



CLI Help



*IBM Rational Synergy*

*CLI Help*

*Release 7.1*

---

Before using this information, be sure to read the general information under [Notices](#) on page 644.

This edition applies to Version 7.1, Rational Synergy (product number 5724V66) and to all subsequent releases and modifications until otherwise indicated in new editions.

© Copyright IBM Corporation 1992, 2009

US Government Users Restricted Rights—Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

---

# Contents

<b>General usage information</b>	<b>6</b>
IBM Rational Synergy Readme .....	7
Using the CLI - Windows users .....	8
Using the CLI - UNIX users .....	9
Rational Synergy Interfaces .....	10
Terminology and name changes .....	12
Rational Synergy CLI Help .....	13
Contacting IBM Rational Software Support .....	15
Command and argument syntax .....	18
Global formatting options .....	40
Naming restrictions .....	55
Case and file name limit database options .....	58
Built-In keywords .....	59
Regular expressions .....	62
Administering purposes and templates .....	64
<b>Default settings</b>	<b>65</b>
How defaults are set .....	66
Default options .....	68
Initialization file - Windows .....	96
Initialization file - UNIX .....	96
Environment variables .....	97
Setting model object attribute options .....	99
Creating a list box for a new attribute .....	101
Setting object type attribute options .....	102
Setting options in the system or personal ini file .....	103
Setting options using the ccm set command .....	104
<b>Commands</b>	<b>105</b>
alias command .....	106
attribute command .....	111
baseline command .....	121

---

bom command . . . . .	148
candidates command . . . . .	151
cat command . . . . .	155
change_type command . . . . .	158
checkin command . . . . .	161
checkout command . . . . .	169
checkpoint command . . . . .	179
cmdhistory command . . . . .	185
conflicts command . . . . .	190
copy_project command . . . . .	196
copy_to_file_system command . . . . .	203
create command . . . . .	206
dcm command . . . . .	215
delete command . . . . .	286
delimiter command . . . . .	292
dir command . . . . .	295
edit command . . . . .	301
finduse command . . . . .	304
folder command . . . . .	318
folder_template command . . . . .	344
groups command . . . . .	365
history command . . . . .	370
In command . . . . .	375
ls command . . . . .	379
merge command . . . . .	384
move command . . . . .	388
process command . . . . .	394
process_rule command . . . . .	410
project Command . . . . .	432
project_grouping command . . . . .	436
project_purpose command . . . . .	467
properties command . . . . .	474
query command . . . . .	479
reconcile command . . . . .	485
relate command . . . . .	496

---

release command . . . . .	501
save_offline_and_delete command . . . . .	520
soad_scope command . . . . .	530
set command . . . . .	543
show command . . . . .	548
show_servers command . . . . .	554
start command . . . . .	557
stop command . . . . .	562
sync command . . . . .	565
task command . . . . .	569
unalias command . . . . .	599
undo_update command . . . . .	602
unrelate command . . . . .	607
unset command . . . . .	610
unuse command . . . . .	613
update command . . . . .	620
use command . . . . .	625
view command . . . . .	631
work_area command . . . . .	634

<b>Links to Rational Synergy Help</b>	<b>642</b>
---------------------------------------	------------

<b>Notices</b>	<b>644</b>
----------------	------------

## General usage information

This section describes how to use IBM® Rational® Synergy. The following topics are discussed:

- [IBM Rational Synergy Readme](#)
  - [Using the CLI - Windows users](#)
  - [Using the CLI - UNIX users](#)
  - [Rational Synergy Interfaces](#)
  - [Terminology and name changes](#)
  - [Rational Synergy CLI Help](#)
  - [Command and argument syntax](#)
  - [Global formatting options](#)
  - [Naming restrictions](#)
  - [Case and file name limit database options](#)
  - [Built-In keywords](#)
  - [Regular expressions](#)
  - [Administering purposes and templates](#)
- [Notices](#)



---

## IBM Rational Synergy Readme

Be sure to read the latest [Readme](#) prior to using or administering Rational Synergy. The *Rational Synergy Readme* contains general information about the product release, and includes the following topics:

- System Requirements
- Compatibility with Other Rational Products and Releases
- New Release Features
- Notification of Future Changes

## Using the CLI - Windows users

Rational Synergy supports the command line interface (CLI) under all supported Windows platforms.

You can execute any Rational Synergy command from the Windows command prompt.

Note that you cannot start the command line interface from the Rational Synergy interface. You must start the CLI separately.

### **Option delimiter**

By default, the Windows client supports the slash ( / ) option delimiter. The dash ( - ) option delimiter is also supported. Examples in this help are shown using the dash ( - ) option delimiter.

### **Universal naming convention**

Use the universal naming convention (UNC) any time you enter a path to an administrative command. UNC makes network access to files, machines, and other devices easier. Use UNC to refer to remote machines and files by using a particular format. The format is:

`\\computer_name\share_name\path.`

In the following example, `\\loon\ccmdb\tstgonzo` is a UNC-style path.

```
> ccm message /d \\loon\ccmdb\tstgonzo "Server going down for repair."
```

All Rational Synergy commands accept **both** UNC paths and paths with drive letters (for example, `c:\users\ccmdb\base`). However, three commands, `ccmdb create`, `ccmdb copy`, and `ccmdb unpack` require UNC paths for the database to be created.

### **File paths**

The Windows client supports the standard Windows file specification, which usually is written as:

`drive:\directory\filename`

Rational Synergy Help uses the following to represent file paths:

`c:\directory\filename`

Although your files might reside on a different drive, Rational Synergy Help uses drive `c:` for consistency.

### **Location of CCM\_HOME**

`CCM_HOME` is the directory where the Rational Synergy product was installed. For example, if you want to edit the `remexec.cfg` file, which resides in the `etc` directory in the Rational Synergy installation area, you will need to change directory to `CCM_HOME\etc`.

The default install directory for a client installation is:

`C:\Program Files\IBM\Rational\Synergy\7.1.`

---

## Using the CLI - UNIX users

Rational Synergy supports the command line interface (CLI) under all supported UNIX platforms.

You can run any Rational Synergy command from the UNIX shell.

Note that you cannot start the command line interface from the Rational Synergy interface. You must start the CLI separately.

### ***Option delimiter***

By default, the UNIX client supports the dash ( - ) option delimiter.

### ***Location of CCM\_HOME***

`CCM_HOME` is the directory where the Rational Synergy product was installed. For example, if you want to edit the `remexec.cfg` file, which resides in the `etc` directory in the Rational Synergy installation area, you will need to change directory to `$CCM_HOME/etc`.

## Rational Synergy Interfaces

Rational Synergy provides the following interfaces:

- Synergy GUI

This interface provides full functionality for developers and build managers. It does not support administrative operations, but you can use it for non-administrative use. You can run the Synergy GUI in two modes: Web mode or Traditional mode. These modes are described [Web mode and Traditional mode](#) below.

- Synergy CLI

This interface provides near full functionality for developers and build managers. It does not support all administrative operations, and runs in Web mode only. Web Mode is described in [Web mode and Traditional mode](#) below.

- Synergy Classic GUI

This interface is available primarily for administrative operations. Users who work with the Classic GUI should plan to switch to the Synergy GUI. The Classic GUI is not available in Web mode. This interface will be phased out in a future release.

- Synergy Classic CLI

This interface is available primarily for administrative operations, and to provide a transition period for converting existing scripts to use the Synergy CLI. The Classic CLI is not available in Web mode. This interface will be phased out in a future release.

### ***Web mode and Traditional mode***

Synergy 7.0 introduces a new, faster way of working called Web mode. Web mode uses a new underlying architecture for communication between the client and server. It is intended primarily for use across a wide area network (WAN), but can be used on a local area network (LAN) as well. Your Synergy administrator will provide information about which mode you should use.

---

Web mode and Traditional mode differ in the following ways:

	<b>Traditional mode</b>	<b>Web mode</b>
<b>Performance</b>	Same as previous releases	Much faster, especially over a WAN
<b>Synergy GUI</b>	Same as previous releases	Available in Web mode
<b>Classic GUI</b>	Same as 6.5	Not available in Web mode
<b>Synergy CLI</b>	Not available in Traditional mode	In Release 7.0., there is limited support for administrative commands.
<b>Classic CLI</b>	Same as previous releases	Not available in Web mode
<b>ActiveCM</b>	Same as previous releases	Not available in Web mode
<b>Administration</b>	Same as 6.5	Same as 6.5, plus Synergy server configuration and TDS (see below)
<b>Installation</b>	Same as previous releases	No different from Traditional mode
<b>Network protocol</b>	Proprietary (RFC)	HTTP or HTTPS
<b>User authentication</b>	Operating system (OS)	Rational Directory® Server™ (RDS), (LDAP)

Eventually Traditional mode will be phased out, but it is supported now because it offers capabilities not yet available through the GUI or CLI in Web mode. You must use Traditional mode for most administration functions, such as changing delimiters, adding or modifying type definitions, performing an upgrade, database backups and integrity checks, and migrating data using the migration utility.

## Terminology and name changes

Effective with the 7.0 release, the Rational Synergy interface can be run in two modes: Web mode and Traditional mode. The other graphical interface is still referred to as Rational Synergy Classic. The corresponding command interface is called Rational Synergy Classic CLI. ActiveCM is not supported.

Terms used in the Rational Synergy Classic CLI and the Rational Synergy Classic GUI interfaces have not changed. The following table shows the terms used in the Classic interfaces and the corresponding terms in the Synergy interfaces:

<b>Classic GUI and CLI</b>	<b>Synergy GUI and CLI</b>
Reconfigure/Update Members	Update
Reconfigure Template	Process Rule
Reconfigure Properties	Update Properties
Undo Reconfigure	Undo Update
Check Out (Project)	Copy Project
Work Area Snapshot	Copy to File System
Default Task	Current Task

---

## Rational Synergy CLI Help

You are currently viewing Help updated for the command line interface (CLI) running Rational Synergy CLI in Web mode for Release 7.1. The information in this Help system is the most current available for Release 7.1. Always check the [Readme](#) for the latest changes to the information contained here.

### ***Rational Synergy Classic GUI and CLI, and Help***

If you use the Rational Synergy Classic GUI, you might read accompanying CLI information in the CLI part of the Help system that is part of that interface. The CLI Help that displays as part of the same Help system as the GUI Help is not the most current command line information; it was last updated for release 6.3. For the latest commands and options available while running the Rational Synergy CLI, Traditional mode, start Help from `CCM_HOME\jetty\webapps\help_cli` (Windows) and `$CCM_HOME/jetty/webapps/help_cli` (UNIX). The top-level file is called `synergy_cli.html`.

For the latest commands and options available while running the Rational Synergy CLI, Web mode, start Help from `CCM_HOME\jetty\webapps\help_cli_web` (Windows) and `$CCM_HOME/jetty/webapps/help_cli_web` (UNIX). The top-level file is called `synergy_web_cli.html`.

If you are not sure which Help system you are viewing, check the footer on the bottom of each HTML page, which identifies the product and release.

### ***Rational Synergy Classic CLI, Traditional mode and Help***

If you're running Rational Synergy Classic CLI, Traditional mode, the Help system included has undergone minimal updates since the 6.5 release. Very few changes have been made to the documented commands.

### ***Rational Synergy CLI, Web mode and Help***

If you're running Rational Synergy CLI, the Help system included has undergone major updates. Aside from a reorganization of the subcommand information, several commands are not yet supported in this release, but will be in a future release. The Rational Synergy CLI supports most developer and build manager functions, but you will need to use the Rational Synergy Classic CLI for some administrative commands.

The `ccm reconfigure_properties` command was retired and is no longer supported.

The following list shows the commands not supported when running Rational Synergy CLI in Web mode for Release 7.1.

- `ccm archive_fix` (used by IBM® Rational® Software Support)
- `ccm clean_cache`
- `ccm cleanup`
- `ccm collapse`

- `ccm dcm (-init)`
- `ccm db_update`
- `ccm delimiter` (changing version delimiter)
- `ccm depend`
- `ccm expand`
- `ccm export`
- `ccm fs_check`
- `ccm groups (-create, -list)`
- `ccm import`
- `ccm migrate`
- `ccm release (-delimiter, -rename)`
- `ccm resync`
- `ccm show -mar`
- `ccm source`
- `ccm typedef`
- `ccm update_properties`
- `ccm users`
- `ccm work_area (-find, -dbpath)`
- `ccm win_fixup` (used by IBM Rational Software Support)



---

## Contacting IBM Rational Software Support

If the self-help resources have not provided a resolution to your problem, you can contact IBM Rational Software Support for assistance in resolving product issues.

**Note** If you are a heritage Telelogic customer, a single reference site for all support resources is located at <http://www.ibm.com/software/rational/support/telelogic/>

### Prerequisites

To submit your problem to IBM Rational Software Support, you must have an active Passport Advantage® software maintenance agreement. Passport Advantage is the IBM comprehensive software licensing and software maintenance (product upgrades and technical support) offering. You can enroll online in Passport Advantage from <http://www.ibm.com/software/lotus/passportadvantage/howtoenroll.html>

- To learn more about Passport Advantage, visit the Passport Advantage FAQs at [http://www.ibm.com/software/lotus/passportadvantage/brochures\\_faqs\\_quickguides.html](http://www.ibm.com/software/lotus/passportadvantage/brochures_faqs_quickguides.html).
- For further assistance, contact your IBM representative.

To submit your problem online (from the IBM Web site) to IBM Rational Software Support, you must additionally:

- Be a registered user on the IBM Rational Software Support Web site. For details about registering, go to <http://www.ibm.com/software/support/>.
- Be listed as an authorized caller in the service request tool.

### Submitting problems

To submit your problem to IBM Rational Software Support:

1. Determine the business impact of your problem. When you report a problem to IBM, you are asked to supply a severity level. Therefore, you need to understand and assess the business impact of the problem that you are reporting.

Use the following table to determine the severity level.

Severity	Description
1	The problem has a <i>critical</i> business impact: You are unable to use the program, resulting in a critical impact on operations. This condition requires an immediate solution.
2	This problem has a <i>significant</i> business impact: The program is usable, but it is severely limited.

Severity	Description
3	The problem has <i>some</i> business impact: The program is usable, but less significant features (not critical to operations) are unavailable.
4	The problem has <i>minimal</i> business impact: The problem causes little impact on operations or a reasonable circumvention to the problem was implemented.

2. Describe your problem and gather background information, When describing a problem to IBM, be as specific as possible. Include all relevant background information so that IBM Rational Software Support specialists can help you solve the problem efficiently. To save time, know the answers to these questions:
  - What software versions were you running when the problem occurred?  
To determine the exact product name and version, use the option applicable to you:
    - \* Start the IBM Installation Manager and select **File > View Installed Packages**. Expand a package group and select a package to see the package name and version number.
    - \* Start your product, and click **Help > About** to see the offering name and version number.
  - What is your operating system and version number (including any service packs or patches)?
  - Do you have logs, traces, and messages that are related to the problem symptoms?
  - Can you recreate the problem? If so, what steps do you perform to recreate the problem?
  - Did you make any changes to the system? For example, did you make changes to the hardware, operating system, networking software, or other system components?
  - Are you currently using a workaround for the problem? If so, be prepared to describe the workaround when you report the problem.
3. Submit your problem to IBM Rational Software Support. You can submit your problem to IBM Rational Software Support in the following ways:
  - **Online:** Go to the IBM Rational Software Support Web site at <https://www.ibm.com/software/rational/support/> and in the Rational support task navigator, click **Open Service Request**. Select the electronic problem reporting tool, and open a Problem Management Record (PMR), describing the problem accurately in your own words.

---

For more information about opening a service request, go to <http://www.ibm.com/software/support/help.html>

You can also open an online service request using the IBM Support Assistant. For more information, go to <http://www.ibm.com/software/support/isa/faq.html>.

- **By phone:** For the phone number to call in your country or region, go to the IBM directory of worldwide contacts at <http://www.ibm.com/planetwide/> and click the name of your country or geographic region.
- **Through your IBM Representative:** If you cannot access IBM Rational Software Support online or by phone, contact your IBM Representative. If necessary, your IBM Representative can open a service request for you. You can find complete contact information for each country at <http://www.ibm.com/planetwide/>.

## Command and argument syntax

You can enter the commands to run Rational Synergy as follows:

- The `ccm` command prefix precedes each user command. Enter user commands individually, such as:

```
ccm dir
```

- On Windows and UNIX, some administrative commands use the `ccmdb` and `ccmsrv` prefixes.

The Windows administrative commands are discussed in the [Rational Synergy Administration Guide for Windows](#). The UNIX administrative commands are described in the documents below

- On UNIX, some administrative commands use the `ccm_` prefix. Enter these administrative commands individually, such as:

```
ccm_install
```

The UNIX administrative commands are discussed in the [Rational Synergy Administration Guide for UNIX](#). Users running on an Oracle database should use [Rational Synergy Administration Guide for UNIX \(Oracle\)](#).

Many commands accept specifications for one or more objects in a Synergy database. The most common specification is a [File specification](#), which specifies a project, directory, or file in the database.

The following specifications are described in this section:

- [Baseline specification](#)
- [Change request specification](#)
- [Database specification](#)
- [File specification](#)
- [Folder specification](#)
- [Folder template specification](#)
- [Object specification](#)
- [Process specification](#)
- [Process rule specification](#)
- [Project specification](#)
- [Project grouping specification](#)
- [Task specification](#)
- [Transfer set specification](#)

---

Each specification accepts the following global forms:

- [Object name form](#)
- [Query selection set reference form](#)
- [Cvid reference form](#)
- [File contents form](#)

## Object name form

You can reference an object in a database by using a four-part object name form.

- *name:version:type:instance*

For example `ClientSessionContext.java:23:java:J#1` refers to an object named `ClientSessionContext.java` with a version of 23, a type of `java`, and an instance of `J#1`.

- *name version\_delimiter version:instance*

The *version\_delimiter* is the current version delimiter used in the database. For example, with the default version delimiter of - (hyphen), the specification `ClientSessionContext.java-23:java:J#1` refers to an object named `ClientSessionContext.java` with a version of 23, a type of `java`, and an instance of `J#1`.

**Note** When you enable the [allow delimiter in name](#) feature and use the second form above, Synergy determines the **version** field by taking the part after the right-most version delimiter. For example, with a hyphen version delimiter, the specification `my-file-23:ascii:1` displays a name of `my-file` and a version of 23.

## Related topics

- [Query selection set reference form](#)
- [Cvid reference form](#)
- [File contents form](#)

---

### **Query selection set reference form**

Commands that perform queries or list objects typically set the query selection set. The selection set represents the results of the previous query and the order in which the objects were displayed in the previous command. The output from these commands is numbered by default. You can refer to objects in the selection set by using:

- @*n*

where *n* is the number of the displayed object from a previous query. For example, @1 refers to the first object, @2 the second object and so on.

- @*n-m*
- @*n-@m*

where *n* and *m* are the numbers of displayed objects from a previous query, and where *m* is greater than or equal to *n*. This refers to the *n*'th to *m*'th object, inclusive. For example, @2-5 refers to the second through fifth object, inclusive; @3-@6 refers to the third through sixth object, inclusive.

- @

This refers to all of the objects in the query selection set.

### **Related topics**

- [Object name form](#)
- [Cvid reference form](#)
- [File contents form](#)

### ***Cvid reference form***

Each object in a Synergy database has a unique integer index called a *cvid*. You can refer to these objects by using their *cvid* with the following form:

- @=*n*

where *n* is the object's *cvid*. For example, @=12345 refers to the object whose *cvid* is 12345.

### **Related topics**

- [Object name form](#)
- [Query selection set reference form](#)
- [File contents form](#)



---

### **File contents form**

The filename contents form allows you to create a file containing zero, one, or more specifications that are valid for the required type of specification. Additionally, you can refer to the contents of that file.

- `@:file_path`

where *file\_path* is the path to a file containing zero, one, or more specifications. For example, `@:myobjects.txt` will take the specifications from a file named `myobjects.txt` in the current directory.

**Note** The file specified by a file contents form cannot contain further specifications that are file contents forms.

### **Related topics**

- [Object name form](#)
- [Query selection set reference form](#)
- [Cvid reference form](#)

## **Baseline specification**

A *baseline\_spec* refers to one or more baseline objects in a Rational Synergy database.

A *baseline\_spec* can have any of the following forms.

- Any of the following global forms set to baseline objects:

- [Object name form](#)
- [Query selection set reference form](#)
- [Cvid reference form](#)
- [File contents form](#)

- *baseline\_name*

This is the name of the baseline that was created in the current database. For example, `Client build 46` refers to the baseline named `Client build 46` that was created in the current database.

- *datasase\_id dcm\_delimiter baseline\_name*

where *database\_id* is a DCM database identifier where the baseline was created, *dcm\_delimiter* is the current DCM delimiter (`#` is the default), and *baseline\_name* is the name of the baseline. For example with a default DCM delimiter of `#` the baseline specification `A#api build 67` refers to the baseline named `api build 67` that was created in database `A`.

---

## Change request specification

A *change\_request\_spec* is a reference to one or more change requests. A *change\_request\_spec* can have any of the following forms:

- Any of the following global forms set to change request objects:
  - [Object name form](#)
  - [Query selection set reference form](#)
  - [Cvid reference form](#)
  - [File contents form](#)
- *change\_request\_id*

This refers to a single change request by its change request identifier. In a DCM initialized database, this might take either of the forms shown below. In a non-DCM initialized database, only the first form is valid.

  - *change\_request\_number*

This refers to the change request with a specified number that was created in the current database. For example, 34 refers to change request 34 created in the current database.
  - *database\_id dcm\_delimiter change\_request\_number*

where *database\_id* is a DCM database identifier that describes where the change request was created, *dcm\_delimiter* is the current DCM delimiter (# is the default), and *change\_request\_number* is the change request number. For example, B#543 refers to change request number 543 created in database B.
- *change\_request\_id(,change\_request\_id)...*

This refers to a collection of change requests specified as a comma separated list of *change\_request\_id* elements. For example, B#543,23 refers to change request 543 created in database B and change request 23 created in the current database.
- *change\_request\_id-change\_request\_id*

This refers to a collection of change requests specified as a range. The range includes the change request specified by the first *change\_request\_id* to the second *change\_request\_id*, inclusive.

The starting change request and ending change request of the range must be for the same creating database. For example, 34-37 refers to change requests 34, 35, 36, and 37 created in the current database. The range A#34-B#37 is invalid because the starting change request is for database A while the ending change request of the range is for database B.

### **Database specification**

A *database\_spec* refers to one or more DCM database definitions. See the [Rational Synergy Distributed](#) book for more details about DCM and DCM database definitions. It can have any of the following forms:

- Any of the following global forms set to DCM database definition objects:
  - [Object name form](#)
  - [Query selection set reference form](#)
  - [Cvid reference form](#)
  - [File contents form](#)
- *database\_id*

where *database\_id* is the DCM database identifier associated with the DCM database definition. For example, `IRVJ` refers to the DCM database definition with database identifier `IRVJ`.

---

## File specification

A *file\_spec* refers to one or more projects, directories or files in a Rational Synergy database. It can have any of the following forms:

- Any of the following global forms set to projects, directories, or files:
  - [Object name form](#)
  - [Query selection set reference form](#)
  - [Cvid reference form](#)
  - [File contents form](#)
- [Work area reference form](#)

This refers to an object in a maintained project work area by its path in the work area.
- [Project reference form](#)

This refers to an object by its relative path within a specified project.

An object name can contain up to 155 characters; the object version can contain up to 32 characters.

### Work area reference form

When an object version is a member of a project and the project is synchronized under a work area directory in the file system, you can reference the object version by its path in the projected directory structure.

The following example uses Windows paths. The functionality is the same on UNIX. For example, if the `foo.c-4` object version is a member of the `jobA-1` project under the `dir1` directory and the project work area is `c:\users\joe\ccm_tutorial`, the work area reference form for `foo.c-4` is:

```
c:\users\joe\ccm_tutorial\jobA-1\jobA\dir1\foo.c
```

If the current working directory is `c:\users\joe\ccm_tutorial\jobA-1\jobA`, you can reference the `foo.c-4` object version by using the relative path:

```
dir1\foo.c
```

You can augment the work area reference with the version to refer to another version of the object:

```
path\object_name[:version]
```

```
path\object_name[version_delimiter version]
```

Use this file specification to refer to any different versions of the object version.

For example, if `foo.c-4` is a member of the current project and has a predecessor `foo.c-3`, you can reference the predecessor by using the relative path belonging to `foo.c-4`:

```
dir1\foo.c-3
```

**Note** When you enable [allow\\_delimiter\\_in\\_name](#) and use the second form with the `version_delimiter`, the complete field including the version delimiter is taken as the name, and the version will be the current version used in the project. For example, with a version delimiter of `-`, the specification `c:\users\joe\ccm_tutorial\jobA-1\jobA\dir1\foo.c-23` gets a name of `foo.c-23`. This refers to the current version of the object used in the work area. Avoid potential confusion when using `allow_delimiter_in_name` by specifying any explicit version that uses the `:version` form after the path and name.

### Project reference form

A project reference form refers to a project, directory, or file by its relative path within a specified project. This can be used even if the project does not have a maintained work area or if that work area is not visible to the client.

*relative\_path@project\_spec*

*relative\_path:version@project\_spec*

*relative\_path version\_delimiter version@project\_spec*

where *relative\_path* is a relative path that can use `/` or `\` as file separators, *project\_spec* is a [Project specification](#), *version\_delimiter* is the current version delimiter (default `-`, hyphen), and *version* is an optional version of the object. If you don't specify a version, the referenced object is the version used in the project.

The relative path is relative to the top of the project. For example, the specification `myproject\src\ClientContext.java@myproject:1` refers to the current version of a file named `ClientContext.java` under a directory named `src` under the root directory `myproject` of the project `myproject:1`.

You can specify an optional version after the relative path. For example, the specification `myproject\src\ClientContext.java:23@myproject:1` refers to the version 23 of a file named `ClientContext.java` under a directory named `src` under the root directory `myproject` of the project `myproject:1`.

**Note** When you enable [allow\\_delimiter\\_in\\_name](#) and use the last form with the `version_delimiter`, the complete field including the version delimiter is taken as the name, and the version will be the current version used in the project. For example, with a version delimiter of `-`, the specification `myproject\src\ClientContext.java-23@myproject:1` gets a name of `ClientContext.java-23`. This refers to the current version of the object used in the project. Avoid potential confusion when using

---

`allow_delimiter_in_name` by specifying any explicit version that uses the `:version` form after the path and name.

## Folder specification

A *folder\_spec* refers to one or more folders in a Rational Synergy database. It can have any of the following forms:

- Any of the following global forms set to folder objects:

- [Object name form](#)
- [Query selection set reference form](#)
- [Cvid reference form](#)
- [File contents form](#)

- *folder\_id*

This refers to a single folder by its folder identifier. In a DCM initialized database, this can use the forms shown below. In a non-DCM initialized database, only the first form is valid.

- *folder\_number*

This refers to the folder with the specified number that was created in the current database. For example, 34 refers to *folder 34* created in the current database.

- *database\_id dcm\_delimiter folder\_number*

where *database\_id* is a DCM database identifier where the folder was created, *dcm\_delimiter* is the current DCM delimiter (# is the default), and *folder\_number* is the folder number. For example, B#543 refers to folder number 543 created in database B.

- *folder\_id(,folder\_id)...*

This refers to a collection of folders specified as a comma-separated list of *folder\_id* elements. For example, B#543,23 refers to folder 543 created in database B and folder 23 created in the current database.

- *folder\_id-folder\_id*

This refers to a collection of folders specified as a range. The range includes the folder specified by the first *folder\_id* to the second *folder\_id*, inclusive. The starting folder and ending folder of the range must be for the same creating database. For example, 34-37 refers to folders 34, 35, 36, and 37 created in the current database. However, A#34-B#37 is invalid because the starting folder is for database A while the ending folder of the range is for database B.



---

## **Folder template specification**

A *folder\_template\_spec* refers to one or more folder templates in a Rational Synergy database. It can have any of the following forms:

- Any of the following global forms set to folder template objects:

- [Object name form](#)
- [Query selection set reference form](#)
- [Cvid reference form](#)
- [File contents form](#)

- *folder\_template\_name*

This specifies the folder template by its name. For example `All completed tasks for %release` refers to the folder template named **All completed tasks for %release**.

## **Object specification**

An *object\_spec* refers to one or more objects in a Rational Synergy database. It can have any of the following forms:

- Any of the following global forms:
  - [Object name form](#)
  - [Query selection set reference form](#)
  - [Cvid reference form](#)
  - [File contents form](#)
- [Work area reference form](#)

This refers to an object in a maintained project work area by its path in the work area.
- [Project reference form](#)

This refers to an object by its relative path within a specified project.

An *object\_spec* supports the same syntax as a [File specification](#) but can refer to any object in a Synergy database, not just projects, directories, or files.

---

## ***Process specification***

A *process\_spec* refers to one or more process definitions in a Rational Synergy database. It can have any of the following forms:

- Any of the following global forms set to process definition objects:

- [Object name form](#)
- [Query selection set reference form](#)
- [Cvid reference form](#)
- [File contents form](#)

- *process\_name*

This specifies the process by its name. For example `Standard` refers to the process named **Standard**.

## **Process rule specification**

A *process\_rule\_spec* refers to one or more process rules in a Rational Synergy database. It can have any of the following forms:

- Any of the following global forms set to process rule objects:

- [Object name form](#)
- [Query selection set reference form](#)
- [Cvid reference form](#)
- [File contents form](#)

- *generic\_process\_rule\_name*

This specifies a generic process rule by its name. For example `Integration Testing` refers to the generic process rule named **Integration Testing**.

- *release:generic\_process\_rule\_name*

This specifies a release-specific process rule. *release* is a valid release, and *generic\_process\_rule\_name* is the name of the generic process rule from which it was created. For example, `1.0:Integration Testing` refers to the release-specific process rule for release 1.0 created from the **Integration Testing** generic process rule.

---

## Project specification

A *project\_spec* refers to one or more projects in a Synergy database. It can have any of the following forms:

- Any of the following global forms set to projects objects:

- [Object name form](#)
- [Query selection set reference form](#)
- [Cvid reference form](#)
- [File contents form](#)

- *name:version*  
*name version\_delimiter version*

where *name* is the name of the project, *version* is the version of the project, and *version\_delimiter* is the current version delimiter (default is -, hyphen). This specifies a project with a default instance that was created in the current database. In a non-DCM initialized database, the default instance is 1. For example, `myproject:23` refers to the project `myproject:23:project:1`. In a DCM-initialized database with a DCM database identifier of `A` and a DCM delimiter of `#`, the default instance would be `A#1`. For example, `myproject:23` refers to the project `myproject:23:project:A#1`.

- *name:version:project:instance*  
*name version\_delimiter version:project:instance*

where *name* is the name of the project, *version* is the version of the project, *instance* is the instance of the project, and *version\_delimiter* is the current version delimiter (default is -, hyphen). For example, `myproject:23:project:A#1` refers to the project named `myproject` with version `23` and instance `A#1`.

## **Project grouping specification**

A *project\_grouping\_spec* refers to one or more project groupings in a Synergy database. It can have any of the following forms:

- Any of the following global forms set to project grouping objects:

- [Object name form](#)
- [Query selection set reference form](#)
- [Cvid reference form](#)
- [File contents form](#)

- *project\_grouping\_displayname*

This specifies the project grouping by its display name. For project groupings for the current owner this can be of the forms:

- *My release purpose Projects*
- *My release purpose Projects for Database database*

For example, *My client/2.0 Integration Testing Projects* specifies a project grouping owned by me for release **client/2.0** with a purpose of **Integration Testing**.

For project groupings owned by any user, this can be of the forms:

- *user's release purpose Projects*
- *user's release purpose Projects for Database database*

For example, *Linda's client/2.0 Integration Testing Projects* specifies a project grouping owned by *Linda* for release **client/2.0** with a purpose of **Integration Testing**.

---

## **Release specification**

A *release\_spec* refers to one or more release definitions in a Rational Synergy database. It can have any of the following forms:

- Any of the following global forms set to release definition objects:

- [Object name form](#)
- [Query selection set reference form](#)
- [Cvid reference form](#)
- [File contents form](#)

- *release\_name*

This specifies a release by its release. For example, the release specification `client/2.0` refers to the release definition for release **client/2.0**.

## Task specification

A *task\_spec* refers to one or more tasks in a Rational Synergy database. It can have any of the following forms:

- Any of the following global forms set to tasks:
  - [Object name form](#)
  - [Query selection set reference form](#)
  - [Cvid reference form](#)
  - [File contents form](#)
- *task\_id*

This refers to a single task by its task identifier. In a DCM-initialized database, this can use the forms shown below. In a non-DCM initialized database, only the first form is valid.

  - *task\_number*

This refers to the task with the specified number that was created in the current database. For example, 34 refers to task 34 created in the current database.
  - *database\_id dcm\_delimiter task\_number*

where *database\_id* is a DCM database identifier where the task was created, *dcm\_delimiter* is the current DCM delimiter (# is the default), and *task\_number* is the task number. For example, B#543 refers to task number 543 created in database B.
- *task\_id(,task\_id)...*

This refers to a collection of tasks specified as a comma-separated list of *task\_id* elements. For example, B#543,23 refers to task 543 created in database B and task 23 created in the current database.
- *task\_id-task\_id*

This refers to a collection of tasks specified as a range. The range includes the task specified by the first *task\_id* to the second *task\_id*, inclusive. The starting task and ending task of the range must be in the same database. For example, 34-37 refers to tasks 34, 35, 36, and 37 created in the current database. However, A#34-B#37 is invalid because the starting task is for database A while the ending task of the range is for database B.



---

## ***Transfer set specification***

A *transfer\_set\_spec* refers to one or more transfer sets in a Rational Synergy database. It can have any of the following forms:

- Any of the following global forms set to transfer sets:

- [Object name form](#)
- [Query selection set reference form](#)
- [Cvid reference form](#)
- [File contents form](#)

- *transfer\_set\_name*

This specifies a transfer set by its name. For example, `Entire Database` specifies the predefined **Entire Database** transfer set.

## Global formatting options

Rational Synergy CLI commands allow you to define how information is presented, how to format it, sort it, and group it. The following is the complete set of formatting options:

- [Advanced use of format strings](#)
- [Column alignment](#)
- [Column format elements](#)
- [Column headers](#)
- [Keywords](#)
- [Numbering](#)
- [Property formatting](#)
- [Sorting and grouping](#)

Not all commands support all formatting options. The command syntax for each subcommand shows which formatting options are supported for the subcommand.

At the end of this section, see [Formatting usage examples](#) for examples of how to incorporate options to enhance your output. The following formatting options are available:

- [-ch|-column headers](#)
- [-nch|-nocolumn headers](#)
- [-f|-format](#)
- [-nf|-noformat](#)
- [-sby|-sortby](#)
- [-ns|-nosort](#)
- [-gby|-groupby](#)
- [-sep|-separator](#)
- [-u|-unnumbered](#)

---

## Formatting strings

A format string can consist of a number of string literals or keywords or string literals and keywords. When processed, the format string is represented in the command output by an expanded (unchanged) string. For example, the format string "Release=%release" consists of two parts: the string literal "Release=" and a keyword %release. For each reported object, the string literal appears in the output and the keyword is replaced with a string representation of the object's *release* property. A command that reports on three objects might show the following output:

```
Release=client/1.0
Release=client/2.0
Release=server/1.0
```

## Column alignment

The output for most commands is arranged into aligned columns by default. The format string is distributed into columns using the default separator character, which is a space.

For example, a format string of "%name Release=%release" is distributed into the following columns:

- The first column contains one element, the %name keyword.
- The second column contains two elements:
  - \* The string literal "Release="
  - \* The keyword %release

A command that reports on three objects might show the following output:

```
ClientSessionContext.java      Release=client/1.0
DisplayNameResourceBundle.java Release=client/2.0
DataNotFoundException.java     Release=client/1.0
```

In the example output, the first column contains spaces to vertically align the second column. The alignment is calculated assuming that the terminal display device uses a mono-spaced font. If you see incorrectly aligned columns, check that your display device uses a fixed space font rather than a proportional font. You might need to set `ccm.cli.format.font` to your display device's fixed space font to correct alignment.

A newline character causes a format string to end at the current column, and makes the text following the newline character part of the first column in the next line.

## -sep|-separator

The `-sep|-separator separator` option allows you to specify a different separator character. For example, consider the format string "%name|Release %release". With the default separator character, the format string consists of two columns as follows:

- The first column contains two elements:

- \* The keyword `%name`
- \* The string literal `"|Release"`
- The second column contains the keyword `%release`

A command that reports on three objects might show the following output:

```
ClientSessionContext.java|Release      client/1.0
DisplayNameResourceBundle.java|Release client/2.0
DataNotFoundException.java|Release     client/1.0
```

Now consider the same format string but with the option `-sep "|"` specified. The format string is distributed in columns using the pipe separator character:

- The first column contains the keyword `%name`
- The second column contains two elements:
  - \* The string literal `"Release "`
  - \* The keyword `%release`

With the same data this would result in the following output:

```
ClientSessionContext.java      Release client/1.0
DisplayNameResourceBundle.java Release client/2.0
DataNotFoundException.java     Release client/1.0
```

### **-nf|-noformat**

The `-nf|-noformat` option specifies not to use column alignment. When specified, the format string is interpreted as a single column containing all of the elements. Use this option to produce a delimiter-separated output (such as comma-separated).

### **Advanced use of format strings**

In the examples described so far, the format strings have consisted of either string literals or simple keywords, such as `%release`. This might be sufficient to achieve the format you want. However, the Synergy CLI client supports more advanced forms that provide greater control and flexibility.

As described earlier, a format string is first distributed into columns unless you specify the `-nf|-noformat` option. Each column can consist of a collection of any of the following elements:

- A string literal  
This is displayed as is in the output. If you want a single percent sign `%` to appear in the output, specify a pair, for example `%%`.
- A keyword  
This is replaced with a string representation of the value of a specified property on the object being reported.

- A column property format element

This controls some overall formatting of the contents of that column. See [Column format elements](#) for more information.

## Keywords

In the previous examples, the simplest form of a keyword was a percent sign followed by the name of a property. For example, `%release` is replaced with a string representation of the value of a property named `release`. When the objects being reported are objects in a Synergy database, the property name will be either the name of an attribute, or [Built-In keywords](#).

In some cases, the objects being reported might not be database entities. For example, a project membership conflict consists of data, such as the object in conflict, the parent project, the type of conflict, the associated tasks, and so on. To specify what data is reported, you can use an `objectkey` keyword:

```
%[objectkey]propertyname
```

The `objectkey` specifies the part of the object that the named property refers to. For example, for a project membership conflict, the `object` object key refers to the file or directory in conflict and the `project` object key refers to the parent project associated with the conflict. A format string of `%[project]displayname %[object]release` shows the built-in `displayname` property of the project associated with each membership conflict, and the `release` property of the file or directory in conflict.

The details of which object keys are supported are described in each command.

The Synergy CLI supports advanced keywords with any of the following forms:

```
{keywordspec}
{keywordspec:-substitutionstring}
{keywordspec:+substitutionstring}
```

where a `keywordspec` can be any of the following forms:

```
propertyname
+propertyname
-propertyname
[objectkey]propertyname
+[objectkey]propertyname
-[objectkey]propertyname
propertyname[propertyformat]
+propertyname[propertyformat]
-propertyname[propertyformat]
[objectkey]propertyname[propertyformat]
+[objectkey]propertyname[propertyformat]
-[objectkey]propertyname[propertyformat]
```

In each form above, the `objectkey` specifies which object to reference, and the `propertyname` is the name of a property, such as an attribute or [Built-In keywords](#).

The *propertyformat* specifies how to convert the property value to a string and then format it. See [Property formatting](#) for more information.

The forms using the *substitutionstring* use an alternate string literal for the expanded keyword whether or not the specified property or attribute exists.

- The *-substitutionstring* form means that the specified substitution string will be used if the property does not exist. If the property exists, its value is used.
- The *+substitutionstring* form means that the specified substitution string will be used if the property exists, otherwise a string representation of a null value is used.

The optional leading + or - specifies a default sorting order. See [Sorting and grouping](#) for more information.

**Examples:**

The keyword `"%{release:-No release}"` references the `release` property of the object. If the property exists, its value is displayed. If the property does not exist, the string `"No release"` is displayed.

The keyword `"%{[project]release[truncate='20']}"` references the `release` property of the project associated with the object. If the property exists, its value is displayed, truncated with ellipsis if it exceeds a width of 20.

---

## Property formatting

When a keyword references a property of an object, its value is converted into a string and formatted using property formatting options. For example, a boolean attribute is converted into a string that represents the boolean true or false value. If a property format is not specified, the property value is converted to a string as follows:

Type of value	Conversion
<i>null</i>	For most commands, if the specified property doesn't exist, "<Not available>" is displayed.
boolean	The boolean value true is displayed as "TRUE", and false is displayed as "FALSE".
String (string or text attributes)	The string value is shown with any trailing newlines removed.
Date (time attributes)	Date is converted to a date string. If <code>ccm.cli.format.date</code> is defined, this is used as a date format string. If it is not defined, then the default date format for your locale is used.
Integer	Integer is displayed as a decimal integer string.
Float	Float is displayed as a decimal floating point string.
Collection	When the property value is a collection of elements, the value is formatted with a each element is formatted as a property and separated by commas.

In keywords or [Column format elements](#), you can specify additional controls to format each property value. This is done by specifying a *propertyformat* that consists of zero, one, or more of the following items separated by white-space characters:

```
propertyFormatName=propertyFormatValue  
propertyFormatName=' singleQuotedPropertyFormatValue '  
propertyFormatName=" doubleQuotedPropertyFormatValue "
```

where:

*propertyFormatName* is a name of a supported property format option.

*propertyFormatValue* is a value that does not contain white-space characters. This value must consist of "word" characters as defined by Java regular expression rules.

*singleQuotedPropertyFormatValue* is a string that can contain any character except a single quote (apostrophe).

*doubleQuotedPropertyFormatValue* is a string that can contain any character except a double quote.

**Example**

```
wrap=20 truncate=100
```

This specifies that the value will be truncated to a maximum of 100 characters, and the characters wrapped to fit in a column width of 20.

Rational Synergy supports the following property format names:

Property format name	Description
null	Specifies a string that will be used to represent a <code>null</code> value when the property does not exist.
false	Specifies a string that will be used to represent a boolean <code>false</code> value.
true	Specifies a string that will be used to represent a boolean <code>true</code> value.
format	Specifies a format string to be used to format the property value. If the string contains "{0}", the formatting uses <i>MessageFormat</i> , a standard <a href="#">Java formatting and localization</a> feature. If the string does not contain "{0}", the formatting uses <code>printf</code> , a standard <a href="#">Java string formatting</a> feature.
dateformat	Specifies a date format string to be used to format a date property value. The date format must be a valid string for <i>SimpleDateFormat</i> , a standard <a href="#">Java date formatter</a> .
list_begin	Specifies the starting string to be used when the property value is a collection.
list_end	Specifies the ending string to be used when the property value is a collection.
separator	Specifies the item separator to be used when the property value is a collection.
truncate	Specifies the maximum length for the displayed string. If the string is truncated, the string will end with the truncate indicator and be the specified maximum length. By default, the truncate indicator is ellipsis (...).
truncate_each	Specifies the maximum length for each element in a value that is a collection. If the string is truncated, the string will end with the truncate indicator and be the specified maximum length. By default, the truncate indicator is ellipsis (...).



---

Property format name	Description
truncate_indicator	Specifies the string to be used to show the truncated display value. By default, the value is ellipsis (...).
wordwrap	Specifies the maximum column width. Also specifies to wrap characters wider than the specified width.
wrap	Specifies the maximum column width. Also specifies to wrap characters wider than the specified width.
keep_trailing_newlines	A value of <code>true</code> preserves the value's trailing newlines in the output.
nocolumn	A value of <code>true</code> specifies to ignore the column for alignment.
indent	Specifies the line indentation width for the property value. The value must be an integer between 1 and 1000, inclusive.

### Column format elements

In a format string, you can specify a column format element to control how to format data in that column. When you specify a column format element, the entire column is formatted using the specified options in it. A column format element takes the form:

```
%[propertyformat]
```

where *propertyformat* is a property format described in [Property formatting](#).

For example, consider the format string "`%name|[%wrap=20]%version  
%{release[truncate=30]}`" with a separator character of pipe `|`. The format string consists of two columns:

The first column contains the `%name` keyword.

The second column contains four elements:

- A column format element "`%[wrap=20]`" that will result in the column being character wrapped at a width of 20.
- The keyword `%version`
- The string literal " "
- The keyword `%{release[truncate=30]}` that will show the release value truncated to a width of 30.

When the second column is processed, the displayed string is determined as follows:

The release property value is converted into a string as required.

1. The string literal " " is appended.
2. The release property value is converted into a string, as required.

3. If necessary, the release string is truncated to a maximum width of 30.
4. The potentially truncated string is appended to the column.
5. The whole column value is character wrapped at a width of 20.

### **-f|-format**

The `-f|-format format` option specifies a format string to be used. It can use any of the formatting options described in previous sections. When omitted, commands will use a default format string specific to the subcommand.

### **Sorting and grouping**

With most commands, if sorting is to be performed without sorting or grouping options specified, a default sort algorithm sorts the objects that are reported. If the Selection set is updated, it reflects the order in which the objects were reported in the sorted output.

The default sort algorithm uses the format string and the real property values referenced by the keywords in that format string. For example, if you use the format string `"%create_time %release %displayname"` for objects in a Synergy database, you'll see the following sorting:

1. The `create_time` attribute as an ascending primary key and using the real date value (not the date string that is displayed)
2. The `release` attribute as an ascending secondary key
3. The `displayname` built-in keyword as an ascending tertiary key

By default, each sort key performs an ascending sort of its property value. You can change it by using one of the advanced keyword forms. See [Keywords](#) for further details. For example, the format string `"%{-create_time} %{-release} %displayname"` sorts using:

1. The `create_time` attribute as a descending primary key and the real date value (not the date string that is displayed)
2. The `release` attribute as a descending secondary key
3. The `displayname` built-in keyword as an ascending tertiary key

Commands that process arguments in the order they are specified generally do not perform sorting and do not support any of the sorting or grouping related options. An example of such a command is `ccm properties`.

### **-ns|-nosort**

The `-ns|-nosort` option disables all sorting and grouping. Use this option to preserve the order of objects as found.

---

## **-sby|-sortby**

The `-sby|-sortby sort_spec` option specifies explicit sorting for the output that overrides the default sort algorithm based on the format string. The *sort\_spec* is a list of one or more items of the following form with optional comma separator characters:

```
propertyname  
+propertyname  
-propertyname  
[objectkey]propertyname  
+[objectkey]propertyname  
-[objectkey]propertyname
```

An item with a leading **+** is used as an ascending sort key. An item with a leading **-** is used as a descending sort key. An item without either is used as an ascending sort key.

### **Examples**

- `create_time,release`  
Sort with ascending `create_time` as the primary key and ascending `release` as the secondary key.
- `+create_time,release`  
Sort with descending `create_time` as the primary key and ascending `release` as the secondary key.
- `+release-create_time`  
Sort with ascending `release` as the primary key and descending `create_time` as the secondary key.

Note that the properties listed in the *sort\_spec* do not need to appear in the format string. If you specify `-sby|-sortby`, sorting does not depend on the properties referenced in the format string.

## **-gby|-groupby**

Grouping is another form of sorting. When you specify the `-gby|-groupby groupformat` option, the values specified in the *groupformat* format string perform the initial levels of sorting. Objects having an identical set of grouped data values are kept together and preceded by a group header in the output. The group header is formed by expanding the specified *groupformat* format string. See [Formatting strings](#) for further information.

For example, if you specify `-groupby "Release %release:"`, all objects are sorted using the `release` property value as the primary key. This takes precedence over any other sort keys. For example if the command reports five objects, two with a `release` value of **1.0** and three with a `release` value of **2.0**, the objects are grouped as follows:

Release 1.0:  
*first object with release 1.0*  
*second object with release 1.0*

Release 2.0:  
*first object with release 2.0*  
*second object with release 2.0*  
*third object with release 2.0*

If you use both grouping and sorting, the sort keys specified by the groupformat are used first, followed by the sort keys from any *sort\_spec*, or if not specified, the *format* string. For example, if you specify `-groupby "Release %release:" -sortby create_time`, sorting uses release as an ascending primary key and `create_time` as an ascending secondary key.

See the second example in [Formatting usage examples](#) for a detailed example of grouping, sorting, and column headers.

### Column headers

Column headers are title strings that are derived from the format string. They appear as headings above each column in the output. For example, if you specify a format string of `"%name %release %create_time"`, the corresponding column headings are Name, Release, and Create\_time for columns one, two, and three, respectively.

See the second example in [Formatting usage examples](#) for a detailed example of grouping, sorting, and column headers.

### **-ch|-column\_headers**

Specifies to use column headers for the output. This option cannot be used with the `-nf|-noformat` option.

The default is for no column headers to be used.

### **-nch|-nocolumn\_headers**

Specifies not to use column headers for the output. This is the default for most commands.

### Numbering

Commands whose output represents objects in a Synergy database and that set the selection set use numbered output by default. The numbering reflects the selection set reference number used to refer to that object using a [Query selection set reference form](#).

---

**-u|-unnumbered**

This specifies that the output will not be numbered. This option has no effect on the ordering of the output or the resulting selection set.

### Formatting usage examples

The examples in this section are not tied to specific commands. The following data describes the objects and properties used as output in the examples.

displayname	task_synopsis	owner	release
101	Fix defect M#1234	fred	1.0
102	Implement sorting	joe	1.0
900	Fix defect M#12345	susan	1.0
901	Implement date formatting	john	1.0_patch
1000	Fix defect M#1357	fred	1.1
1001	Fix defect M#6523	fred	1.1
1002	Implement new property formats	susan	2.0
1003	Add grouping and extend sorting	joe	2.0
1004	Fix defect J#1234	susan	1.1
1005	Extend grouping feature	john	2.0

- The following example shows data that isn't numbered (`-u`) and isn't sorted (`-nosort`), so the output displays in the original order of the data. The `-format` values organize the data into four columns: display name, task synopsis, owner, and release. The columns are aligned because `-noformat` was not used.

```
-u -nosort -format "%displayname %task_synopsis %owner %release"
101 Fix defect M#1234          fred 1.0
102 Implement sorting         joe 1.0
900 Fix defect M#12345       susan 1.0
901 Implement date formatting john 1.0_patch
1000 Fix defect M#1357       fred 1.1
1001 Fix defect M#6523       fred 1.1
1002 Implement new property formats susan 2.0
1003 Add grouping and extending sorting joe 2.0
1004 Fix defect J#1234       susan 1.1
1005 Extend grouping feature john 2.0
```

- 
- The following example shows the output numbered by default. Additionally, it shows:
    - The format string defines two columns: display name and task synopsis.
    - Grouping uses a format that references the `owner` property.
    - Sorting is ascending on the `release` value. Note that the `release` value is not displayed in the output, but is used only to sort. The sort order uses `owner` as the ascending primary key (grouping takes precedence), and `release` as the ascending secondary key.
    - Objects with the same owner value are grouped together under the same grouping header.
    - Column headers are displayed.

```
-format "%displayname %task_synopsis" -groupby "Owner: %owner" -sortby  
+release -column_headers
```

```
Owner: fred
```

```
1) 101 Fix defect M#1234  
2) 1000 Fix defect M#1357  
3) 1001 Fix defect M#6523
```

```
Owner: joe
```

```
4) 102 Implement sorting  
5) 1003 Add grouping and extending sorting
```

```
Owner: john
```

```
6) 901 Implement date formatting  
7) 1005 Extend grouping feature
```

```
Owner: susan
```

```
8) 900 Fix defect M#12345  
9) 1004 Fix defect J#1234  
10) 1002 Implement new property formats
```

- The following example shows output numbered by default. The format string defines three columns: display name, task synopsis (the value will be **truncated** after 20 characters), and release. Sorting is ascending on `owner` (not a displayed property) and then `release`.

```
-format "%displayname %{task_synopsis[truncate=20]} %release"
-sortby +owner+release

1) 101 Fix defect M#1234 1.0
2) 1000 Fix defect M#1357 1.1
3) 1001 Fix defect M#6523 1.1
4) 102 Implement sorting 1.0
5) 1003 Add grouping and ... 2.0
6) 901 Implement date fo... 1.0_patch
7) 1005 Extend grouping f... 2.0
8) 900 Fix defect M#12345 1.0
9) 1004 Fix defect J#1234 1.1
10)1002 Implement new pro... 2.0
```

If you wanted to **wrap** the text after 20 characters rather than truncating it, you could substitute `wrap=20` for `truncate=20` in the above example, and the output will look as follows:

```
1) 101 Fix defect M#1234 1.0
2) 1000 Fix defect M#1357 1.1
3) 1001 Fix defect M#6523 1.1
4) 102 Implement sorting 1.0
5) 1003 Add grouping and ext 2.0
   ending sorting
6) 901 Implement date forma 1.0_patch
   tting
7) 1005 Extend grouping feat 2.0
   ure
8) 900 Fix defect M#12345 1.0
9) 1004 Fix defect J#1234 1.1
10)1002 Implement new proper 2.0
   ty formats
```



---

## Naming restrictions

This section describes the Rational Synergy object, release, database, and DCM naming restrictions.

### ***Restricted object names***

An object name can contain any combination of alpha numerics and symbols **except** for those characters that are restricted.

You cannot use any of the restricted characters as a version delimiter. For more information, see "delimiter command" in [Rational Synergy CLI Help, Traditional mode](#).

Following are some of the Rational Synergy object naming restrictions.

- 8-bit and double-byte characters (with the top bit set) are not permitted in object names.
- Project names must not contain tabs. Makefile names must not contain tabs or spaces.

Other restricted characters, and the reasons they are restricted, are shown in the following table.

Character	Why restricted
/	UNIX path delimiter; internal delimiter
\	Windows path delimiter; escape character
'	UNIX quoting character (forward quote)
"	Windows quoting character
:	Windows drive letter delimiter; Rational Synergy object specification delimiter
?	INFORMIX single-character wild card; regular expression
*	INFORMIX multiple-character wild card; regular expression
[	INFORMIX match syntax; regular expression
]	INFORMIX match syntax; regular expression
@	Rational Synergy object specifications delimiter
-	Rational Synergy version delimiter

You cannot use the following characters as the first character in an object name:

- , (comma)
- + (plus sign)

- - (dash)
- ~ (tilde)

### ***Restricted releases***

Each Rational Synergy release must conform to the following conventions:

- The release cannot contain any of the restricted characters shown in the preceding table.
- The values `none` and `as_is` are used as keywords in some commands, such as `ccm checkout`. Do not use these values for releases.
- The component name must contain 64 or fewer characters.
- The component release must contain 32 or fewer characters.

### ***Restricted database names***

Each Rational Synergy database name must conform to the following conventions:

- If two databases use the same database server, they cannot have the same name. The name is the leaf directory in the full database path.
- The database name can contain letters, digits, and underscores only.
- The database name must begin with a letter.
- The database name must contain 18 characters or fewer.

**Note** When naming a Rational Synergy database, uppercase and lowercase characters are equivalent.

### ***Restricted baseline names***

Each Rational Synergy baseline name must conform to the naming restriction.

The baseline name cannot contain the DCM delimiter, which by default is the # character, and can only be changed on DCM initialization or afterwards.

---

## **DCM restrictions**

Following are naming restrictions for DCM databases.

The following characters cannot be used in the database identifier of the DCM database.

<b>Character</b>	<b>Why restricted</b>
/	UNIX path delimiter; internal delimiter
\	Windows path delimiter; escape character
'	UNIX quoting character (forward quote)
"	Windows quoting character
:	Windows drive letter delimiter; Rational Synergy object specification delimiter
\$	INFORMIX single-character wild card; regular expression
?	INFORMIX single-character wild card; regular expression
*	INFORMIX multiple-character wild card; regular expression
[	INFORMIX match syntax; regular expression
]	INFORMIX match syntax; regular expression
@	Rational Synergy object specifications delimiter
<space>	The database identifier cannot be enclosed in quotes
#	Rational Synergy DCM delimiter; comment in GNU makefiles

In addition to the restricted characters listed above, the database identifier cannot be longer than 8 characters and cannot be the name "probtrac". The database identifier cannot contain the version delimiter by default. This can be changed by the [allow delimiter in name](#) attribute.

The DCM database identifier is case-sensitive. In any DCM cluster that uses lowercase databases, DCM database identifiers must be unique without respect to case. That is, you must not use database identifiers that are only different with regards to case.

You cannot use characters a-z, A-Z, or 0-9 for the DCM delimiter. You can use "!", "~", or "=" as an alternative delimiter. The default DCM delimiter is #, the pound sign.

## Case and file name limit database options

The following two database options, [Case](#) and [File name limit](#), could have an impact on the names you give your objects in the Rational Synergy database.

**Note** You can change these options by using Rational Synergy CLI in Traditional mode.

### **Case**

Rational Synergy supports case-sensitive file names. The keywords that support this option preserves the case of object names or to make object names lowercase in a Rational Synergy database.

If you want to view the case setting for your database, enter the following command:

```
ccmdb info database_path [-k case]
```

For a discussion of how to change the case option, refer to the `ccmdb info` command in the appropriate [Rational Synergy Administration Guide](#).

### **File name limit**

File name limits are dependent on both file system and Rational Synergy limitations. By default, you can create objects (files, directories, and projects) with names up to 155 characters in a Rational Synergy database. (See [Command and argument syntax](#) for a list of illegal symbols.)

To view the file name limit keyword for your database, enter the following command:

```
ccmdb info database_path [-k filelimit]
```

To change the file name limit keyword, you must be working as user `ccm_root`. For a complete discussion of how to change the file name limit mode, refer to the `ccmdb info` command in the appropriate [Rational Synergy Administration Guide](#).

---

## Built-In keywords

The following keywords are built into Rational Synergy. You can use these keywords to control the format of the output from query, list, and show operations on the command line, and query operations in the GUI.

**Note** You also can use attribute names as keywords. To list the attributes that are associated with an object, use the `ccm attr` command with the `-list` option.

Keyword	Description
<code>%baseline</code>	Returns a baseline project for a project. Returns <code>&lt;No baseline&gt;</code> if no baseline exists.
<code>%change_request</code>	Displays one or more change requests that are associated with the object. For a file, these change requests are determined based on the associated tasks and the change requests that are associated with those tasks.
<code>%change_request_duplicates</code>	Returns a list of a duplicate change requests for a change request.
<code>%change_request_original</code>	For a change request in the duplicate state, returns the original change request of which it is a duplicate.
<code>%change_request_release</code>	Displays the release property of change requests that are associated with the object.
<code>%change_request_status</code>	Displays the status of one or more change requests that are associated with the object.
<code>%change_request_synopsis</code>	Displays the synopsis of one or more change requests that are associated with the object.
<code>%displayname</code>	Defaults to <code>name-version</code> for files, directories, and projects.
<code>%fullname</code>	Returns the four-part name in <code>subsystem/cvtype/name/version</code> format.
<code>%has_relationship</code>	Displays those objects that have a relationship from the object in the query.
<code>%in_baseline</code>	Returns the displayname of a baseline for a project, if the project is in a baseline.

Keyword	Description (Continued)
%in_build	Returns the build number of the baseline for projects that are members of a baseline, if the project is in a baseline.
%instance	Alias for the %subsystem part of the object name.
%is_relationship_of	Displays those objects that have a relationship to the object in the query.
%model	Returns the %fullname of the current model object.
%objectname	Returns object name in <i>name-version:cvtype:subsystem</i> format.
%problem_duplicates	Returns a list of a duplicate problems for a problem.
%problem_original	For a problem in the duplicate state, returns the original problem of which it is a duplicate.
%purpose	Displays a project purpose.
%requirement_id	Displays the requirement ID saved on the change requests that are associated with the associated tasks for a task or object.
%root	Returns the <no_root> string if the object is not a root directory, and the %fullname project if it is a root directory.
%sourcename	Defaults to the name of the object.
%states	Returns legal object states separated by spaces.
%task	Returns a comma-separated list of task numbers associated with this object. Returns <void> if no associated tasks exist.
%task_platform	Returns a comma-separated list of platform values of the tasks associated with this object. Returns <void> if no associated tasks exist.

---

Keyword	Description (Continued)
%task_release	Returns a comma-separated list of release values of the tasks associated with this object. Returns <void> if no associated tasks exist.
%task_status	Returns a comma-separated list of task statuses associated with this object. Returns <void> if no associated tasks exist.
%task_subsystem	Returns a comma-separated list of subsystems ( <i>task_subsys</i> ) values of the tasks associated with this object. Returns <void> if no associated tasks exist.
%task_synopsis	If the object is a task, returns the <i>task_synopsis</i> attribute. Otherwise, returns a semi-colon-separated list of <i>task_synopses</i> for the tasks associated with this object, or <void> if no tasks are associated.
%type	Returns the type of the object (stored in the <i>cvtype</i> attribute).

## Regular expressions

- Ordinary characters

An ordinary character in a regular expression matches itself. The ordinary characters are characters other than those described below as special characters.

( ) [ ] ^ \$ . \* + ? | \

- Special characters

The special characters affect the matching behavior of regular expressions as described in the table below. Note that constructs that match arbitrary-length character sequences, i.e., \* + ?, will always match the longest left-most string that permits a match.

The table below shows special characters and their meaning.

