must have entered some design text using the **-remarks** flag with the **-design** action.

-verbose

TeamConnection displays a confirmation message after you issue the command.

defect -view Name . . .

Shows all current information for the specified defects.

Attributes

The defect -view Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

Defect

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-long

Displays detailed information for the specified objects. Specify only one of -long or -processInfo.

-processInfo

Displays the current process setting and associated TeamConnection subprocesses for the specified components when used with the **-view** action flag. Specify only one of -long or -processInfo.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

The following command displays information about defect **4244**, including its history, notes, work areas, verification records, process name, and associated subprocess settings.

```
teamc defect -view 4244 -long
```

Related information

See the following related commands:

- Feature
- Fix
- Report
- Size
- Verify
- Workarea

Use the **report** command to get more information on existing configuration table values:

- **teamc report -view config**
- **teamc report -view config -where "name = 'symptom' "**

To see the defect state diagrams, refer to the *TeamConnection User's Guide*.

Chapter 10. Driver

Use the **driver** command to:

- Create and delete drivers
- Commit the part changes related to drivers
- Extract the part tree represented by drivers
- Obtain information about existing drivers

A driver is a set of part changes for a release. To create a driver, you assign a name to it and relate it to a release. You then define a set of work areas as *driver members*. For information on how to define work areas as driver members, see “Chapter 11. DriverMember” on page 91. These work areas contain the parts that you want in a driver. If you create a driver, you become the driver owner by default; however, you can reassign ownership of the driver to another user.

A driver can be extracted to the client at any time after work areas are added as driver members. A *delta part tree*, which contains only the parts that have been changed for the driver, is extracted by default. You can also extract a *full part tree* that contains all of the parts for the driver after the driver has been committed.

If a driver is committed, you can extract a full part tree that includes all the parts in the associated release at the version that was current when the driver was committed. You can also process and distribute a committed driver for testing. After a driver has been committed, however, you can change only its type; you cannot otherwise modify the driver or the part changes associated with that driver.

Combining the delta part tree for a current driver with a full part tree for the last committed driver results in a complete directory structure of all parts in a release. This directory structure incorporates the new part changes. The process of making part changes, extracting a delta part tree and a full part tree, combining the two into a new directory structure, and compiling the directory structure can be repeated as needed.

To make permanent all part changes associated with the driver, you move the driver to the commit state. Before you can do this:

- All driver members must be in the integrate or commit state.
- All prerequisite and corequisite work areas must be included in the driver.
- You must have explicit access authority or be a superuser.

If you have explicit access authority, you can indicate when a driver is ready for formal testing by specifying that the driver is complete. This action changes the state of the associated work areas to the test state if an environment list exists for the release associated with the work areas. Otherwise, the work areas move to the complete state.

Command Summary

In the following syntax statements, the first letter of each argument is capitalized, for example `-family Name`. Replace these arguments with a value that is valid for your TeamConnection database. If you want to issue a command for the family named **testfam**, for example, replace `-family Name` with `—family testfam`.

Driver

The syntax statements for the **driver** command are:

```
teamc driver -assign Name ... -to Name -release Name -family Name  
[-become Name] [-verbose]
```

```
teamc driver -check Name ... [-long] -release Name -family Name  
[-become Name] [-verbose]
```

```
teamc driver -commit Name ... -release Name  
-family Name [-force] [-become Name]  
[-verbose]
```

```
teamc driver -complete Name ... -release Name -family Name  
[-become Name] [-verbose]
```

```
teamc driver -create Name ... -release Name -family Name [-type Name]*  
[-become Name] [-verbose]
```

```
teamc driver -delete Name ... -release Name -family Name  
[-become Name] [-verbose]
```

```
teamc driver -export Name ... -release Name -family Name  
[-become Name] -file Name [-verbose]
```

```
teamc driver -extract Name ... -release Name -root Name  
-family Name [-full] [-nokeys] [-fmask Octal_number]  
[-dmask Octal_number] [-crlf|nocrlf]  
[-become Name] [-verbose]
```

```
teamc driver -freeze Name ... -release Name -family Name  
[-become Name] [-verbose]
```

```
teamc driver -modify Name ... -release Name -family Name { -name Name  
-type Name } [-become Name] [-verbose]
```

```
teamc driver -refresh Name ... -release Name -family Name  
[-become Name] [-verbose]
```

```
teamc driver -restrict Name ... -release Name -family Name  
[-become Name] [-verbose]
```

```
teamc driver -view Name ... [-long] -release Name -family Name  
[-become Name] [-verbose]
```

Notes:

* required when no default value is set for the TeamConnection family

driver -assign Name . . .

Assigns ownership of drivers to another user ID.

Attributes

The driver `-assign Name . . .` command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-to Name

The user ID to which you want to reassign the object. The user ID you specify becomes the owner of the object.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

Assume that you own driver **b1992** and that your TC_RELEASE environment variable is set to the release associated with that driver. The following command assigns the driver to user ID **sara**. The person with the TeamConnection user ID **sara** becomes the new owner of the driver **b1992** for the release defined by the TC_RELEASE environment variable. If the environment variable were set differently, you would have had to use the **-release** attribute flag to specify the appropriate release for the driver.

```
teamc driver -assign b1992 -to sara
```

driver -check Name . . .

Lists the outstanding prerequisite and corequisite work areas for the specified drivers.

Attributes

The driver `-check Name . . .` command takes the following attributes:

-become Name

Driver

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-long

Displays detailed information for the specified objects.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

Assume driver **9032** is in the integrate state. The following command checks whether any outstanding prerequisites or corequisites exist in driver **9032** for release **21debugr**. Any existing unsatisfied prerequisite and corequisite work areas required for driver **9032** for the **21debugr** release are listed.

```
teamc driver -check 9032 -release 21debugr
```

driver -commit Name . . .

The driver -commit Name . . . action moves the specified drivers to the commit state. After a driver is committed only its type can be modified. All part changes associated with driver members become permanent.

Before you can commit a driver to which you have added work areas in fix state, you must integrate the work areas and then add them to the driver again.

To perform this action, the associated release's process must include the driver subprocess.

Attributes

The driver -commit Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-force

Continues with the operation even if build outputs are out-of-date.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-verbose

TeamConnection displays a confirmation message after you issue the command.

driver -complete Name . . .

Moves the specified drivers to the complete state where they are ready to be tested. All driver members change to the test or complete state.

Attributes

The driver -complete Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-verbose

TeamConnection displays a confirmation message after you issue the command.

driver -create Name . . .

Creates drivers with the specified names. The user who creates a driver is the driver owner by default.

To perform this action, the associated release's process must include the driver subprocess.

Attributes

The driver -create Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-type Name

Specifies the type of driver when creating a driver. If no default has been set for this attribute, then you must include it with the command. You can use the **report** command to find out what the types of drivers are. This attribute is configurable.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

Assume that driver type has a default value. The following command creates a driver called **9032** for the **21debugr** release. The person who issues this command is the driver owner and it is in the working state. Use the **DriverMember** command to add work areas to the driver.

```
teamc driver -create 9032 -release 21debugr
```

driver -delete Name . . .

Deletes the specified drivers before they are committed. This command also deletes all driver members associated with the driver.

Attributes

The driver `-delete Name . . .` command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-verbose

TeamConnection displays a confirmation message after you issue the command.

driver -export Name . . .

The driver `-export Name . . .` command exports the parts in the driver (or drivers) specified by the *Name* argument to the file name specified in the `-file` attribute.

This function is useful for exporting information from one family into another. You can export the information from one family into a CDF file and then import the CDF file into another family.

For more information about the TeamConnection import and export functions refer to the *Tool Builder's Development Guide*. The *Tool Builder's Development Guide* also provides more information about common data format (CDF), the file format from which TeamConnection imports information and to which it exports information.

Note: You cannot use the import and export functions to migrate information from CMVC to TeamConnection or from one release of TeamConnection to another. Refer to the *Administrator's Guide* for information on the TeamConnection migration tool.

Attributes

The driver `-export Name . . .` command takes the following attributes:

-file filename

Specifies the name of the file to which the part or parts are to be exported. The information is exported to the file in CDF format.

Driver

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

driver -extract Name . . .

Creates a part tree by extracting the parts defined by the member work areas of specified drivers. The default is to extract only changed parts.

When extracting multiple drivers, you must specify the driver names in the chronological order in which they were committed or created.

Attributes

The driver -extract Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-crlf

Provides transparent file conversion between UNIX- and Intel-based operating systems. This attribute enables parts shared between UNIX and Intel platforms to have the proper format for the platform to which they are extracted. When parts are extracted to an Intel platform, the -crlf attribute will add carriage-returns, expand tabs, and add end-of-file characters (if the parts do not already have EOF characters). When parts are extracted to a UNIX platform, the -crlf attribute will remove carriage-returns, replace spaces with tabs, and remove end-of-file characters.

If you omit this attribute, no file format conversion is performed.

-dmask Octal_number

Specifies the read, write, and execute directory permissions for extracted parts in octal notation.

The default is 750 (read, write and execute access for directory owner, read and execute access for others in the owner's group, and no access for all other users).



While the OS/2 client accepts **-dmask**, it has no effect.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-fmask Octal_number

Specifies the read, write, and execute file permissions for extracted parts in octal notation. The default is the file's mode less the write permission for the part owner, others in the owner's group and all others.

-full

Includes all of the parts in a release in the extracted part tree. Use to extract a full part tree. Only available for committed drivers.

-nokeys

Indicates that you do not want to substitute assigned values in place of keywords embedded in the extracted parts.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-root Name

Specifies a directory on the client where the extracted part tree is to be placed.

Note: You can mount a directory from another machine to the client machine, so that the client machine will treat that directory as a local directory.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

Assume that you want to extract all of the parts for the committed driver **9032** to a specific directory and host. Also assume that the directory **\tmp** has been exported on a host with write permission given to the TeamConnection client and that the directory is mounted on the client using NFS, Netware, IBM LAN Server, or another LAN product. The following command places the full part tree in the **\tmp** directory. A full part tree produces a snapshot of all the parts in a release at the time the driver was committed.

Driver

```
teamc driver -extract 9032 -release 21debugr -full -root \tmp
```

driver -freeze Name . . .

Saves the state of a driver.

Attributes

The driver -freeze Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-verbose

TeamConnection displays a confirmation message after you issue the command.

driver -modify Name . . .

Changes the name or type of the specified drivers. Type is configured by family. Use the **report** command to find out the driver types for your family.

After a driver has been committed, only its type can be changed.

Attributes

The driver -modify Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-name Name

Specifies a name for the object. Either -name or -type is required.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-type Name

Specifies the type of driver when creating or modifying a driver. If no default has been set for this attribute and the -name attribute is not specified, then you must include it with the command. You can use the **report** command to find out what the types of drivers are. Either -name or -type is required. This attribute is configurable.

-verbose

TeamConnection displays a confirmation message after you issue the command.

driver -refresh Name . . .

Refreshes the contents of one driver with another driver.

Attributes

The driver -refresh Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-verbose

TeamConnection displays a confirmation message after you issue the command.

driver -restrict Name . . .

The driver -restrict Name . . . command moves a driver from the integrate state to the restrict state. When a driver is in restrict state, users must have MemberCreateR or MemberDeleteR access to the driver to add or delete members from the driver. You can, however, check, commit, extract, freeze, refresh, or view drivers in restrict state.

Attributes

The driver -restrict Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

The driver -restrict Name . . . command is especially useful if your installation performs highly automated builds. You can issue the driver -restrict Name . . . command prior to building to prevent any changes to the content of the driver. If the build succeeds, the driver is committed; if it fails, the build administrator (provided he or she has MemberCreateR or MemberDeleteR access to the driver) can remove the work areas that caused the build to fail. The following command moves driver **b1992** release **debugr** from the integrate state to the restrict state.

```
teamc driver -restrict b1992 -release debugr -verbose
```

driver -view Name . . .

Shows all current information for the specified drivers.

Attributes

The driver `-view Name . . .` command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-long

Displays detailed information for the specified objects.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

The following command displays information about driver **b1992** for release **debugr**, including all of its driver members.

```
teamc driver -view b1992 -release debugr -long
```

Related information

See the following related commands:

- Defect
- DriverMember
- Feature
- Part
- Report
- Workarea

For a list of supported keywords, refer to the *TeamConnection User's Guide*.



Parts that are deleted or renamed in the current driver must be deleted

Driver

from an extracted part tree. TeamConnection creates a part named **.gone** that specifies the full path name of each part deleted or renamed that has not already been committed. This part is extracted with the files in a delta tree extraction. Extracting the delta part tree of an uncommitted driver extracts all parts listed in the **.gone** part.

After merging a delta tree with a *base part tree*, or when extracting uncommitted drivers, run the following command from the top of the extracted part tree to remove deleted and renamed parts from the tree:

```
xargs rm < .gone
```

In this command, **.gone** is a part created as part of the extraction that contains the names of all of the deleted and renamed parts.

When you extract a committed driver, any parts that were contained in that driver when it was committed are accessed. When you destroy a part, the database record for that part is destroyed but the part remains on the server so that previously committed drivers can be rebuilt.

Chapter 11. DriverMember

Use the **driverMember** command to specify the work areas you want to include in a given driver. Driver members are used only if the release has the driver process turned on. The work areas can be in the integrate or fix state. A single work area can be a member of more than one driver. After a work area is committed in a driver, the other drivers in which it is a member ignore the committed work area.

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. The work areas must also be members of the release associated with the driver.

You cannot create driver members for or delete driver members from a driver after it is committed.

Command summary

In the following syntax statements, the first letter of each argument is capitalized, for example `-family Name`. Replace these arguments with a value that is valid for your TeamConnection database. If you want to issue a command for the family named **testfam**, for example, replace `-family Name` with `—family testfam`.

The syntax statements for the **driverMember** command are:

```
teamc driverMember -create -driver Name -release Name -family Name
                        -workarea Name ... [-force] [-become Name] [-verbose]
teamc driverMember -delete -driver Name -release Name -family Name
                        -workarea Name ... [-version Name ...]
                        [-become Name] [-verbose]
```

driverMember -create

Creates work areas as members of a specific driver. If the work area is in integrate state, the latest version of the work area is added to the driver. If the driver is in the restrict state, then you must have MemberCreateR access to the driver to issue this command. Superusers and driver owners have implicit MemberCreateR authority.

You can add a work area in fix state to a driver. When a work area in fix state is added to a driver, the latest frozen version of the work area is added. If there is no frozen version of the work area, TeamConnection returns an error. If you update the work area after freezing it and adding it to a driver, you can issue another `driverMember -create` command to refresh the driver with the updated work area. If a driver has multiple driver members for the same work area, you can use the `-version` attribute to identify a specific version of the work area in the driver. Before you can commit a driver to which you have added work areas in fix state, you must integrate the work areas and then add them to the driver again.

Note: If you have recently migrated from CMVC to TeamConnection, please be aware of differences between adding work areas in fix state to drivers and adding tracks in fix state to levels. In CMVC, when a track in fix state is added to a level, all current and future changes to the track are considered part of the level. In TeamConnection, when you add a work area in fix state to a driver, only the current frozen version of the work area is considered

DriverMember

part of the driver. To include any further changes to the work area, you need to freeze the work area again and issue another driverMember -create command to add the further changes to the work area.

If adding a driver member to a driver would result in collision records, the driverMember -create action will fail. The collision records need to be resolved before adding the workarea as a driver member.

To perform a driverMember -create action, the associated release's process must include the driver subprocess.

Attributes

The driverMember -create command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-driver Name

Specifies the driver for which the command is issued.

- For defects this is the driver in which the defect was discovered.
- For driver members and work areas this is the driver in which the member is created or modified.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-force

Continues with the operation even if build outputs are out-of-date.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-verbose

TeamConnection displays a confirmation message after you issue the command.

-workarea Name . . .

The work area name.

(Environment variable: TC_WORKAREA)

Examples

Assume that driver **9012** already exists and that you own it. The following command creates driver members using the work areas **8761** and **8690** in release **21graphix**. All parts changed in reference to the work areas **8761** and **8690** in the release **21graphix** are included in driver **9012**. Driver **9012** must be associated with release **21graphix**.

```
teamc driverMember -create -workarea 8761 8690 -release 21graphix
-driver 9012
```

driverMember -delete

The driverMember -delete action deletes work areas as members of a specific driver.

If the driver is in the restrict state, then you must have MemberDeleteR access to the driver to issue this command. Superusers and driver owners have implicit MemberDeleteR authority.

If you added a work area to a driver when the work area was in fix state, then to delete it you must use the -version attribute to specify the work area to be deleted rather than the -workarea attribute.

Attributes

The driverMember -delete command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-driver Name

Specifies the driver for which the command is issued.

- For defects this is the driver in which the defect was discovered.
- For driver members and work areas this is the driver in which the member is created or modified.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-verbose

DriverMember

TeamConnection displays a confirmation message after you issue the command.

-version Name

Specifies the version of the object. Specify either -version or -workarea or both.

-workarea Name . . .

The work area name.

(Environment variable: TC_WORKAREA)

Examples

Assume that you own driver **9010**. The following command deletes driver members, specifically work areas **8744** and **8759** in the release defined in the TC_RELEASE environment variable. Driver **9010** must be associated with the release defined in the TC_RELEASE environment variable.

```
teamc driverMember -delete -workarea 8744 8759 -driver 9010
```

Related information

See the following related commands:

Driver

Part

Report

Workarea

Chapter 12. Environment

Use the **environment** command to create and modify entries on, and delete entries from a release environment list. Each environment list entry consists of an environment name and the user ID of a designated tester for that environment. You can specify the environments in which a resolved defect or an implemented feature must be tested. One user can be responsible for testing more than one environment, so a user ID can have more than one entry on the environment list.

Test records are created according to the environment list of the release for each work area that is created for that release. See the **test** command for information on entering environment test results. If an environment list does not exist for a release, then the testing process using test records is bypassed for all work areas associated with that release.

Command summary

In the following syntax statements, the first letter of each argument is capitalized, for example `-family Name`. Replace these arguments with a value that is valid for your TeamConnection database. If you want to issue a command for the family named **testfam**, for example, replace `-family Name` with `-family testfam`.

The syntax statements for the **environment** command are:

```
teamc environment -create Name ... -tester Name -release Name  
-family Name [-become Name] [-verbose]
```

```
teamc environment -delete Name ... -release Name -family Name  
[-become Name] [-verbose]
```

```
teamc environment -modify Name ... -tester Name -release Name  
-family Name [-become Name] [-verbose]
```

environment -create Name . . .

Creates one or more environment list entries for a release by specifying an environment and a user who is responsible for testing in that environment.

Attributes

The environment -create Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

Environment

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-tester Name

The user responsible for testing in a given environment.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

Assume you own the **debugr** release. Work being done in reference to that release needs to be tested in the **PCVersion1** environment as well as in the **PCVersion2** environment. The following command specifies **jon** as the tester on the environment list associated with the **debugr** release: Two new environment list entries are created for the **debugr** release: one for the **PCVersion1** environment and one for the **PCVersion2** environment. The owner of the user ID **jon** is responsible for testing both environments. Therefore, **jon** owns 2 test records for every work area that is created for the **debugr** release.

```
teamc environment -create PCVersion1 PCVersion2 -tester jon  
-release debugr
```

environment -delete Name . . .

Deletes one or more environment list entries for a release.

Attributes

The environment -delete Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

The following command deletes all entries for environment **ModelA** from the environment list associated with the release set in the TC_RELEASE environment variable.

```
teamc environment -delete ModelA
```

environment -modify Name . . .

Modifies one or more environment list entries for a release.

Attributes

The environment -modify Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-tester Name

The user responsible for testing in a given environment.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

The following command assigns a new tester for **PCVersion2** in the **graphix** release. The owner of the user ID **lisa** replaces the previous person responsible for testing the **PCVersion2** environment for the **graphix** release.

```
teamc environment -modify PCVersion2 -tester lisa -release graphix
```

Related information

See the following related commands:

- Release
- Report
- Test
- Workarea

Chapter 13. Feature

Use the **feature** command to open requests for design changes or ideas for future functions. Also use this command to delete, modify properties of, change the state of, and view information about features.

The states a feature moves through depend on the TeamConnection subprocesses included in its associated component process. A component process can include the feature design, size, and review (dsrFeature) or verifyFeature subprocesses, or none at all. For more information on the feature states and their associated subprocesses, refer to the *TeamConnection User's Guide*.

When you open a feature, you become the originator of the feature. You must describe the proposed design change and name the primary component affected by the feature. The owner of the component you assign the feature to becomes the feature owner. If the dsrFeature subprocess is included in the component's process, the feature owner responds to the feature by moving it to the design state, returning it, or assigning it to a different component or user ID. If the dsrFeature subprocess is not included in the component's process, the feature owner either accepts the feature, returns it, or assigns it to a different component or user ID.

As the originator of the feature, you can cancel or reopen it if it is returned by the feature owner. You can also modify selected properties of a feature. Originators of duplicate features are notified when the corresponding active defect or feature is closed or canceled. Thus, they can either cancel or reopen the duplicate feature, as appropriate.

Command summary

In the following syntax statements, the first letter of each argument is capitalized, for example `-family Name`. Replace these arguments with a value that is valid for your TeamConnection database. If you want to issue a command for the family named **testfam**, for example, replace `-family Name` with `—family testfam`.

The syntax statements for the **feature** command are:

```
teamc feature -accept Name ... -family Name [-configField] [-remarks Text]
[-become Name] [-verbose]
```

```
teamc feature -assign Name ... -family Name [-remarks Text]
[-verbose] { -component Name -owner Name } [-become Name]
```

```
teamc feature -cancel Name ... -family Name [-remarks Text]
[-become Name] [-verbose]
```

```
teamc feature -configInfo -family Name [-become Name] [-raw]
```

```
teamc feature -design Name ... -family Name [-remarks Text]
[-become Name] [-verbose]
```

```
teamc feature -modify Name ... -family Name { -prefix Name
-target Name -reference Name -originator Name -abstract Text
-priority Name -name Name } [-remarks Text] [-become Name]
[-verbose]
```

Feature

```
teamc feature -note Name ... -remarks Text -family Name  
[-become Name] [-verbose]
```

```
teamc feature -open -remarks Text -component Name -family Name  
[-name Name] [-verbose] [-prefix Name]* [-reference Name]  
[-abstract Text] [-become Name]
```

```
teamc feature -reopen Name ... -family Name [-remarks Text]  
[-become Name] [-verbose]
```

```
teamc feature -return Name ... -family Name [-duplicate Name]  
[-remarks Text] [-become Name] [-verbose]
```

```
teamc feature -review Name ... -family Name [-remarks Text]  
[-become Name] [-verbose]
```

```
teamc feature -size Name ... -family Name [-remarks Text]  
[-become Name] [-verbose]
```

```
teamc feature -verify Name ... -family Name [-remarks Text]  
[-become Name] [-verbose]
```

```
teamc feature -view Name ... -family Name [-processInfo | -long]  
[-become Name] [-verbose]
```

Notes:

* required when no default value is set for the TeamConnection family

feature -accept Name . . .

Accepts features, in the open or review state, depending on the subprocess configuration of the component so that problems can be resolved. You can specify fields to be user-configurable using the **-configField** attribute overriding field defaults.

Attributes

The feature -accept Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-configField Name

Specify a field to be user-configurable, overriding the field default. NULL is a valid value.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-remarks Text

Describes the change being requested, the actual design for the defect or feature, or the reason for modifying or changing the state of the defect or feature. Remarks are limited to 29,999 characters. If you need to include more than this limit, break your remarks into smaller files and load them from these files. After you issue a command that adds remarks, you cannot change the remarks (that is, you cannot use the **-modify** action to change the remarks).

Note: To move a defect or feature to the size state, you must have entered some design text using the **-remarks** flag with the **-design** action.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

Assume that you own feature **4312** and that it is currently in the review state. After you have reviewed the feature information, you decide to accept the feature for implementation. The following command accepts the feature and moves it to the working state. Work areas and fix records are created according to the sizing records for this feature.

```
teamc feature -accept 4312
```

feature -assign Name . . .

Assigns features to another owner or another component.

Attributes

The feature -assign Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-component Name

The component associated with the object. Different components can manage different versions of the same object.

(Environment variable: TC_COMPONENT.)

-component Name

Feature

The component associated with the object. Different components can manage different versions of the same object. Either -component or -owner is required.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-owner Name

Specifies the user ID of the owner of the object. Either -component or -owner is required.

-remarks Text

Describes the change being requested, the actual design for the defect or feature, or the reason for modifying or changing the state of the defect or feature. Remarks are limited to 29,999 characters. If you need to include more than this limit, break your remarks into smaller files and load them from these files. After you issue a command that adds remarks, you cannot change the remarks (that is, you cannot use the **-modify** action to change the remarks).

Note: To move a defect or feature to the size state, you must have entered some design text using the **-remarks** flag with the **-design** action.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

Assume that you are the owner of feature **4312**. The following command assigns feature 4312 to the **graphix** component; the owner of **graphix** becomes the new owner of this feature.

```
teamc feature -assign 4312 -component graphix
```

feature -cancel Name . . .

Cancels features that are in the open or the returned state.

Attributes

The feature -cancel Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-remarks Text

Describes the change being requested, the actual design for the defect or feature, or the reason for modifying or changing the state of the defect or feature. Remarks are limited to 29,999 characters. If you need to include more than this limit, break your remarks into smaller files and load them from these files. After you issue a command that adds remarks, you cannot change the remarks (that is, you cannot use the **-modify** action to change the remarks).

Note: To move a defect or feature to the size state, you must have entered some design text using the **-remarks** flag with the **-design** action.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

Assume that you are the originator of feature **4298** and that it is currently in the returned state. The following command cancels this feature.

```
teamc feature -cancel 4298
```

feature -configInfo

The feature -configInfo action shows configurable field properties for features in the specified family.

Attributes

The feature -configInfo command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-raw

Displays configurable field information in raw format.

Feature

Examples

- The following command displays the configurable fields defined for the features in family **rdev**.

```
teamc feature -configInfo -family rdev
```

The following is an example of the output provided for this command. This example assumes that the only configurable field defined is called **feaTest**.

Attribute Name	DB Column Name	Create/Required	Type
feaTest	feaTest	yes/no	test

- The following command displays the configurable fields defined for users in family **rdev** in raw format.

```
teamc feature -configInfo -family rdev -raw
```

The following is an example of the output provided for this command. This example assumes that the only configurable field defined is called **feaTest**.

```
Feature Test|Feature Test|feaTest|feaTest|yes|no|test
```

feature -design Name . . .

Moves features to the design state or specifies design text. Features can move to the design state from the open, returned, design, size, or review state.

Attributes

The feature -design Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-remarks Text

Describes the change being requested, the actual design for the defect or feature, or the reason for modifying or changing the state of the defect or feature. Remarks are limited to 29,999 characters. If you need to include more than this limit, break your remarks into smaller files and load them from these files. After you issue a command that adds remarks, you cannot change the remarks (that is, you cannot use the **-modify** action to change the remarks).

Note: To move a defect or feature to the size state, you must have entered some design text using the **-remarks** flag with the **-design** action.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

Assume that you are a component owner and that feature **4312** was opened against your component. The following command moves the feature to the design state. You can issue the **feature -design** command with the **-remarks** flag when you are ready to enter actual design information. You can issue the **feature -design** command more than once.

```
teamc feature -design 4312
```

feature -modify Name . . .

The feature -modify Name . . . action modifies selected properties of features. At least one of these attributes is required with the feature -modify Name . . . action.

- abstract**
- name**
- originator**
- prefix**
- priority***
- reference**
- target***

You cannot modify existing remarks in a feature.

*Because your family administrator can modify or delete certain configurable fields and create new fields, the attributes for the **-modify** action might differ from those in your family or might not appear at all. Those listed here represent the shipped default fields only. For a list of the field properties and flags in use in your family, use the **feature -configInfo** command or see your family administrator. For more information on configurable fields, refer to the *Administrator's Guide*.

Attributes

The feature -modify Name . . . command takes the following attributes:

-abstract Text

Lets users enter concise text to summarize a defect or feature. Up to 63 characters are allowed. This text appears in reports and notification messages. If this flag is not specified when you are opening a defect or feature, the first 63 characters or the text up to the first new-line character of the **-remarks** flag serves as the abstract.

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

Feature

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-name Name

Specifies the defect or feature identifier. Up to 15 alphanumeric characters are allowed for user-generated IDs. The TeamConnection product checks the uniqueness of the ID. If the ID already exists in the TeamConnection product, the action fails and you receive a message indicating that the identifier is not unique. You must then enter a new identifier or let the TeamConnection product generate one.

-originator Name

Specifies the user ID of the new originator when you modify a defect or feature. The previous originator's verification record is automatically reassigned to the new originator.

-prefix Name

Identifies a prefix that categorizes the defect or feature by type. This value precedes the identifier in report output. If no default has been set for this attribute, then you must include it with the command. This attribute is configurable.

-priority Name

When modifying a defect, specifies the timing or scheduling requirements for resolving a defect. This attribute is configurable.

-reference Name

Assigns a value, name, or keyword to a defect or feature.

-remarks Text

Describes the change being requested, the actual design for the defect or feature, or the reason for modifying or changing the state of the defect or feature. Remarks are limited to 29,999 characters. If you need to include more than this limit, break your remarks into smaller files and load them from these files. After you issue a command that adds remarks, you cannot change the remarks (that is, you cannot use the **-modify** action to change the remarks).

Note: To move a defect or feature to the size state, you must have entered some design text using the **-remarks** flag with the **-design** action.

-target Name

Specifies a target (such as, a driver or a date) for defect resolution or availability.

-verbose

TeamConnection displays a confirmation message after you issue the command.

feature -note Name . . .

Adds remarks to features. These notes cannot be modified or deleted after they are in the system.

Attributes

The feature -note Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-remarks Text

Describes the change being requested, the actual design for the defect or feature, or the reason for modifying or changing the state of the defect or feature. Remarks are limited to 29,999 characters. If you need to include more than this limit, break your remarks into smaller files and load them from these files. After you issue a command that adds remarks, you cannot change the remarks (that is, you cannot use the **-modify** action to change the remarks).

Note: To move a defect or feature to the size state, you must have entered some design text using the **-remarks** flag with the **-design** action.

-verbose

TeamConnection displays a confirmation message after you issue the command.

feature -open

Opens a feature. A unique identifier is generated by the TeamConnection product to identify the new feature unless you specify an identifier using the optional **-name** flag.

Because your family administrator can modify or delete certain configurable feature fields and create new fields, the attributes for the **-open** action might differ from those in your family or might not appear at all. Those listed here represent the shipped default fields only. For a list of the field properties and flags in use in your family, use the **feature -configInfo** command or see your family administrator. For more information on configurable fields, refer to the *Administrator's Guide*.

Feature Attributes

The feature -open command takes the following attributes:

-abstract Text

Lets users enter concise text to summarize a defect or feature. Up to 63 characters are allowed. This text appears in reports and notification messages. If this flag is not specified when you are opening a defect or feature, the first 63 characters or the text up to the first new-line character of the **-remarks** flag serves as the abstract.

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-component Name

The component associated with the object. Different components can manage different versions of the same object.

(Environment variable: TC_COMPONENT.)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-name Name

Specifies the defect or feature identifier. Up to 15 alphanumeric characters are allowed for user-generated IDs. The TeamConnection product checks the uniqueness of the ID. If the ID already exists in the TeamConnection product, the action fails and you receive a message indicating that the identifier is not unique. You must then enter a new identifier or let the TeamConnection product generate one.

-prefix Name

Identifies a prefix that categorizes the defect or feature by type. This value precedes the identifier in report output. If no default has been set for this attribute, then you must include it with the command. This attribute is configurable.

-reference Name

Assigns a value, name, or keyword to a defect or feature.

-remarks Text

Describes the change being requested, the actual design for the defect or feature, or the reason for modifying or changing the state of the defect or feature. Remarks are limited to 29,999 characters. If you need to include more than this limit, break

your remarks into smaller files and load them from these files. After you issue a command that adds remarks, you cannot change the remarks (that is, you cannot use the **-modify** action to change the remarks).

Note: To move a defect or feature to the size state, you must have entered some design text using the **-remarks** flag with the **-design** action.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

The following command opens a feature against the **debugr** component, assuming a default prefix value is set. This command creates a feature change request against the **debugr** component. The feature identifier appears on the screen when the command is completed. The person issuing this command is the originator of the feature. The owner of component **debugr** is the owner of the feature.

```
teamc feature -open -remarks "Change format of parameter values display"
-component debugr
```

feature -reopen Name . . .

Reopens features that are in the returned or canceled state.

Attributes

The feature -reopen Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-remarks Text

Describes the change being requested, the actual design for the defect or feature, or the reason for modifying or changing the state of the defect or feature. Remarks are limited to 29,999 characters. If you need to include more than this limit, break your remarks into smaller files and load them from these files. After you issue a command that adds remarks, you cannot change the remarks (that is, you cannot use the **-modify** action to change the remarks).

Note: To move a defect or feature to the size state, you must have entered some design text using the **-remarks** flag with the **-design** action.

-verbose

Feature

TeamConnection displays a confirmation message after you issue the command.

Examples

Assume that you are the originator of feature **4245** and that it is currently in the canceled state. The following command reopens the feature against the component that owned it when the feature was canceled.

```
teamc feature -reopen 4245 -remarks "Disagree with restriction  
classification"
```

feature -return Name . . .

Returns features from any state except the verify, closed, or canceled states. A feature in the working state can be returned only if it does not have work areas associated with it.

Attributes

The feature -return Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-duplicate Name

Specifies that another feature (that is not canceled, returned, or closed) already exists to address the feature being returned.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-remarks Text

Describes the change being requested, the actual design for the defect or feature, or the reason for modifying or changing the state of the defect or feature. Remarks are limited to 29,999 characters. If you need to include more than this limit, break your remarks into smaller files and load them from these files. After you issue a command that adds remarks, you cannot change the remarks (that is, you cannot use the **-modify** action to change the remarks).

Note: To move a defect or feature to the size state, you must have entered some design text using the **-remarks** flag with the **-design** action.

-verbose

TeamConnection displays a confirmation message after you issue the command.

feature -review Name . . .

Moves features from the size state to the review state so that the proposed design implementation and sizing information can be reviewed.

Attributes

The feature -review Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-remarks Text

Describes the change being requested, the actual design for the defect or feature, or the reason for modifying or changing the state of the defect or feature. Remarks are limited to 29,999 characters. If you need to include more than this limit, break your remarks into smaller files and load them from these files. After you issue a command that adds remarks, you cannot change the remarks (that is, you cannot use the **-modify** action to change the remarks).

Note: To move a defect or feature to the size state, you must have entered some design text using the **-remarks** flag with the **-design** action.

-verbose

TeamConnection displays a confirmation message after you issue the command.

feature -size Name . . .

Moves features from the design state to the size state for sizing. Design text must first be entered using **feature -design -remarks**.

Attributes

The feature -size Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

Feature

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-remarks Text

Describes the change being requested, the actual design for the defect or feature, or the reason for modifying or changing the state of the defect or feature. Remarks are limited to 29,999 characters. If you need to include more than this limit, break your remarks into smaller files and load them from these files. After you issue a command that adds remarks, you cannot change the remarks (that is, you cannot use the **-modify** action to change the remarks).

Note: To move a defect or feature to the size state, you must have entered some design text using the **-remarks** flag with the **-design** action.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

Assume that you are the owner of feature **3129**. This feature is in the design state and text has been entered using **feature -design -remarks**. The following command moves the feature to the size state. After the feature is in the size state, you can create sizing records using the **size** command. (One sizing record is required for each component and release combination affected by the feature change.)

```
teamc feature -size 3129
```

feature -verify Name . . .

Moves features from the working state to the verify state.

Attributes

The feature -verify Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-remarks Text

Describes the change being requested, the actual design for the defect or feature, or the reason for modifying or changing the state of the defect or feature. Remarks are limited to 29,999 characters. If you need to include more than this limit, break your remarks into smaller files and load them from these files. After you issue a command that adds remarks, you cannot change the remarks (that is, you cannot use the **-modify** action to change the remarks).

Note: To move a defect or feature to the size state, you must have entered some design text using the **-remarks** flag with the **-design** action.

-verbose

TeamConnection displays a confirmation message after you issue the command.

feature -view Name . . .

Shows information about features.

Attributes

The feature -view Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-long

Displays detailed information for the specified objects. Specify only one of -long or -processInfo.

-processInfo

Displays the current process setting and associated TeamConnection subprocesses for the specified components when used with the **-view** action flag. Specify only one of -long or -processInfo.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

The following command displays information about feature **1244**, including its purpose, originator, owner, and current state.

Feature

teamc feature -view 1244

Related information

See the following related commands:

- Defect
- Fix
- Report
- Size
- Verify
- Workarea

To see the feature state diagrams, refer to the poster *Staying on Track with TeamConnection Processes*.

Chapter 14. Fix

Fix records are associated with work areas. A fix record reflects the status of all the part changes made to resolve a defect or implement a feature for a work area and release in reference to one component. A work area has one or more fix records associated with it, depending on the number of components in which parts are changed. The component manages the parts to be changed in relation to the work area.

Use the **fix** command to create, delete, and reassign fix records and to change the state of fix records.

Each fix record is uniquely identified by a work area, a release, and a component. The owner of a fix record is, by default, the owner of the related component; however, this ownership can be reassigned using the **-assign** action flag.

Each fix record refers to the part changes required within one component. The state of the fix record indicates the state of part changes for that component.

When the defect or feature is accepted, and the design, size, and review process is enabled for the component the defect or feature is assigned to, one work area is created for each release referenced by a sizing record, if the release has the track process enabled. Fix records are created for all sizing records in this scenario. Fix records are created in the notReady state if the associated work area is in the approve state; otherwise, they are created in the ready state. Additional fix records are created if parts are changed and checked in to the TeamConnection product for a work area associated with a defect or a feature in a component for which there is no existing fix record. In this case, the fix record is in the active state. Active state means that part changes have been checked in to a work area associated with a defect or feature in the component. You can create fix records using the **-create** action flag if a work area is in the approve state or the fix state.

Use the **-complete** action flag to indicate that the part changes for the work area associated with a defect or feature within that component are completed. This moves the fix record to the complete state.

When all fix records for the work area are completed, the work area moves from the fix state to the integrate state. Use the **-activate** action flag to reactivate a fix record that is in the complete state if additional part changes are needed. You must change the work area state from integrate to fix before activating the fix records. You can make changes in the work area only when it is in the fix state.

If you decide that no part changes are required for a component that has a fix record, you can use the **-delete** action flag to delete the fix record from the associated work area.

Command summary

In the following syntax statements, the first letter of each argument is capitalized, for example `-family Name`. Replace these arguments with a value that is valid for your TeamConnection database. If you want to issue a command for the family named **testfam**, for example, replace `-family Name` with `—family testfam`.

The syntax statements for the **fix** command are:

Fix

```
teamc fix -activate -workarea Name ... -family Name  
-release Name ... -component Name [-become Name] [-verbose]  
  
teamc fix -assign -to Name -workarea Name ... -release Name ...  
-component Name -family Name [-become Name] [-verbose]  
  
teamc fix -complete -workarea Name ... -family Name  
-release Name ... -component Name [-become Name] [-verbose]  
  
teamc fix -create -workarea Name ... -release Name ...  
-component Name -family Name [-developer Name] [-become Name]  
[-verbose]  
  
teamc fix -delete -workarea Name ... -release Name ...  
-component Name -family Name [-become Name] [-verbose]
```

fix -activate

Moves a fix record from the complete state to the active state so that additional part changes can be made. You can change the fix record state only when the corresponding work area is in the fix state.

To perform this action, the associated release's process must include the fix subprocess.

Attributes

The fix -activate command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-component Name

The component associated with the object. Different components can manage different versions of the same object.

(Environment variable: TC_COMPONENT.)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-verbose

TeamConnection displays a confirmation message after you issue the command.

-workarea Name . . .

The work area name.

(Environment variable: TC_WORKAREA)

Examples

Assume additional part changes are required to parts managed by component **graphix** for work area **412** and release **font38**. Also, assume that the TC_COMPONENT and the TC_RELEASE environment variables are set to **graphix** and **font38**, respectively, and the work area is in the fix state. The following command reactivates the fix record. The fix record for work area **412** in the component and release specified in the environment variables is moved to the active state, and additional part changes can now be checked in.

```
teamc fix -activate -workarea 412
```

fix -assign

Assigns ownership of a fix record to another user ID. You cannot reassign the component.

To perform this action, the associated release's process must include the fix subprocess.

Attributes

The fix -assign command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-component Name

The component associated with the object. Different components can manage different versions of the same object.

(Environment variable: TC_COMPONENT.)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

Fix

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-to Name

The user ID to which you want to reassign the object. The user ID you specify becomes the owner of the object.

-verbose

TeamConnection displays a confirmation message after you issue the command.

-workarea Name . . .

The work area name.

(Environment variable: TC_WORKAREA)

Examples

Assume that you own a fix record that monitors changes made to parts in the **debugr** component for work area **955** in release **21gos**. The following command reassigns the fix record to **joel**, assuming the TC_RELEASE environment variable is set to **21gos**. If your TC_RELEASE environment variable were not set to the proper release, you would have to use the **-release** attribute flag.

```
teamc fix -assign -workarea 955 -component debugr -to joel
```

fix -complete

Moves a fix record to the complete state to indicate that all part changes required in the associated component are completed. If no other fix records exist or all other records are completed, this causes the work area to change from the fix state to the next valid state, which is governed by the release's process.

To perform this action, the associated release's process must include the fix subprocess.

Attributes

The fix -complete command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-component Name

The component associated with the object. Different components can manage different versions of the same object.

(Environment variable: TC_COMPONENT.)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-verbose

TeamConnection displays a confirmation message after you issue the command.

-workarea Name . . .

The work area name.

(Environment variable: TC_WORKAREA)

fix -create

Creates a fix record for a work area in relation to a component. You can create a fix record only if the work area is in the approve state or in the fix state.

To perform this action, the associated release's process must include the fix subprocess.

Attributes

The fix -create command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-component Name

The component associated with the object. Different components can manage different versions of the same object.

(Environment variable: TC_COMPONENT.)

-developer Name

When creating a fix record, specifies the user ID of the owner of the fix record.

-family Name

Fix

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-verbose

TeamConnection displays a confirmation message after you issue the command.

-workarea Name . . .

The work area name.

(Environment variable: TC_WORKAREA)

Examples

Assume that part changes are required to resolve defect **909** in release **21gos**. The component **graphix** manages the parts to be changed. The TeamConnection product creates work area **909** for the changes. The following command creates a fix record to monitor changes made to parts in the **graphix** component to resolve defect **909** for the release **21gos**. The owner of the new fix record is the owner of the **graphix** component.

```
teamc fix -create -workarea 909 -component graphix -release 21gos
```

fix -delete

Deletes the fix record for the specified work area and component. You cannot delete a fix record that is in the active state or in the complete state because it has part changes associated with it.

Attributes

The fix -delete command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-component Name

The component associated with the object. Different components can manage different versions of the same object.

(Environment variable: TC_COMPONENT.)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-verbose

TeamConnection displays a confirmation message after you issue the command.

-workarea Name . . .

The work area name.

(Environment variable: TC_WORKAREA)

Related information

See the following related commands:

- Component
- Part
- Release
- Workarea

Fix

Chapter 15. Host

Use the **host** command to create and delete entries on a TeamConnection user's host list. Host list entries are only required if the family is using host-based security.

Each entry identifies client access for a user ID on one host and consists of a user ID and a host in the format **login@hostName**. The **host** command is used in conjunction with the **user** command when you initially create a new user ID. A host list is attached to a user ID and must have at least one entry to establish client access for the user. Additional entries can be defined to let a user complete TeamConnection commands from multiple hosts (and user IDs).

A TeamConnection superuser must create the first host list entry for a new user ID. The owner of the user ID can make subsequent entries to gain client access on the hosts where the user has logins. Each user ID can have multiple host list entries.

When you use the **-become** flag or the TC_BECOME environment variable, the host list must have an entry of the user ID specified by the **-become** flag. This entry gives you authority to act on behalf of that user ID.

Host list entries can be deleted; however, a user ID must always have one host list entry to be able to access the TeamConnection product. If all host list entries are deleted for a user ID, only a TeamConnection superuser can create a host list entry to reestablish client access for that user ID.

There is a difference between the *graphical user interface (GUI)* host command and the TeamConnection command line that the GUI prepares. The following fields appear when you add a host entry to the list:

Logins	smith
Host names	carcps20
User IDs	jane

However, the GUI prepares the following TeamConnection command line:

```
teamc host -create smith@carcps20 -login jane
```

Note that **-create smith** is the login for the GUI and that **-login jane** is the user ID for the GUI. Thus, the GUI login field is not the same as the TeamConnection line command **-login**.

Command summary

In the following syntax statements, the first letter of each argument is capitalized, for example **-family Name**. Replace these arguments with a value that is valid for your TeamConnection database. If you want to issue a command for the family named **testfam**, for example, replace **-family Name** with **—family testfam**.

The syntax statements for the **host** command are:

```
teamc host -create Name ... -family Name [-login Name] [-become Name]
[-verbose]
```

```
teamc host -delete Name ... -family Name [-login Name] [-become Name]
[-verbose]
```

host -create Name . . .

Creates one or more host list entries for an existing user ID, using the format *login@hostName*. The login is optional if it matches the user's current login.

The initial host list entry for each user must be created by someone with TeamConnection superuser privilege. During the create operation, the host login is resolved. The login **myUser@myHostname**, for example, is resolved to **myUser@myHostname.myCompany.com**.

Should you attempt to create a user at an unreachable host, an error message will appear.

Attributes

The host -create Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-login Name

TeamConnection user ID.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

Assume that your login on host **lab1** is **jane** and you have an identical TeamConnection user ID that has a host list entry for **lab1**; that is, the TeamConnection user ID **jane** has a host entry for **jane@lab1**. You also have the login **jane** on two other hosts, **lab2** and **lab3** (also on the network), and you want to use the TeamConnection product on those hosts as well. The following command gives user **jane** TeamConnection client access on these additional hosts. This command creates host list entries **jane@lab2** and **jane@lab3** for the TeamConnection user ID **jane**.

```
teamc host -create lab2 lab3 -login jane
```

Because your login on the current host is identical to the TeamConnection user ID for which you are making a host list entry, the command could also be:

```
teamc host -create lab2 lab3
```

Assume that **jane** and **pete** are logins on host **lab2**. The following command gives them access to the **admin** TeamConnection user ID at that host. Adding this entry host list entry for the user ID **admin** lets logins **jane** and **pete** perform TeamConnection commands using the TeamConnection user ID **admin** while logged on to the host **lab2**. They can use the **-become** attribute flag to move between the **admin** user ID and their own user IDs.

Note: You assume the access authority of the user ID you specify.

```
teamc host -create jane@lab2 pete@lab2 -login admin
```

host -delete Name . . .

Deletes one or more host list entries for an existing user ID, using the format *login@hostName*. The login is optional if it matches the user's current login. Each user must have at least one host list entry to have TeamConnection access.

During the delete operation, the host login is not resolved. You must use a fully qualified host name. To delete the host list entry for login **myUser@myHostname**, for example, you need to issue the command with the fully qualified login **myUser@myHostname.myCompany.com**.

Attributes

The host -delete Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-login Name

TeamConnection user ID.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

Assume that your user ID **joan** has a number of host list entries, including one for the host **johnson.kap.uwo.com**. The following command deletes the entry from the host list associated with your user ID. Your user ID, **joan**, can no longer perform TeamConnection commands from that host.

```
teamc host -delete joan@johnson.kap.uwo.com
```

Host

Related information

See the following related commands:

User

Chapter 16. Notify

By default, you receive notification when an action is required on your part. To receive additional notification, you can add entries to notification lists for specific components.

Use the **notify** command to create or delete entries on a component notification list. Each entry associates a user ID with a notification interest group. An interest group represents a list of actions; whenever an action on this list is performed, the users who are associated with the interest group receive notification. For a list of the notification interest groups shipped with TeamConnection, refer to the *TeamConnection User's Guide*. Your family administrator can modify or create new interest groups.

When you create the user ID, the TeamConnection product sends notification messages to the address you specified for each user ID. You can modify the address using **user -modify**.

A user ID can have more than one entry on the notification list for a given component. Interest groups defined on notification lists are inherited down the component hierarchy, but the notification lists of child components do not show the notification list entries that are inherited from ancestor components.

Command summary

In the following syntax statements, the first letter of each argument is capitalized, for example `-family Name`. Replace these arguments with a value that is valid for your TeamConnection database. If you want to issue a command for the family named **testfam**, for example, replace `-family Name` with `—family testfam`.

The syntax statements for the **notify** command are:

```
teamc notify -create -login Name ... -interest Name -component Name
              -family Name [-become Name] [-verbose]
```

```
teamc notify -delete -login Name ... -interest Name -component Name
              -family Name [-become Name] [-verbose]
```

notify -create

Creates one or more notification list entries for a specified component.

Attributes

The `notify -create` command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

Notify

-component Name

The component associated with the object. Different components can manage different versions of the same object.

(Environment variable: TC_COMPONENT.)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-interest Name

A preconfigured notification interest group for the specified user ID. For a list of interest groups, refer to the *TeamConnection User's Guide* or issue the following **report** command:

```
teamc report -view interest -where "order by name"
```

-login Name . . .

One or more TeamConnection user IDs. Specify only one of -inherited or -login.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

You own the **graphix** component. The following command creates notification list entries for the owners of user IDs **pam**, **jack**, and **lisa** with **general** notification interest for that component. Three entries are added to the notification list associated with the **graphix** component, one for each of the user IDs you specified with the **-login** attribute flag. Each user is notified when an action configured in the **general** notification interest group is performed in reference to the **graphix** component or to any child components of that component. If this group includes the DefectOpen action, then these users are notified each time a defect is opened against the **graphix** component.

```
teamc notify -create -login pam jack lisa -interest general  
-component graphix
```

notify -delete

Deletes one or more notification list entries from the specified component. Owners of user IDs do not need special authority to delete their user IDs from a notification list.

Attributes

The notify -delete command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-component Name

The component associated with the object. Different components can manage different versions of the same object.

(Environment variable: TC_COMPONENT.)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-interest Name

A preconfigured notification interest group for the specified user ID. For a list of interest groups, refer to the *TeamConnection User's Guide* or issue the following **report** command:

```
teamc report -view interest -where "order by name"
```

-login Name . . .

One or more TeamConnection user IDs. Specify only one of -inherited or -login.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

You do not own the **debugr** component, but you want to remove a notification list entry for that component for your own user ID, **pam**. The following command removes the notification list entry that gives you **developer** interest for that component. You are no longer notified when actions configured for the **developer** interest group are performed in reference to the **debugr** component or its child components.

```
teamc notify -delete -login pam -interest developer -component debugr
```

Related information

See the following related commands:

Component

Report

Use the **report** command to view the interest groups and the actions they include. For example:

```
teamc report -view interest -where "order by name"
```

Notify

For a list of the notification groups shipped with TeamConnection, refer to the *TeamConnection User's Guide*.

See your family administrator for information about configuring new notification interest groups and modifying existing ones.

Chapter 17. Parser

Use the **parser** command to create, modify, view, or delete parsers.

To automate some of the work involved in defining and maintaining a build tree, you can use a parser object. The task of a parser is to inspect part contents to determine dependencies. For example, a parser can inspect `hello.c`, recognize that it has a dependency on `hello.h`, and create that dependency in the TeamConnection build tree.

Because parsers are language-dependent, you probably need a different parser for each language you use in a particular release. For example, you might have both a COBOL parser and a C parser in a release. Many parts in the release can use the same parser.

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

Command summary

In the following syntax statements, the first letter of each argument is capitalized, for example `-family Name`. Replace these arguments with a value that is valid for your TeamConnection database. If you want to issue a command for the family named **testfam**, for example, replace `-family Name` with **—family testfam**.

The syntax statements for the **parser** command are:

```
teamc parser -create Name -command Name -release Name -family Name
               [-include Paths]
               [-become UserName] [-verbose]
```

```
teamc parser -delete Name -release Name -family Name
               [-become UserName] [-verbose]
```

```
teamc parser -modify Name -release Name -family Name
               [-command Name] [-include Paths]
               [-become UserName] [-verbose]
```

```
teamc parser -view Name -release Name -family Name
               [-become UserName] [-verbose]
```

parser -create Name

Creates a new parser.

Attributes

The `parser -create Name` command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

Parser

(Environment variable: TC_BECOME)

-command Name

Specifies the command file you want to associate with the parser. This can be an .exe, a .com, a .cmd, or a .bat file. The executable file needs to be in the execution path of the TeamConnection family server.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-include Paths

Specifies a concatenated set of paths that define where the parser looks for parts when processing the set of dependencies returned from the command file. These dependencies come in two types:

- A dependency in which the file is stored in the TeamConnection database. For example, hello.c includes hello.h, and both files are stored in the TeamConnection database. During a build, these dependencies must be extracted to a path accessible by the build processor.
- A dependency on a file that is not stored in the TeamConnection database. An example of such a dependency is stdio.h, which is typically stored in a compiler's include path and not in the TeamConnection database.

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

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-verbose

TeamConnection displays a confirmation message after you issue the command.

parser -delete Name

Deletes the specified parser.

Attributes

The parser -delete Name command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-verbose

TeamConnection displays a confirmation message after you issue the command.

parser -modify Name

Modifies an existing parser.

Attributes

The parser -modify Name command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-command Name

Specifies the command file you want to associate with the parser. This can be an .exe, a .com, a .cmd, or a .bat file. The executable file needs to be in the execution path of the TeamConnection family server.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-include Paths

Specifies a concatenated set of paths that define where the parser looks for parts when processing the set of dependencies returned from the command file. These dependencies come in two types:

- A dependency in which the file is stored in the TeamConnection database. For example, hello.c includes hello.h, and both files are stored in the TeamConnection database. During a build, these dependencies must be extracted to a path accessible by the build processor.

Parser

- A dependency on a file that is not stored in the TeamConnection database. An example of such a dependency is `stdio.h`, which is typically stored in a compiler's include path and not in the TeamConnection database.

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

-release Name

The release for which this command is being issued.

(Environment variable: `TC_RELEASE`)

-verbose

TeamConnection displays a confirmation message after you issue the command.

parser -view Name

Gets the specified parser definition.

Attributes

The parser -view Name command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: `TC_BECOME`)

-family Name

The family for which this command is being issued.

(Environment variable: `TC_FAMILY`)

-release Name

The release for which this command is being issued.

(Environment variable: `TC_RELEASE`)

-verbose

TeamConnection displays a confirmation message after you issue the command.

Related information

See the following related commands:

- Builder
- Part
- Release

Chapter 18. Part

Use the **part** command to put files or parts into the TeamConnection product and to work with individual files afterwards. For example, you can store VisualAge Generator parts in the TeamConnection product. You put a part into the TeamConnection product using **part -create**. When you create a part, you can do any of the following:

- Take an existing file that is on your workstation and place it in the TeamConnection product.
- Create a place-holder part that will not have any content until a successful build of its inputs, using the **-empty** flag. Place holders are used in building an application.
- Create a part that acts as a collector object so that you can synchronize the build of unrelated parts. For more information, refer to the *TeamConnection User's Guide*. These parts do not have bulk contents (such as ASCII or binary data).

When you create a part in the TeamConnection product, you associate it with a release (to relate the part to a development effort) and with a component (to control the ownership of and the access to a part). You have the option of specifying a file mode when creating a part using the **-fmode** flag. For example, you can use the **-fmode** flag to specify that a part is read-only. If file mode is not specified, the current file mode is used. After a part is successfully created on the server, the TeamConnection product modifies the file attributes of the working copy of the part left on the client to read-only. All subsequent **part** access commands must specify the part name, release name, and possibly a work area name to identify the correct part.

By default, parts are created with the part type of TcPart and the file type of text. You can specify a part type other than TcPart. For part types of TcPart, you can specify that the file contents are text, binary, or none.

You can have two parts with the same base name, as long as the **-type** is different for each part. Unless you specify a type on a part, all commands that reference parts default to TcPart. Once the type is specified, it cannot be changed. unless you delete the part and then recreate it.

Parts that have a unique base name or path name within a release can be specified by their base name or path name; other parts must be specified by their full path name when performing TeamConnection actions against the part.

What you can do with parts

You can perform various actions against parts in the TeamConnection product, depending on the access authority you have for the components that manage parts.

- Parts can be extracted or checked out for editing and subsequently checked in to save the changes.
- Part properties can be modified, such as the path name, release, component, and file mode. For example, you can change the name of a part using the **-rename** action.
- Parts can be built. You can use the **-connect** and **-disconnect** actions to indicate how parts are related. You can also use the **-touch** action to ensure that

Part

the part participates in the next build. To cancel a build, you can use the **-cancel** attribute of the **part -build** action. For more information about building parts, refer to the *TeamConnection User's Guide*.

- Parts can be deleted, using the **-delete** action, and then re-created using the **-recreate** action.
- Actions performed on parts can be undone, although certain limitations apply. The **-undo** action negates all of the uncommitted changes to a part in a work area since the last work area **-freeze**.

Working with parts

When working with parts in a release whose process includes the tracking and driver subprocesses, multiple sets of part changes can be checked in and included in one driver; however, other types of part changes can be specified only once in a driver. For example, a part cannot be created and renamed in the same driver. However, an existing unfrozen **-create** action can be undone so that you can create a part with the name you want.

If the release uses concurrent development mode, then multiple users can work on the same part at the same time. This is in contrast to serial mode, where only one user can work on a part at a time. For more information about concurrent development, refer to page 41.

Parts can be linked to identify them as common to more than one release or work area. A common part has identical content in all of the releases or work areas in which it is linked. For example, a part is common to two releases if the same version of the part is being used in those releases. Common parts follow a single path of development.

The TeamConnection product maintains commonality of parts unless you use the **-force** flag to break the link. When a link is broken, the parts still share the same name, but the information contained in the parts is different.

Specifying the full path name of a TeamConnection part

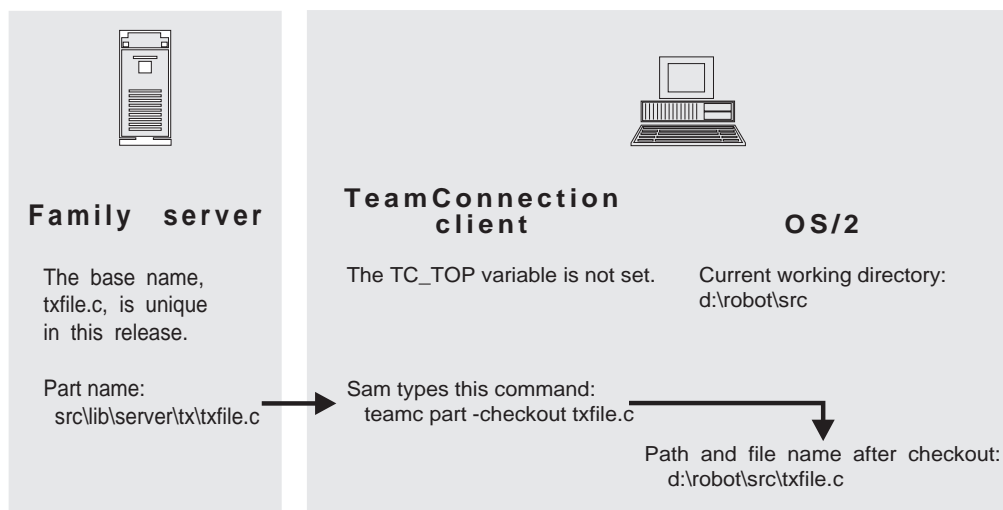
There are a number of ways to specify the full path of a TeamConnection part:

- Specify the part name (which includes TeamConnection path name and base name), and take the workstation path from your current working directory.
- Use the **-relative** flag to specify a path name that is outside your current working directory. In this case, the values for the **-relative** flag and the name are concatenated. This flag can be used only with the **-create**, **-checkout**, **-checkin**, **-extract**, and **-unlock** actions.
- If your workstation directory structure matches the way parts are named, you can set the TC_TOP variable to the leading portion of your current working directory, and specify only the base name in the actions of the **part** command. TeamConnection subtracts the contents of TC_TOP from the current working directory. This option might take some time to understand, but it can save you a great deal of typing.

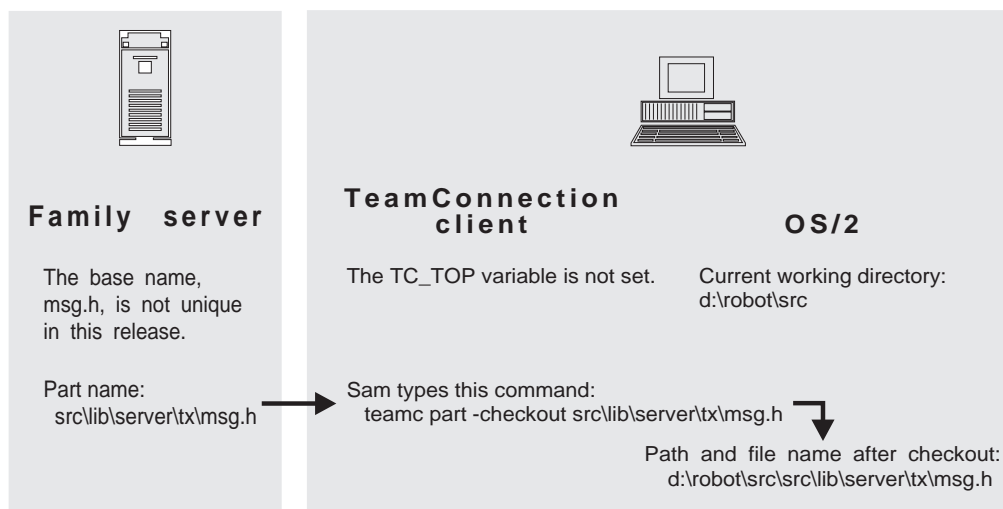
The example that follows starts with the simplest case and moves to the more complex. Throughout the example, assume that the environment variables for TC_FAMILY, TC_RELEASE, and TC_WORKAREA have been set.

Working from your current working directory

Sam is working with a part named txfile.c. The path name and base name for this part is src\lib\server\tx\txfile.c; the current working directory on Sam's client workstation is d:\robot\src. There is no other part in this release named txfile.c.