Character	Meaning
^	Matches the beginning of the string. For example: <code>str ? * "^abc"</code> matches <code>str</code> starting with <code>abc</code>
\$	Matches the end of the string. For example: <code>str ? * "abc\$"</code> matches values of <code>str</code> ending with <code>abc</code>
.	Matches any single character. For example: <code>str ? * "a.c\$"</code> matches values of <code>str</code> containing <code>abc</code> , <code>axc</code> , etc
*	Matches zero or more of the immediately preceding expression. For example: <code>str ? * "ab*c\$"</code> matches values of <code>str</code> containing <code>ac</code> , <code>abc</code> , <code>abbc</code> , etc
+	Matches one or more of the immediately preceding expression. For example: <code>str ? * "a+c\$"</code> matches values of <code>str</code> containing <code>abc</code> , <code>abbc</code> , <code>agggc</code> , but not <code>ac</code>
?	Matches zero or one of the immediately preceding expression. For example: <code>str ? * "ab?c"</code> matches values of <code>str</code> containing <code>ac</code> or <code>abc</code>
	Matches either the preceding or following expression. For example: <code>str ? * "a b c"</code> matches values of <code>str</code> containing <code>a</code> , <code>b</code> , or <code>c</code>



---

Character	Meaning
[ ]	Matches any single character listed between brackets. For example: <code>str ? * "[ab]c"</code> matches values of <code>str</code> containing <code>ac</code> or <code>bc</code>
[ ^ ]	This combination of characters matches any single character not listed between brackets. For example: <code>str ? * "a[^b]c"</code> matches values of <code>str</code> containing <code>axc</code> for any replacement of <code>x</code> except for <code>b</code>
\	Escapes the character which immediately follows. For example: <code>str ? * "a\.c\$" </code> matches values of <code>str</code> containing <code>a.c</code> , and <code>str ? * "a\\c\$" </code> matches values of <code>str</code> containing <code>a\c</code> To embed a backslash character in a string, the string literal must contain two consecutive backslashes.
( )	Delimits subexpressions. For example: <code>str ? * "a(b c)*d*" </code> matches values of <code>str</code> containing <code>a</code> followed by any number of <code>b</code> 's or <code>c</code> 's followed by <code>d</code> , such as <code>"ad"</code> or <code>"acbbccd"</code>

### ***Wild card match regular expressions***

The following characters can be used with the query operator MATCHES.

Character	Meaning
*	Matches zero or more characters
?	Matches any single character
\	Removes the special significance of the next character (used to match* or ? by writing \* or \?)

## Administering purposes and templates

The CM administrator (the *ccm\_admin* role) can define the roles allowed to perform administrative operations for project purposes and process rules.

### **Project purpose manager**

A project purpose manager is any user with a role that contains the privilege `PRIVILEGE_MANAGE_PROJECT_PURPOSES`. By default, the privilege is contained in two roles: *build\_mgr* and *ccm\_admin*. Each site can add or remove this privilege from any role.

A project purpose manager can create or delete the project purposes for a database. However, if a build manager tries to modify a purpose that would require modification of a project for which the build manager does not have permission to modify, the operation will fail.

Only a user in the *ccm\_admin* role can edit this privilege.

### **Process rules manager**

The process rules manager exists for databases that use process rules.

**Note** Manage the process rules using Rational Synergy CLI in Traditional mode.

A process rules manager is any user with a role that contains the privilege `PRIVILEGE_MANAGE_PROCESS_RULES`. By default, this privilege is contained by two roles: *build\_mgr* and *ccm\_admin*, and only a user in the *ccm\_admin* role can edit this privilege. Each site can add or remove this privilege from any role.

A process rules manager can create or edit a process rule. However, if a build manager tries to modify a process rule that would require modification of a project for which the build manager does not have permission to modify, the operation will fail. In addition, a build manager cannot delete a process rule that is in use in any developer's working project. Only a user in the *ccm\_admin* role can delete a process rule.

For information about setting roles, see [role definitions](#).

### **Release manager**

The release manager is any user with a role that contains the privilege `PRIVILEGE_MANAGE_RELEASES`. By default, this privilege is contained by two roles: *build\_mgr* and *ccm\_admin*. Each site can add or remove this privilege from any role.

A release manager can create or edit release information. However, some operations require the *ccm\_admin* role, for example, remaining or deleting a release that is in use.

---

## Default settings

Defaults are a set of pre-designed values or settings shipped with Rational Synergy. These default settings have been defined as the settings that a majority of users would choose. However, you can modify these settings to best meet your needs. The information presented here defines the default values and where they are stored, describes how to change them, and explains any interaction between the settings. The following topics are discussed:

- [How defaults are set](#)
- [Default options](#)
- [Initialization file - Windows](#)
- [Initialization file - UNIX](#)
- [Environment variables](#)

## How defaults are set

The typical ways you can set or change default values are as follows:

- [System-wide settings](#)
- [Database-wide settings](#)
- [Personal settings](#)
- [Command line settings](#)

Rational Synergy reads the system-wide or database-wide settings first, then the personal settings, and then any values set from the command line. The last values that are read override the previous settings. The following paragraphs describe these common ways of setting default values.

### **System-wide settings**

System-wide default settings affect all users of an installation area. These defaults are usually set in the *system initialization* (*ini*) file.

The initialization file is called `ccm.ini`, and is found in the `etc` directory in `CCM_HOME`. All users of the installation area must restart their sessions to be able to use a new default setting in the system initialization file.

### **Database-wide settings**

Database-wide settings affect all users of a specific database. These defaults are usually set in an attribute on the model object, or in an attribute on a specific type object. When you change a setting by modifying an attribute in the model, you may need to restart your session for the new setting to take effect.

### **Personal settings**

Personal settings affect only your own sessions and databases. You set these defaults in one of three places, depending on the particular option:

- In the `[Options]` section in your *personal initialization* (*ini*) file
    - On Windows, the initialization file is called `ccm.ini`, and you create it in your Windows Documents and Settings directory (for example, `C:\Documents and Settings\user_name_directory`).
    - On UNIX, the initialization file is called `.ccm.ini`, and you create it in your `$HOME` directory.
- You must restart your session to be able to use a new default setting in your initialization file.

- 
- On the command line

Some personal settings are set using the command line.

### ***Command line settings***

You can use the `ccm set` command to set many Rational Synergy options by setting variables from the command line. This action sets the defaults for your immediate use so that you do not need to restart your session to have take effect. Depending on the option, the setting may apply for your current session only, or may persist between sessions. The syntax of the `set` command is:

```
ccm set variable_name variable_value
```

Most of the options that can be set with the `ccm set` command apply for the current session only. Those that are persistent are marked as such.

## Default options

The following section contains the Rational Synergy options, their default values, and where to set them. The options are listed in alphabetical order. The option names are *case-insensitive*, when specified as a variable on the command line or in a personal initialization file. However, for those options that are specified in a model attribute, the name of the attribute must be in lower case.

Rational Synergy uses settings made to the default options in the following order of precedence:

1. On the system or database level (that is, the system `ccm.ini` file or model attribute)  
Rational Synergy reads the options set in the system `ini` file or the appropriate model attribute first.
2. On the personal level (that is, in your personal `ccm.ini` file)  
Options set in the personal `ini` file override system `ini`-level settings.
3. Using the `ccm set` command  
Changes made using the `ccm set` command override both the system and personal `ini`-level settings.

The default line continuation character in an initialization file is a plus sign ( + ) on Windows and a backslash sign ( \ ) on UNIX.

For information about how to set model attribute options, see [Setting model object attribute options](#).

For information about how to set options in the system or personal `ini` file, see [Setting options in the system or personal ini file](#).

For information about how to set options using the [set command](#), see [Setting options using the ccm set command](#).

---

### ***activecm.disable\_sync\_at\_startup***

**Set Option:** System or personal `ini` file

Specifies whether ActiveCM performs a sync of all projects in the work area at startup. If not enabled, you must manually sync the work area and the database after starting a Taskbar interface session.

The default for `activecm.disable_sync_at_startup` is `FALSE`.

For information about how to set options in the system or personal `ini` file, see [Setting options in the system or personal ini file](#).

### ***add\_object\_task\_assoc***

**Set Option:** Model object attribute

Ensures that an existing object being added to a project is associated with the current task. This option is used with the Paste operation (from the GUI) or the [use command](#) (from the CLI).

The default is `TRUE`.

If you want your model to match Rational Synergy release 4.5 or earlier, you must set this option to `FALSE`.

You must restart your session for this change to take affect. For information about how to set model attribute options, see [Setting model object attribute options](#).

### ***add\_used\_subcomponents\_to\_task***

**Set Option:** Model object attribute

Ensures that an existing subcomponent being added to a project is associated with the current task. This option is used with the Paste operation (from the GUI) or the [use command](#) (from the CLI).

The default is `TRUE`.

If you want your model to match Rational Synergy release 4.5 or earlier, you must set this option to `FALSE`.

You must restart your session for this change to take affect. For information about how to set model attribute options, see [Setting model object attribute options](#).

### ***allow\_delimiter\_in\_name***

**Set Option:** Model object or type-specific attribute

Controls whether the delimiter is a restricted character.

When set to `TRUE`, the current delimiter is no longer a restricted character for non-project object names. The delimiter is still restricted for versions, types, instances, and projects.

With this feature enabled, object parsing is done from right to left in the sense that the right-most delimiter character is taken to be the delimiter. As a practical matter, object

identification is done by first attempting to identify a given string as a name, and if that fails, it is identified as *name<delimiter>version*. This capability is particularly an issue with create, move, and use.

**Note** All databases in the same DCM cluster **must** use the same value for this attribute. Failure to synchronize this value may result in undesirable behavior similar to having objects with "~" in them, and changing the delimiter to a "~."

With the feature enabled, you can also create non-project objects with versions. However, you will also be unable to use `ccm move` to set a version on a renamed file. (You can work around this limitation by using the `ccm attr` command or the Properties dialog to change the version.)

This attribute can also exist on individual types. In this case, the database setting is overridden if the database setting is `FALSE` and the type-specific setting is `TRUE`.

The following built-in types have `allow_delimiter_in_name` set to `TRUE`:

```
process_rule
processdef
saved_query
releasedef
project_grouping
folder_temp
```

The default is `FALSE`.

This option has the following restrictions and effects:

- Project names cannot contain the delimiter. If a user attempts to create or migrate a project whose name contains the delimiter, it will fail with an error message.
- After turning on this option, the version can no longer be specified for the create operation in the GUI or CLI; it will always treat the `object_spec` as the name. (Before this change, you can specify both the name and the version when creating an object, for example, specifying `f00-one` would create an object named `f00` with version `one`. After the delimiter change, it will create an object named `f00-one` with version `1`.) Otherwise, there is no other way to create an object whose name contains the delimiter; you would need to create it and then rename it.
- After turning on this option, CLI commands that use the object reference form `name<delim>version` will first try to find an object with that name, and if that fails, will try again without the part to the right of the delimiter. For example, if files named `f00-one` and `f00` both exist in the work area, and you specify `f00-one`, it will first look for a file named `f00-one`. Only if a file with that name is not found will it look for a file named `f00` with version `one`. You can still identify the other file (`f00` version `one`) using its 4-part name or the selection set reference form.
- After turning on this option, CLI commands will fail for objects that have the delimiter as the first character of the name, if the delimiter is `-` (dash or minus) because the



---

delimiter is also the option delimiter. For example, the command `ccm create -foo.c` will fail.

For information about how to set model attribute options, see [Setting model object attribute options](#).

### ***allow\_prep***

**Set Option:** Project or project type attribute.

Includes subprojects in the *prep* state when you update your project, if they are candidates (either from using a build management project as a baseline or from including in your update properties a task or folder that contains them).

This option is provided to support alternatives to the standard methodology. However, if you use this option, you run the risk of overwriting *prep* products with different (inappropriate) contents. Here is what can happen:

-- If a project includes *prep* subprojects, and the project owner (or build manager, in a build management project) is running in the *build\_mgr* role, then when he builds his project, the projects within the *prep* subprojects can be rebuilt if they are determined to be out of date. After being rebuilt, they may be out of sync with respect to the rest of the products that make up the software for which they were last built.

The default is `FALSE`, and *prep* subprojects cannot be used when a project is updated.

A project type attribute is set the same way as model attributes. For information about how to set model attribute options, see [Setting model object attribute options](#).

### ***auto\_create\_component\_tasks***

**Set Option:** Releasedef type attribute

Specifies whether component tasks are created by default when a new baseline is created for a release. This option can be overridden by the corresponding setting on the individual release.

If the value is `TRUE`, component tasks are created for new baselines; if the value is `FALSE`, they are not. The default value for this attribute is `TRUE`.

To override this setting on an individual release, see the [Modifying a release](#).

### ***baseline\_template***

**Set Option:** Model object attribute or `ccm set` command or **Options** dialog box

Specifies the version template to be used for project and products in a baseline when none is explicitly specified in the create baseline or modify baseline operation.

The default is `%{version}_%date`

Use the `ccm set` command to change the template to be used. The setting is persistent and applies to all sessions on all clients for the given user in the given database.

This option is available in the **Options** dialog box. This setting is also persistent and applies to all sessions on all clients for the given user in the given database.

The syntax for a baseline template is defined in the [baseline command](#).

For information about how to set model attribute options, see [Setting model object attribute options](#).

---

For information about how to set options using the [set command](#), see [Setting options using the ccm set command](#).

### ***baseline\_template\_date\_format***

**Set Option:** Model object attribute or `ccm set` command or **Options** dialog box  
Specifies the date format to be used when creating a baseline when expanding the `date` keyword in the `baseline_template`.

The default is `= %Y%m%d`

Use the `ccm set` command to change the date format to be used. The setting applies to all sessions on all clients for the given user in the given database.

This option is available in the **Options** dialog box. This setting is also persistent and applies to all sessions on all clients for the given user in the given database.

For information about how to set model attribute options, see [Setting model object attribute options](#).

For information about how to set options using the [set command](#), see [Setting options using the ccm set command](#).

### ***baseline\_template\_repl\_char***

**Set Option:** Model object attribute or `ccm set` command or **Options** dialog box  
Sets the default version string replacement character that is used if the instantiated `version_template` for any project or product in the baseline contains characters that are not allowed in a version string.

The default is the underscore character (`_`).

For example, if `%platform` is part of a project version template, and the build management project has a platform of `SPARC-solaris`, then the version string contains the string `SPARC_solaris`. Or, if `%release` is part of a product version template, and the prep product has a release of `CM/6.5`, then the version string contains the string `CM_6.5`.

Use the `ccm set` command to change the character to be used. The setting applies to all sessions on all clients for the given user in the given database.

This option is available in the **Options** dialog box. This setting is also persistent and applies to all sessions on all clients for the given user in the given database.

For information about how to set model attribute options, see [Setting model object attribute options](#).

For information about how to set options using the [set command](#), see [Setting options using the ccm set command](#).

### ***check\_release***

**Set Option:** Model object attribute

Compares the release values of an object and its associated task to ensure they are the same. If the values do not match, a message is written to the Message View (`ccm_ui.log`) informing you of the mismatch.

The default is `TRUE`.

For information about how to set model attribute options, see [Setting model object attribute options](#).

### ***cli\_compare\_cmd***

#### ***cli\_proj\_compare\_cmd***

#### ***cli\_symlink\_compare\_cmd***

#### ***cli\_merge\_cli***

**Set Option:** System or personal `ini` file, or object, or object type attribute or `ccm set` command

`cli_compare_cmd` is the default command that is executed when two normal files are compared from the CLI. A normal file is an object that is not a project, directory, or symbolic link. It defaults to the `cli_compare_cmd` attribute on the first object selected, which is, by default, `ccm_dff -o %outfile %file1 %file2`.

`cli_proj_compare_cmd` is the default command that is executed when two projects are compared from the CLI. It defaults to the `cli_compare_cmd` attribute on the first project selected, which is, by default, `sdiff -w 80 %file1 %file2`.

`cli_dir_compare_cmd` is the default command that is executed when two directories are compared from the CLI. It defaults to the `cli_compare_cmd` attribute on the first directory, which is, by default, `%ccm_merge`.

`cli_symlink_compare_cmd` is the default command that is executed when two symbolic links are compared from the CLI. It defaults to the `cli_compare_cmd` attribute on the first symbolic link, which is, by default, `ccm_dff -o %outfile %file1 %file2`.

`cli_merge_cmd` is the default command that is executed when two normal files are merged from the CLI. A normal file is an object that is not a project, directory, or symbolic link. It defaults to the `cli_compare_cmd` attribute on the first object selected, which is, by default, `%ccm_merge`.

On Windows, defaults for all four of these options are specified on the system initialization file, overriding the defaults on the object types. The Windows defaults for all four options are the same: `ccm_dff -o %outfile %file1 %file2`

All lines in your initialization file that reference the `ccm_dff` command use `-o %outfile` to write the results of the diff to `%outfile` rather than to standard output. (`%outfile` is the file that contains the results of the two diff'd files.)

For information about how to set options in the system or personal `ini` file, see [Setting options in the system or personal ini file](#).

---

For information about how to set object type options, see [Setting object type attribute options](#).

For information about how to set options using the [set command](#), see [Setting options using the ccm set command](#).

### ***cli.text\_editor***

**Set Option:** System or personal `ini` file, or `ccm set` command

Specifies the text editor used to modify the source for an object. The `cli` prefix indicates that the default is for command line use. The user interface uses this variable to determine which tool to use to edit a text attribute. Be sure to include the full path name to the program, or the directory must be included in your path.

The default GUI and CLI text editor is `Notepad` on Windows and `vi` on UNIX.

For information about how to set options in the system or personal `ini` file, see [Setting options in the system or personal ini file](#).

For information about how to set options using the [set command](#), see [Setting options using the ccm set command](#).

### ***cli.text\_viewer***

**Set Option:** System or personal `ini` file, or `ccm set` command

Specifies the text editor used to view the source for an object. The `cli` prefix indicates that the default is for command line use.

The default CLI text viewer is `Notepad` on Windows and `vi` on UNIX.

For information about how to set options in the system or personal `ini` file, see [Setting options in the system or personal ini file](#).

For information about how to set options using the [set command](#), see [Setting options using the ccm set command](#).

***conflict\_exclude\_rules*****Set Option:** Model object attribute

This attribute will cause the new conflicts to be excluded, depending on attribute values of the conflicting objects. The following syntaxes are supported:

Syntax	Description
<code>attrname=attrvalue</code>	Excludes any conflict where the <code>attrname</code> object attribute value matches <code>attrvalue</code> .
<code>attrname!=attrvalue</code>	Excludes any conflicts where the <code>attrname</code> object attribute value does not match <code>attrvalue</code> .
<code>EXISTS(attrname)</code>	Excludes any conflicts where the object has an attribute named <code>attrname</code> .
<code>NOT_EXISTS(attrname)</code>	Excludes any conflicts where the object does not have an attribute named <code>attrname</code> .
<code>MATCHING(attrname)</code>	Excludes any conflicts where the <code>attrname</code> object attribute value matches that of the project.
<code>NOT_MATCHING(attrname)</code>	Excludes any conflicts where the <code>attrname</code> object attribute value does not match that of the self version.

The default value of `conflict_exclude_rules` is unset.

**Additional Information:**

- The `!` and `!=` rules support values of type string and boolean.
- The values specified in the rules cannot contain the new line character.
- None of the rules should contain character sequences `=` or `!=`, except as delimiters for the equal/not-equal rules.
- Any lines that the parser does not understand will be ignored.
- Quotes are not needed around attribute names or string values, and should not be used. If present, they will be considered as literals, in other words, part of the name or value.
- The value of the `conflict_exclude_rules` attribute is cached in the model code, so if the rules are changed, user with active sessions will need to restart their sessions to get the new value.
- This attribute must be set manually, or through a model install. No customization interface is provided.

---

For information about how to set model attribute options, see [Setting model object attribute options](#).

### ***conflict\_parameters***

**Set Option:** Model object attribute.

Specifies which types of conflicts users in this database will see when they show conflicts for a project. The default value of the attribute lists each type of conflict and whether or not conflicts of that type are displayed when you request to see conflicts for a project.

The default editor displays the attribute settings, which contain one conflict setting per line. The line has the following format: `conflict_number: TRUE|FALSE`. Lines that begin with the pound character (#) are treated as comments.

The conflict default values for this option are:

```
# No task associated with object
1: TRUE
# Multiple tasks associated with object
2: FALSE
# Implicitly included object
3: FALSE
# Object included by use operation?
4: TRUE
# Object implicitly required but before baseline
5: FALSE
# Object implicitly required but not included - newer
6: TRUE
# Object implicitly required but not included - parallel
7: TRUE
# Object explicitly specified but before baseline
8: FALSE
# Object explicitly specified but not included - newer
9: TRUE
# Object explicitly specified but not included - parallel
10: TRUE
# Object explicitly specified but no versions of object in project
11: FALSE
# Object implicitly required but no versions of object in project
12: FALSE
# Task implicitly included
13: TRUE
# Task implicitly required but not included
14: TRUE
# Task explicitly specified but not included
15: TRUE
# Task explicitly specified but none of its associated objects
# in project
16: FALSE
```

```
# Excluded task explicitly included
17: TRUE
# Excluded task implicitly included
18: TRUE
# Completed fix task not included
19: TRUE
# Assigned fix task not included
20: FALSE
# Task fixed by this task not included
21: FALSE
# Implicit task from explicit object
22: TRUE
# Implicitly required by multiple tasks - newer
23: TRUE
# Implicitly required by multiple tasks - parallel
24: TRUE
```

For information about how to set model attribute options, see [Setting model object attribute options](#).

### ***copy\_db\_always***

**Set Option:** System or personal ini file

On Windows, forces a database copy when set to TRUE.

On UNIX, forces a database copy on `ccm start -rc` when set to TRUE.

The default for `copy_db_always` is unset, which causes a database copy to occur only when the `_timetag` file has been touched.

For information about how to set options in the system or personal ini file, see [Setting options in the system or personal ini file](#).

### ***date\_modified***

**Set Option:** Model object attribute

Creates a keyword to indicate the last time the file was modified. The default model does not include a `date_modified` keyword. You can create this keyword and set its value to the current time by adding the line `date_modified current_time` to the selection set. This will indicate when the file was checked in, if the keyword is expanded on check in.

For information about how to set model attribute options, see [Setting model object attribute options](#).

### ***dcm\_broadcast\_dbid***

**Set Option:** Model object attribute



---

Creates a database ID used as an identifier in order to receive transfer packages for the correct database. If `dcm_broadcast_dbid` is set to a non-blank string, DCM initialize automatically creates a DCM database definition for the broadcast database using the value of that attribute as the DCM database identifier. If `dcm_broadcast_dbid` is set to a non-blank string, then DCM receives DCM transfer packages that were generated for a matching DCM broadcast database ID.

The default setting is `TRUE`.

For information about how to set model attribute options, see [Setting model object attribute options](#).

### ***dcm\_log\_enabled***

**Set Option:** Model object attribute

Specifies that a `dcm_log` attribute be created and updated after the import phase of a DCM receive. This will show each object that DCM tells import or XML import to process. Each line will be of the form:

```
<action> from transfer set "<tset>" from database <dbid> on <date>
```

where `<action>` is one of the following:

created

updated (<A|R|AR[I])

A is attributes eligible for update

R is relations eligible for update

I is image handling

`<tset>` is the transfer set name

`<dbid>` is the database ID

The `dcm_log` attribute will be excluded by export and XML export, and ignored by import and XML import. It will not be copied on checkout.

The default setting is `FALSE`. This option is intended for use by IBM Rational Software Support to assist customers in debugging DCM issues. If you do not need to debug a DCM issue, this option can remain disabled.

For information about how to set model attribute options, see [Setting model object attribute options](#).

### ***dcm\_time\_sync\_tolerance***

**Set Option:** Model object attribute

Controls the amount of time (in seconds) subtracted from the server current time to compensate for different time settings between machines accessing the database. For additional information about synchronizing servers used in DCM transfers, see "Synchronize Engines and Servers" in [Rational Synergy Distributed](#).

The default setting is 60 seconds.

For information about how to set model attribute options, see [Setting model object attribute options](#).

### ***default\_task\_query***

**Set Option:** System or personal `ini` file

Specifies a user-defined query that can be used to specify a folder query. You can modify this default query by using query values, which are described in "Query expressions" in [Rational Synergy CLI Help, Traditional mode](#).

When you specify the `default_task_query` option in the `ini` file, the `ccm folder` command `task_scope` option contains a `user_defined` value. Selecting the `User Defined` value causes the query specified by the `default_task_query` option to be part of the folder query.

When the `default_task_query` is specified in the `ini` file, the `-task_scope` option of the `ccm folder` command supports the `user_defined` value. Using `task_scope user_defined` with the `ccm folder` command will use the query specified by the `default_task_query` option as part of the folder query.

For information about how to set options in the system or personal `ini` file, see [Setting options in the system or personal ini file](#).

### ***default\_version***

**Set Option:** Model object attribute

Specifies the default string for the first version of an object. Use this option to specify an alternative first version string, such as `0001`.

The default for `default_version` is `1`.

For information about how to set model attribute options, see [Setting model object attribute options](#).

### ***engine\_host***

**Set Option:** System or personal `ini` file

Specifies the machine on which the Rational Synergy engine runs.

The default for `engine_host` is unset.

This option is not used by the Rational Synergy GUI.

For information about how to set options in the system or personal `ini` file, see [Setting options in the system or personal ini file](#).

---

### ***expand\_on\_checkin***

**Set Option:** Set on the cvtype

Adds an attribute to any cvtype, which will cause keywords to expand on the specified cvtype. Keyword expansion normally occurs at check out, but with this option will occur at checkin. You must be a database administrator to use this option.

To force keyword expansion at checkin for objects of a certain type, add the `expand_on_checkin` attribute to any type. For example, to enable it for all text type objects, add the `expand_on_checkin` attribute to the `ascii` cvtype; this value will be inherited by all objects in the `ascii` hierarchy.

The default for `expand_on_checkin` is unset. This option is boolean, so it must be set to either `TRUE` or `FALSE`.

An example of how to use this for the `ascii` type is

```
$ ccm set role ccm_admin
$ ccm query -t cvtype -n ascii
$ ccm attr -c expand_on_checkin -t boolean -v TRUE @1
```

### ***html\_browser***

**Set Option:** System or personal `ini` file

Specifies the HTML browser used to view Rational Synergy HTML help. The value must be the fully qualified path to the executable; for example:

```
html_browser = /usr/local/bin/netscape
```

The default on Windows is `ccm_exec`; on UNIX the default is `ccm_browser`.

This option is not used by the Rational Synergy GUI.

For information about how to set options in the system or personal `ini` file, see [Setting options in the system or personal ini file](#).

### ***html\_location***

**Set Option:** System or personal `ini` file

Specifies an alternative Rational Synergy HTML help file location. The value can be an Internet location (for example, `www.rational.com/new_help`).

The default is `$(CCM_HOME)/help`.

For information about how to set options in the system or personal `ini` file, see [Setting options in the system or personal ini file](#).

### ***include\_required\_tasks***

**Set Option:** Model object attribute

Specifies that when a task is added to Added Tasks of a project grouping, the required tasks on which the task depends are computed and added as well.

The default is unset.

For information about how to set model attribute options, see [Setting model object attribute options](#).

### ***initial\_role***

**Set Option:** System or personal `ini` file

Specifies the role with which you start the Rational Synergy CLI. The role and user name determine the access that you have to the objects in the system.

Besides setting your role in your `ini` file, you can change roles by executing the `ccm set` command (described in [role](#)). When using the `ccm set` command, the variable name is `role`.

The default for `initial_role` is unset.

**Note** To change roles successfully using the `ccm set` command, be sure you have privileges for the role you are changing to; otherwise the command will fail.

For information about how to set options in the system or personal `ini` file, see [Setting options in the system or personal ini file](#).

### ***initials***

**Set Option:** System or personal `ini` file

Specifies the default next version for a project or product object to be the initials you specify. The next version will default to `initials` when you check out a project or product object if the new project or purpose has a private purpose. If this option is unset, the default next version for private projects and products is your user name. (The default next version is numeric for all other purposes, regardless of the `initials` option setting.)

To change the default next version to use your initials, enter the following in the system or personal initialization file:

```
initials=your_initials
```

For example:

```
initials=leb
```

The default for `initials` is unset.

For information about how to set options in the system or personal `ini` file, see [Setting options in the system or personal ini file](#).

---

## ***mail\_cmd***

**Set Option:** System or personal *ini* file

Rational Synergy uses a default mail tool for DCM e-mail notification. If you want to use your own mailer instead of the Rational Synergy mailer, enter the following line in the [Options] section in your *ini* file.

```
mail_cmd = user-defined_mail_command
```

The syntax for *user-defined\_mail\_command* depends on the mailer you want to use. However, your mailer typically will require recipients, subject, and content options and arguments. For example, the following is a *mail\_cmd* definition for *ccmail*:

```
mail_cmd = C:\ccmail\mailer.exe -r %recipients -s %subject -f %content
```

The *%recipients*, *%subject*, and *%content* arguments are expanded automatically by Rational Synergy, which uses the information you supply in your dialogs.

For information about how to set options in the system or personal *ini* file, see [Setting options in the system or personal ini file](#).

## ***multiple\_local\_proj\_instances***

**Set Option:** Model object attribute

Sets the behavior on project creation. Normally, if a project (any version) is created and another instance of that project exists and that project is local to the database, the create will fail. Multiple instances of the same project cannot be created locally. However, if a non-local project has been received from another database, this does not prevent a local project from being created with the same name.

If this attribute is set to `TRUE`, multiple local project instances are allowed. In a non-DCM initialized database, whenever a project is created, it will start with an instance of 1; if that instance already exists, the next available instance number is used. If the user specifies a project without an instance number, then 1 is assumed by default.

The default is `FALSE`.

For information about how to set model attribute options, see [Setting model object attribute options](#).

***parallel\_exclude\_rules***

**Set Option:** Model object attribute

Contains a set of rules defining which version will be excluded form parallel notification.

The following syntaxes are supported:

Syntax	Description
<code>attrname=attrvalue</code>	Excludes any parallel CVs whose <code>attrname</code> attribute value matches <code>attrvalue</code> . Example: <code>status=rejected</code>
<code>attrname!=attrvalue</code>	Excludes any parallel CVs whose <code>attrname</code> attribute value does not match <code>attrvalue</code> . Example: <code>is_product!=TRUE</code>
<code>EXISTS(attrname)</code>	Excludes any parallel CVs that have an attribute named <code>attrname</code> . Example: <code>EXISTS(is_product)</code>
<code>EXCLUDE_NON_LEAF_NODES</code>	Excludes any non-leaf nodes on parallel versions.
<code>NOT_EXISTS(attrname)</code>	Excludes any parallel CVs that don't have an attribute named <code>attrname</code> . Example: <code>NOT_EXISTS(is_product)</code>
<code>MATCHING(attrname)</code>	Excludes any parallel versions for which the <code>attrname</code> attribute value matches that of the self version. Example: <code>MATCHING(release)</code>
<code>NOT_MATCHING(attrname)</code>	Excludes any parallel versions for which the <code>attrname</code> attribute doesn't match that of the self version. Example: <code>NOT_MATCHING(release)</code>

The default value of this attribute is:

```
status=rejected
is_product=TRUE
EXCLUDE_NON_LEAF_NODES
NOT_MATCHING(release)
```

---

Additional Information:

- The ! and != rules support values of type string and boolean.
- The values specified in the rules cannot contain the new line character.
- None of the rules should contain character sequences = or !=, except as delimiters for the equal/not-equal rules.
- Any lines that the parser does not understand will be ignored.
- Quotes are not needed around attribute names or string values, and should not be used. If present, they will be considered as literals (in other words, part of the name or value).
- This attribute must be set manually, or through a model install. No customization interface is provided.
- Some customers may want to add the following rules to the default value, to exclude parallel variant branches from notification:

```
NOT_MATCHING(release)
NOT_MATCHING(platform)
```

- The rule `MATCHING(owner)` should not be used. This rule will not work for checkout, because it uses the version you are deriving *from* to detect parallels.

For information about how to set model attribute options, see [Setting model object attribute options](#).

### ***proj\_idx\_wa\_cache***

**Set Option:** System or personal `ini` file

Controls the size of the second work area path cache. The default value is 2500. Users can increase this value to improve the performance of file accesses in very large projects.

For information about how to set options in the system or personal `ini` file, see [Setting options in the system or personal ini file](#).

### ***project\_subdir\_template***

**Set Option:** Model object attribute or `ccm set` command or **Options** dialog box

Changes the default template defining the project-specific directory of your work area path. Setting this option affects the work area path for projects created after the setting has been saved; it does not change the work area path of existing projects.

When changing the value of this option from the command line, you set the `project_subdir_template` variable. This automatically sets the option for the given platform where your interface is running - either UNIX or Windows. The setting is persistent and applies to all sessions on all clients for the given user in the given database.

When changing the model-wide default setting, you need to append `_unix` or `_windows` to the name of the attribute, to indicate whether the template applies to Windows or UNIX work areas. For example, to set a model-wide template for UNIX work areas, you would create an attribute called `project_subdir_template_unix`.

This option is available in the **Options** dialog box as **Add project-specific directory**. This setting is also persistent and applies to all sessions on all clients for the given user in the given database.

The following keywords are valid:

- `%project_name` replaces `%project_name` with the new project name.
- `%project_version` replaces `%project_version` with the new version.
- `%release` replaces `%release` with the new release value.
- `%platform` replaces `%platform` with the new platform name.
- `%delimiter` replaces `%delimiter` with the new delimiter.

The default is `%project_name%delimiter%project_version`.

If you need to change the non-project-specific portion of the work area path, see [wa\\_path\\_template](#).

For information about how to set model attribute options, see [Setting model object attribute options](#).

For information about how to set options using the [set command](#), see [Setting options using the ccm set command](#).

### ***range\_for\_keyword\_expand***

**Set Option:** System or personal `ini` file

Establishes how many characters in a file will be scanned for keywords when an object is created or derived, starting from the beginning of the file.

When you check out a file, it scans the file and replaces keywords with values. If you have a large file with keywords defined in all parts of the file, the amount of time to scan the entire file can cause the create or check-out operation to be very slow.

The default number is 20480, which refers to the maximum number of characters that will be scanned for keywords. (If your file is set up for 80 characters per line, the default setting will allow at least the first 256 lines per file.)

The default setting works well if you have all of your keywords in the header area. If the keywords are spread throughout your file, you will need to reset this preference so that the keyword expansion can be done throughout the file.

For information about how to set options in the system or personal `ini` file, see [Setting options in the system or personal ini file](#).



---

### ***reconcile.control\_files\_below\_new\_project***

**Set Option:** System or personal `ini` file

Specifies whether uncontrolled files are added to new projects derived from directories during the reconcile operation.

The default is `FALSE`.

For information about how to set options in the system or personal `ini` file, see [Setting options in the system or personal ini file](#).

### ***reconcile.save\_uncontrolled***

**Set Option:** System or personal `ini` file

Specifies whether uncontrolled files that are removed from the work area due to conflict resolution should be saved in the work area wastebasket. Setting the option to `TRUE` will store an uncontrolled file in the work area wastebasket if the file is removed from the work area by an Update Work Area from Database resolution.

The default is `FALSE`.

For information about how to set options in the system or personal `ini` file, see [Setting options in the system or personal ini file](#).

### ***reconf\_consider\_all\_cands***

**Set Option:** Model object attribute.

Specifies that a directory be populated with the best match when there are no candidates in the update properties for a project. If this attribute does not exist or if the value is `FALSE`, the directory entries are left empty when there are no candidates in the update properties for a project.

The default is `FALSE`.

For information about how to set model attribute options, see [Setting model object attribute options](#).

### ***reconf\_release\_score***

**Set Option:** Model object attribute.

Specifies that release scoring be used to select object versions whose release best matches the project's release. Release scoring is not used by default for task-based update, but may be considered if you develop parallel releases and one release includes the other release's changes. As there are definite caveats to using this option, it should be used only after serious consideration.

The default is `FALSE`.

For information about how to set model attribute options, see [Setting model object attribute options](#).

### ***reconf\_stop\_on\_fail***

**Set Option:** System or personal `ini` file

Stops the update process when an individual operation fails. When set to `TRUE`, update stops if an individual operation within the update fails. When set to `FALSE`, the update process continues if an individual operation fails, allowing you to find all errors at one time.

The default is `TRUE`.

For information about how to set options in the system or personal `ini` file, see [Setting options in the system or personal ini file](#).

### ***reconfigure\_parallel\_check***

**Set Option:** System or personal `ini` file

Indicates whether parallel version notification is given on update.

This option can have the value `FALSE`, `TRUE`, or `FULL`. If set to `FALSE` or not specified, no parallel detection is done. If set to `TRUE`, parallel detection is done only among the candidates chosen by the update selection rules. This setting shows parallel versions specified by the saved baselines and tasks. If set to `FULL`, parallel detection is done among all version of the selected object.

The default is `FALSE` (no notification).

For information about how to set options in the system or personal `ini` file, see [Setting options in the system or personal ini file](#).

### ***reconfigure\_using\_tasks***

**Set Option:** Model object attribute

Indicates whether you are using task-based Rational Synergy to update projects. This setting applies to your entire database.

The default is `TRUE`.

For information about how to set model attribute options, see [Setting model object attribute options](#).

---

### ***release\_phase\_list***

**Set Option:** Model object attribute

Defines the various phases of development or deployment of a release. This feature tracks the status of a release during the development process. You can customize this list to match the development phases of your products, or use the default list. The default phase list contains the following phases: `New`, `Requirements Definition`, `Function Definition`, `Implementation`, `Validation`, and `Released`.

The model attribute is formatted with each entry on a separate line. The default value when a release is created is the first value in the list.

For information about how to set model attribute options, see [Setting model object attribute options](#).

### ***replace\_subproj***

**Set Option:** System or personal ini file

Indicates whether the update operation replaces subprojects as default behavior.

This option can have the value `TRUE` (replace subprojects during update), or `FALSE` (do not replace subprojects).

The default is `TRUE` (that is, replace subprojects).

This option is used by the CLI and Rational Synergy Classic, but is not used by the Rational Synergy GUI.

There is another option in the **Options** dialog box for replacing subprojects in the GUI. That option only applies to the GUI.

For information about how to set options in the system or personal `ini` file, see [Setting options in the system or personal ini file](#).

### ***required\_attributes***

**Set Option:** Object type attribute.

Specifies whether users are required to fill in certain fields prior to transitioning an object of the given type to a static state. If one of the required fields is missing or contains an illegal value, the object will not transition to the static state.

The contents of this attribute should be the name of the required attributes, one per line. For example, if you want `release`, `task_description` and `priority` to be required fields on a task, you should specify:

```
release
task_description
priority
```

The task will not be able to be completed unless the attributes specified each have a valid value.

The default is an empty string.

For information about how to set object type attribute options, see [Setting object type attribute options](#).

### ***restrict\_reconf\_setting***

**Set Option:** Model object attribute.

Specifies whether developers can change their projects' update properties from "object status" to "tasks," and vice-versa. This option also controls whether the update properties can be set at the time a project is created.

**Note** With a setting of `FALSE`, each user can change his update properties whenever he wants, which could result in unexpected build results. If set to `FALSE`, be sure teams agree on the type of update properties to use.

By default this option is set to `TRUE`, and developers are restricted from changing the update properties setting. This option is a model object attribute. You must be working as a build manager or be in the `ccm_admin` role to set or change this option.

For information about how to set model attribute options, see [Setting model object attribute options](#).

### ***role***

**Set Option:** System or personal `ini` file, or `ccm set` command

Specifies the default role for using the Rational Synergy CLI.

To change the default role in your initialization file, use the [initial\\_role](#) option.

To change roles successfully using the `ccm set` command, be sure you have privileges for the role you are changing to; otherwise, the command will fail.

The default is *developer*.

This option does not affect the Rational Synergy GUI.

For information about how to set options in the system or personal `ini` file, see [Setting options in the system or personal ini file](#).

For information about how to set options using the [set command](#), see [Setting options using the ccm set command](#).

### ***role\_definitions***

**Set Option:** Model object attribute

This attribute on the model object specifies what privileges are available to users in the various roles.

You would also modify this attribute to:

- Change the default roles that are allowed to modify a process rule, by adding or removing the privilege `PRIVILEGE_MANAGE_PROCESS_RULES` from a given role.

- 
- Add a new role to manage releases by adding the privilege `PRIVILEGE_MANAGE_RELEASES` to the new role.
  - Change the default roles that are allowed to create and assign tasks, by adding or removing the privilege `PRIVILEGE_CREATE_AND_ASSIGN_TASKS` from a given role.
  - Change the default roles that are allowed to assign DCM tasks, by adding or removing the privilege `PRIVILEGE_ASSIGN_FOREIGN_TASKS` from a given role.

After modifying this attribute, you need to restart your sessions.

For information about how to set model attribute options, see [Setting model object attribute options](#).

### ***save\_to\_wastebasket***

**Set Option:** System or personal `ini` file

Causes any uncontrolled file in your work area that needs to be removed to be moved to a wastebasket directory. If the `update_db_from_workarea` option is `TRUE`, files involved in collisions with controlled files are copied to the database, not to the wastebasket.

The default is `TRUE`.

For information about how to set options in the system or personal `ini` file, see [Setting options in the system or personal ini file](#).

### ***shared\_project\_directory\_checkin***

**Set Option:** System or personal `ini` file

Causes non-writable directories in shared projects to be checked in to the *integrate* state automatically when objects are added to or deleted from such directories.

The default is `TRUE`.

For more information about shared projects, see "Shared projects" in [Rational Synergy CLI Help, Traditional mode](#).

For information about how to set options in the system or personal `ini` file, see [Setting options in the system or personal ini file](#).

### ***start\_day\_of\_week***

**Set Option:** Model object attribute

Specifies the start day of the week to be used when calculating queries that use relative time keywords: `%this_week_begin`, `%this_week_end`, `%last_week_begin`, and `%last_week_end`. Valid entries are integers from 0 - 6, with 0 being Sunday, 1 being Monday, etc.

The default is 0.

For information about how to set model attribute options, see [Setting model object attribute options](#).

### **system\_filename\_filters**

**Set Option:** Model object attribute.

Specifies the database default file patterns to be ignored when users sync the work area. This is set by the CM administrator (the *ccm\_admin* role) and is used by Rational Synergy.

If certain files are being ignored in the work area based on their extension, and they should be included, the solution is to remove the extension from the `system_filename_filters` attribute.

The table below lists the default filters.

<b>Default filters</b>	
*.APS	*.BAK
*.BSC	*.class
*.CLW	*.ENC
*.EXP	*.IDB
*.ILK	*.INCR
*.MD#	*.MD~
*.MD%	*.NCB
*.OBJ	*.OPT
*.PCH	*.PDB
*.PLG	*.PROJDATA
*.RES	*.SBR
*.SUO	*.TLH
*.TLI	*.TMP
*.USER	*.WBK
_vti*	_vti*\*
~*	*~
Copy of *	New Folder
timestamp.inf	

---

For information about how to set model attribute options, see [Setting model object attribute options](#).

### ***update\_on\_checkin\_if\_equal***

**Set Option:** System or personal *ini* file

When you use some editors or you perform scripted check-outs and check-ins, the timestamps on the database and work area versions of a file can appear to be identical. When set to `TRUE`, the `update_on_checkin_if_equal` option forces Rational Synergy to copy such files from the work area to the database, even though their timestamps indicate that they are not newer than their database versions.

The default is `FALSE`.

For information about how to set options in the system or personal *ini* file, see [Setting options in the system or personal ini file](#).

### ***verbosity***

**Set Option:** System or personal *ini* file

Specifies the default verbosity for messages output from the `ccm update` command. A level of 5 or greater causes additional information to be displayed by the update operation. The model your database uses also can use the verbosity level.

The default is 0 (zero), the lowest setting.

The **Options** dialog box also has a verbosity setting, which has the same effect as this setting.

For information about how to set options in the system or personal *ini* file, see [Setting options in the system or personal ini file](#).

### ***wastebasket***

**Set Option:** System or personal *ini* file

Use the `wastebasket` option to specify the location of your wastebasket directory.

`%database` becomes the name of the database for which the wastebasket is being used. (The wastebasket directory is hidden.)

`%database` and `%user` are keywords you can use to specify the wastebasket path to create directory names that differ for each user or database or user and database using the same template. These keywords are replaced at startup. If the directory does not exist, Rational Synergy creates it.

The default path resides in your home directory and is:

```
Windows: HOME\%user\ccm_wa\.moved\%database
UNIX:    $HOME/%user_name/ccm_wa/.moved/%database
```

where `%user` replaces `%user` with your user name.

For information about how to set options in the system or personal `ini` file, see [Setting options in the system or personal ini file](#).

### ***wa\_path\_cache\_size***

**Set Option:** System or personal `ini` file

Controls the size of the work area path cache. The default value is 500. Users can increase this value to improve the performance of file accesses in large projects.

For information about how to set options in the system or personal `ini` file, see [Setting options in the system or personal ini file](#).

### ***wa\_path\_template***

**Set Option:** Model object attribute

Changes the default template defining the non-project-specific directory of your work area path. Setting this option affects the work area path for projects created after the setting has been saved; it does not change the work area path of existing projects.

When changing the value of this option from the command line, you set the `wa_path_template` variable. This automatically sets the option for the given platform where your interface is running - either UNIX or Windows. The setting is persistent and applies to all sessions on all clients for the given user in the given database.

When changing the model-wide default setting, you need to append `_unix` or `_windows` to the name of the attribute, to indicate whether the template applies to Windows or UNIX work areas. For example, to set a model-wide template for UNIX work areas, you would create an attribute called `wa_path_template_unix`.

Set the following path using the `ccm set` command.

```
ccm set wa_path_template %home%\%database\location
```

The following keywords are valid:

- `%database` replaces `%database` with the new database name.
- `%user` replaces `%user` with your user name.
- `%owner` replaces `%owner` with the name of the project owner.
- `%home` replaces `%home` with your home directory.

The Windows default is `%home\ccm_wa\%database`, where `%home` is your home directory (wherever you designated that to be in the **Startup** dialog's **Home Directory** text box).

The UNIX default is `%home/ccm_wa/%database`, where `%home` is your UNIX home directory.

This option is available in the **Options** dialog box as **Set default path for all work areas**. This setting is also persistent and applies to all sessions on all clients for the given user in the given database.

If you need to change the project-specific portion of the work area path, see [project subdir template](#). For information about how to set model attribute options, see



---

[Setting model object attribute options](#). For information about how to set options using the `set` command, see [Setting options using the `ccm set` command](#).

## Initialization file - Windows

### ***Run Rational Synergy from the PC server***

Typically, the `ccm.ini` file is located in two places: the system file is located in the product installation area, `CCM_HOME\etc`, and a personal file usually exists in the Windows Documents and Settings directory for each user. If you do not have a `ccm.ini` file in your Windows Documents and Settings directory, you can copy the system file, and then modify it with the options you want to set.

The personal `ccm.ini` file overrides the system file.

You can edit your `ccm.ini` file by using any text editor.

### ***Run Rational Synergy on your PC***

If you are running Rational Synergy from an installation on your PC, the `ccm.ini` file is in `CCM_HOME\etc`. You can modify this file directly since it will affect your machine only.

## Initialization file - UNIX

The default `ccm.ini` file is located in `$CCM_HOME/etc/ccm.ini`. You can copy the default file to your home directory, and then modify it for your use. After you do this, the settings in your `.ccm.ini` file will override those of the system `.ccm.ini` file. Even if you haven't changed any settings, if you don't already have a personal `.ccm.ini` file, it is created automatically when you exit from your session.

You can create and edit your personal `.ccm.ini` file by using any text editor. The `.ccm.ini` file must reside in your home directory. Options that are preceded with the word *Motif* affect only the graphical user interface. When no interface is specified, the option applies to all interfaces, as appropriate.

---

## Environment variables

You can define the following variables to affect the way Rational Synergy runs. The following table shows environment variables that you can set..

Environment variable	Required	Use
AUTOMOUNT_FIX (UNIX only)	No	Used to determine the portion of path names that should be stripped to support automounter usage. Not needed if the <code>/tmp_mnt</code> prefix is used by the automounter. This variable is also used by SunOS automounter patch. For more information, refer to the <a href="#">Rational Synergy Administration Guide for UNIX</a> .
CCM_ADDR	No	Specifies the remote function call (RFC) address ( <code>host:socket</code> ) for the Rational Synergy interface.
CCM_ENGLOG	No	Used to redirect the engine log. If not set, the <code>ccm_eng.log</code> file in your Windows installation directory or UNIX home directory is used. The engine log file must be visible and writable by <code>ccm_root</code> .
CCM_HOME	No	Specifies the Rational Synergy installation directory, typically <code>C:\Program Files\IBM\Rational\Synergy\7.1</code> for a Windows client, and <code>/usr/local/ccm</code> on UNIX.
CCM_PAGER (UNIX only)	No	Specifies the name of an executable used to display a file or report output using <code>ccm monitor</code> on UNIX. This takes preference over <code>PAGER</code> , if set.
CCM_UILOG	No	Used to redirect the user interface log. If not set, the <code>ccm_ui.log</code> in your installation directory on Windows or home directory on UNIX is used.
DISPLAY (UNIX only)	Yes	Name of X display server; for example, <code>unix:0.0</code> .
HOME (UNIX only)	Yes	Specifies your UNIX home directory.
LD_LIBRARY_PATH (Sun only)	Yes	Specifies a list of directories used to search for dynamic object libraries, for example: <code>/usr/lib/X11:/usr/openwin/lib</code>

Environment variable	Required	Use
PAGER (UNIX only)	No	Specifies the name of an executable used to display a file or report output using <code>ccm monitor</code> on UNIX.
PATH	Yes	Specifies a list of directories used to search for executables.
PRINT_EDIT_CMD	No	If this variable is set to a value, the model-defined editor method that is being used is displayed whenever the editor command is executed.
PRINT_TOOL_CMD	No	If this variable is set to any value at all, the model-defined tool method (for example, <code>debug</code> , <code>print</code> , or <code>execute</code> ) is displayed whenever the tool command is executed.
RECONF_TIME	No	Times and displays the execution of an update command.
SHELL (UNIX only)	Yes	Specifies the name of a UNIX command interpreter shell to invoke for subprocesses, for example, <code>/bin/csh</code> .
TERM (UNIX only)	Yes	Specifies the type of terminal from which UNIX commands are entered, for example, <code>xterm</code> .
UIDPATH (UNIX only)	No	Specifies a list of directories used to search for files.
USER (UNIX only)	Yes	Specifies your user name.

**Note** Rational Synergy uses other variables starting with `CCM_` or `AC_` for internal diagnostic purposes. **Do not** set any such variables, nor `INFORMIXDIR`, `INFORMIXSERVER`, or `ONCONFIG` variables, unless you are told to do so by IBM Rational Software Support.

---

## Setting model object attribute options

You can set model-wide attributes on model objects; these settings affect all users of databases into which that model has been installed. You must be in the *ccm\_admin* role to change model attribute objects.

The first example shows how to create the attribute, which is required in some cases. After creating the attribute, you can then set it. The second example shows how to modify an attribute that is already set.

Substitute the appropriate option name and syntax for the option you are changing.

### **Create an attribute**

This example uses the `allow_delimiter_in_name` attribute, which specifies whether the version delimiter is allowed in an object name. By default, this attribute does not exist, which means the delimiter is not allowed.

To set this option the first time, you must create the attribute as follows:

1. Set your role to *ccm\_admin*.

```
ccm set role ccm_admin
```

2. Query for the model object in the Rational Synergy database.

```
ccm query -t model -n base
```

3. Create the attribute.

```
ccm attr -c allow_delimiter_in_name -t boolean @1
```

4. Set the value.

```
ccm attr -m allow_delimiter_in_name -v TRUE @1
```

5. Restart your session. (All users of this database must restart their sessions.)

### **Modify an attribute**

This example uses the `wa_path_template_unix` attribute, which specifies the default non-project-specific work area directory. By default, this directory is `%home/ccm_wa/%database`.

To modify this attribute, do the following:

1. Set your role to *ccm\_admin*.

```
ccm set role ccm_admin
```

2. Query for the model object in the Rational Synergy database.

```
ccm query -t model -n base
```

3. Specify the new path.

```
ccm attr -m wa_path_template_unix @1 -v "%home/workareas/%database"
```

4. View the new contents of the attribute.

```
$ ccm attr -show wa_path_template_unix @1
```

5. Change back to your previous role.

```
ccm set role previous_role
```

6. Restart your session. (All users of this database must restart their sessions.)

---

## Creating a list box for a new attribute

You can create list boxes for newly created attributes. The syntax for creating new list boxes is as follows:

```
attr_name:attr_type[:[label][:#textlines]] |  
attr_name:attr_type[:[label][:#textlines]:values_ref]
```

where `values_ref` is defined in a new values definition entry, in a separate attribute.

Each `values_ref` values definition entry must be defined in a separate text attribute on an object or type called `info_attrs.values_ref`, where `values_ref` is the name of the list of values referred to in the `info_attrs` definition. This allows the list of values to be easily populated from external tools.

A `values_ref` must be a legal attribute name because it becomes part of the name of an attribute. A `values_ref` attribute name is limited to 21 characters because the limit on the length of an attribute name is 32 characters and 11 characters are used by the string, `info_attrs`.

The contents of an `info_attrs.values_ref` attribute must be a newline-separated list of possible values.

A value in a value list may be any ASCII string with embedded white space allowed, but no leading or trailing white space (since such white space will be considered part of the newline delimiter).

### Example

Suppose you want to add a custom attribute to the task type called `approval_level` and the possible values for this attribute are from the following list:

```
" new  
" pending  
" approved level 1  
" approved level 2
```

You could create an entry in the `info_attrs` attribute on the `task` type as follows:

```
approval_level:string:Approval Level::approval_values
```

You could then create an attribute on the `task` type called `info_attrs.approval_level` with the following contents:

```
new  
pending  
approved level 1  
approved level 2
```

## Setting object type attribute options

You can set object type attributes on objects; these settings affect all users of databases having that object type. You must be in the *ccm\_admin* role to change object types. Substitute the appropriate option name and syntax for the option you are changing.

1. Set your role to *ccm\_admin*.

```
ccm set role ccm_admin
```

2. Query for the type whose setting you want to change.

```
ccm query -t cvtype -n misc
```

3. Modify the attribute.

```
ccm attr -m required_attributes @1
```

This will start an editor on the attribute. Make your changes, and then save the value.

4. Change back to your previous role.

```
ccm set role previous_role
```

5. Restart your session. (All users of this database must restart their sessions.)



---

## Setting options in the system or personal ini file

Some options can be set by changing your personal `ini` file or using the `ccm set` command. Changing the option using your personal `ini` file will cause the change to be in effect every time you start a session. Changing the option using the `ccm set` command makes the change at the run-time level (that is, you do not need to restart a session for the change to take effect).

The following examples change the `cli.text_editor` option, the first in your Windows `ccm.ini` file, and the second, in your UNIX `.ccm.ini` file. Substitute the appropriate option name and syntax for the option you are changing.

- To specify `Notepad` as the default editor, your `ccm.ini` file should contain the following setting:

```
cli.text_editor=notepad %filename
```

- To specify `vi` as the editor, your `.ccm.ini` file should contain the following settings:

```
cli.text_editor="vi %filename"
```

## Setting options using the `ccm set` command

Some options can be set by changing your personal `ini` file or using the `ccm set` command. Changing the option using your personal `ini` file will cause the change to be in effect every time you start a session. Changing the option using the `ccm set` command makes the change immediately, (that is, you do not need to restart a session for the change to take effect).

The following example changes the `wa_path_template` option.

Substitute the appropriate option name and syntax for the option you are changing.

Enter the following to change the work area path template:

```
ccm set wa_path_template %home/workareas/%database
```

# Commands

## alias command

See [Description and uses](#) for details. The `alias` command supports the following subcommands:

- [Defining an alias](#)
- [Showing aliases](#)
- [Showing an alias](#)

---

## Defining an alias

Use this subcommand to define a new alias. The *alias\_name* is the name of the new alias to create.

```
ccm alias alias_name alias_string...
```

*alias\_name*

Specifies the name of the new alias you are defining.

*alias\_string*

Specifies a value to be used for the alias definition. If you specify a single *alias\_string*, it is split by white space to form the items for the alias definition. If you specify more than one argument after *alias\_name*, each argument is taken as an item for the alias, but is not split.

## Examples

- Create an alias to check out a file with a new version.

```
ccm alias getf "checkout -t"
```

When you use the new alias, it will be in the following form:

```
ccm getf myversion foo.c
```

- Change the value of an alias.

```
ccm alias alias_name "new alias value"
```

For example, assume you have an alias, *my\_query*, defined to query for objects. Now you want to change the value of *my\_query* to query for tasks. You would change the *my\_query* alias value to query for tasks by running the *alias* command.

- Define an alias named *vprop* with two items, *properties* and *-verbose*. (The *alias\_string* ["*properties -verbose*"] will be split by white space to form the items for the alias definition.)

```
ccm alias vprop "properties -verbose"
```

- Define an alias named *vprop* with two items, *properties* and *verbose*. (Each argument [*properties -verbose*] is taken as an item for the alias, but is not split.)

```
ccm alias vprop properties -verbose
```

## Related topics

- [Showing aliases](#)
- [Showing an alias](#)
- [unalias command](#)

### **Showing aliases**

Use this subcommand to show the defined aliases for the current session.

```
ccm alias
```

### **Example**

- List all defined aliases.

```
ccm alias
```

### **Related topics**

- [Defining an alias](#)
- [Showing an alias](#)
- [unalias command](#)

---

## ***Showing an alias***

Use this subcommand to show a specified alias.

```
ccm alias alias_name
```

*alias\_name*

Specifies the name of the alias to show.

## **Related topics**

- [Defining an alias](#)
- [Showing aliases](#)
- [unalias command](#)

## **Description and uses**

An alias represents a macro that you can use to create another name for an existing command or another alias. Aliases only exist for the current session. An alias might reference another alias and so on, providing this does not lead to a circular dependency. When Rational Synergy recognizes a command as an alias name, it expands the alias and replaces the alias name with its defined items. It repeats this expansion until the command is no longer an alias, or a circular reference is found.



## attribute command

See [Description and uses](#) for details. The `attribute` command supports the following subcommands:

- [Copying attributes](#)
- [Creating an attribute](#)
- [Deleting an attribute](#)
- [Modifying an attribute](#)
- [Showing attributes](#)
- [Listing attributes](#)

---

## Copying attributes

```
ccm attr|attribute -cp|-copy attr_names
                    [-append] from_object_spec to_object_spec
ccm attr|attribute -cp|-copy attr_names
                    [-append] [-subproj] [-suball]
                    -p|-project from_project_spec to_project_spec...
```

-append

Appends the specified attribute values to the specified object. If you do not use this option, any existing values for the specified attributes are overwritten by the new values.

-cp|-copy attr\_names

Copies an attribute or set of attributes to a selected set of object or project versions in a single operation. You can use the colon or white space or colon and white space as the separator character if you want to specify more than one attribute name.

*from\_object\_spec to\_object\_spec*

Specifies that *from\_object\_spec* is the file from which the attribute is copied and *to\_object\_spec* is the file to which the attribute is copied. You can specify one object for *from\_object\_spec* and multiple objects for *to\_object\_spec*.

*from\_project\_spec to\_project\_spec*

Specifies that *from\_project\_spec* is the project from which the attribute is copied and *to\_project\_spec* is the project to which the attribute is copied. You can specify one object for *from\_project\_spec* and multiple objects for *to\_project\_spec*.

If *-subproj* or *-suball* is used, the project is applied to *to\_proj\_spec*.

-suball

Recursively copies the specified attributes to subproject objects and all members of the specified project. This option applies to *to\_proj\_spec*, requires the *-p* option, and cannot be used with *-subproj*.

-subproj

Recursively copies the specified attribute or set of attributes to the subproject objects in the specified project. This option applies to *to\_proj\_spec*, requires the *-p* option, and cannot be used with *-suball*.

---

### **Example**

- Copy the `version` attribute from a project, `attr_test-1`, to its subprojects.

```
ccm attr -copy version -project attr_test-1 -subproj attr_test-1
```

---

## Creating an attribute

```
attr|attribute -c|-create attr_name -p|-project [-f|-force]
               -t|-type attr_type [-v|-value attr_value] project_spec...
attr|attribute -c|-create attr_name -t|-type attr_type
               [-v|-value attr_value] [-f|-force] object_spec...
```

`-c|-create attr_name`

Creates an attribute.

`-f|-force`

Checks whether the attribute to be created exists and has the same type, and then causes one of the following to occur:

- If the attribute to be created exists and has the same type, the attribute value is changed (if you use the `-value` option).
- If the attribute does not exist, the new attribute is created.
- If an attribute with the same name exists, but has a different type, the operation fails.

The difference between `ccm attr -c attr_name -t type` and `ccm attr -c attr_name -f -t type` is that the command without the `-force` option fails if the attribute already exists.

`-p|-project`

Specifies the project whose attribute will be created.

`-t|-type attr_type`

Specifies the type of the attribute. Use this option only when you create attributes. Valid built-in values include the following:

- `string` (used for single-line `ascii` attributes)
- `boolean`
- `text` (used for multi-line `ascii` attributes)

`-v|-value attr_value`

Specifies the value of the attribute.

## Example

- Create a string attribute named `new_attr` for the `driver.c` object.

```
ccm attr -c new_attr -type string driver.c
```

---

## Related topics

- [Copying attributes](#)
- [Listing attributes](#)
- [Showing attributes](#)

---

### **Deleting an attribute**

`attr|attribute -d|-delete attr_name -p|-project project_spec...`

`attr|attribute -d|-delete attr_name object_spec...`

`-d|-delete attr_name`

Deletes an attribute.

`-p|-project`

Specifies the project whose attribute will be deleted.

### **Example**

- Delete an attribute named `new_attr` from the `driver-1` project.

```
ccm attr -d new_attr -project driver-1
```

### **Related topics**

- [Listing attributes](#)
- [Showing attributes](#)

---

## ***Listing attributes***

attr|attribute -p|-project ([-l|-list] | [-la] | [-li]) *project\_spec...*  
attr|attribute ([-l|-list] | [-la] | [-li]) *object\_spec...*

-l

Lists all local attributes.