Next Sam needs to check out \src\lib\server\tx\msg.h. Because other files named msg.h exist in release **9501**, he must specify the entire path name.



Note: If you create a part without specifying **-relative** or setting the TC_TOP variable, then a file with the same name must exist in the current working directory. For example, you use the following command:

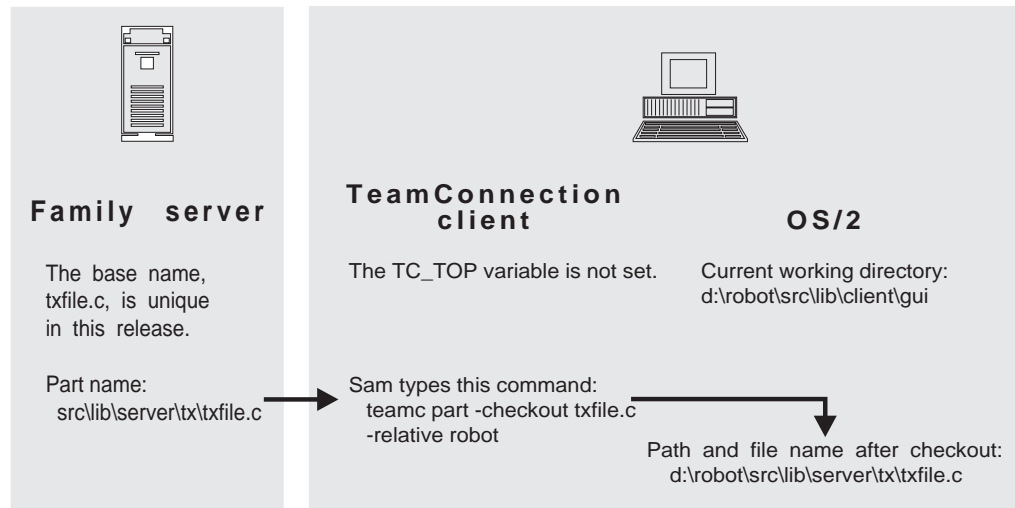
```
teamc part -create src\lib\server\read.me
```

In this case, read.me must exist in your current directory; the TeamConnection product does not search for the part in x:\src\lib\server (x is your current drive).

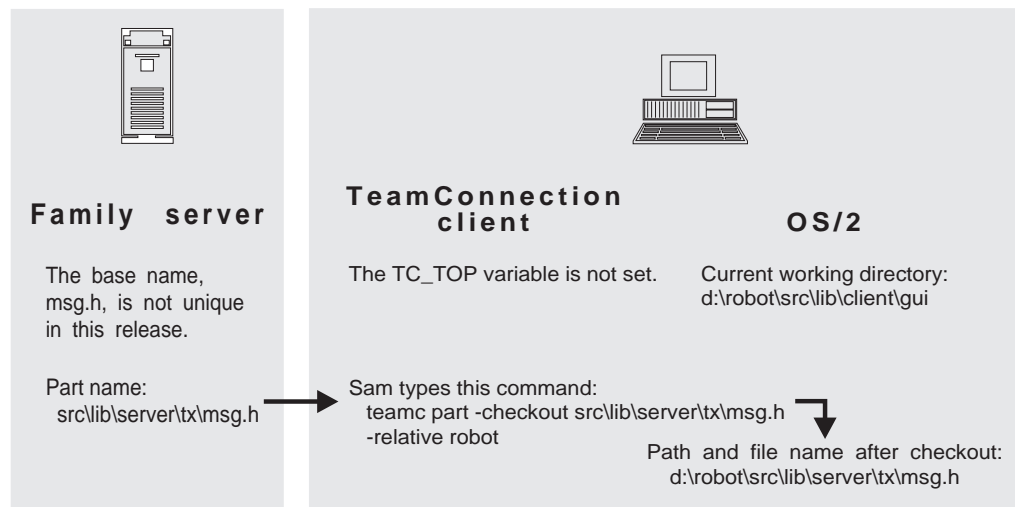
Part

Using the -relative flag

The next day, Sam's current working directory is `d:\robot\src\lib\client\gui`. But he wants to check out `txfile.c` to the same directory as before. He uses the **-relative** flag to do this:



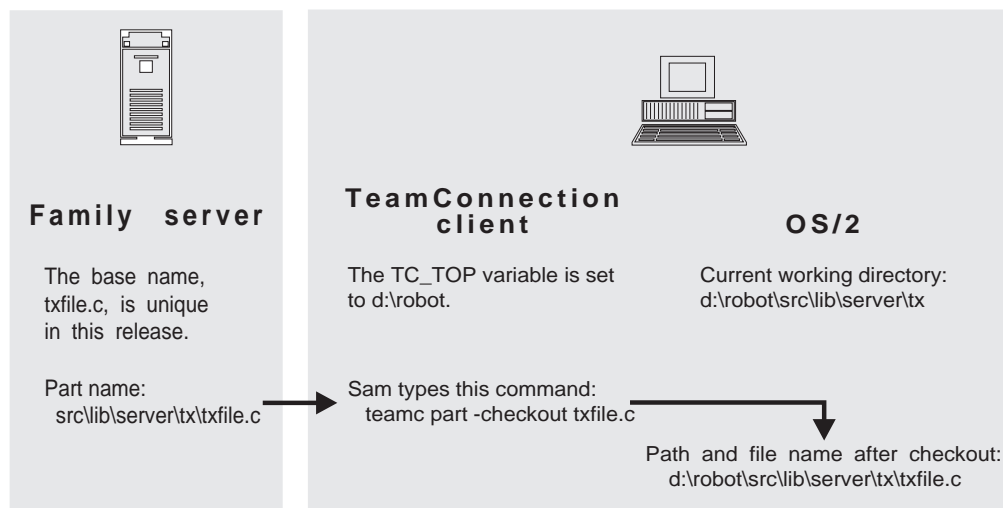
Because `txfile.c` is unique in the release, Sam did not need to specify the entire path name. But `msg.h` is not a unique base name:



Using the TC_TOP variable

A few days later, Sam has checked out many files from the release into subdirectories of `d:\robot`, using a directory structure that mimics the names of TeamConnection parts. For his own convenience, he sets the `TC_TOP` variable to `d:\robot`.

Now when he works with parts, he can specify just the base name of files in his current working directory, as in this example:



The additional information added to the leading portion of the path is formed by subtracting the value of TC_TOP from the current working directory. In other words, the TC_TOP variable tells the TeamConnection product which portion of the current working directory path belongs to the client workstation.

This command is completely equivalent to the following command if the TC_TOP variable is not set:

```
teamc part -checkout src\lib\server\tx\txfile.c -relative d:\robot
```

However, using TC_TOP lets Sam save his fingers for typing code instead of long path names.

Common parts in releases

When a common part is checked out for editing, it is locked in all releases and work areas where it is common. You need to do only one check-in to have the change reflected in all releases and work areas in which the part is common. The TeamConnection product maintains commonality of parts unless you use the **-force** flag to break the common link. If you want to edit a locked part and cannot wait for the part to be checked in, you can break the common link (and thus the lock on the common version) by specifying the **-force** flag when you issue **part -checkout**. The force action applies only in that release and work area so that the part associated with that release is no longer common. You must explicitly link parts to make them common again after that time.

When a common part is checked in using the **part -checkin** command, all work areas must be specified for each release in which the part is common. The associated work areas must be in the fix state, and the associated fix records in the ready or active state.

If you want to check in a common part, and you do not want the changes to be reflected in the other work areas in which the part is common, use the **-force** flag to break the common link. If you want to check in a common part, but you want the changes to be reflected in some of the work areas to which the part is common but not in others, use the the following attributes:

- **-force**

Part

- **-common**, including as arguments the names of releases you want to keep common and excluding the name of the release associated with the part you are checking in or modifying
- **-workarea**, including as arguments the names of the work areas for which you want to maintain part commonality

Command summary

In the following syntax statements, the first letter of each argument is capitalized, for example `-family Name`. Replace these arguments with a value that is valid for your TeamConnection database. If you want to issue a command for the family named **testfam**, for example, replace `-family Name` with **—family testfam**.

The syntax statements for the **part** command are:

```
teamc part -build Name -pool Name -release Name -family Name
           -workarea Name
           {-force | -normal* | -unconditional | -report}
           -detail FileSpec
           -parameters Parameters
           -type Name
           -top Name
           -become Name -verbose
```

```
teamc part -build Name -cancel -release Name -family Name
           -workarea Name
           -type Name
           -top Name
           -become Name -verbose
```

```
teamc part -checkin Name ... -release Name -family Name
           -workarea Name ...
           -common Name ... -force -retainLock -remarks Text
           -binary | -text | -none -stdin | -from fileSpec **
           -relative Name | -top Name
           -become Name -verbose
```

```
teamc part -checkout Name ...
           -workarea Name -release Name -family Name
           -force -stdout
           -relative Name | -top Name
           -become Name -verbose
```

```
teamc part -childInfoView Name ...
           -release Name -family Name -long
           -workarea Name*** | -version Name | -driver
           -type Name
           -become Name -verbose
```

```
teamc part -configInfo -family Name -become Name -raw
```

```
teamc part -connect Name ...-parent Name
           {-input* | -output | -dependent}
           -workarea Name -release Name -family Name
           -type Name
```

```

    -parenttype Name
    -become Name -verbose

teamc part -create Name ... -component Name
    -workarea Name -release Name -family Name
    -builder Name -fmode Octal_number
    -relative Name | -top Name
    -binary | -text* | -none
    -parameters Parameters
    -parent Name {-input* | -output | -dependent}
    -parser Name
    -stdin | -from fileSpec | -empty **
    -parenttype Name*
    -remarks Text
    -temporary* | +temporary
    -become Name -verbose
    [-configField]

teamc part -delete Name ...
    -workarea Name ...
    -release Name -family Name
    -common Name ... -force -type Name
    -top Name -become Name -verbose

teamc part -destroy Name ... -release Name -family Name
    -type Name
    -top Name
    -become Name -verbose

teamc part -disconnect Name ...
    -workarea Name -parent Name -release Name -family Name
    -parenttype Name
    -type Name
    -become Name -verbose

teamc part -export Name ... -type Name [-viewType Name]
    -file Name -family Name [-become Name]
    {-workarea Name | -version Name}
    [-verbose]

teamc part -extract Name ...
    -release Name -family Name
    -workarea Name*** | -version Name | -driver
    -nokeys -relative Name | -top Name -stdout
    -dmask Octal_number -fmask Octal_number
    -become Name -verbose

teamc part -link Name ... -workarea Name -release Name -family Name
    -fromworkarea Name | -version Name -fromrelease Name
    -type Name
    -top Name -become Name -verbose

teamc part -lock Name ...
    -workarea Name
    -release Name -family Name -force
    -type Name

```

Part

-top Name -become Name -verbose

teamc part -modify**** Name ...
-workarea Name -release Name -family Name
{-parameters Parameters
-parser Name -builder Name
-temporary* | +temporary}
-type Name
-top Name -become Name -verbose

teamc part -modify**** Name ...
-release Name -family Name
{-fmode Octal_number -component Name}
-top Name -become Name -verbose

teamc part -recreate Name ...
-workarea Name ...
-release Name -family Name -top Name
-common Name ... -force
-type Name
-become Name -verbose

teamc part -refresh Name ... -workarea Name -release Name -family Name
[-fromworkarea Name | -version Name]
[-type Name]
[-top Name] [-become Name] [-verbose]

teamc part -rename Name -path Name
-workarea Name ...
-release Name -family Name
-top Name -common Name ... -force
-type Name
-become Name -verbose

teamc part -resolve Name ... -release Name -family Name -quiet
-type Name
-top Name -become Name -verbose

teamc part -touch Name ...
-workarea Name
-release Name -family Name -top Name
-type Name
-become Name -verbose

teamc part -undo Name ...
-workarea Name ...
-release Name -family Name
-top Name-common Name ...
-type Name
-force -become Name -verbose

teamc part -unlock Name ...
-workarea Name
-release Name -family Name
-relative Name | -top Name
-type Name

-become Name -verbose

```
teamc part -view Name ...
  -release Name -family Name -long -top Name
  -workarea Name*** | -version Name | -driver
  -type Name
  -become Name -verbose
```

```
teamc part -viewmsg Name ... -release Name -family Name -top Name
  -workarea Name | -version Name | -driver
  -type Name
  -become Name -verbose
```

Notes:

- * default if not specified
 - type and -parenttype will default to TcPart if not specified
 - temporary is the default, meaning NOT a temporary part
- ** Only valid with file type of text or binary.
- *** Required if part has not been committed.

part -build Name

Starts building the specified part.

Attributes

The part -build Name command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-detail FileSpec

Specifies a file in which all of the build messages are collected.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-force

Forces a break between common parts.

-force

Builds all parts, even if they are not out-of-date. Processing stops after the first error is returned.

Part

If you specify this attribute, do not also specify `-normal`, `-unconditional`, or `-report`. If you specify none of these attributes, `-normal` is the default.

-normal

Builds only the parts that are out-of-date. Processing stops after the first error is returned.

If you specify this attribute, do not also specify `-force`, `-unconditional`, or `-report`. If you specify none of these attributes, `-normal` is the default.

-parameters

Specifies the parameters you use to build a part. These parameters replace the ones in the builder that is associated with this part. If you want to include the builder parameters, you can imbed the **\$(BUILDERPARMS)** variable in your part. You can imbed this variable wherever you want the variables substituted in your part.

In UNIX environments, you need to include an escape character before the **\$**: `\$(variable_name)`. The following is an example: `\$(TC_INPUT)`.

-pool Name

Specifies the name of the build pool that contains the build servers that will perform the build.

(Environment variable: `TC_BUILDPOOL`)

-release Name

The release for which this command is being issued.

(Environment variable: `TC_RELEASE`)

-report

Gives a preview of what would be built if you invoked a build. The report identifies what steps would occur without any translations taking place.

If you specify this attribute, do not also specify `-force`, `-normal`, or `-unconditional`. If you specify none of these attributes, `-normal` is the default.

-top Name

Specifies the leading portion of the path name that is a subset of the current working directory on the client machine.

(Environment variable: `TC_TOP`)

-type Name

The type of the parts. The default is `TCPart`.

-unconditional

Builds only parts that are out-of-date, but continues processing even if errors are returned. Outputs are not rebuild for inputs that have failed.

If you specify this attribute, do not also specify `-force`, `-normal`, or `-report`. If you specify none of these attributes, `-normal` is the default.

-verbose

TeamConnection displays a confirmation message after you issue the command.

-workarea Name . . .

The work area name.

(Environment variable: TC_WORKAREA)

Examples

The following command builds **robot.exe** using pool **OS2_486**.

```
teamc part -build robot.exe -pool os2_486 -workarea 501 -release 11debgr
```

part -build Name -cancel

Cancels building the specified part.

Attributes

The part -build Name -cancel command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-top Name

Specifies the leading portion of the path name that is a subset of the current working directory on the client machine.

(Environment variable: TC_TOP)

Part

-type Name

The type of the parts. The default is TcPart.

-verbose

TeamConnection displays a confirmation message after you issue the command.

-workarea Name . . .

The work area name.

(Environment variable: TC_WORKAREA)

part -checkin Name . . .

Submits the changes made to a specified part to the TeamConnection server and unlocks the part. Any associated work areas must be in the fix state and the associated fix records in the ready or active state.

When a part is checked-in successfully, the part mode is read-only. Setting the environment variable **tc_modperm** can change the mode.

- Set to off/OFF part mode is read-only
- Set to on/ON part mode is read-write

Attributes

The part -checkin Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-binary

Specifies that the object is a binary file. Specify only one of -binary, -none, or -text. If you specify none of these attributes, -text is the default.

-common Name . . .

Specifies the releases in which common parts are to be maintained or whether the specific part change is to apply to all releases in which the part is common. All releases must be specified unless the **-force** flag is specified as well.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-force

Forces a break between common parts.

-from script location

Specifies the location of the file contents.

-none

Indicates that the part will never contain any data. For example, the part might be a collector object.

-relative Name

Places the specified part relative to the directory location specified according to the complete path name of the part. Directories are created if necessary when extracting or checking out in order to copy the part by its full path name. Specify either -relative or -top.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-remarks Text

Adds explanatory remarks for the action. Remarks are limited to 29,999 characters. If you need to include more than this limit, break your remarks into smaller files and load them from these files.

-retainLock

Submits the changes made to a specified part without unlocking the part.

-stdin

Loads the part from standard input.

-text

Specifies that the object is a text file. If you specify neither -binary, -none, nor -text, -text is the default.

-top Name

Specifies the leading portion of the path name that is a subset of the current working directory on the client machine. Specify either -relative or -top.

(Environment variable: TC_TOP)

-verbose

TeamConnection displays a confirmation message after you issue the command.

-workarea Name . . .

Part

The work area name.

(Environment variable: TC_WORKAREA)

Examples

Assume that your TC_RELEASE environment variable is set to the release associated with the part **graphix\x.c** and that the tracking subprocess is included in the associated release process. Your current working directory is **\janeltest**, and your TC_TOP environment variable is not set. The following command checks in the part after editing. Changes made to the part **graphix\x.c** are submitted to the TeamConnection server creating a new version of the part. The changes relate to the work areas corresponding to defect **8117** in the release indicated by the TC_RELEASE environment variable. The edited part **graphix\x.c** must exist in your current working directory.

```
teamc part -checkin graphix\x.c -workarea 8117
```

The following command checks in changes to part **file1.c**, which is common to two work areas: 400 and 428.

```
teamc part -checkin file1.c -workarea 400 428 -release R1
```

part -checkout Name . . .

Retrieves a working copy of a specified part and locks it for editing purposes. Only the most recent version of a part can be checked out.

Attributes

The part -checkout Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-force

Forces a break between common parts.

-relative Name

Places the specified part relative to the directory location specified according to the complete path name of the part. Directories are created if necessary when extracting or checking out in order to copy the part by its full path name. Specify either -relative or -top.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-stdout

Redirects the specified part to standard output when extracting it from the TeamConnection server.

-top Name

Specifies the leading portion of the path name that is a subset of the current working directory on the client machine. Specify either -relative or -top.

(Environment variable: TC_TOP)

-verbose

TeamConnection displays a confirmation message after you issue the command.

-workarea Name . . .

The work area name.

(Environment variable: TC_WORKAREA)

Examples

Assume that your current working directory is **c:\jane\graphix** and that your TC_TOP environment variable is set to **c:\jane**. The following command checks out a working copy of a part **graphix\x.c** in work area **501** and locks it for editing. In this example, the value of the TC_TOP environment variable, **\jane**, is stripped from the head of your current working directory. The result indicates the name of the part, **graphix\x.c**, within the TeamConnection environment. The part is copied to **c:\jane\graphix** with the name **x.c**, and the current version of the part in work area **501** is locked.

```
teamc part -checkout graphix\x.c -release 10graphix -workarea 501
```

The following command checks out part **file1.c** from work area **400**.

```
teamc part -checkout file1.c -workarea 400 -release R1
```

part -childInfoView Name . . .

Shows the immediate children (of the part) in the build tree.

Attributes

The part -childInfoView Name . . . command takes the following attributes:

-become Name

Part

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-driver Name

Specifies the driver for which the command is issued. Specify only one of -driver, -version, or -workarea.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-long

Displays detailed information for the specified objects.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-type Name

The type of the parts. The default is TCPart.

-verbose

TeamConnection displays a confirmation message after you issue the command.

-version Name

Specifies the version of the object. Specify only one of -driver, -version, or -workarea.

-workarea Name . . .

The name of the work area for the part. Specify only one of -driver, -version, or -workarea.

part -configInfo

Shows configurable field properties for parts in the specified family.

Attributes

The part -configInfo command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-raw

Displays configurable field information in raw format.

Examples

- The following command displays the configurable fields defined for the parts in family **rdev**.

```
teamc part -configInfo -family rdev
```

The following is an example of the output provided for this command. This example assumes that the only configurable field defined is called **partTest**.

Attribute Name	DB Column Name	Create/Required	Type
partTest	partTest	yes/no	test

- The following command displays the configurable fields defined for parts in family **rdev** in raw format.

```
teamc part -configInfo -family rdev -raw
```

The following is an example of the output provided for this command. This example assumes that the only configurable field defined is called **partTest**.

```
Part Test|Part Test|partTest|partTest|yes|no|test
```

part -connect Name . . .

Connects a part to a parent in the build tree.

Attributes

The part -connect Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-dependent

Specifies that the part is not an input to a build, but that it is needed for a build of its parent. A C language `#include` file is an example. Specify only one of the attributes -input, -output, or -dependent. If none of these is specified, the default is -input.

Part

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-input

Specifies that the part is an input that is needed to build its parent. A C language source file is an example. Specify only one of the attributes -input, -output, or -dependent. If none of these is specified, the default is -input.

-output Name

Specifies that the part is built at the same time as its parent. A .map file is an example. Specify only one of the attributes -input, -output, or -dependent. If none of these is specified, the default is -input.

-parent Name

The target of the operation.

-parenttype Name

Specifies the type of the parent part.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-type Name

The type of the parts. The default is TcPart.

-verbose

TeamConnection displays a confirmation message after you issue the command.

-workarea Name . . .

The work area name.

(Environment variable: TC_WORKAREA)

Examples

The following command connects dependent file **mymsg.h** to its parent **hello.c**.

```
teamc part -connect mymsg.h -parent hello.c -dependent -workarea 501
```

part -create Name . . .

Creates parts with the specified names; this creates a TeamConnection record for the part and copies it to the server. A part must have a unique path name and type within a release.

You can specify fields to be user-configurable using the **-configField** attribute overriding field defaults. When you create a part, TeamConnection enables you to specify whether it is a text or binary file using the **-text** and **-binary** attributes. If you omit these attributes, TeamConnection can determine from the file itself what format to create it in.

When a part is created successfully, the part mode is read-only. Setting the environment variable **tc_modperm** can change the mode.

- Set to off/OFF part mode is read-only
- Set to on/ON part mode is read-write

Attributes

The part -create Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-binary

Specifies that the object is a binary file. Specify only one of -binary, -none, or -text. If you specify none of these attributes, -text is the default.

-builder Name

Specifies the name of the builder you want to use.

-component Name

The component associated with the object. Different components can manage different versions of the same object.

(Environment variable: TC_COMPONENT.)

-configField Name

Specify a field to be user-configurable, overriding the field default. NULL is a valid value.

-dependent

Part

Specifies that the part is not an input to a build, but that it is needed for a build of its parent. A C language `#include` file is an example. Specify only one of the attributes `-input`, `-output`, or `-dependent`. If none of these is specified, the default is `-input`.

-empty

Indicates that the part being created is a place holder. The part does not contain data when you create it, but it will contain data later. Typically, you use this attribute for files that you plan to generate later using **part -build**. Specify only one of `-empty`, `-from`, or `-stdin`. You cannot use this attribute if you also use the `-none` attribute. You can use it only with the `-binary` and `-text` attributes.

-family Name

The family for which this command is being issued.

(Environment variable: `TC_FAMILY`)

-fmode Octal_number

Specifies the file mode in the TeamConnection product when creating or modifying. If no mode is specified, the current file mode is accepted.

-from script location

Specifies the location of the file contents.

-input

Specifies that the part is an input that is needed to build its parent. A C language source file is an example. Specify only one of the attributes `-input`, `-output`, or `-dependent`. If none of these is specified, the default is `-input`.

-none

Indicates that the part will never contain any data. For example, the part might be a collector object. Specify only one of `-binary`, `-none`, or `-text`. If you specify none of these attributes, `-text` is the default.

-output Name

Specifies that the part is built at the same time as its parent. A `.map` file is an example. Specify only one of the attributes `-input`, `-output`, or `-dependent`. If none of these is specified, the default is `-input`.

-parameters

Specifies the parameters you use to build a part. These parameters replace the ones in the builder that is associated with this part. If you want to include the builder parameters, you can imbed the **\$(BUILDERPARMS)** variable in your part. You can imbed this variable wherever you want the variables substituted in your part.

In UNIX environments, you need to include an escape character before the **\$**: `\$(variable_name)`. The following is an example: `\$(TC_INPUT)`.

-parent Name

The target of the operation.

-parenttype Name

Specifies the type of the parent part.

-parser Name

Specifies the name of the parser you want to use.

-relative Name

Places the specified part relative to the directory location specified according to the complete path name of the part. Directories are created if necessary when extracting or checking out in order to copy the part by its full path name. Specify either **-relative** or **-top**.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-remarks Text

Adds explanatory remarks for the action. Remarks are limited to 29,999 characters. If you need to include more than this limit, break your remarks into smaller files and load them from these files.

-stdin

Loads the part from standard input.

-temporary

Specifies that this part is not temporary. This is the default.

+temporary

Specifies that the part is temporary. A temporary part is one whose bulk contents will be deleted after a successful build.

-text

Specifies that the object is a text file. If you specify neither **-binary**, **-none**, nor **-text**, **-text** is the default.

-top Name

Specifies the leading portion of the path name that is a subset of the current working directory on the client machine. Specify either **-relative** or **-top**.

(Environment variable: TC_TOP)

Part

-verbose

TeamConnection displays a confirmation message after you issue the command.

-workarea Name . . .

The work area name.

(Environment variable: TC_WORKAREA)

Examples

Assume that your current working directory is **c:\jane**, and that your TC_TOP environment variable is set to **c:\jane**. You have a part with the path name **c:\jane\src\bar\option\tic.c** on your workstation. You also have a work area **341** to fix defect **341**.

The following command creates this part within the TeamConnection product as **src\bar\option\tic.c** and associates it with a release whose process includes the tracking subprocess. The part is managed by the **graphs** component and is created as part of the fix for defect **341**.

```
teamc part -create src\bar\option\tic.c -component graphs  
-release 32charting -workarea 341
```

part -delete Name . . .

Deletes the specified parts.

Attributes

The part -delete Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-common Name . . .

Specifies the releases in which common parts are to be maintained or whether the specific part change is to apply to all releases in which the part is common. All releases must be specified unless the **-force** flag is specified as well.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-force

Forces a break between common parts.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-top Name

Specifies the leading portion of the path name that is a subset of the current working directory on the client machine.

(Environment variable: TC_TOP)

-type Name

The type of the parts. The default is TcPart.

-verbose

TeamConnection displays a confirmation message after you issue the command.

-workarea Name . . .

The work area name.

(Environment variable: TC_WORKAREA)

part -disconnect Name . . .

Disconnects a part from its parent in the build tree.

Attributes

The part -disconnect Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-parent Name

The target of the operation.

-parenttype Name

Specifies the type of the parent part.

Part

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-type Name

The type of the parts. The default is TcPart.

-verbose

TeamConnection displays a confirmation message after you issue the command.

-workarea Name . . .

The work area name.

(Environment variable: TC_WORKAREA)

part -export Name . . .

The `part -export Name . . .` command exports the part or parts specified by the *Name* argument to the file name specified in the `-file` attribute. With this command you need to specify the class in which the part is defined using the `-type` attribute.

This function is useful for exporting information from one family into another. You can export the information from one family into a CDF file and then import the CDF file into another family.

For more information about the TeamConnection import and export functions refer to the *Tool Builder's Development Guide*. The *Tool Builder's Development Guide* also provides more information about common data format (CDF), the file format from which TeamConnection imports information and to which it exports information.

Note: You cannot use the import and export functions to migrate information from CMVC to TeamConnection or from one release of TeamConnection to another. Refer to the *Administrator's Guide* for information on the TeamConnection migration tool.

Attributes

The `part -export Name . . .` command takes the following attributes:

-file filename

Specifies the name of the file to which the part or parts are to be exported. The information is exported to the file in CDF format.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-type Name

Specifies the name of the class of the part being exported.

-version Name

Specifies the version of the object.

-viewType viewTypeName

Defines the name of the view for the part being exported. If you do not specify this attribute, it defaults to the part type.

-workarea Name . . .

The work area name.

(Environment variable: TC_WORKAREA)

Examples

part -extract Name . . .

Retrieves a copy of a specified part. The current version is extracted by default.

Attributes

The part -extract Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-dmask Octal_number

Specifies the read, write, and execute directory permissions for extracted parts in octal notation.

The default is 750 (read, write and execute access for directory owner, read and execute access for others in the owner's group, and no access for all other users).



While the OS/2 client accepts **-dmask**, it has no effect.

-driver Name

Specifies the driver for which the command is issued. Specify only one of -driver, -version, or -workarea.

Part

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-fmask Octal_number

Specifies the read, write, and execute file permissions for extracted parts in octal notation. The default is the file's mode less the write permission for the part owner, others in the owner's group and all others.

-nokeys

Indicates that you do not want to substitute assigned values in place of keywords embedded in the extracted parts.

-relative Name

Places the specified part relative to the directory location specified according to the complete path name of the part. Directories are created if necessary when extracting or checking out in order to copy the part by its full path name. Specify either -relative or -top.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-stdout

Redirects the specified part to standard output when extracting it from the TeamConnection server.

-top Name

Specifies the leading portion of the path name that is a subset of the current working directory on the client machine. Specify either -relative or -top.

(Environment variable: TC_TOP)

-verbose

TeamConnection displays a confirmation message after you issue the command.

-version Name

Specifies the version of the object. Specify only one of -driver, -version, or -workarea.

-workarea Name . . .

The name of the work area for the part. Specify only one of -driver, -version, or -workarea.

Examples

Assume that your TC_RELEASE environment variable is set to the release associated with a part you want to extract. The following command extracts the latest copy of the part committed to the release. Part **graphix\x.c** in the release specified by the TC_RELEASE environment variable is copied to the file **View_x.c** in your current working directory. If the **-stdout** flag is not specified and your TC_TOP environment variable is not set, the part is copied to your current working directory using the base name.

```
teamc part -extract graphix\x.c -stdout > View_x.c
```

part -link Name . . .

Makes common parts in the specified release or work area.

Attributes

The part -link Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-fromrelease Name

Specifies the release from which you want to create a link for a common or shared part; use it when you link specified parts.

-fromworkarea Name

Specifies the work area from which you want to create a link for a common part; use it when you link specified parts. Specify only one of -fromworkarea or -version.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-top Name

Specifies the leading portion of the path name that is a subset of the current working directory on the client machine.

(Environment variable: TC_TOP)

Part

-type Name

The type of the parts. The default is TcPart.

-verbose

TeamConnection displays a confirmation message after you issue the command.

-version Name

Specifies the source version name for the link. Specify only one of **-fromworkarea** or **-version**.

-workarea Name . . .

The work area name.

(Environment variable: TC_WORKAREA)

Examples

The following command creates a common part between two releases. The committed version of the part **debugr\x.c** in release **10debugr** is linked to the **20debugr** release. This creates a common part link between the releases **10debugr** and **20debugr** for the committed version of the part in release **10debugr**. Alternatively, you can specify that you want to link from the work area, using the **-fromworkarea** flag. Future changes to the part **debugr\x.c** must reference a work area for each release to maintain part commonality.

```
teamc part -link debugr\x.c -fromrelease 10debugr -release 20debugr  
-workarea 866
```

Assume that you are working in work area **400** and that you need the latest version of a part that is in work area **428**. The following command links the part in work area **428** to your work area, so that the part is common across the two work areas within the same release.

```
teamc part -link file1.c -from workarea 428 -workarea 400 -release R1
```

part -lock Name . . .

Locks a part in the TeamConnection server. This prevents other users from checking out the part. Only the current version of a part can be locked.

If your release uses concurrent development mode, you cannot lock parts.

When a part is locked successfully, the part mode is read-only. Setting the environment variable **tc_modperm** can change the mode.

- Set to off/OFF part mode is read-only
- Set to on/ON part mode is read-write

Attributes

The part -lock Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-force

Forces a break between common parts.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-top Name

Specifies the leading portion of the path name that is a subset of the current working directory on the client machine.

(Environment variable: TC_TOP)

-type Name

The type of the parts. The default is TcPart.

-verbose

TeamConnection displays a confirmation message after you issue the command.

-workarea Name . . .

The work area name.

(Environment variable: TC_WORKAREA)

part -modify Name . . .

Reassigns the part to another component or changes the file permission. When you reassign the part to another component, the component you specify manages access to the part. Different components can manage different versions of the same part.

The part -modify Name . . . action has two variations: The -workarea attribute is required when modifying the following attributes:

- -builder
- -parameters
- -parser

Part

- -temporary or +temporary
- -type

The -workarea attribute is not required when modifying the following attributes:

- -fmode
- -component

Because your family administrator can create new fields, the attributes for the **-modify** action might be different from those in your family. Those listed here represent the shipped default fields only. For a list of the field properties and flags in use in your family, use the **part -configInfo** command or see your family administrator. For more information on configurable fields, refer to the *Administrator's Guide*.

Attributes

The part -modify Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-builder Name

Specifies the name of the builder you want to use.

-component Name

The component associated with the object. Different components can manage different versions of the same object.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-fmode Octal_number

Specifies the file mode in the TeamConnection product when creating or modifying. If no mode is specified, the current file mode is accepted.

-parameters

Specifies the parameters you use to build a part. These parameters replace the ones in the builder that is associated with this part. If you want to include the builder parameters, you can imbed the **\$(BUILDERPARMS)** variable in your part. You can imbed this variable wherever you want the variables substituted in your part.

In UNIX environments, you need to include an escape character before the **\$**: **\\$(variable_name)**. The following is an example: **\\$(TC_INPUT)**.

-parser Name

Specifies the name of the parser you want to use.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-temporary

Specifies that this part is not temporary. This is the default.

+temporary

Specifies that the part is temporary. A temporary part is one whose bulk contents will be deleted after a successful build.

-top Name

Specifies the leading portion of the path name that is a subset of the current working directory on the client machine.

(Environment variable: TC_TOP)

-type Name

The type of the parts. The default is TcPart.

-verbose

TeamConnection displays a confirmation message after you issue the command.

-workarea Name . . .

The work area name.

(Environment variable: TC_WORKAREA)

Examples

Assume that your TC_RELEASE environment variable is set to the release associated with the part **src\bar\option\tic.c**. The following command reassigns that part to another component.

```
teamc part -modify src\bar\option\tic.c -component debugr
```

part -recreate Name . . .

Re-creates previously deleted parts.

Part

Attributes

The part `-recreate Name . . .` command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-common Name . . .

Specifies the releases in which common parts are to be maintained or whether the specific part change is to apply to all releases in which the part is common. All releases must be specified unless the **-force** flag is specified as well.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-force

Forces a break between common parts.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-top Name

Specifies the leading portion of the path name that is a subset of the current working directory on the client machine.

(Environment variable: TC_TOP)

-type Name

The type of the parts. The default is TcPart.

-verbose

TeamConnection displays a confirmation message after you issue the command.

-workarea Name . . .

The work area name.

(Environment variable: TC_WORKAREA)

part -refresh Name . . .

Use the part -refresh Name . . . action to refresh a part (the target part) with the contents of the same part from another release or another version of the part (the source part). The behavior of this command varies according to the relationship between the target and source parts:

- If the source part is a successor of the target part, then the target part is updated from the source part. The action performed is a part -link Name . . . and the two parts become shared or common parts.
- If the source part is a predecessor of the target part, no change is made to the target part.
- If the source and target parts are alternate versions, then TeamConnection generates collision records.

Attributes

The part -refresh Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-fromrelease Name

Specifies the release from which you want to create a link for a common or shared part; use it when you link specified parts.

-fromworkarea Name

Specifies the work area from which you want to create a link for a common part; use it when you link specified parts. Specify only one of -fromworkarea or -version.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-top Name

Specifies the leading portion of the path name that is a subset of the current working directory on the client machine.

(Environment variable: TC_TOP)

Part

-type Name

The type of the parts. The default is TCPart.

-verbose

TeamConnection displays a confirmation message after you issue the command.

-version Name

Specifies the source version name for the link. Specify only one of **-fromworkarea** or **-version**.

-workarea Name . . .

The work area name.

(Environment variable: TC_WORKAREA)

Examples

The following command refreshes a part called **debugr\x.c** in release **20debugr** with the same part in release **10debugr**. This creates a common part link between the releases **10debugr** and **20debugr** for the committed version of the part in release **10debugr**. Alternatively, you can specify that you want to refresh the part from the work area, using the **-fromworkarea** flag.

```
teamc part -refresh debugr\x.c -fromrelease 10debugr -release 20debugr  
-workarea 866
```

part -rename Name

Specifies a new path name for a part. To select the part to be renamed, replace *Name* with the name of the part. Use the **-path** attribute to specify the new name. You can modify both the path name and the base name using the **-path** attribute.

Attributes

The part -rename Name command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-common Name . . .

Specifies the releases in which common parts are to be maintained or whether the specific part change is to apply to all releases in which the part is common. All releases must be specified unless the **-force** flag is specified as well.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-force

Forces a break between common parts.

-path Name

The path name of the part. Part names, consisting of the base name and the path name, must be unique within a release.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-top Name

Specifies the leading portion of the path name that is a subset of the current working directory on the client machine.

(Environment variable: TC_TOP)

-type Name

The type of the parts. The default is TcPart.

-verbose

TeamConnection displays a confirmation message after you issue the command.

-workarea Name . . .

The work area name.

(Environment variable: TC_WORKAREA)

Examples

The following command renames an existing part in a release whose process includes the tracking subprocess. The part **debugr\src\xyz.c** in release **20debugr** and work area **560** is renamed **debugr\v2\xyz2.c**.

```
teamc part -rename debugr\src\xyz.c -path debugr\v2\xyz2.c
-release 20debugr -workarea 560
```

part -resolve Name . . .

Displays the full path name in a specific release for specified part base names. Only works on parts in an integrated work area or committed driver.

Part

Attributes

The part -resolve Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-quiet

Suppresses explanatory remarks and new line characters in the output.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-top Name

Specifies the leading portion of the path name that is a subset of the current working directory on the client machine.

(Environment variable: TC_TOP)

-type Name

The type of the parts. The default is TCPart.

-verbose

TeamConnection displays a confirmation message after you issue the command.

part -touch Name . . .

Marks the part out-of-date, to ensure it participates in the next build. This command does not affect the part's status with respect to the release, so the part does not have to be integrated into the release again.

Attributes

The part -touch Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-top Name

Specifies the leading portion of the path name that is a subset of the current working directory on the client machine.

(Environment variable: TC_TOP)

-type Name

The type of the parts. The default is TcPart.

-verbose

TeamConnection displays a confirmation message after you issue the command.

-workarea Name . . .

The work area name.

(Environment variable: TC_WORKAREA)

part -undo Name . . .

Undoes the most recent uncommitted action that changed specified parts. Reverts part back to the last frozen version within a specified work area. If the tracking subprocess is turned on, the work area must be in the fix state. If the latest version of the part is frozen, the command fails and TeamConnection returns an error message.

This action is not available for build outputs.

Attributes

The part -undo Name . . . command takes the following attributes:

-become Name

Part

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-common Name . . .

Specifies the releases in which common parts are to be maintained or whether the specific part change is to apply to all releases in which the part is common. All releases must be specified unless the **-force** flag is specified as well.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-force

Forces a break between common parts.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-top Name

Specifies the leading portion of the path name that is a subset of the current working directory on the client machine.

(Environment variable: TC_TOP)

-type Name

The type of the parts. The default is TcPart.

-verbose

TeamConnection displays a confirmation message after you issue the command.

-workarea Name . . .

The work area name.

(Environment variable: TC_WORKAREA)

Examples

The following command reverts to a previous version of a part in a work area. The most recent change submitted for part **debugr\l.c** in the release indicated by the TC_RELEASE environment variable is reversed or undone.

```
teamc part -undo debugr\l.c -workarea 501
```

part -unlock Name . . .

Unlocks a part that is checked out so that it is no longer reserved for editing purposes. Or, unlocks a part that has been previously locked using **-lock**.

Attributes

The part -unlock Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-relative Name

Places the specified part relative to the directory location specified according to the complete path name of the part. Directories are created if necessary when extracting or checking out in order to copy the part by its full path name. Specify either -relative or -top.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-top Name

Specifies the leading portion of the path name that is a subset of the current working directory on the client machine. Specify either -relative or -top.

(Environment variable: TC_TOP)

-type Name

The type of the parts. The default is TcPart.

-verbose

TeamConnection displays a confirmation message after you issue the command.

-workarea Name . . .

The work area name.

(Environment variable: TC_WORKAREA)

Part

Examples

The following command unlocks a part that was checked out in work area **501**.

```
teamc part -unlock debugr\x.c -release 11debugr -workarea 501
```

part -view Name . . .

Shows all information for the specified parts.

Attributes

The part -view Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-driver Name

Specifies the driver for which the command is issued. Specify only one of -driver, -version, or -workarea.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-long

Displays detailed information for a part, including the part history, whether the file is checked out for editing, all associated common parts, and any change information. The part change information includes the existing active changes for the part, the defect or feature, and the part version associated with those changes. The part change information also shows all versions of the part requested in the command.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-top Name

Specifies the leading portion of the path name that is a subset of the current working directory on the client machine.

(Environment variable: TC_TOP)

-type Name

The type of the parts. The default is TcPart.

-verbose

TeamConnection displays a confirmation message after you issue the command.

-version Name

Specifies the version of the object. Specify only one of -driver, -version, or -workarea.

-workarea Name . . .

The name of the work area for the part. Specify only one of -driver, -version, or -workarea.

Examples

The following command displays information about a specified part.

```
teamc part -view graphix\x.c -release 10graphix -workarea 501
```

The following command displays additional information about a specified part, including the part history, whether the part is locked for editing, all common parts, and change information.

```
teamc part -view graphix\x.c -release 10graphix -workarea 501 -long
```

part -viewmsg Name . . .

Shows the result of the latest build of the part.

Attributes

The part -viewmsg Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-driver Name

Specifies the driver for which the command is issued. Specify only one of -driver, -version, or -workarea.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

Part

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-top Name

Specifies the leading portion of the path name that is a subset of the current working directory on the client machine.

(Environment variable: TC_TOP)

-type Name

The type of the parts. The default is TCPart.

-verbose

TeamConnection displays a confirmation message after you issue the command.

-version Name

Specifies the version of the object. Specify only one of -driver, -version, or -workarea.

-workarea Name . . .

The name of the work area for the part. Specify only one of -driver, -version, or -workarea.

Related information

See the following related commands:

- Builder
- Collision
- Component
- Driver
- Parser
- Part
- Release
- Report
- Workarea

For a list of supported keywords, refer to the *TeamConnection User's Guide*.

Chapter 19. Prereq

Use the **prereq** command to create and delete prerequisite relationships between two or more work areas that are in the fix or integrate state. A work area that you identify as a prerequisite for another work area can be built and committed prior to or at the same time as the work area at hand. Work areas defined implicitly as prerequisites by the TeamConnection product must also be built and committed prior to or at the same time as the work area at hand.

Identify prerequisite relationships between work areas to indicate that work being done in one or more work areas is dependent on changes to parts associated with changes in another work area. This action ensures that a driver that includes the work area with the prerequisite cannot be committed unless its prerequisite work areas is included in the driver.

Command summary

In the following syntax statements, the first letter of each argument is capitalized, for example `-family Name`. Replace these arguments with a value that is valid for your TeamConnection database. If you want to issue a command for the family named **testfam**, for example, replace `-family Name` with `—family testfam`.

The syntax statements for the **prereq** command are:

```
teamc prereq -create Name ... -workarea Name ... -release Name ... -family  
[-become Name] [-verbose]
```

```
teamc prereq -delete Name ... -workarea Name ... -release Name ... -family  
[-become Name] [-verbose]
```

prereq -create Name . . .

Creates a prerequisite relationship between the specified work areas. The work area that you specify as an argument to the `-create` action is the prerequisite. The work area that you specify as an argument to the `-workarea` attribute is the one for which you are creating the prerequisite.

Attributes

The `prereq -create Name . . .` command takes the following attributes.

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

Prereq

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-verbose

TeamConnection displays a confirmation message after you issue the command.

-workarea Name . . .

The name of the work area for which you are creating or deleting a prerequisite.

prereq -delete Name . . .

Deletes the specified work areas from an existing group of corequisite work areas. The work area that you specify as an argument to the **-delete** action is the prerequisite. The work area that you specify as an argument to the **-workarea** attribute is the one for which you are deleting the prerequisite.

Attributes

The prereq -delete Name . . . command takes the following attributes.

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-verbose

TeamConnection displays a confirmation message after you issue the command.

-workarea Name . . .

The name of the work area for which you are creating or deleting a prerequisite.

Related information

See the following related commands:

- Coreq
- Report
- Workarea

For more information about corequisite and prerequisite relationships, refer to the *TeamConnection User's Guide*.

Chapter 20. Release

Use the **release** command to:

- Create, modify, delete, and re-create releases
- Extract the set of parts associated with a release
- Link parts within releases with those in other releases
- View information about existing releases

A release is a set of parts that must be built, tested, and distributed as a whole. A release must be created in relation to a component that manages access and notification for the release. If you create a release, you become its owner, and you have implicit authority to define an approval list and an environment list for that release. Release names must be unique within a family.

When you create a release, you must choose a process using the **-process** flag. A process contains different combinations of TeamConnection subprocesses. TeamConnection subprocesses determine the states of the work areas within a release. For release processes, you can specify the track, approval, fix, driver, and test subprocesses. Processes are configured by your family administrator, who can modify current processes and define new ones. For a list of the valid release processes and the TeamConnection subprocesses they include, use the **report -view cfgrelproc** command.

Also, when you create a release, you must choose between serial and concurrent development. Once the release is set to concurrent, it can not be changed back to serial. Choose concurrent development by using the **-concurrent** flag on the **release -create** command. (For more information about concurrent development, refer to page 41. Otherwise, the release defaults to serial development.

You can change the process for an existing release by using the **-modify** flag. For more information on how TeamConnection subprocesses relate to the states of TeamConnection objects, refer to the *TeamConnection User's Guide*.

To modify an environment or a tester for a release, use the **environment** command. See page "Chapter 12. Environment" on page 95. for more information.

To modify an approver for a release, use the **approver** command to add or delete an approver. See page "Chapter 4. Approver" on page 29. for more information.

You cannot delete releases that have parts, outstanding work areas, uncommitted drivers, or active sizing records associated with them. You cannot reuse the name of a deleted release, but you can re-create a deleted release and modify the name of a re-created release.

If autopruning is turned on for a release, you can prune committed branches (work areas or drivers) from that release. The TeamConnection server automatically prunes branches when you specify the **+autopruning** flag either on the **release -create** command or the **release -modify** command. You can also directly prune a committed branch by typing the **release -prune** command.

Note: A pruned branch cannot be recovered.

You can limit the number of committed output versions that you want to keep using the **-outputVersions** attribute on the **release -create** or **release -modify**

Release

commands. (The default for **release -create** is to keep all output versions.) When you commit your work area, all output versions that exceed the specified limit will change to type empty. For an example, refer to page “Examples” on page 191.

To keep all output versions on the release, specify the **+outputVersions** attribute on the **release -modify** command. .*

When you link parts in one release to those in another, you can link the current or the committed version of each active part. The committed version is the default setting. If you want to specify the current version of each active part, you must also supply the name of the associated work area using the **-fromworkarea** parameter.

When you extract parts associated with a release, the current version of the associated parts is extracted by default. Alternatively, you can extract parts changed after a certain date. For a release whose process includes the track and driver subprocesses, you can extract the last committed version of the parts.

Command Summary

In the following syntax statements, the first letter of each argument is capitalized, for example `-family Name`. Replace these arguments with a value that is valid for your TeamConnection database. If you want to issue a command for the family named **testfam**, for example, replace `-family Name` with `—family testfam`.

The syntax statements for the **release** command are:

```
teamc release -create Name ... -component Name -process Name
               -family Name
               [-environment Name -tester Name]* [-approver Name]*
               [-description Text] [-owner Name] [-become Name]
               [-db DBPath_and_Name] [-concurrent] [-verbose]
               [+autopruning] [-outputVersions Number]
```

```
teamc release -delete Name ... -family Name [-become Name] [-verbose]
```

```
teamc release -export Name ... -file Name -family Name [-become Name]
               [-verbose]
```

```
teamc release -extract Name ... -root Name -family Name
               [-nokeys] [-fmask Octal_number] [-version Name]
               [-dmask Octal_number] [-crlf | -nocrlf] [-component]
               [-become Name] [-verbose]
```

```
teamc release -link Name ... -workarea Name -family Name
               -fromrelease Name [-fromworkarea Name | -version Name]
               [-become Name] [-verbose]
```

```
teamc release -modify Name ... -family Name
               { -component Name -process Name [-environment Name -tester Name]*
               [-approver Name]* -description Text -owner Name
               [+autopruning | -autopruning]
               [-outputVersions Number | +outputVersions]}
               [-become Name] [-verbose]
```

```
teamc release -prune Name -version Name -family Name
[-become Name] [-verbose]
```

```
teamc release -recreate Name ... -family Name
[-environment Name -tester Name]* [-approver Name]*
[-become Name] [-verbose]
```

```
teamc release -view Name ... -family Name [-processInfo]
[-become Name] [-verbose]
```

Notes:

- * Required only when their related subprocess has been specified for the release

release -create Name . . .

The release -create Name . . . action creates releases with the specified names. Certain attributes of this action are used only when certain release subprocesses are in effect:

- **-approver** is used by the approval subprocess.
- **-environment** is used by the test subprocess.
- **-tester** is used by the test subprocess.

Attributes

The release -create Name . . . command takes the following attributes:

-approver Name

The user ID of the approver.

-approver Name

The user ID of the approver for the release. This attribute is used only when the approval subprocess is in effect.

+autopruning

Turns autopruning on for the release. Specify only one of +autopruning or -autopruning.

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-component Name

The component associated with the object. Different components can manage different versions of the same object.

Release

(Environment variable: TC_COMPONENT.)

-concurrent

Turns on concurrent development mode for the release.

-db DBPathAndName

The name of the remote database for the release.

-description Text

Specifies a description of the object.

-environment environment

Specifies the environment in which the testing is done.

Note: The tester/environment name combination becomes an entry on the environment list for the release.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-outputVersions

Saves a specified number of committed outputs. Specify only one of +outputVersions or -outputVersions.

-owner Name

Specifies the user ID of the owner of the object.

-process Name

Specifies a process when creating or modifying a release. Processes for your environment are configured by your family administrator. For a list of the valid release processes and the TeamConnection subprocesses they include, use the **report -view cfgrelproc** command.

-tester Name

Specifies the user responsible for testing in the given environment if the test subprocess is included in the release process. The tester/environment name combination becomes an entry on the environment list for the release.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

The following command creates a release named **10debugr** which is associated with component **comp1** and the process **preship** (which specifies the TeamConnection track, approval, fix, driver, and test subprocesses). Because the **preship** process includes the TeamConnection test subprocess, an environment, **PCVersion1**, and an initial tester, **jon**, are specified. And because the **preship** process includes the TeamConnection approval subprocess, an approver, **michael**, is specified. The release will use serial development.

```
teamc release -create 10debugr -component comp1 -process preship
-environment PCVersion1 -tester jon -approver michael
```

The following command creates a concurrent release named **20debugr** which is associated with component **comp1** and the process **no_track**.

```
teamc release -create 20debugr -process no_track -concurrent
-component comp1
```

release -delete Name . . .

Deletes the specified releases. Releases cannot have parts, outstanding work areas, uncommitted drivers, or active sizing records associated with them.

Attributes

The release -delete Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-verbose

TeamConnection displays a confirmation message after you issue the command.

release -export Name . . .

The release -export Name . . . command exports the parts in the release (or releases) specified by the *Name* argument to the file name specified in the -file attribute.

This function is useful for exporting information from one family into another. You can export the information from one family into a CDF file and then import the CDF file into another family.

Release

For more information about the TeamConnection import and export functions refer to the *Tool Builder's Development Guide*. The *Tool Builder's Development Guide* also provides more information about common data format (CDF), the file format from which TeamConnection imports information and to which it exports information.

Note: You cannot use the import and export functions to migrate information from CMVC to TeamConnection or from one release of TeamConnection to another. Refer to the *Administrator's Guide* for information on the TeamConnection migration tool.

Attributes

The release -export Name . . . command takes the following attributes:

-file filename

Specifies the name of the file to which the part or parts are to be exported. The information is exported to the file in CDF format.

-verbose

TeamConnection displays a confirmation message after you issue the command.

release -extract Name . . .

Extracts the part tree for the specified releases. By default, the current versions of all parts in the releases are extracted.

You can issue this command with the -component attribute to limit the release extract to parts in one or more components.

Attributes

The release -extract Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-component Name

The components whose parts you want to extract. This attribute restricts the extract action to the components you include with the attribute.

-crlf

Provides transparent file conversion between UNIX- and Intel-based operating systems. This attribute enables parts shared between UNIX and Intel platforms to have the proper format for the platform to which they are extracted. When parts are extracted to an Intel platform, the -crlf attribute will add carriage-returns, expand tabs, and add end-of-file characters (if the parts do not already have EOF

characters). When parts are extracted to a UNIX platform, the `-crlf` attribute will remove carriage-returns, replace spaces with tabs, and remove end-of-file characters.

If you omit this attribute, no file format conversion is performed.

-dmask Octal_number

Specifies the read, write, and execute directory permissions for extracted parts in octal notation.

The default is 750 (read, write and execute access for directory owner, read and execute access for others in the owner's group, and no access for all other users).



While the OS/2 client accepts **-dmask**, it has no effect.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-fmask Octal_number

Specifies the read, write, and execute file permissions for extracted parts in octal notation. The default is the file's mode less the write permission for the part owner, others in the owner's group and all others.

-nokeys

Indicates that you do not want to substitute assigned values in place of keywords embedded in the extracted parts.

-root Name

Specifies a directory on the client where the extracted part tree is to be placed.

Note: You can mount a directory from another machine to the client machine, so that the client machine will treat that directory as a local directory.

-verbose

TeamConnection displays a confirmation message after you issue the command.

-version Name

Specifies the version of the object.

Release

Examples

You own the **21graphix** release. The following command extracts all of the parts associated with that release and writes them to the **\tmp\test\graphix** directory. A part tree is created relative to the location **\tmp\test\graphix**. This part tree represents all parts associated with the **21graphix** release that have changed.

```
teamc release -extract 21graphix -root \tmp\test\graphix
```

Assume that the directory **\tmp\9604** has been exported on a host with write permission given to the TeamConnection client and that the directory is mounted on the client using NFS, Netware, IBM LAN Server, or another LAN product. The following command extracts all of the parts associated with release **9604** and components **cmdref** and **using** and writes them to the **\tmp\9604** directory. A part tree is created relative to the location **\tmp\9604**. This part tree represents all parts associated with the **9604** release and the **cmdref** and **using** components that have changed.

```
teamc release -extract 9604 -component cmdref using -root \tmp\9604
```

release -link Name . . .

Links the active parts in a specified release to those in another specified release. If the **-committed** or **-date** flag is not supplied, the current version of each part is linked by default.

Attributes

The release -link Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-fromrelease Name

Specifies the release from which you want to create a link for a common or shared part; use it when you link specified parts.

-fromworkarea Name

Specifies the work area from which you want to create a link for a common part; use it when you link specified parts. Specify only one of -fromworkarea or -version.

-verbose

TeamConnection displays a confirmation message after you issue the command.

-version Name

Specifies the source version name for the link. Specify only one of `-fromworkarea` or `-version`.

-workarea Name . . .

The work area name.

(Environment variable: TC_WORKAREA)

Examples

The following command links the committed version of release **10debugr** to release **20debugr**.

```
teamc release -link 20debugr -fromrelease 10debugr -workarea 12
```

release -modify Name . . .

Modifies the following properties of the specified releases. One or more of the following attributes is required.

- approver
- autopruning or +autopruning
- component
- description
- environment
- outputVersions or +outputVersions
- owner
- process
- tester

Attributes

The release `-modify Name . . .` command takes the following attributes:

-approver Name

The user ID of the approver.

-approver Name

The user ID of the approver for the release. This attribute is used only when the approval subprocess is in effect.

+autopruning

Turns autopruning on for the release. Specify only one of `+autopruning` or `-autopruning`.

-autopruning

Release

Turns autopruning off for the release. Specify only one of +autopruning or -autopruning.

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-component Name

The component associated with the object. Different components can manage different versions of the same object.

(Environment variable: TC_COMPONENT.)

-component Name

The component associated with the object. Different components can manage different versions of the same object.

-description Text

Specifies a description of the object.

-environment environment

Specifies the environment in which the testing is done.

Note: The tester/environment name combination becomes an entry on the environment list for the release.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

+outputVersions

Saves all committed outputs. Specify only one of +outputVersions or -outputVersions.

-outputVersions

Saves a specified number of committed outputs. Specify only one of +outputVersions or -outputVersions.

-owner Name

Specifies the user ID of the owner of the object.

-process Name

Specifies a process when creating or modifying a release. Processes for your environment are configured by your family administrator. For a list of the valid release processes and the TeamConnection subprocesses they include, use the **report -view cfgrelproc** command.

-tester Name

Specifies the user responsible for testing in the given environment if the test subprocess is included in the release process. The tester/environment name combination becomes an entry on the environment list for the release.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

The following command turns on autopruning and limits the number of committed output versions to **10** for release **10debugr**.

```
teamc release -modify 10debugr +autopruning -outputVersions 10
```

You own release **10debugr**. The following command makes **pam** the new owner.

```
teamc release -modify 10debugr -owner pam
```

You own release **10debugr**. The following command changes the process associated with the release to **prototype**.

```
teamc release -modify 10debugr -process prototype
```

release -prune Name

Prunes the specified branch (work area or driver) for the release.

Attributes

The release -prune Name command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-verbose

TeamConnection displays a confirmation message after you issue the command.

-version Name

Release

Specifies the name of the first version of the branch being pruned. Also the source version name for the release link.

release -recreate Name . . .

Re-creates previously deleted releases. Certain attributes of this action are used only when certain release subprocesses are in effect:

- **-approver** is used by the approval subprocess.
- **-environment** is used by the test subprocess.
- **-tester** is used by the test subprocess.

Attributes

The release -recreate Name . . . command takes the following attributes:

-approver Name

The user ID of the approver.

-approver Name

The user ID of the approver for the release. This attribute is used only when the approval subprocess is in effect.

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-environment environment

Specifies the environment in which the testing is done.

Note: The tester/environment name combination becomes an entry on the environment list for the release.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-tester Name

Specifies the user responsible for testing in the given environment if the test subprocess is included in the release process. The tester/environment name combination becomes an entry on the environment list for the release.

-verbose

TeamConnection displays a confirmation message after you issue the command.

release -view Name . . .

Shows all current information for the specified releases.

Attributes

The release -view Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-processInfo

Displays the current process setting and associated TeamConnection subprocesses for the specified components when used with the **-view** action flag.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Related information

See the following related commands:

- Approver
- Builder
- Collision
- Component
- Environment
- Parser
- Part
- Workarea

For a list of supported keywords, refer to the *TeamConnection User's Guide*.

For a list of the valid release processes and the TeamConnection subprocesses they include, use the **report -view cfgrelproc** command.

Chapter 21. Report

Use the **report** command to query the tables and views associated with the TeamConnection product and generate output showing the results of that query. TeamConnection uses the information provided in a **report** command to build a SELECT statement. The **-view** flag specifies the database table or view to query, and the **-where** flag specifies the selection criteria for the query.

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 the user is not authorized to access.

Use the **-verbose** flag to have TeamConnection display message 0010-213 when all or some objects requested are not included in a report.

The **-help** flag displays a list of valid table and view names that you can use as arguments for the **-view** flag. All view and table names as well as their corresponding fields are listed in "Appendix. Querying the TeamConnection database" on page 261. the *TeamConnection Commands Reference*.

Views are also available to report all inherited notification list members for a specified component. These views are designated by the suffix UpView. Upnviews are valid for **defect**, **feature**, **access**, and **notify**. You must specify a component in the selection criteria of the **report** command to query this view.

Views are available to report all objects of a certain type for all descendants of a specified component; these views are designated by the suffix DownView. Downviews are valid for **defect**, **feature**, **access**, and **notify**. You must specify a component in the selection criteria of the **report** command to query this view.

The *Text* argument of the **-where** attribute flag defines the search criteria and the conditions of the data you want to select, and it must follow the TeamConnection database syntax rules. It can include subselects and valid functions. For a discussion of syntax and its use, refer to "Appendix. Querying the TeamConnection database" on page 261 of the *Commands Reference*.

By default, report results are displayed in 132-column table format, but you can request the output to be displayed in 80-column stanza format or in long format, which is a combination of stanza and table formats. You can also request output in raw format if you want the results to be used in another program or utility.

Command summary

In the following syntax statements, the first letter of each argument is capitalized, for example `-family Name`. Replace these arguments with a value that is valid for your TeamConnection database. If you want to issue a command for the family named **testfam**, for example, replace `-family Name` with `—family testfam`.

The syntax statements for the **report** command are:

Report

```
teamc report -help -family Name [-become Name] [-verbose]

teamc report -testClient -family Name [-become Name] [-verbose]

teamc report -testServer -family Name [-become Name] [-verbose]

teamc report -userExitInfo -family Name [-long] [-become Name]

teamc report -view name -family Name [-where Text] [-become Name]
    [-stanza | -raw | -table* | -long] [-verbose]

teamc report -view partView -family Name [-where Text] [-become Name]
    {-workarea Name | -version Name}
    -release Name
    [-parent Name] [-parenttype Name**]
    [-stanza | -raw | -table* | -long] [-verbose]

* Default
** Default parenttype if not specified is file.
```

report -help

Displays a list of the valid view and table names you can use as arguments for the **-view** flag.

Attributes

The report -help command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-verbose

TeamConnection displays a confirmation message after you issue the command.

report -testClient

Tests the availability of the TeamConnection message catalog on the client's host, and returns a message informing the user of its availability.

Attributes

The report `-testClient` command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-verbose

TeamConnection displays a confirmation message after you issue the command.

report -testServer

Tests the availability of the TeamConnection message catalog on the TeamConnection server, and returns a message informing the user of its availability.

The report returned provides the names and values of certain environment variables that are set on the TeamConnection server. The report is a file that can be updated by the administrator and used by the TeamConnection server without restarting the server.