-la

Lists all attributes.

-li

Lists the inherited attributes.

-p|-project

Specifies the project whose attribute you want to list.

## **Related topics**

- [Copying attributes](#)
- [Deleting an attribute](#)
- [Modifying an attribute](#)
- [Showing attributes](#)

---

## **Modifying an attribute**

```
attr|attribute -m|-modify attr_name -p|-project [-v|-value attr_value]
                project_spec...
attr|attribute -m|-modify attr_name [-v|-value attr_value] object_spec...

-m|-modify attr_name
```

Modifies an attribute. If you do not specify the `-v` option, the default editor uses the appropriate attribute type to update the attribute on the specified objects.

```
-p|-project
```

Specifies the project whose attribute will be modified.

```
-v|-value value
```

See [-v|-value attr\\_value](#).

## **Example**

- Change the release attribute of `foo.c` to `4.2_int`.

```
ccm attr -m release -v 4.2_int foo.c
```

## **Related topics**

- [Listing attributes](#)
- [Showing attributes](#)



---

## Showing attributes

```
attr|attribute -s|-sh|-show attr_name -p|-project project_spec...  
attr|attribute -s|-sh|-show attr_name object_spec...
```

`-p|-project`

Specifies the project whose attribute you want to show.

`-s|-show attr_name`

Shows the value of an attribute.

## Example

- Show the value of the `comment` attribute for the `driver.c` object.

```
ccm attr -s comment driver.c
```

## Related topics

- [Copying attributes](#)
- [Deleting an attribute](#)
- [Listing attributes](#)
- [Modifying an attribute](#)

---

## Description and uses

Use the `attribute` command to manipulate the Rational Synergy attributes associated with objects.

## baseline command

See [Description and uses](#) for details. The `baseline` command supports the following subcommands:

- [Comparing baselines](#)
- [Creating or previewing a baseline](#)
- [Creating component tasks for a baseline](#)
- [Deleting a baseline](#)
- [Listing baselines](#)
- [Marking a baseline for deletion](#)
- [Modifying a baseline](#)
- [Publishing a baseline](#)
- [Releasing a baseline](#)
- [Restoring a deleted baseline](#)
- [Showing a baseline property](#)
- [Showing the projects, objects, tasks or change requests for a baseline](#)
- [Showing baseline information](#)

---

## Comparing baselines

This subcommand compares two baselines.

```
ccm baseline -compare [-tasks] [-objects] [-projects] [-change_requests]
                    baseline_spec1 baseline_spec2
```

*baseline\_spec1*

Specifies the first baseline to be compared. For more information, see [Baseline specification](#).

*baseline\_spec2*

Specifies the second baseline to be compared. For more information, see [Baseline specification](#).

`-cr` | `-change_request` | `-change_requests`

Specifies that the baseline comparison should include details about change requests (CRs) that are partially included and fully included in the two baselines.

The default format is `%displayname: %problem_synopsis`.

`-objects`

Specifies to include a comparison of object members in the baseline.

`-projects`

Specifies to include a comparison of the projects between the two baselines.

`-tasks`

Specifies to include a comparison of the tasks between the two baselines.

### Example

- Compare the projects that are in a baseline named 20020401\_1 and a baseline named 20020401\_2.

```
ccm baseline -compare 20020401_1 20020401_2 -projects
```

### Related topics

- [Listing baselines](#)
- [Modifying a baseline](#)

- 
- [Showing a baseline property](#)

---

## Creating or previewing a baseline

Creates or previews the creation of a baseline. Build managers or users in the *ccm\_admin* role can create baselines from one or more projects, baselines, or project groupings.

```
ccm baseline -c|-create [(-p|-project project_spec)...]
                    [(-bl|-baseline baseline_spec)...]
                    [(-pg|-project_grouping project_grouping_spec)...]
                    [-rehearse] [-r|-release release_spec] [-purpose purpose]
                    [-d|-desc|-description description]
                    [-vt|-version_template version_template] [-b|-build build]
                    [-s|-state state] ([-subprojects] | [-all_subprojects] |
                    [-no_subprojects]) [baseline_name]
```

**-all\_subprojects**

Specifies to include all subprojects that are members of the project being added. This option applies to projects added with the `-project`, `-project_grouping`, and `-baseline` options.

**-b|-baseline *baseline\_spec***

If you use this option with `-create` and specify a [Baseline specification](#), the projects in the specified existing baselines are added to the new baseline.

Note that the use of `-subprojects`, `-no_subprojects`, and `-all_subprojects` affect which subprojects are also added.

By default, if you use `-baseline` but not `-project`, subprojects are not included. However, if you use both `-project` and `-baseline`, then the `-subprojects` default implied by `-project` overrides the `-no_subprojects` default implied by `-baseline`.

*baseline\_name*

Specifies the name that is assigned to the baseline. When you create a baseline, you can assign any legal baseline name to it.

If you do not specify a *baseline\_name*, a unique name is automatically assigned to the baseline. This default name is in the form `yyyymmdd`. If needed, the default name is followed by an underscore and an incremental number to make it unique. For example, the first baseline created on April 1, 2002 has a default name of `20020401`. The second such baseline created on the same day has a default name of `20020401_1`.

---

`-b|-build build`

Specifies a build number or identification for the new baseline. The build number or ID can be any single line text value. Typically, the build value includes some form of build number.

`-d|-desc|-description description`

Specifies the description to be used for the new baseline. The description is a single line of text and cannot contain any newline characters.

`-no_subprojects`

Specifies that when a project is added, its subprojects should not be added. This option impacts projects added using `-project`, `-project_grouping` and `-baseline`. A subproject will be included if it is explicitly specified as a project or is a member of a specified project grouping or baseline.

If neither `-project_grouping` nor `-baseline` is specified, the default is `-subprojects`. If either `-project_grouping` or `-baseline` is specified, the default is `-no_subprojects`.

`-p|-project project_spec`

Specifies to add one or more projects to the new baseline. By default, when a project is added, its entire hierarchy is also added. You can override this by using the `-no_subprojects` option.

`-pg|-project_grouping project_grouping_spec`

Specifies to add projects in the specified project groupings to the new baseline. By default, when a project grouping is added, only those projects in the project grouping are added; subprojects that are not part of the project grouping are not added. To override this behavior, use the `-all_subprojects` option.

Note that `-subprojects`, `-no_subprojects`, and `-all_subprojects` affect which subprojects are added along with the project groupings.

`-purpose purpose`

Specifies the purpose to be used for the new baseline. If not specified, the default purpose is set using the following precedence:

- If a baseline is specified, Synergy uses the first specified baseline's purpose.

- 
- If no baseline is specified, but a project grouping is specified, Synergy uses the first specified project grouping's purpose.
  - If neither a baseline or project grouping is specified, Synergy uses the purpose of the first specified project.

A *purpose* is a setting that specifies the use of a project (e.g., Insulated Development, Integration Testing, System Testing).

Note that if you specify `-purpose`, you must also specify [-release release\\_spec](#).

#### `-rehearse`

Lists the projects and products that will make up the baseline and the name of the baseline that will be created, but does not actually create the baseline.

If any version conflicts are found, you will see a warning that lists all the product and project versions that are in conflict. The conflicts exist because the resulting version already exists in a new baseline or because the resulting version would not be a legal version string.

#### `-release release_spec`

Specifies the release to be used for the new baseline. When creating a baseline, you can specify one active release. If not specified, the default release is set using the following precedence:

- If a baseline is specified, Synergy uses the first specified baseline's release.
- If no baseline is specified, but a project grouping is specified, Synergy uses the first specified project grouping's release.
- If neither a baseline or project grouping is specified, Synergy uses the release of the first specified project.

Note that if you specify `-release`, you must also specify [-purpose purpose](#).

#### `-state state`

Specifies the state of the baseline when it is created. When creating a baseline, valid states are *test\_baseline*, *published\_baseline*, and *released*. The default state for a baseline is *test\_baseline*. Developers can see the baseline in this state and can use it manually; however, they won't get it automatically as the latest baseline. SQE can use it for testing. After it passes testing, the build manager must transition the test baseline to *published\_baseline* to make it available for developers to use.



---

Creating a baseline in the *released* state is equivalent to creating one in the *published\_baseline* state, and then releasing it.

`-subprojects`

Specifies to include subprojects whose release match the component name of the project being added. The component name must match exactly. A release without a component name can only match another release without a component name.

This option applies to projects added with the `-project`, `-project_grouping` and `-baseline` options. This is the default behavior if neither `-project_grouping` nor `-baseline` is specified. If either `-project_grouping` or `-baseline` is specified, the default is `-no_subprojects`.

`-vt | -version_template version_template`

Specifies the version template to be used for any project or product in the new baseline. New project and product versions created during the command use the *version\_template* for their versions.

A *version\_template* is any string, with optional keywords, with the form `%keyword` or `%{keyword}`. The keyword can be any Rational Synergy attribute or built-in keyword.

When an attribute is expanded, the corresponding attribute value from the build management project or product being examined is used. If an attribute or built-in keyword is not found for a specified keyword name, the empty string replaces the keyword.

If the instantiated *version\_template* for any project or product in the baseline contains characters that are not allowed in a version string, those characters are replaced with the default version string replacement character. This is specified in the `ccm.ini` file, with the `baseline_template_repl_char` option. This default character is an underscore (`_`). For example, if `%platform` is part of a version template, and the build management project has a platform of `SPARC-solaris`, then the version string contains the string `SPARC_solaris`. Or, if `%release` is part of a product version template, and the prep product has a release of `CM/6.5`, then the version string contains the string `CM_6.5`.

If the instantiated *version\_template* for any project or product in the baseline is already in use for another version of that project or product, the version is made

---

unique by appending an underscore (\_) and the first integer that will make the version unique, starting with 1. If this causes the version string to be too long, then a version based on the current date is used for that project or product, and a warning is given.

If you don't specify `-version_template`, the default template is used. For more information, see [Version template specification](#).

The work area is updated if the work area template for the project includes the version. If a work area cannot be updated because it is not visible, and `-skip_nonvisible_projects` is not used, the operation continues and all errors are reported. If the work area is visible, but cannot be updated for other reasons, such as lack of proper file permissions or lack of disk space, the operation continues and all failures are reported.

### **Example**

- Create a baseline named `Build_1234_int` for Release **2.0**, for the purpose of **Integration Testing**, that includes a project named `proj1-sqa_3` and its subprojects.

```
ccm baseline -c Build_1234_int -d "Integration build 1234" -r 2.0
-purpore "Integration Testing" -projects proj1-sqa_3 -subprojects
```

### **Related topics**

- [Comparing baselines](#)
- [Deleting a baseline](#)
- [Listing baselines](#)
- [Publishing a baseline](#)
- [Releasing a baseline](#)
- [Showing a baseline property](#)

---

## **Creating component tasks for a baseline**

This subcommand creates component tasks matching the release and purpose of specified baselines. You must be in the *build\_mgr* or *ccm\_admin* role to create component tasks for a baseline.

```
ccm baseline -cct|-create_component_tasks baseline_spec...
```

```
baseline_spec...
```

Specifies the baselines to create component tasks for. For more information, see [Baseline specification](#).

### **Example**

- Create component tasks for a baseline called `2.0_build_34`.

```
ccm baseline -create_component_tasks 2.0_build_34
```

### **Related topics**

- [Creating or previewing a baseline](#)
- [Listing baselines](#)
- [Showing a baseline property](#)
- [Showing the projects, objects, tasks or change requests for a baseline](#)
- [Showing baseline information](#)

---

## Deleting a baseline

This subcommand deletes the specified baselines, optionally deleting the baseline's projects and products. Only users in the *ccm\_admin* role can delete baselines. To prevent a baseline from being used without the *ccm\_admin* role, see [Marking a baseline for deletion](#).

```
ccm baseline -delete ([-wp|-with_projects_and_products] |  
                    [-np|-no_projects_and_products]) baseline_spec...
```

*baseline\_spec...*

Specifies the baselines to be deleted. For more information, see [Baseline specification](#).

`-np|-no_projects_and_products`

Specifies that only the baseline is deleted. The baseline's projects and products will not be deleted.

`-wp|-with_projects_and_products`

Specifies to delete the projects and products in the baseline. This option is the default.

Only projects and products that were not in a static state before the baseline was created will be deleted.

## Example

- Delete a baseline named 20090213.

```
ccm baseline -delete 20090213
```

## Related topics

- [Comparing baselines](#)
- [Listing baselines](#)
- [Marking a baseline for deletion](#)
- [Modifying a baseline](#)
- [Showing a baseline property](#)

---

## Listing baselines

This subcommand lists the baselines that match the specified criteria. If no criteria are specified, all baselines will be listed.

```
ccm baseline -l|-list [(-r|-release release_spec)...] [(-purpose purpose)...]
                [-f|-format "format_string"] [-nf|-noformat]
                ([-ch|-column_header] | [-nch|-nocolumn_header])
                [-sep|-separator separator]
                ([-sby|-sortby sortspec] | [-ns|-nosort|-no_sort])
                [-gby|-groupby groupformat] [-u|-unnumbered]
```

`-ch|-column_header`

Specifies to use a column header in the output format. See [-ch|-column\\_headers](#) for details.

`-f|-format format`

Specifies the command output format. See [-f|-format](#) for details.

`-gby|-groupby groupformat`

Specifies how to group the command output. See [-gby|-groupby](#) for details.

`-nch|-nocolumn_header`

Specifies not to use a column header in the output format. See [-nch|-nocolumn\\_headers](#) for details.

`-nf|-noformat`

Specifies not to use column alignment. See [-nf|-noformat](#) for details.

`-ns|-no_sort`

Specifies that the command's output will not be sorted. See [-ns|-nosort](#) for details.

`-purpose purpose`

Specifies to list only baselines with the specified purposes. If purposes are not specified, then Synergy lists baselines with any purpose.

`-release release_spec`

Specifies to list only baselines with the specified releases. If releases are not specified, then Synergy lists baselines with any release.

---

`-sep|-separator separator`

Used only with the [-f|-format](#) option. Specifies a different separator character. See [-sep|-separator](#) for details.

`-sby|-sortby sortspec`

Specifies how to sort the command output. See [-sby|-sortby](#) for details.

`-u|-unnumbered`

Suppresses automatic numbering of the command's output. See [-u|-unnumbered](#) for details.

### **Example**

- List baselines for release **2.2** and purpose **Integration Testing**.

```
ccm baseline -list -release 2.2 -purpose "Integration Testing"
```

Additionally, see [Formatting usage examples](#) for detailed formatting examples.

### **Related topics**

- [Comparing baselines](#)
- [Deleting a baseline](#)
- [Marking a baseline for deletion](#)
- [Modifying a baseline](#)
- [Publishing a baseline](#)
- [Releasing a baseline](#)
- [Restoring a deleted baseline](#)

---

### **Marking a baseline for deletion**

This subcommand marks a baseline for deletion. This transitions the baseline to the *deleted\_baseline* state, which prevents the baseline from being used by the update members operation.

Only build managers or users with *ccm\_admin* role can perform this operation. Baselines marked for deletion can be deleted later by a user in the *ccm\_admin* role.

```
ccm baseline -mfd|-mark_for_deletion baseline_spec...
```

```
-mfd|-mark_for_deletion
```

Specifies the baselines to mark for deletion. For more information, see [Baseline specification](#).

### **Related topics**

- [Listing baselines](#)
- [Restoring a deleted baseline](#)
- [Showing the projects, objects, tasks or change requests for a baseline](#)
- [Showing baseline information](#)

---

## Modifying a baseline

This subcommand modifies the name or build of the specified baselines, or updates the projects and products to use versions that match the current default baseline version template. Build managers and users in the *ccm\_admin* role can modify a baseline.

```
ccm baseline -modify [-n|-name name] [-b|-build build]  
                [-v|-versions [-vt|-version_template version_template]  
                [-skip_nonvisible_projects]] baseline_spec...
```

*baseline\_spec...*

Specifies the baseline to be modified. For more information, see [Baseline specification](#).

**-b|-build** *build*

Specifies to set the build number or ID for the specified baselines to the *build* value. This can be any single line text value, such as **Turn3**. Typically, the *build* value includes some form of build number.

Users in the *ccm\_admin* role can change the build for a baseline in any state. Build managers can change the build for a baseline in all states except the *released* state.

**-name** *name*

Specifies to set the name of the specified baseline to *name*. Users in the *ccm\_admin* role can modify the name of a baseline in any state. Build managers can modify the name of a baseline in all states except the *released* state.

**-skip\_nonvisible\_projects**

Specifies that projects without a visible work area will not be changed.

For each work area that cannot be updated because it is not visible, a warning is displayed; the operation continues and is successful if there are no other problems and **-skip\_nonvisible\_projects** was not used. If **-skip\_nonvisible\_projects** was used, an error is returned, but the operation continues and does not clean up. All errors are reported at the end. The message indicates whether the failure was because the work area was not visible or because it could not be modified

**-v|-versions**

Specifies to modify the version of the projects and products in the specified baselines.



---

If you specify a `-version_template`, that version template is used for the new versions. Otherwise, the default baseline version template is used. For more information, see [baseline template](#).

`-vt|-version_template version_template`

Specifies the version template to be used for any project or product in the modified baseline. For more information, see [Version template specification](#).

## Related topics

- [Listing baselines](#)
- [Releasing a baseline](#)
- [Showing the projects, objects, tasks or change requests for a baseline](#)

---

## **Publishing a baseline**

This subcommand publishes the specified baselines and transitions them to the *published\_baseline* state. This makes the baseline eligible for selection during an update members operation.

```
ccm baseline -publish baseline_spec...
```

```
-publish baseline_spec...
```

Specifies the baselines to be published. Transitions baselines in the *test\_baseline* state to the *published\_baseline* state. Only build managers or users in the *ccm\_admin* role can complete this transition. For more information, see [Baseline specification](#).

## **Related topics**

- [Releasing a baseline](#)
- [Showing a baseline property](#)
- [Showing the projects, objects, tasks or change requests for a baseline](#)
- [Showing baseline information](#)

---

## Releasing a baseline

This subcommand releases the specified baselines and transitions them into the *published\_baseline* state. This makes the baseline eligible for selection during an update members operation and indicates that the baseline is suitable for release. Build managers and users in the *ccm\_admin* role can release a baseline.

```
ccm baseline -rb|-release_baseline [-c|-comment comment]  
                [-ce|-commentedit] [-cf|-commentfile file_path] baseline_spec...
```

*baseline\_spec...*

Specifies the *baseline\_name* or [Query selection set reference form](#) to be used where a baseline name is allowed. For more information, see [Baseline specification](#).

`-c|-comment comment`

Specifies a comment that will be appended on all baseline projects and their members when they are checked in to the *released* state. The *comment* can contain more than one line and accepts backslash encoded values.

You can use this option with `-commentedit` and `-commentfile`. If you use the `-commentedit` option, the comment displays in the default text editor.

`-ce|-commentedit`

Specifies that the default text editor will be invoked to allow the comment to be composed and edited. The result saved from the text editor is used as the final comment. You can use this option with the `-comment` and `-commentfile` options.

`-cf|-commentfile file_path`

Specifies that the contents of the specified file will be used for the comment. If you specified `-comment`, it is appended to that comment. You can use this option with the `-commentedit` option.

`-rb|-release_baseline`

Releases the baseline that has the *baseline\_spec* you specify. Also checks in all of the projects for a baseline and their members to the *released* state.

---

## Related topics

- [Showing a baseline property](#)
- [Showing the projects, objects, tasks or change requests for a baseline](#)
- [Showing baseline information](#)

---

### ***Restoring a deleted baseline***

This subcommand reverses the mark for deletion operation on the specified baselines. Note that this operation is successful only if performed before the baselines are deleted from the database.

Build managers and users in the *ccm\_admin* role can restore a deleted baseline.

```
ccm baseline -undelete baseline_spec...
```

```
baseline_spec...
```

Specifies the *baseline\_name* or [Query selection set reference form](#) to be used where a baseline name is allowed. For more information, see [Baseline specification](#).

### **Related topics**

- [Comparing baselines](#)
- [Modifying a baseline](#)
- [Publishing a baseline](#)
- [Showing a baseline property](#)
- [Showing the projects, objects, tasks or change requests for a baseline](#)

---

### ***Showing a baseline property***

This subcommand shows a specified property for the specified baselines.

```
ccm baseline -sh|-show ((r|release) | (p|purpose) | (o|owner) |  
                        (desc|description) | (b|build)) baseline_spec...
```

*baseline\_spec...*

Specifies the *baseline\_name* or [Query selection set reference form](#) to be used where a baseline name is allowed. For more information, see [Baseline specification](#).

### **Related topics**

- [Showing the projects, objects, tasks or change requests for a baseline](#)
- [Showing baseline information](#)

---

## Showing the projects, objects, tasks or change requests for a baseline

This subcommand shows the projects, files, directories, tasks, component tasks, or change requests for a baseline. For change requests, it shows whether the change request is fully included or partially included in the baseline.

```
ccm baseline -sh|-show ((proj|project|projects) | (obj|objs|objects) |
    (t|task|tasks) | component_tasks |
    (cr|change_request|change_requests) |
    (fcr|fully_included_change_request|fully_included_change_requests)
    (pcr|partially_included_change_request|
    partially_included_change_requests))
[-f|-format format] [-nf|-noformat]
([-ch|-column_header] | [-nch|-nocolumn_header])
[-sep|-separator separator] ([-sby|-sortby sortspec] |
[-ns|-nosort|-no_sort]) [-gby|-groupby groupformat]
[-u|-unnumbered] baseline_spec...
```

*baseline\_spec...*

Specifies the *baseline\_name* or [Query selection set reference form](#) to be used where a baseline name is allowed. For more information, see [Baseline specification](#).

-ch|-column\_header

Specifies to use a column header in the output format. See [-ch|-column headers](#) for details.

-f|-format *format*

Specifies the command output format. See [-f|-format](#) for details.

-gby|-groupby *groupformat*

Specifies how to group the command output. See [-gby|-groupby](#) for details.

-nch|-nocolumn\_header

Specifies not to use a column header in the output format. See [-nch|-nocolumn headers](#) for details.

-nf|-noformat

Specifies not to use column alignment. See [-nf|-noformat](#) for details.

---

`-ns` | `-no_sort`

Specifies that the command's output will not be sorted. See [-ns|-nosort](#) for details.

`-sep` | `-separator` *separator*

Used only with the [-f|-format](#) option. Specifies a different separator character. See [-sep|-separator](#) for details.

`-sby` | `-sortby` *sortspec*

Specifies how to sort the command output. See [-sby|-sortby](#) for details.

`-u` | `-unnumbered`

Suppresses automatic numbering of the command's output. See [-u|-unnumbered](#) for details.

## Related topics

- [Modifying a baseline](#)
- [Publishing a baseline](#)
- [Releasing a baseline](#)



---

## Showing baseline information

This subcommand shows general information about the specified baselines.

```
ccm baseline -sh|-show (i|info|information) -f|-format format
                    [-nf|-noformat]
                    ([-ch|-column_header] | [-nch|-nocolumn_header])
                    [-sep|-separator separator] baseline_spec...
ccm baseline -sh|-show (i|info|information) baseline_spec...
```

*baseline\_spec...*

Specifies the *baseline\_name* or [Query selection set reference form](#) to be used where a baseline name is allowed. For more information, see [Baseline specification](#).

`-ch|-column_header`

Specifies to use a column header in the output format. See [-ch|-column\\_headers](#) for details.

`-f|-format format`

Specifies the command output format. See [-f|-format](#) for details.

`-nch|-nocolumn_header`

Specifies not to use a column header in the output format. See [-nch|-nocolumn\\_headers](#) for details.

`-nf|-noformat`

Specifies not to use column alignment. See [-nf|-noformat](#) for details.

`-sep|-separator separator`

Used only with the [-f|-format](#) option. Specifies a different separator character. See [-sep|-separator](#) for details.

## Related topics

- [Deleting a baseline](#)
- [Marking a baseline for deletion](#)
- [Modifying a baseline](#)
- [Publishing a baseline](#)
- [Releasing a baseline](#)

---

## Description and uses

A baseline is a set of projects and tasks used to represent your data at a specific point in time. A baseline has many uses. When you perform an update, Rational Synergy uses a baseline as a starting point to look for new changes. You can also compare two baselines to see what changes have been made relative to a particular build. If you use Change, you can use baselines to generate change request reports.

Typically, a build manager creates a baseline; a developer doesn't need to create a baseline because he doesn't make his builds available to other users.

You might find it useful to create a baseline as soon as you perform a build. You can create a baseline and make it available to the test group without making it available to all developers. Making the baseline as soon as you build saves a representation of the build in Rational Synergy in case it is needed later to create a fix for that particular build.

Creating a baseline for each `Integration Testing` and `System Testing` build helps testers and developers to refer back to the set of changes that were used to create the build. Typically, you'll create a baseline for all projects in the same release and purpose. For example, you would create a baseline for each `Integration Testing` build using all `Integration Testing` projects for that release.

**Note** When you create a baseline, you'll specify a list of projects to be included in the baseline. Be sure to include all related projects in your baseline so that you have a complete set for reference.

Baselines can be used by process rules to define the baseline for the projects that use that template. For example, a build manager may create a baseline named `Integration Build 20040913` containing static projects `toolkit-int_20040913`, `calculator-int_20040913`, etc. The numeric designation is the date (`yyyymmdd`) the baseline was created.

A process rule can specify that its projects use a particular baseline; the projects that reference that process rule use the baseline to identify which baseline project to use when updating. For example, if the `Integration Testing` process rule for the current release specified that the `Integration Build 20040913` baseline should be used, a developer's `calculator-bob` project would select `calculator-20040913` as its baseline project.

Using baselines has the following benefits:

- Build managers have a lightweight way to save a set of projects that were successfully built and tested.
- Process rules are more flexible; they can specify a particular baseline or the latest baseline with certain characteristics, enabling build managers to control the team process more precisely. If problems were discovered in a newer baseline, the build manager can reset the team baseline to a previous, successfully-built baseline.
- The update operation uses the baseline to streamline which tasks are evaluated, thereby improving update performance. Only those tasks on top of the baseline are

---

considered when computing update candidates. When a baseline is created, the set of tasks is taken from either the project grouping or from the projects themselves for the projects that update manually. In addition, the tasks from the baseline (for the project grouping) are added to the new baseline, unless the release is different.

- Team members can compare baselines to identify which tasks were introduced in the latest baseline, or identify whether a baseline includes a particular task. This is useful for testers who need to know what features to test, as well as whether to expect a known problem to be fixed in a particular build.
- Specify that a project should be updated to match the latest successful build.

### **How is a new baseline created?**

Both *prep*-state projects and static projects may be added to a new baseline. However, if a build management project is added to a baseline, the actual project is not added. Instead, a copy of the project is created and added to the baseline and checked in. Build management projects and their work areas are preserved as is so as not to cause unnecessary rebuilds. Moreover, new versions are checked out and checked in for all non-static products that are members of the build management project. Other than that, the new project has the same members as the build management project. The new projects and products are checked in to the `member_status` that is associated with the purpose for the baseline. If this `member_status` is not a valid state, the projects and products are checked in to the *integrate* state.

For example, a baseline that has the `Integration Testing` purpose has projects and products that are in the *integrate* state.

If a build management project contains any non-static members that are not projects or products, you cannot add it to a baseline. Before you can add such a project to a baseline, you must check in its non-static members. In addition, you cannot add to a baseline any project whose update properties include a task that is not complete.

A new project or new product version is created based on the build management project version, the date, and if necessary to make the version unique, an incremental number that is appended. For example, if project `ccm_gui-sol_int` is saved as part of a baseline, the new baseline project becomes something like `ccm_gui-sol_int_20040709`. If it is not possible to append an underscore, the date, and an incremental number to the existing version string (and also stay within the limit of 32 characters), then just the date and the number are used.

After a baseline is created, the history view links are changed so that it appears that existing build management projects are checked out from the new baseline projects. In addition, project histories are updated to make it look as though existing prep products are checked out from the products that are created for the baseline projects.

New projects that are created as part of a baseline do not have work areas. If you want the projects to have work areas, you must enable work area maintenance after the baseline has been created. When a project that has a visible work area is added to a baseline, it is checked for work area conflicts. If any non-resolvable conflicts are found, the create baseline operation will fail. To resolve this issue, you must reconcile the project.

---

If a project with a non-visible work area is added to a baseline, the latest-built product may not have been copied to the database. In such a case, the baseline will contain what is in the database, not what is in the non-visible work area. To avoid this problem, the build manager must make sure that changes to all non-visible work areas of projects that are added to a baseline have been synchronized to the database. This must be done before adding such projects to a baseline.

Use the `baseline` command to:

- Create a baseline from an existing prep hierarchy or set of hierarchies.
- Save a baseline instead of manually populating the Tested Tasks folder in order to publish the latest tested changes to developers.
- Show information or associated projects, objects, and tasks for a specific baseline.
- List baselines.
- Modify or rename a baseline
- Release a baseline or compare two baselines.
- Delete an existing baseline, or mark a baseline for deletion.
- Restore a deleted baseline.

You must be working as a build manager to create or release a baseline. You must be working as in the `ccm_admin` role to delete a baseline or modify the build of a released baseline.

### Version template specification

A `version_template` is any string, with optional keywords, with the form `%keyword` or `#{keyword}`. The keyword can be any Rational Synergy attribute, the special keyword `%baseline_name`, or the special keywords, `%date` and `%build`.

When an attribute is expanded, the corresponding attribute value from the build management project or product being examined is used. If no attribute or built-in keyword is found for a specified keyword name, the empty string is used to replace the keyword.

The versions of the project and product versions that became static when the baseline was created are updated to match `version_template`. However, projects that existed in a static state before the baseline was created are not reversioned. For example, if a CM/6.5 SP2 baseline was created with 20 existing static projects from the CM/6.5 SP1 baseline and five new projects from the CM/6.5 SP2, only the five new projects will be reversioned.

If the instantiated `version_template` for any project or product in the baseline contains characters that are not allowed in a version string, then those characters are replaced with the default version string replacement character. This is specified in the `ccm.ini` file, with the option `baseline_template_repl_char`. This character default is an underscore (`_`). For example, if `%platform` is part of a version template, and the build management project has a platform of `SPARC-solaris`, then the version string contains the string

---

SPARC\_solaris. Or, if `%release` is part of a product version template, and the prep product has a release of `CM/6.5`, then the version string contains the string `CM_6.5`.

If the instantiated `version_template` for any project or product in the baseline is already in use for another version of that project or product, then the version is made unique by appending an underscore (`_`) and the first integer that will make the version unique, starting with `1`. If this causes the version string to be too long, then a version based on the current date is used for that project or product, and a warning is given.

If `-version_template` is not specified, then the default (i.e., saved) template is used.

The work area is updated if the work area template for the project includes the version. If a work area cannot be updated because it is not visible, and `-skip_nonvisible_projects` is not used, the operation continues and all errors are reported. If the work area is visible, but cannot be updated for other reasons, such as lack of proper file permissions or disk space, the operation continues and all failures are reported.

### Alternate keyword syntax for version template

The keyword syntax provides a way to alter the expansion behavior of keywords based on their existence.

- `%{keyword:-string}` If `keyword` is set and is non-null, it expands normally; otherwise it expands to `string`. Note that `string` can be an empty string if you want to see nothing when the keyword is not found.
- `%{keyword:+string}` If `keyword` is set and is non-null, it expands to `string`; otherwise it expands to the empty string (substitute nothing).

To get `solaris_7.0` or `7.0` (depending on whether `platform` exists), specify the following:

```
%{platform:-}%{platform:+_}7.0
```

- `%{platform:-}` expands to `solaris` if the platform exists (and was `solaris`); otherwise it expands to the empty string.
- `%{platform:+_}` expands to `_` if the platform exists; otherwise it expands to the empty string.

---

## **bom command**

See [Description and uses](#) for details. The `bom` command supports the [Showing a bill of materials](#) subcommand.

---

## **Showing a bill of materials**

`ccm bom file_spec...`

*file\_spec*

Specifies the product for which the Bill-of-Materials is displayed. This information normally exists only for a controlled product. See [File specification](#) for details.

---

## **Description and uses**

This subcommand shows the Bill-of-Materials for a product.



## **candidates command**

See [Description and uses](#) for details. The `candidates` command supports the [Showing candidates](#) subcommand.

---

## Showing candidates

```
ccm cand|candidates [-r|-recommend] [-f|-format format] [-nf|-noformat]
                    ([-ch|-column_header] | [-nch|-nocolumn_header])
                    [-sep|-separator separator] ([-sby|-sortby sortspec] |
                    [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat]
                    [-u|-unnumbered] file_spec...
```

`-ch|-column_header`

Specifies to use a column header in the output format. See [-ch|-column\\_headers](#) for details.

*file\_spec*

Specifies the name of the object or directory entry for which the candidate versions are listed. See [File specification](#) for details.

`-f|-format format`

Specifies to use a column header in the output format. See [-f|-format](#) for details.

A keyword can be built-in (%fullname, %displayname, %objectname) or the name of any existing attribute such as %modify\_time or %status.

See [Built-In keywords](#) for a list of keywords.

`-gby|-groupby groupformat`

Specifies how to group the command output. See [-gby|-groupby](#) for details.

`-nch|-nocolumn_header`

Specifies not to use a column header in the output format. See [-ch|-column\\_headers](#) for details.

`-nf|-noformat`

Specifies not to use column alignment. See [-nf|-noformat](#) for details.

`-ns|-nosort|-no_sort`

Specifies that the command's output will not be sorted. See [-ns|-nosort](#) for details.

`-r|-recommend`

Results in the recommended version being determined-based. This is the version that would be selected by Rational Synergy based on its selection rules. With the default format, the recommended version is marked with an asterisk (\*). In a user specified

---

format, use the keyword `%recommended` to show the computed, recommended version.

`-sby|-sortby sortspec`

Specifies how to sort the command output. See [-sby|-sortby](#) for details.

`-sep|-separator separator`

Used only with the `-f|-format` option. Specifies a different separator character. See [-sep|-separator](#) for details.

`-u|-unnumbered`

Suppresses automatic numbering of the command's output (that is, the output is unnumbered). See [-u|-unnumbered](#) for details.

### **Example**

- List the versions of `Xincls.h` that can be members of the current project in the directory, and recommend the version to use.

```
ccm cand Xincls.h -recommend
1) Xincls.h-1 integrate john incl projX 1 5
2) Xincls.h-2 integrate john incl projX 1 12
3) Xincls.h-3 integrate mary incl projX 1 13
4) Xincls.h-4 integrate mary incl projX 1 15 *
```

### **Related topics**

- [update command](#)
- [use command](#)

---

## Description and uses

The `candidates` command lists all versions of an object that are eligible for selection when you perform a use or update operation in a directory entry. An object is a candidate for use if the name, type, and object instance attribute values of the object match those of the directory entry.

The output shows the name, version, state, owner, project in which it was created, instance, and associated task number for each object version. The `cand` command supports numbered format options and sets the query selection set. The property *keyword recommended\_version* in a format string represents the recommended version. It expands to "\*" for the recommended version, or a blank string for a non-recommended version.

## cat command

See [Description and uses](#) for details. The `cat` command supports the [Showing source contents](#) subcommand.

---

## **Showing source contents**

Use the `cat` command to display the source of an object.

```
ccm cat | type file_spec...
```

*file\_spec*

Specifies the file to be shown. See [File specification](#) for details.

### **Example**

- Display the second instance of the `foo.c-9` object version, which is a `csrc` object.

```
ccm cat foo.c-9:csrc:2
```

### **Related topics**

- [view command](#)

---

## **Description and uses**

This subcommand is useful for displaying the contents of an object that is not currently a member of the directory. If you specify the file in the context project, and the corresponding work area is visible, the work area file is shown. If not, a temporary copy from the database is shown.

## **change\_type command**

See [Description and uses](#) for details. The `change_type` command supports the [Changing the type for an object](#) subcommand.



---

## ***Changing the type for an object***

This subcommand changes the type of a specific file or directory.

```
ccm change_type -t|-type new_type [-task task_spec] file_spec
```

*file\_spec*

Specifies the file or directory to be changed. See [File specification](#) for details.

-task *task\_spec*

Associates any checked out directory and newly-created object with the specified task. If the current task is set and you do not specify a different task, the checked out directory and newly-created object are automatically associated with the current task.

You can set *task\_spec* to a single task. Setting *task\_spec* to be a blank string means that "no task" is not supported.

-type *new\_type*

Specifies the new type that the object will have. You can set *new\_type* to a single object.

---

## Description and uses

This subcommand changes the type of a file or directory.

When you change a type, Synergy creates a new version of the object with the specified type. If the specified object is in the *working* state, Synergy replaces it with a new object, and deletes the specified object from the database.

If the object is a member of a project and you execute the `change_type` command within the project, Synergy replaces the old object with a new object. If the parent directory is not modifiable, it is automatically checked out for you. The project must be writable by you.

If the object is a member of more than one project, or if you don't execute the command is within the project where the object is a member, the command fails.

## checkin command

See [Description and uses](#) for details. The `checkin` command supports the following subcommands:

- [Checking in a project](#)
- [Checking in a task](#)
- [Checking in an object](#)

---

## Checking in a project

This subcommand checks in a project and, optionally, the sources and products that are members of the project. When checking in a project to a static state, such as *integrate* or *released*, the project's members must also be in a static state. The command should be run on a client that has visibility to the project's work area, if it is being maintained, so changes made in that work area can be automatically synchronized back to the database.

```
ccm ci|checkin -p|-project [-s|-state state] [-task task_spec]  
  [-c|-comment comment_string] [-ce|-commentedit]  
  [-cf|-commentfile file_path] [-cr|-commentreplace] [-nc|-nocomment]  
  [-source|-sources [-ss|-source_state source_state]]  
  [-products [-ps|-product_state product_state]] [-projects]  
  [-h|-hierarchy] project_spec...
```

`-c|-comment comment`

Specifies a comment that will be appended on all baseline projects and their members when they are checked in to the *released* state. The *comment* can contain more than one line and accepts backslash encoded values.

You can use this option with `-commentedit` and `-commentfile`. If you use the `-commentedit` option, the comment displays in the default text editor.

`-ce|-commentedit`

Specifies that the default text editor will be invoked to allow the comment to be composed and edited. The result saved from the text editor is used as the final comment. You can use this option with the `-comment` and `-commentfile` options.

`-cf|-commentfile file_path`

Specifies that the contents of the specified file will be used for the comment. If you specified `-comment`, it is appended to that comment. You can use this option with the `-commentedit` option.

`-cr|-commentreplace`

Normally, any newly specified comments is appended to an existing comment. Use the `-cr` option to replace an existing comment. You can replace a comment only on writable objects.

`-h|-hierarchy`

Applies the check-in scope (for source, products, or projects) to the project hierarchy.

---

`-nc` | `-nocomment`

Do not prompt for comments. If no comment is supplied with any of the options `-c` | `-comment`, `-cf` | `-commentfile` or `ce` | `-commentedit`, and this option is not specified, you are prompted for a comment which is used for all objects checked in by the command. Use the `-nc` | `-nocomment` option to suppress comment prompting.

`-products`

Checks in all the product members of the current project. If `-h` | `-hierarchy` is also specified, this applies to all product members in the hierarchy.

`-p` | `-project`

Specifies that any subprojects within the specified project are also checked in. If `-h` | `-hierarchy` is also specified, this applies to all subprojects in the hierarchy.

*project\_spec*

Specifies the project to be checked in. See [Project specification](#) for details.

`-ps` | `-product_state`

Specifies the state for any product objects that are to be checked in. If not specified, the default next state is determined automatically.

Specifies the state of product objects when checking in a product. This applies both to hierarchy and non-hierarchy check-ins (that is, this option does not require the `-s` option).

`-s` | `-state state`

Specifies the state for the project to be checked in. If not specified, the default next state is determined automatically.

`-source` | `-sources`

Checks in all members of the current project that are source objects. Source objects are files or directories that are not marked as products. If `-h` | `-hierarchy` is also specified, this applies to all source objects in the hierarchy.

---

`-ss|-source_state source_state`

Specifies the state for source objects when source objects are checked in. If not specified, the default next state is determined automatically.

`-task task_spec`

Specifies the task to be associated with any source objects that are checked in. A source object is a file or directory that is not marked as a product. You can set `task_spec` to a single task. See [Task specification](#) for details.

### **Caveats**

To check in a project version to a non-modifiable state, be sure that all members are in a non-modifiable state already because you cannot check in a project to a non-modifiable state if it has modifiable members.

### **Examples**

- Check in the `projB-3` project.

```
ccm ci -c "configuration sent to customer A" -p projB-3
```

- Check in all members of the `tools-5` project, with product members going to the *checkpoint* state, and source (non-product) members going to the *integrate* state.

```
ccm ci -p tools-5 -products -s checkpoint
```

```
ccm ci -p tools-5 -source -ss integrate
```

### **Related topics**

- [checkout command](#)
- [task command](#)

---

## Checking in a task

This subcommand completes a task, checking in the task's associated objects to a non-modifiable state, and transitioning the task to the completed state. This is equivalent to the `ccm task -complete` command. You must be the resolver of the task or an administrator to use this command.

```
ccm ci|checkin -task (task_spec|(current|default))
               [-c|-comment comment_string] [-ce|-commentedit]
               [-cf|-commentfile file_path] [-cr|-commentreplace]
```

`-c|-comment comment_string`

See [-c|-comment \*comment\*](#).

`-ce|-commentedit`

See [-ce|-commentedit](#).

`-task (task_spec|(current|default))`

Specifies the task to be checked in or completed. You can set *task\_spec* to a single task. See [Task specification](#) for details.

## Related topics

- [checkout command](#)
- [task command](#)

---

## Checking in an object

This subcommand checks in specific objects, such as files and directories. Run this command on a client that has visibility to the associated project's work area, if it is being maintained, so changes made in that work area can be automatically synchronized back to the database.

This subcommand applies when you specify one or more arguments, and you don't specify the `-project` option.

```
ccm ci|checkin [-s|-state state] [-task task_spec]  
               [-c|-comment comment_string] [-ce|-commentedit]  
               [-cf|-commentfile file_path] [-cr|-commentreplace]  
               [-nc|-nocomment] file_spec...
```

`-c|-comment comment_string`

See [-c|-comment \*comment\*](#).

`-ce|-commentedit`

See [-ce|-commentedit](#).

`-cf|-commentfile file_path`

See [-cf|-commentfile \*file\\_path\*](#).

`-cr|-commentreplace`

See [-cr|-commentreplace](#).

`-nc|-nocomment`

Do not prompt for comments. If no comment is supplied with any of the options `-c|-comment`, `-cf|-commentfile` or `ce|-commentedit`, and this option is not specified, you are prompted for a comment used for all the objects checked in by the command. Use the `-nc|-nocomment` option to suppress comment prompting.

*file\_spec*

Specifies the file or directory to check in. See [File specification](#) for details.

`-s|-state state`

Specifies the state for the object to be checked in. If not specified, the default next state is determined automatically.



---

`-task task_spec`

Specifies the task to be associated with any source objects that are checked in. A source object is a file or directory that is not marked as a product. You can set the `task_spec` to a single task. See [Task specification](#) for details.

### Examples

- Check in the current version of `foo.c` with a state of *visible*.

```
ccm checkin -s visible foo.c
```

- Check in three files (`clear.c`, `concat.c`, and `display.c`).

- Check in the directory `utils` without any new comments.

```
ccm ci -nc utils
```

```
ccm ci -nc clear.c concat.c display.c
```

- Check in the `c_includes` symbolic link to the *checkpoint* state (UNIX only).

```
ccm ci -c "let others edit" -state checkpoint c_includes
```

### Related topics

- [checkout command](#)
- [task command](#)

---

## Description and uses

Use the `checkin` command to check in one or more objects and, if necessary, set the next state.

You can check in source (non-product objects), product and project objects, assign task numbers to objects you will check in, and add, modify, or replace a comment for the object you will check in.

**Note** You should make changes from only one work area, and perform check in operations with that work area visible.

## checkout command

See [Description and uses](#) for details. The `checkout` command supports the following subcommands:

- [Checking out an object](#)
- [Checking out a project](#)

---

## Checking out an object

This subcommand checks out files or directories.

```
ccm co|checkout [-task task_spec] [-t|-to version/file_spec]  
               [-c|-comment comment_string] [-ce|-commentedit]  
               [-cf|-commentfile file_path] file_spec...
```

`-c|-comment comment`

Specifies a comment that will be appended on all baseline projects and their members when they are checked in to the *released* state. The *comment* can contain more than one line and accepts backslash encoded values.

You can use this option with `-commentedit` and `-commentfile`. If you use the `-commentedit` option, the comment displays in the default text editor.

`-ce|-commentedit`

Specifies that the default text editor will be invoked to allow the comment to be composed and edited. The result saved from the text editor is used as the final comment. You can use this option with the `-comment` and `-commentfile` options.

`-cf|-commentfile file_path`

Specifies that the contents of the specified file will be used for the comment. If you specified `-comment`, it is appended to that comment. You can use this option with the `-commentedit` option.

*file\_spec*

Specifies the file or directory to be checked out. The object must be specified using either a [Work area reference form](#) or a [Project reference form](#) to provide a context project for the check out. See [File specification](#) for details.

`-t|-to version|file_spec`

Specifies the version and changes the name of the new, non-project object, or specify the version of a new project or project hierarchy.

By default, the `-to` argument is interpreted as a new version. For example, run the following command:

```
ccm co foo.c -to bar
```

The new object version is `foo.c-bar`.

---

To change the name, you must include the object name and the version in the destination argument. For example, run the following command:

```
ccm co foo.c -to bar.c-1
```

The new object version is `bar.c-1`.

If you are checking out a project, you can specify the version only. If you are checking out a hierarchy of projects, the new version is used for the project as well as its subprojects. Use the `-versions` option to map new versions to old versions of projects in the hierarchy. The `-to` and `-versions` options are mutually exclusive. Also, if you do not specify the `-to` or `-version` option, the default next version is computed automatically using a Rational Synergy built-in algorithm.

If you are checking out a new version of an object that is used in your current project, the newly checked-out version (the "to" version) also will be used in your project.

**Note** When you check out to a new object name in a non-writable directory, a new directory version is checked out automatically.

If you are in a shared project and your current directory is non-writable, the directory is checked out and associated automatically with the default (or specified) task and is checked in to the *integrate* state. You can disable this feature by setting [shared project directory checkin](#) to `FALSE` in your initialization file. (See [shared project directory checkin](#).)

`-task task_spec`

Specifies the task with which the newly checked out objects are associated. If the current task is set and you do not specify a different task, the objects you are checking out are associated with the current task automatically. (See [Setting or clearing the current task](#) for details). You can set the `task_spec` to a single task. See [Task specification](#) for details.

---

## Examples

- Check out version patch1 from version 1 of `foo.c` (version 3 of `foo.c` is in the current directory).

```
ccm co -c "patch1: fix symbol table bug" -to patch1 foo.c-1
```

- Check out the `utils\tools` (Windows) or `utils/tools` (UNIX) directory, which currently is at version 4.

Windows:

```
> ccm co -c "added new files" c:\users\john\ccm_wa\test_db\projA-3\utils\tools
```

UNIX:

```
$ ccm co -c "added new files" ~/ccm_wa/test_db/projA-3/utils/tools
```

- Set the comment and associate a task with the object versions you are checking out.

```
ccm co -c "comment string" -task task_number object_name1 object_name2
```

## Related topics

- [checkin command](#)
- [copy\\_project command](#)

---

## Checking out a project

This subcommand makes a copy of a project, which sets up a work space. In the work space, you can make changes to project members. Note that this command is now called the `copy_project` operation in Rational Synergy. See [copy project command](#) for details.

```
ccm co|checkout -p|-project [-purpose purpose] [-platform platform]
  [-release (release_spec|as_is)] [-subprojects] ([-t|-to version] |
  [(-versions old_version:new_version,old_version:new_version...)...])
  ([-u|-update] | [-no_u|-no_update]) ([-cb|-copy_based] |
  [-lb|-link_based|-ncb|-not_copy_based])
  ([-rel|-relative] | [-nrel|-not_relative])
  ([-set|-path|-setpath absolute_path] ([-mod|-modifiable] |
  [-nmod|-not_modifiable]) ([-tl|-translate|-translation] |
  [-ntl|-no_translate|-no_translation]) ([-wa|-maintain_wa] |
  [-nwa|-no_wa]) ([-wat|-wa_time] | [-nwat|-no_wa_time])
  [-c|-comment comment_string] [-ce|-commentedit]
  [-cf|-commentfile file_path] project_spec...
```

`-c|-comment comment`

Specifies a comment that will be appended on all baseline projects and their members when they are checked in to the *released* state. The *comment* can contain more than one line and accepts backslash encoded values.

You can use this option with `-commentedit` and `-commentfile`. If you use the `-commentedit` option, the comment displays in the default text editor.

`-cb|-copy_based`

Specifies that a work area is copy based.

`-ce|-commentedit`

Specifies that the default text editor will be invoked to allow the comment to be composed and edited. The result saved from the text editor is used as the final comment. You can use this option with the `-comment` and `-commentfile` options.

`-cf|-commentfile file_path`

Specifies that the contents of the specified file will be used for the comment. If you specified `-comment`, it is appended to that comment. You can use this option with the `-commentedit` option.

---

`-lb|-link_based|-ncb|-not_copy_based`

Makes the work area link-based. This option is available to UNIX users only. You must use this option with the `-p` option.

See the [work area command](#) for more information.

`-mod|-modifiable_wa`

Specifies that files in the work area has permissions set so they are modifiable even if they are not checked out. The default is `-nmod|-not_modifiable_wa`.

`-nmod|-not_modifiable_wa`

Specifies that files in the work area has permissions set so they are modifiable by default only if they are in a writable state such as *working*. This is the default.

`-no_u|-no_update`

Specifies that the checked out project is not updated when it is copied. This is the default.

`-ntl|-no_translate`

Specifies that ASCII files in the work area are copied between Windows and UNIX without newline translation. The default is `-tl|-translate`.

`-nrel|-not_relative`

Specifies that any work area is located on an absolute path. By default, a new project uses the same relative setting as the project being checked out.

`-nwa|-no_wa`

Specifies that the project does not have a maintained work area. This default is `-wa|-maintain_wa`.

`-nwat|-no_wa_time`

Specifies that the files in the project's work area use timestamps that show the Rational Synergy modification time rather than the time they were copied to the work area. This the default.

`-platform platform`

Specifies the platform to be used for the new checked out project. The platform must be the name of a valid platform. The platform choices are listed in the



---

CCM\_HOME\etc\om\_hosts.cfg file (Windows) or \$CCM\_HOME/etc/om\_hosts.cfg file (UNIX) in your Rational Synergy installation. If the option is not specified, the default is to use the same platform value as the project being checked out.

*project\_spec*

Specifies the project to copy. See [Project specification](#) for details.

*-purpose purpose*

Specifies the purpose for the new copied project. The purpose must be the name of a valid defined purpose and valid for the project's release. See [project purpose command](#) for details.

If this option is not specified, and you are in the *developer* role, the default is Insulated Development. If this option is not specified, and you are in the *build\_mgr* or *ccm\_admin* role, the default is Integration Testing.

*-rel|-relative*

Specifies that a work area is located on a path relative to the parent project's path. The default is for the new project to use the same relative setting as the project being checked out.

*-release (release\_spec|as\_is)*

Specifies the release to use for the new copied project. If the keyword "as\_is" is specified, or the option is not specified, the default is to use the release of the project being checked out. You can set the *release\_spec* to a release defined in the current database.

*-set|-path|-setpath absolute\_path*

Specifies the work area path to use for the copied project. If not specified, a default work area path will be determined using the current Work Area Path Template and Project Subdirectory Template.

*-subprojects*

Specifies to copy all subprojects in the specified project hierarchy.

*-t|-to version|file\_spec*

Specifies the version of the checked out project. If you do not specify *-t|-to* or *-versions*, the default next version is computed automatically using a Rational Synergy built-in algorithm.

---

`-tl|-translate|-translation`

Specifies that ASCII files in the work area to copy between Windows and UNIX with newline translation. This is the default.

`-u|-update`

Specifies that the checked out project is updated when it is copied. If specified, the project is checked out without a work area and is updated according to the setting for the project grouping that indicates whether the baseline and tasks should be refreshed. If the project has a maintained work area, the project is synchronized. The default is

`-no_u|-no_update`.

`-versions old_version:new_version,old_version:new_version,...`

Specifies the new versions to use for copying a project or project hierarchy. Each mapping applies to all projects in the hierarchy that currently have that value. If `new_version` is `NoCheckOut`, projects with the corresponding `old_version` are not copied.

If neither `-t|-to` or `-versions` are specified, the default next version is computed using a Rational Synergy built-in algorithm.

`-wa|-maintain_wa`

Specifies that the project has a maintained work area. This is the default.

`-wat|-wa_time`

Specifies that the files in the project's work area use timestamps that show the time they were copied into the work area, rather than the Rational Synergy modification time. The default is `-nwat|-no_wa_time`.

## Examples

- Check out a new development projects hierarchy from an existing project hierarchy. Set the versions of all of the projects to your name.

```
ccm co -p toolkit-int -subprojects -to john
```

- Check out a new build management project hierarchy for system testing. Set the release and platform values and versions.

Windows:

```
ccm co -p tool_top-1.0 -subprojects -release 2.0 -platform win32 -
```

---

```
purpose "System Testing" -versions  
"1.0:sqa,win16_1.0:win16_sqa,win32_1.0:win32_sqa"
```

**UNIX:**

```
ccm co -p tool_top-1.0 -subprojects -release 2.0 -platform SunOS -  
purpose "System Testing" -versions  
"1.0:sqa,win16_1.0:win16_sqa,win32_1.0:win32_sqa"
```

- **Modify the version for a top-level project and propagate the change to its subproject versions.**

```
ccm co -p top_project_spec -subprojects -to version
```

## Related topics

- [checkin command](#)
- [copy\\_project command](#)

---

## Description and uses

When you check out an object in a non-shared project, its default state is *working*. When you check out a file or directory in a shared project, its default state is *visible* if it is a non-product, and shared if it is a product.

When you check out an object, a writable version of the object is placed in the directory (use the `ccm dir` or `ccm ls` command to verify the object). When you check out a directory, no visible change is made to the file system. When you use the `-t` option to specify a new version at check out, you can specify the version and change the name of the new object. On UNIX, you can change the location where the symbolic link points to when you check out a symbolic link.

The object to check out must be specified in a form that provides a context project and parent directory.

- [Work area reference form](#)

The specified path must be in a project's maintained work area.

- [Project reference form](#)

For example, `sub_proj\foo.c@my_proj-1` (Windows) or `sub_proj/foo.c@my_proj-1` (UNIX).

You can use the project reference form even when the project does not have a maintained work area.

You cannot check out an object using a *file\_spec* that does not provide a context project, such as a selection set reference form (e.g. "@1") or an object name form (e.g. `foo.c-1:csrc:1`).

## checkpoint command

See [Description and uses](#) for details. The `checkpoint` command supports the following subcommands:

- [Checkpointing a project](#)
- [Checkpointing an object](#)

---

## Checkpointing a project

This subcommand saves a personal version of an object for your use only, by preserving it in a state that is not modifiable, but that you can delete later when you no longer need it. You must own the object to perform a `checkpoint`. Only working objects can be checkpointed.

```
ccm ckpt|checkpoint -p|-project [-t|-to version]
                    [-c|-comment comment_string] [-ce|-commentedit]
                    [-cf|-commentfile file_path] [-cr|-commentreplace] project_spec...
```

`-c|-comment comment`

Specifies a comment that will be appended on all baseline projects and their members when they are checked in to the *released* state. The *comment* can contain more than one line and accepts backslash encoded values.

You can use this option with `-commentedit` and `-commentfile`. If you use the `-commentedit` option, the comment displays in the default text editor.

`-ce|-commentedit`

Specifies that the default text editor will be invoked to allow the comment to be composed and edited. The result saved from the text editor is used as the final comment. You can use this option with the `-comment` and `-commentfile` options.

`-cf|-commentfile file_path`

Specifies that the contents of the specified file will be used for the comment. If you specified `-comment`, it is appended to that comment. You can use this option with the `-commentedit` option.

`-cr|-commentreplace`

Normally, the comment specified is appended to any existing comment. However, if you use the `-cr` option, the new comment will replace any existing comment.

`-p|-project project_spec`

Checkpoint a project. See [Project specification](#) for details.

`-t|-to version`

Sets the version of the newly checked-out object. You also can do this by adding the version to the object name.

---

## Related topics

- [Checkpointing an object](#)
- [Checking out a project](#)

---

## Checkpointing an object

This subcommand saves a personal version of an object for your use only, by preserving it in a state that is not modifiable, but that you can delete later when you no longer need it. You must own the object to perform a `checkpoint`. Only working objects can be checkpointed.

```
ccm ckpt|checkpoint [-task task_spec] [-t|-to version|file_spec]  
    [-c|-comment comment_string] [-ce|-commentedit]  
    [-cf|-commentfile file_path] [-cr|-commentreplace] file_spec...
```

`-c|-comment comment`

Specifies a comment that will be appended on all baseline projects and their members when they are checked in to the *released* state. The *comment* can contain more than one line and accepts backslash encoded values.

You can use this option with `-commentedit` and `-commentfile`. If you use the `-commentedit` option, the comment displays in the default text editor.

`-ce|-commentedit`

Specifies that the default text editor will be invoked to allow the comment to be composed and edited. The result saved from the text editor is used as the final comment. You can use this option with the `-comment` and `-commentfile` options.

`-cf|-commentfile file_path`

Specifies that the contents of the specified file will be used for the comment. If you specified `-comment`, it is appended to that comment. You can use this option with the `-commentedit` option.

`-cr|-commentreplace`

Normally, the comment specified is appended to any existing comment. However, if you use the `-cr` option, the new comment will replace any existing comment.

*file\_spec*

Specifies the file, directory, or project to checkpoint. See [File specification](#) for details.

`-p|-project`

Shows the history of a project.



---

*project\_spec*

Specifies the project to list. See [Project specification](#) for more information.

*-t|-to version|file\_spec*

Specifies the version and changes the name of the new, non-project object, or specify the version of a new project or project hierarchy.

*-task task\_spec*

Specifies the task with which you want your newly checked-out object to be associated. See [Task specification](#) for details.

If you do not specify a task but a current task is set, the newly created object version is associated with the current task. Any task associated with the checkpoint object version remains unchanged.

## Examples

- Checkpoint the current *working* version of `foo.c`, and add a comment.

```
ccm ckpt -c "Phase 1 works." foo.c
```

```
Adding 'release' attribute with value '2.0' to object foo.c-3:csrc:11
```

```
Associated object foo.c-3:csrc:11 with task 36
```

```
Checkpointed object version: 'foo.c-2:csrc:11'
```

- Checkpoint the current working version of `foo.c`. Add a comment and specify the new *working* object version to be *joe*.

```
ccm ckpt -c "Trying Joe's algorithm." -t joe foo.c
```

```
Adding 'release' attribute with value '2.0' to object foo.c-  
joe:csrc:11
```

```
Associated object foo.c-joe:csrc:11 with task 36.
```

```
Checkpointed object version: 'foo.c-3:csrc:11'
```

## Related topics

- [Checking out an object](#)
- [Checkpointing a project](#)

---

## Description and uses

The `checkpoint` command saves a personal version of an object for your use only, by preserving it in a state that is not modifiable, but that you can delete later when you no longer need it. You must own the object to perform a `checkpoint`. Only working objects can be checkpointed.

When you perform a checkpoint, the current version of the object is moved to the *checkpoint* state and a new version of the object is created. All comments specified on the `checkpoint` command are applied to the checkpointed object.

## cmdhistory command

See [Description and uses](#) for details. The `cmdhistory` command supports the following subcommands:

- [Clearing entries from history](#)
- [Showing command history](#)
- [Setting a maximum number of commands to record](#)

---

## ***Clearing entries from history***

```
ccm cmdhistory -clear
```

```
-clear
```

Clears the command history of all commands executed in the current session.

### **Related topics**

- [Setting a maximum number of commands to record](#)
- [Showing command history](#)
- [Showing history for an object](#)

---

## Showing command history

```
ccm cmdhistory -s|-sh|-show [count]
```

```
-s|-sh|-show
```

Shows the commands executed in the current session, up to the specified maximum, and excluding the `cmdhistory` command.

*count*

If you use the `count` argument, the number of commands you've entered for the session is displayed. For example, if you request a count of 10 and have set a maximum command history of 50, and you've executed 200 commands in that session, the count will show the last 10 commands. If you've only executed five commands in that session, the count shows the last five commands.

### Example

- Show the last three commands executed in this session.

```
ccm cmdhistory -show 3
copy_project -c "test projA" projA-3
task -query -owner sue -release cm/7.0 -f "%priority %task_synopsis"
task -default 26
```

### Related topics

- [Clearing entries from history](#)
- [Setting a maximum number of commands to record](#)
- [Showing history for an object](#)

---

## **Setting a maximum number of commands to record**

`ccm cmdhistory -set maximum`

`-set`

Specifies the number of commands saved in the history.

The default maximum value of saved commands is 100.

`maximum`

Change the maximum value of saved commands to a number of your choosing.

### **Example**

- Set the command history to record 60 commands maximum.

```
ccm cmdhistory -set 60
```

### **Related topics**

- [Clearing entries from history](#)
- [Showing command history](#)
- [Showing history for an object](#)

---

## Description and uses

Use `cmdhistory` to obtain a record of commands executed for a session. The following examples show ways you might use this command:

- SQE performs ad-hoc testing and then captures a series of commands to be included in new scripted tests.
- IBM Rational Software Support obtains the CLI command history for a customer to assist in problem investigation and analysis.