Attributes

The report `-testServer` command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-verbose

TeamConnection displays a confirmation message after you issue the command.

report -userExitInfo

The report -userExitInfo command displays user exit parameters for all TeamConnection commands that support user exits. The report produced by this command includes the following information:

- The parameters that can be passed to a user exit program at each user exit ID:

Exit ID 0

Before checking the command request

Exit ID 1

Before processing the command

Exit ID 2

After the action completes

Exit ID 3

If the action or the user exit program fails

- Any configurable fields that can be passed to the user exit program.

Attributes

The report -userExitInfo command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-long

Displays detailed information for the specified objects.

Examples

- The following command displays user exit information in the default format:
teamc report -userExitInfo

The following is an excerpt of the report produced by this command:

```

                                User Exit Information
-----
Action Name: AccessCreate
User Exits 0 and 3:
  Parameter List:      TeamcUserID, component, authority, effectiveUserID,
                        TeamcUserID, verboseFlag
User Exit 1:
  Parameter List:      TeamcUserID, component, authority, effectiveUserID,
                        verboseFlag
User Exit 2:
  Parameter List:      TeamcUserID, component, authority, effectiveUserID,
```

verboseFlag
No configurable fields.

Action Name: AccessDelete

User Exits 0 and 3:

Parameter List: TeamcUserID, component, authority, effectiveUserID,
TeamcUserID, verboseFlag

User Exit 1:

Parameter List: TeamcUserID, component, authority, effectiveUserID,
verboseFlag

User Exit 2:

Parameter List: TeamcUserID, component, authority, effectiveUserID,
verboseFlag

No configurable fields.

- Issuing the report `-userExitInfo` command with the `-long` attribute displays information about your user exit programs as well as the user exit parameters. User exit information is shown at the end of the report. The following is an example of the report `-userExitInfo` command with the `-long` attribute:

```
teamc report -userExitInfo -long
```

The following is an example of the information added to the report by the `-long` attribute:

Configured User Exits:

Action Name: DefectModify

User Exit 0; Program: viewexit.cmd "mod defect 0"

Parameter List: remarks

Configurable Fields: phaseFound

User Exit 2; Program: viewexit.cmd "DefMod2"

No parameter list fields.

No configurable fields.

Action Name: PartAdd

User Exit 0; Program: viewexit.cmd "PartAdd 0"

Parameter List: release

No configurable fields.

User Exit 1; Program: viewexit.exe "PartAdd1"

No parameter list fields.

No configurable fields.

report -view Name

Specifies the database table or view you want to query. You can use a unique prefix abbreviation of the table and view names. You can view the following names:

AccessDownView

AccessView

ApprovalView

ApproverView

Authority

BuilderView

Cfgcomproc

Cfgrelproc

ChangeView

CollisionView

CompView (long)

Config

CoreqView

Report

DefectDownView (long)
DefectView (long)
DriverMemberView
DriverView (long)
EnvView
FeatureDownView (long)
FeatureView (long)
FixView
HostView
Interest
NoteView
NotifyDownView
NotifyUpView
NotifyView
ParserView
PartFullView
PartsOutView
PartView
PrereqView
ReleaseView
SizeView
TestView
Users (long)
VerifyView
VersionView (long)
WorkAreaView (long)

Attributes

The report -view Name command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-long

Produces report output in stanza format, with additional important information shown in a table format:

- Each database record is a stanza.

- Each stanza line consists of a field and its corresponding values.

Specify only one of -long, -raw, -stanza, or -table.

-raw

Displays reports in raw format. Specify only one of -long, -raw, -stanza, or -table.

See “Appendix. Querying the TeamConnection database” on page 261 Refer to the appendix in the *Commands Reference* for the format of raw output for each report view.

-stanza

Produces report output in stanza format:

- Each database record is a stanza.
- Each stanza line consists of a field and its corresponding values.

Specify only one of -long, -raw, -stanza, or -table.

-table

Produces report output in the following table format:

- Each field is displayed as a column heading.
- Field values appear under respective column headings.
- Each row corresponds to one database record.

Specify only one of -long, -raw, -stanza, or -table. The table view is the default format of report output.

-verbose

TeamConnection displays a confirmation message after you issue the command.

-where Text

Defines the selection criteria to query the specified table or view using valid syntax.

Examples

The following command displays all users who have **developer** authority for the **graphix** component.

```
teamc report -view accessView -where "compname = 'graphix' and
authorityName='developer'"
```

The preceding command could be abbreviated as follows. This command shows access explicitly defined for the **graphix** component. Additional access can be inherited at the component driver.

```
teamc report -vi accessv -wh "compname = 'graphix'
and authorityName='developer'"
```

The following command displays all the approval records for the **20graphix** release that were updated on or after December 1, 1995. Because date fields include the date of the action as well as the time of the action, the approval records selected

Report

using the above example are those that were updated after 12:00 p.m. on November 30, 1995. The date field must be enclosed in single quotation marks because it is a character field.

```
teamc report -view approvalview -where "releasename='20graphix' and  
lastupdate > '1995/12/01'"
```

The following command displays all members of the approver list for any of the debugger releases. The percent sign (%) is a wildcard character used with the **like** operator to match zero or more characters. For a more granular search, use the underscore (_) wildcard character to match a single character.

```
teamc report -view approverView -where "releasename like '%debug%'"
```

The following command displays all authority groups that include the **DriverCommit** action.

```
teamc report -view authority -where "action = 'DriverCommit'"
```

The following command displays all actions that are included in the definition of the **general** authority group.

```
teamc report -view authority -where "name = 'general'"
```

The following command displays all drivers for the **20debugr** release that have been updated on or after April 29, 1995. Because date fields include date and time, the drivers selected using the above example are those that were updated after 12:00 p.m. on April 28, 1995. The date field must be enclosed in single quotation marks because it is a character field.

```
teamc report -view driverView -where "releaseName='20debugr' and  
lastUpdate > '1995/04/29'"
```

The following command displays, in raw format, all drivers that were committed earlier than March 3, 1995. The drivers committed on or before 12:00 p.m. on March 3, 1995, are selected. The date field must be enclosed in single quotation marks because it is a character field.

```
teamc report -view driverView -where "commitDate < '1995/03/03'" -raw
```

The following command displays all returned defects originated by the user ID **michael**.

```
teamc report -view defectView -where "originLogin = 'michael' and state =  
'returned'"
```

The following command displays all defects for the **graphix** component that are in the working state.

```
teamc report -view defectView -where "state = 'working' and compName =  
'graphix'"
```

The following command displays all defects in the open or working state that are owned by users in the area **e50**. If some user areas are **E50**, they are not selected.

```
teamc report -view defectView -where "state in ('open','working') and  
ownerArea='e50'"
```

The following command displays all release environment list entries that designate the user **michael** as the tester of the **PCVersion1** environment.

```
teamc report -view envView -where "userlogin = 'michael'  
and name = 'PCVersion1'"
```

The following command displays the release environment list members for the **21debugr** release.

```
teamc report -view envView -where "releaseName = '21debugr'"
```

The following command displays member parts of the **10debugr** release that were last updated on or after August 8, 1995. The date field must be in the yyyy/mm/dd format, and it must be enclosed in single quotation marks because it is a character field. Because the date field includes date and time, all parts updated after August 7, 1995, at 12:00 p.m. are selected.

```
teamc report -view partFullView -where "releaseName = '10debugr'
and lastUpdate > '1995/08/07'"
```

The following command displays member parts of the **20graphix** release that are currently checked out.

```
teamc report -view partsOutView -where "releaseName = '20graphix'"
```

The following command displays all actions that define the **developer** interest group.

```
teamc report -view interest -where "Name = 'developer'"
```

The following command displays all the notes for defect **7627** that were added before September 2, 1995. The date field must be in the yyyy/mm/dd format, and it must be enclosed in single quotation marks because it is a character field.

```
teamc report -view noteView -where "defectName = '7627' and addDate
< '1995/09/02'"
```

The following command displays all the notes for defect **4866** written by the owners of the user IDs **sam** and **sara**.

```
teamc report -view noteView -where "defectName = '4866' and
(userlogin = 'sam' or userlogin = 'sara')"
```

The following command displays all test records for the environment **PCVersion2** that have **reject** or **abstain** test results recorded.

```
teamc report -view testView -where "envName = 'PCVersion2' and (state
= 'reject' or state = 'abstain')"
```

The following command displays all test records for the defect **9821** that have an environment name beginning with **PCV**.

```
teamc report -view testView -where "envName like 'PCV%' and
defectName = '9821'"
```

The following command displays all existing work areas for defect **5490** that are in the fix state.

```
teamc report -view workAreaView -where "defectName = '5490' and
state = 'fix'"
```

The following command displays all existing work areas created on or after September 17, 1995, for the **21debugr** release.

```
teamc report -view workAreaView -where "releasename = '21debugr' and
addDate > '1995/09/16'"
```

The following command displays all users in areas that include **tools** as part of the area name.

```
teamc report -view users -where "area like '%tools%'"
```

Report

The following command displays all users who have TeamConnection superuser privilege.

```
teamc report -view users -where "superuser = 'yes'"
```

The following command displays an order by clause with two column names. **Asc** orders the version SID column in ascending order and **desc** orders the path name column in descending order. This query reports changes to parts in ascending order, and the path names of the parts in descending order.

```
teamc report -view changeview -where "defectName = '1491' and  
releaseName = 'projectA_r11' order by versionSID asc, pathName desc"
```

The following command displays all work areas that are in the integrate state and that are not in a driver.

```
teamc report -view workAreaView -where "state = 'integrate' and  
releasename = 'projectA_r1' and id not in (select workareaid  
from drivemembers)"
```

The following command displays all part changes for **src\kernel\ibmesa\io\dkios.c** that were committed in a driver on or before October 21, 1995.

```
teamc report -view changeview -where "pathname =  
'src\kernel\ibmesa\io\dkios.c' and drivename in (select name from  
driverview where commitdate < '1995/10/21')"
```

The following command displays the component **comp1** and all its children.

```
teamc report -view compview -where "id in (select childid from compmemberview  
where parentCompName='comp1')"
```

The following command displays the children of component **comp1**.

```
teamc report -view compview -where "id in (select childid from compmemberview  
where parentCompName='comp1') and id not in (select parentid from  
compmemberview where parentCompName='comp1')"
```

The following command displays the component **comp1** and all parents.

```
teamc report -view compview -where "id in (select parentid from compmemberview  
where childCompName='comp1')"
```

The following command displays the parents of component **comp1**.

```
teamc report -view compview -where "id in (select parentid from compmemberview  
where childCompName='comp1') and id not in (select parentid from  
compmemberview where parentCompName='comp1')"
```

report -view partView

The report -view partView action enables you to query the PartView table in the TeamConnection database. This action is somewhat different from the report -view Name action. With this action you can specify the following additional attributes to focus your database query:

- -workarea Name or -version Name
- -release Name
- -parent Name
- -parenttype Name

Attributes

The report -view partView command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-long

Produces report output in stanza format, with additional important information shown in a table format:

- Each database record is a stanza.
- Each stanza line consists of a field and its corresponding values.

Specify only one of -long, -raw, -stanza, or -table.

-parent Name

Specifies the parent of the object.

-parenttype Name

Specifies the type of the parent part.

-raw

Displays reports in raw format. Specify only one of -long, -raw, -stanza, or -table.

See “Appendix. Querying the TeamConnection database” on page 261. Refer to the appendix in the *Commands Reference* for the format of raw output for each report view.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-stanza

Produces report output in stanza format:

- Each database record is a stanza.
- Each stanza line consists of a field and its corresponding values.

Report

Specify only one of -long, -raw, -stanza, or -table.

-table

Produces report output in the following table format:

- Each field is displayed as a column heading.
- Field values appear under respective column headings.
- Each row corresponds to one database record.

Specify only one of -long, -raw, -stanza, or -table. The table view is the default format of report output.

-verbose

TeamConnection displays a confirmation message after you issue the command.

-version Name

Specifies the version of the object. Specify either -version or -workarea or both.

-where Text

Defines the selection criteria to query the specified table or view using valid syntax.

-workarea Name . . .

The work area name. Specify either -version or -workarea.

(Environment variable: TC_WORKAREA)

Related information

For the order and description of field names that are output for various views when you issue the **report** command using the **-raw** flag, refer to “Appendix. Querying the TeamConnection database” on page 261. the *Comands Reference*.

For a list of the views and a description of their fields, or a list of the tables that can be specified as subselects in the **-where** clause and a description of their fields, refer to “Appendix. Querying the TeamConnection database” on page 261. the *Commands Reference*.

Chapter 22. Size

Use the **size** command to create, delete, and reassign sizing records for a defect or feature that is in the size state, or to indicate sizing information. A sizing record must be created explicitly by the defect or feature owner. A sizing record indicates the time and resources needed to resolve a defect or implement a feature in one component for a release. Each sizing record is uniquely identified by a defect or feature identifier, a component, and a release.

If you are the owner of the component in which the defect must be resolved or the feature must be implemented, you are also the owner of the sizing record by default. Sizing information must be entered as text on a sizing record.

All sizing records must be marked either with **accept** or **reject** in order to move the defect or feature from the size state to the review state. Work areas and fix records are created for all sizing records marked accept, when the defect or feature is accepted, when the track process is enabled for the release and the design, size, and review process is enabled for the component. Otherwise, you must create the work area manually.

Command summary

In the following syntax statements, the first letter of each argument is capitalized, for example `-family Name`. Replace these arguments with a value that is valid for your TeamConnection database. If you want to issue a command for the family named **testfam**, for example, replace `-family Name` with `-family testfam`.

The syntax statements for the **size** command are:

```
teamc size -accept { -feature Name ... -defect Name ... }
                  -component Name ... -release Name
                  -family Name -sizing Text [-become Name] [-verbose]

teamc size -assign -to Name { -feature Name ... -defect Name ... }
                  -component Name ... -release Name
                  -family Name [-become Name] [-verbose]

teamc size -create { -feature Name ... -defect Name ... }
                  -component Name ... -release Name
                  -family Name [-become Name] [-verbose]

teamc size -delete { -feature Name ... -defect Name ... }
                  -component Name ... -release Name
                  -family Name [-become Name] [-verbose]

teamc size -reject { -feature Name ... -defect Name ... }
                  -component Name ... -release Name
                  -family Name [-become Name] [-verbose]
```

size -accept

Indicates that the sizing information is entered and complete for the corresponding defect or feature, release, and component. This action is used to record initial sizing information.

Size

Attributes

The size -accept command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-component Name

The component associated with the object. Different components can manage different versions of the same object.

(Environment variable: TC_COMPONENT.)

-defect Name

The defect identifier for the object. Either -defect or -feature is required.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-feature Name

The feature identifier for the object. Either -defect or -feature is required.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-sizing Text

The sizing information for the proposed defect or feature change in the specified component and release.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

Assume you own the sizing record for feature **483** in component **graphix** for release **21graphix**. The following command specifies that 10 person-days are required to implement the feature.

```
teamc size -accept -feature 483 -component graphix -release 21graphix  
-sizing "10 person-days"
```

size -assign

Reassigns ownership of the specified sizing record to another user ID.

Attributes

The size -assign command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-component Name

The component associated with the object. Different components can manage different versions of the same object.

(Environment variable: TC_COMPONENT.)

-defect Name

The defect identifier for the object. Either -defect or -feature is required.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-feature Name

The feature identifier for the object. Either -defect or -feature is required.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-to Name

The user ID to which you want to reassign the object. The user ID you specify becomes the owner of the object.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

You own the sizing record for feature **483** in the **graphix** component for the **20graphix** release. The following command assigns the sizing record to user ID

Size

mary. Responsibility for sizing the feature is reassigned to the user **mary**. The sizing record is uniquely defined by the feature number **483**, release **20graphix**, and component **graphix**.

```
teamc size -assign -feature 483 -component graphix -release 20graphix  
-to mary
```

size -create

Creates a sizing record for the corresponding defect or feature, release, and component.

The associated component's process must include the design, size, and review (DSR) subprocess.

Attributes

The size -create command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-component Name

The component associated with the object. Different components can manage different versions of the same object.

(Environment variable: TC_COMPONENT.)

-defect Name

The defect identifier for the object. Either -defect or -feature is required.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-feature Name

The feature identifier for the object. Either -defect or -feature is required.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

The following command creates a sizing record for feature **483** in the **graphix** component for the **21graphix** release. The owner of the sizing record is the owner of the **graphix** component.

```
teamc size -create -feature 483 -component graphix -release 21graphix
```

size -delete

Deletes the specified sizing record.

Attributes

The size -delete command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-component Name

The component associated with the object. Different components can manage different versions of the same object.

(Environment variable: TC_COMPONENT.)

-defect Name

The defect identifier for the object. Either -defect or -feature is required.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-feature Name

The feature identifier for the object. Either -defect or -feature is required.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-verbose

TeamConnection displays a confirmation message after you issue the command.

size -reject

Indicates that resolving the defect or implementing the feature does not require changes in the corresponding component. If old sizing information exists for this sizing record, it is deleted.

Attributes

The size -reject command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-component Name

The component associated with the object. Different components can manage different versions of the same object.

(Environment variable: TC_COMPONENT.)

-defect Name

The defect identifier for the object. Either -defect or -feature is required.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-feature Name

The feature identifier for the object. Either -defect or -feature is required.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

Assume you own the sizing record for defect **APAR20** in component **charting** for release **32charting**. The following command specifies that no changes are required in that component and no additional resources are required. The **-reject** action flag indicates that no changes are required for defect **APAR20** in the component **charting** for the release **32charting**.

```
teamc size -reject -defect APAR20 -component charting -release 32charting
```

Related information

See the following related commands:

- Defect
- Feature
- Release
- Report
- Workarea

Size

Chapter 23. Tclogin

Use the **tclogin** command to login to, log out of, or display a list of user login IDs currently logged in to the family server. This command is required only if your family database uses the PASSWORD_ONLY security option. It may also be needed if your family database uses the PASSWORD_OR_HOST security option.

Command summary

In the following syntax statements, the first letter of each argument is capitalized, for example `-family Name`. Replace these arguments with a value that is valid for your TeamConnection database. If you want to issue a command for the family named **testfam**, for example, replace `-family Name` with **—family testfam**.

The syntax statements for the **tclogin** command are:

```
teamc tclogin -login -family Name [-password Name] [-become Name] [-verbose]
```

```
teamc tclogin -logout -family Name [-become Name] [-verbose]
```

```
teamc tclogin -view
```

tclogin -login

Use the `tclogin -login` command to log in a user ID to the family server.

Attributes

The `tclogin -login` command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-password Name

Specifies the password for the user ID. This attribute is required if the family uses the PASSWORD_ONLY or PASSWORD_OR_HOST and the user has no host list entry security options. A password must be at least one character long and can be up to thirty-one characters long.

The `-password` attribute is not available on Windows 3.1. Password authentication is not available when multiple server daemons are running.

Tclogin

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

To login to a TeamConnection family, issue the following command:

```
teamc tclogin -login -family testfam -password myPassword
```

tclogin -logout

Use the tclogin -logout command to log out a user ID from the family server.

Attributes

The tclogin -logout command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

To log out from a TeamConnection family, issue the following command:

```
teamc tclogin -logout -family testfam
```

tclogin -view

Use the tclogin -view command to display a list of TeamConnection login IDs currently logged in to the family server.

Examples

To display a list of TeamConnection login IDs currently logged into a family server, issue the following command:

```
teamc tclogin -view
```

This command displays a list similar to the following:

userLogin	family
chrisc	testfam
billw	testfam
gener	testfam

3 records selected

Related information

See the following related commands:

User

Tclogin

Chapter 24. Test

Use the **test** command to indicate the results of an environment test on a test record associated with a work area.

If a release has an environment list, test records are created according to the entries in that list whenever a new work area is created for that release (provided that the release's process includes the test subprocess). Each test record includes the environment name and user ID specified on the release environment list, and the defect or feature identifier of the work area. The owner of the user ID is the tester who owns the test record.

Test records are activated (that is, they are moved to the ready state) when the associated work area moves to the test state and the proposed change is ready for environment testing. When results are entered for all the environment test records associated with a work area, the state of that work area changes to complete. Even if you reject a test record, the work area changes to the complete state. Create another work area to address any changes still required.

Command summary

In the following syntax statements, the first letter of each argument is capitalized, for example `-family Name`. Replace these arguments with a value that is valid for your TeamConnection database. If you want to issue a command for the family named **testfam**, for example, replace `-family Name` with `—family testfam`.

The syntax statements for the **test** command are:

```
teamc test -abstain -workarea ... -family Name
          -release Name ... -environment Name ...
          [-tester Name] [-become Name] [-verbose]
```

```
teamc test -accept -workarea ... -family Name
          -release Name ... -environment Name ...
          [-tester Name] [-become Name] [-verbose]
```

```
teamc test -assign -to Name -workarea ... -release Name ...
          -environment Name ... -family Name [-tester Name]
          [-become Name] [-verbose]
```

```
teamc test -create -workarea ... -family Name
          -release Name ... -environment Name ...
          [-tester Name] [-become Name] [-verbose]
```

```
teamc test -delete -workarea ... -family Name
          -release Name ... -environment Name ...
          [-become Name] [-verbose]
```

```
teamc test -reject -workarea ... -family Name
          -release Name ... -environment Name ...
          [-tester Name] [-become Name] [-verbose]
```

test -abstain

Abstains from testing for a release environment.

Attributes

The test -abstain command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-environment environment

Specifies the environment in which the testing is done.

Note: The tester/environment name combination becomes an entry on the environment list for the release.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-tester Name

The user responsible for testing in a given environment.

-verbose

TeamConnection displays a confirmation message after you issue the command.

-workarea Name . . .

The work area name.

(Environment variable: TC_WORKAREA)

Examples

Assume that you have superuser privilege and that your TC_RELEASE environment variable is set to **tripod3**. The following command indicates that **jane**, the owner of the test record for work area **7966** in release **tripod3**, will abstain from marking test results in the **PCVersion2** environment. The test record owned by the user **jane** for work area **7966** in the environment **PCVersion2** is marked **abstain**.

You can mark the test record owned by another user because you have superuser privilege. The work area moves to the complete state if this is the last test record for the work area to be marked with test results.

```
teamc test -abstain -workarea 7966 -environment PCVersion2 -tester jane
```

test -accept

Indicates successful results for a release environment test.

Attributes

The test -accept command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-environment environment

Specifies the environment in which the testing is done.

Note: The tester/environment name combination becomes an entry on the environment list for the release.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-tester Name

The user responsible for testing in a given environment.

-verbose

TeamConnection displays a confirmation message after you issue the command.

-workarea Name . . .

The work area name.

(Environment variable: TC_WORKAREA)

Test

Examples

Assume that you are responsible for testing whether or not defect **7966** was successfully resolved in the **PCVersion1** environment for release **tripod3**. The following command accepts the test record for work area **7966** if your TC_RELEASE environment variable is set to **tripod3**. The test record you own is marked **accept**, indicating successful test results. The work area moves to the complete state if this is the last test record for the work area to be marked with test results.

```
teamc test -accept -workarea 7966 -environment PCVersion1
```

test -assign

Assigns a test record to another user ID.

Attributes

The test -assign command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-environment environment

Specifies the environment in which the testing is done.

Note: The tester/environment name combination becomes an entry on the environment list for the release.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-tester Name

The user responsible for testing in a given environment.

-to Name

The user ID to which you want to reassign the object. The user ID you specify becomes the owner of the object.

-verbose

TeamConnection displays a confirmation message after you issue the command.

-workarea Name . . .

The work area name.

(Environment variable: TC_WORKAREA)

Examples

Assume that you own the test record for work area **7562** in release **20gos** and environment **Model1**. The following command assigns this test record to user ID **amy** so that the owner of that user ID assumes the testing responsibility.

```
teamc test -assign -to amy -workarea 7562 -environment Model1
-release 20gos
```

test -create

Creates a test record.

Attributes

The test -create command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-environment environment

Specifies the environment in which the testing is done.

Note: The tester/environment name combination becomes an entry on the environment list for the release.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-tester Name

The user responsible for testing in a given environment.

Test

-verbose

TeamConnection displays a confirmation message after you issue the command.

-workarea Name . . .

The work area name.

(Environment variable: TC_WORKAREA)

Examples

Assume that you own the test environment **ModelT** and tester **gdixon** has been added to release **bos41M**. When work areas are created in the future, test records will be created for this new environment. But work areas for defects **7703** and **7715** already exist and should also be tested for the new environment. The following command creates test records for these defects.

```
teamc test -create -tester gdixon -workarea 7703 7715 -environment ModelT -release bos41M
```

test -delete

Deletes a test record.

Attributes

The test -delete command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-environment environment

Specifies the environment in which the testing is done.

Note: The tester/environment name combination becomes an entry on the environment list for the release.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-verbose

TeamConnection displays a confirmation message after you issue the command.

-workarea Name . . .

The work area name.

(Environment variable: TC_WORKAREA)

Examples

You decide that defect **7711** is not applicable to environment **ModelT** in release **bos41M**, so the associated test record is no longer needed. The following command would delete it.

```
teamc test -delete -workarea 7711 -environment ModelT -release bos41M
```

test -reject

Indicates unsuccessful results for a release environment test.

Attributes

The test -reject command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-environment environment

Specifies the environment in which the testing is done.

Note: The tester/environment name combination becomes an entry on the environment list for the release.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-tester Name

The user responsible for testing in a given environment.

-verbose

Test

TeamConnection displays a confirmation message after you issue the command.

-workarea Name . . .

The work area name.

(Environment variable: TC_WORKAREA)

Related information

See the following related commands:

- Driver
- Environment
- Release
- Report
- Workarea

Chapter 25. User

Use the **user** command to create new user IDs, to modify information associated with user IDs, and delete user IDs. Superuser privilege is required to create user IDs for new users, delete other user IDs, and modify the superuser privilege of a user. You can modify your own user ID information but cannot give yourself TeamConnection superuser privilege.

Although the **user** command establishes required user ID information, actual host access for a user ID must be created using the **host** command. Host access is only needed for host-based security.

User IDs cannot be deleted if they have work pending or if they own objects. You cannot use a deleted user ID, but you can re-create it and rename it.

Command summary

In the following syntax statements, the first letter of each argument is capitalized, for example `-family Name`. Replace these arguments with a value that is valid for your TeamConnection database. If you want to issue a command for the family named **testfam**, for example, replace `-family Name` with `—family testfam`.

The syntax statements for the **user** command are:

```
teamc user -create -login Name -address Name -family Name
          [-name Text] [-area Name] [+super] [-become Name] [-verbose]

teamc user -configInfo -family Name [-become Name] [-raw]

teamc user -delete Name ... -family Name [-become Name] [-verbose]

teamc user -modify Name ... -family Name { -login Name -name Text
          -address Name -area Name [+super | -super]
          -password [newpass &#brk;oldpass]] } [-become Name] [-verbose]

teamc user -recreate Name ... -family Name [-become Name] [-verbose]

teamc user -view Name ... -family Name [-long] [-become Name]
          [-verbose]
```

user -create

Adds user IDs by specifying a value for the **-login** flag. User IDs must be unique within a family. Should you attempt to create a user at an unreachable host, an error message will appear.

Because your family administrator can create new fields, the attributes for the **-create** action can be different from those in your family. Those listed here represent the shipped default fields only. For a list of the field properties and flags in use in your family, use the **user -configInfo** command or see your family administrator. For more information on configurable fields, refer to the *Administrator's Guide*.

User Attributes

The user -create command takes the following attributes:

-address Name

Specifies where notification messages are sent for this user. For this attribute, specify the user's mail address in the form `login@hostname`.

-area Name

Specifies the area or department of a user.

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-login Name

TeamConnection user ID.

-name Name

Specifies the user's full name.

+super

Grants TeamConnection superuser privilege to a specified user ID.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

The following command creates a user ID for a new user with the user ID **dorrie** and the mailing address **dorrie@cansas**. Because TeamConnection superuser authority was not specified, this user does not have this authority.

```
teamc user -create -login dorrie -address dorrie@cansas  
-name "Julie Karland" -area tools
```

user -configInfo

Shows configurable field properties for users in the specified family. The information is returned in a fixed ASCII table format.

Attributes

The user `-configInfo` command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-raw

Displays configurable field information in raw format.

Examples

- The following command displays the configurable fields defined for users in family **rdev**.

```
teamc user -configInfo -family rdev
```

The following is an example of the output provided for this command. This example assumes that the only configurable field defined is called **userTest**.

Attribute Name	DB Column Name	Create/Required	Type
userTest	userTest	yes/no	test

- The following command displays the configurable fields defined for users in family **rdev** in raw format.

```
teamc user -configInfo -family rdev -raw
```

The following is an example of the output provided for this command. This example assumes that the only configurable field defined is called **userTest**.

```
User Test|User Test|userTest|userTest|yes|no|test
```

user -delete Name . . .

Deletes specified user IDs.

Attributes

The user `-delete Name . . .` command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

User

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

The following command deletes the user ID **jack**. You can delete a user ID only if all associated objects the user ID owns have been deleted or reassigned and the user ID is removed from approver, environment, access, and notification lists. A deleted user ID can be re-created.

```
teamc user -delete jack
```

user -modify Name . . .

The user -modify Name . . . action modifies information associated with specified user IDs. Only a superuser can modify superuser privilege. You must include at least one attribute to be modified with the command.

Use the user -modify Name . . . command to set passwords for user IDs. Passwords are required if the family uses the PASSWORD_ONLY or PASSWORD_OR_HOST and the user has no host list entry security options.

Because your family administrator can create new fields, the attributes for the **-modify** action can be different from those in your family. Those listed here represent the shipped default fields only. For a list of the field properties and flags in use in your family, use the **user -configInfo** command or see your family administrator. For more information on configurable fields, refer to the *Administrator's Guide*.

Attributes

The user -modify Name . . . command takes the following attributes:

-address Name

Specifies where notification messages are sent for this user. For this attribute, specify the user's mail address in the form login@hostname.

-area Name

Specifies the area or department of a user.

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-login Name

TeamConnection user ID.

-name Name

Specifies the user's full name.

-password Name

Specifies the password for the user ID. This attribute is required if the family uses the PASSWORD_ONLY or PASSWORD_OR_HOST and the user has no host list entry security options. A password must be at least one character long and can be up to thirty-one characters long.

The -password attribute is not available on Windows 3.1. Password authentication is not available when multiple server daemons are running.

+super

Grants TeamConnection superuser privilege to a specified user ID.

-super

Removes TeamConnection superuser privilege for a specified user ID. Specify only one of +super or -super.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

Assume that you are logged on to host **lab2** with the TeamConnection user ID **jane**. That user ID does not have superuser privilege. You have a host list entry for user ID **admin** (that is, **jane@lab2** is on the host list for **admin**), and that user ID does have superuser privilege. To give the user ID **george** superuser privilege, you must become user **admin** to issue the command successfully. The following command changes your TeamConnection user ID to **admin** and provides superuser authority to user ID **george**.

```
teamc user -modify george +super -become admin
```

The following command grants superuser privilege to the user with the login name **dorothy**.

```
teamc user -modify dorothy +super
```

The following command modifies information for multiple user IDs. The area specification is changed to **tools07** for the user IDs **jack** and **sally**. This command also removes TeamConnection superuser privilege from both user IDs.

User

```
teamc user -modify jack sally -area tools07 -super
```

user -recreate Name . . .

Re-creates previously deleted user IDs.

Attributes

The user -recreate Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-verbose

TeamConnection displays a confirmation message after you issue the command.

user -view Name . . .

Shows information about specified user IDs.

Attributes

The user -view Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-long

Displays detailed information for the specified objects.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

The following command displays all the host list entries for user ID **shirley**.

```
teamc user -view shirley -long
```

The following command displays information for specified user IDs.

```
teamc user -view dorothy jack sally
```

Related information

See the following related commands:

- Host

- Report

User

Chapter 26. Verify

Use the **verify** command to verify the resolution of defects or the implementation of features or to reassign ownership of existing *verification records*.

A verification record is created for the originator of a defect or a feature when the defect or feature is accepted and the component that manages the defect or feature has a process that includes the defectVerify or featureVerify subprocesses. Additional verification records are created for the originators of duplicate defects or features and attached to the active defect or feature. Defects can be specified as duplicates of features, and features can be specified as duplicates of defects.

Verification records become active when a defect or feature changes from the working state to the verify state. When results have been recorded for all the verification records for a defect or feature, and when all of the work areas of the defect or feature are complete, the defect or feature changes from the verify state to the closed state. If a verification record is rejected, the defect or feature is returned to the working state.

Command summary

In the following syntax statements, the first letter of each argument is capitalized, for example `-family Name`. Replace these arguments with a value that is valid for your TeamConnection database. If you want to issue a command for the family named **testfam**, for example, replace `-family Name` with `—family testfam`.

The syntax statements for the **verify** command are:

```
teamc verify -abstain { -defect Name ... -feature Name ... }
                    -family Name [-tester Name] [-become Name] [-verbose]
```

```
teamc verify -accept { -defect Name ... -feature Name ... }
                    -family Name [-tester Name] [-become Name] [-verbose]
```

```
teamc verify -assign -to Name { -defect Name ...
                    -feature Name ... }
                    -family Name [-tester Name] [-become Name] [-verbose]
```

```
teamc verify -reject { -defect Name ... -feature Name ... }
                    -family Name [-tester Name] [-become Name] [-verbose]
```

verify -abstain

Indicates that the owner of the verification record abstains from verifying defect resolution or feature implementation.

Attributes

The `verify -abstain` command takes the following attributes:

-become Name

Verify

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-defect Name

The defect identifier for the object. Either -defect or -feature is required.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-feature Name

The feature identifier for the object. Either -defect or -feature is required.

-tester Name

The user responsible for testing in a given environment.

-verbose

TeamConnection displays a confirmation message after you issue the command.

verify -accept

Indicates successful verification of the defect resolution or feature implementation.

Attributes

The verify -accept command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-defect Name

The defect identifier for the object. Either -defect or -feature is required.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-feature Name

The feature identifier for the object. Either -defect or -feature is required.

-tester Name

The user responsible for testing in a given environment.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

The following command indicates that a defect resolution was verified successfully.

```
teamc verify -accept -defect 976
```

verify -assign

Reassigns the ownership of the verification record for a specified defect or feature to another user.

Attributes

The verify -assign command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-defect Name

The defect identifier for the object. Either -defect or -feature is required.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-feature Name

The feature identifier for the object. Either -defect or -feature is required.

-tester Name

The user responsible for testing in a given environment.

-to Name

The user ID to which you want to reassign the object. The user ID you specify becomes the owner of the object.

-verbose

Verify

TeamConnection displays a confirmation message after you issue the command.

Examples

The following command reassigns ownership of a verification record. If you are a superuser, the current owner of the verification record for feature **899**, or you have VerifyAssign explicit authority, then you can type this command to reassign the verification record to the user ID **lee**.

```
teamc verify -assign -feature 899 -to lee
```

verify -reject

Indicates unsuccessful verification of the defect resolution or feature implementation.

Attributes

The verify -reject command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-defect Name

The defect identifier for the object. Either -defect or -feature is required.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-feature Name

The feature identifier for the object. Either -defect or -feature is required.

-tester Name

The user responsible for testing in a given environment.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

The following command indicates that a defect resolution was not verified successfully, type:

```
teamc verify -reject -defect 1001
```

Related information

See the following related commands:

- Defect
- Feature
- Release
- Workarea

Verify

Chapter 27. Workarea

A work area monitors the progress of changes to resolve a defect or implement a feature.

Use the **workarea** command to create, modify, reassign, delete, freeze, refresh, and view information about a work area, and to change the state of a work area. The states a work area moves through depend on the TeamConnection subprocesses included in the associated release process. A release process can include the track, approval, fix, driver, or test subprocesses, or none at all. For more information on the work area states and their associated subprocesses, refer to the *TeamConnection User's Guide*.

If the track process is turned on for the release, a work area must be associated with a defect or feature. The default name for these work areas is the name of the defect or feature.

You can also create a work area with a specified name. If the release does not have the track process, you must create the work area with a specified name. The user who creates the work area becomes the owner of the work area unless a different owner is specified when the work area is created.

You can also freeze or refresh your work area. You freeze a work area to save the current version of a part or parts in the work area by using the **-freeze** action. The version that you freeze represents a snapshot in time of the parts in your work area. You refresh your work area to ensure that parts in the work area are the most current by using the **-refresh** action. When you refresh a work area, you get the most current view of all the parts in your work area from that source.

Changes associated with the work area are not visible to the release until you commit the work area. If your release does not have the driver process enabled, the TeamConnection product commits the work area implicitly via the **workarea -integrate** command.

If a defect or feature is linked to more than one release, multiple work areas exist for that defect or feature. The work areas required for a defect or feature are created according to sizing records only if the dsr subprocess is in effect. Otherwise, the workarea must be created when the defect or feature changes to the working state. Defect or feature owners can create additional work areas if the defect or feature is in the working state.

To determine the prerequisite and corequisite work areas for a particular work area, use the **workarea -check** action. By default, the **workarea -check** command lists all prerequisite and corequisite areas relative to the current state of the release. Specify the driver name to determine the prerequisite and corequisite work areas relative to an earlier committed driver. You will get a list of all prerequisite and corequisite work areas including any that were integrated after the specified driver was committed.

Command summary

In the following syntax statements, the first letter of each argument is capitalized, for example `-family Name`. Replace these arguments with a value that is valid for your TeamConnection database. If you want to issue a command for the family named **testfam**, for example, replace `-family Name` with `—family testfam`.

The syntax statements for the **workarea** command are:

```
teamc workarea -assign Name ... -to Name
               -release Name ... -family Name [-become Name]
               [-verbose]

teamc workarea -cancel Name ... -release Name ...
               -family Name [-become Name] [-verbose]

teamc workarea -check Name ... -release Name ... [-full] [-exclude]
               -family Name [-driver Name] [-level Name] [-become Name] [-verbose]

teamc workarea -commit Name ... -release Name ...
               -family Name [-become Name] [-verbose] [-trackcommithold]

teamc workarea -complete Name ... -release Name ...
               -family Name [-become Name] [-verbose]

teamc workarea -create -name Name ... -release Name ...
               -family Name [-owner Name] [-target Name] [-become Name]
               [-verbose] *

teamc workarea -create -name Name ... {-defect Name |
               -feature Name} -release Name ... -family Name [-owner Name]
               [-target Name] [-become Name] [-verbose] **

teamc workarea -create {-defect Name ... -feature Name ...}
               -release Name ... -family Name [-owner Name] [-target Name]
               [-become Name] [-verbose] ***

teamc workarea -export Name ... -release Name -file Name -family Name
               [-become Name] [-verbose]

teamc workarea -fix Name ... -release Name ...
               -family Name [-become Name] [-verbose] [-trackfixhold]

teamc workarea -freeze Name ... -release Name ...
               -family Name [-become Name] [-verbose]

teamc workarea -import Name ... -release Name -file Name [-makeComponent]
               -family Name [-become Name] [-verbose]

teamc workarea -integrate Name ... -release Name ...
               -family Name [-force] [-become Name] [-verbose]

teamc workarea -modify Name ... -target Name
               -release Name ... -family Name [-become Name] [-verbose]
```

```
teamc workarea -refresh Name ... -release Name ...
                    -family Name [-become Name] [-source Name] [-verbose]
```

```
teamc workarea -test Name ... -release Name ... -family Name
                    [-become Name] [-verbose] [-tracktesthold]
```

```
teamc workarea -undo Name ... -release Name ...
                    -family Name [-become Name] [-verbose]
```

```
teamc workarea -view Name ... -family Name -release Name ...
                    [-long] [-become Name] [-verbose]
```

Notes:

- * Use this syntax to create work areas in non-tracking releases
- ** This syntax will create named work areas in tracking releases
- *** This syntax will default the work area name to be the same as the defect or feature name. Used in tracking releases.

workarea -assign Name . . .

Reassigns ownership of specified work areas to another user ID.

Attributes

The workarea -assign Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-to Name

The user ID to which you want to reassign the object. The user ID you specify becomes the owner of the object.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Workarea

Examples

The following command reassigns owner responsibility for a work area to the user ID **jack**:

```
teamc workarea -assign 8803 -release 20graphix -to jack
```

workarea -cancel Name . . .

Cancels the specified work areas. This is valid only if no changes have been made to parts referencing the work areas.

If the work area being canceled is a prerequisite for another workarea, this command removes the prerequisite relationship.

Attributes

The workarea -cancel Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-verbose

TeamConnection displays a confirmation message after you issue the command.

workarea -check Name . . .

Displays the prerequisite and corequisite work areas for the specified work areas.

- **full** and **exclude** attributes are optional
- **exclude** and **full** both can not be specified
- **exclude** and **level** can not be specified
- **full** and **level** can be specified
- **full** and **no level** can be specified

Attributes

The workarea -check Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-driver Name

Specifies the driver for which the command is issued.

- For defects this is the driver in which the defect was discovered.
- For driver members and work areas this is the driver in which the member is created or modified.

-exclude

Specifies that all given work areas are to be checked as a group. The command output will identify the workarea(s) in the group that have requisites that are not in the group.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-full Name

Specifies that a complete list of requisites be provided. Without this attribute, a given requisite will only be listed once.

-level Name

Specifies the name of the committed level to use as a base when determining prerequisite and corequisite work areas for a given work area.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

The following command checks whether prerequisite or corequisite work areas exist for a work area relative to a particular driver. All prerequisite and corequisite work

Workarea

areas that exist for the work area for defect **8734** in the **10graphix** release are displayed, including those for part changes committed after the commit date of the driver **9028**.

```
teamc workarea -check 8734 -release 10graphix -driver 9028
```

workarea -commit Name . . .

Changes the state of the specified work areas from integrate to commit, if no part changes were made for the work area. This action is required only if the work area is not committed in a driver.

The associated release's process must include the track subprocess.

An optional attribute may be used to keep a track in the commit state.

Attributes

The workarea -commit Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-trackcommithold

With the **trackcommithold** attribute a track will remain in the **commit** state:

- When the **Level - complete** command is issued for a release with a level process.
- When the final **Fix - complete** command is issued for a release without a level subprocess and with the fix subprocess.
- When the **Track - integrate** command is issued for a release without a level subprocess and without the fix subprocess.

-verbose

TeamConnection displays a confirmation message after you issue the command.

workarea -complete Name . . .

Changes the state of the specified work areas from test to complete if no part changes were made for the work area. No additional state changes can occur after a work area reaches the complete state.

The associated release's process must include the track subprocess.

Attributes

The workarea -complete Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-verbose

TeamConnection displays a confirmation message after you issue the command.

workarea -create

The workarea -create action creates a work area for the specified release.

This action is one of three variations on workarea -create:

- Use workarea -create to create work areas in non-tracking releases.
- Use workarea -create {-defect | -feature} to create named work areas in tracking releases.
- Use workarea -create -defect -feature to create work areas in tracking releases. The work area name is the same as the defect or feature name.

Attributes

The workarea -create command takes the following attributes:

-become Name

Workarea

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-name Name

Specifies a name for the object.

-owner Name

Specifies the user ID of the owner of the object.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-target Name

Destination of the copy.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

The following command creates a work area named **work1** in release **10graphix**. Release **10graphix** does not have the track process turned on.

```
teamc workarea -create -name -work1 -release 10graphix
```

workarea -create {-defect | -feature}

The `workarea -create {-defect | -feature}` action creates a work area for the specified defect or feature in a given release. If there is no approver list for the related release, then the new work area is created in the fix state. If an approver list exists, the initial state is approve.

The associated release's process must include the track subprocess.

This command is a variation of the `workarea -create` command.

- Use `workarea -create` to create work areas in non-tracking releases.
- Use `workarea -create {-defect | -feature}` to create named work areas in tracking releases.
- Use `workarea -create -defect -feature` to create work areas in tracking releases. The work area name is the same as the defect or feature name.

In this variation you include either the `-defect` or the `-feature` attribute to create a work area for a specific defect or feature.

Attributes

The workarea `-create` `{-defect | -feature}` command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-defect Name

The defect identifier for the object.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-feature Name

The feature identifier for the object.

-name Name

Specifies a name for the object.

-owner Name

Specifies the user ID of the owner of the object.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-target Name

Specifies a target (such as, a driver or a date) for defect resolution or availability.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

The following command creates work areas for a defect that requires resolution for three releases.

Workarea

This command creates three work areas for defect **8734**: one for each of the three releases and all named **8734**.

```
teamc workarea -create -defect 8734 -release 20graphix 10graphix  
21charting
```

workarea -create -defect -feature

The workarea -create -defect -feature action creates a work area for the specified defect and feature in a given release. If there is no approver list for the related release, then the new work area is created in the fix state. If an approver list exists, the initial state is approve.

The associated release's process must include the track subprocess.

This command is a variation of the workarea -create command.

- Use workarea -create to create work areas in non-tracking releases.
- Use workarea -create {-defect | -feature} to create named work areas in tracking releases.
- Use workarea -create -defect -feature to create work areas in tracking releases. The work area name is the same as the defect or feature name.

In this variation you include both the -defect and the -feature attributes to create a work area for a specific defect and feature.

Attributes

The workarea -create -defect -feature command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-defect Name

The defect identifier for the object.

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-feature Name

The feature identifier for the object.

-owner Name

Specifies the user ID of the owner of the object.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-target Name

Specifies a target (such as, a driver or a date) for defect resolution or availability.

-verbose

TeamConnection displays a confirmation message after you issue the command.

workarea -export Name . . .

The workarea -export Name . . . command exports the parts in the work area (or work areas) specified by the *Name* argument from a release to the file name specified in the -file attribute.

This function is useful for exporting information from one family into another. You can export the information from one family into a CDF file and then import the CDF file into another family.

For more information about the TeamConnection import and export functions refer to the *Tool Builder's Development Guide*. The *Tool Builder's Development Guide* also provides more information about common data format (CDF), the file format from which TeamConnection imports information and to which it exports information.

Note: You cannot use the import and export functions to migrate information from CMVC to TeamConnection or from one release of TeamConnection to another. Refer to the *Administrator's Guide* for information on the TeamConnection migration tool.

Attributes

The workarea -export Name . . . command takes the following attributes:

-file filename

Specifies the name of the file to which the part or parts are to be exported. The information is exported to the file in CDF format.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-verbose

TeamConnection displays a confirmation message after you issue the command.

Workarea

Examples

workarea -fix Name . . .

Moves the specified work areas from the integrate state to the fix state.

The associated release's process must include the track subprocess.

An optional attribute may be used to prevent a track from going to integrate when all fix records are complete. Only relevant when level process not being used.

Attributes

The workarea -fix Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-trackfixhold

With the **trackfixhold** attribute and the fix subprocess, a track will remain in the **fix** state rather than moving to the **integrate** state when the final **Fix - complete** command has been issued.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

Assume that the work area **8734** and the release specified by your TC_RELEASE environment variable is not a member of a driver. The following command changes the work area from integrate state to fix state. If a work area is in a driver, you must delete it from the driver before you can move it back to the fix state.

```
teamc workarea -fix 8734
```

workarea -freeze Name . . .

Saves the state of a work area.

Attributes

The workarea -freeze Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

The following command freezes work area **work1** in release **10graphix**

```
teamc workarea -freeze work1 -release 10graphix
```

workarea -import Name . . .

The workarea -import Name . . . command imports the parts in the work area (or work areas) specified by the *Name* argument from the file name specified in the -file attribute to a release.

This function is useful for exporting information from one family into another. You can export the information from one family into a CDF file and then import the CDF file into another family.

For more information about the TeamConnection import and export functions refer to the *Tool Builder's Development Guide*. The *Tool Builder's Development Guide* also provides more information about common data format (CDF), the file format from which TeamConnection imports information and to which it exports information.

Workarea

Note: You cannot use the import and export functions to migrate information from CMVC to TeamConnection or from one release of TeamConnection to another. Refer to the *Administrator's Guide* for information on the TeamConnection migration tool.

Attributes

The workarea -import Name . . . command takes the following attributes:

-file filename

Specifies the name of the file from which the work area is to be imported.

-makeComponent

If the component defined in the file from which the work area is being imported does not already exist, this attribute enables TeamConnection to create the component. If the component does not exist and this attribute is not included with the command, then the import operation fails.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

workarea -integrate Name . . .

Changes the state of the specified work areas from fix to the next valid state governed by the release's process. For a release whose process includes the driver subprocess, this action is valid only if no part changes were made for the work area and the work area is not committed in a driver.

Attributes

The workarea -integrate Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-force

Integrates the work area even if all fix records are not complete.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

The following command integrates work area **work1** in release **10graphix**.

```
teamc workarea -integrate work1 -release 10graphix
```

workarea -modify Name . . .

Modifies the target field for the specified work areas.

Attributes

The workarea -modify Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-target Name

Destination of the copy.

-verbose

Workarea

TeamConnection displays a confirmation message after you issue the command.

workarea -refresh Name . . .

Refreshes the contents of one work area with another work area, driver, or release. This command also freezes the work area.

Attributes

The workarea -refresh Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-source Name

Specifies the name of the work area, driver, or release with the work area is to be refreshed.

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

The following command refreshes work area **work1** with respect to what is currently in release **10graphix**.

```
teamc workarea -refresh work1 -release 10graphix
```

workarea -test Name . . .

Changes the state of the specified work areas from commit to test. This action is required only if the work area is not committed in a driver. Usually, this change occurs when you issue the **driver -complete** command.

The associated release's process must include the track subprocess.

An optional attribute may be used to prevent a track from going to complete when all fix records are complete. Only relevant when level process not being used.

Attributes

The workarea -test Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-tracktesthold

With the **tracktesthold** attribute and the test subprocess, a track will remain in the **test** state rather than moving to the **complete** state when the final test is marked.

-verbose

TeamConnection displays a confirmation message after you issue the command.

workarea -undo Name . . .

The workarea -undo Name . . . action reverts a work area back to the last frozen version within a specified release. If the tracking subprocess is turned on, the work area must be in the fix state.

To revert a work area back to its last frozen state after a workarea -refresh Name . . . action, you need to execute the workarea -undo Name . . . action twice. The workarea -refresh Name . . . action actually freezes the work area twice: once before the refresh and once after the refresh. Freezing the work area before a refresh ensures that you can roll back the work area if an error occurs during the refresh action. To return the work area to its original state before the refresh, you need to undo both freezes.

Attributes

The workarea -undo Name . . . command takes the following attributes:

-become Name

Workarea

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-verbose

TeamConnection displays a confirmation message after you issue the command.

workarea -view Name . . .

Shows all information for the specified work areas.

Attributes

The workarea -view Name . . . command takes the following attributes:

-become Name

The user ID you want to issue TeamConnection commands from, if the user ID differs from your *login*. You assume the access authority of the user ID you specify.

(Environment variable: TC_BECOME)

-family Name

The family for which this command is being issued.

(Environment variable: TC_FAMILY)

-long

Displays detailed information for the specified objects.

-release Name

The release for which this command is being issued.

(Environment variable: TC_RELEASE)

-verbose

TeamConnection displays a confirmation message after you issue the command.

Examples

The following command displays information about a specified work area associated with the release set in the TC_RELEASE environment variable, type:

```
teamc workarea -view 8667
```

Related information

See the following related commands:

- Approval
- Collision
- Coreq
- Defect
- Driver
- Feature
- Fix
- Release
- Report
- Size
- Test

Workarea

Appendix. Querying the TeamConnection database

You can write queries against the TeamConnection database from the following interfaces:

- The **Edit Task List** window
- The **Query** field on the filter windows
- The TeamConnection command window
- An operating system command line prompt

Queries written in these environments issue a **report -view** command to query the TeamConnection database for information about TeamConnection objects. You specify select criteria for the query using the **-where** attribute. This appendix provides the names of views you can request reports for. These views represent tables in the TeamConnection database. The field names provided for each view represent column names. To query the database for parts whose base names begin with prt, for example, you write a query against the **baseName** column of the **PartView** table. The following is an example:

```
report -view PartView -release tcid20 -where "baseName like 'prt%'"
```

Constructing queries

You construct queries differently in different interfaces. The following shows how to construct the query in the previous example from each interface:

- **Edit Task List** window:
 1. Select the **Query** radio button
 2. Select **Parts** from the Query list box
 3. In the **Task** field, type **baseName like 'prt%'**
 4. In the **Release** field, type **tcid20**
- **Query** field on the Parts filter window:
 1. In the **Release** field, type **tcid20**
 2. In the **Query** field, type **baseName like 'prt%'**

- **Command Window** or command line prompt:

Type the following command:

```
report -view PartView -release tcid20 -where "baseName like 'prt%'"
```

"Views and report output" on page 262 lists all the field names that are in each TeamConnection view.

Rules for defining queries

TeamConnection queries follow the syntax of SQL queries, except for the addition of **report -view** at the beginning of the command. The following are basic rules for defining queries:

- View and field names are not case-sensitive. Any of the following retrieves information from the **baseName** field: **baseName**, **basename**, **BASENAME**.
- You can abbreviate view names to their shortest unique string. **AccessDownView**, for example, can be abbreviated to **AccessD**. You cannot abbreviate field names.
- When you specify values for date fields, use the format *yyyy/mm/dd hh:mm:ss*.

- When you search for specific field values, you must type the value of the field exactly as it exists in the database. The database values are case sensitive.
- Use keywords such as **and**, **or**, and **not** to enhance search conditions.
- Enclose values for character fields in single quotation marks.
- Use the following relational operators (also called *comparison operators*) to describe a relationship between two values:

Table 7. Relational operators

Operators	The search is based on the following:
=	Only one item you type.
<>	All items except for the one item you type.
>	An alphanumeric string greater than the one you type.
>=	An alphanumeric string greater than or equal to the one you type.
<	An alphanumeric string less than the one you type.
<=	An alphanumeric string less than or equal to the one you type.
in	Items that you type.
not in	All items except for those you type.
like	<p>The character string you type. You can use the following:</p> <ul style="list-style-type: none"> – The % wildcard character to match 0 or more characters – The _ wildcard character to match exactly one character. <p>If the string you are searching for contains an underscore or percent character, as in the file name prt_new.txt, for example, you can use an escape character, \$, to indicate that the underscore or percent character is to be taken literally. The following selects all part names starting with prt_:</p> <pre>report -view PartView -release tcid20 -where "baseName like 'prt\$_'"</pre>
between	The items falling between the two values you type.
is null	The items that have null values for the associated field.
is not null	The items that have values for the associated field.

Note: Notes * Selections in reports for date fields using <= or >= return the same information as if you entered < or >. This is because the date data type consists of a date and a time. Use the **like** operator with the < or > operators to return <= or >= information.

Views and report output

The remaining sections in this appendix list all the field names that are in each TeamConnection view. The following explains how this information is presented:

- View names that you can use with the report -view command are marked with a double asterisk symbol (**). View names without a double asterisk are available only as subselect criteria. CompMemberView, for example, can be used only as subselect criteria. You include subselect criteria by writing a nested select statement with your query. Nested select statements are included in the -where argument of the report -view commands or on the **Query** fields of the Edit Task List and filter windows.
- Field names that appear in italics indicate that you can use them as search criteria, but they do not provide data output.

- If you have defined configurable fields for defects, features, parts, or users, these fields are added to the end of the report output. They appear in the same order in which they are defined in the config.ld file.

If your family administrator configures the database fields, the field names in your database may differ from the field names in the following tables.

The actual size limit for many of the character attributes listed in these tables is smaller than the specified limit. For example, the length of the login field in the Users table is limited to 31 bytes, but you can actually use only 15 characters. The fields affected are usually related to names, such as userLogin and releaseName.

AccessDownView**

Access list entries including those of all child components

Field name (length)	Data type	Description
<i>compld</i>	integer	Database ID of the component where access is granted or restricted
<i>userId</i>	integer	Database ID of the user who owns the access authority
<i>authorityName</i> (31)	char	Access authority group name
<i>authorityType</i> (15)	char	Access authority type (granted or restricted)
<i>compName</i> (63)	char	Component name
<i>childCompName</i> (63)	char	Child component name
<i>userLogin</i> (31)	char	User's TeamConnection user ID
<i>userName</i> (63)	char	User's full name
<i>userArea</i> (31)	char	User's area or department

Raw output:

childCompName|*userLogin*|*userName*|*userArea*|*authorityName*|*authorityType*

AccessFastView

Low level actions granted to users at a component

Field name (length)	Data type	Description
<i>userId</i>	integer	Database ID of the user who owns the access authority
<i>action</i> (15)	char	TeamConnection action name
<i>authorityType</i> (15)	char	Access authority type (granted or restricted)
<i>compld</i>	integer	Database ID of the component where access is granted or restricted

AccessNInheritView

Low level actions restricted to users at a component

Field name (length)	Data type	Description
<i>userId</i>	integer	Database ID of the user who owns the access authority

Field name (length)	Data type	Description
<i>action</i> (15)	char	TeamConnection action name
<i>authorityType</i> (15)	char	Access authority type (granted or restricted)
<i>compld</i>	integer	Database ID of the component where access is granted or restricted

AccessTable

Component access list table

Field name (length)	Data type	Description
<i>compld</i>	integer	Database ID of the component where access is granted or restricted
<i>userId</i>	integer	Database ID of the user who owns the access authority
<i>authorityName</i> (31)	char	Access authority group name
<i>authorityType</i> (15)	char	Access authority type (granted or restricted)

AccessUpView

Access list entries including those of all parent components

Field name (length)	Data type	Description
<i>compld</i>	integer	Database ID of the component where access is granted or restricted
<i>userId</i>	integer	Database ID of the user who owns the access authority
<i>authorityName</i> (31)	char	Access authority group name
<i>authorityType</i> (15)	char	Access authority type (granted or restricted)
<i>compName</i> (63)	char	Component name
<i>parentName</i> (63)	char	Parent component name
<i>userLogin</i> (31)	char	User's TeamConnection user ID
<i>userName</i> (63)	char	User's full name
<i>userArea</i> (31)	char	User's area or department

AccessView**

Component access list entries

Field name (length)	Data type	Description
<i>compld</i>	integer	Database ID of the component where access is granted or restricted
<i>userId</i>	integer	Database ID of the user who owns the access authority
<i>authorityName</i> (31)	char	Access authority group name
<i>authorityType</i> (15)	char	Access authority type (granted or restricted)
<i>compName</i> (63)	char	Component name
<i>userLogin</i> (31)	char	User's TeamConnection user ID

Field name (length)	Data type	Description
userName (63)	char	User's full name
userArea (31)	char	User's area or department

Raw output:

compName|userLogin|userName|userArea|authorityName|authorityType

Approvals

Work area approval records table

Field name (length)	Data type	Description
<i>trackId</i>	integer	
<i>workAreaId</i>	integer	Database ID of the work area for the approval record
<i>userId</i>	integer	Database ID of the user who owns the approval record
<i>state</i> (15)	char	Approval record state
<i>addDate</i> (25)	char	Date created
<i>lastUpdate</i> (25)	char	Date of last update

ApprovalView**

Work area approval records

Field name (length)	Data type	Description
<i>trackId</i>	integer	
<i>workAreaId</i>	integer	Database ID of the work area for the approval record
<i>userId</i>	integer	Database ID of the user who owns the approval record
<i>state</i> (15)	char	Approval record state
<i>addDate</i> (25)	char	Date created
<i>lastUpdate</i> (25)	char	Date of last update
<i>releaseName</i> (31)	char	Release name
<i>defectPrefix</i> (31)	char	Defect or feature prefix
<i>defectName</i> (31)	char	Defect or feature identifier
<i>defectReference</i> (31)	char	Defect or feature reference
<i>defectAbstract</i> (127)	char	Defect or feature abstract
<i>defectType</i> (7)	char	Defect or feature
<i>userName</i> (63)	char	Approver's full name
<i>userLogin</i> (31)	char	Approver's TeamConnection user ID
<i>userArea</i> (31)	char	Approver's area or department
<i>workAreaName</i> (15)	char	Name of work area

Raw output:

workAreaName|defectPrefix|defectName|releaseName|userLogin|userName|userArea|state|addDate|lastUpdate|defectReference|defectAbstract|defectType

Approvers

Release approver list table

Field name (length)	Data type	Description
<i>releaseId</i>	integer	Database ID of the release
<i>userId</i>	integer	Database ID of the user

ApproverView**

Release approver list entries

Field name (length)	Data type	Description
<i>releaseId</i>	integer	Database ID of the release associated with the approver list
<i>userId</i>	integer	Database ID of the user who is the approver
releaseName (31)	char	Release name
userLogin (31)	char	Approver's TeamConnection user ID
userName (63)	char	Approver's full name
userArea (31)	char	Approver's area or department

Raw output:

releaseName|userLogin|userName|userArea

Authority**

Authority table

Field name (length)	Data type	Description
name (31)	char	Access authority group name
action (15)	char	Low-level action name

BchangeView**

Part changes (Tree view information)

Note: This view is only available in raw output.

Field name (length)	Data type	Description
workAreaName	char	
partType (7)	char	Type of the part
compName (63)	char	Component name of the part where the change is included
driverName (31)	char	Name of the driver where the change is included
<i>userId</i>	integer	Database ID of the user who made the change to the part
versionSID (47)	char	Version name of the changed part
pathName (195)	char	Part's full path name
releaseName (31)	char	Release name
defectType (7)	char	Defect or feature
<i>defectAbstract</i> (127)	char	Defect or feature abstract
<i>defectReference</i> (31)	char	Defect or feature reference

Field name (length)	Data type	Description
<i>defectName</i> (31)	char	Defect or feature identifier
<i>defectPrefix</i> (31)	char	Defect or feature prefix
<i>changeType</i> (8)	char	Type of part change
<i>driverId</i>	integer	Database ID of the driver where the change is included and committed
<i>compVersionId</i>	integer	
<i>versionId</i>	integer	Database ID of the part's version
<i>partId</i>	integer	Database ID of the part
<i>fileId</i>	integer	
<i>workAreaId</i>	integer	Database ID of the work area for this part change
<i>trackId</i>	integer	
<i>pathId</i>	integer	Database ID of the part's path name

Raw output:

workareaName|defectName|compName|pathName|versionSID|defectType|releaseName|
driverName|partType

BcompView**

Component properties

Note: This view is only available in raw output.