---

## conflicts command

See [Description and uses](#) for details. The `conflicts` command supports the following subcommands:

- [Showing object conflicts for a project](#)
- [Showing task conflicts for a project](#)



---

## Showing object conflicts for a project

This subcommand shows the object conflicts in a project.

```
ccm conflicts [-r|-recurse] [-f|-format format] [-nf|-noformat]
              ([-ch|-column_header] | [-nch|-nocolumn_header])
              [-sep|-separator separator] ([-sby|-sortby sortspec] |
              [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat] project_spec
```

`-ch|-column_header`

Specifies to use a column header in the output format. This is the default. See [-ch|-column\\_headers](#) for details.

`-f|-format format`

Specifies the command output format. See [-f|-format](#) for details.

`-gby|-groupby groupformat`

Specifies how to group the command output. See [-gby|-groupby](#) for details.

`-nch|-nocolumn_header`

Specifies not to use a column header in the output format. See [-nch|-nocolumn\\_headers](#) for details.

`-nf|-noformat`

Specifies not to use column alignment. See [-nf|-noformat](#) for details.

`-ns|-no_sort`

Specifies that the command's output will not be sorted. See [-ns|-nosort](#) for details.

*project\_spec*

Specifies the project to analyze for membership conflicts. You can set *project\_spec* to one project only. See [Project specification](#) for details.

`-r|-recurse`

Specifies to show membership conflicts in all projects in the hierarchy for the specified top-level project.

---

`-sep|-separator separator`

Used only with the [-f|-format](#) option. Allows you to specify a different separator character. See [-sep|-separator](#) for details.

`-sby|-sortby sortspec`

Specifies how to sort the command output. See [-sby|-sortby](#) for details.

## Related topics

- [Global formatting options](#)
- [Formatting usage examples](#)

---

## Showing task conflicts for a project

This subcommand shows the task conflicts in a project.

```
ccm conflicts -t|-tasks [-r|-recurse] [-f|-format format] [-nf|-noformat]
                ([-ch|-column_header] | [-nch|-nocolumn_header])
                [-sep|-separator separator] ([-sby|-sortby sortspec] |
                [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat] project_spec
```

`-ch|-column_header`

Specifies to use a column header in the output format. This is the default. See [-ch|-column\\_headers](#) for details.

`-f|-format format`

Specifies the command output format. See [-f|-format](#) for details.

`-gby|-groupby groupformat`

Specifies how to group the command output. See [-gby|-groupby](#) for details.

`-nch|-nocolumn_header`

Specifies not to use a column header in the output format. See [-nch|-nocolumn\\_headers](#) for details.

`-nf|-noformat`

Specifies not to use column alignment. See [-nf|-noformat](#) for details.

`-ns|-no_sort`

Specifies that the command's output will not be sorted. See [-ns|-nosort](#) for details.

*project\_spec*

Specifies the project to analyze for membership conflicts. You can set *project\_spec* for one project only. See [Project specification](#) for details.

`-r|-recurse`

Specifies to show membership conflicts in all projects in the hierarchy for the specified top-level project.

---

`-sep|-separator separator`

Used only with the [-f|-format](#) option. Allows you to specify a different separator character. See [-sep|-separator](#) for details.

`-sby|-sortby sortspec`

Specifies how to sort the command output. See [-sby|-sortby](#) for details.

### **Example**

- Show the conflict detection information for the `ProjectTwo-newVer` project.

```
ccm conflicts ProjectTwo-newVer
```

```
Project: ProjectTwo-newVer
```

Objectname	Task	Conflicts	Category
NewFile.txt-one:ascii:1	9	Implicitly included	Extra Changes
File1.txt-ab:ascii:1	9	Included by 'use' operation?	Extra Changes
File2.txt-cd:ascii:1	9	Included by 'use' operation?	Extra Changes
File2.txt-ef:ascii:1	9	Implicitly included	Extra Changes
ProjectTwo-2:dir:1	9	Included by 'use' operation?	Extra Changes

### **Related topics**

- [Global formatting options](#)
- [Formatting usage examples](#)

---

## Description and uses

The `conflicts` command displays the conflicts for a project whose update properties use tasks and a baseline.

A conflict represents an inconsistency between the set of changes associated with a project update properties and the set of changes included in the membership of a project.

For detailed information about conflicts and how they are identified, see "Conflict detection" in the [Rational Synergy CLI Help, Traditional mode](#).

## copy\_project command

See [Description and uses](#) for details. The `copy_project` command supports the [Copying a project](#) subcommand.

---

## Copying a project

This subcommand creates a modifiable version of a project or project hierarchy. By default, when you copy a project, it is created in the database and a work area is created automatically. You can set work area properties at the time you copy the project.

```
ccm copy_project|cp [-purpose purpose] [-platform platform]
  [-release (release_spec|as_is)] [-subprojects] ([-t|-to version] |
  [(-versions old_version:new_version,old_version:new_version...)...])
  ([-u|-update] | [-no_u|-no_update]) ([-cb|-copy_based] |
  [-lb|-link_based|-ncb|-not_copy_based])
  ([-rel|-relative] | [-nrel|-not_relative])
  [-set|-path|-setpath absolute_path] ([-mod|-modifiable] |
  [-nmod|-not_modifiable]) ([-tl|-translate|-translation] |
  [-ntl|-no_translate|-no_translation]) ([-wa|-maintain_wa] |
  [-nwa|-no_wa]) ([-wat|-wa_time] | [-nwat|-no_wa_time])
  [-c|-comment comment_string] [-ce|-commentedit]
  [-cf|-commentfile file_path] project_spec...
```

**-c|-comment *comment***

Specifies a comment that will be appended on all baseline projects and their members when they are checked in to the *released* state. The *comment* can contain more than one line and accepts backslash encoded values.

You can use this option with `-commentedit` and `-commentfile`. If you use the `-commentedit` option, the comment displays in the default text editor.

**-ce|-commentedit**

Specifies that the default text editor will be invoked to allow the comment to be composed and edited. The result saved from the text editor is used as the final comment. You can use this option with the `-comment` and `-commentfile` options.

**-cf|-commentfile *file\_path***

Specifies that the contents of the specified file will be used for the comment. If you specified `-comment`, it is appended to that comment. You can use this option with the `-commentedit` option.

**-cb|-copy\_based**

Specifies that a work area is copy based.

**-lb|-link\_based|-ncb|-not\_copy\_based**

Makes the work area link-based. This option is available to UNIX users only.

---

See the [work area command](#) for more information.

`-mod|-modifiable_wa`

Specifies that files in the work area has permissions set so they are modifiable even if they are not checked out. The default is `-nmod|-not_modifiable_wa`.

`-nmod|-not_modifiable_wa`

Specifies that files in the work area have permissions set so they are modifiable by default only if they are in a writable state such as *working*. This is the default.

`-no_u|-no_update`

Specifies that the checked out project is not updated when it is copied. This is the default.

`-ntl|-no_translate|-no_translation`

Specifies that ASCII files in the work area are copied between Windows and UNIX without newline translation. The default is `-tl|-translate`.

`-nrel|-not_relative`

Specifies that any work area is located on an absolute path. The default is for the new project to use the same relative setting as the project being checked out.

`-nwa|-no_wa`

Specifies that the project does not have a maintained work area. This default is `-wa|-maintain_wa`.

`-nwat|-no_wa_time`

Specifies that the files in the project's work area use timestamps that show the Rational Synergy modification time rather than the time they were copied into the work area. This the default.

`-platform platform`

Specifies the platform to be used for the new checked out project. The *platform* should be the name of a valid platform. The platform choices are listed in the `CCM_HOME\etc\om_hosts.cfg` file (Windows) or `$CCM_HOME/etc/om_hosts.cfg` file (UNIX) in your Rational Synergy installation. If the option is not specified, the default is to use the same platform value as the project being checked out.



---

*project\_spec*

Specifies the project to copy. See [Project specification](#) for details.

`-purpose purpose`

Specifies the purpose for the new copied project. The *purpose* must be the name of a valid defined purpose and that purpose must be valid for the project's release. See [project purpose command](#) for details.

If this option is not specified, and you are in developer role, the default is Insulated Development. If you don't specify this option, and you're in the *build\_mgr* or *ccm\_admin* role, the default is Integration Testing.

`-rel|-relative`

Specifies that any work area will be located on a path relative to the parent project's path. The default is for the new project to use the same relative setting as the project being checked out.

`-release release_spec`

Specifies the release to be used for the new, copied project. If the keyword "as\_is" is specified, or the option is not specified, the default is to use the release of the project being checked out. You can set the *release\_spec* to a release defined in the current database. See [Release specification](#) for details.

Rational Synergy projects must have a release value because manual update properties are not supported, and project groupings and their corresponding process rules must always be associated with a release.

`-set|-path|-setpath absolute_path`

Specifies the work area path to be used for the copied project. If not specified, a default work area path is determined using the current [wa\\_path\\_template](#) and [project\\_subdir\\_template](#).

`-subprojects`

Specifies to copy all subprojects in the specified project hierarchy.

`-tl|-translate|-translation`

Indicates that ASCII files should be translated when they are copied between Windows and UNIX within the project work area.

---

`-t|-to version`

Specifies the version of the checked out project. If `-to` or `-versions` are not specified, the default next version is computed automatically.

`-u|-update`

Specifies that the checked out project is updated when it is copied. If specified, the project is checked out without a work area and is updated according to the project grouping setting indicating whether the baseline and tasks should be refreshed. If the project is to have a maintained work area, the project is synchronized. The default is `-no_u|-no_update`.

`-versions "old_ver:new_ver,old_ver:new_ver,..."`

Specifies the new versions to use for copying a project or project hierarchy. Each mapping applies to all projects in the hierarchy that currently have that value. If `new_version` is `NoCheckOut`, projects with the corresponding `old_version` are not copied.

If `-to` or `-versions` are not specified, the default next version is computed automatically.

`-wa|-maintain_wa`

Specifies that the project has a maintained work area. This is the default.

`-wat|-wa_time`

Specifies that the files in the project's work area use timestamps that show the time they were copied into the work area, rather than the Rational Synergy modification time. The default is `-nwat|-no_wa_time`.

## Examples

- Copy a new version of the `projA-3` project.  

```
ccm copy_project -c "test projA" projA-3
```
- Copy a new development projects hierarchy from an existing project hierarchy. Set the versions of all of the projects to your name.  

```
ccm copy_project toolkit-int -subprojects -to john
```

- 
- Copy a new build management project hierarchy for system testing. Set the release and platform values and versions.

```
ccm copy_project tool_top-1.0 -subprojects -release 2.0 -platform  
win32 -purpose "System Testing" -versions  
"1.0:sqa,win16_1.0:win16_sqa,win32_1.0:win32_sqa"
```

- Modify the version for a top-level project and propagate the change to its subproject versions.

```
ccm copy_project top_project_spec -subprojects -to version
```

### Related topics

- [Checking in a project](#)
- [Checking out a project](#)

---

## Description and uses

The `copy_project` command functions the same as the `checkout` command with the `-project` option, and the `copy_project` operation was referred to as the `checkout -project` operation in prior releases.

When you copy a project from a static (non-modifiable) project and do not copy subprojects, and the subprojects have relative work areas, new copies of those subprojects' work areas are created in the appropriate locations within the work area of the project being copied. Developers can reuse static subprojects that have relative work areas.

Static work areas are not maintained and cannot be reconciled with the database; they are ignored during reconcile. Synchronizing a static work area replaces any files that have been modified with files from the database. Copying a project with a static work area leaves the original work area in place; you must reconcile it to discard or keep changes.

## copy\_to\_file\_system command

See [Description and uses](#) for details. The `copy_to_file_system` command supports the [Copying a project to the file system](#) subcommand.

---

## Copying a project to the file system

The `copy_to_file_system` command makes a copy of a non-writable project in your work area.

You cannot maintain and reconcile the project after it is created.

```
ccm cfs|copy_to_file_system|wa_snapshot [-p|-path path] [-r|-recurse]
      project_spec...
```

`-p|-path path`

Specifies the path to which the copied project is written. The path defaults to the expanded default work area path; (`ccm_wa\database_name` on Windows, or `ccm_wa/database_name` on UNIX in your home directory).

**Note** if a path is not specified, the path will default to the expanded default work area path template. Also, the path must be empty and the directory must not contain files.

*project\_spec*

Specifies the project to be copied. See [Project specification](#) for details.

`-r|-recurse`

Creates copied projects for the subprojects as well as the selected project (`ccm_wa\database_name` on Windows, or `ccm_wa/database_name` on UNIX in your home directory).

**Note** This option will create work area copies for the specified projects and all subprojects. If this option is not on, subprojects are ignored.

## Example

- Create a copied project in your work area for project list `proj1-1 proj2-2`:  

```
ccm copy_to_file_system -path C:\ccm_wa\ccm_docs proj1-1 proj2-1
```

---

## **Description and uses**

A project copied to your work area has the following characteristics:

- Always copy-based, never link-based
- Files are read-only
- File modification time is set to the time the copy is created
- Can be created on a project that does not have a work area

## create command

See [Description and uses](#) for details. The `create` command supports the following subcommands:

- [Creating a top-level project](#)
- [Creating an object](#)



---

## Creating a top-level project

This subcommand creates a new top-level project. If you want to make it a subproject in an existing Rational Synergy project, use the `ccm use -p` command after you have created the project. When you create a project, Rational Synergy creates a work area for it automatically. By default, the work area is formed by expanding the default work area path template. With the default setting, this will be `%HOMEPATH%\My`

`Documents\Synergy\ccm_wa\databaseName\projectName-projectVersion` on Windows, or `$HOME/ccm_wa/databaseName/projectName-projectVersion` on UNIX.

```
ccm create -t|-type project [-platf|-platform platform]  
    [-purp|-purpose purpose] [-rel|-release release_spec]  
    ([-cb|-copy_based] | [-lb|-link_based|-ncb|-not_copy_based])  
    [-set|-path|-setpath absolute_path]  
    [-wa|-maintain_wa] [-nwa|-no_wa]  
    [-task task_spec] [-c|-comment comment_string] [-ce|-commentedit]  
    [-cf|-commentfile file_path] new_project_spec...
```

`-c|-comment comment`

Specifies a comment that will be appended on all baseline projects and their members when they are checked in to the *released* state. The *comment* can contain more than one line and accepts backslash encoded values.

You can use this option with `-commentedit` and `-commentfile`. If you use the `-commentedit` option, the comment displays in the default text editor.

`-ce|-commentedit`

Specifies that the default text editor will be invoked to allow the comment to be composed and edited. The result saved from the text editor is used as the final comment. You can use this option with the `-comment` and `-commentfile` options.

`-cf|-commentfile file_path`

Specifies that the contents of the specified file will be used for the comment. If you specified `-comment`, it is appended to that comment. You can use this option with the `-commentedit` option.

`-cf|-commentfile file_path`

Specifies that the contents of the specified file will be used for the comment. If you specified `-comment`, it is appended to that comment. You can use this option with the `-commentedit` option.

---

`-cb|-copy_based`

Specifies that a work area is copy based.

`-lb|-link_based|-ncb|-not_copy_based`

Makes the work area link-based. This option is available to UNIX users only.

See the [work area command](#) for more information.

`new_project_spec`

Specifies the name and version (optional) of the project to be created. The `new_project_spec` must be in one of the following forms:

- A name
- A name, colon, and version
- A [File contents form](#) that contains one of the above

Note that `new_project_spec` is not a general project specification. You cannot use forms such as a [Object name form](#) or [Query selection set reference form](#).

`-nwa|-no_wa`

Specifies that the new project does not have a maintained work area. Use the work area command if you want the project to have a maintained work area later. By default, the project is created with a maintained work area.

`-platf|-platform platform`

Specifies the platform for the new project. The platform must be a valid platform name.

`-purpose purpose`

Specifies the purpose for the new project. The purpose should be the name of a defined purpose that is valid for the specified release. Use the `project_purpose -show` command to list valid purposes.

`-release release_spec`

Specifies the release that will be used for the new project. You can set the `release_spec` to a single release that is defined and active. See [Release specification](#) for details.

`-set|-path|-setpath absolute_path`

Specifies the work area path that will be used for the project. The `absolute_path` should be an absolute path that you can see and modify.

---

`-task task_spec`

Specifies the task with which the new project's root directory will be associated. You can set the *task\_spec* to a single task. By default, the project's root directory is associated with the current task. See [Task specification](#) for details.

`-wa|-maintain_wa`

Specifies that the new project has a maintained work area. This is the default if neither `-wa|-maintain_wa` or `-nwa|-no_wa` are specified. The work area is updated with changes made to the new project. Use the `work area` command to turn off work area maintenance.

---

## Examples

- Create an initial project called `proj1` in the work area.

```
ccm create -t project proj1
```

- Create an initial project and maintain a work area.

```
ccm create -t project -c "test" -wa -set "/tmp" testwa-1.0
```

- Create `MainPrj-1` and `SubPrj-1` with `-wa`. Use `SubPrj-1` inside the `MainPrj-1` root directory:

```
ccm create -t project MainPrj-1 -release 1.0 -task 11 -purp  
"Integration Testing" -wa
```

```
ccm create -t project SubPrj-1 -release 1.0 -task 12 -purp "Integration  
Testing" -wa
```

```
cd WAPATH\MainPrj-1\MainPrj (Windows) OR cd WAPATH/MainPrj-1/MainPrj  
(Unix)
```

```
ccm use -p SubPrj-1 -task 13
```

- Create `MainPrj-1` and `SubPrj-1` with `-nwa`. Use `SubPrj-1` inside the `MainPrj-1` root directory:

```
ccm create -t project MainPrj-1 -release 1.0 -task 11 -purp  
"Integration Testing" -nwa
```

```
ccm create -t project SubPrj-1 -release 1.0 -task 12 -purp "Integration  
Testing" -nwa
```

```
ccm use -task 13 -p SubPrj-1 -dir MainPrj@MainPrj-1
```

## Related topics

- [Adding a project to the current directory](#)
- [Deleting objects from the database](#)
- [Showing projects](#)

---

## Creating an object

This subcommand creates a new object and adds it to the context project and context parent directory associated with the specified object. When a work area reference form is used, the context project and context parent directory is the project associated with the specified work area path. When a project reference spec form is used, the context project and context parent directory is the project specified in that specification.

When you create an object in a non-shared project, its default state is working. When you create a file or directory in a shared project, its default state is visible if it is a non-product, and shared if it is a product.

When you create a new object in a non-writable directory, a new directory version is checked out automatically. You will need to check in the directory and the new object to make the new object available to other users.

If you are in a shared project and your current directory is not modifiable, the directory is checked out and associated automatically with the current (or specified) task and is checked in to the integrate state. You can disable the automatic check-in feature by setting `shared_project_directory_checkin` to `FALSE` in your initialization file. See [shared project directory checkin](#).

```
ccm create [-t|-type type] [-v|-version version] [-task task_spec]  
           [-c|-comment comment_string] [-ce|-commentedit]  
           [-cf|-commentfile file_path] new_file_spec...
```

`-c|-comment comment`

Specifies a comment that will be appended on all baseline projects and their members when they are checked in to the *released* state. The *comment* can contain more than one line and accepts backslash encoded values.

You can use this option with `-commentedit` and `-commentfile`. If you use the `-commentedit` option, the comment displays in the default text editor.

`-ce|-commentedit`

Specifies that the default text editor will be invoked to allow the comment to be composed and edited. The result saved from the text editor is used as the final comment. You can use this option with the `-comment` and `-commentfile` options.

`-cf|-commentfile file_path`

Specifies that the contents of the specified file will be used for the comment. If you specified `-comment`, it is appended to that comment. You can use this option with the `-commentedit` option.

---

*new\_file\_spec*

Specifies the new file or directory to be created. The *new\_file\_spec* must be in one of the following forms:

- A [Work area reference form](#) with a relative path ending with the name of the new object, and optionally, the version delimiter and version or a colon and version. The parent directory must reference a controlled directory in a maintained work area.
- A [Project reference form](#) with a relative path ending with the name of the new object, and optionally, the version delimiter and version or colon and version, located under a directory that exists in the specified project.
- A [File contents form](#) that contains either of the above

The forms provide a context project and a context parent directory. The object is created under the parent directory in the specified context project. Note that *new\_file\_spec* is not a general *file\_spec*. You cannot use forms such as a [Object name form](#) or [Query selection set reference form](#).

If the version is not specified, then 1 is used as the default version.

When [allow\\_delimiter\\_in\\_name](#) is set to TRUE and if *new\_file\_spec* includes a single version delimiter, the string is used as the name of the object. For example, `newfile-2` has the name `newfile-2` with a default version. With this setting, if you want to create a file named `newfile` with version 2, specify a *new\_file\_spec* of `newfile` and use `-version 2`. If [allow\\_delimiter\\_in\\_name](#) is set to FALSE, then any version delimiter in the *new\_file\_spec* is processed as a version delimiter and you can specify the version.

`-task task_spec`

Specifies the task with which the new object is associated. If the directory under which the new object is to be created is not modifiable, it is automatically checked out and associated with that task. You can set the *task\_spec* to a single task. By default, the new object and any automatically checked out directory are associated with the current task. See [Task specification](#) for details.

`-t|-type type`

Specifies the type of the new object. If you do not specify a type, the default is calculated from the extension (for example, a `.c` object defaults to a `csrc` type).

`-v|-version version`

When specified, overrides any version specified in the *new\_file\_spec*. This is primarily intended for use when [allow\\_delimiter\\_in\\_name](#) is set to TRUE. If *new\_file\_spec* includes a single version delimiter, the string is used as the name of

---

the object. For example, `newfile-2` has the name `newfile-2` with a default version. With this setting, if you want to create a file named `newfile` with version 2, specify a `new_file_spec` of `newfile` and use `-version 2`. If `allow_delimiter_in_name` is set to `FALSE`, then any version delimiter in the `new_file_spec` is processed as a version delimiter and you can specify the version.

### Examples

- On Windows, create a new C source object called `sort.c` in the `utils\sym_tool` directory.

```
ccm create -type csrc utils\sym_tool\sort.c
```

- On UNIX, create a new C source object called `sort.c` in the `utils/sym_tool` directory.

```
ccm create -type csrc utils/sym_tool/sort.c
```

- Create a new directory object called `testcase` under the current directory.

```
ccm create -t dir testcase
```

### Related topics

- [Adding a project to the current directory](#)
- [Deleting objects from the database](#)
- [Showing projects](#)

---

## Description and uses

The `create` command creates a new object and adds it to the current project in the following ways.

- When you create a new file or directory, it is added to the current directory, which must be part of a project.
- When you create an object in a non-shared project, its default state is *working*. When you create a file or directory in a shared project, its default state is *visible* if it is a non-product, and *shared* if it is a product.
- When you create a new object in a non-writable directory, a new directory version is checked out automatically.

If you are in a shared project and your current directory is non-modifiable, the directory is checked out and associated automatically with the default (or specified) task and is checked in to the *integrate* state. You can disable the automatic check-in feature by setting `shared_project_directory_checkin` to `FALSE` in your initialization file. (See [shared project directory checkin](#).)

- When you create a new project, it is created as a floating object, but you can make it a subproject in an existing project by using the `use -p` command.
- When you create a project, Rational Synergy creates a work area for it automatically. By default, the work area is located in `My Documents\Synergy\ccm_wa\database\project_name-version` (Windows) or `ccm_wa/database/project_name-version` (UNIX) in your home directory. (See [Modifying work area properties](#) for details.)
- To add members to a directory, it must be writable (that is, checked out). If you try to create an object in a non-modifiable directory, Rational Synergy checks out the directory automatically. You will need to check in the directory and the new object to make the new object available to other users.



---

## dcm command

See [Description and uses](#) for details. The `dcm` command supports the following subcommands:

- [Adding objects to a transfer set](#)
- [Creating a database definition](#)
- [Creating a transfer set](#)
- [Deleting a database definition](#)
- [Deleting a transfer set](#)
- [Deleting a package](#)
- [Generating a transfer package](#)
- [Marking database as up-to-date](#)
- [Modifying a database definition](#)
- [Modifying a transfer set](#)
- [Modifying DCM settings](#)
- [Receiving packages](#)
- [Recomputing the indirect change request members for a transfer set](#)
- [Recomputing the indirect release members for a transfer set](#)
- [Recomputing the members for a transfer set](#)
- [Removing objects from a transfer set](#)
- [Showing database definition information](#)
- [Showing DCM properties](#)
- [Showing database IDs](#)
- [Showing a DCM event summary](#)
- [Showing settings](#)
- [Showing receive lock](#)
- [Showing a DCM event summary](#)
- [Showing a specified DCM event](#)
- [Showing last generate times](#)
- [Showing the current DCM database ID](#)
- [Showing transfer set information](#)
- [Showing transfer set members](#)

- 
- [Showing transfer sets](#)
  - [Transferring generated packages](#)

**Prerequisite**

The current database must be initialized to use DCM.

---

## ***Adding objects to a transfer set***

This subcommand adds the specified objects to the specified transfer set. You can add projects, directories, and files with or without history. Adding a project, directory, or file with history adds all versions of that object to the transfer set. Adding an object to a transfer set might also automatically add associated objects as indirect members. For example, adding a project includes its members, and adding a task adds the task's associated objects. See the [Rational Synergy Distributed](#) book for details about how objects are expanded.

**Note** Adding an object to a transfer set does not guarantee that it will be sent. Some objects are excluded by built-in exclusion rules. The settings on the transfer set might also exclude some objects.

Conversely, the predefined **Entire Database** transfer set automatically includes all projects, directories, files, tasks, folders and baselines. You cannot add objects to the **Entire Database** transfer set.

A DCM manager or an administrator can add objects to a transfer set.

```
ccm dcm -add -ts|-transfer_set transfer_set_spec
          ([-h|-history] | [-nh|-no_history]) object_spec...
```

**-h|-history**

When using this option for files, directories, or projects, specifies to add all versions in the history of the object as history members. This option does not impact tasks or folders. When an object is a history member and a new version is checked out, the new version automatically becomes a history member.

The default is determined by the current DCM settings. The default is not to add versions in the history of the object as history members.

*object\_spec...*

Specifies the object to be added to the transfer set. See [Object specification](#) for details.

**-ts|-transfer\_set *transfer\_set\_spec***

Specifies the transfer set to which objects will be added as members. You must specify a single transfer set with the *transfer\_set\_spec*. See [Transfer set specification](#) for more information.

---

## Example

- Add the `infotec-23` project to the **InfoServer source** transfer set.

```
ccm dcm -add -ts "InfoServer source" infotec-23:project:1
```

You also can use the query selection set to specify object names.

## Related topics

- [Creating a transfer set](#)
- [Deleting a transfer set](#)
- [Generating a transfer package](#)
- [Modifying a transfer set](#)
- [Receiving packages](#)
- [Recomputing the members for a transfer set](#)
- [Removing objects from a transfer set](#)
- [Showing DCM properties](#)
- [Showing database IDs](#)
- [Showing transfer set information](#)
- [Showing transfer set members](#)
- [Showing transfer sets](#)

---

## Creating a database definition

This subcommand creates a DCM database definition representing another database in the DCM cluster. This is required before you can replicate any data to that database. The database definition should use a database identifier that matches the DCM database identifier of that database. The database definition defines how DCM will transfer packages to that database, and if and how those packages can be automatically received.

The **Any** database definition represents a special predefined database for producing *broadcast packages*. These are produced by a DCM generate and are received in any database with a compatible Synergy version.

A DCM manager or an administrator can create a database definition.

```
ccm dcm -c|-create -dbid|-database_id new_database_id
    [-desc|-description description]
    [-tm|-transfer_mode ((manual|manual_copy) | direct |
    (cp|copy|local_copy) | (rcp|remote_copy) |
    (ftp|file_transfer_protocol) | (user|user_defined))]
    [-ar|-automatic_receive] | [-noar|-noautomatic_receive]
    [-rb|-run_in_background] | [-norb|-norun_in_background]
    [-host host] [-os|-operating_system (unix | (windows|windows_nt))]
    [-path path] [-tp|-transfer_path path] [-ccm_home ccm_home]
    ([-zip] | [-nozip]) ([-ga|-generate_allowed] |
    [-noga|-nogenerate_allowed]) ([-handover_allowed] |
    [-nohandover_allowed]) ([-hidden] | [-nohidden])
    [-location location] [-admin_info admin_info]
```

`-admin_info admin_info`

Specifies the contact information for the database administrator. The *admin\_info* value can be any string that doesn't contain newline characters. For example, phone numbers and e-mail addresses are valid entries.

`-ar|-automatic_receive`

Specifies that after a DCM generate to this database, the transfer package will be received automatically. See "Automatic receive setup and use" in the [Rational Synergy Distributed](#) book for details. The default is `-noar|-noautomatic_receive`.

`-ccm_home path`

Specifies the Synergy `$CCM_HOME` installation path. Synergy uses this path to perform an automatic receive for packages generated for the database definition. Enter an absolute path if a UNIX server hosts the corresponding database. Enter a UNC path if a Windows server hosts it.

---

`-dbid|-database_id new_database_id`

Specifies the DCM database identifier for the new database definition. The *new\_database\_id* cannot be longer than 8 characters and must be unique in a DCM cluster. In a cluster using lowercase databases, identifiers should be unique without respect to case. For example, do not use "a" and "A" for two different definitions.

See [DCM restrictions](#) for details on naming restrictions for DCM databases.

`-desc|-description description`

Specifies a description of the database definition you're creating. The *description* cannot contain newline characters.

`-ga|-generate_allowed`

Specifies that the new database can use DCM generate. This is the default.

`-handover_allowed`

Specifies to give control of objects to the new database. The default is `-nohandover_allowed`.

`-hidden`

Specifies to hide the new database definition in dialogs that list database identifiers. The default is `-nohidden`.

`-host host`

Specifies the name of the machine hosting the database. This is required for the remote copy and file transfer protocol transfer modes. Be sure that the *host* name is valid and is set to an IP address.

`-location location`

Specifies the geographic location of the database (for example, *Irvine, California*). The location can be any string not containing newline characters.

`-noar|-noautomatic_receive`

Specifies that after a DCM generate to this database, the transfer package should not be received automatically. The package must be received manually. This is the default.

---

`-noga` | `-nogenerate_allowed`

Specifies that the new database cannot use DCM generate. The default is `-ga` | `-generate_allowed`.

`-nohandover_allowed`

Specifies not to give control of objects to the new database. This is the default.

`-nohidden`

Specifies to show the new database definition for dialogs that list database identifiers. This is the default.

`-norb` | `-norun_in_background`

Specifies not to run automatic receive in the background, if it is being used.

If you're using automatic receive, when DCM generate completes generating and transferring the package, it will start a session on the destination database to start the receive of the package. If you're using `-norun_in_background`, the generate waits for completion of the receive and shows the results of the receive in the destination database.

This is the default.

`-nozip`

Specifies that the transfer packages generated for the database will not be compressed. This is the default for the `direct` transfer mode. You cannot use this option with the `file transfer protocol` transfer mode.

`-rb` | `-run_in_background`

Specifies to run automatic receive in the background, if it is being used.

If you're using automatic receive, when DCM generate completes generating and transferring the package, it will start a session on the destination database to start the receive of the package. If you're using `-run_in_background`, the generate does not wait for completion of the receive and immediately returns. This means that you don't have to wait for the transfer package to be received to continue using your session, but you won't see if the receive was successful. You can view details of the receive later, in the DCM event log in the destination database.

The default is `-norb` | `-norun_in_background`.

---

`-os|-operating_system (unix|(windows|windows_nt))`

Specifies the operating system for the machine hosting the database.

The default is to use the same setting as the server for the current database.

`-path path`

Specifies the path to the database. Use an absolute path for UNIX servers; use a UNC path for Windows servers. This option requires a database path if you use a transfer mode other than manual.

`-tp|-transfer_path transfer_path`

Specifies the transfer path to the database. The transfer path is the location where transfer packages are placed. This is an absolute path for UNIX servers, and a UNC path for Windows servers. If you don't specify a transfer path or if you use an empty string, Synergy places the packages under the **dcm/receive** directory under the database path.

`-tm|-transfer_mode value`

Specifies the transfer mode to use for the new database definition. The default is manual. The transfer mode defines the way transfer packages are transferred to the destination database:

The *transfer\_mode* option must have one of the following values:

- `manual | manual_copy`

The package is not copied by DCM and will have to be manually transferred.

- `cp | copy | local_copy`

The package is generated and then copied to the destination database's transfer path.

- `direct`

The package is generated directly into the destination database's transfer path.

- `ftp | file_transfer_protocol`

The package is generated and then transferred to the destination database's transfer path using *ftp*.

- `rcp | remote_copy`

The package is generated and then remote copied to the destination database's transfer path using *rcp*.

- `user | user_defined`



---

The package is generated and then transferred by invoking a user-customized shell script (Unix) or batch file (Windows).

See the [Rational Synergy Distributed](#) book for further details about transfer modes.

-zip

Specifies that the transfer packages generated for the database will be compressed. This is the default for all transfer modes except `direct`; you cannot use this option with `direct`.

## Related topics

- [Deleting a database definition](#)
- [Modifying a database definition](#)
- [Showing database definition information](#)
- [Showing the current DCM database ID](#)

---

## Creating a transfer set

This subcommand creates a transfer set. A transfer set represents a collection of objects to be replicated to other databases. The objects to be replicated can be added to the transfer set. See [Adding objects to a transfer set](#) for details. The **Entire Database** transfer set is a special predefined database that automatically includes all projects, directories, files, tasks, folders and baselines.

A DCM manager or an administrator can create a transfer set.

```
ccm dcm -c|-create -ts|-transfer_set new_transfer_set_name
  ([-email email_address] | [-noemail])
  [-ep|-email_policy (generate | transfer | always)]
  [-crsc|-change_request_scope|-ps|-problem_scope (none |
  (crs|crs only|change_requests|problems) |
  (crs_and_tasks|crs and tasks|
  change_requests_and_tasks|problems_and_tasks) |
  (crs_tasks_and_objects|crs, tasks and objects|
  change_requests_tasks_and_objects|problems_tasks_and_objects))]
  [-crq|-change_request_query|-pq|-problem_query cr_query]
  ([-cumcrsc|-cumulative|-cumulative_change_request_scope] |
  [-nocumcrsc|-nocumulative|-nocumulative_change_request_scope])
  [-rsc|-release_scope (none | releases |
  (releases_templates|releases_and_templates|releases and templates))]
  [-rq|-release_query release_query]
  ([-cumrsc|-cumulative_release_scope] |
  [-nocumrsc|-nocumulative_release_scope]) ([-exclude_products] |
  [-noexclude_products]) ([-exclude_imported_objects] |
  [-noexclude_imported_objects])
  ([-exclude_nct|-exclude_non_completed_tasks] |
  [-noexclude_nct|-noexclude_non_completed_tasks])
  ([-exclude_typedefs] | [-noexclude_typedefs]) ([-exclude_db_info] |
  [-noexclude_db_info]) ([-ib|-include_baselines] |
  [-noib|-noinclude_baselines]) [-exclude_types type1,type2,...]
  ([-ferp] | [-noferp]) ([-local_parallel] | [-nolocal_parallel])
  [-dir|-directory generate_dir]
```

```
-crq|-change_request_query|-pq|-problem_query cr_query
```

Specifies the change request query to be used with the change request scope for the transfer set. You can use the change request query only if you set the change request scope to a value other than `none`.

The value must either be blank or a valid query expression. A blank value means to query for all change requests. This is the default value.

---

`-crsc | -change_request_scope | -ps | -problem_scope scope`

Specifies how to include change requests and their associated objects in transfer packages generated using the transfer set. Choose from the following scopes:

- `none`

Change requests are not included automatically. This is the default.

- `crs | crs only | change_requests | problems`

Change requests found by the change request query are included along with their attachments.

- `crs_and_tasks | crs and tasks | change_requests_and_tasks | problems_and_tasks`

Change requests found by the change request query are included along with their attachments and associated tasks.

- `crs_tasks_and_objects | crs, tasks and objects | change_requests_tasks_and_objects | problems_tasks_and_objects`

Change requests found by the change request query are included along with their attachments and associated tasks, and the associated objects of each of those tasks.

Note that change requests, tasks, or other objects added explicitly as direct members of the transfer set are not affected by this option.

`-cumcrsc | -cumulative | -cumulative_change_request_scope`

Specifies that the change request scope for the new transfer set is cumulative. The change request scope and query for transfer sets is evaluated with each generate or generate preview operation. However, if you specify `-cumulative`, older members found by previous queries that are not found by current queries will never be removed. That is, the indirect (query-based) membership for change requests will be added to and thus, will be cumulative.

`-cumrsc | -cumulative_release_scope`

Specifies that the release scope for the new transfer set is cumulative. The release scope and query for transfer sets are evaluated with each generate or generate preview operation. However, if `-cumulative_release_scope` is specified, older members found by previous queries that are not found by the current queries will never be removed. That is, the indirect (query-based) membership for releases will only be added to and thus, will be cumulative.

---

`-dir|-directory generate_dir`

For the new transfer sets, specifies to use the specified generate directory to prepare generated transfer packages for transfer modes other than direct. The *generate\_dir* value represents a server path. Use an absolute path for UNIX servers; use a UNC path for Windows servers. If you use a blank string, Synergy uses the default generate directory, located under the **dcm/generate** directory under the database path. This is the default.

`-email email_address`

Specifies the e-mail address of the person(s) who will receive e-mail notification following a generate, receive, or transfer for the transfer set.

You can define multiple e-mail recipients for the transfer set by separating the addresses with a space or comma. If you want to define e-mail lists, you can set up e-mail aliases or distribution lists by using the facilities of your mail server. To learn how to do this, consult your mail server and operating system.

The default is `-noemail`.

`-ep|-email_policy policy`

Specifies the e-mail policy that is used during generate and transfer operations. This option supports the following e-mail policies:

- `Transfer` - Specifies that an e-mail message is sent only when you transfer a non-empty package to the destination database. Moreover, no message is sent if there are no objects included after the DCM generate operation.
- `Generate` - Specifies that an e-mail message is sent when you generate or transfer a populated transfer package. However, no message is sent if there are no objects included after the DCM generate operation. This is the default.
- `Always` - Specifies that an e-mail message is sent whenever you generate or transfer a populated transfer package. This includes occasions when there are no objects included after you perform the DCM generate operation or when you generate a package that is not automatically delivered to the destination database.

`-exclude_db_info`

Specifies to exclude information about database definitions from transfer packages generated from the new transfer set. The default is `-noexclude_db_info`.

`-exclude_imported_objects`

Specifies to exclude objects created in other databases from transfer packages generated from the new transfer set. The default is `-noexclude_imported_objects`.

---

`-exclude_nct|-exclude_non_completed_tasks`

Specifies to exclude tasks that have not been completed from transfer packages generated from the new transfer set.

The default is `-noexclude_non_completed_tasks`.

`-exclude_products`

Specifies to exclude products from transfer packages generated from the new transfer set. The default is `-noexclude_products`.

`-exclude_types type1,type2,...`

Specifies to exclude objects of the specified types from transfer packages generated from the new transfer set. The value must be a list of zero or more type names separated by commas and optional spaces. The default is an empty list.

`-exclude_typedefs`

Specifies to exclude type definitions from transfer packages generated from the new transfer set. The default is `-noexclude_typedefs`.

`-ferp`

Specifies to fully expand the update properties of projects that are members of the transfer set. This option causes the following to occur:

- All associated objects of tasks that are members are also included, even if they are not members of the project hierarchy.
- All folders, tasks, and baseline projects in the project's update properties are included, even for projects in a static state.
- All the subprojects of each baseline project are included, even if they are not directly used by the project hierarchy's update properties.

Note that this can significantly increase the number of indirect members of a transfer set and increase the time it takes to compute indirect members. For details, see "Fully expand reconfigure properties" in the [Rational Synergy Distributed](#) book.

The default is `-noferp`.

---

`-ib|-include_baselines`

Specifies to include any baselines that are associated with objects that are members of the transfer. The default is determined by the DCM setting of `Default Include Baselines`. The default is `-noinclude_baselines`.

`-local_parallel`

Specifies to send parallel notifications through e-mail to local owners of parallel object versions received from transfer packages generated from the new transfer set. This is the default.

`-nocumrsc|-nocumulative|-nocumulative_change_request_scope`

Specifies that the change request scope for the new transfer set is not cumulative. The change request scope and query for transfer sets is evaluated with each `generate` or `generate preview` operation. If you use this option, older members found by previous queries that are not found by the current queries will be removed because they're indirect (query-based) members of the transfer set.

This is the default.

`-nocumrsc|-nocumulative_release_scope`

Specifies that the release scope for the new transfer set is not cumulative. The release scope and query for transfer sets is evaluated with each `generate` or `generate preview` operation. If you use this option, older members found by previous queries that are not found by the current queries will be removed because they're indirect (query-based) members of the transfer set. This is the default.

`-noemail`

Specifies that e-mail should not be sent following a `generate`, `receive`, or `transfer` for the transfer set. This is the default.

`-noexclude_db_info`

Specifies to include information about database definitions in transfer packages generated from the new transfer set. This is the default.

`-noexclude_imported_objects`

Specifies to include objects created in other databases in transfer packages generated from the new transfer set. This is the default.

---

`-noexclude_nct|-exclude_non_completed_tasks`

Specifies to include tasks that have not been completed to transfer packages generated from the new transfer set. This is the default.

`-noexclude_products`

Specifies to include products in transfer packages generated from the new transfer set. This is the default.

`-noexclude_typedefs`

Specifies to include type definitions in transfer packages generated from the new transfer set. This is the default.

`-noferp`

Specifies not to fully expand the update properties of projects that are members of the transfer set. This option causes the following to occur:

- Associated objects of tasks that are in a project's update properties will only be included if they are members of the project hierarchy.
- Folders, tasks, and baseline projects for projects in static states are not included.
- Subprojects of the baseline project are included, even if they are directly used by a project's update properties.

This is the default. For details, see "Fully expand reconfigure properties" in the [Rational Synergy Distributed](#) book.

`-noib|-noinclude_baselines`

Specifies not to automatically include baselines that are associated with objects that are members of the transfer set. The default is determined by the DCM setting of `Default Include Baselines`. This is the default.

`-nolocal_parallel`

Specifies not to send parallel notifications through e-mail to local owners of parallels for received objects from transfer packages generated from the new transfer set. The default is `-local_parallel`.

`-nomail`

Specifies that e-mail should not be sent for a generate, receive, or transfer for the transfer set. This is the default.

---

`-rq|-release_query release_query`

Specifies a release query to use with the release scope for the new transfer set. The `release_query` must either be a blank string or a valid query expression. A blank string means to query for all releases. You can use the release query only if you set the release scope to a value other than `none`. This is the default.

`-rsc|-release_scope (none | releases | (releases_templates | releases_and_templates | releases and templates))`

Specifies the release scope for the new transfer set. Use the following scopes:

- `none`  
Release definitions are not automatically included.
- `releases`  
Release definitions found by the release query are automatically included as indirect query members. However, their corresponding process rules and folder templates are not automatically included.
- `releases_and_templates`  
Release definitions found by the release query are automatically included as indirect query members. For each release, its process rules and any user-defined folder templates used by the process rules are also automatically included as indirect query members. This is the default.

`-ts|-transfer_set transfer_set_name`

Specifies the name of the transfer set you are creating. The name can contain any characters, but it must be unique in this database.

## Examples

- Exclude object types from a transfer set (while you are creating the transfer set).  
`ccm dcm -create -ts transfer_set_spec -exclude_types "list_of_types"`
- Include products in a transfer set (while you are creating the transfer set).  
`ccm dcm -create -ts transfer_set_spec -noexclude_products`
- Exclude imported objects from a transfer set (while you are creating the transfer set).  
`ccm dcm -create -ts transfer_set_spec -exclude_imported_objects`
- Exclude database information from a transfer set (while you are creating the transfer set).  
`ccm dcm -create -ts transfer_set_spec -exclude_db_info`



---

## Related topics

- [Adding objects to a transfer set](#)
- [Creating a transfer set](#)
- [Deleting a transfer set](#)
- [Generating a transfer package](#)
- [Modifying a transfer set](#)
- [Receiving packages](#)
- [Recomputing the indirect release members for a transfer set](#)
- [Recomputing the members for a transfer set](#)
- [Removing objects from a transfer set](#)
- [Showing DCM properties](#)
- [Showing last generate times](#)
- [Showing transfer set information](#)
- [Showing transfer set members](#)
- [Showing transfer sets](#)
- [Transferring generated packages](#)

---

### ***Deleting a database definition***

This subcommand deletes a DCM database definition. When you delete a database definition, you also delete information about when DCM generate operations were performed to it using a transfer set. This is appropriate if the corresponding database no longer exists in the DCM cluster. If you're retiring the database rather than deleting it, you can set the database definition to be hidden or not allowed for generate.

A DCM manager or an administrator can delete a database definition.

```
ccm dcm -d|-delete -dbid|-database_id database_spec...
```

```
database_spec...
```

Specifies database definition(s) to be deleted. See [Database specification](#) for details.

### **Related topics**

- [Creating a database definition](#)
- [Modifying a database definition](#)
- [Showing database definition information](#)
- [Showing database IDs](#)

---

### ***Deleting a transfer set***

This subcommand deletes a transfer set. Deleting a transfer set also disassociates it from all the members that were directly added to it and their corresponding indirect members. When you delete a transfer set, you also delete information about when DCM generate operations were performed to it using that transfer set. You cannot delete the predefined Entire Database transfer set.

A DCM manager or an administrator can delete a transfer set.

```
ccm dcm -d|-delete -ts|-transfer_set transfer_set_spec...
transfer_set_spec...
```

Specifies transfer set(s) to be deleted. See [Transfer set specification](#) for details.

### ***Example***

- Delete one or more transfer sets.

```
ccm dcm -delete -ts "My transfer set"
```

### **Related topics**

- [Adding objects to a transfer set](#)
- [Creating a transfer set](#)
- [Generating a transfer package](#)
- [Modifying a transfer set](#)
- [Recomputing the indirect change request members for a transfer set](#)
- [Recomputing the indirect release members for a transfer set](#)
- [Recomputing the members for a transfer set](#)
- [Removing objects from a transfer set](#)
- [Showing transfer set information](#)
- [Showing transfer set members](#)
- [Showing transfer sets](#)
- [Transferring generated packages](#)

---

## Deleting a package

This subcommand deletes packages from a specified server file system location. A DCM manager or an administrator can delete packages. You must specify one of the options: `-generate`, `-receive`, or `-serverdir`.

```
ccm dcm -d|-delete -packages [-from_dbid|-from_database_id database_spec]
    [-to_dbid|-to_database_id database_spec]
    [-ts|-transfer_set transfer_set_spec] ([-gen|-generate] |
    [-rec|-receive] | [-soad|-save_offline_and_delete] |
    [-serverdir server_path])
```

`-from_dbid|-from_database_id database_spec`

Specifies the destination database from which the transfer package was generated. You can set the `database_spec` to a single database definition. See [Database specification](#) for details.

`-gen|-generate`

Specifies to search the server generate directory for the package to be deleted.

`-rec|-receive`

Specifies to search the server receive directory for the package to be deleted.

`-soad|-save_offline_and_delete`

Specifies to search the default server directory for save offline packages for the package to be deleted.

`-to_dbid|-to_database_id database_spec`

Specifies the destination database to which the transfer package will be generated. You can set the `database_spec` to a single database definition. See [Database specification](#) for details.

`-serverdir server_path`

Specifies the path to the server directory in the file system that contains the packages you want to delete.

---

`-ts|-transfer_set transfer_set_spec`

Specifies to filter the packages by the transfer set name used to generate the package. You can set the `transfer_set_spec` to a single transfer set. For details, see [Transfer set specification](#).

### Examples

- Delete all packages generated using transfer set `webapp` and residing in the server generate directory.

```
ccm dcm -delete -packages -ts webapp -generate
```

- Delete all packages from the server receive directory, generated from database `INDWEB`.

```
ccm dcm -delete -packages -from_dbid INDWEB -receive
```

### Related topics

- [Generating a transfer package](#)
- [Receiving packages](#)
- [Showing packages](#)
- [Transferring generated packages](#)

---

## Generating a transfer package

This subcommand generates a transfer package for a specified transfer set and destination database. If the transfer set's indirect members are out-of-date, then this command recomputes them from the current direct members in the transfer set. The direct and indirect members that have been added to the transfer set since the last generate time for that database, or that have been modified since the last generate time for that database are then included in the transfer package. Optionally, you can transfer the transfer package to the destination database by using the transfer mode defined for that database. If you transfer the package, you have the option to automatically receive the package into the destination database.

A DCM manager or an administrator can generate a transfer package.

```
ccm dcm -gen|-generate -dbid|-database_id database_spec
        -ts|-transfer_set transfer_set_spec
        [-lg|-last_generated last_generated_value]
        ([-email email_address] | [-noemail])
        ([-trn|-transfer [-rec|-receive ( [-wait] | [-nowait] )]] |
        [-notrn|-notransfer])
```

*-dbid|-database\_id database\_spec*

Specifies the destination database for which the transfer package will be generated.

You can set the *database\_spec* to a single database definition. See [Database specification](#) for details.

*-email email\_address*

Specifies the e-mail address of the person(s) who will receive e-mail notification following a generate for the transfer set.

You can define multiple e-mail recipients for the transfer set by separating the addresses with a space or comma. If you want to define e-mail lists, you can set up e-mail aliases or distribution lists by using the facilities of your mail server. To learn how to do this, consult your mail server and operating system.

If you don't set up an e-mail address, Synergy uses the e-mail address defined on the transfer set specified in the `dcm generate` command.

*-lg|-last\_generated last\_generated\_value*

Specifies the last time a generate occurred.

**Note** This option is for advanced users only.

If not specified, the package is generated to use the time at which a transfer package for the specified transfer set and database was last generated. Use this option to

---

generate a transfer package that includes earlier changes, such as when recovering from missing transfer packages.

The *last\_generated\_value* must have one of the following:

— *never*

— an integer index where 1 refers to the most recent generated transfer package.

If you select a timestamp that is not the most recent timestamp, the generated transfer package includes all objects that have changed or become members since that date. Also, the more recent timestamps are removed from the list.

**Caution** The *never* choice causes all previous time stamps to be removed from the list.

When *last\_generated\_value* is set to *never*, the transfer package is generated as if for the first time; the transfer package will not exclude an object regardless of when it was last modified or became a member of the transfer set.

`-noemail`

Specifies that e-mail should not be sent following a generate for the transfer set. By default, Synergy uses the e-mail setting from the transfer set.

`-notrn|-notransfer`

Specifies not to transfer the generated package to the destination database. You can use this option if the destination database has a transfer mode other than *direct*. This is the default.

`-nowait`

Specifies that an automatic receive should not wait until the receiving database has completed receiving other transfer packages.

**Caution** It is not safe to receive multiple transfer packages in a database concurrently. By default, DCM will receive one package at a time into a database. Use this option only if you are certain that the transfer packages do not contain overlapping objects. For example, if the same task object is present in two packages each generated from a different transfer set, these packages have an overlapping object. In this situation, use `-wait`, the default

---

`-rec|-receive`

Specifies to receive the generate package into the destination database. Use this option when the package is transferred to the destination database and if the host and database path are defined for the database definition. See "Automatic receive setup and use" in the [Rational Synergy Distributed](#) book.

`-trn|-transfer`

Specifies to transfer the generated package to the destination database. You can use this option if the destination database has a transfer mode other than `none` or `direct`. The default is `-notransfer`.

`-ts|-transfer_set transfer_set_spec`

Specifies the transfer set to be used for the DCM generate. You can set the `transfer_set_spec` to a single transfer set. For details, see [Transfer set specification](#).

`-wait`

Specifies that an automatic receive should wait until the receiving database has completed receiving other transfer packages. This is the default.

### **Example**

- Generate the transfer package for the **Secure transformer layer** transfer set and the **BST** database, and save it to transfer later.

```
ccm dcm -gen -ts "Secure transformer layer" -dbid BST
Computing transfer package...
Computing transfer package for 'Secure transformer layer' going to
database 'BST'...
115 objects will be included in transfer package for 'Secure
transformer layer' going to database 'BST'...
Generating transfer package...
...
DCM data generated to file
  '\\ccmsrv\ccmdb\appdevdb\dcm\generate\CA#7#BST#865889312.tar.gz'
Updating database...
DCM Generate completed successfully.
```



---

## Related topics

- [Adding objects to a transfer set](#)
- [Creating a transfer set](#)
- [Deleting a transfer set](#)
- [Generating a transfer package](#)
- [Receiving packages](#)
- [Showing last generate times](#)
- [Showing transfer set information](#)
- [Showing transfer set members](#)
- [Showing transfer sets](#)
- [Transferring generated packages](#)

---

## Marking database as up-to-date

This subcommand marks a database definition as up-to-date for a specified transfer set. When you create a database by unpacking a copy of the current database and changing its database identifier, it will already contain all the objects that existed when the database was backed up. This subcommand is useful when setting up replication to an existing database because it stops objects from being sent unnecessarily.

```
ccm dcm -mark_up_to_date -dbid|-database_id database_spec  
        -ts|-transfer_set transfer_set_spec [-force] [date]
```

*-database\_id database\_spec*

Specifies to generate the destination database for the transfer package. You can set the *database\_spec* to a single database definition. See [Database specification](#) for details.

*date*

Specifies the date that the database is up-to-date. By default, Synergy uses the current date and time. If a database was created by unpacking a backup of another database, you can use the date and time of the backup as the date for this command. Any changes made after that data will not exist in the copied database.

*-force*

Specifies that the operation will succeed even if the specified transfer set and destination database have generated previous packages. By default, the operation will succeed only if the transfer set and destination database have never generated a transfer package.

*-ts|-transfer\_set transfer\_set\_spec*

Specifies the transfer set to mark as up-to-date for the specified destination database. You can set the *transfer\_set\_spec* to a single transfer set. For details, see [Transfer set specification](#).

## Related topics

- [Showing DCM properties](#)

---

## Modifying a database definition

This subcommand modifies the specified database definitions. A DCM manager or an administrator can perform this operation.

```
ccm dcm -m|-modify -dbid|-database_id [-desc|-description description]  
      [-tm|-transfer_mode ((manual|manual_copy) | direct |  
      (cp|copy|local_copy) | (rcp|remote_copy) |  
      (ftp|file_transfer_protocol) | (user|user_defined))]  
      ([-ar|-automatic_receive] | [-noar|-noautomatic_receive])  
      ([-rb|-run_in_background] | [-norb|-norun_in_background])  
      [-host host] [-os|-operating_system (unix | (windows|windows_nt))]  
      [-path path] [-tp|-transfer_path path] [-ccm_home path]  
      ([-zip] | [-nozip]) ([-ga|-generate_allowed] |  
      [-noga|-nogenerate_allowed]) ([-handover_allowed] |  
      [-nohandover_allowed]) ([-hidden] | [-nohidden])  
      [-location location] [-admin_info admin_info]  
      [-new_dbid|-new_database_id new_dbid] database_spec...
```

`-admin_info admin_info`

Specifies to modify the contact information for the specified database. The *admin\_info* value can be any string that doesn't contain newline characters. For example, phone numbers and e-mail addresses are valid entries.

`-ar|-automatic_receive`

Specifies to automatically receive transfer packages for the specified databases. See "Automatic receive setup and use" in the [Rational Synergy Distributed](#) book for details.

`-ccm_home path`

For the specified databases, modify the Synergy \$CCM\_HOME installation path to be used to perform an automatic receive. Enter an absolute path if a UNIX server hosts the corresponding database. Enter a UNC path if a Windows server hosts it.

`database_spec`

Specifies the database definitions to be modified. See [Database specification](#).

`-desc|-description description`

Specifies to modify the specified database description. The description cannot contain newline characters.

---

`-ga|-generate_allowed`

Specifies to modify the database definitions so DCM generate can use them.

`-handover_allowed`

Specifies that the database definitions will be modified to allow handover of control of objects to them.

`-hidden`

Specifies to mark specified database definitions as hidden so that they don't appear in dialogs that list database identifiers. This option is useful when you want to retire a database definition without deleting it.

`-host host`

Specifies to modify the host for the specified database definitions. This is required for the remote copy and file transfer protocol transfer modes. Be sure that the *host* name is valid and is set to an IP address.

`-location location`

Specifies to modify the geographic location of the specified database. The location can be any string not containing newline characters.

`-new_dbid|-new_database_id new_dbid`

Specifies the new database ID that you want to assign to the specified database definition. To learn about restricted characters and why they are restricted, see [Naming restrictions](#).

`-noar|-noautomatic_receive`

Specifies not to receive the transfer package automatically. Packages must be received manually.

`-nohandover_allowed`

Specifies that the database definitions will be modified to disallow handover of control of objects to them.

`-nohidden`

Specifies to show the specified database definition in dialogs that list database identifiers.

---

`-noga` | `-nogenerate_allowed`

Specifies to modify the database definitions so DCM generate cannot use them.

[-norb|-norun\\_in\\_background](#)

`-nozip`

Specifies not to compress the transfer packages for the specified database. This is the default for the `direct` transfer mode. You cannot use this option with the `file transfer protocol` transfer mode.

[-rb|-run\\_in\\_background](#)

`-os` | `-operating_system` (`unix` | (`windows` | `windows_nt`))

Specifies that the operating system for the machine hosting the database be modified for the specified database definitions.

`-path` *path*

Specifies to modify the database path for the specified database definitions. Use an absolute path for UNIX servers; use a UNC path for Windows servers. You must enter a database path value if you use a transfer mode other than manual.

`-tm` | `-transfer_mode` *value*

Specifies the transfer mode to be modified for the specified database definition. The transfer mode defines the way transfer packages are transferred to the destination database:

The `-transfer_mode` option must have one of the following values:

- `manual` | `manual_copy`

The package is not copied by DCM and must be transferred manually.

- `cp` | `copy` | `local_copy`

The package is generated and then copied to the destination database's transfer path.

- `direct`

The package is generated directly into the destination database's transfer path.

- `ftp` | `file_transfer_protocol`

---

The package is generated and then transferred to the destination database's transfer path using *ftp*.

- `rcp` | `remote_copy`

The package is generated and then remote copied to the destination database's transfer path using *rcp*.

- `user` | `user_defined`

The package is generated and then transferred by invoking a user-customized shell script (Unix) or batch file (Windows).

See the [Rational Synergy Distributed](#) book for further details about transfer modes.

`-tp|-transfer_path` *transfer\_path*

Specifies to modify the transfer path to the database for the specified database definitions. The transfer path is the location where Synergy places transfer packages. This is an absolute path for UNIX servers, and a UNC path for Windows servers. If you leave a transfer path blank, Synergy places the packages under the **dcm/receive** directory under the database path.

`-zip`

Specifies to compress the transfer packages for the specified database. This is the default for the file transfer protocol transfer mode. You cannot use this option with `direct`.

## Related topics

- [Creating a database definition](#)
- [Deleting a database definition](#)
- [Showing database definition information](#)
- [Showing DCM properties](#)
- [Showing database IDs](#)
- [Showing the current DCM database ID](#)

---

## Modifying a transfer set

This subcommand modifies the specified transfer sets. A DCM manager or an administrator can perform this operation.

```
ccm dcm -m|-modify -ts|-transfer_set ([-email email_address] |
[-noemail]) [-ep|-email_policy (generate | transfer | always)]
[-crsc|-change_request_scope|-ps|-problem_scope (none |
(crs|crs only|change_requests|problems) |
(crs_and_tasks|crs and tasks|change_requests_and_tasks|
problems_and_tasks) | (crs_tasks_and_objects|crs,
tasks and objects|change_requests_tasks_and_objects|
problems_tasks_and_objects)))]
[-crq|-change_request_query|-pq|-problem_query cr_query]
([-cumcrsc|-cumulative|-cumulative_change_request_scope] |
[-nocumcrsc|-nocumulative|-nocumulative_change_request_scope])
[-rsc|-release_scope (none | releases |
(releases_templates|releases_and_templates|releases and templates))]
[-rq|-release_query release_query]
([-cumrsc|-cumulative_release_scope] |
[-nocumrsc|-nocumulative_release_scope]) ([-exclude_products] |
[-noexclude_products]) ([-exclude_imported_objects] |
[-noexclude_imported_objects])
([-exclude_nct|-exclude_non_completed_tasks] |
[-noexclude_nct|-noexclude_non_completed_tasks])
([-exclude_typedefs] | [-noexclude_typedefs]) ([-exclude_db_info] |
[-noexclude_db_info]) ([-ib|-include_baselines] |
[-noib|-noinclude_baselines]) [-exclude_types type1,type2,...]
([-ferp] | [-noferp]) ([-local_parallel] | [-nolocal_parallel])
[-dir|-directory generate_dir]
[-new_ts|-new_transfer_set new_transfer_set_name]
transfer_set_spec...
```

`-crq|-change_request_query|-pq|-problem_query cr_query`

Specifies to modify the change request query for the specified transfer sets. You can use the change request query only if you set the change request scope to a value other than `none`.

The value must either be blank or a valid query expression. A blank value means to query for all change requests.

`-crsc|-change_request_scope|-ps|-problem_scope scope`

Specifies to modify the change request scope of the specified transfer set. The scope defines how to include change requests and their associated objects in transfer packages generated using the transfer set. Choose from the following scopes:

- `none`

---

Change requests are not included automatically. This is the default.

- `crs|crs only|change_requests|problems`

Change requests found by the change request query are included along with their attachments.

- `crs_and_tasks|crs and tasks|change_requests_and_tasks|problems_and_tasks`

Change requests found by the change request query are included along with their attachments and associated tasks.

- `crs_tasks_and_objects|crs, tasks and objects|change_requests_tasks_and_objects|problems_tasks_and_objects`

Change requests found by the change request query are included along with their attachments and associated tasks, and the associated objects of each of those tasks.

Note that change requests, tasks, or other objects added explicitly as direct members of the transfer set are not affected by this option.