Field name (length)	Data type	Description
<i>id</i>	integer	Database ID of the component
<i>name</i> (63)	char	Component name
<i>userId</i>	integer	Database ID of the user who owns the component
<i>description</i> (127)	char	Component description
<i>addDate</i> (25)	char	Date created
<i>dropDate</i> (25)	char	Date deleted
<i>lastUpdate</i> (25)	char	Date of last update
<i>compProcess</i> (31)	char	Component process name
<i>featureDSR</i> (3)	char	Feature design, size, and review subprocess? (yes or no)
<i>featureVerify</i> (3)	char	Feature verify subprocess? (yes or no)
<i>defectDSR</i> (3)	char	Defect design, size, and review subprocess? (yes or no)
<i>defectVerify</i> (3)	char	Defect verify subprocess? (yes or no)
<i>userLogin</i> (31)	char	Component owner's TeamConnection user ID
<i>userName</i> (63)	char	Component owner's full name
<i>userArea</i> (31)	char	Component owner's area or department

Raw output:

hierarchy|name

BpartView**

Part properties

Note: This view is only available in raw output.

Field name (length)	Data type	Description
<i>id</i>	integer	
<i>fileId</i>	integer	
<i>partId</i>	integer	Database ID of the part
<i>workAreaId</i>	integer	Database ID of the work area
<i>builderId</i>	integer	
<i>parserId</i>	integer	
<i>nuPathId</i>	integer	
addDate (25)	char	Date created
dropDate (25)	char	Date deleted
nuAddDate (25)	char	New creation date
nuDropDate (25)	char	New deletion date
lastUpdate (25)	char	Date of last update
baseName (127)	char	Part base name
pathName (195)	char	Part path name
nuPathName (195)	char	Pending new part path name
releaseName (31)	char	Release name
<i>releaseId</i>	integer	
fileType (7)	char	Type of the file, as it exists outside TeamConnection (text or binary)
partType (7)	char	Type of the part
userLogin (31)	char	TeamConnection user ID who locked or checked out the part
fmode (4)	char	File mode
compName (63)	char	Component name
currentVersion (8)	char	Name of the current part version
committedVersion	char	Last committed part version name
workAreaName (14)	char	Name of current work area
changeType (8)	char	The type of change to the part. A value of bulk means the part's content has changed; attrib means the part's attributes have changed; and none means the part has not changed.
builderName (63)	char	Name of the part's builder
parserName (63)	char	Name of the part's parser
buildStatus (15)	char	Status of the build
parameters (1024)	char	Build parameters
<i>sourceId</i>	integer	Database ID of the part's source. Defines where the part was derived from and is used to identify common parts.
temporary (3)	char	Identifies parts that are deleted after a build, because they are no longer necessary.
workAreaChange (8)	char	The type of change to the part in the work area. A value of bulk means the part's content has changed; attributes means the part's attributes have changes; all means both the contents and attributes have changed; and none means the part has not changed.
<i>oid1</i>	integer	
<i>oid2</i>	integer	

Raw output:

```
hierarchy|baseName|releaseName|compName|committedVersion|addDate|dropDate|lastUpdate|
pathName|currentVersion|nuAddDate|nuDropDate|nuPathName|userLogin|fmode|
fileType|changeType|workAreaName|partType|temporary|builderName|
parserName|parentName|hasChildren|buildStatus|typeOfRelation|parameters|
workAreaChange|
```

Builders

Builder properties table

Field name (length)	Data type	Description
<i>id</i>	integer	Database ID of the builder
<i>sequenceId</i>	integer	
<i>releaseId</i>	integer	Database ID of the release associated with the builder
<i>name</i> (63)	char	Builder name
<i>parameters</i> (255)	char	Builder parameters
<i>rebuildTimeStamp</i> (25)	char	Time stamp of most recent update to this builder
<i>buildScript</i> (259)	char	Name you want the build script extracted as
<i>goodRCEvaluator</i> (259)	char	
<i>RCCCondition</i> (9)	char	Return code condition
<i>rcValue</i> (9)	char	Return code value
<i>bulkType</i> (8)	char	Type of data stored (binary, text, or none)
<i>bulkTypeNum</i> (8)	integer	
<i>environment</i> (15)	char	Environment this builder supports
<i>timeout</i> (9)	integer	Number of minutes allowed to build until timeout detected
<i>billOfMaterials</i>	integer	
<i>setupOptions</i>	char	

BuilderView**

Builder properties

Field name (length)	Data type	Description
<i>id</i>	integer	
<i>releaseId</i>	integer	Database ID of the release associated with the builder
<i>name</i> (63)	char	Builder name
<i>releaseName</i> (31)	char	Release name
<i>parameters</i> (255)	char	Builder parameters
<i>rebuildTimeStamp</i> (25)	char	Time stamp of most recent update to this builder
<i>buildScript</i> (259)	char	Name you want build script extracted as
<i>RCCCondition</i> (2)	char	Return code condition
<i>rcValue</i> (9)	char	Return code value
<i>bulkType</i> (8)	char	Type of data stored (binary, text, or none)
<i>bulkTypeNum</i>	integer	
<i>environment</i> (15)	char	Environment this builder supports

Field name (length)	Data type	Description
timeout (9)	integer	Number of minutes allowed to build until timeout detected
<i>billOfMaterials</i>	integer	
<i>setupOptions</i>	char	

Raw output:

name|releaseName|parameters|rebuildTimeStamp|buildScript|rcCondition|
bulkType|rcValue|environment|timeout

BuildView**

Part build properties

Field name (length)	Data type	Description
adName	char	
lastUpdateTimeStamp	ADObject	
buildStatus	char	
type	char	
parser	char	
lockedBy	char	
PartTimeStamp	ADObject	
adName1	char	
beTS	char	
builderRC	integer	
outputFileParms	char	
builder	char	
buildParms	char	
adTarget	BUILDVIEW	
PartTimeStamp1	ADObject	
includeFlag	integer	
adTarget1	BUILDVIEW	
adTarget2	BUILDVIEW	

Raw output:

Cfgcomproc**

Configurable process — component table

Field name (length)	Data type	Description
name (31)	char	Component process name
config (15)	char	Subprocess name

Raw output:

name|config

Cfgrelproc**

Configurable process — release table

Field name (length)	Data type	Description
name (31)	char	Release process name
config (15)	char	Subprocess name

Raw output:

name|config

ChangeExtractView

Part changes

Field name (length)	Data type	Description
<i>pathId</i>	integer	Database ID of the part's path name
<i>trackId</i>	integer	
<i>workAreaId</i>	integer	Database ID of the work area for this part change
<i>fileId</i>	integer	Database ID of the part's path name
<i>partId</i>	integer	Database ID of the part
<i>versionId</i>	integer	Database ID of the part's version
<i>compVersionId</i>	integer	
<i>driverId</i>	integer	Database ID of the driver where the change is included and committed
<i>changeType</i> (8)	char	Type of part change
<i>pathName</i> (195)	char	Part's full path name
<i>partSourceId</i>	integer	Database ID of the part's source. Defines where the part is derived from and is used to identify common parts
<i>versionSID</i> (47)	char	Version name of the changed part
<i>partType</i>	char	
<i>workAreaState</i>	char	

ChangeView**

Part changes, including defect and feature information

Field name (length)	Data type	Description
<i>pathId</i>	integer	Database ID of the part's path name
<i>trackId</i>	integer	
<i>workAreaId</i>	integer	Database ID of the work area for this part change
<i>fileId</i>	integer	
<i>partId</i>	integer	Database ID of the part
<i>versionId</i>	integer	Database ID of the part's version
<i>compVersionId</i>	integer	
<i>driverId</i>	integer	Database ID of the driver where the change is included and committed
<i>changeType</i> (8)	char	Type of part change
<i>defectPrefix</i> (31)	char	Defect or feature prefix

Field name (length)	Data type	Description
defectName (31)	char	Defect or feature identifier
defectReference (31)	char	Defect or feature reference
defectAbstract (127)	char	Defect or feature abstract
defectType (7)	char	Defect or feature
releaseName (31)	char	Release name
pathName (195)	char	Part's full path name
versionSID (47)	char	Version name of the changed part
userId	integer	Database ID of the user who made the change to the part
driverName (31)	char	Name of the driver where the change is included
partType (7)	char	Type of the part
committedVersion	char	
workAreaName (31)	char	Name of the current work area
workAreaState	char	

Raw output:

```
releaseName|workAreaName|defectName|driverName|versionSID|pathName|type|
partType|defectReference|defectAbstract|defectPrefix|userLogin|userName|
userArea
```

Changes

Part changes

Field name (length)	Data type	Description
pathId	integer	Database ID of the part's path name
trackId	integer	
workAreaId	integer	Database ID of the work area for this part change
fileId	integer	
partId	integer	Database ID of the part
versionId	integer	Database ID of the part's version SID
compVersionId	integer	
driverId	integer	Database ID of the driver where the change is included and converted
changeType (8)	char	Type of part change
partType (7)	char	Type of the part
userId	integer	Database ID of the user who made the part changes
workAreaState	char	

Collectors

Collectors table

Field name (length)	Data type	Description
id	integer	
oid1	integer	
oid2	integer	
typeName	char	

Collisions

Part collision records table

Field name (length)	Data type	Description
<i>fileId</i>	integer	Database ID of the part for this collision
<i>workAreaId</i>	integer	Database ID of the work area for this collision
<i>altWorkAreaId</i>	integer	
<i>partType</i> (7)	char	Type of the part
<i>state</i> (9)	char	State of the collision record
<i>addDate</i> (7)	char	Date created
<i>lastUpdate</i> (8)	char	Date of last update
<i>localSourceId</i>	integer	
<i>alternateSourceId</i>	integer	
<i>resolvedSourceId</i>	integer	

CollisionView**

Part collision records

Field name (length)	Data type	Description
<i>fileId</i>	integer	Database ID of the part for this collision
<i>workAreaId</i>	integer	Database ID of the work area for this collision
<i>altWorkAreaId</i>	integer	
<i>state</i> (9)	char	State of the collision record
<i>addDate</i> (25)	char	Date created
<i>lastUpdate</i> (25)	char	Date of last update
<i>localVersion</i> (31)	char	Version in current work area
<i>alternateVersion</i> (31)	char	Version that causes the collision
<i>workAreaName</i> (31)	char	Name of the work area
<i>releaseName</i> (31)	char	Name of the release
<i>relProcess</i> (31)	char	Name of the release process
<i>pathName</i> (195)	char	Part's full path name
<i>partType</i> (63)	char	Type of the part
<i>localSourceId</i>	integer	Source ID of the version in the current work area
<i>alternateSourceId</i>	integer	Source ID of the version that causes the collision
<i>resolvedSourceId</i>	integer	

Raw output:

```
pathName|workAreaName|releaseName|localVersion|alternateVersion|partType|
state|addDate|lastUpdate|relProcess
```

CompMembers

Component hierarchy table

Field name (length)	Data type	Description
<i>parentId</i>	integer	Database ID of the parent component

Field name (length)	Data type	Description
<i>childId</i>	integer	Database ID of the component
<i>rank</i>	integer	Number of links or generations between parent and child
<i>reference</i>	integer	Number of references to this parent-child-rank combination

CompMemberView

Component hierarchy linkages

Field name (length)	Data type	Description
<i>parentId</i>	integer	Database ID of the parent component
<i>childId</i>	integer	Database ID of the component
<i>rank</i>	integer	Number of links or generations between parent and child
<i>reference</i>	integer	Number of references to this parent-child-rank configuration
<i>parentCompName</i> (63)	char	Parent component name
<i>childCompName</i> (63)	char	Component name

Components

Component properties table

Field name (length)	Data type	Description
<i>id</i>	integer	Database ID of the component
<i>name</i> (63)	char	Component name
<i>userId</i>	integer	Database ID of the user who owns the component
<i>featureDSR</i> (3)	char	Feature design, size, and review subprocess? (yes or no)
<i>featureVerify</i> (3)	char	Feature verify subprocess? (yes or no)
<i>defectDSR</i> (3)	char	Defect design, size, and review subprocess? (yes or no)
<i>defectVerify</i> (3)	char	Defect verify subprocess? (yes or no)
<i>description</i> (127)	char	Component description
<i>addDate</i> (25)	char	Date created
<i>dropDate</i> (25)	char	Date deleted
<i>lastUpdate</i> (25)	char	Date of last update
<i>comProcess</i> (31)	char	Component process name

CompView**

Component properties

Field name (length)	Data type	Description
<i>id</i>	integer	Database ID of the component
<i>name</i> (63)	char	Component name
<i>userId</i>	integer	Database ID of the user who owns the component

Field name (length)	Data type	Description
featureDSR (3)	char	Feature design, size, and review subprocess? (yes or no)
featureVerify (3)	char	Feature verify subprocess? (yes or no)
defectDSR (3)	char	Defect design, size, and review subprocess? (yes or no)
defectVerify (3)	char	Defect verify subprocess? (yes or no)
description (127)	char	Component description
addDate (25)	char	Date created
dropDate (25)	char	Date deleted
lastUpdate (25)	char	Date of last update
compProcess (31)	char	Component process name
userLogin (31)	char	Component owner's TeamConnection user ID
userName (63)	char	Component owner's full name
userArea (31)	char	Component owner's area or department

Raw output:

```
name|userLogin|userName|userArea|addDate|dropDate|lastUpdate|description|
compProcess|featureDSR|featureVerify|defectDSR|defectVerify
```

Config**

Configurable field types table

Field name(length)	Data type	Description
configType (15)	char	Name of the configurable field type
name (31)	char	Value for the configurable field type
dflt (3)	char	Is this entry the default value? (yes or no)
value1	integer	(not currently used)
value2	integer	(not currently used)
description (127)	char	Description of the value

Raw Output

```
configType|name|dflt|value1|value2|description
```

Coreqs

Corequisites table

Field name (length)	Data type	Description
<i>trackId</i>	integer	
<i>workAreaId</i>	integer	Database ID of the work area
<i>groupId</i>	integer	Database ID of the corequisite group to which the work area belongs

CoreqView**

Coreq properties

Field name (length)	Data type	Description
<i>trackId</i>	integer	Database ID of the work area

Field name (length)	Data type	Description
<i>workAreaId</i>	integer	Database ID of the work area
groupID (10)	char	ID of a group of corequisite work areas
<i>releaseId</i>	integer	
<i>defectId</i>	integer	
name	char	
state (15)	char	The state of the corequisite work area
defectPrefix	char	
defectName (15)	char	Defect name for the work area
releaseName (15)	char	The name of the release

Raw output:

groupID|workareaName|defectName|defectType|releaseName|state

DefectDownView**

Defect properties, including those of child components

Field name (length)	Data type	Description
target	char	
priority	char	
phaseInject	char	
phaseFound	char	
symptom	char	
<i>id</i>	integer	Database ID of the defect
<i>defectType</i> (7)	char	Defect
prefix (31)	char	Defect prefix
name (31)	char	Defect identifier
<i>compId</i>	integer	Database ID of the component associated with the defect
<i>releaseId</i>	integer	Database ID of the release where discovered
envName (31)	char	Environment where discovered
state (15)	char	Defect state
severity (31)	char	Severity level
abstract (127)	char	Defect abstract
reference (31)	char	Defect reference
answer (31)	char	Accept or return answer type
driverName (31)	char	Name of driver where discovered
lastUpdate (25)	char	Date of last update
addDate (25)	char	Date created
assignDate (25)	char	Date when reassigned
responseDate (25)	char	Date accepted or returned
endDate (25)	char	Date closed or canceled
<i>originId</i>	integer	Database ID of the defect originator
<i>ownerId</i>	integer	Database ID of the defect owner
age	integer	Age of the defect (dependent on the family's aging mechanism)
duplicate (31)	char	Duplicate defect or feature identifier
releaseName (31)	char	Name of release where discovered
<i>compName</i> (63)	char	Component name

Field name (length)	Data type	Description
childCompName (63)	char	Child component name
originLogin (31)	char	Defect originator's TeamConnection user ID
originName (63)	char	Defect originator's full name
originArea (31)	char	Defect originator's area or department
ownerLogin (31)	char	Defect owner's TeamConnection user ID
ownerName (63)	char	Defect owner's full name
ownerArea (31)	char	Defect owner's area or department

Raw output:

```
prefix|name|childCompName|releaseName|ownerLogin|state|answer|severity|
abstract|age|envName|driverName|duplicate|lastUpdate|addDate|assignDate|
responseDate|endDate|ownerName|ownerArea|reference|originLogin|originName|
originArea
```

Defects

Defect properties table

(This table is also used to record features.)

Field name (length)	Data type	Description
<i>id</i>	integer	Database ID of the defect or feature
<i>defectType</i> (7)	char	defect or feature
<i>prefix</i> (31)	char	Defect or feature prefix
<i>name</i> (31)	char	Defect or feature identifier
<i>compld</i>	integer	Database ID of the component associated with the defect or feature
<i>releaseld</i>	integer	Database ID of the release associated with the defect or feature
<i>envName</i> (31)	char	Environment where discovered (when type is defect, blank otherwise)
<i>state</i> (15)	char	Defect or feature state
<i>severity</i> (31)	char	Severity level (when type is defect)
<i>abstract</i> (127)	char	Defect or feature abstract
<i>reference</i> (31)	char	Defect or feature reference
<i>answer</i> (31)	char	Accept or return answer type
<i>driverName</i> (31)	char	Name of driver where discovered (when type is defect)
<i>lastUpdate</i> (25)	char	Date of last update
<i>addDate</i> (25)	char	Date created
<i>assignDate</i> (25)	char	Date when reassigned
<i>responseDate</i> (25)	char	Date accepted or returned
<i>endDate</i> (25)	char	Date closed or canceled
<i>originId</i>	integer	Database ID of the user who opened the defect or feature
<i>ownerId</i>	integer	Database ID of the defect or feature owner
<i>age</i>	integer	Age of defect or feature (dependent on the aging mechanism of the family)
<i>duplicate</i> (31)	char	Defect or feature for which this defect or feature is a duplicate

DefectView**

Defect properties

Field name (length)	Data type	Description
target	char	
priority	char	
phaseInject	char	
phaseFound	char	
symptom	char	
<i>id</i>	integer	Database ID of the defect
<i>defectType</i> (7)	char	Defect
prefix (31)	char	Defect prefix
name (31)	char	Defect identifier
<i>compId</i>	integer	Database ID of the component associated with the defect
<i>releaseId</i>	integer	Database ID of the release where discovered
envName (31)	char	Environment where discovered
state (15)	char	Defect state
severity (31)	char	Severity level
abstract (127)	char	Defect abstract
reference (31)	char	Defect reference
answer (31)	char	Accept or return answer type
driverName (31)	char	Name of driver where discovered
lastUpdate (25)	char	Date of last update
addDate (25)	char	Date created
assignDate (25)	char	Date when reassigned
responseDate (25)	char	Date accepted or returned
endDate (25)	char	Date closed or canceled
<i>originId</i>	integer	Database ID of the defect originator
<i>ownerId</i>	integer	Database ID of the defect owner
age	integer	Age of the defect (dependent on the family's aging mechanism)
duplicate (31)	char	Duplicate defect or feature identifier
releaseName (31)	char	Name of release where discovered
<i>compName</i> (63)	char	Component name
originLogin (31)	char	Defect originator's TeamConnection user ID
originName (63)	char	Defect originator's full name
originArea (31)	char	Defect originator's area or department
ownerLogin (31)	char	Defect owner's TeamConnection user ID
ownerName (63)	char	Defect owner's full name
ownerArea (31)	char	Defect owner's area or department

Raw output:

```
prefix|name|compName|releaseName|ownerLogin|state|answer|severity|abstract|age|
envName|driverName|duplicate|lastUpdate|addDate|assignDate|responseDate|
endDate|ownerName|ownerArea|reference|originLogin|originName|originArea
```

DriverMembers

Driver members table

Field name (length)	Data type	Description
<i>driverId</i>	integer	Database ID of the driver
<i>trackId</i>	integer	
<i>workAreaId</i>	integer	Database ID of the work area
<i>committedVersion</i>	char	Name of the committed version for the work area
<i>driverState</i>	char	

DriverMemberView**

Driver members

Field name (length)	Data type	Description	
<i>driverId</i>	integer	Database ID of the driver	
<i>trackId</i>	integer	Name of the committed version for the work area	
<i>workAreaId</i>	integer	Database ID of the work area	
<i>driverName</i> (31)	1	char	Driver name
<i>driverState</i>	char		
<i>releaseName</i> (31)	char	Release name	
<i>workAreaName</i> (15)	char	Name of the current work area	
<i>trackUserLogin</i> (31)	char	Work area owner's TeamConnection User ID	
<i>trackUserName</i> (63)	char	Work area owner's full name	
<i>trackUserArea</i> (31)	char	Work area owner's area or department	
<i>defectPrefix</i> (31)	char	Defect or feature prefix	
<i>defectName</i> (31)	char	Defect or feature identifier	
<i>defectReference</i> (31)	char	Defect or feature reference	
<i>defectAbstract</i> (127)	char	Defect or feature abstract	
<i>defectType</i> (7)	char	Defect or feature	
<i>committedVersion</i> (1)	char	Name of the committed version for the work area	
<i>workAreaVersion</i> (1)	char	The name of the latest version of the work area added to the driver. If the workarea was in the integrate state when added to the driver, this field is the name of the latest version in the work area.	

Raw output:

```
driverName|releaseName|workAreaName|defectName|defectReference|
workAreaUserLogin|workAreaUserName|workAreaUserArea|defectPrefix|
defectType|committedVersion|workAreaVersion
```

Drivers

Driver properties table

Field name (length)	Data type	Description
<i>id</i>	integer	Database ID of the driver
<i>name</i> (31)	char	Driver name
<i>releaseId</i>	integer	Database ID of the release associated with the driver
<i>userId</i>	integer	Database ID of the user who owns the driver
<i>addDate</i> (25)	char	Date created
<i>commitDate</i> (25)	char	Date committed
<i>lastUpdate</i> (25)	char	Date of last update
<i>state</i> (15)	char	Driver state
<i>driverType</i> (31)	char	Driver type

DriverView**

Driver properties

Field name (length)	Data type	Description
<i>id</i>	integer	Database ID of the driver
<i>name</i> (31)	char	Driver name
<i>releaseId</i>	integer	Database ID of the release
<i>userId</i>	integer	Database ID of the user who owns the driver
<i>addDate</i> (25)	char	Date created
<i>commitDate</i> (25)	char	Date committed
<i>lastUpdate</i> (25)	char	Date of last update
<i>state</i> (15)	char	Driver state
<i>driverType</i> (31)	char	Driver type
<i>releaseName</i> (31)	char	Release name
<i>userLogin</i> (31)	char	Driver owner's TeamConnection user ID
<i>userName</i> (63)	char	Driver owner's full name
<i>userArea</i> (31)	char	Driver owner's area or department

Raw output:

```
name|releaseName|type|userLogin|userName|userArea|addDate|commitDate|
lastUpdate|state
```

Environments

Release environment list table

Field name (length)	Data type	Description
<i>name</i> (31)	char	Environment name
<i>releaseId</i>	integer	Database ID of the release
<i>userId</i>	integer	Database ID of the user

EnvView**

Release environment list entries

Field name (length)	Data type	Description
name (31)	char	Environment name
<i>releaseId</i>	integer	Database ID of the release
<i>userId</i>	integer	Database ID of the user
releaseName (31)	char	Release name
userLogin (31)	char	Tester's TeamConnection user ID
userName (63)	char	Tester's full name
userArea (31)	char	Tester's area or department

Raw output:

name|releaseName|userLogin|userName|userArea

FeatureDownView**

Feature properties, including those of child components

Note: FeatureDownView and DefectView share the same field names. Some of the field names are used only in DefectView, so these fields are listed as not applicable (N/A) for FeatureDownView in the following table.

Field name (length)	Data type	Description
target	char	
priority	char	
id	integer	Database ID of the feature
<i>defectType</i> (7)	char	Feature
prefix (31)	char	Feature prefix
name (31)	char	Feature identifier
<i>compId</i>	integer	Database ID of the component associated with the feature
<i>releaseId</i>	integer	Database ID of the release where discovered
<i>envName</i> (31)	char	N/A
state (15)	char	Feature state
<i>severity</i> (31)	char	N/A
abstract (127)	char	Feature abstract
reference (31)	char	Feature reference
<i>answer</i> (31)	char	N/A
<i>driverName</i> (31)	char	N/A
lastUpdate (25)	char	Date of last update
addDate (25)	char	Date created
assignDate (25)	char	Date when reassigned
responseDate (25)	char	Date accepted or returned
endDate (25)	char	Date closed or canceled
<i>originId</i>	integer	Database ID of the feature originator
<i>ownerId</i>	integer	Database ID of the feature owner
age	integer	Age of the feature (dependent on the family's aging mechanism)
duplicate (31)	char	Duplicate defect or feature identifier

Field name (length)	Data type	Description
<i>releaseName</i> (31)	char	N/A
<i>compName</i> (63)	char	N/A
<i>childCompName</i> (63)	char	Child component name
<i>originLogin</i> (31)	char	Feature originator's TeamConnection user ID
<i>originName</i> (63)	char	Feature originator's full name
<i>originArea</i> (31)	char	Feature originator's area or department
<i>ownerLogin</i> (31)	char	Feature owner's TeamConnection user ID
<i>ownerName</i> (63)	char	Feature owner's full name
<i>ownerArea</i> (31)	char	Feature owner's area or department

Raw output:

```
prefix|name|childCompName|ownerLogin|ownerName|state|abstract|age|duplicate|
lastUpdate|addDate|assignDate|responseDate|endDate|ownerArea|reference|
originLogin|originName|originArea
```

FeatureView**

Feature properties

Note: FeatureView and DefectView share the same field names. Some of the field names are used only in DefectView, so these fields are listed as not applicable (N/A) for FeatureView in the following table.

Field name (length)	Data type	Description
<i>target</i>	char	
<i>priority</i>	char	
<i>id</i>	integer	Database ID of the feature
<i>defectType</i> (7)	char	Feature
<i>prefix</i> (31)	char	Feature prefix
<i>name</i> (31)	char	Feature identifier
<i>compld</i>	integer	Database ID of the component associated with the feature
<i>releaseld</i>	integer	Database ID of the release where discovered
<i>envName</i> (31)	char	N/A
<i>state</i> (15)	char	Feature state
<i>severity</i> (31)	char	N/A
<i>abstract</i> (127)	char	Feature abstract
<i>reference</i> (31)	char	Feature reference
<i>answer</i> (31)	char	N/A
<i>driverName</i> (31)	char	N/A
<i>lastUpdate</i> (25)	char	Date of last update
<i>addDate</i> (25)	char	Date created
<i>assignDate</i> (25)	char	Date when reassigned
<i>responseDate</i> (25)	char	Date accepted or returned
<i>endDate</i> (25)	char	Date closed or canceled
<i>originId</i>	integer	Database ID of the feature originator
<i>ownerId</i>	integer	Database ID of the feature owner
<i>age</i>	integer	Age of the feature (dependent on the family's aging mechanism)

Field name (length)	Data type	Description
<i>duplicate</i> (31)	char	Duplicate defect or feature identifier
<i>releaseName</i> (31)	char	N/A
<i>compName</i> (63)	char	N/A
<i>originLogin</i> (31)	char	Feature originator's TeamConnection user ID
<i>originName</i> (63)	char	Feature originator's full name
<i>originArea</i> (31)	char	Feature originator's area or department
<i>ownerLogin</i> (31)	char	Feature owner's TeamConnection user ID
<i>ownerName</i> (63)	char	Feature owner's full name
<i>ownerArea</i> (31)	char	Feature owner's area or department

Raw output:

```
prefix|name|compName|ownerLogin|ownerName|state|abstract|age|duplicate|
lastUpdate|addDate|assignDate|responseDate|endDate|ownerArea|reference|
originLogin|originName|originArea
```

Files**

Part properties

Field name (length)	Data type	Description
<i>id</i>	integer	
<i>partId</i>	integer	Database ID of the part
<i>compId</i>	integer	Database ID of the component
<i>pathId</i>	integer	
<i>addDate</i> (25)	char	Date created
<i>dropDate</i> (25)	char	Date deleted
<i>lastUpdate</i> (25)	char	Date of last update
<i>fmode</i> (4)	char	File mode
<i>partType</i> (7)	char	Type of the part
<i>versionId</i>	integer	
<i>releaseId</i>	integer	

Raw output:

```
need raw output
```

Fix

Fix records table

Field name (length)	Data type	Description
<i>workAreaId</i>	integer	Database ID of the work area
<i>userId</i>	integer	Database ID of the user who owns the fix record
<i>compId</i>	integer	Database ID of the component associated with the fix record
<i>state</i> (15)	char	Fix record state
<i>addDate</i> (25)	char	Date created
<i>lastUpdate</i> (25)	char	Date of last update

FixDownView

Fix records, including those for child components

Field name (length)	Data type	Description
<i>trackId</i>	integer	Database ID of the work area
<i>workAreaId</i>	integer	Database ID of the work area
<i>userId</i>	integer	Database ID of the user who owns the fix record
<i>compId</i>	integer	Database ID of the component associated with the fix record
<i>state</i> (15)	char	Fix record state
<i>addDate</i> (25)	char	Date created
<i>lastUpdate</i> (25)	char	Date of last update
<i>releaseName</i> (31)	char	Release name
<i>defectPrefix</i> (31)	char	Defect or feature prefix
<i>defectName</i> (31)	char	Defect or feature identifier
<i>defectAbstract</i> (127)	char	Defect or feature abstract
<i>defectReference</i> (31)	char	Defect or feature reference
<i>defectType</i> (7)	char	Defect or feature
<i>compName</i> (63)	char	Component name
<i>childCompName</i> (63)	char	Child component name
<i>userLogin</i> (31)	char	Fix record owner's TeamConnection user ID
<i>userName</i> (63)	char	Fix record owner's full name
<i>userArea</i> (31)	char	Fix record owner's area or department

FixView**

Fix records

Field name (length)	Data type	Description
<i>trackId</i>	integer	Database ID of the work area
<i>workAreaId</i>	integer	Database ID of the work area
<i>userId</i>	integer	Database ID of the user who owns the fix record
<i>compId</i>	integer	Database ID of the component associated with the fix record
<i>state</i> (15)	char	Fix record state
<i>addDate</i> (25)	char	Date created
<i>lastUpdate</i> (25)	char	Date of last update
<i>releaseName</i> (31)	char	Release name
<i>defectPrefix</i> (31)	char	Defect or feature prefix
<i>defectName</i> (31)	char	Defect or feature identifier
<i>defectAbstract</i> (127)	char	Defect or feature abstract
<i>defectReference</i> (31)	char	Defect or feature reference
<i>defectType</i> (7)	char	Defect or feature
<i>compName</i> (63)	char	Component name
<i>userLogin</i> (31)	char	Fix record owner's TeamConnection user ID
<i>userName</i> (63)	char	Fix record owner's full name
<i>userArea</i> (31)	char	Fix record owner's area or department
<i>workAreaName</i> (31)	char	

Raw output:

workAreaName|releaseName|compName|state|userLogin|userArea|defectAbstract|addDate|
lastUpdate|userName|defectPrefix|defectName|defectType|defectReference

History

Defect or feature state change history table

Field name (length)	Data type	Description
<i>defectId</i>	integer	Database ID of the defect or feature
<i>userId</i>	integer	Database ID of the user who caused the defect or feature state change
<i>action</i> (15)	char	Action that occurred and caused the state change
<i>addDate</i> (25)	char	Date action occurred