`-cumcrsc|-cumulative|-cumulative_change_request_scope`

Specifies to modify the change request scope for the specified transfer sets to cumulative. The change request scope and query for transfer sets is evaluated with each generate or generate preview operation. However, if you specify `-cumulative`, older members found by previous queries that are not found by current queries will never be removed. That is, the indirect (query-based) membership for change requests will be added to and thus, will be cumulative.

`-cumrsc|-cumulative_release_scope`

Specifies that the release scope for the specified transfer set is cumulative. The release scope and query for transfer sets are evaluated with each generate or generate preview operation. However, if you specify `-cumulative_release_scope`, older members found by previous queries that are not found by the current queries will never be removed. That is, the indirect (query-based) membership for releases will only be added to and thus, will be cumulative.

`-dir|-directory generate_dir`

Specifies to modify the generate directory for the specified transfer sets. Use the generate directory to prepare generated transfer packages for transfer modes other than `direct`. The `generate_dir` value represents a server path. Use an absolute path for UNIX servers; use a UNC path for Windows servers. If you use a blank string, Synergy uses the default generate directory, located under the **dcm/generate** directory under the database path. This is the default.



---

`-email email_address`

Specifies to modify the e-mail address for the specified transfer sets. This option delineates the e-mail address of the person(s) who will receive e-mail notification following a generate, receive, or transfer for the transfer set.

You can define multiple e-mail recipients for the transfer set by separating the addresses with a space or comma. If you want to define e-mail lists, you can set up e-mail aliases or distribution lists by using the facilities of your mail server. To learn how to do this, consult your mail server and operating system.

`-ep|-email_policy value`

Specifies to modify the e-mail policy for the specified transfer sets. This option supports the following e-mail policies:

- `Transfer` - Specifies that an e-mail message is sent only when you transfer a non-empty package to the destination database. Moreover, no message is sent if there are no objects included after the DCM generate operation.
- `Generate` - Specifies that an e-mail message is sent when you generate or transfer a populated transfer package. However, no message is sent if there are no objects included after the DCM generate operation.
- `Always` - Specifies that an e-mail message is sent whenever you generate or transfer a populated transfer package. This includes occasions when there are no objects included after you perform the DCM generate operation or when you generate a package that is not automatically delivered to the destination database.

`-exclude_db_info`

Specifies to exclude information about database definitions from transfer packages generated from the specified transfer set.

`-exclude_imported_objects`

Specifies to exclude objects created in other databases from transfer packages generated from the specified transfer set.

`-exclude_nct|-exclude_non_completed_tasks`

Specifies to exclude tasks that have not been completed from transfer packages generated from the specified transfer set.

---

`-exclude_products`

Specifies to exclude products from transfer packages generated from the specified transfer set.

`-exclude_types type1,type2,...`

Specifies to exclude objects of the specified types from transfer packages generated from the specified transfer set. The value must be a list of zero or more type names separated by commas and optional spaces.

`-exclude_typedefs`

Specifies to exclude type definitions from transfer packages generated from the specified transfer set.

`-ferp`

Specifies to fully expand the update properties of projects that are members of the specified transfer set. This option causes the following to occur:

- All associated objects of tasks that are members are also included, even if they are not members of the project hierarchy.
- All folders, tasks, and baseline projects in the project's update properties are included, even for projects in a static state.
- All the subprojects of each baseline project are included, even if they are not directly used by the project hierarchy's update properties.

Note that this can significantly increase the number of indirect members of a transfer set and increase the time it takes to compute indirect members. For details, see "Fully expand reconfigure properties" in the [Rational Synergy Distributed](#) book.

`-ib|-include_baselines`

Specifies to include baselines that are associated with objects that are members of the specified transfer sets. Use this option cautiously. If used incorrectly, you could replicate partial baselines. See "Include associated baselines" in the *Rational Synergy Distributed* book for details.

`-local_parallel`

Specifies to send parallel notifications through e-mail to local owners of parallel object versions received from transfer packages generated from the specified transfer set.

---

`-new_ts | -new_transfer_set new_transfer_set_name`

Specifies the new name of the transfer set.

`-nocumcrsc | -nocumulative | -nocumulative_change_request_scope`

Specifies to modify the change request scope for the specified transfer set to be not cumulative. The change request scope and query for transfer sets is evaluated with each generate or generate preview operation. If you use this option, older members found by previous queries that are not found by the current queries will be removed because they're indirect (query-based) members of the transfer set.

`-nocumrsc | -nocumulative_release_scope`

Specifies that the release scope for the specified transfer set is not cumulative. The release scope and query for transfer sets is evaluated with each generate or generate preview operation. If you use this option, older members found by previous queries that are not found by the current queries will be removed because they're indirect (query-based) members of the transfer set.

`-noemail`

Specifies that e-mail should not be sent following a generate, receive, or transfer for the specified transfer set.

`-noexclude_db_info`

Specifies to include information about database definitions in transfer packages generated from the specified transfer set.

`-noexclude_imported_objects`

Specifies to include objects created in other databases in transfer packages generated from the specified transfer set.

`-noexclude_nct | -exclude_non_completed_tasks`

Specifies to include tasks that have not been completed to transfer packages generated from the specified transfer set.

`-noexclude_products`

Specifies to include products in transfer packages generated from the specified transfer set.

---

`-noexclude_typedefs`

Specifies to include type definitions in transfer packages generated from the specified transfer set.

`-noferp`

Specifies not to fully expand the update properties of projects that are members of the transfer set. This option causes the following to occur:

- Associated objects of tasks that are in a project's update properties will only be included if they are members of the project hierarchy.
- Folders, tasks, and baseline projects for projects in static states are not included.
- Subprojects of the baseline project are included, even if they are directly used by a project's update properties.

This is the default. For details, see "Fully expand reconfigure properties" in the [Rational Synergy Distributed](#) book.

`-noib|-noinclude_baselines`

Specifies not to automatically include baselines that are associated with objects that are members of the specified transfer set.

`-nolocal_parallel`

Specifies not to send parallel notifications through e-mail to local owners of parallel object versions received from transfer packages generated from the new transfer set.

`-rq|-release_query release_query`

Specifies to modify the release query to use with the release scope for the specified transfer sets. The *release\_query* must either be a blank string or a valid query expression. A blank string means to query for all releases. You can use the release query only if you set the release scope to a value other than `none`.

`-rsc|-release_scope value`

Specifies the release scope for the specified transfer set. Use the following scopes:

- `none`  
Release definitions are not automatically included.
- `releases`

---

Release definitions found by the release query are automatically included as indirect query members. However, their corresponding process rules and folder templates are not automatically included.

- `releases_templates|releases_and_templates |releases and templates`

Release definitions found by the release query are automatically included as indirect query members. For each release, its process rules and any user-defined folder templates used by the process rules are also automatically included as indirect query members.

*transfer\_set\_spec...*

Specifies the transfer set(s) to be modified. See [Transfer set specification](#) for details.

### **Example**

- Change the change request scope and release scope for transfer sets **client** and **server**.

```
ccm dcm -modify -ts -crsc crs -rsc releases_and_templates client
server
```

### **Related topics**

- [Adding objects to a transfer set](#)
- [Creating a transfer set](#)
- [Deleting a transfer set](#)
- [Modifying a transfer set](#)
- [Recomputing the indirect change request members for a transfer set](#)
- [Recomputing the indirect release members for a transfer set](#)
- [Recomputing the members for a transfer set](#)
- [Removing objects from a transfer set](#)
- [Showing transfer set information](#)
- [Showing transfer set members](#)
- [Showing transfer sets](#)
- [Transferring generated packages](#)

---

## Modifying DCM settings

This subcommand indicates which DCM settings are to be changed. You must specify at least one of the options associated with this command.

```
ccm dcm -m|-modify -settings
  [-desc|-description description_of_database]
  [-location location]
  [-admin_info admin_info]
  [-default_add_history|-nodefault_add_history]
  [-default_include_baselines|-nodefault_include_baselines]
  [-ignore_maintain_wa|-noignore_maintain_wa]
  [-update_db_info|-noupdate_db_info]
  [-keep_typedefs|-nokeep_typedefs]
  [-event_log_size log_size]
  [-parallel_checking parallel_check_keyword]
  [-update_releases release_action_keyword]
  [-add_receive_control_transition transition]
  [-remove_receive_control_transition transition]
  [-no_of_generate_times generate_times]
  [-no_of_old_generate_times old_generate_times]
  [-old_generate_time_resolution old_generate_resolution]
  [-update_rft|-noupdate_rft]
```

`-add_receive_control_transition transition`

Adds a valid state transition to the Receive Control Transitions list. This added state transition will be allowed when receiving an object that is controlled in the current database. The value that you specify for *transition* must be in the following form:

*from\_state:to\_state*

where:

*from\_state* must be a valid state.

*to\_state* must be a valid state for which a transition exists from the specified *from\_state*.

`-admin_info admin_info`

Specifies the contact information for the database administrator. The *admin\_info* value can be any string that doesn't contain newline characters. For example, phone numbers and e-mail addresses are valid entries.

`-default_include_baselines`

Specifies that baselines that are associated with transfer set members are included in transfers sets.

---

`-desc|-description description`

Specifies a description for this database. The *description* cannot contain newline characters.

`-default_add_history`

Specifies that objects that are added to a transfer set are added along with their predecessors.

`-event_log_size log_size`

Specifies the maximum number of entries in the event log. You can set the *log\_size* option to be any whole number except zero. When the event log list reaches the value that you specify for *log\_size*, each new entry replaces the oldest entry.

`-ignore_maintain_wa`

Specifies that the projects created by import or XML import (such as during a DCM receive operation) will be created without a maintained work area. You can turn on work area maintenance for a project again by using the `ccm wa -wa` command.

`-keep_typedefs`

Specifies that type definitions are kept after a receive operation is completed.

`-location location`

Specifies the geographic location of the database (for example, Irvine, California). The location can be any string not containing newline characters.

`-no_of_generate_times generate_times`

Specifies the number of generate times that DCM stores. The value that you specify for *generate\_times* can be any whole number except zero that is greater than or equal to the number of old generate times.

`-no_of_old_generate_times old_generate_times`

Specifies the number of old generate times that DCM stores. The value that you specify for *old\_generate\_times* can be any whole number except zero that is less than or equal to the number of generate times.

---

`-nodefault_add_history`

Specifies that objects that are added to a transfer set are added without their predecessors. This is the default.

`-nodefault_include_baselines`

Specifies that baselines that are associated with transfer set members are excluded from transfers sets. This is the default.

`-noignore_maintain_wa`

Specifies that projects that were created by import or XML import will have a maintained work area. This is the default.

`-nokeep_typedefs`

Specifies that type definitions are deleted after a receive operation is completed. This is the default.

`-nouupdate_db_info`

Specifies that DCM database information is not updated during a receive operation.

`-nouupdate_rtf`

Specifies that process rules and their associated folder templates are not updated during a receive operation.

`-old_generate_time_resolution old_generate_resolution`

Specifies the interval between the old generate times (expressed in units of days). The value that you specify for *old\_generate\_resolution* can be any whole number, except zero, that is a floating number.

`-parallel_checking (none|created|updated)`

Specifies the type of parallel checking that is performed during a DCM receive operation. The values that you can specify, *none*, *created*, and *updated*, are case sensitive.

`-remove_receive_control_transition transition`

Removes the specified *transition* from the Receive Control Transitions list. The removed state transition will no longer be allowed when receiving an object that is



---

controlled in the current database. The value that you specify for *transition* must be in the following form:

*from\_state:to\_state*

where:

*from\_state* must be a valid state that is in the Receive Control Transitions list.

*to\_state* must be a valid state for which a transition exists from the specified *from\_state*.

-update\_db\_info

Specifies that DCM database information might be updated during a receive operation. This information is updated from data that is in the DCM Information file. This is the default.

-update\_rtf

Specifies that process rules and their associated folder templates might be updated during a receive operation. This is the default.

-update\_releases (none|active|inactive)

Specifies how release definitions are updated on a DCM receive. The following values that you can specify, *none*, *active*, and *inactive*, are case sensitive:

- *none*

Specifies that release definitions are neither created nor updated.

- *active*

If the DCM transfer package includes release definitions, any release definitions that exist in the receiving database are updated, but only new active release definitions are created. If the DCM transfer package includes release table information only (from Telelogic Synergy 6.2 or earlier), release definitions for releases are created as active releases.

- *inactive*

If the DCM transfer package includes release definitions, these are created or updated in the receiving database. If the DCM transfer package includes release table information only (from Telelogic Synergy 6.2 or earlier), release definitions for releases are created as inactive releases.

### **Example**

- Change the settings for the event log size and the number of old generate times.

```
ccm dcm -m -settings -event_log_size 25 -no_of_old_generate_times 2
```

---

## Related topics

- [Showing settings](#)
- [Showing last generate times](#)
- [Showing a DCM event summary](#)
- [Showing transfer set information](#)

---

## Receiving packages

This subcommand receives transfer packages that were generated for this database or that are broadcast packages or save offline packages. You can specify which packages to receive by specifying the generating database or transfer set or generating database and transfer set. By default, packages generated for this database by any other database and with any transfer set are received. Packages are received in the same time order they are generated.

This subcommand can be performed by an administrator.

```
ccm dcm -rec|-receive [-dbid|-database_id database_spec]  
    [-ts|-transfer_set transfer_set_spec] [-a|-all] [-im|-ignore_missing]  
    ([-wait] | [-nowait]) ([-ic|-ignore_checks] |  
    [-noic|-noignore_checks]) ([-ivdc|-ignore_version_delimiter_check] |  
    [-noivdc|-noignore_version_delimiter_check])  
    ([-irdc|-ignore_release_delimiter_check] |  
    [-noirdc|-noignore_release_delimiter_check])  
    ([-itsc|-ignore_time_sync_check] |  
    [-noitsc|-noignore_time_sync_check]) [-dir|-directory receive_dir]
```

**-a|-all**

Specifies to receive all transfer packages for all transfer sets. This option cannot be used with `-database_id` or `-transfer_set`.

**-database\_id *database\_spec***

Specifies to receive packages only from the specified database. You can set the *database\_spec* to a single database definition. See [Database specification](#) for details.

**-dir|-directory *receive\_dir***

Specifies that the transfer packages are in the specified *receive\_dir* on the server. By default, packages are received from the **dcm/receive** directory under the current database path. The *receive\_dir* represents a server path. For UNIX servers, use an absolute path. For Windows servers, use a UNC path.

**-ignore\_checks**

Specifies that if any of the following checks fail, the operation will ignore them and continue:

- version delimiter check
- release delimiter check
- time synchronization check

---

The option is equivalent to specifying the [-ivdc-ignore version delimiter check](#), [-irdc-ignore release delimiter check](#), and [-itsc-ignore time sync check](#) options.

`-irdc|-ignore_release_delimiter_check`

Specifies that if the release delimiter check fails, the operation will ignore the condition and continue.

By default, if the release delimiter in the generate database is not the same as that in the receiving database, the receive will fail. All databases in a DCM cluster should use the same release delimiter.

`-itsc|-ignore_time_sync_check`

If the transfer package appears to have been generated in the future, specifies to ignore the condition and continue.

By default, the receive will fail if this condition is detected. This usually happens when either the computer that generated the package or the one receiving it or both have an incorrect time zone or time setting. Correcting the time allows DCM to work properly across time zones.

`-ivdc|-ignore_version_delimiter_check`

If the version delimiter check fails, specifies to ignore the condition and continue.

By default, if the version delimiter in the generate database is not the same as that in the receive database, the receive will fail.

`-im|-ignore_missing`

Tells DCM to ignore missing transfer packages.

**Caution** This option might result in empty directory entries or missing relationships.

`-noic|-noignore_checks`

Specifies to report a warning and fail to continue if any of the following checks fail:

- Version delimiter check

- 
- Release delimiter check
  - Time synchronization check

This option is equivalent to specifying [--noivdc|noignore\\_version\\_delimiter\\_check](#), [-noirdc|noignore\\_release\\_delimiter\\_check](#), and [-noitsc|noignore\\_time\\_sync\\_check](#). This is the default.

`-noirdc|noignore_release_delimiter_check`

Specifies to report an error and fail to continue if the release delimiter check fails. This is the default. See "Description and uses" in the [release command](#) for details on release delimiter settings.

`-noitsc|noignore_time_sync_check`

If the transfer package appears to have been generated in the future, specifies to report an error and fail to continue.

By default, the receive will fail if this condition is detected. This usually happens when either the computer that generated the package or the one receiving it or both have an incorrect time zone or time setting. Correcting the time allows DCM to work properly across time zones.

`-noivdc|noignore_version_delimiter_check`

Specifies to report an error and fail to continue if the version delimiter check fails. This is the default. See [DCM restrictions](#) for details on version delimiter restrictions.

`-nowait`

Specifies that the receive should not wait until the receiving database has completed receiving other transfer packages.

**Caution** It is not safe to receive multiple transfer packages in a database concurrently. By default, DCM will receive one package at a time into a database. Use this option only if you are certain that the transfer packages do not contain overlapping objects. For example, if the same task object is present in two packages each generated from a different transfer set, these packages have an overlapping object. In this situation, use `-wait`, the default

---

`-ts|-transfer_set transfer_set_spec...`

The `transfer_set_spec...` specifies the transfer set to use for the DCM receive. You can set the `transfer_set_spec` to a single transfer set. For details, see [Transfer set specification](#).

`-wait`

Specifies that the receive should wait until the receiving database has completed receiving other transfer packages. If you need to cancel this operation, use `CTRL+c`.

This is the default.

### **Example**

- Receive a transfer package from a source database.

```
ccm dcm -receive -ts "Entire Database" -dbid USIRJA
```

### **Related topics**

- [Generating a transfer package](#)
- [Recomputing the indirect change request members for a transfer set](#)
- [Recomputing the indirect release members for a transfer set](#)
- [Recomputing the members for a transfer set](#)
- [Showing last generate times](#)
- [Showing transfer set information](#)
- [Showing transfer set members](#)
- [Showing transfer sets](#)
- [Transferring generated packages](#)

---

### ***Recomputing the indirect change request members for a transfer set***

This subcommand recomputes the indirect change request members of a transfer set based on the current change request scope, change request query, and whether the scope is cumulative or not. This is automatically done when you perform a DCM generate.

This subcommand can be performed by a DCM manager or an administrator.

```
ccm dcm -recompute -crs|-change_requests|-problems  
        [-dbid|-database_id database_spec]  
        -ts|-transfer_set transfer_set_spec...
```

```
-database_id database_spec
```

Specifies the database identifier to use for the %to\_dbid keyword in a change request query for the transfer set. You can set the *database\_spec* to a single database definition. See [Database specification](#) for details.

```
transfer_set_spec...
```

Specifies the transfer set to use to recompute the indirect CR members. For details, see [Transfer set specification](#).

### **Related topics**

- [Adding objects to a transfer set](#)
- [Creating a transfer set](#)
- [Deleting a transfer set](#)
- [Modifying a transfer set](#)
- [Recomputing the indirect release members for a transfer set](#)
- [Recomputing the members for a transfer set](#)
- [Removing objects from a transfer set](#)
- [Showing transfer set information](#)
- [Showing transfer set members](#)
- [Showing transfer sets](#)

---

### ***Recomputing the indirect release members for a transfer set***

This subcommand recomputes the indirect release members of a transfer set based on the current release scope, release query, and whether the scope is cumulative or not. This is automatically done after a DCM generate.

This subcommand can be performed by a DCM manager or an administrator.

```
ccm dcm -recompute -rel|-release|-releases -ts|-transfer_set  
transfer_set_spec...
```

```
transfer_set_spec...
```

Specifies the transfer set to use to recompute the indirect release members. For details, see [Transfer set specification](#).

### **Related topics**

- [Adding objects to a transfer set](#)
- [Creating a transfer set](#)
- [Deleting a transfer set](#)
- [Modifying a transfer set](#)
- [Recomputing the members for a transfer set](#)
- [Removing objects from a transfer set](#)
- [Showing transfer set information](#)
- [Showing transfer set members](#)
- [Showing transfer sets](#)



---

### ***Recomputing the members for a transfer set***

This subcommand recomputes the indirect members of a transfer set based on the current direct members of the transfer set. This is automatically done when a DCM generate is performed.

This subcommand can be performed by a DCM manager or an administrator.

```
ccm dcm -recompute -ts|-transfer_set transfer_set_spec...
```

```
transfer_set_spec...
```

Specifies to recompute the specified transfer set. For details, see [Transfer set specification](#).

### **Related topics**

- [Adding objects to a transfer set](#)
- [Creating a transfer set](#)
- [Deleting a transfer set](#)
- [Modifying a transfer set](#)
- [Recomputing the indirect change request members for a transfer set](#)
- [Recomputing the indirect release members for a transfer set](#)
- [Removing objects from a transfer set](#)
- [Showing transfer set information](#)
- [Showing transfer set members](#)
- [Showing transfer sets](#)

---

## **Removing objects from a transfer set**

This subcommand removes the specified objects from the specified transfer sets. An object can be removed if it is a direct member of the transfer set. In other words, you can remove an object that was explicitly added to the transfer set. If an object is a history member, all versions of the object are removed. Note that removing an object does not immediately recompute the indirect members of the transfer set. To see the indirect members with the objects removed (based on the current members), recompute the members of the transfer set. (See [Recomputing the members for a transfer set](#).)

This subcommand can be performed by a DCM manager or an administrator.

```
ccm dcm -remove -ts|-transfer_set transfer_set_spec object_spec...
```

```
object_spec...
```

Specifies the objects to be removed from the transfer set. See [Object specification](#) for more information.

```
-ts|-transfer_set transfer_set_spec...
```

Specifies to remove objects from the specified transfer set. You can set the *transfer\_set\_spec* to a single transfer set. For details, see [Transfer set specification](#).

## **Related topics**

- [Adding objects to a transfer set](#)
- [Creating a transfer set](#)
- [Deleting a transfer set](#)
- [Modifying a transfer set](#)
- [Recomputing the members for a transfer set](#)
- [Removing objects from a transfer set](#)
- [Showing transfer set information](#)
- [Showing transfer set members](#)
- [Showing transfer sets](#)

---

### ***Showing database definition information***

This subcommand shows information about the specified database definitions.

```
ccm dcm -s|-sh|-show -dbid|-database_id database_spec...
```

*database\_spec*

Specifies the database definitions to show. See [Database specification](#) for details.

### **Related topics**

- [Creating a database definition](#)
- [Deleting a database definition](#)
- [Modifying a database definition](#)
- [Showing database definition information](#)
- [Showing DCM properties](#)

---

## Showing packages

This subcommand shows a summary of packages, based on specified criteria, or gives a preview of the objects contained in the packages. Packages can be generated by any database, with any transfer set, and by using the `save_offline_and_delete` command. If several packages for the same criteria are generated at different times, the output displays the earliest generated package first and the latest (most recently) generated package last.

```
ccm dcm -s|-sh|-show -packages [-members]
    [-from_dbid|-from_database_id database_spec]
    [-to_dbid|-to_database_id database_spec]
    [-ts|-transfer_set transfer_set_spec]
    ([-gen|-generate] | [-rec|-receive] |
    [-soad|-save_offline_and_delete] | [-serverdir server_path])
```

`-from_dbid|-from_database_id database_spec`

Specifies the destination database from which the package was generated. You can set the *database\_spec* to a single database definition. See [Database specification](#) for details.

`-gen|-generate`

Specifies to search the server generate directory for packages to be shown.

`-members`

Specifies to include information from both the DCM preview file and the package summary in the output.

`-rec|-receive`

Specifies to search the server receive directory for the packages.

`-serverdir server_path`

Specifies the path to the server directory in the file system that contains the packages.

`-soad|-save_offline_and_delete`

Specifies to search the default server directory for soad packages for the packages to be deleted.

---

`-to_dbid|-to_database_id database_spec`

Specifies the destination database to which the package will be generated. You can set the `database_spec` to a single database definition. See [Database specification](#) for details.

`-ts|-transfer_set transfer_set_spec`

Specifies to filter the packages by the transfer set name used to generate the package. You can set the `transfer_set_spec` to a single transfer set. For details, see [Transfer set specification](#).

### **Example**

- Show all packages in the server receive directory that were generated from the `ukweb` database.

```
ccm dcm -show -packages -receive -from_dbid ukweb
```

### **Related topics**

- [Deleting a package](#)
- [Transferring generated packages](#)

---

## Showing DCM properties

This subcommand shows the DCM properties of the specified objects. This shows whether the objects are members of any transfer sets and whether they have been modified since they were last sent to other databases.

```
ccm dcm -s|-sh|-show -prop|-properties object_spec...
```

*object\_spec...*

Specifies to show DCM properties for the specified object(s). See [Object specification](#) for details.

## Example

- Show DCM properties for the second object found by a previous query.

```
ccm dcm -show -properties @2
Object foo.txt-2:ascii:test71#1

Transfer Sets
Member of  Member Type  Member Since
preptest  indirect      09/03/09 10:06

Transfer Status
Database ID  Modified since last sent
b7int      TRUE
```

## Related topics

- [Showing last generate times](#)
- [Transferring generated packages](#)

---

## **Showing settings**

This subcommand shows a list of DCM settings. The output includes the following:

- Description of the current database.
- Location of the current database.
- Information about the person who is responsible for DCM administration issues for the current database.
- Default for history setting on DCM add operation (may be overridden by .ini file).
- Number of entries in event log.
- Whether `maintain_wa` is ignored by default.
- How release definitions are updated during a receive operation.
- Whether type definitions are kept after a receive operation.
- How parallel checking is performed during a receive operation.
- Which state transitions, if any, may be received for an object that is controlled in the current database.
- Whether DCM database information is updated during a receive operation.
- Whether process rules are updated during a receive operation.
- The number of generate times that DCM stores.
- The number of old generate times that DCM stores.
- The interval between the old generate times (expressed in units of days).

```
ccm dcm -show -settings
```

## **Related topics**

- [Modifying a transfer set](#)
- [Showing last generate times](#)

---

## Showing database IDs

This subcommand shows the DCM database identifiers for the known database definitions.

```
ccm dcm -s|-sh|-show -dbid|-database_id -a|-all [-f|-format format]  
          [-nf|-noformat] ([-ch|-column_header] | [-nch|-nocolumn_header])  
          [-sep|-separator separator] ([-sby|-sortby sortspec] |  
          [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat] [-u|-unnumbered]
```

`-a|-all`

Specifies to show all DCM database identifiers for the known database definitions.

`-ch|-column_header`

Specifies to use a column header in the output format. See [-ch|-column\\_headers](#) for details.

`-f|-format format`

Specifies the command output format. See [-f|-format](#) for details.

`-gby|-groupby groupformat`

Specifies how to group the command output. See [-gby|-groupby](#) for details.

`-nch|-nocolumn_header`

Specifies not to use a column header in the output format. See [-nch|-nocolumn\\_headers](#) for details.

`-nf|-noformat`

Specifies not to use column alignment. See [-nf|-noformat](#) for details.

`-ns|-no_sort`

Specifies that the command's output will not be sorted. See [-ns|-nosort](#) for details.

`-sep|-separator separator`

Specifies a different separator character. See [-sep|-separator](#) for details.

`-sby|-sortby sortspec`

Specifies how to sort the command output. See [-sby|-sortby](#) for details.



---

-u | -unnumbered

Suppresses automatic numbering of the command's output (that is, the output is unnumbered). See [-u|-unnumbered](#) for details.

### Related topics

- [Showing the current DCM database ID](#)
- [Transferring generated packages](#)

---

## Showing a DCM event summary

For each event, DCM captures the following types of information:

- **Event Information**

This is a summary of the event details. For DCM generate, transfer, and receive operations, this is the same information that would be included in any email that would be sent.
- **Messages**

This captures all messages that were shown for that operation. In the event of a failed operation, you should review these messages for further information about the failure.

```
ccm dcm -s|-sh|-show -el|-event_log [-dbid|-database_id database_spec
    [-ts|-transfer_set transfer_set_spec] [-f|-format format]
    [-nf|-noformat] ([-ch|-column_header] | [-nch|-nocolumn_header])
    [-sep|-separator separator] ([-sby|-sortby sortspec] |
    [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat]
```

`-ch|-column_header`

Specifies to use a column header in the output format. See [-ch|-column\\_headers](#) for details.

`-database_id database_spec`

Specifies the database definition to use to show the event log information for a specified database definition and transfer set. You can set the *database\_spec* to a single database definition. See [Database specification](#) for details.

`-el|-event_log`

Specifies that captured messages are displayed in the event log for the specified event.

`-f|-format format`

Specifies the command output format. See [-f|-format](#) for details.

`-gby|-groupby groupformat`

Specifies how to group the command output. See [-gby|-groupby](#) for details.

`-nch|-nocolumn_header`

Specifies not to use a column header in the output format. See [-nch|-nocolumn\\_headers](#) for details.

---

`-nf|-noformat`

Specifies not to use column alignment. See [-nf|-noformat](#) for details.

`-ns|-no_sort`

Specifies that the command's output will not be sorted. See [-ns|-nosort](#) for details.

`-sep|-separator separator`

Specifies a different separator character. See [-sep|-separator](#) for details.

`-sby|-sortby sortspec`

Specifies how to sort the command output. See [-sby|-sortby](#) for details.

`transfer_set_spec`

Specifies the transfer set to use. You can set `transfer_set_spec` to a single transfer set. For details, see [Transfer set specification](#).

### **Example**

- Show DCM events for database `sdg1`.  
`ccm dcm -show -event_log -dbid sdg1`

### **Related topics**

- [Showing a specified DCM event](#)

---

## Showing a specified DCM event

This subcommand shows the event log information for a specific DCM event.

```
ccm dcm -s|-sh|-show -el|-event_log -index number [-info]
          [-messages] [-f|-format format] [-nf|-noformat]
          ([-ch|-column_header] | [-nch|-nocolumn_header])
          [-sep|-separator separator] ([-sby|-sortby sortspec] |
          [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat]
```

`-ch|-column_header`

Specifies to use a column header in the output format. See [-ch|-column\\_headers](#) for details.

`-f|-format format`

Specifies the command output format. See [-f|-format](#) for details.

`-gby|-groupby groupformat`

Specifies how to group the command output. See [-gby|-groupby](#) for details.

`-index number`

Specifies the index number of an event. The index number is a unique number that identifies the event entry. The first event that is logged is assigned an index number of 1, the second event that is logged is assigned an index number of 2, and so on.

A positive non-zero index number is the absolute index of a specific event that is in the DCM event log. A zero or negative index number signifies a relative index; where 0 means the last entry, and a negative number is the number of entries back from the last entry.

`-info`

Displays information that is stored in the summary file for the specified event.

`-messages`

Displays captured messages from the log file for the specified event.

`-nch|-nocolumn_header`

Specifies not to use a column header in the output format. See [-nch|-nocolumn\\_headers](#) for details.

---

`-nf` | `-noformat`

Specifies not to use column alignment. See [-nf|-noformat](#) for details.

`-ns` | `-no_sort`

Specifies that the command's output will not be sorted. See [-ns|-nosort](#) for details.

`-sep` | `-separator separator`

Specifies a different separator character. See [-sep|-separator](#) for details.

`-sby` | `-sortby sortspec`

Specifies how to sort the command output. See [-sby|-sortby](#) for details.

`-ts` | `-transfer_set transfer_set_spec`

Specifies to filter the event summary by the transfer set name. You can set the `transfer_set_spec` to a single transfer set. For details, see [Transfer set specification](#).

## Related topics

- [Showing a DCM event summary](#)

---

## Showing last generate times

This subcommand shows the last times that a DCM generate was performed for a specified database definition and transfer set.

```
ccm dcm -s|-sh|-show -ts|-transfer_set transfer_set_spec
      -dbid|-database_id -a|-all
ccm dcm -s|-sh|-show -ts|-transfer_set transfer_set_spec
      -dbid|-database_id database_spec
```

`-a|-all`

Specifies to show the generate times for the specified transfer set to all databases.

`-database_id database_spec`

Specifies the database definition to use to show the last DCM generate times for a specified database definition and transfer set. You can set the *database\_spec* to a single database definition. See [Database specification](#) for details.

*transfer\_set\_spec*

Specifies the transfer set to use. You can set *transfer\_set\_spec* to a single transfer set. For details, see [Transfer set specification](#).

## Related topics

- [Modifying a transfer set](#)
- [Showing transfer set information](#)
- [Showing transfer set members](#)
- [Showing transfer sets](#)

---

### ***Showing the current DCM database ID***

This subcommand shows the DCM database identifier for the current database.

```
ccm dcm -s|-sh|-show -dbid|-database_id
```

### **Related topics**

- [Showing database IDs](#)
- [Transferring generated packages](#)

---

### ***Showing receive lock***

A receive lock ensures that only one DCM receive operation is performed in a database. This subcommand shows whether a database has a receive lock set. If so, the subcommand shows information about the currently running receive with the lock set.

```
ccm dcm -show -receive_lock
```

### **Related topics**

- [Receiving packages](#)
- [Showing transfer sets](#)



---

### **Showing transfer set information**

This subcommand shows information about the specified transfer sets.

```
ccm dcm -s|-sh|-show -ts|-transfer_set transfer_set_spec...
```

```
transfer_set_spec...
```

Specifies the transfer set(s) you want to show. You can set the *transfer\_set\_spec* to a single transfer set. For details, see [Transfer set specification](#).

### **Related topics**

- [Adding objects to a transfer set](#)
- [Creating a transfer set](#)
- [Deleting a transfer set](#)
- [Modifying a transfer set](#)
- [Recomputing the members for a transfer set](#)
- [Removing objects from a transfer set](#)
- [Showing transfer set information](#)
- [Showing transfer set members](#)
- [Showing transfer sets](#)

---

## Showing transfer set members

This subcommand shows the direct members and, optionally, the indirect members of the specified transfer sets.

```
ccm dcm -s|-sh|-show -ts|-transfer_set -members (direct | all)
        [-f|-format format] [-nf|-noformat] ([-ch|-column_header] |
        [-nch|-nocolumn_header]) [-sep|-separator separator]
        ([-sby|-sortby sortspec] | [-ns|-nosort|-no_sort])
        [-gby|-groupby groupformat] [-u|-unnumbered] transfer_set_spec...
```

[-ch|-column header](#)

[-f|-format \*format\*](#)

[-gby|-groupby \*groupformat\*](#)

`-members (direct|all)`

Specifies what kind of members to show. If you specify `-members direct`, only direct members are displayed. If you specify `all`, both direct and indirect members are displayed. To learn more about direct and indirect members, see "Add objects to a transfer set" in the [Rational Synergy Distributed](#) book.

[-nch|-nocolumn header](#)

[-nf|-noformat](#)

[-ns|-no\\_sort](#)

[-sep|-separator \*separator\*](#)

[-sby|-sortby \*sortspec\*](#)

[-u|-unnumbered](#)

## Related topics

- [Adding objects to a transfer set](#)
- [Creating a transfer set](#)
- [Deleting a transfer set](#)
- [Modifying a transfer set](#)

- 
- [Recomputing the members for a transfer set](#)
  - [Removing objects from a transfer set](#)
  - [Showing transfer set information](#)
  - [Showing transfer set members](#)
  - [Showing transfer sets](#)

---

## Showing transfer sets

This subcommand shows the names of all defined transfer sets.

```
ccm dcm -s|-sh|-show -ts|-transfer_set -a|-all [-f|-format format]  
        [-nf|-noformat] ([-ch|-column_header] | [-nch|-nocolumn_header])  
        [-sep|-separator separator] ([-sby|-sortby sortspec] |  
        [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat] [-u|-unnumbered]
```

[-a|-all](#)

Specifies to show all the names of all defined transfer sets for a specified database.

[-ch|-column\\_header](#)

[-f|-format \*format\*](#)

[-gby|-groupby \*groupformat\*](#)

[-nch|-nocolumn\\_header](#)

[-nf|-noformat](#)

[-ns|-no\\_sort](#)

[-sep|-separator \*separator\*](#)

[-sby|-sortby \*sortspec\*](#)

[-u|-unnumbered](#)

## Related topics

- [Adding objects to a transfer set](#)
- [Creating a transfer set](#)
- [Deleting a transfer set](#)
- [Modifying a transfer set](#)
- [Recomputing the members for a transfer set](#)
- [Removing objects from a transfer set](#)

- 
- [Showing transfer set information](#)
  - [Showing transfer set members](#)

---

## ***Transferring generated packages***

This subcommand transfers the packages generated from this database to their destination databases. You can specify the packages to be transferred by the destination database identifier or transfer set name.

This subcommand can be performed by a DCM manager or by an administrator.

```
ccm dcm -trn|-transfer [-dbid|-database_id database_spec]  
                ([-ts|-transfer_set transfer_set_spec] | [-a|-all])
```

`-a|-all`

Specifies to transfer packages generated using all transfer sets to the specified destination database.

`-database_id database_spec`

Specifies to show the last generate times for the specified database. You can set the *database\_spec* to a single database definition. See [Database specification](#) for details.

`-ts|-transfer_set transfer_set_spec...`

Specifies to transfer only packages generated from the specified transfer set. You can set the *transfer\_set\_spec* to a single transfer set. For details, see [Transfer set specification](#).

## **Related topics**

- [Generating a transfer package](#)
- [Receiving packages](#)
- [Showing DCM properties](#)
- [Showing last generate times](#)

---

## Description and uses

The `dcm` command generates a transfer package, sends a transfer package to a destination database, receives a transfer package, and adds objects to a transfer set. The `dcm` command options perform one or more of these operations.

You must be working as the DCM manager to use the `-add`, `-create`, `-gen`, `-modify`, `-delete`, and `-remove` options. You must be working in the `ccm_admin` role to use the `-rec`, `-init`, `-change`, `-modify`, and `-settings` options.

## delete command

See [Description and uses](#) for details. The `delete` command supports the [Deleting objects from the database](#) subcommand.



---

## Deleting objects from the database

The `delete` command deletes a specific version of a file or directory or project from a directory and from the database. Additionally, you can delete a project hierarchy from the command line or from the GUI.

```
ccm del|delete -p|-project ([-scope (project_only |
    project_and_non-project_members | project_and_subproject_hierarchy |
    entire_project_hierarchy)] | [-r|-recurse [-h|-hierarchy]])
    project_spec...
ccm del|delete ([-scope (directory_only |
    directory_and_non-project_members | entire_directory_hierarchy)] |
    [-r|-recurse [-h|-hierarchy]]) [-repl|-replace] [-t|-task task_spec]
    object_spec...
```

`-h|-hierarchy`

Causes the operation to delete the entire project hierarchy. This must be used with the `-recurse` option.

*object\_spec*

Specifies the object to delete.

`-p|-project`

Specifies the project form of the command.

*project\_spec*

Specifies the project to delete. See [Project specification](#) for details.

`-r|-recurse`

Specifies whether the delete operation recurses into directories or subprojects. When the object is a project, the recursive subprojects are also deleted. When the object is a directory, the recursive children of the directory are also deleted. For any other type of object, this option has no effect.

When using this option to hierarchically delete objects, the following apply:

- For a project object, `-recurse` is equivalent to specifying `-scope project_and_non-project_members`. It deletes the project and its members excluding subprojects.
- For a project object, `-recurse -hierarchy` is equivalent to specifying `-scope entire_project_hierarchy`. It deletes the projects and its recursive members including subprojects.

- 
- For a directory object, `-recurse` is equivalent to specifying `-scope directory_and_non-project_members`. It deletes the directory and its recursive children excluding subprojects.
  - For a directory object, `-recurse -hierarchy` is equivalent to specifying `-scope entire_directory_hierarchy`. It deletes the directory and its recursive children including subprojects.
  - For any other type of object, the option has no effect.

`-repl|-replace`

Deletes an object and replaces it with its predecessor.

`-scope (project_only | project_and_non-project_members | project_and_subproject_hierarchy | entire_project_hierarchy)`

Specifies the scope of the project deletion. The `project_only` scope means that only the project and its root directory are deleted. The `project_and_non-project_members` scope means that the project and any members except subprojects are deleted. The `project_and_subproject_hierarchy` scope means the entire project hierarchy including all subprojects are deleted.

`-scope (directory_only | directory_and_non-project_members | entire_directory_hierarchy)`

Specifies the scope for the deletion of any directory objects. The `directory_only` scope means that only the directory itself is deleted. The `directory_and_non-project_members` scope means that the directory and any children under the directory except subprojects are deleted. The `entire_directory_hierarchy` scope means the directory and all its recursive children including subprojects are deleted.

`-t|-task task_spec`

When you delete an object whose parent directory is read-only, a new version of the directory is checked out automatically. This option associates the newly checked-out directory with a task if the object was deleted from a read-only directory. If the current task is set and you do not specify a different task, the newly checked-out directory is associated with the current task automatically. See [Task specification](#) for details.

## Examples

- Delete the `sort.c` file and replace it with the previous version (actual output may differ from that shown below).

```
ccm delete sort.c
Member sort.c-1 deleted from project ico_proj-1
```

- Delete a file `sort.c`.

---

```
ccm delete sort.c-1:csrc:J#1
```

- Delete a project.

```
ccm delete -p Project_delete-1:project:M#1
```

- Delete a project with hierarchy and recursive.

```
ccm delete -p Project_Top-int:project:W#1 -recurse -h
```

## Related topics

- [create command](#)
- [unuse command](#)
- [use command](#)

---

## Description and uses

An object version can be deleted if it is not a member of a project or if it is only a member of the current project and has no successors.

**Note** When you delete an object from a non-writable directory, a new directory version is checked out automatically.

If you are in a shared project and your current directory is non-writable, the directory is checked out and associated automatically with the current (or specified) task and is checked in to the *integrate* state. You can disable the automatic check-in feature by setting `shared_project_directory_checkin` to `FALSE` in your initialization file. (See [shared project directory checkin](#).)

**Caution** The delete operation is permanent.

---

## delimiter command

See [Description and uses](#) for details. The `delimiter` command supports the [Showing the current version delimiter](#) subcommand.

---

### ***Showing the current version delimiter***

The delimiter is the character that separates the project or object name and version values. Use it to separate the project name from the version when copying a project.

```
ccm delim|delimiter
```

### **Examples**

- Show the current version delimiter.

```
ccm delimiter
```

### **Related topics**

- [Copying a project](#)
- [Modifying work area properties](#)
- [Showing work area properties](#)

---

## Description and uses

The `delimiter` command shows the value of the delimiter character.



## dir command

See [Description and uses](#) for details. The `dir` command supports the [Listing files](#) subcommand.

---

## Listing files

The `dir` command lists the contents of a project or a directory object version in a work area. By default, the output consists of a list of objects and their associated projections in the file system sorted in case-insensitive order of name.

```
ccm dir -p|-project [-m] ([-w] | [-f|-format format]) [-s] [-nf|-noformat]
      ([-ch|-column_header] | [-nch|-nocolumn_header])
      [-sep|-separator separator] ([-sby|-sortby sortspec] |
      [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat] project_spec...
ccm dir [-m] ([-w] | [-f|-format format]) [-s] [-nf|-noformat]
      ([-ch|-column_header] | [-nch|-nocolumn_header])
      [-sep|-separator separator] ([-sby|-sortby sortspec] |
      [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat]
      [path_or_file_spec...]
```

`-ch|-column_header`

Specifies to use a column header in the output format. See [-ch|-column\\_headers](#) for details.

`-f|-format format`

Specifies the command output format. See [-f|-format](#) for details.

A keyword can be built-in (`%fullname`, `%displayname`, `%objectname`) or the name of any existing attribute such as `%modify_time` or `%status`.

See [Built-In keywords](#) for a list of keywords.

`-gby|-groupby groupformat`

Specifies how to group the command output. See [-gby|-groupby](#) for details.

`-m`

Shows both controlled and uncontrolled files and directories. If a user-defined format is not specified with the `-f|-format` option, the default format (short or long form) includes a column indicating the synchronization status for files as follows:

- Local copy (LC) - denotes files that are in the project, but have a local copy rather than a symbolic link in the work area. If files are displayed with this mark and your work area is link-based, perform a *reconcile* operation. For more information, see [reconcile command](#).
- Not synchronized (NS) - denotes files that are in the project, but not in the work area. This situation occurs when you add files to the project, but your work area is not visible, or when the link or local copy of a file is deleted. If most of the files in

---

your work area are displayed with this mark, perform a reconcile operation. For more information, see [reconcile command](#).

- Uncontrolled (UC) - denotes files that are in the work area, but not in the project. To view uncontrolled files marked with UC, you must use the `-m` option with the `-l` option. In user-defined formats, use the `%Sync` keyword to show the synchronization status.

`-nch|-nocolumn_header`

Specifies not to use a column header in the output format. See [-nch|-nocolumn\\_headers](#) for details.

`-nf|-noformat`

Specifies not to use column alignment. See [-nf|-noformat](#) for details.

`-ns|-nosort|-no_sort`

Specifies that the command's output is not sorted. See [-ns|-nosort](#) for details.

*path\_or\_file\_spec*

Specifies the path list. You can set the *path\_or\_file\_spec* to a project, directory, or file defined in the database. This can also be an empty directory entry. If omitted, the current working directory is listed. See [File specification](#) for details.

`-p|-project`

Specifies that a project is listed.

*project\_spec*

Specifies the project to list. See [Project specification](#) for details.

`-s`

Displays subdirectory members recursively. The command does not recurse into subprojects.

`-sby|-sortby sortspec`

Specifies how to sort the command output. See [-sby|-sortby](#) for details.

---

`-sep|-separator separator`

Used only with the `-f|-format` option. Specifies a different separator character. See [-sep|-separator](#) for details.

`-u|-unnumbered`

Suppresses automatic numbering of the command's output (that is, the output is unnumbered). See [-u|-unnumbered](#) for details.

`-w`

Specifies to use the default short-form. This shows the displayname of each object.

### Example

- List the files that are not controlled by Rational Synergy.

```
ccm dir -m
(UC) symlink _ccmwaid.inf
working john 6/20/08 4:05 PM ascii 1 a.txt-one 10
working john 6/20/08 4:06 PM ascii 1 b.txt-one 10
```

- List the current directory in the long format. (Files preceded by LC are local copy files.)

```
ccm dir
working john 6/20/08 4:05 PM ascii 1 a.txt-one 10
working john 6/20/08 4:06 PM ascii 1 b.txt-one 10
```

- In the current directory, list the file name and version for all objects.

```
ccm dir -w
ext_incl-1
incl-1
src-1
```

- In the current directory, show all members, including subdirectories.

```
ccm dir /s
integrate joe Jun 19 2008 dir J#1 include,2 J#5565
(LC) integrate bob Jan 26 15:41 makefile J15 Makefile.pc,#7 J#6103
      released joe Jan 16 2006 dir J#12 src,1 J#120
include:
(LC) integrate pat Jan 26 15:42 makefile J#1 make_include.pc,13 J#6103
src:
(LC) integrate max Mar 27 2008 java J#1 Main.c,6 J#5339
```

- In the current directory, show the absolute paths for all objects.

```
ccm dir /f "%displayname %type %path"
```

---

a.txt-one ascii C:\ccm\_wa\turn\_3349\SubPrj-1\SubPrj\a.txt  
b.txt-one ascii C:\ccm\_wa\turn\_3349\SubPrj-1\SubPrj\b.txt

---

## Description and uses

The `dir` command displays two categories of files: objects under Rational Synergy control and files that exist in the file system only. By default, the command only shows controlled objects. Use the `-m` option to display uncontrolled objects as well as controlled objects.

## edit command

See [Description and uses](#) for details. The `edit` command supports the [Editing a file](#) subcommand.

---

## **Editing a file**

This subcommand edits the specified file. The default editor is used to edit the file.

```
ccm edit file_spec...
```

*file\_spec*

Specifies the file to edit. See [File specification](#) for details.

## **Example**

- Edit version 8 of the `log.c` file. To edit an object, it must be writable by you.

```
ccm edit log.c-8
```

## **Caveats**

On Windows, modifiable files can be edited only from within a project with a visible work area.

On UNIX, only files that are modifiable by the current user can be edited.

## **Related topics**

- [view command](#)
- [reconcile command](#)
- [cli.text editor](#)
- [cli.text viewer](#)



---

## Description and uses

The editor is started on the work area location if the file to be edited is specified in a form that provides a context project, such as a [Work area reference form](#) or a [Project reference form](#), and the corresponding work area location is visible to the client. If a project context is not available or the corresponding work area is not visible, the editor is launched with a temporary read-only copy of the file from the database.

**Note** When you edit a file in a copy-based work area, the corresponding database object is not automatically updated with your changes. For best results, reconcile your work area regularly when you are editing and saving a file.

## finduse command

See [Description and uses](#) for details. The `finduse` command supports the following subcommands:

- [Finding where a baseline is used](#)
- [Finding where a change request is used](#)
- [Finding where a folder is used](#)
- [Finding where a folder template is used](#)
- [Finding where a project is used](#)
- [Finding use for a task](#)
- [Finding where an object is used](#)
- [Finding where objects found by a query are used](#)

---

## ***Finding where a baseline is used***

This subcommand finds which project groupings and process rules use a baseline.

**Note** See [Scopes](#) for a definition of the available scopes for this subcommand.

```
ccm finduse -baseline
  [-working_proj|-working_projs|-working_project|-working_projects]
  [-shared_proj|-shared_projs|-shared_project|-shared_projects]
  [-prep_proj|-prep_projs|-prep_project|-prep_projects]
  [-released_proj|-released_projs|-released_project|-released_projects]
  [-all_proj|-all_projs|-all_project|-all_projects]
  [-personal_fold|-personal_folds|-personal_folder|-personal_folders]
  [-shared_fold|-shared_folds|-shared_folder|-shared_folders]
  [-prep_fold|-prep_folds|-prep_folder|-prep_folders]
  [-non_write_fold|-non_write_folds|-non_write_folder|
  -non_write_folders] [-all_fold|-all_folds|-all_folder|-all_folders]
  [-all_baseline|-all_baselines]
  [-wpg|-working_project_grouping|-working_project_groupings]
  [-mpg|-my_project_grouping|-my_project_groupings]
  [-ppg|-prep_project_grouping|-prep_project_groupings]
  [-spg|-shared_project_grouping|-shared_project_groupings]
  [-apg|-all_project_grouping|-all_project_groupings]
  [-all_process_rule|-all_process_rules] baseline_spec...
```

*baseline\_spec*

Specifies the baseline uses to be shown. See [Baseline specification](#) for details.

### ***Example***

- Find project grouping that uses the baseline **A#Base\_One**.

```
ccm finduse -baseline A#Base_One
```

### **Related topics**

- [Finding where a project is used](#)
- [Finding use for a task](#)
- [Finding where an object is used](#)
- [Finding where objects found by a query are used](#)

---

## ***Finding where a change request is used***

This subcommand finds which baselines or project groupings use a change request.

**Note** See [Scopes](#) for a definition of the available scopes for this subcommand.

```
ccm finduse -cr|change_request
  [-working_proj|-working_projs|-working_project|-working_projects]
  [-shared_proj|-shared_projs|-shared_project|-shared_projects]
  [-prep_proj|-prep_projs|-prep_project|-prep_projects]
  [-released_proj|-released_projs|-released_project|-released_projects]
  [-all_proj|-all_projs|-all_project|-all_projects]
  [-personal_fold|-personal_folds|-personal_folder|-personal_folders]
  [-shared_fold|-shared_folds|-shared_folder|-shared_folders]
  [-prep_fold|-prep_folds|-prep_folder|-prep_folders]
  [-non_write_fold|-non_write_folds|-non_write_folder|
  -non_write_folders] [-all_fold|-all_folds|-all_folder|-all_folders]
  [-all_baseline|-all_baselines]
  [-wpg|-working_project_grouping|-working_project_groupings]
  [-mpg|-my_project_grouping|-my_project_groupings]
  [-ppg|-prep_project_grouping|-prep_project_groupings]
  [-spg|-shared_project_grouping|-shared_project_groupings]
  [-apg|-all_project_grouping|-all_project_groupings]
  [-all_process_rule|-all_process_rules] change_request_spec...
```

*change\_request\_spec*

Specifies the change requests specified by *change\_request\_spec*. See [Change request specification](#) for details.

### ***Example***

- Find all my project groupings that use change request P#1265.

```
ccm finduse -my_project_grouping -cr P#1265
```

---

## Finding where a folder is used

This subcommand finds which projects, project groupings, or process rules use a folder.

**Note** See [Scopes](#) for a definition of the available scopes for this subcommand.

```
ccm finduse -folder
  [-working_proj|-working_projs|-working_project|-working_projects]
  [-shared_proj|-shared_projs|-shared_project|-shared_projects]
  [-prep_proj|-prep_projs|-prep_project|-prep_projects]
  [-released_proj|-released_projs|-released_project|-released_projects]
  [-all_proj|-all_projs|-all_project|-all_projects]
  [-personal_fold|-personal_folds|-personal_folder|-personal_folders]
  [-shared_fold|-shared_folds|-shared_folder|-shared_folders]
  [-prep_fold|-prep_folds|-prep_folder|-prep_folders]
  [-non_write_fold|-non_write_folds|-non_write_folder|
  -non_write_folders] [-all_fold|-all_folds|-all_folder|-all_folders]
  [-all_baseline|-all_baselines]
  [-wpg|-working_project_grouping|-working_project_groupings]
  [-mpg|-my_project_grouping|-my_project_groupings]
  [-ppg|-prep_project_grouping|-prep_project_groupings]
  [-spg|-shared_project_grouping|-shared_project_groupings]
  [-apg|-all_project_grouping|-all_project_groupings]
  [-all_process_rule|-all_process_rules] folder_spec...
```

*folder\_spec*

Finds all objects that include *folder\_spec*. See [Folder specification](#) for details.

## Example

- Find all projects that use folder 7.

```
ccm finduse -folder 7
Folder EAP#7: bill's Completed Tasks for Release 1.2
draw_proj-bill
```

## Related topics

- [Comparing folders](#)
- [Listing folders](#)
- [Showing a folder property](#)
- [Showing folder information](#)

---

## Finding where a folder template is used

This subcommand finds which process rules use a folder template.

**Note** See [Scopes](#) for a definition of the available scopes for this subcommand.

```
ccm finduse -ft|-folder_temp|-folder_template
  [-working_proj|-working_projs|-working_project|-working_projects]
  [-shared_proj|-shared_projs|-shared_project|-shared_projects]
  [-prep_proj|-prep_projs|-prep_project|-prep_projects]
  [-released_proj|-released_projs|-released_project|-released_projects]
  [-all_proj|-all_projs|-all_project|-all_projects]
  [-personal_fold|-personal_folds|-personal_folder|-personal_folders]
  [-shared_fold|-shared_folds|-shared_folder|-shared_folders]
  [-prep_fold|-prep_folds|-prep_folder|-prep_folders]
  [-non_write_fold|-non_write_folds|-non_write_folder|
  -non_write_folders] [-all_fold|-all_folds|-all_folder|-all_folders]
  [-all_baseline|-all_baselines]
  [-wpg|-working_project_grouping|-working_project_groupings]
  [-mpg|-my_project_grouping|-my_project_groupings]
  [-ppg|-prep_project_grouping|-prep_project_groupings]
  [-spg|-shared_project_grouping|-shared_project_groupings]
  [-apg|-all_project_grouping|-all_project_groupings]
  [-all_process_rule|-all_process_rules] folder_template_spec...
```

*folder\_template\_spec*

Finds all objects that include *folder\_template\_spec*. See [Folder template specification](#) for details.

### Example

- Find all process rules that use the All completed tasks for release %release folder template.

```
ccm finduse -all_process_rules -ft "All completed tasks for release
%release"
```

---

## ***Finding where a project is used***

This subcommand finds which projects, project groupings or baselines use a project.

**Note** See [Scopes](#) for a definition of the available scopes for this subcommand.

```
ccm finduse -p|-project
  [-working_proj|-working_projs|-working_project|-working_projects]
  [-shared_proj|-shared_projs|-shared_project|-shared_projects]
  [-prep_proj|-prep_projs|-prep_project|-prep_projects]
  [-released_proj|-released_projs|-released_project|-released_projects]
  [-all_proj|-all_projs|-all_project|-all_projects]
  [-personal_fold|-personal_folds|-personal_folder|-personal_folders]
  [-shared_fold|-shared_folds|-shared_folder|-shared_folders]
  [-prep_fold|-prep_folds|-prep_folder|-prep_folders]
  [-non_write_fold|-non_write_folds|-non_write_folder|
  -non_write_folders] [-all_fold|-all_folds|-all_folder|-all_folders]
  [-all_baseline|-all_baselines]
  [-wpg|-working_project_grouping|-working_project_groupings]
  [-mpg|-my_project_grouping|-my_project_groupings]
  [-ppg|-prep_project_grouping|-prep_project_groupings]
  [-spg|-shared_project_grouping|-shared_project_groupings]
  [-apg|-all_project_grouping|-all_project_groupings] project_spec...
```

*project\_spec*

Finds all projects that include *project\_spec*. See [Project specification](#) for details.

### ***Example***

- Find all objects that use the **project\_sub2-win** project.

```
ccm finduse -project project_sub2-win -all_projs -apg -all_folders -
all_process_rules -all_baselines
```

### **Related topics**

- [Finding where a baseline is used](#)
- [Finding use for a task](#)
- [Finding where an object is used](#)
- [Finding where objects found by a query are used](#)

---

## Finding use for a task

This subcommand finds which projects, project groupings, folders or baselines use a task.

**Note** See [Scopes](#) for a definition of the available scopes for this subcommand.

```
ccm finduse -task
  [-working_proj|-working_projs|-working_project|-working_projects]
  [-shared_proj|-shared_projs|-shared_project|-shared_projects]
  [-prep_proj|-prep_projs|-prep_project|-prep_projects]
  [-released_proj|-released_projs|-released_project|-released_projects]
  [-all_proj|-all_projs|-all_project|-all_projects]
  [-personal_fold|-personal_folds|-personal_folder|-personal_folders]
  [-shared_fold|-shared_folds|-shared_folder|-shared_folders]
  [-prep_fold|-prep_folds|-prep_folder|-prep_folders]
  [-non_write_fold|-non_write_folds|-non_write_folder|
  -non_write_folders] [-all_fold|-all_folds|-all_folder|-all_folders]
  [-all_baseline|-all_baselines]
  [-wpg|-working_project_grouping|-working_project_groupings]
  [-mpg|-my_project_grouping|-my_project_groupings]
  [-ppg|-prep_project_grouping|-prep_project_groupings]
  [-spg|-shared_project_grouping|-shared_project_groupings]
  [-apg|-all_project_grouping|-all_project_groupings] task_spec...
```

*task\_spec*

Finds all tasks that include *task\_spec*. See [Task specification](#) for details.

## Example

- Find all objects that use the **GA#1** task.

```
ccm finduse -task GA#1 -all_projs -all_baselines -all_folders -
all_process_rules -apg
```

## Related topics

- [Finding where a baseline is used](#)
- [Finding where a project is used](#)
- [Finding where an object is used](#)
- [Finding where objects found by a query are used](#)



---

## Finding where an object is used

This subcommand finds which projects, project groupings, process rules, folders or baselines use an object.

**Note** See [Scopes](#) for a definition of the available scopes for this subcommand.

```
ccm finduse
  [-working_proj|-working_projs|-working_project|-working_projects]
  [-shared_proj|-shared_projs|-shared_project|-shared_projects]
  [-prep_proj|-prep_projs|-prep_project|-prep_projects]
  [-released_proj|-released_projs|-released_project|-released_projects]
  [-all_proj|-all_projs|-all_project|-all_projects]
  [-personal_fold|-personal_folds|-personal_folder|-personal_folders]
  [-shared_fold|-shared_folds|-shared_folder|-shared_folders]
  [-prep_fold|-prep_folds|-prep_folder|-prep_folders]
  [-non_write_fold|-non_write_folds|-non_write_folder|
  -non_write_folders] [-all_fold|-all_folds|-all_folder|-all_folders]
  [-all_baseline|-all_baselines]
  [-wpg|-working_project_grouping|-working_project_groupings]
  [-mpg|-my_project_grouping|-my_project_groupings]
  [-ppg|-prep_project_grouping|-prep_project_groupings]
  [-spg|-shared_project_grouping|-shared_project_groupings]
  [-apg|-all_project_grouping|-all_project_groupings]
  [-all_process_rule|-all_process_rules] object_spec...
```

*object\_spec*

Finds all objects that include *object\_spec*. See [Object specification](#) for details.

## Examples

- Find all uses of the object version named `display.c` in projects.  

```
ccm finduse -name display.c
```
- Find all uses in projects of the version of `draw.c` being used in the current directory.  

```
ccm finduse draw.c
```
- Find all personal folders containing object `draw.c-2:csrc:EAP#1`.  

```
ccm finduse -personal_folder draw.c-2:csrc:EAP#1
```

## Related topics

- [Finding where a baseline is used](#)
- [Finding where a project is used](#)
- [Finding use for a task](#)

- 
- [Finding where objects found by a query are used](#)

---

## ***Finding where objects found by a query are used***

This subcommand queries for objects matching a query expression or criteria, and then for each, finds which projects, project groupings, process rules, folders, or baselines use them. You can modify the query expression in either or both of the following ways:

- a query expression
- one or more query-related options that generate a query clause

Each query-related option generates a query clause. For example, the `-name name` option generates a query clause in the form `(name='name')`.

When the same query option is repeated, the query clauses are combined with an `or`. For example, `-name file1 -name file2` generates a query clause `(name='file1' or name='file2')`.

Query clauses from different options or from the query expression are combined with an `and`. For example, `-name file1 -owner joe` generates a query clause `(name='file1') and (owner='joe')`.

**Note** See [Scopes](#) for a definition of the available scopes for this subcommand.

```
ccm finduse [-q|-query query_expression] [(-n|-name name)...]
  [(-o|-owner owner)...] [(-st|-state state)...] [(-t|-type type)...]
  [(-v|-version version)...] [(-i|-instance instance)...]
  [(-release release_spec)...]
  [-working_proj|-working_projs|-working_project|-working_projects]
  [-shared_proj|-shared_projs|-shared_project|-shared_projects]
  [-prep_proj|-prep_projs|-prep_project|-prep_projects]
  [-released_proj|-released_projs|-released_project|-released_projects]
  [-all_proj|-all_projs|-all_project|-all_projects]
  [-personal_fold|-personal_folds|-personal_folder|-personal_folders]
  [-shared_fold|-shared_folds|-shared_folder|-shared_folders]
  [-prep_fold|-prep_folds|-prep_folder|-prep_folders]
  [-non_write_fold|-non_write_folds|-non_write_folder|
  -non_write_folders] [-all_fold|-all_folds|-all_folder|-all_folders]
  [-all_baseline|-all_baselines]
  [-wpg|-working_project_grouping|-working_project_groupings]
  [-mpg|-my_project_grouping|-my_project_groupings]
  [-ppg|-prep_project_grouping|-prep_project_groupings]
  [-spg|-shared_project_grouping|-shared_project_groupings]
  [-apg|-all_project_grouping|-all_project_groupings]
  [-all_process_rule|-all_process_rules]
```

`-i|-instance instance`

Includes a query clause of the form `subsystem='instance'` to find objects with the specified instance.

---

`-n|-name name`

Includes a query clause of the form `name='name'` to find objects with the specified name.

`-o|-owner owner`

Includes a query clause of the form `owner='owner'` to find objects with the specified owner.

`-release release_spec`

Includes a query clause of the form `release='releasename'` to find objects with the specified release. You can set the `release_spec` to one or more release definitions or releases. See [Release specification](#) for details.

`-st|-state state`

Includes a query clause of the form `status='state'` to find objects of the specified status.

`-t|-type type`

Includes a query clause of the form `type='type'` to find objects of the specified type.

`-v|-version version`

Includes a query clause of the form `version='version'` to find objects of the specified version.

### **Example**

- Find the default scopes for the objects that are found by using a query.

```
ccm finduse -query "(cvtype='ascii' or (cvtype='task' and
status='completed'))"
```

### **Related topics**

- [Finding where a baseline is used](#)
- [Finding where a project is used](#)
- [Finding use for a task](#)
- [Finding where an object is used](#)

---

## Description and uses

The `finduse` command searches the database for uses of a specified object and returns a list of where the specified object is used.

Each `finduse` command supports options that define a scope for the returned objects. These scope-related options define what type of searches to perform. You can perform the following kind of searches:

- Project related  
The results include the projects that match the scope and use of the specified object. Each result is shown in a [Project reference form](#).
- Project grouping related  
The results include the project groupings that match the scope and use the specified object. Each result is shown as a [Project grouping specification](#).
- Process rule related  
The results include the process rules that match the scope and use the specified object. Each result is shown as a [Process rule specification](#).
- Folder template related  
The results include the folder templates that match the scope and use the specified object. Each result is shown as a [Folder template specification](#).
- Folder related  
The results include the folders that match the scope and use the specified object. Each result is shown as a [Project specification](#).
- Change request related  
The results include the CRs that match the scope and use the specified object. Each result is shown as a [Change request specification](#).
- Baseline related  
The results include the baselines that use the specified object. Each baseline is shown as a [Baseline specification](#).

The default scope used depends on the type of object you use with the `finduse` command.

- Change request  
If you do not specify a scope option, the default is all baselines.
- Task  
If you do not specify a scope option, the default is all projects.
- Project, file or directory  
If you do not specify a scope option, the default is all projects.

- 
- **Folder**  
If you do not specify a project or project grouping scope option, all projects are always included in the scope.
  - **Folder template**  
If you do not specify a scope, the default is all process rules.
  - **Baseline**  
If you do not specify a project grouping scope option, then all project groupings are included in the scope.

## Scopes

The scopes available specify the scope of the search and include the following.

### Baseline scope

- `-all_baseline|-all_baselines`  
Specifies to show all uses in baselines.

### Change request scopes

Specifies to show all uses in change requests.

### Folder scopes

- `-all_fold|-all_folds|-all_folder|-all_folders`  
Specifies to show all uses in folders in any state.
- `-personal_fold|-personal_folds|-personal_folder|-personal_folders`  
Specifies to show all uses in personal folders.
- `-shared_fold|-shared_folds|-shared_folder|-shared_folders`  
Specifies to show all uses in shared folders.
- `-prep_fold|-prep_folds|-prep_folder|-prep_folders`  
Specifies to show all uses in prep (build management) folders.
- `-non_write_fold|-non_write_folds|-non_write_folder|-non_write_folders`  
Specifies to show all uses in non-writable folders.

### Folder template scopes

Specifies to show all uses in folder templates.

### Project scopes

- `-all_proj|-all_projs|-all_project|-all_projects`  
Specifies to show all uses in projects in any state.

- 
- `-shared_proj|-shared_projs|-shared_project|-shared_projects`  
Specifies to show all uses in shared projects.
  - `-prep_proj|-prep_projs|-prep_project|-prep_projects`  
Specifies to show all uses in prep (build management) projects.
  - `-working_proj|-working_projs|-working_project|-working_projects`  
Specifies that all uses in development projects should be shown.
  - `-released_proj|-released_projs|-released_project|-released_projects`  
Specifies to show all uses in released projects.

#### **Process rule scope**

- `-all_process_rules`  
Specifies to show all uses in process rules of a baseline, folder, or folder template object with a type of that is specified or found by using a query.

#### **Project grouping scopes**

- `-apg|-all_project_grouping|-all_project_groupings`  
Specifies to show all uses in project groupings for any state or owner).
- `-mpg|-my_project_grouping|-my_project_groupings`  
Specifies to show all uses in project groupings owned by you.
- `-ppg|-prep_project_grouping|-prep_project_groupings`  
Specifies to show all uses in prep (build management) project groupings.
- `-spg|-shared_project_grouping|-shared_project_groupings`  
Specifies to show all uses in shared project groupings.
- `-wpg|-working_project_grouping|-working_project_groupings`  
Specifies to show all uses in working project groupings.

## folder command

See [Description and uses](#) for details. The `folder` command supports the following subcommands:

- [Comparing folders](#)
- [Copying a folder](#)
- [Creating a folder](#)
- [Deleting a folder](#)
- [Finding where a folder is in use](#)
- [Listing folders](#)
- [Modifying a folder](#)
- [Showing a folder property](#)
- [Showing the associated tasks or objects for a folder](#)
- [Showing folder information](#)



---

## Comparing folders

This subcommand compares the contents of two folders.

You must specify `-union`, `-intersection`, or `-not_in`.

```
ccm folder -comp|-compare ([-un|-union] | [-int|-intersection] |
    [-not|-not_in]) [-f|-format format] [-nf|-noformat]
    ([-ch|-column_header] | [-nch|-nocolumn_header])
    [-sep|-separator separator] ([-sby|-sortby sortspec] |
    [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat]
    [-u|-unnumbered] folder_spec1 folder_spec2
```

`-ch|-column_header`

Specifies to use a column header in the output format. See [-ch|-column\\_headers](#) for details.

*folder\_spec1*

Specifies the first folder to be compared. See [Folder specification](#) for details.

*folder\_spec2*

Specifies the second folder to be compared. See [Folder specification](#) for details.

`-f|-format format`

Specifies the command output format. See [-f|-format](#) for details.

`-gby|-groupby groupformat`

Specifies how to group the command output. See [-gby|-groupby](#) for details.

`-int|-intersection`

Specifies that the folder comparison will show the tasks that are in both folders.

`-nch|-nocolumn_header`

Specifies not to use a column header in the output format. See [-ch|-column\\_headers](#) for details.

`-nf|-noformat`

Specifies not to use column alignment. See [-nf|-noformat](#) for details.

---

`-ns|-no_sort`

Specifies that the command's output will not be sorted. See [-ns|-nosort](#) for details.

`-not|-not_in`

Specifies that the folder comparison will show the tasks in *folder\_spec1* that are not in *folder\_spec2*.

`-sep|-separator separator`

Specifies a different separator character. See [-sep|-separator](#) for details.

`-sby|-sortby sortspec`

Specifies how to sort the command output. See [-sby|-sortby](#) for details.

`-un|-union`

Specifies the folder comparison to show the tasks that are in either of the two folders.

`-u|-unnumbered`

Suppresses automatic numbering of the command's output (that is, the output is unnumbered). See [-u|-unnumbered](#) for details.

## Examples

- Show the tasks that are in either folder **154** or folder **155**.

```
ccm folder -compare -union 154 155
1) Task 12: System error when time zone changes
2) Task 15: Correct spelling errors in output
3) Task 19: Rewrite messaging module
4) Task 26: Close box no longer active
5) Task 31: Wrong window receives message
6) Task 40: Auto-calculation gives incorrect result
7) Task 53: Download of images occurs too slowly
```

- Show the tasks that folders **154** and **155** have in common.

```
ccm folder -comp -int 154 155
1) Task 15: Correct spelli304
ng errors in output
2) Task 19: Rewrite messaging module
3) Task 26: Close box no longer active
4) Task 40: Auto-calculation gives incorrect result
```

- 
- Show the tasks that are in **folder 154**, but not in **folder 155**.

```
ccm folder -compare -not_in 154 155
```

```
1) Task 12: System error when time zone changes
```

```
2) Task 31: Wrong window receives message
```

## Related topics

- [Finding where a folder is in use](#)
- [Listing folders](#)
- [Showing a folder property](#)
- [Showing folder information](#)

---

## Copying a folder

This subcommand copies the contents of a folder to a new folder or an existing folder. A developer or build manager can copy a folder to a new folder. When copying a folder to an existing folder, the destination folder must be modifiable by you.

```
ccm folder -cp|-copy folder_spec -new new_folder_name [-q|-quiet]
ccm folder -cp|-copy folder_spec -e|-existing folder_spec [-append]
[-q|-quiet]
```

`-append`

When used with [-e|-existing folder\\_spec](#) option, specifies to append the tasks in the destination folder rather than replacing them with tasks from the source folder.

`-e|-existing folder_spec`

Specifies to copy the folder to the existing folder specified by *folder\_spec*. You can set the *folder\_spec* to a single folder. See [Folder specification](#) for details.

By default, the tasks in the source folder replace those in the destination folder. Use the `-append` option to append rather than replace tasks.

*folder\_spec*

Specifies the folder to copy from. See [Folder specification](#) for details.

`-new new_folder_name`

Specifies to copy the folder to a new folder with the specified *new\_folder\_name*. The *new\_folder\_name* cannot contain newline characters.

`-q|-quiet`

Specifies to show the display name of the updated or created folder. The display name shows a valid [Folder specification](#)

## Examples

- Copy folder **95** to a new folder named **Tasks Completed for Release 3.4 on September 15, 1997**.

```
ccm folder -copy 95 -new "Tasks Completed for Release 3.4 on September
15, 1997"
Folder '95: Tasks Completed for Release 3.4' copied to '158: Tasks
Completed for Release 3.4 on September 15, 1997'
```

- 
- Copy folder 95 to an existing folder, number 103.

```
ccm folder -cp 95 -existing 103
```

```
Folder '95: Tasks Completed for Release 3.4' copied to '103: Tested  
Tasks for Release 3.4"
```

- Copy folder folder 95 to an existing folder 103, appending the tasks.

```
ccm folder -copy 95 -append -existing 103
```

## Related topics

- [Finding where a folder is in use](#)
- [Listing folders](#)
- [Showing a folder property](#)
- [Showing folder information](#)

---

## Creating a folder

This subcommand creates a folder.

When you use an option more than once, the query expression relating to each usage is combined with an "or". For example, if you specify `-release 1.0 -release 2.0`, this contributes a query expression of `(release='1.0' or release='2.0')`.

Contributions from different options are combined with "and". For example, if you specify `-release 1.0 -platform windows`, this contributes a query expression of `(release='1.0')` and `(platform='windows')`.

```
ccm folder -cr|-create -n|-name folder_name ([-qu|-query] |
  [-mode ((man|manual) | (uq|use_query))])
  [-w|-writable (owner | (build_mgr|build_manager|buildmanager) |
  all | none)] [-q|-quiet] [-cus|-custom custom_query]
  [(-db|-dbid|-database_id database_spec)...]
  [(-plat|-platform platform)...] [(-purpose purpose)...]
  [(-rel|-release release_spec)...]
  [(-sub|-subsystem subsystem)...] [-ts|-scope|-task_scope
  (user_defined | (all_my_assigned|all_owners_assigned) |
  (all_my_assigned_or_completed|all_owners_assigned_or_completed) |
  (all_my_completed|all_owners_completed) |
  (all_my_tasks|all_owners_tasks) | all_completed | all_tasks)]
  (ct_projs|ct_projects|component_task_projects) |
  (ct_prods|ct_products|component_task_products) |
  (ct_projs_prods|ct_projects_products |
  component_task_projects_products))]
```

`-cus|-custom custom_query`

Specifies to include the specified custom query expression in the new folder query.

`-db|-dbid|-database_id database_spec...`

When used with the [-ts|-scope|-task\\_scope](#) option, specifies a database identifier that modifies the query generated from the task scope. See [Database specification](#) for further details.

`-mode ((man|manual) | (uq|use_query))`

Specifies whether to add tasks to the new folder manually or by using a query.

If you don't specify `-mode` or `-query`, the default mode depends on whether you specify query-related options. If you specify `-custom`, `-dbid`, `-platform`, `-release`, `-subsystem`, `-task_scope`, the default mode is query-based. If you don't specify any of these options, the default is to add tasks to the folder manually.

---

If you specify `-mode use_query` or `-query`, but don't specify `-custom`, `-dbid`, `-platform`, `-release`, `-subsystem`, or `-task_scope`, a default task query is used in the following way:

- If you have defined a [default task query](#), it is used.
- If you haven't defined a default query, the task scope **All my assigned and completed tasks** is used.

`-n|-name` *folder\_name*

Specifies the name of the new folder to be created. The *folder\_name* cannot contain newline characters.

`-plat|-platform` *platform*

Specifies that the task query for the folder will include a query for the specified platform.

`-purpose` *purpose*

Specifies to create the folder with a task query that includes a query for the specified purpose. See the `project_purpose` command [Description and uses](#) section for a detailed description of purposes.

This option typically applies to queries for component tasks that are specified with one of the following scopes: `component_task_projects`, `component_task_products`, or `component_task_projects_products`.

`-qu|-query`

Specifies that the new folder will be query-based. This is synonymous with using `-mode use_query`. See [-mode \(\(man|manual\) | \(uq|use\\_query\)\)](#) for details.

`-q|-quiet`

Specifies to show only the display name of the created folder. The display name shows a valid [Folder specification](#).

`-rel|-release` *release\_spec*

Specifies to create the folder with a task query that includes a query for the specified release. You can set the *release\_spec* to multiple releases. See [Release specification](#) for further details.

---

`-sub|-subsystem subsystem`

Specifies to create the folder by using a task query that includes a query expression for task subsystem.

`-ts|-scope|-task_scope`

Specifies to use a task query. The task query will include a query expression that depends on the specified scope. The query expression associated with the specified scope also depends on the `-database_id` option. You can use the following scopes:

- `user_defined`  
This scope is defined by the [default task query](#) option. If you specify `-database_id`, the query also includes a query expression for tasks modifiable in or completed in the specified database.
- `all_my_assigned|all_owners_assigned`  
This scope queries for all tasks assigned to you. If you specify `-database_id`, the query is for all tasks assigned to you that are modifiable in the specified database.
- `all_my_assigned_or_completed|all_owners_assigned_or_completed`  
This scope queries for all tasks assigned to you or completed by you. If you specify `-database_id`, the query is for all tasks assigned to you and modifiable in the specified database, or completed by you in the specified database.
- `all_my_completed|all_owners_completed`  
This scope queries for all tasks completed by you. If you specify `-database_id`, the query is for all tasks completed by you in the specified database.
- `all_my_tasks|all_owners_tasks`  
This scope queries for all tasks for which you are the task resolver. If you specify `-database_id`, the query is for all tasks for which you are the task resolver and that are modifiable in the specified database or were completed in the specified database.
- `all_completed`  
This scope queries for all completed tasks. If you specify `-database_id`, the query is for all tasks completed in the specified database.
- `all_tasks`  
This scope queries for all tasks. If you specify `-database_id`, the query is for all tasks that are modifiable in the specified database or that were completed in the specified database.
- `component_task_projects|component_task_products|  
component_task_projects_products`



---

This scope queries for component tasks for projects, products, or projects and products. If you specify `-database_id`, the query is for all component tasks that were created in the specified database. If you specify `-purpose`, the query is for component tasks with the specified purpose.

`-w|-writable (owner | (build_mgr|build_manager|buildmanager) | all | none)`

Specifies who can modify the new folder.

### Examples

- Create a new folder named **Tested Tasks for Release 3.5** that is writable by its owner, and suppress all output from the command except for the folder ID.

```
ccm folder -cr -n "Tested Tasks for Release 3.5" -w Owner -q 159
```

- Create a new folder named **My Tasks for Release 3.5** that uses a `task_spec` and a `release` value for a `query_spec`.

```
ccm folder -cr -name "My Tasks for Release 3.5" -ts all_my_tasks -rel 3.5
```

```
Created folder 160.
```

### Related topics

- [Finding where a folder is in use](#)
- [Listing folders](#)
- [Showing a folder property](#)
- [Showing folder information](#)

---

## ***Deleting a folder***

This subcommand deletes the specified folders. The folders must be modifiable by you.

```
ccm folder -d|-delete folder_spec...
```

*folder\_spec*

Specifies the folder to delete. See [Folder specification](#) for details.

### ***Example***

- Delete folders **109,110**, and **158**.

```
ccm folder -delete 109-110,158
```

```
Deleted folder '109: Tasks Completed for Release 2.1 on April 1, 1996'.
```

```
Removed 1 folder.
```

```
Deleted folder '110: Tasks Completed for Release 2.2 on June 1, 1996'.
```

```
Removed 1 folder.
```

```
Deleted folder '158: Tasks Completed for Release 3.4 on July 15, 1997'.
```

```
Removed 1 folder.
```

### **Related topics**

- [Finding where a folder is in use](#)
- [Listing folders](#)
- [Showing a folder property](#)
- [Showing folder information](#)

---

## ***Finding where a folder is in use***

This subcommand finds where the specified folder is in use in the current database.

```
ccm folder -fu|-finduse|-find_use
           [-f|-format format] [-nf|-noformat] ([-ch|-column_header]
           | [-nch|-nocolumn_header]) [-sep|-separator separator]
           ([-sby|-sortby sortspec] | [-ns|-nosort|-no_sort])
           [-gby|-groupby groupformat] [-u|-unnumbered]
           folder_spec...
```

`-ch|-column_header`

Specifies to use a column header in the output format. See [-ch|-column\\_headers](#) for details.

*folder\_spec*

Specifies the folders to search for when performing a finduse in projects.

See [Folder specification](#) for details.

`-f|-format format`

Specifies the command output format. See [-f|-format](#) for details.

`-gby|-groupby groupformat`

Specifies how to group the command output. See [-gby|-groupby](#) for details.

`-nch|-nocolumn_header`

Specifies not to use a column header in the output format. See [-ch|-column\\_headers](#) for details.

`-nf|-noformat`

Specifies not to use column alignment. See [-nf|-noformat](#) for details.

`-ns|-no_sort`

Specifies that the command's output will not be sorted. See [-ns|-nosort](#) for details.

`-sep|-separator separator`

Specifies a different separator character. See [-sep|-separator](#) for details.

---

`-u` | `-unnumbered`

Suppresses automatic numbering of the command's output (that is, the output is unnumbered). See [-u|-unnumbered](#) for details.

### **Examples**

- Find the projects that use folders 123 and 234.

```
ccm folder -finduse 123 234
```

### **Related topics**

- [Copying a folder](#)
- [Deleting a folder](#)
- [Listing folders](#)
- [Modifying a folder](#)
- [Showing a folder property](#)
- [Showing the associated tasks or objects for a folder](#)
- [Showing folder information](#)

---

## Listing folders

This subcommand lists the folders that satisfy the specified criteria. If you don't specify options or arguments, the command lists all folders; otherwise, the command lists the following:

- The `all_personal` scope lists folders that are writable by their owners.
- The `all_build_mgrs` scope lists folders that are writable by build managers.
- The `all_shared` scope lists folders that are writable by everyone.
- The `all_non_writable` scope lists folders that are read-only.

```
ccm folder -l|-list [-f|-format format] [-nf|-noformat] ([-ch|-column_header]
| [-nch|-nocolumn_header]) [-sep|-separator separator]
([-sby|-sortby sortspec] | [-ns|-nosort|-no_sort])
[-gby|-groupby groupformat] [-u|-unnumbered]
[(all_personal | all_build_mgrs | all_shared |
all_non_writable | all)]
```

`-ch|-column_header`

Specifies to use a column header in the output format. See [-ch|-column\\_headers](#) for details.

`-f|-format format`

Specifies the command output format. See [-f|-format](#) for details.

`-gby|-groupby groupformat`

Specifies how to group the command output. See [-gby|-groupby](#) for details.

`-nch|-nocolumn_header`

Specifies not to use a column header in the output format. See [-ch|-column\\_headers](#) for details.

`-nf|-noformat`

Specifies not to use column alignment. See [-nf|-noformat](#) for details.

`-ns|-no_sort`

Specifies that the command's output will not be sorted. See [-ns|-nosort](#) for details.

---

`-sep|-separator separator`

Specifies a different separator character. See [-sep|-separator](#) for details.

`-sby|-sortby sortspec`

Specifies how to sort the command output. See [-sby|-sortby](#) for details.

`-u|-unnumbered`

Suppresses automatic numbering of the command's output (that is, the output is unnumbered). See [-u|-unnumbered](#) for details.

### Examples

- List all of the build management folders in the current database.

```
ccm folder -list all_build_mgrs
1) Folder 42: All Completed Tasks for Release 2.1
2) Folder 95: Tasks Completed for Release 3.4
```

- List all of your personal folders.

```
ccm folder -list all_personal
1) Folder 111: bob's Insulated Development Folder
2) Folder 145: bob's Completed Tasks for Release 4.2
3) Folder 146: bob's Assigned Tasks
```

### Related topics

- [Comparing folders](#)
- [Finding where a folder is used](#)
- [Showing a folder property](#)
- [Showing folder information](#)

---

## Modifying a folder

This subcommand modifies the specified folders. The folders must be modifiable by you.

When you define a folder query, the `-custom`, `-platform`, `-release`, `-subsystem`, and `-task_scope` options contribute to the final generated task query. You can use the `-platform`, `-release` and `-subsystem` options multiple times. When you use an option more than once, the query expression relating to each usage is combined with an "or". For example, if you specify `-release 1.0 -release 2.0`, this contributes a query expression of `(release='1.0' or release='2.0')`. Contributions from different options are combined with "and". For example, if you specify `-release 1.0 -platform windows`, this contributes a query expression of `(release='1.0') and (platform='windows')`. The `-task_scope` option also results in a contribution to the task query based on the specified scope, and this is modified by any `-database_id` options specified. The final task query used combines all of these elements in a single query expression.

```
ccm folder -m|-modify [-n|-name folder_name]
    [-mode ((man|manual) | (uq|use_query))]
    [-w|-writable (owner | (build_mgr|build_manager|buildmanager) |
    all | none)] [-cus|-custom custom_query]
    [(-db|-dbid|-database_id database_spec)...]
    [(-plat|-platform platform)...] [(-purpose purpose)...]
    [(-rel|-release release_spec)...]
    [(-sub|-subsystem subsystem)...] [-ts|-scope|-task_scope
    (user_defined | (all_my_assigned|all_owners_assigned) |
    (all_my_assigned_or_completed|all_owners_assigned_or_completed) |
    (all_my_completed|all_owners_completed) |
    (all_my_tasks|all_owners_tasks) | all_completed | all_tasks)]
    (ct_projs|ct_projects|component_task_projects) |
    (ct_prods|ct_products|component_task_products) |
    (ct_projs_prods|ct_projects_products|
    component_task_projects_products))
    [(-at|-add_task|-add_tasks task_spec)...]
    [(-rt|-remove_task|-remove_tasks task_spec)...]
    [-up|-update] folder_spec...
```

`-at|-add_task|-add_tasks task_spec`

Adds the specified tasks to the specified folders. See [Task specification](#) for details.

`-cus|-custom custom_query`

Specifies to update the folder query to include the specified custom query expression.

---

`-db|-dbid|-database_id database_spec`

When used with the `-task_scope` option, specifies a database identifier that modifies the query generated from the task scope. See [-ts|-scope|-task\\_scope](#) and [Database specification](#) for further details.

`folder_spec`

Specifies the folder to modify. See [Folder specification](#) for details.

`-mode ((man|manual) | (uq|use_query))`

Specifies whether to modify folders to add tasks manually or by using a query.

If you modify a folder from manual to query-based, never defined a task query, and didn't specify any available options, then the folder is created as query-based with a [default task query](#) defined as follows:

- If you have defined a default task query, it is used.
- If you haven't defined a default query, the task scope **All my assigned and completed tasks** is used.

`-n|-name folder_name`

Specifies to rename the specified folders to the specified folder name. The `folder_name` cannot contain newline characters.

`-plat|-platform platform`

Specifies to update the folder query to use a query expression for the specified platform.

`-purpose purpose`

Specifies to create the folder with a task query that includes a query for the specified purpose. See the `project_purpose` command [Description and uses](#) section for a detailed description of purposes.

This option typically applies to queries for component tasks that are specified with one of the following scopes: `component_task_projects`, `component_task_products`, or `component_task_projects_products`.

`-rel|-release release_spec`

Specifies to update the folder query to use a query expression for the specified release. You can set the `release_spec` to multiple releases. See [Release specification](#) for details.



---

`-rt|-remove_task|-remove_tasks task_spec`

Removes the specified tasks from the specified folders. See [Task specification](#) for details.

`-sub|-subsystem subsystem`

Specifies to update the folder query to use a query expression for task subsystem.

`-ts|-scope|-task_scope`

Specifies to use a task query. The task query will include a query expression that depends on the specified scope. The query expression associated with the specified scope also depends on the `-database_id` option. You can use the following scopes:

- `user_defined`  
This scope is defined by the [default task query](#) option. If you specify `-database_id`, the query also includes a query expression for tasks modifiable in or completed in the specified database.
- `all_my_assigned|all_owners_assigned`  
This scope queries for all tasks assigned to you. If you specify `-database_id`, the query is for all tasks assigned to you that are modifiable in the specified database.
- `all_my_assigned_or_completed|all_owners_assigned_or_completed`  
This scope queries for all tasks assigned to you or completed by you. If you specify `-database_id`, the query is for all tasks assigned to you and modifiable in the specified database, or completed by you in the specified database.
- `all_my_completed|all_owners_completed`  
This scope queries for all tasks completed by you. If you specify `-database_id`, the query is for all tasks completed by you in the specified database.
- `all_my_tasks|all_owners_tasks`  
This scope queries for all tasks for which you are the task resolver. If you specify `-database_id`, the query is for all tasks for which you are the task resolver and that are modifiable in the specified database or were completed in the specified database.
- `all_completed`  
This scope queries for all completed tasks. If you specify `-database_id`, the query is for all tasks completed in the specified database.
- `all_tasks`

---

This scope queries for all tasks. If you specify `-database_id`, the query is for all tasks that are modifiable in the specified database or that were completed in the specified database.

- `component_task_projects|component_task_products|  
component_task_projects_products`

This scope queries for component tasks for projects, products, or projects and products. If you specify `-database_id`, the query is for all component tasks that were created in the specified database. If you specify `-purpose`, the query is for component tasks with the specified purpose.

`-up|-update`

Specifies to update a query-based folder by running the folder's query. If a specified folder is not query-based, an error is reported.

`-w|-writable (owner | (build_mgr|build_manager|buildmanager) | all | none)`

Specifies who can modify the specified folder.

## Examples

- Add tasks **5-9** to folder **95**.

```
ccm folder -modify -at 5-9 95
Updating folder 95: Tested Tasks for Release 3.2 ...
    Added task 5
    Added task 6
    Added task 7
    Added task 8
    Task 9 is already in the folder
    Added 4 tasks.
```

- Remove tasks **5-9** from folder **95**.

```
ccm folder -modify -rt 5-9 95
Updating folder 95: Tested Tasks for Release 3.2 ...
    Removed task 5
    Removed task 6
    Removed task 7
    Removed task 8
    Removed task 9
    Removed 5 tasks.
```

- Add multiple tasks (**5, 12, 14**) to folder **51**.

```
ccm folder -modify -add_task 5,12,14 51
```

- Update the contents of folder **160**.

---

```
ccm folder -m -up 160
```

```
Updated folder '160: My Tasks for Release 3.5'.
```

- Change the mode of folder **111** so that it uses a query to add tasks.

```
ccm folder -modify -mode use_query 111
```

- Change folder **111** so that it uses the `all_my_tasks` scope and release **3.5** to add tasks.

```
ccm folder -modify -ts all_my_tasks -rel 3.5 111
```

```
The query for folder '111: bob's Insulated Development Folder' has been changed to: owner='bob' and release='3.5'
```

- Change the name of folder **85** to **Completed tasks for release 3.5**.

```
ccm folder -modify -name "Completed tasks for release 3.5" 85
```

## Related topics

- [Deleting a folder](#)
- [Listing folders](#)
- [Showing a folder property](#)
- [Showing the associated tasks or objects for a folder](#)
- [Showing folder information](#)

---

## ***Showing a folder property***

This subcommand shows the specified folder property.

```
ccm folder -s|-sh|-show (mode | (n|na|name) | (q|qu|query) | (w|wr|writable))  
    folder_spec...
```

*folder\_spec*

Specifies the folder whose properties you want to view. See [Folder specification](#) for details.

## **Related topics**

- [Copying a folder](#)
- [Deleting a folder](#)
- [Listing folders](#)
- [Modifying a folder](#)
- [Showing the associated tasks or objects for a folder](#)
- [Showing folder information](#)

---

## Showing the associated tasks or objects for a folder

For the specified folders, this command shows the associated tasks or objects of the associated tasks.

```
ccm folder -s|-sh|-show (t|task|tasks) -v|-verbose folder_spec...
ccm folder -s|-sh|-show ((t|task|tasks) | (obj|objs|objects))
    [-f|-format format] [-nf|-noformat] ([-ch|-column_header] |
    [-nch|-nocolumn_header]) [-sep|-separator separator]
    ([-sby|-sortby sortspec] | [-ns|-nosort|-no_sort])
    [-gby|-groupby groupformat] [-u|-unnumbered] folder_spec...
```

`-ch|-column_header`

Specifies to use a column header in the output format. See [-ch|-column\\_headers](#) for details.

`folder_spec`

Specifies the folders to show. See [Folder specification](#) for details.

`-f|-format format`

Specifies the command output format. See [-f|-format](#) for details.

`-gby|-groupby groupformat`

Specifies how to group the command output. See [-gby|-groupby](#) for details.

`-nch|-nocolumn_header`

Specifies not to use a column header in the output format. See [-ch|-column\\_headers](#) for details.

`-nf|-noformat`

Specifies not to use column alignment. See [-nf|-noformat](#) for details.

`-ns|-no_sort`

Specifies that the command's output will not be sorted. See [-ns|-nosort](#) for details.

`-sep|-separator separator`

Specifies a different separator character. See [-sep|-separator](#) for details.

---

`-sby|-sortby sortspec`

Specifies how to sort the command output. See [-sby|-sortby](#) for details.

`-u|-unnumbered`

Suppresses automatic numbering of the command's output (that is, the output is unnumbered). See [-u|-unnumbered](#) for details.

`-v|-verbose`

Specifies that the folder information should use the verbose format.

## Examples

- Show the tasks in folder **111**.

```
ccm folder -show tasks 111
1) Task 19: Rewrite messaging module
2) Task 26: Close box no longer active
3) Task 31: Wrong window receives message
4) Task 40: Auto-calculation gives incorrect result
5) Task 53: Download of images occurs too slowly
```

- Show the objects that are associated with folder **160**.

```
ccm folder -sh objects 160
1) UTIL.C-2:csrc:1    integrate  bob 19
2) MSGS.C-3:csrc:1   integrate  bob 19
3) MSGS.H-2:incl:1   integrate  bob 19
4) DIALOG.C-8:csrc:1 integrate  bob 57
5) DIALOG.H-13:incl:1 integrate  bob 57
```

## Related topics

- [Copying a folder](#)
- [Deleting a folder](#)
- [Listing folders](#)
- [Modifying a folder](#)
- [Showing a folder property](#)
- [Showing folder information](#)

---

## Showing folder information

This subcommand shows information about the specified folders.

```
ccm folder -s|-sh|-show (i|info|information) -f|-format format
                [-nf|-noformat] ([-ch|-column_header] | [-nch|-nocolumn_header])
                [-sep|-separator separator] folder_spec...
ccm folder -s|-sh|-show (i|info|information) [-v|-verbose] folder_spec...
```

`-ch|-column_header`

Specifies to use a column header in the output format. See [-ch|-column headers](#) for details.

*folder\_spec*

Specifies the folders to show. See [Folder specification](#) for details.

`-f|-format format`

Specifies the command output format. See [-f|-format](#) for details.

`-gby|-groupby groupformat`

Specifies how to group the command output. See [-gby|-groupby](#) for details.

`-nch|-nocolumn_header`

Specifies not to use a column header in the output format. See [-ch|-column headers](#) for details.

`-sep|-separator separator`

Specifies a different separator character. See [-sep|-separator](#) for details.

`-v|-verbose`

Specifies that you want additional folder information.

---

## Examples

- Show information for folder 2.

```
ccm folder -show info 2
```

```
Folder 2:
```

```
Description:   joe's Assigned Or Completed Tasks for Release 1.0
Owner:         joe
Writable By:   Owner
Mode:          Adds Tasks Using a Query
Query:         owner='joe' and (status='task_assigned' or
status='completed') and release='1.0'
Modifiable In: <not available>
```

## Related topics

- [Copying a folder](#)
- [Deleting a folder](#)
- [Listing folders](#)
- [Modifying a folder](#)
- [Showing a folder property](#)
- [Showing the associated tasks or objects for a folder](#)



---

## **Description and uses**

Use folders to define the update properties of projects and project groupings. Most commonly, folders are created from folder templates, which are defined in process rules.

## folder\_template command

See [Description and uses](#) for details. The `folder_template` command supports the following subcommands:

- [Creating folder templates](#)
- [Deleting folder templates](#)
- [Finding where a folder template is in use](#)
- [Listing folder templates](#)
- [Modifying a folder template](#)
- [Setting controlling database for a folder template](#)
- [Showing a folder template property](#)
- [Showing folder template information](#)

---

## Creating folder templates

This subcommand creates a folder template. You must be in the *build\_mgr* or *ccm\_admin* role to use this subcommand.

When you use an option more than once, the query expression relating to each usage is combined with an "or". For example, if you specify `-release 1.0 -release 2.0`, this contributes a query expression of `(release='1.0' or release='2.0')`.

Contributions from different options are combined with "and". For example, if you specify `-release 1.0 -platform windows`, this contributes a query expression of `(release='1.0') and (platform='windows')`.

```
ccm ft|folder_temp|folder_template -c|-create
  [-w|-writable (owner | (build_mgr|build_manager|buildmanager) |
  all | none)] [-mode ((man|manual) | (uq|use_query))]
  ([-must_be_local] | [-nomust_be_local])
  [-desc|-description description] [-cus|-custom custom_query]
  [(-db|-dbid|-database_id database_spec)...]
  [(-plat|-platform platform)...] [(-purpose purpose)...]
  [(-rel|-release release_spec)...]
  [(-sub|-subsystem subsystem)...] [-ts|-scope|-task_scope
  (user_defined | (all_my_assigned|all_owners_assigned) |
  (all_my_assigned_or_completed|all_owners_assigned_or_completed) |
  (all_my_completed|all_owners_completed) |
  (all_my_tasks|all_owners_tasks) | all_completed | all_tasks)] name
  (ct_projs|ct_projects|component_task_projects) |
  (ct_prods|ct_products|component_task_products) |
  (ct_projs_prods|ct_projects_products |
  component_task_projects_products))] name
```

`-cus|-custom custom_query`

Includes the specified custom query expression in the new folder template query.

`-desc|-description description`

When creating folders from the folder template, specifies a string used after keyword expansion. The description cannot contain newline characters. If you do not specify *description*, the folder template name is the default value. For details about keyword expansion, see [Folder template description](#).

`-db|-dbid|-database_id database_spec`

Specifies the database ID that is associated with the folder template you are creating. See [Database specification](#) for further details.

---

`-mode ((man|manual) | (uq|use_query))`

Defines the folder template contents to be either manual or query-based.

If you have not defined a query, [default task query](#) is used.

`-must_be_local`

Specifies that the folder template must use a local folder for update properties of locally created projects. The default is `-nomust_be_local`.

*name*

Specifies the name of the new folder template to be created. The *name* cannot contain newline characters.

`-nomust_be_local`

Specifies that the folder template can use a non-local folder for update properties of locally created projects. This is the default.

`-plat|-platform platform`

Specifies a query for folders created from the folder template that includes `platform='platform'`. The platform choices are defined in the `CCM_HOME\etc\om_hosts.cfg` file (Windows), or `$CCM_HOME/etc/om_hosts.cfg` file (UNIX). If a folder template applies to multiple platforms, you do not set a platform value.

`-purpose purpose`

Specifies to create the folder with a task query that includes a query for the specified purpose. See the `project_purpose` command [Description and uses](#) section for a detailed description of purposes.

This option typically applies to queries for component tasks that are specified with one of the following scopes: `component_task_projects`, `component_task_products`, or `component_task_projects_products`.

`-rel|-release release_spec`

Specifies a query for folders created from the folder template that includes `release='releasename'`. You can set `release_spec` to multiple releases. See [Release specification](#) for further details.

---

`-sub|-subsystem subsystem`

Specifies a query for folders created from the folder template that includes `task_subsys='subsystem'`.

`-ts|-scope|-task_scope`

Specifies to use a task query. The task query will include a query expression that depends on the specified scope. The query expression associated with the specified scope also depends on the `-database_id` option. You can use the following scopes:

- `user_defined`  
This scope is defined by the [default task query](#) option. If you specify `-database_id`, the query also includes a query expression for tasks modifiable in or completed in the specified database.
- `all_my_assigned|all_owners_assigned`  
This scope queries for all tasks assigned to you. If you specify `-database_id`, the query is for all tasks assigned to you that are modifiable in the specified database.
- `all_my_assigned_or_completed|all_owners_assigned_or_completed`  
This scope queries for all tasks assigned to you or completed by you. If you specify `-database_id`, the query is for all tasks assigned to you and modifiable in the specified database, or completed by you in the specified database.
- `all_my_completed|all_owners_completed`  
This scope queries for all tasks completed by you. If you specify `-database_id`, the query is for all tasks completed by you in the specified database.
- `all_my_tasks|all_owners_tasks`  
This scope queries for all tasks for which you are the task resolver. If you specify `-database_id`, the query is for all tasks for which you are the task resolver and that are modifiable in the specified database or were completed in the specified database.
- `all_completed`  
This scope queries for all completed tasks. If you specify `-database_id`, the query is for all tasks completed in the specified database.
- `all_tasks`  
This scope queries for all tasks. If you specify `-database_id`, the query is for all tasks that are modifiable in the specified database or that were completed in the specified database.
- `component_task_projects|component_task_products|component_task_projects_products`

---

This scope queries for component tasks for projects, products, or projects and products. If you specify `-database_id`, the query is for all component tasks that were created in the specified database. If you specify `-purpose`, the query is for component tasks with the specified purpose.

`-w|writable` (owner | (build\_mgr|build\_manager|buildmanager) | all | none)

Specifies who can modify folders created using the folder template. If not specified, the default is `owner`, and only the owner of the folder can modify it.

## Examples

- Create a folder template whose description is "%owner's Completed Tasks for Release %release from Database X", set the folder template to use a query, and enter a folder query. You do not need to set who can write and use the folder template because the default setting is `owner`.

```
ccm folder_template -create -description "%owner's Completed Tasks for
Release %release from Database X" -task_scope all_owners_completed
-release "%release" -database_id X "Tasks completed by %owner for
Release %release from Database X"
```

- Do the following to define a default query that a folder template will use to populate its folders with tasks:

1. Set the scope.
2. Set the release.

For parallel development and folder template management reasons, be sure to set this attribute.

3. Set the subsystem, if necessary.
4. Set the platform, if necessary.

If a folder applies to multiple platforms, you do not need to set the platform value.

5. Set the database, if it is initialized to use DCM. For example, create a new folder template. Folders created from this template will collect all completed tasks for the current release, and will be writable by build managers.

```
ccm folder_template -create -desc "All Completed Tasks for Release
%release" -task_scope all_completed -release "%" -writable
build_manager
```

## Related topics

- [Deleting folder templates](#)
- [Listing folder templates](#)

- 
- [Setting controlling database for a folder template](#)

---

## ***Deleting folder templates***

This subcommand deletes a folder template. You must be in the *build\_mgr* or *ccm\_admin* role to use this subcommand. Note that system predefined folder templates cannot be deleted by a build manager.

```
ccm ft|folder_temp|folder_template -d|-delete folder_template_spec...
```

*folder\_template\_spec*

Specifies the folder template to be deleted. See [Folder template specification](#) for details.

## **Related topics**

- [Finding where a folder template is in use](#)
- [Listing folder templates](#)
- [Showing a folder template property](#)
- [Showing folder template information](#)



---

## ***Finding where a folder template is in use***

This subcommand finds the process rule that uses the specified folder template.

```
ccm folder -fu|-finduse|-find_use
           [-f|-format format] [-nf|-noformat] ([-ch|-column_header]
           | [-nch|-nocolumn_header]) [-sep|-separator separator]
           ([-sby|-sortby sortspec] | [-ns|-nosort|-no_sort])
           [-gby|-groupby groupformat] [-u|-unnumbered]
           folder_template_spec...
```

`-ch|-column_header`

Specifies to use a column header in the output format. See [-ch|-column\\_headers](#) for details.

`folder_spec`

Specifies the folder templates to search for when performing a finduse in projects.

See [Folder template specification](#) for details.

`-f|-format format`

Specifies the command output format. See [-f|-format](#) for details.

`-gby|-groupby groupformat`

Specifies how to group the command output. See [-gby|-groupby](#) for details.

`-nch|-nocolumn_header`

Specifies not to use a column header in the output format. See [-ch|-column\\_headers](#) for details.

`-nf|-noformat`

Specifies not to use column alignment. See [-nf|-noformat](#) for details.

`-ns|-no_sort`

Specifies that the command's output will not be sorted. See [-ns|-nosort](#) for details.

`-sep|-separator separator`

Specifies a different separator character. See [-sep|-separator](#) for details.

---

`-u` | `-unnumbered`

Suppresses automatic numbering of the command's output (that is, the output is unnumbered). See [-u|unnumbered](#) for details.

### **Examples**

- Find the process rules that use the folder template All completed tasks for release %release.

```
ccm folder -finduse "All completed tasks for release %release"
```

### **Related topics**

- [Deleting folder templates](#)
- [Modifying a folder template](#)
- [Setting controlling database for a folder template](#)
- [Showing a folder template property](#)
- [Showing folder template information](#)

---

## ***Listing folder templates***

This subcommand lists folder templates.

```
ccm ft|folder_temp|folder_template -l|-list [-f|-format format]  
      [-nf|-noformat] ([-ch|-column_header] | [-nch|-nocolumn_header])  
      [-sep|-separator separator] ([-sby|-sortby sortspec] |  
      [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat]  
      [-u|-unnumbered]
```

`-ch|-column_header`

Specifies to use a column header in the output format. See [-ch|-column\\_headers](#) for details.

`-f|-format format`

Specifies the command output format. See [-f|-format](#) for details.

`-gby|-groupby groupformat`

Specifies how to group the command output. See [-gby|-groupby](#) for details.

`-nch|-nocolumn_header`

Specifies not to use a column header in the output format. See [-ch|-column\\_headers](#) for details.

`-nf|-noformat`

Specifies not to use column alignment. See [-nf|-noformat](#) for details.

`-ns|-no_sort`

Specifies that the command's output will not be sorted. See [-ns|-nosort](#) for details.

`-sep|-separator separator`

Specifies a different separator character. See [-sep|-separator](#) for details.

`-sby|-sortby sortspec`

Specifies how to sort the command output. See [-sby|-sortby](#) for details.

---

`-u` | `-unnumbered`

Suppresses automatic numbering of the command's output (that is, the output is unnumbered). See [-u|-unnumbered](#) for details.

### **Examples**

- View all personal folder templates.

```
ccm folder_template -list -template all_personal
```

- View all folder templates.

```
ccm folder_template -list
```

### **Related topics**

- [Deleting folder templates](#)
- [Modifying a folder template](#)
- [Setting controlling database for a folder template](#)

---

## Modifying a folder template

This subcommand modifies a folder template. You must be in the *build\_mgr* or *ccm\_admin* role to use this subcommand. Note that system predefined folder templates cannot be modified by a build manager.

When you use an option more than once, the query expression relating to each usage is combined with an "or". For example, if you specify `-release 1.0 -release 2.0`, this contributes a query expression of `(release='1.0' or release='2.0')`.

Contributions from different options are combined with "and". For example, if you specify `-release 1.0 -platform windows`, this contributes a query expression of `(release='1.0') and (platform='windows')`.

```
ccm ft|folder_temp|folder_template -m|-modify
  [-w|-writable (owner | (build_mgr|build_manager|buildmanager) |
  all | none)] [-mode ((man|manual) | (uq|use_query))]
  ([-must_be_local] |[-nomust_be_local])
  [-desc|-description description] [-cus|-custom custom_query]
  [(-db|-dbid|-database_id database_spec)...]
  [(-plat|-platform platform)...] [(-purpose purpose)...]
  [(-rel|-release release_spec)...]
  [(-sub|-subsystem subsystem)...]
  [-ts|-scope|-task_scope (user_defined |
  (all_my_assigned|all_owners_assigned) |
  (all_my_assigned_or_completed|all_owners_assigned_or_completed) |
  (all_my_completed|all_owners_completed) |
  (all_my_tasks|all_owners_tasks) | all_completed | all_tasks)]
  (ct_projs|ct_projects|component_task_projects) |
  (ct_prods|ct_products|component_task_products) |
  (ct_projs_prods|ct_projects_products |
  component_task_projects_products))
  folder_template_spec...
```

`-cus|-custom custom_query`

Specifies to include the specified custom query expression in the modified folder template query.

`-desc|-description description`

When modifying folders from the folder template, specifies a string used after keyword expansion. The description cannot contain newline characters. If you do not specify *description*, the folder template name is the default value. For details about keyword expansion, see [Folder template description](#).

---

`-db|-dbid|-database_id database_spec`

Specifies the database ID that is associated with the folder template you are modifying. See [Database specification](#) for further details.

`folder_template_spec`

Specifies the folder template to be modified. See [Folder template specification](#) for details.

`-mode ((man|manual) | (uq|use_query))`

Specifies whether to add tasks to the folder manually or by using a query. Rational Synergy treats changes in mode from manual to query-based in the following ways.

- If you change a manual folder template to a query-based folder template and a query is also defined in the modify command, then the specified query is used.
- If you change a manual folder template to a query-based folder template and no query is specified in the command:
  - \* If the folder template was previously query-based, its last query is used.
  - \* If the folder template was never query-based and there is a user-defined query ([default task query](#)), the user-defined query becomes the query.
  - \* If the folder template was never query-based and there is not a user-defined query ([default task query](#)), the query becomes **All Tasks Assigned to your\_user\_name**.

`-must_be_local`

Modifies the folder template so that it must use a local folder for update properties of locally created projects.

`-nomust_be_local`

Modifies the folder template so that it must use a non-local folder for update properties of locally created projects.

`-plat|-platform platform`

Specifies to update the folder template query. The new query will include `platform='platform'`. The platform choices are defined in the `CCM_HOME\etc\om_hosts.cfg` file (Windows), or `$CCM_HOME/etc/om_hosts.cfg` file (UNIX). If a folder template applies to multiple platforms, you do not set a platform value.

---

`-purpose purpose`

Specifies to create the folder with a task query that includes a query for the specified purpose. See the `project_purpose` command [Description and uses](#) section for a detailed description of purposes.

This option typically applies to queries for component tasks that are specified with one of the following scopes: `component_task_projects`, `component_task_products`, or `component_task_projects_products`.

`-rel|-release release_spec`

Specifies to update the folder template with a new query. The query will include `release='releasename'`. You can set `release_spec` to multiple releases. See [Release specification](#) for further details.

`-sub|-subsystem subsystem`

Specifies to update the folder template with a new query. The query will include `task_subsys='subsystem'`.

`-ts|-scope|-task_scope`

Specifies to use a task query. The task query will include a query expression that depends on the specified scope. The query expression associated with the specified scope also depends on the `-database_id` option. You can use the following scopes:

- `user_defined`  
This scope is defined by the [default task query](#) option. If you specify `-database_id`, the query also includes a query expression for tasks modifiable in or completed in the specified database.
- `all_my_assigned|all_owners_assigned`  
This scope queries for all tasks assigned to you. If you specify `-database_id`, the query is for all tasks assigned to you that are modifiable in the specified database.
- `all_my_assigned_or_completed|all_owners_assigned_or_completed`  
This scope queries for all tasks assigned to you or completed by you. If you specify `-database_id`, the query is for all tasks assigned to you and modifiable in the specified database, or completed by you in the specified database.
- `all_my_completed|all_owners_completed`  
This scope queries for all tasks completed by you. If you specify `-database_id`, the query is for all tasks completed by you in the specified database.
- `all_my_tasks|all_owners_tasks`

---

This scope queries for all tasks for which you are the task resolver. If you specify `-database_id`, the query is for all tasks for which you are the task resolver and that are modifiable in the specified database or were completed in the specified database.

- `all_completed`

This scope queries for all completed tasks. If you specify `-database_id`, the query is for all tasks completed in the specified database.

- `all_tasks`

This scope queries for all tasks. If you specify `-database_id`, the query is for all tasks that are modifiable in the specified database or that were completed in the specified database.

- `component_task_projects|component_task_products|component_task_projects_products`

This scope queries for component tasks for projects, products, or projects and products. If you specify `-database_id`, the query is for all component tasks that were created in the specified database. If you specify `-purpose`, the query is for component tasks with the specified purpose.

`-w|-writable (owner | (build_mgr|build_manager|buildmanager) | all | none)`

Specifies to update the writable property for the folder template. All folders that were created from and are controlled by the folder template will be updated to reflect the new permissions.

## Related topics

- [Listing folder templates](#)
- [Modifying a folder](#)
- [Showing a folder template property](#)
- [Showing folder template information](#)



---

## Setting controlling database for a folder template

This subcommand takes local control, handover control, or accept control for a folder template. See "Replication of generic and release-specific processes" in the [Rational Synergy Distributed](#) book for details.

```
ccm ft|folder_temp|folder_template -cdb|-controlling_database
    -local folder_template_spec...
ccm ft|folder_temp|folder_template -cdb|-controlling_database
    -handover database_spec folder_template_spec...
ccm ft|folder_temp|folder_template -cdb|-controlling_database
    -accept database_spec folder_template_spec...
```

`-accept database_spec`

Specifies to accept DCM updates from a specified database. You can set `database_spec` to one DCM database definition. See [Database specification](#) for details about using `database_spec`. See "Replication of generic and release-specific processes" in the *Rational Synergy Distributed* book for details about accepting DCM updates.

`folder_template_spec`

Specifies the folder template to be updated. See [Folder template specification](#) for details.

`-handover database_spec`

Specifies that control of the object is handed over from the current database to the specified database. The default value when creating a DCM database definition is a blank string. This means that when you hand over control to a spoke via a hub database, you must specify the hub `database_spec` for the `database_spec` value. The specified `database_spec` must be either a known DCM database definition for which a generate is permitted, or a blank string. A blank string means that control can't be handed over to that database.

A detailed description of this option is in [Rational Synergy Distributed](#), in the "Controlling database and handover of control" section.

`-local`

Specifies that local control should be taken over for the folder template. The object will no longer be updated by DCM replication from another database. A detailed description of this option is in "Replication of generic and release-specific processes" in the *Rational Synergy Distributed* book.

---

### **Example**

- Hand over control of a folder template called **All system testing tasks for server** to database A1.

```
ccm folder_template -controlling_database -handover A1 "All system
testing tasks for server"
```

### **Related topics**

- [Finding where a folder template is in use](#)
- [Listing folder templates](#)
- [Showing a folder template property](#)
- [Showing folder template information](#)

---

### ***Showing a folder template property***

This subcommand shows a specific property of a folder template.

```
ccm ft|folder_temp|folder_template -s|-sh|-show ((desc|description) |  
mode | query | (w|wr|writable)) folder_template_spec...
```

*folder\_template\_spec*

Specifies the folder template to show. See [Folder template specification](#) for details.

### **Related topics**

- [Deleting folder templates](#)
- [Modifying a folder template](#)

---

## Showing folder template information

This subcommand shows information about a folder template.

```
ccm ft|folder_temp|folder_template -s|-sh|-show (i|info|information)
    -f|-format format [-nf|-noformat] ([-ch|-column_header] |
    [-nch|-nocolumn_header]) [-sep|-separator separator]
    folder_template_spec...
ccm ft|folder_temp|folder_template -s|-sh|-show (i|info|information)
    folder_template_spec...
```

`-ch|-column_header`

Specifies to use a column header in the output format. See [-ch|-column headers](#) for details.

`folder_template_spec`

Specifies the folder template to show. See [Folder template specification](#) for details.

`-f|-format format`

Specifies the command output format. See [-f|-format](#) for details.

`-gby|-groupby groupformat`

Specifies how to group the command output. See [-gby|-groupby](#) for details.

`-nch|-nocolumn_header`

Specifies not to use a column header in the output format. See [-ch|-column headers](#) for details.

`-nf|-noformat`

Specifies not to use column alignment. See [-nf|-noformat](#) for details.

`-ns|-no_sort`

Specifies that the command's output will not be sorted. See [-ns|-nosort](#) for details.

`-sep|-separator separator`

Specifies a different separator character. See [-sep|-separator](#) for details.

---

`-sby|-sortby sortspec`

Specifies how to sort the command output. See [-sby|-sortby](#) for details.

`-u|-unnumbered`

Suppresses automatic numbering of the command's output (that is, the output is unnumbered). See [-u|-unnumbered](#) for details.

## Related topics

- [Modifying a folder template](#)
- [Showing a folder template property](#)

---

## Description and uses

Folder templates provide a pattern used to create folders. Folders created from a folder template are controlled by that folder template. Therefore, when you make changes to the folder template, the folders controlled by it are updated.

### Folder template description

You can give a folder template any description, using any character, except the pipe (|) character. The folder template description can include the following keywords in any combination: %owner, %release, %database. A description for a folder template does not have to include keywords. When the description of a folder template does not contain a keyword, all folders created from this folder template will have the same description.

### Keyword %release

For example, if you create a folder template whose description is **Completed Tasks for Release %release**, the keyword %release is expanded to the release value of the projects that are using a process rule containing this folder template. The keyword %release is expanded when this folder template creates a folder. For instance, when a project with **release 2.0** uses a process rule that contains a folder template whose description is **Completed Tasks for Release %release**, the folder created from this template and added to the update properties for the project will have a description of **Completed Tasks for Release 2.0**.

### Keyword %owner

The %owner keyword expands to the owner of the project whose update properties contain folders created from a folder template. For example, if a project owned by *jsmith* with a release of **3.1** uses a process rule that contains a folder template with a description of **%owner's Completed Tasks for Release %release**, a folder with a description of **jsmith's Completed Tasks for Release 3.1** is created from this folder template and added to the update properties for the project.

### Keyword %database

The %database keyword expands to the DCM database identifier of the database where the project using the folder was created. For example, if a project owned by *jsmith* with a release of **3.1** is in a DCM database called `Bristol`, and is using a process rule that contains a folder template with a description of **%owner's Completed Tasks for Release %release from Database %database**, a folder with a description of **jsmith's Completed Tasks for Release 3.1 from Database Bristol** is created from this folder template and added to the update properties for the project.

## groups command

See [Description and uses](#) for details. The `groups` command supports the following subcommands:

- [Assigning groups to objects](#)
- [Unassigning groups from objects](#)

---

## Assigning groups to objects

This subcommand assigns one or more group restrictions to an object. This adds the specified entries to the object's *groups* attribute.

For a file, directory or project, one of the following must be true:

- The object must be the first version of the object and the user must have write access to it.
- The user must be in the *group\_mgr* role and be a member of the object's current groups.
- The user must be in the *ccm\_admin* role.

For any other type of object, one of the following must be true:

- The user must be in the *group\_mgr* role and be a member of the object's current groups.
- The user must be in the *ccm\_admin* role.

```
ccm groups -a|-assign (-v|-value group_item_list)... object_spec...
```

```
object_spec...
```

Specifies the object to be updated. See [Object specification](#) for details.

```
-v|-value group_item_list
```

Specifies the group restriction to be added. The *group\_item\_list* is a list of one or more items separated by commas or spaces and comma and spaces, where each item is in the following form:

```
group_name
```

```
group_name:readsource
```

The *group\_name* is the name of a group and defines modify or check out access restrictions. The *group\_name:readsource* defines access restrictions on the visibility of the source attribute.

### Example

- Assign the groups **sqe\_team** and **design\_team** to an object named **makefile.pc-1:makefile:tut70#4**.

```
ccm groups -assign "sqe_team, design_team" makefile.pc-1:makefile:tut70#4
```

### Related topics

- [Unassigning groups from objects](#)



---

## ***Unassigning groups from objects***

This subcommand unassigns one or more group restrictions from an object. This removes the specified entries to the object's *groups* attribute.

For a file, directory or project, one of the following must be true:

- The object must be the first version of the object and the user must have write access to it.
- The user must be in the *group\_mgr* role and be a member of the object's current groups.
- The user must be in the *ccm\_admin* role.

For any other type of object, one of the following must be true:

- The user must be in the *group\_mgr* role and be a member of the object's current groups.
- The user must be in the *ccm\_admin* role.

```
ccm groups -unassign (-v|-value group_item_list)... object_spec...
```

```
object_spec...
```

Specifies the object to be updated. See [Object specification](#) for details.

```
-v|-value group_item_list
```

Specifies the group restriction to be added. The *group\_item\_list* is a list of one or more items separated by commas or spaces and comma and spaces, where each item is in the following form:

```
group_name
```

```
group_name:readsource
```

The *group\_name* is the name of a group and defines modify or check out access restrictions. The *group\_name:readsource* defines access restrictions on the visibility of the source attribute.

## **Related topics**

- [Assigning groups to objects](#)

---

## Description and uses

A Rational Synergy database can contain many different collections of objects. It may not always be appropriate for all users with the applicable role to view, check out, and modify all objects. Group security restricts check out and modifies permissions to a specified group of users. In addition, read security, which limits visibility of the source contents of objects to designated groups, can be specified. Use the `groups` command to implement and define security for objects.

If you are working as the group manager, use group security to:

- define a named group of users.
- define and modify the users that are members of that named group.
- restrict check out, read, and modify access of an object to specified named groups by assigning one or more groups to it.
- restrict visibility of the source contents of a file to specified named groups by assigning one or more groups with read source controls to it.

A user working in a role that can create objects, such as *developer*, *writer*, *component\_developer* or *build\_mgr*, can restrict access of an object to specified groups if that object is the only version and the user can modify that object.

Read security is implemented by providing access control to the source attribute of an object. Users can query for objects and see other attributes regardless of any read restrictions. Read security applies to source objects which can be versioned, and does not apply to directories and projects.

**Note** If you apply read security to an object that is currently in users' work areas, the files will still be readable by the users.

Three different levels of read access security can be defined:

- An object that has no read access restrictions to its source can be accessed by any user.
- An object that has one or more groups defined for read access will only allow access to the source if the user is a member of at least one of those groups. All other users will be denied access to the source contents of that object.
- An object with the highest level of security (no access to the source) can't be viewed, checked out, or modified, but other attributes can be viewed. However, users working in the *ccm\_admin* role can always view the source contents of files.

Any object that is checked out inherits the same group security restrictions as its predecessor, including read security restrictions.

The following examples illustrate how security is applied and used for an object:

- 
1. When no groups exist, or no groups are assigned to an object, there are no restrictions, meaning everyone can view, check out, and modify source files.
  2. When one or more groups are created, and one group is assigned to that object, only users in the specified group can view, check out, and modify files. Users not in the specified group can only view the source objects. In other words, check out and modify security is implemented, but read security does not yet exist.
  3. When one or more groups are created, and one group is given read security access (the ability to view source files), then all other groups do not have read access to the files. So after you start using the read security option, access to source contents is denied by default.

Note that read security can only be used with copy-based work areas. For additional information about setting up databases for read security, see the appropriate *Administration Guide*. On Windows, refer to the [Rational Synergy Administration Guide for Windows](#). On UNIX, refer to the [Rational Synergy Administration Guide for UNIX](#).

If you have a directory in the *public* state that uses group security and the user is not a member of any of the directory's groups, the user can still create new objects, add them to, or unuse them from the directory. Users can easily overwrite each others' changes if you use public directories; exercise caution when using them.

For detailed information about using group security with DCM, see the [Rational Synergy Distributed](#) book

## history command

See [Description and uses](#) for details. The `history` command supports the [Showing history for an object](#) subcommand.

---

## Showing history for an object

```
ccm hist|history -p|-project [-f|-format format] [-nf|-noformat]
    ([-ch|-column_header] | [-nch|-nocolumn_header])
    [-sep|-separator separator] ([-sby|-sortby sortspec] |
    [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat] project_spec...
ccm hist|history [-f|-format format] [-nf|-noformat]
    ([-ch|-column_header] | [-nch|-nocolumn_header])
    [-sep|-separator separator] ([-sby|-sortby sortspec] |
    [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat] object_spec...
```

`-ch|-column_header`

Specifies to use a column header in the output format. See [-ch|-column headers](#) for details.