HistoryView

Defect or feature state change history

Field name (length)	Data type	Description
<i>defectId</i>	integer	Database ID of the defect or feature
<i>userId</i>	integer	Database ID of the user who caused the defect or feature state change
<i>action</i> (15)	char	Action that occurred and caused the state change
<i>addDate</i> (25)	char	Date action occurred
<i>defectPrefix</i> (31)	char	Defect or feature prefix
<i>defectName</i> (31)	char	Defect or feature identifier
<i>defectReference</i> (31)	char	Defect or feature reference
<i>userLogin</i> (31)	char	TeamConnection user ID of the user performing the actions to the defect or feature
<i>userName</i> (63)	char	Full name of the user performing the actions to the defect or feature
<i>userArea</i> (31)	char	Area or department of the user performing the actions to the defect or feature

Hosts

User host list table

Field name (length)	Data type	Description
<i>userId</i>	integer	Database ID of the user
<i>name</i> (127)	char	Client host name
<i>login</i> (31)	char	User's login name on client host
<i>loginType</i> (31)	char	

HostView**

User host list entries

Field name (length)	Data type	Description
<i>userId</i>	integer	Database ID of the user
<i>name</i> (127)	char	Name of the client host
<i>login</i> (31)	char	User's login name on the client host
<i>loginType</i> (31)	char	User's login name on client host
<i>userLogin</i> (31)	char	User's TeamConnection user ID
<i>userName</i> (63)	char	User's full name
<i>userArea</i> (31)	char	User's area or department

Raw output:

`login|name|userLogin|userName|userArea`

Interest**

Interest table

Field name (length)	Data type	Description
<i>name</i> (31)	char	Interest group name
<i>action</i> (15)	char	Low-level action name

Raw output

`name|action`

Notes

Defect or feature notes table

Field name (length)	Data type	Description
<i>defectId</i>	integer	Database ID of the defect or feature
<i>userId</i>	integer	Database ID of the user who added remarks
<i>action</i> (15)	char	Action occurring when remarks were added
<i>remarks</i> (29999)	long	Text of remarks
<i>addDate</i> (25)	char	Date remarks were added

NoteView**

Defect or feature notes

Field name (length)	Data type	Description
<i>defectId</i>	integer	Database ID of the defect or feature
<i>userId</i>	integer	Database ID of the user who added remarks
<i>action</i> (15)	char	Action occurring when remarks were added
<i>remarks</i> (29999)	char	Text of remarks
<i>addDate</i> (25)	char	Date created
<i>defectPrefix</i> (31)	char	Defect or feature prefix
<i>defectName</i> (31)	char	Defect or feature identifier
<i>defectReference</i> (31)	char	Defect or feature reference

Field name (length)	Data type	Description
userLogin (31)	char	User's TeamConnection user ID
userName (63)	char	User's full name
userArea (31)	char	User's area or department

Raw output:

```
defectName|defectReference|action|addDate|userLogin|userName|userArea|
defectPrefix|remarks
```

Notification

Notification list table

Field name (length)	Data type	Description
<i>compId</i>	integer	Database ID of the component where notification is controlled
<i>userId</i>	integer	Database ID of the user who owns the notification interest
<i>interestName</i> (31)	char	Interest group name

NotifyDownView**

Notification list members, including descendant members

Field name (length)	Data type	Description
<i>compId</i>	integer	Database ID of the component where notification is controlled
<i>userId</i>	integer	Database ID of the user who owns the notification interest
<i>interestName</i> (31)	char	Interest group name
<i>compName</i> (63)	char	Component name
<i>childCompName</i> (63)	char	Child component name
<i>userAddress</i> (159)	char	User's sendmail address
<i>userLogin</i> (31)	char	User's TeamConnection user ID
<i>userName</i> (63)	char	User's full name
<i>userArea</i> (31)	char	User's area or department

Raw output:

```
childCompName|userLogin|userName|userArea|userAddress|interestName
```

NotifyUpView**

Notification list entries, including those inherited from parents

Field name (length)	Data type	Description
<i>compId</i>	integer	Database ID of the component where notification is controlled
<i>userId</i>	integer	Database ID of the user who owns the notification interest
<i>interestName</i> (31)	char	Interest group name
<i>compName</i> (63)	char	Component name
<i>parentName</i> (63)	char	Parent component name

Field name (length)	Data type	Description
userAddress (159)	char	User's sendmail address
userLogin (31)	char	User's TeamConnection user ID
userName (63)	char	User's full name
userArea (31)	char	User's area or department

Raw output:

parentName|userLogin|userName|userArea|userAddress|interestName

NotifyView**

Notification list entries

Field name (length)	Data type	Description
<i>compId</i>	integer	Database ID of the component where notification is controlled
<i>userId</i>	integer	Database ID of the user who owns the notification interest
interestName (31)	char	Interest group name
compName (63)	char	Component name
userAddress (159)	char	User's sendmail address
userLogin (31)	char	User's TeamConnection user ID
userName (63)	char	User's full name
userArea (31)	char	User's area or department

Raw output:

compName|userLogin|userName|userArea|userAddress|interestName

OVersions

Version record properties table

Field name (length)	Data type	Description
<i>releaseId</i>	integer	Database ID of the release
<i>workAreaId</i>	integer	Database ID of the work area
name (31)	char	Name of the version
addDate (25)	char	Date created
freezeDate (25)	char	Date frozen
releaseVersion (31)	char	Version in the release
hasSuccessor (3)	char	Does this version have a successor? (yes or no)
hasOutputs (3)	char	Does the version have outputs? (yes or no)
sidSequence	integer	
remarks	char	

Parsers

Parser properties table

Field name (length)	Data type	Description
<i>id</i>	integer	Database ID of the parser

Field name (length)	Data type	Description
<i>sequenceId</i>	integer	
<i>releaseId</i>	integer	Database ID of the release associated with the parser
<i>name</i> (63)	char	Parser name
<i>commandName</i> (259)	char	Parser command that will be used
<i>includePaths</i> (255)	char	Include paths for parser

ParserView**

Parser properties

Field name (length)	Data type	Description
<i>id</i>	integer	Database ID of the parser
<i>sequenceId</i>	integer	
<i>releaseId</i>	integer	Database ID of the release associated with the parser
<i>name</i> (63)	char	Parser name
<i>releaseName</i> (31)	char	Release name
<i>commandName</i> (259)	char	Parser command that will be used
<i>includePaths</i> (255)	char	Include paths for parser

Raw output:

name|releaseName|commandName|includePaths

PartFullView**

Part properties

Field name (length)	Data type	Description
<i>id</i>	integer	
<i>partId</i>	integer	Database ID of the part
<i>fileId</i>	integer	
<i>workAreaId</i>	integer	Database ID of the work area
<i>nuPathId</i>	integer	Database ID of the path name
<i>addDate</i> (25)	char	Date created
<i>dropDate</i> (25)	char	Date deleted
<i>nuAddDate</i> (25)	char	New creation date
<i>nuDropDate</i> (25)	char	New deletion date
<i>lastUpdate</i> (25)	char	Date of last update
<i>baseName</i> (127)	char	Part's base name
<i>pathName</i> (195)	char	Part's path name
<i>nuPathName</i> (195)	char	Pending new part path name
<i>releaseName</i> (31)	char	Name of the release
<i>releaseId</i>	integer	
<i>fileType</i> (7)	char	text or binary
<i>partType</i> (7)	char	Type of the part
<i>userLogin</i> (31)	char	TeamConnection user ID who locked or checked out the part
<i>fmode</i> (4)	char	File mode
<i>compName</i> (63)	char	Component name

Field name (length)	Data type	Description
currentVersion (8)	char	Name of the current part version
committedVersion	char	Last committed part version name
workAreaName (14)	char	Name of the current work area
changeType (8)	char	The type of change to the part. A value of bulk means the part's content has changed; attrib means the part's attributes have changed; and none means the part has not changed.
<i>sourceId</i>	integer	Source ID used to identify common parts
<i>temporary</i>	char	
<i>oid1</i>	integer	
<i>oid2</i>	integer	

Raw output:

```
baseName|releaseName|compName|committedVersion|addDate|dropDate|
lastUpdate|pathName|currentVersion|nuAddDate|nuDropDate|nuPathName|
userLogin|fmode|type|workAreaLock|workAreaName|partType
```

Parts

Part properties table

Field name (length)	Data type	Description
<i>id</i>	integer	Database ID of the part
<i>oid1</i>	integer	
<i>oid2</i>	integer	
<i>partType</i> (7)	char	Type of the part

PartsOut

Parts currently locked for editing table

Field name (length)	Data type	Description
<i>partId</i>	integer	Database ID of the part
<i>versionId</i>	integer	Database ID of the part's version
<i>userId</i>	integer	Database ID of the user who has the part locked
<i>seqPartId</i>	integer	
<i>workAreaId</i>	integer	Database ID of the work area
<i>checkedOut</i> (3)	char	Is the part checked out? (yes or no)
<i>checkOutDate</i> (25)	char	Date part was locked
<i>newSID</i> (47)	char	Part's pending version name on check in
<i>versionModified</i> (3)	char	Has the current version been modified? (yes or no)

PartsOutView**

Parts currently locked for editing

Field name (length)	Data type	Description
<i>fileId</i>	integer	Database ID of the part

Field name (length)	Data type	Description
<i>partId</i>	integer	Database ID of the part
<i>seqPartId</i>	integer	Database ID of the part
<i>seqFileId</i>	integer	Database ID of the part
<i>versionId</i>	integer	Database ID of the part's version
<i>userId</i>	integer	Database ID of the user who has the part locked
<i>workAreaId</i>	integer	Database ID of the work area
checkedOut (3)	char	Checked out? (yes or no)
versionModified (3)	char	Version modified? (yes or no)
checkOutDate (25)	char	Date part was locked
newSID (47)	char	New part version ID
partNuPath (195)	char	Part path name
releaseName (31)	char	Release name
userLogin (31)	char	User's TeamConnection user ID
userName (63)	char	User's full name
userArea (31)	char	User's area or department
workAreaName (31)	char	Work area name
<i>compld</i>	integer	

Raw output:

```
partNuPath|releaseName|checkOutDate|newSID|userLogin|userName|userArea|
workAreaName|checkedOut|versionModified
```

PartView**

Part properties

Field name (length)	Data type	Description
<i>id</i>	integer	
<i>fileId</i>	integer	
<i>partId</i>	integer	Database ID of the part
<i>workAreaId</i>	integer	Database ID of the work area
<i>builderId</i>	integer	
<i>paserId</i>	integer	
<i>nuPathId</i>	integer	
addDate (25)	char	Date created
dropDate (25)	char	Date deleted
nuAddDate (25)	char	New creation date
nuDropDate (25)	char	New deletion date
lastUpdate (25)	char	Date of last update
baseName (127)	char	Part base name
pathName (195)	char	Part path name
nuPathName (195)	char	Pending new part path name
releaseName (31)	char	Release name
<i>releasId</i>	integer	
fileType (7)	char	Type of the file, as it exists outside TeamConnection (text or binary)
partType (7)	char	Type of the part
userLogin (31)	char	TeamConnection user ID who locked or checked out the part
fmode (4)	char	File mode

Field name (length)	Data type	Description
compName (63)	char	Component name
currentVersion (8)	char	Name of the current part version
committedVersion	char	Last committed part version name
workAreaName (14)	char	Name of current work area
changeType (8)	char	The type of change to the part. A value of bulk means the part's content has changed; attrib means the part's attributes have changed; and none means the part has not changed.
builderName (63)	char	Name of the part's builder
parserName (63)	char	Name of the part's parser
buildStatus (15)	char	Status of the build
parameters (1024)	char	Build parameters
sourceId	integer	Database ID of the part's source. Defines where the part was derived from and is used to identify common parts.
temporary (3)	char	Identifies parts that are deleted after a build, because they are no longer necessary.
workAreaChange (8)	char	The type of change to the part in the work area. A value of bulk means the part's content has changed; attributes means the part's attributes have changes; all means both the contents and attributes have changed; and none means the part has not changed.
oid1	integer	
oid2	integer	

Raw output:

```
baseName|releaseName|compName|committedVersion|addDate|dropDate|lastUpdate|
pathName|currentVersion|nuAddDate|nuDropDate|nuPathName|userLogin|fmode|
fileType|changeType|workAreaName|partType|temporary|builderName|
parserName|parentName|hasChildren|buildStatus|typeOfRelation|parameters|
workAreaChange|
```

Path

Part path names table

Field name (length)	Data type	Description
id	integer	Database ID of the part's path name
name (195)	char	Full path name of the part

Prereqs

Prerequisites table

Field name (length)	Data type	Description
trackId	integer	
workAreaId	integer	Database ID of the work area
prereqId	integer	Database ID of the prerequisite group to which the work area belongs

Field name (length)	Data type	Description
<i>prereqWorkAreald</i>	integer	

PrereqView**

Prereq properties

Field name (length)	Data type	Description
<i>trackId</i>	integer	
<i>workAreald</i>	integer	Database ID of the work area
<i>prereqId</i>	integer	Database ID of the prereq
<i>prereqWorkAreald</i>	integer	
<i>releaseld</i>	integer	Database ID of the release
<i>defectId</i>	integer	Database ID of the defect associated with the work area
<i>prereqDefectId</i>	integer	Database ID of the defect associated with the prereq
name (15)	char	
state (15)	char	The state of the prerequisite work area
defectPrefix	char	
defectName (15)	char	Defect name for the work area
defectType (7)	char	Defect type for the work area
prereqName (15)	char	The name of the prerequisite work area
prereqPrefix	char	
prereqDefect (15)	char	Defect name for the prerequisite work area
prereqType (7)	char	Defect type for the prerequisite work area
releaseName (15)	char	The name of the release

Raw output:

```
workarea|defectName|defectType|releaseName|prereqName|prereqDefect|
prereqType|state
```

Releases

Release table

Field name (length)	Data type	Description
<i>id</i>	integer	Database ID of the release
<i>name</i> (31)	char	Release name
<i>compld</i>	integer	Database ID of the component associated with the release
<i>binding</i> (3)	char	Binding status
<i>approve</i> (3)	char	Approval subprocess? (yes or no)
<i>fix</i> (3)	char	Fix subprocess? (yes or no)
<i>lvl</i> (3)	char	Driver subprocess? (yes or no)
<i>test</i> (3)	char	Test subprocess? (yes or no)
<i>description</i> (127)	char	Release description
<i>userId</i>	integer	Database ID of the user who owns the release
<i>addDate</i> (25)	char	Date created
<i>dropDate</i> (25)	char	Date deleted
<i>lastUpdate</i> (25)	char	Date of last update

Field name (length)	Data type	Description
<i>track</i> (3)	char	Track subprocess? (yes or no)
<i>relProcess</i> (31)	char	Release process name
<i>developmentMode</i> (10)	char	Development mode of the release? (concurrent or serial)
<i>autoPrune</i> (3)	char	Autopruning on? (yes or no)
<i>outputVersions</i> (16)	char	Number of committed output versions saved in the release
<i>dbName</i>	char	

ReleaseView**

Release properties

Field name (length)	Data type	Description
<i>id</i>	integer	Database ID of the release
<i>name</i> (31)	char	Release name
<i>compId</i>	integer	Database ID of the component associated with the release
<i>binding</i> (3)	char	Binding status
<i>approve</i> (3)	char	Approval subprocess? (yes or no)
<i>fix</i> (3)	char	Fix subprocess? (yes or no)
<i>lvl</i> (3)	char	Driver subprocess? (yes or no)
<i>test</i> (3)	char	Test subprocess? (yes or no)
<i>description</i> (127)	char	Release description
<i>userId</i>	integer	Database ID of the user who owns the release
<i>addDate</i> (25)	char	Date created
<i>dropDate</i> (25)	char	Date deleted
<i>lastUpdate</i> (25)	char	Date of last update
<i>track</i> (3)	char	Track subprocess? (yes or no)
<i>relProcess</i> (31)	char	Release process name
<i>compName</i> (63)	char	Component name
<i>userLogin</i> (31)	char	Release owner's TeamConnection user ID
<i>userName</i> (63)	char	Release owner's full name
<i>userArea</i> (31)	char	Release owner's area or department
<i>developmentMode</i> (10)	char	Development mode of the release? (concurrent or serial)
<i>autoPrune</i> (3)	char	Autopruning on? (yes or no)
<i>outputVersions</i> (16)	char	Number of committed output versions saved in the release
<i>dbName</i> (63)	char	Name of the remote database for the release

Raw output:

```
name|compName|relProcess|userLogin|userName|userArea|addDate|dropDate|
lastUpdate|description|track|approve|fix|lvl|test|developmentMode|
autoPrune|outputVersions|dbName
```

Sequence

Sequences

Field name (length)	Data type	Description
<i>name</i> (15)	char	Sequence type
<i>lastSerial</i>	integer	Database ID of the last sequence number assigned for the sequence type

Sizes

Sizing records table

Field name (length)	Data type	Description
<i>defectId</i>	integer	Database ID of the defect or feature
<i>compld</i>	integer	Database ID of the component associated with the sizing record
<i>userId</i>	integer	Database ID of the user who owns the sizing record
<i>addDate</i> (25)	char	Date created
<i>lastUpdate</i> (25)	char	Date of last update
<i>state</i> (7)	char	Sizing record state
<i>sizing</i> (127)	char	Text of sizing information
<i>releaseId</i>	integer	Database ID of the release associated with the sizing record

SizeView**

Sizing records

Field name (length)	Data type	Description
<i>defectId</i>	integer	Database ID of the feature or defect
<i>compld</i>	integer	Database ID of the component associated with the sizing record
<i>userId</i>	integer	Database ID of the user who owns the sizing record
<i>addDate</i> (25)	char	Date created
<i>lastUpdate</i> (25)	char	Date of last update
<i>state</i> (7)	char	Sizing record state
<i>sizing</i> (127)	char	Text of sizing information
<i>releaseId</i>	integer	Database ID of the release associated with the sizing record
<i>featurePrefix</i> (31)	char	Feature or defect prefix
<i>featureName</i> (31)	char	Feature or defect identifier
<i>featureAbstract</i> (127)	char	Feature or defect abstract
<i>featureReference</i> (31)	char	Feature or defect reference
<i>compName</i> (63)	char	Component name
<i>userLogin</i> (31)	char	Sizing record owner's TeamConnection user ID
<i>userName</i> (63)	char	Sizing record owner's full name
<i>userArea</i> (31)	char	Sizing record owner's area or department
<i>releaseName</i> (31)	char	Release name

Raw output:

```
featureName|featureReference|compName|releaseName|sizing|addDate|state|
userName|userLogin|userArea|lastUpdate|featurePrefix|featureAbstract
```

Tests

Environment test records table

Field name (length)	Data type	Description
<i>trackId</i>	integer	
<i>workAreaId</i>	integer	Database ID of the work area
<i>envName</i> (31)	char	Environment name
<i>userId</i>	integer	Database ID of the user who owns the test record
<i>state</i> (15)	char	Environment test record state
<i>addDate</i> (25)	char	Date created
<i>lastUpdate</i> (25)	char	Date of last update

TestView**

Environment test records

Field name (length)	Data type	Description
<i>trackId</i>	integer	Database ID of the work area
<i>workAreaId</i>	integer	Database ID of the work area
<i>envName</i> (31)	char	Environment name
<i>userId</i>	integer	Database ID of the user who owns the test record
<i>state</i> (15)	char	Environment test record state
<i>addDate</i> (25)	char	Date created
<i>lastUpdate</i> (25)	char	Date of last update
<i>defectPrefix</i> (31)	char	Defect or feature prefix
<i>defectName</i> (31)	char	Defect or feature identifier
<i>defectReference</i> (31)	char	Defect or feature reference
<i>defectAbstract</i> (127)	char	Defect or feature abstract
<i>defectType</i> (7)	char	Defect or feature
<i>userLogin</i> (31)	char	Test record owner's TeamConnection user ID
<i>userName</i> (63)	char	Test record owner's full name
<i>userArea</i> (31)	char	Test record owner's area or department
<i>releaseName</i> (31)	char	Release name
<i>workAreaName</i> (15)	char	Name of the current work area

Raw output:

```
workAreaName|releaseName|defectPrefix|defectName|envName|state|addDate|
lastUpdate|userLogin|defectAbstract|userName|userArea|defectReference
```

Tracks

Tracks properties table

Field name (length)	Data type	Description
<i>id</i>	integer	Database ID of the track
<i>releaseId</i>	integer	Database ID of the release associated with the track
<i>defectId</i>	integer	Database ID of the defect or feature
<i>userId</i>	integer	Database ID of the user who owns the track
<i>workAreaId</i>	integer	
<i>state</i> (15)	char	Track state
<i>target</i> (31)	char	Target for completion (such as a driver or a date)
<i>actual</i> (31)	char	Driver commit date where track included (actual driver name)
<i>addDate</i> (25)	char	Date created
<i>lastUpdate</i> (25)	char	Date of last update
<i>name</i>	character	

Users**

Users table

Field name (length)	Data type	Description
<i>id</i>	integer	Database ID of the user
<i>login</i> (31)	char	TeamConnection ID
<i>name</i> (63)	char	User's full name
<i>superUser</i> (3)	char	Superuser privilege? (yes or no)
<i>area</i> (31)	char	User's area or department
<i>address</i> (159)	char	User's sendmail address
<i>addressType</i>	char	
<i>addDate</i> (25)	char	Date created
<i>dropDate</i> (25)	char	Date deleted
<i>lastUpdate</i> (25)	char	Date of last update
<i>AuthType</i>	char	
<i>pswStatus</i>	char	
<i>pswModifyTime</i>	char	
<i>pswCreateTime</i>	char	
<i>pswFailCount</i>	integer	
<i>pswBlock</i>	ADObject_sequence	

Raw output:

login|*name*|*area*|*address*|*addDate*|*dropDate*|*lastUpdate*|*superUser*

Verify

Verification record properties table

Field name (length)	Data type	Description
<i>defectId</i>	integer	Database ID of the defect or feature

Field name (length)	Data type	Description
<i>userId</i>	integer	Database ID of the user who owns the verification record
<i>type</i> (15)	char	original or duplicate
<i>state</i> (15)	char	Verification record state
<i>addDate</i> (25)	char	Date created
<i>lastUpdate</i> (25)	char	Date of last update

VerifyView**

Verification record properties

Field name (length)	Data type	Description
<i>defectId</i>	integer	Database ID of the defect or feature
<i>userId</i>	integer	Database ID of the user who owns the verification record
<i>verifyType</i> (15)	char	Original or duplicate
<i>state</i> (15)	char	Verification record state
<i>addDate</i> (25)	char	Date created
<i>lastUpdate</i> (25)	char	Date of last update
<i>defectPrefix</i> (31)	char	Defect or feature prefix
<i>defectName</i> (31)	char	Defect or feature identifier
<i>defectReference</i> (31)	char	Defect or feature reference
<i>defectAbstract</i> (127)	char	Defect or feature abstract
<i>userLogin</i> (31)	char	Verification record owner's TeamConnection user ID
<i>userName</i> (63)	char	Verification record owner's full name
<i>userArea</i> (31)	char	Verification record owner's area or department
<i>compName</i> (63)	char	Name of component associated with the defect or feature
<i>defectType</i> (7)	char	Defect or feature

Raw output:

```
defectName|state|addDate|userLogin|userArea|type|userName|defectAbstract|
lastUpdate|defectPrefix|defectReference|compName
```

Versions

Version record properties table

Field name (length)	Data type	Description
<i>id</i>	integer	Database ID of the part version
<i>previousId</i>	integer	Database ID of the part's previous version
<i>partId</i>	integer	
<i>nuPathId</i>	integer	
<i>nuAddDate</i> (25)	char	New creation date for the part
<i>nuDropDate</i> (25)	char	New delete date for the part
<i>changeDate</i> (25)	char	Date the version was made
<i>SID</i> (47)	char	Part version name
<i>releaseName</i>	char	
<i>changeType</i>	char	

Field name (length)	Data type	Description
<i>workAreaChange</i>	char	
<i>sourceId</i>	integer	Database ID of the part's source that is used to identify common parts

VersionView**

Database version properties

Field name (length)	Data type	Description
<i>releaseId</i>	integer	Database ID of the release
<i>workAreaId</i>	integer	Database ID of the work area
<i>name</i> (31)	char	Name of the version
<i>addDate</i> (25)	char	Date created
<i>freezeDate</i> (25)	char	Date frozen
<i>releaseVersion</i> (31)	char	Version in the release
<i>hasSuccessor</i> (3)	char	Does this version have a successor? (yes or no)
<i>releaseName</i> (31)	char	Release name
<i>workAreaName</i> (31)	char	Work area name associated with version
<i>predecessor</i> (31)	char	Previous version
<i>hasOutputs</i> (3)	char	Does the version have outputs? (yes or no)
<i>sidSequence</i>	integer	
<i>remarks</i>	char	

Raw output:

```
name|workAreaName|releaseName|predecessor|hasSuccessor|releaseVersion|
hasOutputs|addDate|freezeDate
```

Workareas

Work areas properties table

Field name (length)	Data type	Description
<i>id</i>	integer	Database ID of the work area
<i>releaseId</i>	integer	Database ID of the part's previous version
<i>trackId</i>	integer	
<i>name</i> (15)	char	Work area name
<i>branchPoint</i> (47)	char	The branch point of the work area

WorkAreaView**

Work area properties

Field name (length)	Data type	Description
<i>id</i>	integer	Database ID of the work area
<i>trackId</i>	integer	Database ID of the associated track
<i>releaseId</i>	integer	Database ID of the release associated with the work area
<i>defectId</i>	integer	Database ID of the defect or feature
<i>userId</i>	integer	Database ID of the user who owns the work area

Field name (length)	Data type	Description
<i>workAreaId</i>	integer	Database ID of the associated work area
name (15)	char	Name of the current work area
state (15)	char	Work area state
target (31)	char	Target for completion
actual (31)	char	Name of driver where work area is committed
addDate (25)	char	Date created
lastUpdate (25)	char	Date of last update
defectPrefix (31)	char	Defect or feature prefix
defectName (31)	char	Defect or feature identifier
defectReference (31)	char	Defect or feature reference
defectAbstract (127)	char	Defect or feature abstract
defectType (7)	char	Defect or feature
releaseName (31)	char	Release name
userLogin (31)	char	Work area owner's TeamConnection user ID
userName (63)	char	Work area owner's full name
userArea (31)	char	Work area owner's area or department
branch point (47)	char	Branch point of the work area

Raw output:

```
name|releaseName|defectName|defectReference|state|target|addDate|userLogin|
  userName|userArea|actual|lastUpdate|defectPrefix|defectAbstract|defectType|
  branchPoint
```

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.
- **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**
 - 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

DB/2

The following publications are part of the IBM DB/2 Universal Database library of documents for DB/2 administration.

- *Administration Getting Started* (S10J-8154-00)
An introductory guide to basic administration tasks and the DB/2 administration tools.
- *SQL Getting Started* (S10J-8156-00)
Discusses basic concepts of DB/2 SQL.
- *Administration Guide* (S10J-8157-00)
A complete guide to administration tasks and the DB/2 administration tools.
- *SQL Reference* (S10J-8165-00)
A reference to DB/2 SQL for programmers and database administrators.
- *Troubleshooting Guide* (S10J-8169-00)
A guide to identifying and solving problems with DB/2 servers and clients and to using the DB/2 diagnostic tools.
- *Messages Reference* (S10J-8168-00)
Provides detailed information about DB/2 messages..
- *Command Reference* (S10J-8166-00)
Provides information about DB/2 system commands and the command line processor.
- *Replication Guide* (S10J-0999-00)
Describes how to plan, configure, administer, and operate IBM replication tools available with DB/2.
- *System Monitor Guide and Reference* (S10J-8164-00)
Describes how to monitor DB/2 database activity and analyze system performance.
- *Glossary*

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

Special Characters

-undo 136

A

access
 description 15
 related information 19
 syntax 15
action flag
 description 2
approval
 description 21, 29
 related information 27, 31
 syntax 21, 29
attribute flag
 description 3
 null 4
authority
 base 11
 explicit 12
 implicit 12
 requirements 11
 restricted 12
 superuser 11

B

build script 33
build tree 131
builder
 description 33
 related information 40
 syntax 33

C

collision
 description 41
 related information 46
 syntax 41
component
 description 47
 related information 54
 syntax 47
concurrent development mode 41, 136
configurable fields
 defects 63, 65, 69
 features 103, 105, 107
 parts 150, 164
 users 227, 228, 230
coreq
 description 55
 related information 57
 syntax 55

D

defect
 description 59
 related information 76
 syntax 59
directory permissions 4, 84, 159, 187
driver
 description 77
 related information 89
 syntax 77
driver member
 description 91
 related information 94
 syntax 91

E

environment
 description 95
 related information 98
 syntax 95
environment variable 6
 setting 11

F

feature
 description 99
 related information 114
 syntax 99
file permissions 4, 85, 160, 163
fix
 description 115
 related information 121
 syntax 115
flag
 action 2
 arguments 3
 attribute 3
 description 2

H

host
 description 123
 related information 126
 syntax 123

L

LANG 6

N

NLSPATH 6
notify
 description 127

notify (*continued*)
 examples 129
 syntax 127
null keyword 4

O

octal number 4
OS_NETWORK 6
OS_ROOTDIR 6
OS_TMPDIR 6

P

parser
 description 131
 related information 134
 syntax 131
part
 building 135
 collector object 135
 common 139
 deleting 136
 description 135
 destroying 136
 extracting 135
 linking 136
 merging 41
 modifying 135
 place-holder part 135
 related information 176
 specifying full path name 136
 syntax 140
 TC_TOP 136
PATH 6
prereq
 description 177
 related information 179
 syntax 177

R

release
 description 181
 related information 193
 syntax 182
report
 description 195
 related information 206
 syntax 195

S

serial development mode 41, 136
size
 description 207
 related information 213
 syntax 207
standard input 6
syntax statements
 description 12

T

TC_BECOME 6
TC_BUILD_AGENTS_FILE 6
TC_BUILD_NOKEY 6
TC_BUILD_PROCESSORS_FILE 6
TC_BUILDPOOL 6
TC_BULKCHUNKSIZE 6
TC_CACHEPRUNEMETHOD 6
TC_CACHESIZELIMIT 6
TC_CASESENSE 6
TC_COMPONENT 6
TC_DBPATH 6
TC_FAMILY 6, 11
TC_MAKEIMPORTRULES 6
TC_MAKEIMPORTTOP 6
TC_MAKEIMPORTVERBOSE 6
TC_MIGRATERULES 6
TC_NOTIFY_DAEMON 6
TC_RELEASE 6
TC_SYSTEM_LOG 6
TC_TMP 6
TC_TOP 6, 136
TC_TRACE 6
TC_TRACEATTEMPTS 6
TC_TRACEDELAY 6
TC_TRACEFILE 6
TC_TRACESIZE 6
TC_USER 6
TC_WORKAREA 6
tlogin
 description 215
 related information 217
 syntax 215
teamagt 2
TeamConnection Merge tool 41
teamcpak 2
teamproc 2
test
 description 219
 related information 226
 syntax 219
TMP 6

U

undo 136
user
 description 227
 related information 233
 syntax 227

V

verify
 description 235
 related information 239
 syntax 235

W

work area
 corequisite 241

work area (*continued*)
description 241
freeze 241
prerequisite 241
refresh 241
related information 259
syntax 242



Part Number: 33H2571



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

SC34-4501-03



33H2571