`-f|-format format`

Specifies the command output format. See [-f|-format](#) for details.

A keyword can be built-in (%fullname, %displayname, %objectname) or the name of any existing attribute such as %modify\_time or %status.

See [Built-In keywords](#) for a list of keywords.

`-gby|-groupby groupformat`

Specifies how to group the command output. See [-gby|-groupby](#) for details.

`-nch|-nocolumn_header`

Specifies not to use a column header in the output format. See [-nch|-nocolumn headers](#) for details.

`-nf|-noformat`

Specifies not to use column alignment. See [-nf|-noformat](#) for details.

`-ns|-nosort|-no_sort`

Specifies that the command's output will not be sorted. See [-ns|-nosort](#) for details.

`object_spec`

Specifies the project, file, directory, or release definition to show.

`-p|-project`

Shows the history of a project.

---

*project\_spec*

Specifies the project for which to show history. See [Project specification](#) for details.

*-sby|-sortby sortspec*

Specifies how to sort the command output. See [-sby|-sortby](#) for details.

*-sep|-separator separator*

Used only with the *-f|-format* option. Specifies a different separator character. See [-sep|-separator](#) for details.

### **Example**

- Examine the history of `main.c` from within the parent project work area.

```
ccm history main.c
Object:  main.c-1 (csrc:2)
Owner:   john
State:   integrate
Created: Tue Jun 4 13:04:23 1999
Task:    4
Comment:
Predecessors:
Successors:

    main.c-2:csrc:2
*****
Object:  main.c-2 (csrc:2)
Owner:   tom
State:   integrate
Created: Mon Jun 24 18:02:22 1999
Task:    7
Comment:
Predecessors:
    main.c-1:csrc:2
Successors:
    main.c-3:csrc:2
*****
Object:  main.c-3 (csrc:2)
Owner:   john
State:   working
Created: Mon Aug 12 18:03:31 1999
Task:    12
Comment:
Predecessors:
```

---

```
main.c-2:csrc:2
Successors:
*****
```

## Related topics

- [Clearing entries from history](#)
- [Showing command history](#)
- [Setting a maximum number of commands to record](#)

---

## Description and uses

Use the `history` command to show the version history of a project, directory, file or release. Any user may perform this operation.



## In command

See [Description and uses](#) for details. The `ln` command supports the [Creating a symbolic link](#) subcommand.

---

## Creating a symbolic link

This subcommand is available to UNIX users only.

```
ccm ln [-s] [-c|-comment comment_string] [-ce|-commentedit]
      [-cf|-commentfile file_path] [-t|-task task_spec] file_path file_spec
```

`-c|-comment comment`

Specifies a comment that will be appended on all baseline projects and their members when they are checked in to the *released* state. The *comment* can contain more than one line and accepts backslash encoded values.

You can use this option with `-commentedit` and `-commentfile`. If you use the `-commentedit` option, the comment displays in the default text editor.

`-ce|-commentedit`

Specifies that the default text editor will be invoked to allow the comment to be composed and edited. The result saved from the text editor is used as the final comment. You can use this option with the `-comment` and `-commentfile` options.

`-cf|-commentfile file_path`

Specifies that the contents of the specified file will be used for the comment. If you specified `-comment`, it is appended to that comment. You can use this option with the `-commentedit` option.

*file\_spec*

Specifies the name or the name and version of the symbolic link. The *file\_spec* must provide a context project and parent directory. You may use a work area reference form or project reference spec form (see [File specification](#) for details). If a version is not specified, then a default version is used.

*file\_path*

Specifies the path to which the symbolic link points. This does not have to be a controlled file or a path in a maintained work area.

`-s`

Provides UNIX-style compatibility; otherwise, the option is ignored.

---

`-t|-task <task_spec>`

Specifies that the symbolic link is associated with the specified task. You can set the `task_spec` to a single task (see [Task specification](#) for details). If this option is not specified, the symbolic link is associated with any current task that is set.

### **Example**

- Create a symbolic link called `sort.c` to the `sort.c` object in the `ico_2-1` project.

```
ccm ln -s \  
/user/ccm_user_Aug10/ico_2-1/ico_2/utils/sort.c sort.c  
Member sort.c-1 added to project ico_2-1  
ccm ln -t 44 /users/kg/ccm_wa/keng421/john-unix/john/init.c /users/gke
```

### **Related topics**

- [work\\_area command](#)

---

## Description and uses

This subcommand creates a symbolic link object in the database. If you execute the command within the context of a project with a maintained work area, you may only execute it on a UNIX client. Symbolic links can only be represented in a UNIX work area that is updated by a UNIX client.

The `ln` command creates a controlled symbolic link from `file_spec` to `path_name`. The link may point to any path. It does not have to be a controlled object or a path within a maintained project work area.

**Note** When you create a new link in a non-writable directory, a new directory version is checked out automatically.

If you are working in a shared project and your current directory is not writable, the directory is checked out, associated automatically with the default (or specified) task, and checked in to the *integrate* state. Set [shared project directory checkin](#) to FALSE in your initialization file to disable this feature.

## ls command

See [Description and uses](#) for details. The `ls` command supports the [Listing files](#) subcommand.

---

## Listing files

The `ls` command operates on UNIX operating systems only.

```
ccm ls -p|-project [-m] ([-l] | [-f|-format format]) [-R] [-nf|-noformat]
      ([-ch|-column_header] | [-nch|-nocolumn_header])
      [-sep|-separator separator] ([-sby|-sortby sortspec] |
      [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat] project_spec...
ccm ls [-m] ([-l] | [-f|-format format]) [-R] [-nf|-noformat]
      ([-ch|-column_header] | [-nch|-nocolumn_header])
      [-sep|-separator separator] ([-sby|-sortby sortspec] |
      [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat]
      [path_or_file_spec...]
```

`-ch|-column_header`

Specifies to use a column header in the output format. See [-ch|-column\\_headers](#) for details.

`-f|-format format`

Specifies the command output format. See [-f|-format](#) for details.

A keyword can be built-in (`%fullname`, `%displayname`, `%objectname`) or the name of any existing attribute such as `%modify_time` or `%status`.

See [Built-In keywords](#) for a list of keywords.

`-gby|-groupby groupformat`

Specifies how to group the command output. See [-gby|-groupby](#) for details.

`-l`

Specifies that a long default format should be used. This can only be used when a user-defined format has not been specified with `-f|-format`.

`-m`

Shows both uncontrolled files and directories as well as controlled ones. If no user-defined format has been specified with the `-f|-format` option, the default format (short or long form) includes a column that indicates the synchronization status for files as follows:

- Local copy ((LC)

Denotes files that are in the project, but have a local copy rather than a symbolic link in the work area.

---

If files are displayed with this mark and your work area is link-based, perform a reconcile operation. For more information, see the [reconcile command](#).

- Not synchronized (NS)

Denotes files that are in the project, but not in the work area. This situation occurs when you add files to the project, but your work area is not visible, or when the link or local copy of a file is deleted.

If most of the files in your work area are displayed with this mark, perform a reconcile operation. For more information, see the [reconcile command](#).

- Uncontrolled (UC)

Denotes files that are in the work area, but not in the project. To view uncontrolled files marked with UC, you must use the `-m` option with the `-l` option.

In user-defined formats, you can use the `%Sync` keyword to show the synchronization status.

If the object is more than six months old, the year is shown instead of the time.

`-nch|-nocolumn_header`

Specifies not to use a column header in the output format. See [-nch|-nocolumn\\_headers](#) for details.

`-nf|-noformat`

Specifies not to use column alignment. See [-nf|-noformat](#) for details.

`-ns|-nosort|-no_sort`

Specifies that the command's output will not be sorted. See [-sby|-sortby](#) for details.

*path\_or\_file\_spec*

Specifies the path to be listed. You can set the *path\_or\_file\_spec* to a project, directory or file defined in the database. This can also be an empty directory entry. If omitted, the current working directory is listed. See [File specification](#) for details.

`-p|-project`

Shows the history of a project.

*project\_spec*

Specifies the project to list. See [Project specification](#) for details.

---

-R

Displays subdirectory members recursively. The command does not recurse into subprojects.

-sby|-sortby *sortspec*

Specifies how to sort the command output. See [-sby/-sortby](#) for details.

-sep|-separator *separator*

Used only with the `-f|-format` option. Specifies a different separator character. See [-sep/-separator](#) for details.

### **Example**

- List the current directory in the long format.

```
ccm ls -l
working john 2008-07-25 11:56 csrc 1 alias.c-4.5 27
working john 2008-07-25 11:56 csrc 1 diff.c-4.5 27
working john 2008-07-25 11:56 csrc 1 move.c-4.5 27
working john 2008-07-25 11:57 csrc 1 start.c-4.5 27
```



---

## Description and uses

The `ls` command lists the contents of a directory object version in a work area. By default, the output consists of a list of objects and their associated projections in the file system.

The `ls` command displays two categories of files: objects under Rational Synergy control and files that exist in the file system only. To find out how to display these files, see the `-l` option and the `-m` option.

## merge command

See [Description and uses](#) for details. The `merge` command supports the [Merging files or directories](#) subcommand.

---

## Merging files or directories

This subcommand merges files or directories.

```
ccm merge [[-create_task] | [-t|-task task_spec]]
          [-c|-comment comment_string]
          [-ce|-commentedit] [-cf|-commentfile file_path]
          file_spec1 file_spec2
```

`-c|-comment comment`

Specifies a comment that will be appended on all baseline projects and their members when they are checked in to the *released* state. The *comment* can contain more than one line and accepts backslash encoded values.

You can use this option with `-commentedit` and `-commentfile`. If you use the `-commentedit` option, the comment displays in the default text editor.

`-ce|-commentedit`

Specifies that the default text editor will be invoked to allow the comment to be composed and edited. The result saved from the text editor is used as the final comment. You can use this option with the `-comment` and `-commentfile` options.

`-cf|-commentfile file_path`

Specifies that the contents of the specified file will be used for the comment. If you specified `-comment`, it is appended to that comment. You can use this option with the `-commentedit` option.

`-create_task`

Causes a task to be created automatically when Rational Synergy creates the new, merged object version, and associates the new object version with that task.

The task is assigned to the user who performed the merge. The release value of a task is set to the release value of the project in which the new object version is created. If the object version is created outside of a project, the release value is not set.

---

*file\_spec1*

*file\_spec1* specifies the first file or directory to be merged. This is used for determining the default next version for the merged object. You can set *file\_spec1* to be a [File specification](#) for one file or directory.

*file\_spec2*

*file\_spec2* specifies the second file or directory to be merged. You can set *file\_spec2* to be a [File specification](#) for one file or directory. If *file\_spec1* is a file, then *file\_spec2* must also be a file. If *file\_spec1* is a directory, then *file\_spec2* must also be a directory.

-t|-task *task\_spec*

Specifies the task to associate the new merged version with. If you don't specify `-task` but you've set a current task, the current task is used by default. See [Task specification](#) for further details.

---

## Description and uses

When you use the `merge` command to merge source files or directories, the merge tool compares the versions you selected, then compares the differences of each version to the closest common ancestor. Rational Synergy creates a new, merged, controlled version automatically when you exit your merge tool.

The `merge` operation works with both static and modifiable, non-static objects. This feature merges parallel versions, even when parallel check-in is prevented, in order to use the merge tool. In previous releases, the parallel check-in was not allowed, so the merge tool was not invoked.

If you request Rational Synergy to create a new task automatically at the merge, Rational Synergy obtains the `release` value of the task from the project in which `file_spec1` resides.

### **File Merge**

Each type of object for which you can merge source has default merge tools predefined by Rational Synergy for both the CLI and the GUI.

The `automerge` tool creates a new, controlled version of your file. If the merge is successful, the merge results are written to the new file.

An area "in conflict" occurs when both versions have changes in the same place relative to the predecessor. If your merged file contains any conflicts, a warning is issued, the tool marks the conflicts so you can find them quickly and easily, then `automerge` writes the merge results to the new file.

The following example shows how the temporary file is marked:

```
<<<<<<file1 (file1 changes recommended)
unique lines in file1
===== (common lines)
unique lines in file2
>>>>>>file2 (file2 changes recommended)
```

### **Directory Merge**

From the command line, the merge tool automatically includes all members from both directories in a new, controlled, merged directory. If one of the objects to be merged is a member of the current project, Rational Synergy uses the new merged directory in the project. This applies to both root directories and subdirectories.

## move command

See [Description and uses](#) for details. The `move` command supports the following subcommands:

- [Renaming a project](#)
- [Renaming or moving an object](#)

---

## ***Renaming a project***

This subcommand renames a project. After you rename the project, the root directory is renamed automatically to reflect the new name of the project.

If you attempt to rename a project that is writable by you, but has a root directory that is not writable, the operation fails. You must check out the root directory first. Then, Rational Synergy automatically renames the root directory when you rename the product.

**Note:** If the project is used as a subproject, you must check out the parent directories that use the renamed project and update them to use the renamed project.

```
ccm move -p|-project [-task task_spec] project_spec new_project_spec
```

*new\_project\_spec*

Specifies the name and optionally the version to be used for the rename. This can be:

- a name
- a name, version delimiter, and version
- a name, colon, and version.

*project\_spec*

Specifies the project to rename. See [Project specification](#) for details.

-task *task\_spec*

Specifies the task with which the renamed project root directory is associated. If not specified and a current task is set the current task is used by default. See [Task specification](#) for details.

---

## Renaming or moving an object

This subcommand renames a file or directory or moves one or more files, directories or projects to another directory (which may be in a different project).

If the last argument is not set to an existing directory and there are two arguments, the command is interpreted as a rename operation. Otherwise, the command is interpreted as a move operation.

When you rename a file or directory, if the parent directory is not writable, Rational Synergy checks out a new parent directory version automatically and associates it with the specified task. If no task is specified, any current task is used by default. You must check in the directory to make it available to other users. This is normally done when you complete the task used for the operation.

When you move an object to or from a directory that is not writable, if the parent directory is not writable, Rational Synergy checks out a new parent directory version automatically and associates it the specified task. If no task is specified, any current task is used by default. You must check in the directory to make it available to other users. This is normally done when you complete the task used for the operation.

If you are in a shared project and the parent directory is not writable, Rational Synergy checks out the directory, associates it automatically with specified or current task, and checks it in to the *integrate* state. You can disable this feature by setting `shared_project_directory_checkin` to `FALSE` in your initialization file (see [shared project directory checkin](#)).

You do not need to be in a work area to use this command if as you use the [Project reference form](#).

**Note:** If you attempt to rename a file or directory used in other directories or directory versions other than the current or specified directory, you should review the other locations using the object. Rational Synergy updates the parent directory, but not other parent directories. You must check out those other directories to use the newly renamed object.

```
ccm move [-task task_spec] file_spec... file_spec
```

*file\_spec*

Specifies the files or directories to be renamed or moved, and the new name or new location. If the last *file\_spec* argument is set to an existing controlled directory, the objects are moved from their current location to that directory. The directory can be in the same or different project as the objects being moved. If the last *file\_spec* argument is not set to an existing controlled directory and only two arguments are specified, the file or directory specified by the first argument is renamed to the name and optional version specified by the second (last) argument. See [File specification](#) for details.



---

`-task task_spec`

Specifies the task with which any parent directory that was automatically checked out is associated. If not specified and a current task is set, the current task is used by default. See [Task specification](#) for details.

## Examples

- Move the file `octagon.h` from the `src` directory to the `incl` directory in your current project.

Windows:

```
ccm move src\octagon.h incl/
```

UNIX:

```
ccm move src/octagon.h incl/
```

Rename the file `turquoise.c` to `magenta.c` in the current project.

```
ccm move turquoise.c magenta.c
```

- Rename the `ccm_aug8-1` project to `test_a-1`.

Windows:

```
ccm move -p ccm_aug8-1 test_a-1
```

Setting path for work area of 'test\_a-1' to  
'c:\users\mary\ccm\_wa\ccmint07\test\_a-1'...

UNIX:

```
ccm move -p ccm_aug8-1 test_a-1
```

Setting path for work area of 'test\_a-1' to '/mary/ccm\_wa/ccmint07/  
test\_a-1'...

- Rename the `hello.c` file to `hi_world.c`, then move it to another project directory.

Windows:

```
ccm move proj\hello.c@proj-1 screen\src\hi_dir\hi_world.c@beta-3
```

UNIX:

```
ccm move proj/hello.c@proj-1 screen/src/hi_dir/hi_world.c@beta-3
```

- Move the file `hello.c` from `beta-1` to a new project called `final-1`.

Windows:

```
ccm move beta-1\hello.c@beta-1 final@final-1
```

UNIX:

```
ccm move beta-1/hello.c@beta-1 final@final-1
```

---

## Description and uses

The `move` command has the following uses:

- Renames a file or project. After you rename the project, the root directory is renamed automatically to reflect the new project name.
- Moves one or more files to another directory.
- Moves a file to a new project (in a new work area).
- Moves one or more directories and their contents to a particular directory.
- Moves a subproject to a new top-level project.
- Moves one or more subprojects and their contents to another directory.

---

## process command

See [Description and uses](#) for details. The `process` command supports the following subcommands:

- [Copying a process to an existing process](#)
- [Copying a process to a new process](#)
- [Creating a process](#)
- [Deleting a process](#)
- [Listing processes](#)
- [Modifying a process](#)
- [Showing a property for a process](#)
- [Showing process information](#)
- [Showing process rules for a process](#)

---

## **Copying a process to an existing process**

This subcommand copies the properties and process rules from a specified process to another process. You must be in the *build\_mgr* or *ccm\_admin* role to use this subcommand.

```
ccm process -cp|-copy -existing [(-apr|-add_process_rule|
    -add_process_rules generic_process_rule_spec)...]
    [(-rpr|-remove_process_rule|-remove_process_rules
    generic_process_rule_spec)...]
    from_process_spec to_process_spec
```

*-apr|-add\_process\_rule|-add\_process\_rules *generic\_process\_rule\_spec*...*

Specifies to add the generic process rules copied from the source process before copying it to the existing process.

*from\_process\_spec*

Specifies the process to copy. You can set the *from\_process\_spec* specification to a single process. See [Process specification](#) for details.

*to\_process\_spec*

Specifies the process to be updated. You can set the *to\_process\_spec* specification to a single process. See "Process specification" for details.

*-rpr|-remove\_process\_rule|-remove\_process\_rules *generic\_process\_rule\_spec*...*

Specifies to remove the generic process rules in the source process before copying it to the existing process.

### **Example**

- Copy the standard process to an existing process called `custom`, so that it uses the process rules for the standard process.

```
ccm process -copy standard -existing custom
```

### **Related topics**

- [Listing processes](#)
- [Showing process information](#)
- [Showing a property for a process](#)
- [Showing process rules for a process](#)

---

## **Copying a process to a new process**

This subcommand copies the properties and process rules from a specified process to a new process. You must be in the *build\_mgr* or *ccm\_admin* role to use this subcommand.

```
ccm process -cp|-copy -new [(-apr|-add_process_rule|  
-add_process_rules process_rule_spec)...]  
[(-rpr|-remove_process_rule|-remove_process_rules  
process_rule_spec)...]  
process_spec new_process_spec
```

```
-apr|-add_process_rule|-add_process_rules process_rule_spec...
```

Specifies to add the process rules copied from the source process to the new process.

```
-cp|-copy
```

Accepts two [Process specification](#) arguments. You can set each to a single object.

```
process_spec
```

Specifies the process to copy. See "Process specification" for details.

```
new_process_spec
```

Specifies the name of the process to create. See "Process specification" for details.

```
-rpr|-remove_process_rule|-remove_process_rules process_rule_spec...
```

Specifies to remove the process rules in the source process before copying it to the new process.

## **Example**

- Create a custom process based on the standard process with a new custom process rule for Integration Testing.

```
ccm process -copy standard -new custom -apr "Custom Integration  
Testing"
```

## **Related topics**

- [Listing processes](#)
- [Showing process information](#)
- [Showing a property for a process](#)
- [Showing process rules for a process](#)

---

## Creating a process

This subcommand creates a process definition, which includes specified process rules and an optional specified diagram uniform resource locator (URL). You must be in the *build\_mgr* or *ccm\_admin* role to use this subcommand.

```
ccm process -c|-create [-diagram|-diagramurl diagram_url_string]  
                [(-pr|-prs|-process_rule|-process_rules  
                generic_process_rule_spec)...] process_spec
```

```
-diagram|-diagramurl diagram_url_string
```

Specifies the diagram URL for the new process.

The diagram URL points to a file that contains more information about the process. Default processes have a URL to a descriptive file on the help server. The descriptive file contains a diagram that shows the flow of the baseline and tasks to the project groupings. You can view this file by using the `-show diagramurl` subcommand.

If you don't have the diagram ready, later you can set the URL to the path where the descriptive file will reside.

```
process_spec
```

Specifies the name of the new process. See [Process specification](#) for details.

```
-pr|-prs|-process_rule|-process_rules generic_process_rule_spec...
```

Specifies the generic process rules to be added to the new process.

## Example

- Create a custom process with generic process rules for **Insulated Development**, **Integration Testing**, and **System Testing**.

```
ccm process -create custom -pr "Insulated Development" -pr  
"Integration Testing" -pr "System Testing"
```

## Related topics

- [Listing processes](#)
- [Showing process information](#)
- [Showing a property for a process](#)
- [Showing process rules for a process](#)

---

## Deleting a process

This subcommand deletes processes. By default, processes can only be deleted if they are not used by any project grouping. You must be in the *build\_mgr* or *ccm\_admin* role to use this subcommand.

```
ccm process -d|-delete process_spec...
```

**-d|-delete**

The command accepts one or more *process\_spec* arguments. You can set each to multiple process definitions.

**-force**

Deletes the process rule even if it is used by a project grouping. If you omit the *-force* command, the process rule is only deleted if it is not used.

*process\_spec*

Specifies the processes to delete. See [Process specification](#) for details.

### Example

- Delete a process called *custom*.

```
ccm process -delete custom
```

### Related topics

- [Listing processes](#)
- [Showing process information](#)
- [Showing a property for a process](#)
- [Showing process rules for a process](#)



---

## ***Listing processes***

This subcommand lists the processes matching the specified criteria. If no options are specified, it lists all processes.

```
ccm process -l|-list [-f|-format format] [-nf|-noformat]
                ([-ch|-column_header] | [-nch|-nocolumn_header])
                [-sep|-separator separator] ([-sby|-sortby sortspec] |
                [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat]
                [-u|-unnumbered]
```

`-ch|-column_header`

Specifies to use a column header in the output format. See [-ch|-column\\_headers](#) for details.

`-f|-format format`

Specifies the command output format. See [-f|-format](#) for details.

A keyword can be built-in (%fullname, %displayname, %objectname) or the name of any existing attribute such as %modify\_time or %status.

See [Built-In keywords](#) for a list of keywords.

`-gby|-groupby groupformat`

Specifies how to group the command output. See [-gby|-groupby](#) for details.

`-nch|-nocolumn_header`

Specifies not to use a column header in the output format. See [-ch|-column\\_headers](#) for details.

`-nf|-noformat`

Specifies not to use column alignment. See [-nf|-noformat](#) for details.

`-ns|-nosort|-no_sort`

Specifies that the command's output will not be sorted. See [-ns|-nosort](#) for details.

`-sby|-sortby sortspec`

Specifies how to sort the command output. See [-ns|-nosort](#) for details.

---

`-sep|-separator separator`

Used only with the `-f|-format` option. Specifies a different separator character. See [-sep|-separator](#) for details.

`-u|-unnumbered`

Suppresses automatic numbering of the command's output (that is, the output is unnumbered). See [-u|-unnumbered](#) for details.

### **Example**

- List all currently defined processes.

```
ccm process -list
Distributed Process
GUI Retrofit Process
Standard Process
XML Source Team Process
```

### **Related topics**

- [Copying a process to an existing process](#)
- [Copying a process to a new process](#)
- [Deleting a process](#)
- [Modifying a process](#)
- [Showing process rules for a process](#)

---

## Modifying a process

This subcommand modifies the specified processes. You must specify at least one of the options below for the command to succeed.

You must be in the *build\_mgr* or *ccm\_admin* role to use this subcommand.

```
ccm process -m|-modify [(-apr|-add_process_rule|-add_process_rules
    generic_process_rule_spec)...]
    [(-rpr|-remove_process_rule|-remove_process_rules
    generic_process_rule_spec)...]
    [(-pr|-prs|-process_rule|-process_rules
    generic_process_rule_spec)...] [-n|-name name]
    [-diagram|-diagramurl diagram_url_string] process_spec...
```

*-apr|-add\_process\_rule|-add\_process\_rules generic\_process\_rule\_spec...*

Specifies to add the generic process rules to each specified process. This option cannot be used with the *-pr* option, but can be used with the *-rpr* option.

*-diagram|-diagramurl diagram\_url\_string*

Specifies the diagram URL for the new process.

The diagram URL points to a file that contains more information about the process. Default processes have a URL to a descriptive file on the help server. The descriptive file contains a diagram that shows the flow of the baseline and tasks to the project groupings. You can view this file by using the *-show diagramurl* subcommand.

If you don't have the diagram ready, later you can set the URL to the path where the descriptive file will reside.

*generic\_process\_rule\_spec*

Specifies the generic process rules to modify. See [Process rule specification](#) for details.

*-n|-name name*

Specifies the name of the process you want to modify.

*-pr|-prs|-process\_rule|-process\_rules generic\_process\_rule\_spec...*

Specifies an absolute set of generic process rules to set for each process. This option cannot be used with the *-apr* or *-rpr* options.

*process\_spec*

Specifies the process that you want to modify. See [Process specification](#) for details.

---

`-rpr|-remove_process_rule|-remove_process_rules generic_process_rule_spec...`

Specifies the generic process rules to remove from each specified process. This option cannot be used with the `-pr` option, but can be used with the `-apr` option.

### **Example**

- Customize the standard process, replacing the **Integration Testing** generic process rule with a custom **Integration Testing** process rule.

```
ccm process -modify standard -rpr "Integration Testing" -apr "Custom  
Integration Testing"
```

### **Related topics**

- [Listing processes](#)
- [Showing process information](#)
- [Showing a property for a process](#)
- [Showing process rules for a process](#)

---

## Showing process information

This subcommand shows information about the specified processes.

```
ccm process -s|-sh|-show (i|info|information) -f|-format format
                [-nf|-noformat] ([-ch|-column_header] |
                [-nch|-nocolumn_header])
                [-sep|-separator separator] process_spec...
ccm process -show (i|info|information) process_spec...
```

`-ch|-column_header`

Specifies to use a column header in the output format. See [-ch|-column\\_headers](#) for details.

`-f|-format format`

Specifies the command output format. See [-f|-format](#) for details.

A keyword can be built-in (`%fullname`, `%displayname`, `%objectname`) or the name of any existing attribute such as `%modify_time` or `%status`.

See [Built-In keywords](#) for a list of keywords.

`-nch|-nocolumn_header`

Specifies not to use a column header in the output format. See [-ch|-column\\_headers](#) for details.

`-nf|-noformat`

Specifies not to use column alignment. See [-nf|-noformat](#) for details.

*process\_spec*

Specifies the process that you want to show information for. See [Process specification](#) for details.

`-sep|-separator separator`

Used only with the `-f|-format` option. Specifies a different separator character. See [-sep|-separator](#) for details.

## Related topics

- [Listing processes](#)
- [Showing a property for a process](#)

- 
- [Showing process rules for a process](#)

---

## **Showing a property for a process**

This subcommand shows the property of the specified process rules.

```
ccm process -s|-sh|-show (diagram|diagramurl) process_spec...
```

`-diagram|-diagramurl`

Specifies the diagram URL to show.

The diagram URL points to a file that contains more information about the process. Default processes have a URL to a descriptive file on the help server. The descriptive file contains a diagram that shows the flow of the baseline and tasks to the project groupings.

Note that creating a diagram URL is optional; not all processes will have one to view.

`process_spec`

Specifies the process that you want to show properties for. See [Process specification](#) for details.

## **Related topics**

- [Listing processes](#)
- [Showing process information](#)
- [Showing process rules for a process](#)

---

## Showing process rules for a process

This subcommand shows the process rules for the specified process.

```
ccm process -s|-sh|-show (pr|prs|process_rule|process_rules)
    [-f|-format format] [-nf|-noformat]
    ([-ch|-column_header] | [-nch|-nocolumn_header])
    [-sep|-separator separator] ([-sby|-sortby sortspec] |
    [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat]
    [-u|-unnumbered] process_spec...
```

`-ch|-column_header`

Specifies to use a column header in the output format. See [-ch|-column headers](#) for details.

`-f|-format format`

Specifies the command output format. See [-f|-format](#) for details.

A keyword can be built-in (`%fullname`, `%displayname`, `%objectname`) or the name of any existing attribute such as `%modify_time` or `%status`.

See [Built-In keywords](#) for a list of keywords.

`-gby|-groupby groupformat`

Specifies how to group the command output. See [-gby|-groupby](#) for details.

`-nch|-nocolumn_header`

Specifies not to use a column header in the output format. See [-ch|-column headers](#) for details.

`-nf|-noformat`

Specifies not to use column alignment. See [-nf|-noformat](#) for details.

`-ns|-nosort|-no_sort`

Specifies that the command's output will not be sorted. See [-ns|-nosort](#) for details.

`-pr|-prs|-process_rule|-process_rules`

Specifies the process rules to show.



---

*process\_spec*

Specifies the process that you want to show process rules for. See [Process specification](#) for details.

`-sby|-sortby sortspec`

Specifies how to sort the command output. See [-ns|-nosort](#) for details.

`-sep|-separator separator`

Used only with the `-f|-format` option. Specifies a different separator character. See [-sep|-separator](#) for details.

`-u|-unnumbered`

Suppresses automatic numbering of the command's output (that is, the output is unnumbered). See [-u|-unnumbered](#) for details.

## Related topics

- [Listing processes](#)
- [Showing process information](#)
- [Showing a property for a process](#)

---

## Description and uses

A process groups process rules into a named set that are designed to work together.

A process is used to specify the process rules that you can use for a release. The following are the built-in processes, process rules, and purposes:

Process	Process Rule	Purpose
standard	Collaborative Development Custom Development Insulated Development Integration Testing Shared Development System Testing Visible Development	Collaborative Development Custom Development Insulated Development Integration Testing Shared Development System Testing Visible Development
distributed	Custom Development Insulated Development Local Collaborative Development Local Integration Testing Master Integration Testing Shared Development System Testing Visible Development	Custom Development Insulated Development Collaborative Development Integration Testing Master Integration Testing Shared Development System Testing Visible Development

You can create your own process to define a specific way for a team to work. For example, a build manager for a team working on a GUI project creates a process called GUI Process. This process contains a custom purpose called Beta Test with a corresponding process rule called Beta Test. The process rule defines a new level of testing for the beta release. The build manager uses the new Beta Test process rule to perform builds to prepare for the team beta test release, and the process rule can only be found in the new process, GUI Process.

---

## process\_rule command

See [Description and uses](#) for details. The `process_rule` command supports the following subcommands:

- [Adding folders and folder templates to a process rule](#)
- [Comparing process rules](#)
- [Copying a process rule](#)
- [Deleting a process rule](#)
- [Listing process rules](#)
- [Modifying a process rule](#)
- [Removing folders and folder templates from a process rule](#)
- [Setting the controlling database for a process rule](#)
- [Showing baseline projects, folders, folder templates, or members for a process rule](#)
- [Showing a property for a process rule](#)
- [Showing process rule information](#)

---

## ***Adding folders and folder templates to a process rule***

This subcommand adds folders or folder templates and folder and folder templates to the specified process rules. You may only add folder templates to a generic process rule. You may add folders or folder templates to release specific process rules. You must be working as a [Process rules manager](#) to use this command.

```
ccm pr|process_rule|ut|update_temp|update_template|rt|recon_temp|
    reconfigure_template -a|-ad|-add
    [(-fol|-folder|-folders folder_spec)...] [(-ft|-folder_temp|
    -folder_temps|-folder_template|-folder_templates
    folder_template_spec)...] process_rule_spec...
```

`-fol|-folder|-folders folder_spec`

Specifies the folders to be added to each process rule. Generic process rules may only have folder templates. See [Folder specification](#) for details.

`-ft|-folder_temp|-folder_temps|-folder_template|-folder_templates  
folder_template_spec`

Specifies the folder templates to be added to each process rule. See [Folder specification](#) for details.

## ***Examples***

- Add folder 3 to the 2.1:Insulated Development process rule:  

```
ccm pr -add -folders 3 "2.1:Insulated Development"
```
- Add folder template integration tested tasks for release %release to the 2.1:Insulated Development process rule.  

```
ccm pr -add -folder_temp "integration tested tasks for release  
%release" "2.1:Insulated Development"
```

## **Related topics**

- [Comparing process rules](#)
- [Showing baseline projects, folders, folder templates, or members for a process rule](#)
- [Showing a property for a process rule](#)
- [Showing process rule information](#)

---

## Comparing process rules

This subcommand compares two process rules.

```
ccm pr|process_rule|ut|update_temp|update_template|
    rt|recon_temp|reconfigure_template
    -comp|-compare process_rule_spec1 process_rule_spec2
```

```
-comp|-compare process_rule_spec1 process_rule_spec2
```

Specifies the process rules to be compared. See [Process rule specification](#) for details.

### Example

- Compare the process rules for the `toolkit/2.0:Collaborative Development` project and the `toolkit/2.0:Insulated Development` project.

```
ccm process_rule -compare "toolkit/2.0:Collaborative Development"
"toolkit/2.0:Insulated Development"
```

### Related topics

- [Adding folders and folder templates to a process rule](#)
- [Copying a process rule](#)
- [Deleting a process rule](#)
- [Showing baseline projects, folders, folder templates, or members for a process rule](#)
- [Showing a property for a process rule](#)
- [Showing process rule information](#)

---

## Copying a process rule

This subcommand copies a process rule to another process rule. You must be working as a [Process rules manager](#) to use this command. The following rules apply:

- **Generic to generic copies**  
If a generic process rule is copied to an existing generic process rule, the target process rule keeps the old name (the four-part name and the `case_preserved_name` attribute), but all other properties are copied from the source process rule. You can copy a generic process rule to a new generic process rule.
- **Generic to release-specific copies**  
If a generic process rule is copied to an existing release-specific process rule, the target process rule keeps the old name (the four-part name, the `case_preserved_name` attribute, and the release attribute) and its old association to a generic process rule. All other properties are copied from the source process rule.
- **Release-specific to release-specific copies**  
If a release-specific process rule is copied to an existing release-specific process rule, the target process rule keeps its old name (the four-part name, the `case_preserved_name` attribute, and the release attribute), but all other properties are copied from the source process rule. The target release-specific process rule also keeps its existing association with a generic process rule.
- **Release specific to generic copies**  
You cannot copy a release-specific process rule to a generic process rule.

```
ccm pr|process_rule|ut|update_temp|update_template|rt|recon_temp|
reconfigure_template -cp|-copy -baseline_rules_only
from_rspr_spec to_rspr_spec...
ccm pr|process_rule|ut|update_temp|update_template|rt|recon_temp|
reconfigure_template -cp|-copy from_process_rule_spec
to_process_rule_spec
```

`-baseline_rules_only`

Copies the baseline rules for the first process rule to the second process rule. The baseline rules include the baseline selection mode, baseline search list of `release:purpose` pairs, any specific baseline defined on the process rule, and any project version matching string.

The process rules you specify must be release-specific process rules.

`from_process_rule_spec`

Specifies the process rule to copy. See [Process rule specification](#) for details.

---

*to\_process\_rule\_spec*

Specifies the process rule to be updated. See "Process rule specification" for details.

*from\_rspr\_spec*

Specifies the release-specific process rule to copy. See "Process rule specification" for details.

*to\_rspr\_spec*

Specifies the release-specific process rule to be updated. See "Process rule specification" for details.

### **Example**

- Copy the **2.0:Insulated Development** process rule over the existing **2.1:Insulated Development** process rule.

```
ccm process_rule -copy "2.0:Insulated Development" "2.1:Insulated  
Development"
```

### **Related topics**

- [Comparing process rules](#)
- [Showing baseline projects, folders, folder templates, or members for a process rule](#)
- [Showing a property for a process rule](#)
- [Showing process rule information](#)



---

## ***Deleting a process rule***

This subcommand deletes a process rule. By default, a process rule may only be deleted when it is not used by any project grouping. Use the `-force` option to delete a process rule that is in use. You must be working as a [Process rules manager](#) to use this command.

```
ccm pr|process_rule|ut|update_temp|update_template|rt|recon_temp|
    reconfigure_template -d|-delete [-force] process_rule_spec...
```

`-d|-delete`

The command accepts one or more `process_rule_spec` arguments. You can set each to multiple objects.

`-force`

Deletes the process rule even if it is used by a project grouping. If you omit the `-force` command, the process rule is only deleted if it is not used.

`process_rule_spec`

Specifies the process rules delete. See [Process rule specification](#) for details.

## ***Example***

- Delete the **2.1:Shared** process rule.

```
ccm pr -delete "2.1:Shared"
```

## **Related topics**

- [Comparing process rules](#)
- [Showing baseline projects, folders, folder templates, or members for a process rule](#)
- [Showing a property for a process rule](#)
- [Showing process rule information](#)

---

## ***Listing process rules***

This subcommand lists the process rules matching the specified criteria. If no options are specified, it lists all process rules.

```
ccm pr|process_rule|ut|update_temp|update_template|rt|recon_temp|
    reconfigure_template -l|-list [-f|-format format] [-nf|-noformat]
    ([-ch|-column_header] | [-nch|-nocolumn_header])
    [-sep|-separator separator] ([-sby|-sortby sortspec] |
    [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat] [-u|-unnumbered]
```

`-ch|-column_header`

Specifies to use a column header in the output format. See [-ch|-column\\_headers](#) for details.

`-f|-format format`

Specifies the command output format. See [-f|-format](#) for details.

A keyword can be built-in (`%fullname`, `%displayname`, `%objectname`) or the name of any existing attribute such as `%modify_time` or `%status`.

See [Built-In keywords](#) for a list of keywords.

`-gby|-groupby groupformat`

Specifies how to group the command output. See [-gby|-groupby](#) for details.

`-nch|-nocolumn_header`

Specifies not to use a column header in the output format. See [-ch|-column\\_headers](#) for details.

`-nf|-noformat`

Specifies not to use column alignment. See [-nf|-noformat](#) for details.

`-ns|-nosort|-no_sort`

Specifies that the command's output will not be sorted. See [-ns|-nosort](#) for details.

`-sby|-sortby sortspec`

Specifies how to sort the command output. See [-ns|-nosort](#) for details.

---

`-sep|-separator separator`

Used only with the `-f|-format` option. Specifies a different separator character. See [-sep|-separator](#) for details.

`-u|-unnumbered`

Suppresses automatic numbering of the command's output (that is, the output is unnumbered). See [-u|-unnumbered](#) for details.

## Examples

- List all currently defined process rules.

```
ccm process_rule -list
Collaborative Development
Insulated Development
Custom Development
Shared Development
Visible Development
Integration Testing
System Testing
1.0:Integration Testing
1.0:System Testing
1.0:Insulated Development
2.0:Integration Testing
2.0:System Testing
2.0:Collaborative Development
2.0:Insulated Development
Local Collaborative Development
Local Integration Testing
Master Integration Testing
```

- List the process rules for release **1.0**:

```
ccm pr -list -rel 1.0
1) 1.0:Collaborative Development
2) 1.0:Custom Development
3) 1.0:Insulated Development
4) 1.0:Integration Testing
5) 1.0:Shared Development
6) 1.0:System Testing
7) 1.0:Visible Development
```

- 
- List the process rules for **Integration Testing**:

```
ccm pr -list -purp "Integration Testing"
```

- 1) 1.0:Integration Testing
- 2) Integration Testing
- 3) Local Integration Testing
- 4) a/1.0:Integration Testing
- 5) m/1.0:Integration Testing

- List the process rules for **System Testing** and **Integration Testing**:

```
ccm pr -l -rel 1.0 -rel a/1.0 -purpose "System Testing" -purpose  
"Integration Testing"
```

- 1) 1.0:Integration Testing
- 2) 1.0:System Testing
- 3) a/1.0:Integration Testing
- 4) a/1.0:System Testing

## Related topics

- [Comparing process rules](#)
- [Deleting a process rule](#)
- [Modifying a process rule](#)
- [Showing baseline projects, folders, folder templates, or members for a process rule](#)
- [Showing a property for a process rule](#)
- [Showing process rule information](#)

---

## Modifying a process rule

This subcommand modifies the specified process rules. You must be working as a [Process rules manager](#) to use this command.

```
ccm pr|process_rule|ut|update_temp|update_template|rt|recon_temp|
    reconfigure_template -m|-modify
    [(-fol|-folder|-folders folder_spec)...] [(-ft|-folder_temp|
    -folder_temps|-folder_template|-folder_templates
    folder_template_spec)...] [-bn|-baseline_name baseline_spec]
    [-lb|-latest_baseline] [-usb|-user_selected_baseline]
    [-lbp|-latest_baseline_projects] [-lsp|-latest_static_projects]
    [-lsbmp|-latest_static_or_build_management_projects]
    [-brp|-baseline_release_purpose|-baseline_release_purposes
    release_purposes ( [-pr|-prepend] | [-ap|-append] )]
    [-pb|-prep_baseline] [-nopb|-noprep_baseline]
    [-matching version_matching_string] process_rule_spec...
```

-ap|-append

When used with the -brp|-baseline\_release\_purpose|-baseline\_release\_purposes option, specifies that the releases should be appended to the current release-purpose pair list.

-brp|-baseline\_release\_purpose|-baseline\_release\_purposes *release\_purposes*

Specifies the baseline release and purpose pairs for the process rule. The baseline release purpose list is used when the process rule uses the latest baseline selection mode. The order of the list is important. In latest baseline search mode, update looks for baselines matching the first release and purpose. If none are found, it looks for baselines matching the second release and purpose.

The *release\_purposes* value is a list of one or more items each of which is a *release\_spec*, a colon (:), and a purpose name. You can set the *release\_spec* to a single release, or be the keyword %release or %baseline\_release. The %release keyword means the current release for that process rule. The %baseline\_release keyword means the baseline release of the process rule's release. The purpose name must be a defined purpose.

If -ap|-append is specified, the specified release-purpose pairs are appending to the current list. If -pr|-prepend is specified, the specified release-purpose pairs are prepended to the current list. If neither option is specified, the specified release-purposes replace the current list.

-bn|-baseline\_name *baseline\_spec*

Specifies that the process rule uses the selection mode for the specified baseline. You can set *baseline\_spec* to a single baseline.

---

`-fol|-folder|-folders folder_spec`

Specifies the folders to be removed from each process rule. Generic process rules may only have folder templates.

`-ft|-folder_temp|-folder_temps|-folder_template|-folder_templates  
folder_template_spec`

Specifies the folder templates to be removed from each process rule.

`-lb|-latest_baseline`

Specifies that the process rule should use the latest baseline. When a project grouping that uses this process rule is updated, the latest baseline is found that matches the specified baseline release-purpose pair list.

`-lbp|-latest_baseline_projects`

Specifies that the process rule should use the latest baseline project. When a project grouping uses this process rule and a project is updated, the latest project matching the version matching and prep baseline criteria is selected as the baseline project.

`-lsp|-latest_static_projects`

Specifies that the process rule should use the latest static projects. This option cannot be used with the `-pb|-prep_baseline` option.

`-lsbmp|-latest_static_or_build_management_projects`

Specifies that the process rule should use the latest static or build management projects. This option cannot be used with the `-nopb|-noprep_baseline` option.

`-matching version_matching_string`

When the process rule uses a selection mode of latest baseline projects, this specifies any additional criteria to match against the versions of candidate baseline projects.

You can enter a version that can be used to identify the baseline. Use this field if specifying the release of the baseline is insufficient because you have more than one release version of a project with the same release value.

For example, if a company has three released project hierarchies, all for release 1.0: the project versions are **1.0\_alpha**, **1.0\_beta**, and **1.0\_gr**. In this case, specifying the Baseline Release option as **1.0** is not enough to identify projects that use this process

---

rule. You would set the Baseline Versions Matching option to **1.0\_gr** to indicate that the project with a version of **1.0\_gr** should be used as the baseline.

If all baselines in the **1.0\_gr** project hierarchy do not have identical versions, but their versions are similar, you can specify a wildcard. For example, if your project hierarchy contains projects with versions **1.0\_gr**, **1.0\_gr\_unix**, and **1.0\_gr\_windows**, you could set the Baseline Versions Matching option to `1.0_gr*`. This setting would select the version with the prefix **1.0\_gr**, even though the remainder of the version might differ. (If a project has more than one choice for a baseline, it will select the baseline whose platform matches. For example, project **2.0\_int\_unix** might identify **1.0\_gr\_unix** and **1.0\_gr\_windows** as potential baselines, but it will check for a matching platform, then use **1.0\_gr\_unix**. This is because Rational Synergy is set up to support development of parallel platforms by default.)

`-modify`

Specifies to modify the properties of an existing process rule. This subcommand accepts one or more [Process rule specification](#) arguments. You can set each to multiple objects. It accepts the `-bn|-baseline_name` option, which accepts a [Baseline specification](#) that you can set to a single baseline object. The results of this option do not update the query selection set.

`-nopb|-noprep_baseline`

Valid only when the process rule has a baseline selection mode of `latest_baseline_projects`. It indicates that *prep* state projects are not to be considered as potential baseline projects for individual projects that use this process rule. Use the `-lsp|-latest_static_projects` option instead.

`-pb|-prep_baseline`

Valid only when the process rule has a baseline selection mode of `latest_baseline_projects`. It indicates that prep state projects should be considered as potential baseline projects for individual projects that use this process rule. Use the `-lsbmp|-latest_static_or_build_management_projects` option instead.

*process\_rule\_spec*

Specifies the process rules to modify. See [Process rule specification](#) for details.

`pr|-prepend`

When used with the `-brp|-baseline_release_purpose|-baseline_release_purposes` option, the specified release-purpose pairs are prepended to the current list.

---

`-usb|-user_selected_baseline`

Specifies that the process rule does not specify a baseline that is to be used to find baseline projects. The baseline is selected by the user.

### Examples

- Set the 2.1:Insulated Development process rule to use the latest baseline.  

```
ccm pr -m "2.1:Insulated Development" -latest_baseline
```
- Set the 2.1:Insulated Development process rule to use the latest baseline projects with the specified release and purpose combinations.  

```
ccm pr -m "2.1:Insulated Development" -latest_baseline_projects -  
baseline_release_purpose "2.1:Integration Testing,2.1:System  
Testing,2.0:Any"
```
- Modify the list of release/purpose pairs that are used by a specific process rule to search for a baseline.  

```
ccm pr -modify -baseline_release_purposes "2.0:Any,1.0:System Testing"  
-prepend "2.0:Integration Testing"
```
- Select a baseline named `Build_1234_int` for a process rule whose `process_rule_spec` is `2.0:Insulated Development`.  

```
ccm process_rule -modify -bn Build_1234_int "2.0:Insulated  
Development"
```

### Related topics

- [Comparing process rules](#)
- [Deleting a process rule](#)
- [Showing baseline projects, folders, folder templates, or members for a process rule](#)
- [Showing a property for a process rule](#)
- [Showing process rule information](#)



---

## Removing folders and folder templates from a process rule

This subcommand removes folders or folder templates and folder and folder templates from the specified process rules. Generic process rules can only have folder templates. You must be working as a [Process rules manager](#) to use this command.

```
ccm pr|process_rule|ut|update_temp|update_template|rt|recon_temp|
  reconfigure_template -rem|-remove
  [(-fol|-folder|-folders folder_spec)...] [(-ft|-folder_temp|
  -folder_temps|-folder_template|-folder_templates
  folder_template_spec)...] process_rule_spec...
```

*-fol|-folder|-folders folder\_spec*

Specifies the folders to remove from each process rule. Generic process rules may only have folder templates.

*-ft|-folder\_temp|-folder\_temps|-folder\_template|-folder\_templates
folder\_template\_spec*

Specifies the folder templates to remove from each process rule.

*process\_rule\_spec*

Specifies the process rules to update. See [Process rule specification](#) for details.

## Related topics

- [Comparing process rules](#)
- [Showing baseline projects, folders, folder templates, or members for a process rule](#)
- [Showing a property for a process rule](#)
- [Showing process rule information](#)

---

## Setting the controlling database for a process rule

This subcommand sets DCM to either hand over to a specified database, accept updates only from a specified database, or take local control.

```
ccm pr|process_rule|ut|update_temp|update_template|rt|recon_temp|
    reconfigure_template -cdb|-controlling_database -local
    process_rule_spec...
ccm pr|process_rule|ut|update_temp|update_template|rt|recon_temp|
    reconfigure_template -cdb|-controlling_database
    -handover database_spec process_rule_spec...
ccm pr|process_rule|ut|update_temp|update_template|rt|recon_temp|
    reconfigure_template -cdb|-controlling_database -accept database_spec
    process_rule_spec...
```

`-accept database_spec`

Specifies to accept DCM updates from a specified database. You can set the `database_spec` to a single database definition. See [Database specification](#) for details about using `database_spec`. See "Replication of generic and release-specific processes" in the *Rational Synergy Distributed* book for details about accepting DCM updates.

`-handover database_spec`

Specifies that control of the object is handed over from the current database to the specified database. The default value when creating a DCM database definition is a blank string. This means that when you hand over control to a spoke via a hub database, you must specify the hub `database_spec` for the `database_spec` value. The specified `database_spec` must be either a known DCM database definition for which a generate is permitted, or a blank string. A blank string means that control can't be handed over to that database.

A detailed description of this option is in [Rational Synergy Distributed](#), in the "Controlling database and handover of control" section.

`-local`

Specifies to take local control of the specified process rules.

### Example

- Set the controlling database to use the `2.1-patch1:Insulated Development` process rule.

```
ccm pr -controlling_database -accept A "2.1-patch1:Insulated
Development"
```

---

## Showing baseline projects, folders, folder templates, or members for a process rule

This subcommand shows the baseline projects, folders, folder templates, or member objects for the specified process rules. Any user may use this command.

```
ccm pr|process_rule|ut|update_temp|update_template|rt|recon_temp|
    reconfigure_template -s|-sh|-show (baseline_projects |
    (fol|folder|folders) |
    (ft|folder_temp|folder_temps|folder_template|folder_templates) |
    members) [-f|-format format] [-nf|-noformat]
    ([-ch|-column_header] | [-nch|-nocolumn_header])
    [-sep|-separator separator] ([-sby|-sortby sortspec] |
    [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat] [-u|-unnumbered]
    process_rule_spec...
```

[-ch|-column\\_header](#)

[-f|-format \*format\*](#)

[-gby|-groupby \*groupformat\*](#)

[-nch|-nocolumn\\_header](#)

[-ns|-nosort|-no\\_sort](#)

[-sby|-sortby \*sortspec\*](#)

[-sep|-separator \*separator\*](#)

```
s|-sh|-show
(baseline_projects|(fol|folder|folders)|(ft|folder_temp|folder_temps|folder_
template|folder_templates)|members)
```

Specifies what related objects for the process rule to show:

- `baseline_projects`: Shows the baseline projects for the process rule.
- `fol|folder|folders`: Shows the folder members of the process rule.
- `ft|folder_temp|folder_temps|folder_template|folder_templates`: Shows the folder template members of the process rule.
- `members`: Shows both folder and folder template members of the process rule.

---

[-u|unnumbered](#)

### **Related topics**

- [Comparing process rules](#)
- [Deleting a process rule](#)
- [Showing a property for a process rule](#)
- [Showing process rule information](#)

---

## Showing a property for a process rule

This subcommand shows the property of the specified process rules. Any user may use this command.

```
ccm pr|process_rule|ut|update_temp|update_template|rt|recon_temp|
    reconfigure_template -s|-sh|-show
    ((brp|baseline_release_purpose|baseline_release_purposes) |
    (bsm|baseline_selection_mode) | matching | (pb|prep_baseline))
    process_rule_spec...
```

-s|-sh|-show (brp|baseline\_release\_purpose|baseline\_release\_purposes)  
Shows the baseline release purpose list.

-s|-sh|-show (bsm|baseline\_selection\_mode)  
Shows the baseline selection mode.

-s|-sh|-show matching  
Shows the versions matching property.

-s|-sh|-show pb|prep\_baseline  
Shows whether prep baseline projects are eligible.

## Related topics

- [Comparing process rules](#)
- [Deleting a process rule](#)
- [Showing baseline projects, folders, folder templates, or members for a process rule](#)
- [Showing process rule information](#)

---

## Showing process rule information

This subcommand shows information about the specified process rules. Any user may use this command.

```
ccm pr|process_rule|ut|update_temp|update_template|rt|recon_temp|
  reconfigure_template -s|-sh|-show (i|info|information)
  -f|-format format [-nf|-noformat]
  ([-ch|-column_header] | [-nch|-nocolumn_header])
  [-sep|-separator separator] process_rule_spec...
ccm pr|process_rule|ut|update_temp|update_template|rt|recon_temp|
  reconfigure_template -s|-sh|-show (i|info|information)
  process_rule_spec...
```

[-ch|-column header](#)

[-f|-format \*format\*](#)

[-gby|-groupby \*groupformat\*](#)

[-nch|-nocolumn header](#)

[-ns|-nosort|-no sort](#)

[-sby|-sortby \*sortspec\*](#)

[-sep|-separator \*separator\*](#)

[-u|-unnumbered](#)

### Example

- Show properties of the 2.1:Insulated Development process rule.

```
ccm process_rule -show info "2.1:Insulated Development"
```

### Related topics

- [Comparing process rules](#)
- [Deleting a process rule](#)
- [Showing baseline projects, folders, folder templates, or members for a process rule](#)
- [Showing a property for a process rule](#)

---

## Description and uses

The `process_rule` command displays and sets process rules. Note that the `process_rule` command was referred to as the `update_template` and `reconfigure_template` commands in prior releases.

A process rule specifies how a baseline is chosen for a project grouping's update properties and for a project's update properties. Process rules support the following selection modes:

- **Latest baseline**

The latest baseline matching the process rule's baseline release-purpose pair list is chosen as the baseline for the project grouping. For a project, the corresponding project in that baseline is used as the baseline project.

- **Specific baseline**

The baseline specified for the process rule is used for the project grouping. For a project, the corresponding project in that baseline is used as the baseline project.

- **User selected baseline**

A process rule does not specify a baseline. You must select a baseline for the project grouping's update properties manually. For a project, the corresponding project in that baseline is used as the baseline project.

- **Latest baseline projects**

A baseline project is chosen by getting the latest baseline projects that match the release or baseline release, and that match any specified version matching criteria on the project version.

A process rule specifies how a project is updated when an update operation is performed on the project. The combination of a project purpose and release value determines which process rule can be used in the project. Multiple process rules can be created for a release/purpose pair. You can set up rules, and then select rules to apply to a given release and purpose, and switch among the set of process rules during the course of a release. You can also reuse process rules for future releases, rather than continually modifying the process rule for each purpose.

Process rules are automatically created whenever one of the following occurs.

- When a builder manager creates a new release with a specified process, a release-specific process rule is created for each generic process rule. If the build manager creates a release with generic process rules, a release specific process rule is created for each generic process rule.
- When a build manager adds a new purpose to the list of valid purposes for a particular release value, a process rule is created for that unique combination of project purpose and release value.

- 
- When a build manager adds a generic process rule to a release, a release specific process rule is created from it.
  - When a build manager creates a new purpose, a general process rule is created for that purpose. The build manager must then edit this new (empty) process rule.
  - When a build manager copies a process rule.
  - When a build manager copies a generic process rule using a new generic process rule name.

General process rules are shipped with the product for both standard and distributed processes. A non-DCM-initialized database contains the standard process, and a DCM-initialized database contains both the standard and the distributed processes. The process rules have the same behavior in each, with the following exceptions:

- In the standard process, Collaborative Development collects all completed tasks from all databases, while in the distributed process, Local Collaborative Development collects all completed tasks from the local database.
- In the standard process, Integration Testing collects all completed tasks from all databases, while in the distributed process, Local Integration Testing collects all completed tasks from the local database, and master integration tested tasks from foreign databases.

The standard process is used to provide process rules in Rational Synergy Classic and the CLI when a purpose is specified.

You can specify which process rules to use when you create a new release. For more information, see the [release command](#).



---

## project Command

See [Description and Uses](#) for details. The `project` command supports the [Displaying project information](#) subcommand.

---

## Displaying project information

```
ccm project [-f|-format format] [-nf|-noformat] ([-ch|-column_header] |  
[-nch|-nocolumn_header]) [-sep|-separator separator]  
([-sby|-sortby sortspec] | [-ns|-nosort|-no_sort])  
[-gby|-groupby groupformat]  
ccm project [-f|-format format] [-nf|-noformat] ([-ch|-column_header] |  
[-nch|-nocolumn_header]) [-sep|-separator separator]  
([-sby|-sortby sortspec] | [-ns|-nosort|-no_sort])  
[-gby|-groupby groupformat] file_spec
```

`-ch|-column_header`

Specifies to use a column header in the output format. See [-ch|-column headers](#) for details.

*file\_spec*

Specifies an object whose parent directory will be used to determine an associated project. If omitted, the current working directory is the default. See [File specification](#) for detailed information.

`-f|-format format`

Specifies the command output format. See [-f|-format](#) for details.

`-gby|-groupby groupformat`

Specifies how to group the command output. See [-gby|-groupby](#) for details.

`-nch|-nocolumn_header`

Specifies not to use a column header in the output format. See [-nch|-nocolumn headers](#) for details.

`-nf|-noformat`

Specifies not to use column alignment. See [-nf|-noformat](#) for details.

`-ns|-no_sort`

Specifies that the command's output will not be sorted. See [-ns|-nosort](#) for details.

`-sep|-separator separator`

Specifies a different separator character. See [-sep|-separator](#) for details.

---

`-sby|-sortby sortspec`

Specifies how to sort the command output. See [-sby|-sortby](#) for details.

`-u|-unnumbered`

Suppresses automatic numbering of the command's output (that is, the output is unnumbered). See [-u|-unnumbered](#) for details.

The default format is `%displayname`. This subcommand does not update the query selection set.

### **Example**

- Determine the project associated with the `$HOME/ccm_wa/database/example-1/example/doc/readme.txt` work area path.

```
ccm project $HOME/ccm_wa/database/example-1/example/doc/readme.txt
example-1
```

### **Related topics**

- [Formatting usage examples](#)

---

## Description and Uses

The `project` command determines the project associated with a specified `file_spec` or the current working directory. The `file_spec` is typically a [Work area reference form](#) within a maintained work area.

## project\_grouping command

See [Description and uses](#) for details. The `project_grouping` command supports the following subcommands:

- [Adding tasks to the update properties for a project grouping](#)
- [Comparing projects for a project grouping with projects for a baseline](#)
- [Comparing projects for two project groupings](#)
- [Comparing tasks for a project grouping with tasks for a baseline](#)
- [Comparing tasks for project groupings](#)
- [Copying tasks from one project grouping to another](#)
- [Deleting a project grouping and members](#)
- [Listing project groupings](#)
- [Previewing update baseline and tasks for a project grouping](#)
- [Updating the baseline and tasks for a project grouping](#)
- [Removing tasks from the update properties for a project grouping](#)
- [Setting the auto-update mode for a project grouping](#)
- [Showing a project grouping property](#)
- [Showing the associated projects, baseline, tasks, or objects for a project grouping](#)
- [Showing project grouping information](#)

---

## ***Adding tasks to the update properties for a project grouping***

This subcommand adds one or more specified tasks to the specified project groupings, or adds all previously removed tasks back into each specified project grouping. Any user who can modify the specified project groupings can perform this command.

```
ccm pg|project_grouping
    (-at|-add_task|-add_tasks (task_spec|all_removed))...
    project_grouping_spec...
```

```
(-at|-add_task|-add_tasks (task_spec|all_removed))...
```

Specifies the tasks to be added to the specified project groupings. If a task to be added is in the list of removed tasks, it is removed from that list. Otherwise, it is added to the list of manually added tasks for the project grouping. The keyword `all_removed` means add back in all the tasks that are in the list of removed tasks. See [Task specification](#) for details.

*project\_grouping\_spec1*

Specifies the project groupings to which tasks will be added. See [Project grouping specification](#) for details.

*project\_grouping\_spec2*

Specifies the project groupings to which tasks will be added. See [Project grouping specification](#) for details.

## ***Examples***

- Add tasks to a project grouping:

```
ccm pg -at G#123 "All A/1.0 Integration Testing Projects from Database G"
```

- Remove tasks from a project grouping:

```
ccm pg -remove_task all "All A/1.0 Integration Testing Projects from Database G"
```

---

## Comparing projects for a project grouping with projects for a baseline

This subcommand compares projects for a project grouping with projects for a specified baseline.

You must specify `-union`, `-intersection`, or `-not_in`.

```
ccm pg|project_grouping -compare -baseline -projects
    ([-int|-intersection] | [-not|-not_in] | [-un|-union])
    [-f|-format format] [-nf|-noformat]
    ([-ch|-column_header] | [-nch|-nocolumn_header])
    [-sep|-separator separator] ([-sby|-sortby sortspec] |
    [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat]
    [-u|-unnumbered] project_grouping_spec baseline_spec
```

*baseline\_spec*

Specifies the baseline to be compared. For more information, see [Baseline specification](#).

`-ch|-column_header`

Specifies to use a column header in the output format. See [-ch|-column headers](#) for details.

`-f|-format format`

Specifies the command output format. See [-f|-format](#) for details.

`-gby|-groupby groupformat`

Specifies how to group the command output. See [-gby|-groupby](#) for details.

`-int|-intersection`

Specifies the comparison to show the projects in common between the project grouping and the baseline.

`-nch|-nocolumn_header`

Specifies not to use a column header in the output format. See [-ch|-column headers](#) for details.

`-nf|-noformat`

Specifies not to use column alignment. See [-nf|-noformat](#) for details.



---

`-ns|-no_sort`

Specifies that the command's output will not be sorted. See [-ns|-nosort](#) for details.

`-not|-not_in`

Specifies the comparison to show the projects in the project grouping that are not in baseline.

*project\_grouping\_spec*

Specifies the project grouping to be compared. For more information, see [Project grouping specification](#).

`-sep|-separator separator`

Specifies a different separator character. See [-sep|-separator](#) for details.

`-sby|-sortby sortspec`

Specifies how to sort the command output. See [-sby|-sortby](#) for details.

`-un|-union`

Specifies the comparison to show both of the projects in the project grouping and the baseline.

`-u|-unnumbered`

Suppresses automatic numbering of the command's output (that is, the output is unnumbered). See [-u|-unnumbered](#) for details.

### **Example**

- Compare the projects in common between the project grouping **My 2.0 Collaborative Development projects** and the baseline **1.0 build 123**.

```
ccm pg -compare -baseline -project -intersection "My 2.0 Collaborative  
Development projects" "1.0 build 123"
```

### **Related topics**

- [Comparing projects for two project groupings](#)
- [Comparing tasks for a project grouping with tasks for a baseline](#)
- [Comparing tasks for project groupings](#)
- [Listing project groupings](#)

- 
- [Showing a project grouping property](#)
  - [Showing project grouping information](#)

---

## Comparing projects for two project groupings

This subcommand compares projects in two project groupings.

You must specify `-union`, `-intersection`, or `-not_in`.

```
ccm pg|project_grouping -compare -projects
    ([-int|-intersection] | [-not|-not_in] | [-un|-union])
    [-f|-format format] [-nf|-noformat]
    ([-ch|-column_header] | [-nch|-nocolumn_header])
    [-sep|-separator separator] ([-sby|-sortby sortspec] |
    [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat]
    [-u|-unnumbered] project_grouping_spec1 project_grouping_spec2
```

`-ch|-column_header`

Specifies to use a column header in the output format. See [-ch|-column\\_headers](#) for details.

`-f|-format format`

Specifies the command output format. See [-f|-format](#) for details.

`-gby|-groupby groupformat`

Specifies how to group the command output. See [-gby|-groupby](#) for details.

`-int|-intersection`

Specifies the comparison to show the projects in common between both project groupings.

`-nch|-nocolumn_header`

Specifies not to use a column header in the output format. See [-ch|-column\\_headers](#) for details.

`-nf|-noformat`

Specifies not to use column alignment. See [-nf|-noformat](#) for details.

`-ns|-no_sort`

Specifies that the command's output will not be sorted. See [-ns|-nosort](#) for details.

---

`-not|-not_in`

Specifies the comparison to show the projects in the first project grouping that are not in the second project grouping.

`project_grouping_spec1`

Specifies the first project grouping to be compared. For more information, see [Project grouping specification](#).

`project_grouping_spec2`

Specifies the project grouping to compare to `project_grouping_spec1`. For more information, see [Project grouping specification](#).

`-sep|-separator separator`

Specifies a different separator character. See [-sep|-separator](#) for details.

`-sby|-sortby sortspec`

Specifies how to sort the command output. See [-sby|-sortby](#) for details.

`-un|-union`

Specifies the comparison to show both the projects that are in both project groupings.

`-u|-unnumbered`

Suppresses automatic numbering of the command's output (that is, the output is unnumbered). See [-u|-unnumbered](#) for details.

### **Example**

- Show the union of projects in project groupings **My 2.0 Collaborative Development projects** and **My 2.0 Insulated Development projects**.

```
ccm pg -compare -project -union "My 2.0 Collaborative Development projects" "My 2.0 Insulated Development projects"
```

### **Related topics**

- [Comparing projects for a project grouping with projects for a baseline](#)
- [Comparing tasks for a project grouping with tasks for a baseline](#)
- [Comparing tasks for project groupings](#)
- [Showing a project grouping property](#)

- 
- [Showing project grouping information](#)

---

## Comparing tasks for a project grouping with tasks for a baseline

This subcommand compares all tasks belonging to a project grouping with all tasks belonging to a baseline.

You must specify `-union`, `-intersection`, or `-not_in`.

```
ccm pg|project_grouping -compare -baseline -all_tasks
    ([-int|-intersection] | [-not|-not_in] | [-un|-union])
    [-f|-format format] [-nf|-noformat]
    ([-ch|-column_header] | [-nch|-nocolumn_header])
    [-sep|-separator separator] ([-sby|-sortby sortspec] |
    [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat]
    [-u|-unnumbered] project_grouping_spec baseline_spec
```

*baseline\_spec*

Specifies the baseline to be compared. For more information, see [Baseline specification](#).

`-ch|-column_header`

Specifies to use a column header in the output format. See [-ch|-column headers](#) for details.

`-f|-format format`

Specifies the command output format. See [-f|-format](#) for details.

`-gby|-groupby groupformat`

Specifies how to group the command output. See [-gby|-groupby](#) for details.

`-int|-intersection`

Specifies the comparison to show the tasks in common between the saved and added tasks for the project grouping and the tasks in the baseline.

`-nch|-nocolumn_header`

Specifies not to use a column header in the output format. See [-ch|-column headers](#) for details.

`-nf|-noformat`

Specifies not to use column alignment. See [-nf|-noformat](#) for details.

---

`-ns|-no_sort`

Specifies that the command's output will not be sorted. See [-ns|-nosort](#) for details.

`-not|-not_in`

Specifies the comparison to show the saved and added tasks in the project grouping that are not in the baseline.

*project\_grouping\_spec*

Specifies the project grouping to be compared. For more information, see [Project grouping specification](#).

`-sep|-separator separator`

Specifies a different separator character. See [-sep|-separator](#) for details.

`-sby|-sortby sortspec`

Specifies how to sort the command output. See [-sby|-sortby](#) for details.

`-un|-union`

Specifies the comparison to show both the saved and added tasks that are in the project grouping and the tasks that are in the baseline.

`-u|-unnumbered`

Suppresses automatic numbering of the command's output (that is, the output is unnumbered). See [-u|-unnumbered](#) for details.

### **Example**

- Show the saved and added tasks in the project grouping **My 2.0 Collaborative Development projects** but are not in the baseline **1.0 build 123**.

```
ccm pg -compare -tasks -not_in -baseline "My 2.0 Collaborative  
Development projects" "1.0 build 123"
```

### **Related topics**

- [Comparing projects for a project grouping with projects for a baseline](#)
- [Comparing projects for two project groupings](#)
- [Comparing tasks for project groupings](#)
- [Showing a project grouping property](#)

- 
- [Showing project grouping information](#)



---

## Comparing tasks for project groupings

This subcommand compares all tasks for two specified project groupings.

You must specify `-union`, `-intersection`, or `-not_in`.

```
ccm pg|project_grouping -compare ([-at|-added_tasks] |  
  [-rt|-removed_tasks] | [-all_tasks] |  
  [-tob|-tasks_on_top_of_baseline])  
  ([-int|-intersection] | [-not|-not_in] | [-un|-union])  
  [-f|-format format] [-nf|-noformat]  
  ([-ch|-column_header] | [-nch|-nocolumn_header])  
  [-sep|-separator separator] ([-sby|-sortby sortspec] |  
  [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat]  
  [-u|-unnumbered] project_grouping_spec1 project_grouping_spec2
```

`-at|-added_tasks`

Compares the tasks added from the two project groupings.

`-all_tasks`

Compares the added and saved tasks from the two project groupings.

`-ch|-column_header`

Specifies to use a column header in the output format. See [-ch|-column headers](#) for details.

`-f|-format format`

Specifies the command output format. See [-f|-format](#) for details.

`-gby|-groupby groupformat`

Specifies how to group the command output. See [-gby|-groupby](#) for details.

`-int|-intersection`

Specifies the comparison to show the tasks in common between both project groupings.

`-nch|-nocolumn_header`

Specifies not to use a column header in the output format. See [-ch|-column headers](#) for details.

---

`-nf|-noformat`

Specifies not to use column alignment. See [-nf|-noformat](#) for details.

`-ns|-no_sort`

Specifies that the command's output will not be sorted. See [-ns|-nosort](#) for details.

`-not|-not_in`

Specifies the comparison to show the tasks in the first project grouping that are not in second project grouping.

*project\_grouping\_spec1*

Specifies the first project grouping to be compared. For more information, see [Project grouping specification](#).

*project\_grouping\_spec2*

Specifies the project grouping to compare to *project\_grouping\_spec1*. For more information, see [Project grouping specification](#).

`-rt|-removed_tasks`

Compares the removed tasks in the two project groupings.

`-sep|-separator separator`

Specifies a different separator character. See [-sep|-separator](#) for details.

`-sby|-sortby sortspec`

Specifies how to sort the command output. See [-sby|-sortby](#) for details.

`-tob|-tasks_on_top_of_baseline`

Compares the tasks on top of the baseline in the two project groupings.

`-un|-union`

Specifies the comparison to show the tasks that are in both project groupings.

`-u|-unnumbered`

Suppresses automatic numbering of the command's output (that is, the output is unnumbered). See [-u|-unnumbered](#) for details.

---

### **Example**

- Show the tasks on top of the baseline in project grouping **My 2.0 Collaborative Development projects** that are not in the project grouping **All 2.0 Integration Testing projects**.

```
ccm pg -compare -tasks -not_in "My 2.0 Collaborative Development projects" "All 2.0 Integration Testing projects"
```

### **Related topics**

- [Comparing projects for a project grouping with projects for a baseline](#)
- [Comparing projects for two project groupings](#)
- [Comparing tasks for a project grouping with tasks for a baseline](#)
- [Listing project groupings](#)
- [Showing a project grouping property](#)
- [Showing project grouping information](#)

---

## **Copying tasks from one project grouping to another**

This subcommand copies tasks from one project grouping to another. Any user who can modify the destination project grouping can use this subcommand.

```
ccm pg|project_grouping -ct|-copy_tasks project_grouping_spec1
      project_grouping_spec2
```

`-ct|-copy_tasks`

Copies the net tasks (Saved Tasks plus Added Tasks) from one project grouping to another. The tasks are added to the second project grouping in the same way as if the `-add_tasks` option had been used.

However, dependency analysis is not done, and required tasks are not calculated. This gives you a way to add the exact set of tasks to a different project grouping.

*project\_grouping\_spec1*

Specifies the project grouping from which tasks will be copied. See [Project grouping specification](#) for details.

*project\_grouping\_spec2*

Specifies the project groupings to which tasks will be copied. See [Project grouping specification](#) for details.

### **Examples:**

- Copy tasks from one project grouping to another:

```
ccm pg -l
```

- 1) All 1.0 Integration Testing Projects from Database G
- 2) All 2.0 Integration Testing Projects from Database G
- 3) All 2.0 System Testing Projects from Database G
- 4) All A/1.0 Integration Testing Projects from Database G

- Copy tasks from **All 2.0 Integration Testing Projects from Database G** to **All 1.0 Integration Testing Projects from Database G**.

```
ccm pg -ct @2 @1
```

### **Related topics**

- [Listing project groupings](#)
- [Showing a project grouping property](#)
- [Showing the associated projects, baseline, tasks, or objects for a project grouping](#)
- [Showing project grouping information](#)

---

## ***Deleting a project grouping and members***

This subcommand deletes a project grouping, either with or without its member projects. To delete a project grouping, a user must be able to modify it and the project grouping must have no member projects.

```
ccm pg|project_grouping -d|-delete ([-m|-members] | [-nm|-no_members])
    project_grouping_spec...
```

`-delete`

Deletes the specified project grouping. You can set one or more *project\_grouping\_spec* arguments to multiple objects. It does not update the query selection set.

`-m|-members`

Specifies that the project's groupings associated projects should be deleted as well as the project grouping. All associated folders that are not used in any project or project grouping are also deleted. The default is `-nm|-no_members`.

`-nm|-no_members`

Specifies that the project's groupings associated projects should not be deleted. The operation will only succeed if the specified project grouping has no associated projects. This is the default if no options are specified.

*project\_grouping\_spec*

Specifies the project groupings to delete. See [Project grouping specification](#) for details.

## **Related topics**

- [Showing the associated projects, baseline, tasks, or objects for a project grouping](#)
- [Showing project grouping information](#)

---

## ***Listing project groupings***

This subcommand lists the project groupings that match the specified criteria. If you do not specify options, all project groupings are listed.

```
ccm pg|project_grouping -l|-list [(-r|-release release_spec)...]
    [(-purpose purpose)...] [(-o|-owner owner)...] [-f|-format format]
    [-nf|-noformat] ([-ch|-column_header] | [-nch|-nocolumn_header])
    [-sep|-separator separator] ([-sby|-sortby sortspec] |
    [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat] [-u|-unnumbered]
```

`-ch|-column_header`

Specifies to use a column header in the output format. See [-ch|-column\\_headers](#) for details.

`-f|-format format`

Specifies to use a column header in the output format. See [-f|-format](#) for details.

A keyword can be built-in (`%fullname`, `%displayname`, `%objectname`) or the name of any existing attribute such as `%modify_time` or `%status`.

See [Built-In keywords](#) for a list of keywords.

`-gby|-groupby groupformat`

Specifies how to group the command output. See [-gby|-groupby](#) for details.

`-nch|-nocolumn_header`

Specifies not to use a column header in the output format. See [-ch|-column\\_headers](#) for details.

`-nf|-noformat`

Specifies not to use column alignment. See [-nf|-noformat](#) for details.

`-ns|-nosort|-no_sort`

Specifies that the command's output will not be sorted. See [-ns|-nosort](#) for details.

`-o|-owner owner`

Specifies that only project groupings with the specified owner should be listed. The owner can be any string that represents a user name. If not specified, project groupings for all owners are listed.

---

`-purpose purpose`

Specifies to list project groupings for the specified purpose. The purpose should be the name of a valid defined purpose. If not specified, project groupings for all purposes are listed.

`r|-release release_spec`

Specifies that only project groupings for the specified release should be listed. See [Release specification](#) for details. If not specified, project groupings for all releases are listed.

`-sby|-sortby sortspec`

Specifies how to sort the command output. See [-sby|-sortby](#) for details.

`-sep|-separator separator`

Used only with the `-f|-format` option. Specifies a different separator character. See [-sep|-separator](#) for details.

`-u|-unnumbered`

Suppresses automatic numbering of the command's output (that is, the output is unnumbered). See [-u|-unnumbered](#) for details.

### Example

- List the project groupings:

```
ccm pg -list -r Base/1.0 -purpose "Insulated Development" -purpose  
"Integration Testing" -r A/1.0
```

```
1) All A/1.0 Integration Testing Projects from Database G  
2) All Base/1.0 Integration Testing Projects from Database G  
3) My Base/1.0 Insulated Development Projects  
4) bmgr1's Base/1.0 Insulated Development Projects  
5) dev1's Base/1.0 Insulated Development Projects  
6) dev2's Base/1.0 Insulated Development Projects  
7) dev3's Base/1.0 Insulated Development Projects
```

### Related topics

- [Showing a project grouping property](#)
- [Showing project grouping information](#)

---

## **Previewing update baseline and tasks for a project grouping**

This subcommand performs a preview update of the baseline and tasks for specified project groupings. Use the output to view the baseline and tasks that the project grouping would use if you performed an update members operation.

```
ccm pg|project_grouping -pubt|-preview_update_baseline_and_tasks  
    [-iat|-include_automatic_tasks] project_grouping_spec...
```

`-iat|-include_automatic_tasks`

Causes the preview to include any automatic tasks. If not specified, automatic tasks are excluded.

*project\_grouping\_spec*

Specifies the project groupings to be previewed. See [Project grouping specification](#) for details.

If you specify multiple project groupings, each will be processed in the order that you specified.

A project grouping can have only one baseline.

### **Example**

- Preview the baseline and tasks that would be in the project grouping **My 2.0 Collaborative Development projects** if it were updated, including any automatic tasks.

```
ccm pg -pubt -iat "My 2.0 Collaborative Development projects"
```

### **Related topics**

- [Listing project groupings](#)
- [Comparing tasks for project groupings](#)
- [Showing a project grouping property](#)
- [Showing project grouping information](#)
- [Updating the baseline and tasks for a project grouping](#)



---

## **Removing tasks from the update properties for a project grouping**

This subcommand removes tasks from the specified project groupings. The command supports removing:

- Specified tasks
- All tasks currently in the project grouping
- All tasks that were manually added to the project grouping

Any user who can modify the project grouping can use this subcommand.

```
ccm pg|project_grouping
    -rt|-remove_task|-remove_tasks (task_spec|(all | all_added))
    project_grouping_spec...
```

*project\_grouping\_spec*

Specifies the project groupings to update. See [Project grouping specification](#) for details.

```
(-rt|-remove_task|-remove_tasks (task_spec|(all|all_added)))...
```

Specifies the tasks to be removed from the specified project groupings. If a task to be added is in the list of added tasks, it is removed from that list. If the task is in the list of saved tasks, then it is added to the removed tasks list for the project grouping. The keyword `all` means remove all added tasks, and add all saved tasks to the removed tasks list. The keyword `all_added` means remove all added tasks. See [Task specification](#) for details.

## **Related topics**

- [Adding tasks to the update properties for a project grouping](#)
- [Copying tasks from one project grouping to another](#)
- [Listing project groupings](#)
- [Showing the associated projects, baseline, tasks, or objects for a project grouping](#)
- [Showing project grouping information](#)

---

## Setting the auto-update mode for a project grouping

This subcommand defines the auto-update function for the specified project groupings. By default, when a project or project grouping's members are updated, the baseline and tasks for the project grouping are updated. This can be cleared and set.

```
ccm pg|project_grouping -au|-auto_update_baseline_and_tasks|-thaw
    project_grouping_spec...
ccm pg|project_grouping -no_au|-no_auto_update_baseline_and_tasks|-freeze
    project_grouping_spec...
```

`-au|-auto_update_baselines_and_tasks|-thaw`

Specifies that the project grouping always updates the baseline and tasks during an update operation.

However, dependency analysis is not done, and required tasks are not calculated. This gives you a way to add the exact set of tasks to a different project grouping.

`-no_au|-no_auto_update_baselines_and_tasks|-freeze`

Specifies that the project grouping always uses the saved baseline and tasks.

*project\_grouping\_spec*

Specifies the project groupings to be updated. See [Project grouping specification](#) for details.

## Related topics

- [Listing project groupings](#)
- [Showing a project grouping property](#)
- [Showing project grouping information](#)
- [Updating the baseline and tasks for a project grouping](#)

---

## **Showing a project grouping property**

This subcommand shows a specific property of selected project groupings.

```
ccm pg|project_grouping -s|-sh|-show ((r|release) | (p|purpose) |  
    (o|owner) | created_in | (au|auto_update_baselines_and_tasks) |  
    (utime|update_time)) project_grouping_spec...
```

*au|auto\_update\_baselines\_and\_tasks*

If specified, the project grouping always refreshes the baseline and tasks during an update operation.

*created\_in*

If specified, the name of the database where the project grouping was created is displayed.

*o|owner*

If specified, displays the name of the owner of the project grouping.

*project\_grouping\_spec*

Specifies the project groupings to be shown. See [Project grouping specification](#) for details.

*p|purpose*

If specified, displays the purpose of the project grouping.

*r|release*

If specified, displays the release value of the project grouping.

*-s|-sh|-show*

Shows the project grouping properties in the order specified by the arguments.

*utime|update\_time*

If specified, the time that the baseline and tasks were last computed (and saved) is displayed.

---

## **Example**

- Show a project grouping property (auto-update baselines and tasks):

```
ccm pg -s auto_update_baselines_and_tasks "All Base/1.0 Integration  
Testing Projects from Database G"
```

```
Project Grouping All Base/1.0 Integration Testing Projects from  
Database G: TRUE
```

## **Related topics**

- [Listing project groupings](#)
- [Showing project grouping information](#)

---

## Showing the associated projects, baseline, tasks, or objects for a project grouping

This subcommand shows the associated projects, baseline, tasks, or objects for the specified project groupings.

```
ccm pg|project_grouping -s|-sh|-show ((proj|projects) | (bl|baseline) |
    (at|added_tasks) | (rt|removed_tasks)
    (tob|tasks_on_top_of_baseline) | | all_tasks |
    (obj|objs|objects)) [-f|-format format] [-nf|-noformat]
    ([-ch|-column_header] | [-nch|-nocolumn_header])
    [-sep|-separator separator] ([-sby|-sortby sortspec] |
    [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat] [-u|-unnumbered]
    project_grouping_spec...
```

[-ch|-column header](#)

[-f|-format format](#)

[-gby|-groupby groupformat](#)

[-nch|-nocolumn header](#)

[-nf|-noformat](#)

[-ns|-nosort|-no sort](#)

*project\_grouping\_spec*

Specifies the project groupings to show. See [Project grouping specification](#) for details.

[-sby|-sortby sortspec](#)

[-sep|-separator separator](#)

*-s|-sh|-show*

The following keywords are supported with *-show*:

- *proj|projects*

If specified, all projects that are included in the project grouping are displayed.  
The default format is:

---

```
%displayname %status %owner %release %create_time
```

The default format may be overridden by using the `-format` option.

- `bl|baseline`

If specified, the baseline name is displayed. The default format is:

```
%displayname: %description
```

- `at|added_tasks`

If specified, all tasks that are in the project grouping's Added Tasks are displayed. Added Tasks are the tasks that the user added manually.

The default format is:

```
%displayname %release %owner %create_time
```

The default format may be overridden by using the `-format` option.

- `obj|objs|objects`

If specified, all objects that are included in all projects in the project grouping are displayed. The default format is:

```
%displayname %status %owner %release %create_time
```

The default format may be overridden by using the `-format` option.

- `rt|removed_tasks`

If specified, all tasks that are in the project grouping's Removed Tasks are displayed. These are the tasks that the user manually removed.

The default format is:

```
%displayname %release %owner %create_time
```

The default format may be overridden by using the `-format` option.

- `tob|tasks_on_top_of_baseline`

If specified, displays the tasks that are in the project grouping but are not in the baseline.

- `all_tasks`

If specified, shows the set of tasks used by the update operation. The default format is:

```
%displayname %release %owner %create_time
```

The default format may be overridden by using the `-format` option.

---

[-u|-unnumbered](#)

### **Example**

- Show all tasks in the project grouping:

```
ccm pg -s all_tasks "All Base/1.0 Integration Testing Projects from
Database G"
```

```
Project Grouping All Base/1.0 Integration Testing Projects from
Database G:
```

```
1) G#123 Base/1.0 dev3 7/4/08 12:46 PM
```

### **Related topics**

- [Listing project groupings](#)
- [Showing a project grouping property](#)
- [Showing project grouping information](#)

---

## Showing project grouping information

This subcommand shows information about the specified project groupings.

```
ccm pg|project_grouping -s|-sh|-show (i|info|information)
    -f|-format format [-nf|-noformat]
    ([-ch|-column_header] | [-nch|-nocolumn_header])
    [-sep|-separator separator] project_grouping_spec...
ccm pg|project_grouping -s|-sh|-show (i|info|information)
    project_grouping_spec...
```

[-ch|-column\\_header](#)

[-f|-format \*format\*](#)

[-list](#)

Lists the project groupings in the database. The `project_grouping -list` subcommand supports the numbered format options and sets the query selection set. The command accepts zero, one, or many release, owner, and purpose options. Each release option accepts a *ReleaseSpec* option value that you can set to a single object. Each owner option accepts an *owner* string. Each purpose option accepts a *purpose name* string.

[-nch|-nocolumn\\_header](#)

[-nf|-noformat](#)

*project\_grouping\_spec*

Specifies the project groupings to show. See [Project grouping specification](#) for details.

[-sep|-separator \*separator\*](#)



---

## Example

- Show project grouping information:

```
ccm pg -show information "All 1.0 Integration Testing Projects from Database G"
```

```
Project Grouping All 1.0 Integration Testing Projects from Database G:
```

```
Release: 1.0
```

```
Purpose: Integration Testing
```

```
Owner: john
```

```
Projects:
```

```
Prj_J6524-one prep john 1.0 Integration Testing
```

```
Prj_J6614-dir prep john 1.0 Integration Testing
```

## Related topics

- [Listing project groupings](#)
- [Showing a project grouping property](#)

---

## **Updating the baseline and tasks for a project grouping**

This subcommand updates the baseline and tasks for the specified project groupings. If the process rule associated with the project grouping uses a search mode of latest baseline, the latest baseline matching the criteria specified on the process rule is evaluated and selected for the project grouping. The tasks specified by the associated process rule are used for the project grouping.

```
ccm pg|project_grouping -ubt|-update_baseline_and_tasks
    project_grouping_spec...
```

*project\_grouping\_spec*

Specifies the project groupings to be updated. See [Project grouping specification](#) for details.

### **Examples**

- Update the project grouping's baselines and tasks:

```
ccm pg -ubt "All Base/1.0 Integration Testing Projects from Database G"
```

- Update the baseline and tasks of a project grouping named **My CM/7.0 Collaborative Development**.

```
ccm project_grouping -ubt "My CM/7.0 Collaborative Development"
```

### **Related topics**

- [Listing project groupings](#)
- [Showing baseline projects, folders, folder templates, or members for a process rule](#)
- [Showing a project grouping property](#)
- [Showing project grouping information](#)
- [Updating members for a project grouping](#)

---

## Description and uses

Use project groupings to organize projects by release and purpose for the update operation. The task and baseline properties for a project grouping are used when a project is updated so that member selection is consistent across all projects in the group. A project can be a member of only one project grouping. Synergy creates a project grouping automatically when you create a project.

Project groupings can be private or non-private. All projects in a private project grouping have the same owner, release, purpose, and state as the project grouping. Private project groupings are identified in one of the following ways:

- *My release purpose Projects*  
The owner of the project grouping is the same as the current user and the database is not DCM-enabled, or the project grouping was created in the local database, such as *My CM/6.5 Insulated Development Projects*.
- *owner's release purpose Projects*  
The owner of the project grouping is a different user and the database is not DCM-enabled, or the project grouping was created in the local database, such as *John's CM/6.5 Insulated Development Projects*.
- *My release purpose Projects from Database dbid*  
The owner of the project grouping is the same as the current user and the database is DCM-enabled, and the project grouping was not created in the local database, such as *My CM/6.5 Insulated Development Projects from Database D*.
- *owner's release purpose Projects from Database dbid*  
The owner of the project grouping is a different user and the database is DCM-enabled, and the project grouping was not created in the local database, such as *John's CM/6.5 Insulated Development Projects from Database D*.

All projects in a non-private project grouping have the same release, purpose, and state as the project grouping. Non-private project groupings are identified in one of the following ways:

- *All release purpose Projects from Database dbid for DCM-enabled databases, where dbid is the database id of the database in which the project grouping was created, such as All CM/6.5 Integration Testing Projects from Database D.*
- *All release purpose Projects for non DCM-enabled databases, such as All CM/6.5 System Testing Projects.*

Every local project grouping is associated with the process rule that corresponds to its release and purpose. A project grouping can have only one related process rule.

However, note that in some cases, all projects in a project grouping may not have update properties specified by the project grouping. Those that use process rules will have the

---

same update properties. A project grouping can contain projects that do not use process rules, or *even projects that update using objects instead of tasks*. The ability to place them in the same grouping creates baselines from the full set of projects.

To have the appropriate update properties, project groupings have many associations with other objects in the database. Because process rules use folders and tasks, these same folders and tasks are associated with a project grouping that use process rules. In addition, a project grouping has a set of saved tasks, a set of additional tasks, a set of removed tasks, and a set of automatic tasks, each of which is specific to the project grouping. You can also add and remove tasks in the grouping. Every local project grouping also has a relationship to a baseline, if the process rules use baselines.

For more detailed information about how build managers can best use project groupings, see the [Rational Synergy Build Manager's Guide](#).

---

## project\_purpose command

See [Description and uses](#) for details. The `project_purpose` command supports the following subcommands:

- [Creating a project purpose](#)
- [Deleting a project purpose](#)
- [Modifying a project purpose](#)
- [Showing a project purpose](#)

---

## Creating a project purpose

This subcommand creates a project purpose. You must be in the *build\_mgr* or *ccm\_admin* role to use this subcommand.

```
ccm project_purpose -cr|-create -n|-name purpose_name
                  -stat|-status status [-ms|-member_status member_status]
```

```
-ms|-member_status member_status
```

Specifies the member status for a project purpose. The member status differentiates projects of the same state being used for different purposes when you update. The value must be unique in the database.

The purpose and the member status should be similar. For example, if you are creating a **Test Integration** purpose, set the member status value to a similar name, such as **test\_int**.

When creating a purpose, if you don't specify a member status, a unique value is automatically generated and used.

```
-n|-name purpose_name
```

Specifies the name of the new project purpose. The name must be unique in the database.

Note that if you are using a DCM initialized database, you might need to create the same purpose in other databases in the DCM cluster.

```
-stat|-status status
```

Specifies the state of the new purpose. This should be a modifiable state, such as *working*, *visible*, *shared* or *prep*.

## Example

- Create a project purpose with a name of `Test Purpose`, a status of `prep`, and a member status of `test`. View the newly created purpose.

```
ccm project_purpose -cr -name "Test Purpose" -stat prep -ms test
ccm project_purpose -s "Test Purpose"
Purpose      Member Status      Status
Test Purpose      test                prep
```

---

### ***Deleting a project purpose***

This subcommand deletes a project purpose specified by *purpose\_name*. After a project purpose is deleted, the following behavior will occur:

- Projects cannot be copied or created with the deleted purpose.
- Existing projects and products retain the member status setting.
- Existing projects and products cannot have their purpose changed to the deleted purpose.
- Process rules for that purpose are deleted.

You must be in the *build\_mgr* or *ccm\_admin* role to use this subcommand.

```
ccm project_purpose -d|-delete purpose_name...
```

```
-d|-delete purpose_name...
```

Specifies the name of the project purpose to be deleted. The name must be that of a valid purpose.

### ***Example***

- Delete a project purpose called `Test2 Purpose`.

```
ccm project_purpose -d "Test2 Purpose"
```

---

## Modifying a project purpose

This subcommand modifies the name or member status of a project purpose. You must be in the *build\_mgr* or *ccm\_admin* role to use this subcommand.

```
ccm project_purpose -m|-modify [-n|-name purpose_name]  
[-ms|-member_status member_status] purpose_name
```

```
-m|-modify [-n|-name purpose_name]
```

Modifies a project purpose. When used with the `-name purpose_name` options, specifies the name of the project purpose to change. When used with the `-member_status` option, specifies the value of the member status to change.

```
-ms|-member_status
```

Specifies the member status for a project purpose. The member status differentiates projects of the same state being used for different purposes when you update. The value must be unique in the database.

Use this option with the `-modify` option to specify the member status of the purpose you are modifying. Values from the project purpose table are used for each purpose option.

The purpose and the member status should be similar. For example, if you are creating a purpose of `Test Integration`, then the member status should be set to a similar value, such as `test_int`.

## Example

- Change the name and member status of a project purpose.

```
ccm project_purpose -m -n "Test2 Purpose" -ms test2 "Test Purpose"
```



---

## Showing a project purpose

```
ccm project_purpose -s|-sh|-show ([-stat|-status status] |  
[-personal] | [-no_personal|-nopersonal])  
[-rel|-release release_spec] [-f|-format format]  
[-nf|-noformat] ([-ch|-column_header] |  
[-nch|-nocolumn_header]) [-sep|-separator separator]  
([-sby|-sortby sortspec] | [-ns|-nosort|-no_sort])  
[-gby|-groupby groupformat]  
ccm project_purpose -s|-sh|-show [-f|-format format] [-nf|-noformat]  
([-ch|-column_header] | [-nch|-nocolumn_header])  
[-sep|-separator separator]  
([-sby|-sortby sortspec] | [-ns|-nosort|-no_sort])  
[-gby|-groupby groupformat] purpose_name...
```

`-ch|-column_header`

Specifies to use a column header in the output format. See [-ch|-column\\_headers](#) for details.

`-f|-format format`

Specifies the command output format. See [-f|-format](#) for details.

`-gby|-groupby groupformat`

Specifies how to group the command output. See [-gby|-groupby](#) for details.

`-nch|-nocolumn_header`

Specifies not to use a column header in the output format. See [-ch|-column\\_headers](#) for details.

`-nf|-noformat`

Specifies not to use column alignment. See [-nf|-noformat](#) for details.

`-no_personal|-nopersonal`

Shows project purposes associated with states that are not for personal use, such as *shared* or *prep*.

`-ns|-no_sort`

Specifies that the command's output will not be sorted. See [-ns|-nosort](#) for details.

---

`-personal`

Shows project purposes associated with states that are for personal use, such as *working* or *visible*.

`purpose_name`

Specifies the name of a project purpose to show. This must be a defined purpose in the current database.

`-rel|-release release_spec`

Shows the project purposes that are valid for the specified release. You can set `release_spec` to one release. See [Release specification](#) for details.

`-sep|-separator separator`

Specifies a different separator character. See [-sep|-separator](#) for details.

`-sby|-sortby sortspec`

Specifies how to sort the command output. See [-sby|-sortby](#) for details.

`-stat|-status status`

Specifies to show project purposes with the specified status only.

### **Example**

- Show the project purposes for a user in the *developer* role.

```
ccm project_purpose -show -role -personal
```

Purpose Name	Member Status	Status
Collaborative Development	collaborative	working
Insulated Development	working	working
Visible Development	visible	visible

---

## Description and uses

The `project_purpose` command creates or shows (depending on your user role) the project purposes for a Rational Synergy database. All users can show project purposes. A project purpose manager can create a project purpose. Use the project purposes to set up multiple *prep*, *shared*, *working*, or *visible* versions of the same project for different uses, such as different levels of testing.

The project purposes include the following:

- Purpose name  
This name reflects the purpose, for example, performance testing, personal use, etc.
- Member status for the purpose  
The member status differentiates projects of the same state being used for different purposes when you perform an update operation. For example, you could define three unique levels of system testing called `sqa1`, `sqa2`, and `sqa3`.
- Status of the project  
The status shows what state projects (*working*, *prep*, etc.) of this purpose can use.

The project purpose table affects the following:

- Options you can specify in the following commands: `ccm copy_project` and `ccm create -type project`.
- Specifies the `status` and `member_status` values that are used for the projects copied using each purpose option.
- Determines which automatic tasks projects will be associated with
- Affects the synopses of corresponding automatic tasks

Each Rational Synergy database contains one project purpose list only. You can define project purpose lists for each release.

The project purpose table defines the following purposes:

Integration Testing:	<code>prep:</code>	<code>integrate</code>
System Testing:	<code>prep:</code>	<code>sqa</code>
Insulated Development:	<code>working:</code>	<code>working</code>
Collaborative Development:	<code>working:</code>	<code>collaborative</code>
Shared Development:	<code>shared:</code>	<code>shared</code>
Visible Development:	<code>visible:</code>	<code>visible</code>
Master Integration Testing:	<code>master_integrate:</code>	<code>prep</code>

## properties command

See [Description and uses](#) for details. The `properties` command supports the following subcommands:

- [Showing properties](#)
- [Showing properties with a specified format](#)

---

## Showing properties

This subcommand shows information that is appropriate for an object. The attributes that are displayed depend on the type of the object. The format shows the most relevant information.

The command does not set the query selection set. The query shows objects in the order in which they are specified.

```
ccm info|prop|properties -p|-project [-v|-verbose] project_spec...
ccm info|prop|properties [-v|-verbose] object_spec...
```

*object\_spec*

Specifies the objects whose properties are to be shown.

*-p|-project*

Shows the history of a project.

*project\_spec*

Specifies the project to show. See [Project specification](#) for details.

*-v|-verbose*

Shows a more detailed set of information. This only applies to certain types of objects, such as folders. It will be ignored for an object type that does not have a verbose information form.

## Examples

- Obtain information about the `os_ico-1` project, which uses object status to update.

```
ccm prop -p Project-Merge
```

- Obtain information about the `task_ico-2` project, which uses tasks to update.

```
ccm properties a.txt-1:ascii:1
```

- Show the release values of all of the objects in the current directory.

```
ccm prop -f "%objectname %release" *
```

---

## Showing properties with a specified format

This subcommand shows information about an object in a specified format. The command does not set the query selection set. The query shows objects in the order in which they are specified.

```
ccm info|prop|properties -p|-project -f|-format format [-nf|-noformat]
    ([-ch|-column_header] | [-nch|-nocolumn_header])
    [-sep|-separator separator] project_spec...
ccm info|prop|properties -f|-format format [-nf|-noformat]
    ([-ch|-column_header] | [-nch|-nocolumn_header])
    [-sep|-separator separator] object_spec...
```

`-ch|-column_header`

Specifies to use a column header in the output format. See [-ch|-column headers](#) for details.

`-f|-format format`

Specifies the command output format. See [-f|-format](#) for details.

A keyword can be built-in (%fullname, %displayname, %objectname) or the name of any existing attribute such as %modify\_time or %status.

See [Built-In keywords](#) for a list of keywords.

`-nch|-nocolumn_header`

Specifies not to use a column header in the output format. See [-ch|-column headers](#) for details.

`-nf|-noformat`

Specifies not to use column alignment. See [-nf|-noformat](#) for details.

`-ns|-nosort|-no_sort`

Specifies that the command's output will not be sorted. See [-ns|-nosort](#) for details.

`-p|-project`

Shows the history of a project.

`project_spec`

Specifies the project to list. See [Project specification](#) for details.

---

`-sep|-separator separator`

Used only with the `-f|-format` option. Specifies a different separator character. See [-sep|-separator](#) for details.

---

## Description and uses

The `properties` command provides information about one or more objects.

This subcommand displays the attribute values of a group of model-defined attributes for the specified objects to standard output.



## query command

See [Description and uses](#) for details. The `query` command supports the [Querying for objects or showing the query selection set](#) subcommand.

---

## Querying for objects or showing the query selection set

```
ccm query [(-n|-name name)...] [(-o|-owner owner)...]
          [(-s|-state state)...] [(-t|-type type)...]
          [(-v|-version version)...] [(-i|-instance instance)...]
          [(-release release_spec)...] [(-task task_spec)...]
          [-f|-format format] [-nf|-noformat]
          ([-ch|-column_header] | [-nch|-nocolumn_header])
          [-sep|-separator separator] ([-sby|-sortby sortspec] |
          [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat]
          [-u|-unnumbered] [query_string]
```

`-ch|-column_header`

Specifies to use a column header in the output format. See [-ch|-column\\_headers](#) for details.

`-f|-format format`

Specifies the command output format. See [-f|-format](#) for details.

`-gby|-groupby groupformat`

Specifies how to group the command output. See [-gby|-groupby](#) for details.

`-i|-instance instance...`

Includes a query clause of the form `instance='instance'` to find objects with the specified instance.

`-n|-name name...`

Includes a query clause of the form `name='name'` to find objects with the specified name.

`-nch|-nocolumn_header`

Specifies not to use a column header in the output format. See [-ch|-column\\_headers](#) for details.

`-nf|-noformat`

Specifies not to use column alignment. See [-nf|-noformat](#) for details.

`-ns|-no_sort`

Specifies that the command's output will not be sorted. See [-ns|-nosort](#) for details.

---

`-o|-owner owner...`

Includes a query clause of the form `owner='owner'` to find objects with the specified owner.

`query_string`

Specifies a query string to be combined with query clauses. The query clauses are generated from query-related options and form the query expression that is evaluated. See "Query expressions" in [Rational Synergy CLI Help, Traditional mode](#).

`-release release_spec...`

Includes a query clause of the form `release='release'` to find objects with the specified release value. You can set `release_spec` to multiple release definitions. See [Release specification](#) for details.

`-sep|-separator separator`

Specifies a different separator character. See [-sep|-separator](#) for details.

`-sby|-sortby sortspec`

Specifies how to sort the command output. See [-sby|-sortby](#) for details.

`-s|-state state...`

Includes a query clause of the form `state='state'` to find objects with the specified state.

`-task task_spec...`

Includes a query clause of the form `is_associated_cv_of(task('task_spec'))` to find the associated objects of the specified task. You can set `task_spec` to multiple tasks. See [Task specification](#) for details.

`-t|-type type...`

Includes a query clause of the form `cvtype='type'` to find objects with the specified type.

---

`-v|-version version...`

Includes a query clause of the form `version='version'` to find objects with the specified version.

`-u|-unnumbered`

Suppresses automatic numbering of the command's output (that is, the output is unnumbered). See [-u|-unnumbered](#) for details.

## Examples

- List all objects named `foo.c` owned by `valerie`.

```
ccm query -n foo.c -o jane
1) foo.c-1    integrate jane nasub1 csrc 1 1
2) foo.c-1.2 working  jane nasub1 csrc 1 4
3) foo.c-2    working  jane nasub2 csrc 1 5
```

- To look at the source contents of item 3 in the selection set, enter the following.

```
ccm cat @3
```

- List all objects named `foo.c`, owned by `ann`, for task 4.

```
ccm query -n foo.c -o ann -task 4
1) foo.c-1.2 working  ann csrc 1 4
```

- List the name and time last modified of all objects named `brochure.doc` owned by `ann`.

```
ccm query -n brochure.doc -o ann -f "%name %modify_time"
1) brochure.doc Tue Aug  6 12:17:55 1996
```

- List all objects associated with task 3 that are from the `santa_fe` database.

```
ccm query -task 3 -db santa_fe
1) DropEdit.cpp-1    integrate tom c++ diffmerge santa_fe#1 <void>
2) vdifmrgDoc.cpp-1 integrate tom c++ diffmerge santa_fe#1 <void>
```

- List change requests associated with a particular transfer set.

```
ccm query query_expression
```

where *query\_expression* is the change request query that is being used for the transfer set, and includes "`cvtype=problem`".

For example:

```
ccm query "cvtype='problem' and product_name='myproduct'"
```

- Show release-specific process rules that are instantiations of the **Collaborative Development** generic process rule.

```
ccm query ''cvtype='process_rule' and name='Collaborative
Development'' -f "%none %is_generic_pr_of"
```

---

## Related topics

- [finduse command](#)

---

## Description and uses

Use the `query` command to search for objects in the database. Rational Synergy evaluates a query expression during a search operation. The query expression can consist of any query clause from query-related options combined with any `query_string` argument. The results of the query display in the selection set.

By default, the query sorts objects using sorting criteria, which is described in [Sorting and grouping](#).

If you don't specify a query expression using query-related options or a `query_string` argument, the `query` command shows the current selection set and applies any sorting, then updates the selection set.

### Query functions and sorting

To use a query function that provides sorting (for example, `recursive_is_member_of`), the query function's sorting order is applied to the final displayed result if `-no_sort` is specified, and if that query function is not combined with other query operators to make a compound query.

### Selection set ordering and use

By default, the output is numbered to show the selection set reference number. You can then reference specific objects in the selection set by using the selection set reference syntax (for example, `@1`). See [Query selection set reference form](#) for details.

### Query expression construction

The command supports a number of options for constructing a query expression. For example, the `-name` option provides an alternative way of constructing a query clause of the type `name='name'`.

If such an option is repeated, the corresponding query clauses are combined with an `or`. For example, `-n joe -n ann` results in a query clause `(name='joe' or name='ann')`.

Query clauses for different options are combined with an `and`. For example, `-n joe -s working` results in a query clause `(name='joe') and (status='working')`.

These constructed query clauses are combined with any specified `query_string` argument with an `and`. For example, `-n joe "is_hist_leaf()"` results in a query expression of `(name='joe') and (is_hist_leaf())`.

## reconcile command

See [Description and uses](#) for details. The `reconcile` command supports the following subcommands:

- [Showing work area conflicts](#)
- [Synchronizing a work area with changes from the database](#)
- [Synchronizing the database with changes from a work area](#)

---

## Showing work area conflicts

This subcommand identifies and shows work area conflicts but does not perform any action to resolve them.

```
ccm rwa|recon|reconcile -p|-project [-s|-sh|-show]
    ([-cu|-consider_uncontrolled] | [-iu|-ignore_uncontrolled])
    ([-mwaf|-missing_wa_file] | [-imwaf|-ignore_missing_wa_file])
    ([-r|-recurse] | [-nr|-norecurse|-no_recurse])
    [-if|-ignore_files|-ignore_types file_type,...] [-f|-format format]
    [-nf|-noformat] ([-ch|-column_header] | [-nch|-nocolumn_header])
    [-sep|-separator separator] ([-sby|-sortby sortspec] |
    [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat] project_spec...
ccm rwa|recon|reconcile [-s|-sh|-show]
    ([-cu|-consider_uncontrolled] | [-iu|-ignore_uncontrolled])
    ([-mwaf|-missing_wa_file] | [-imwaf|-ignore_missing_wa_file])
    ([-r|-recurse] | [-nr|-norecurse|-no_recurse])
    [-if|-ignore_files|-ignore_types file_type,...] [-f|-format format]
    [-nf|-noformat] ([-ch|-column_header] | [-nch|-nocolumn_header])
    [-sep|-separator separator] ([-sby|-sortby sortspec] |
    [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat] file_spec..
```

`-ch|-column_header`

Specifies to use a column header in the output format. See [-ch|-column headers](#) for details.

`-cu|-consider_uncontrolled`

Specifies to consider uncontrolled files during reconcile. Any files not under source control are reported as work area conflicts. If neither `-cu|-consider_uncontrolled` or `-if|-ignore_files|-ignore_uncontrolled` is specified, the default is to ignore uncontrolled files.

`-f|-format format`

Specifies the command output format. See [-f|-format](#) for details.

A keyword can be built-in (`%fullname`, `%displayname`, `%objectname`) or the name of any existing attribute such as `%modify_time` or `%status`.

See [Built-In keywords](#) for a list of keywords.

*file\_spec*

Specifies the file or directory to be reconciled.



---

`-gby|-groupby groupformat`

Specifies how to group the command output. See [-gby|-groupby](#) for details.

`-if|-ignore_files|-ignore_types file_type,...`

Specifies not to reconcile files with file names containing the specified extension. This option works only for uncontrolled files, and must be used with the `-cu|-consider_uncontrolled` option. The option value should be a list of one or more file extensions, separated by comma.

`-imwf|-ignore_missing_wa_file`

Specifies that files that are missing from the work area should be ignored and not reported as work area conflicts. This is the default if `-mwaf|-missing_wa_file` is not specified.

`-iu|-ignore_uncontrolled`

Specifies to ignore uncontrolled files during reconcile. If neither `-cu|-consider_uncontrolled` or `-if|-ignore_files|-ignore_uncontrolled` is specified, the default is to ignore uncontrolled files.

`-mwaf|-missing_wa_file`

Specifies that missing work area files should be reported as work area conflicts. The default is to ignore missing work area files.

`-nch|-nocolumn_header`

Specifies not to use a column header in the output format. See [-nch|-nocolumn\\_headers](#) for details.

`-nf|-noformat`

Specifies not to use column alignment. See [-nf|-noformat](#) for details.

`-nr|-no_recurse`

Specifies that reconcile does not recurse into the project's subprojects or the directory's files or sub-directories when a project or directory is reconciled. This is the default if `-nr|-norecurse|-no_recurse` is not specified.

`-ns|-nosort|-no_sort`

Specifies that the command's output will not be sorted. See [-sby|-sortby](#) for details.

---

*project\_spec*

Specifies the project to be reconciled.

`-r` | `-recurse`

Specifies that reconcile should also reconcile its recursive subprojects that reconcile should reconcile the files and sub-directories under that directory for a project being reconciled. The default is not to recurse.

This option controls the depth of a reconcile operation when you synchronize a project. This is important because if you are synchronizing a top-level project with many nested subprojects, a recurse reconcile could take a substantial amount of time and resources. You should carefully choose whether to recurse as it will reconcile every subproject beneath your specified top-level project. If you do not synchronize the hierarchy, you will save time and resources. Alternatively, if you need to reconcile the entire hierarchy, you can do so by using this option.

Note that if you specify a directory and `-recurse`, `reconcile` will not recurse into subprojects under that directory

`-sby` | `-sortby` *sortspec*

Specifies how to sort the command output. See [-sby/-sortby](#) for details.

`-sep` | `-separator` *separator*

Used only with the `-f` | `-format` option. Specifies a different separator character. See [-sep/-separator](#) for details.

`-s` | `-show`

Shows the conflicts without resolving them. This is the default.

`-udb` | `-update_db`

Updates the database with versions in your work area. Uses of this option include:

- If you modified a file that was not checked out, reconcile creates a new version by default, and the database is updated with your changes.
- If you updated the database copy of a file from another work area and you changed the same file from this work area, reconcile updates the database from this work area.

Use this option when you are certain that the work area represents the correct set of changes.

---

-uwa | -update\_wa

Updates your work area with versions from your database. Use this option when you are certain that the database represents the correct set of changes.

## Examples

- Reconcile the `ico_june16-1` project, but do not reconcile files whose file name contains any of the following extensions: `.doc`, `.gif`, or `.exe`.

```
ccm reconcile -p ico_june16-1 -ignore_types "*.doc;*.gif;*.exe"
```

- UNIX: Reconcile the `ico_june16-1` project, but discard the updates made in your work area and do not reconcile subprojects belonging to the project.

For this example, assume you were tasked to update the `move.c` object, which was in the *working* state, and the `colname.c` object, which was in the *integrate* state. After you copied and modified these objects in your work area, the direction of the project changed and you ended up not needing these changes after all.

```
% cd ~/john/ccm_wa/ccmint15
% ls
ico_june16-1
$ ccm reconcile -p ico_june16-1 -no_recurse
Examining work area for conflicts...
not recursing hierarchy, conflicts will be automatically discarded
Updating '/users/john/ccm_wa/ccmint15/ico_june16-1'...
Discarding changes to '/users/john/ccm_wa/ccmint15/ico_june16-1/
ico_june16/src/colname.c'..
Discarding changes to '/users/john/ccm_wa/ccmint15/ico_june16-1/
ico_june16/src/move.c'...
Reconciliation complete.
```

Note that the work area was updated with the original files from the database, and that the changes made to `colname.c` and `move.c` were discarded.

## Related topics

- [work\\_area command](#)

---

## Synchronizing a work area with changes from the database

This subcommand updates a work area with changes from the database. For a *working* or *visible* project, only the owner of the project can perform the operation. For a build management project, you must be a build manager to perform this operation. The work area must be visible and modifiable by you.

```
ccm rwa|recon|reconcile -uwa|-update_wa -p|-project
    ([-cu|-consider_uncontrolled] | [-iu|-ignore_uncontrolled])
    ([-mwaf|-missing_wa_file] | [-imwaf|-ignore_missing_wa_file])
    ([-r|-recurse] | [-nr|-norecurse|-no_recurse])
    [-if|-ignore_files|-ignore_types file_type,...] project_spec...
ccm rwa|recon|reconcile -uwa|-update_wa
    ([-cu|-consider_uncontrolled] | [-iu|-ignore_uncontrolled])
    ([-mwaf|-missing_wa_file] | [-imwaf|-ignore_missing_wa_file])
    ([-r|-recurse] | [-nr|-norecurse|-no_recurse])
    [-if|-ignore_files|-ignore_types file_type,...] file_spec...
```

`-cu|-consider_uncontrolled`

Specifies that uncontrolled files are removed from the work area. If the `reconcile_save_uncontrolled` option is set, the files are moved to the wastebasket. If neither `-cu|-consider_uncontrolled` or `-if|-ignore_files|-ignore_uncontrolled` is specified, the default is to ignore uncontrolled files

`file_spec`

Specifies the file or directory to be reconciled.

`-if|-ignore_files|-ignore_types file_type,...`

See [-if|-ignore\\_files|-ignore\\_types file\\_type,...](#).

`-imwaf|-ignore_missing_wa_file`

Specifies that files that are missing from the work area should be ignored and not recreated from the database. This is the default if `-mwaf|-missing_wa_file` is not specified.

`-iu|-ignore_uncontrolled`

Specifies to ignore uncontrolled files during reconcile. If neither `-cu|-consider_uncontrolled` or `-if|-ignore_files|-ignore_uncontrolled` is specified, the default is to ignore uncontrolled files.

---

`-mwaf|-missing_wa_file`

Specifies that missing work area files should be processed by unusing the corresponding members from the project in the database. The objects will not be deleted from the database. The default is to ignore missing work area files.

`-nr|-no_recurse`

See [-nr|-no\\_recurse](#).

*project\_spec*

Specifies the project to be reconciled.

`-r|-recurse`

See [-r|-recurse](#).

`-uwa|-update_wa`

See [-uwa|-update\\_wa](#).

## Examples

- Reconcile the directory `src` in `proj1`, update the work area from database, and check for missing files.

Windows:

```
ccm reconcile -missing_wa_file -update_wa c:\users\john\ccm_wa\proj1-1\src
```

UNIX:

```
ccm reconcile -missing_wa_file -update_wa /users/john/ccm_wa/proj1-1/src
```

- Reconcile the project `proj1` and subprojects, updating the database from the work area, checking for uncontrolled files.

```
ccm reconcile -recurse -consider_uncontrolled -update_db -project proj1-1
```

## Related topics

- [work\\_area command](#)

---

## Synchronizing the database with changes from a work area

This subcommand updates the database from changes made in a work area. For a *working* or *visible* project, only the owner of the project can perform the operation. For a build management project, you must be a build manager to perform this operation. The work area must be visible and modifiable by you.

```
ccm rwa|recon|reconcile -udb|-update_db -p|-project [-t|-task task_spec]
    ([-cu|-consider_uncontrolled] | [-iu|-ignore_uncontrolled])
    ([-mwaf|-missing_wa_file] | [-imwaf|-ignore_missing_wa_file])
    ([-r|-recurse] | [-nr|-norecurse|-no_recurse])
    [-if|-ignore_files|-ignore_types file_type,...] project_spec...
ccm rwa|recon|reconcile -udb|-update_db [-t|-task task_spec]
    ([-cu|-consider_uncontrolled] | [-iu|-ignore_uncontrolled])
    ([-mwaf|-missing_wa_file] | [-imwaf|-ignore_missing_wa_file])
    ([-r|-recurse] | [-nr|-norecurse|-no_recurse])
    [-if|-ignore_files|-ignore_types file_type,...] file_spec...
```

`-cu|-consider_uncontrolled`

Specifies that uncontrolled files are brought under source control and created as objects in the database, copying the file contents from the work area. If neither `-cu|-consider_uncontrolled` or `-if|-ignore_files|-ignore_uncontrolled` is specified, the default is to ignore uncontrolled files.

*file\_spec*

Specifies the file or directory to be reconciled.

`-if|-ignore_files|-ignore_types file_type,...`

See [-if|-ignore\\_files|-ignore\\_types file type,...](#).

`-imwaf|-ignore_missing_wa_file`

Specifies that files that are missing from the work area should be ignored and that the corresponding members of the project should not be removed or deleted. This is the default if `-mwaf|-missing_wa_file` is not specified.

`-iu|-ignore_uncontrolled`

Specifies to ignore uncontrolled files during reconcile. If neither `-cu|-consider_uncontrolled` or `-if|-ignore_files|-ignore_uncontrolled` is specified, the default is to ignore uncontrolled files.

---

`-mwaf` | `-missing_wa_file`

Specifies that missing work area files should be recreated from the corresponding objects in the database. The default is to ignore missing work area files.

`-nr` | `-no_recurse`

See [-nr|-no\\_recurse](#).

`-p` | `-project` *project\_spec*

Specifies the project to be reconciled.

`-r` | `-recurse`

See [-r|-recurse](#).

`-t` | `-task` *task\_spec*

Specifies the task that will be associated with any new files or directories created or checked out by reconcile. If not specified, the current task is used by default. You can set the *task\_spec* to a single task.

`-udb` | `-update_db`

See [-udb|-update\\_db](#).

### **Example**

- Reconcile the file `foo.c` by updating the database from the work area.

```
ccm reconcile -update_db foo.c-1:csrc:1
```

### **Related topics**

- [work\\_area command](#)

---

## Description and uses

The `reconcile` command compares the files in your work area with your database files. Discrepancies between the work area contents and the database are called work area conflicts. The `reconcile` command identifies these work area conflicts and resolves them to make your work area consistent with the database.

Work area conflicts occur in the following cases:

- You modified a file in your work area, whether or not it was checked out.
- You changed the database copy of a file from another work area and you changed the same file in this work area.
- You changed a file in the database, but the work area being updated was not available to update.
- You created a file in the work area, but did not place it under source control.
- You checked in a file from another work area, but the work area was not available to update with changes.
- You removed a file from the work area, but did not delete it from your project.

Additional errors can occur with controlled links and symbolic links and the work area paths. You must manually resolve these types of conflicts.

A few other ways to use this command with files that are checked out include:

- If your work area is on a laptop and you are able to work disconnected from Rational Synergy, you can use the `reconcile` command to bring your work area and the database back in sync.
- On UNIX, if a tool you are using breaks the links between an objects you are modifying and the Rational Synergy database, the `reconcile` command reconciles the changes and reestablishes the links.

For example, if you do not have a Rational Synergy session up and you need to modify an object that is not checked out, you can change it in your work area then update the Rational Synergy database later. Do this by resetting the Read Only attribute on the file and modifying it. Later, when you start a Rational Synergy session, you can use the `reconcile` command to update your database with the work area changes.

**Note** To stop a reconcile from the CLI, enter `<CTRL+C>` at any time.

When you stop the reconcile from the CLI, you will receive a message stating that errors may occur in your work area. The errors will not occur until you try to use the work area. To avoid problems, reconcile the work area completely before you use it.

Some operations perform some reconcile actions automatically:



- 
- When a file is checked in and a context project was available, the corresponding work area is examined for work area conflicts, and where possible, changes to the work area are used to update the database automatically.
  - When a modifiable file is changed in a work area, and a different version is used as a result of the `ccm use` or `ccm update_members` command, the database is updated with the changed file contents from the work area.
  - When a static file is changed in a work area and then checked out, the updated work area contents are used to update the checked out file.
  - When the database is updated with new contents from a work area file, any projects that are modifiable by you that use that file and that have work areas that are visible and modifiable will be updated with the new contents. This occurs when the database is either updated explicitly by a `ccm reconcile` command, or automatically as part of other operations.

## relate command

See [Description and uses](#) for details. The `relate` command supports the following subcommands:

- [Creating a relationship from one object to another](#)
- [Showing relationships to and from an object](#)

---

## Creating a relationship from one object to another

```
ccm relate -n|-name relationship_name -f|-from from_object_spec
           -t|-to to_object_spec
```

**-f|-from *from\_object\_spec***

Specifies the object from which the new relation will be created. You can set the *from\_object\_spec* to a single object.

**-n|-name *relationship\_name***

Specifies the name of the new relation to create.

**t|-to *to\_object\_spec***

Specifies the object to which the new relation is created. You can set the *to\_object\_spec* to a single object.

## Examples

- Make `clear-2` a successor to `clear-1`.

```
ccm relate -n successor -f clear-1 -t clear-2
```

- Link version 5.1.1 of `print.c` to version 6.

```
ccm relate -name successor -from print.c-5.1.1:csrc:1 -to print.c-6:csrc:1
```

## Related topics

- [Showing relationships to and from an object](#)

---

## Showing relationships to and from an object

```
ccm relate -s|-sh|-show [-l] [-fmt|-format format] [-nf|-noformat]
           [-sep|-separator separator] ( [-sby|-sortby sortspec] |
           [-ns|-nosort|-no_sort] ) [-gby|-groupby groupformat]
           [-n|-name relationship_name] [-f|-from from_object_spec]
           [-t|-to to_object_spec]
```

`-ch|-column_header`

Specifies to use a column header in the output format. See [-ch|-column\\_headers](#) for details.

`-fmt|-format format`

Specifies the command output format. See [-f|-format](#) for details.

A keyword can be built-in (`%fullname`, `%displayname`, `%objectname`) or the name of any existing attribute such as `%modify_time` or `%status`.

See [Built-In keywords](#) for a list of keywords.

`-f|-from from_object_spec`

Specifies the object from which relationships are listed. You can set the `from_object_spec` to a single object.

`-gby|-groupby groupformat`

Specifies how to group the command output. See [-gby|-groupby](#) for details.

`-l`

Specifies that a default long format should be used.

`-n|-name`

Specifies the name of the relationship to show.

`-nch|-nocolumn_header`

Specifies not to use a column header in the output format. See [-ch|-column\\_headers](#) for details.

`-nf|-noformat`

Specifies not to use column alignment. See [-nf|-noformat](#) for details.

---

`-ns` | `-nosort` | `-no_sort`

Specifies that the command's output will not be sorted. See [-ns|-nosort](#) for details.

`-sby` | `-sortby` *sortspec*

Specifies how to sort the command output. See [-sby|-sortby](#) for details.

`-sep` | `-separator` *separator*

Used only with the `-f` | `-format` option. Specifies a different separator character. See [-sep|-separator](#) for details.

`-s` | `-show`

Show the relationships among the specified objects.

`-t` | `-to` *to\_object\_spec*

Specifies the object to which relationships are shown. The *to\_object\_spec* must be an *object\_spec*; you can set it to a single object.

## Related topics

- [Creating a relationship from one object to another](#)

---

## Description and uses

The `relate` command adds a relationship (*relation\_name*) between *file\_spec1* and *file\_spec2*, or shows the relationship with the specified data.

More relationships are predefined in Rational Synergy. See "Relationships" in [Rational Synergy CLI Help, Traditional mode](#) for a table showing these relationships. However, you can define new relationships using the `relate` command.

## release command

See [Description and uses](#) for details. The `release` command supports the following subcommands:

- [Creating a release](#)
- [Deleting a release](#)
- [Listing releases](#)
- [Modifying a release](#)
- [Setting the controlling database for a release](#)
- [Showing the process rules for a release](#)
- [Showing release information](#)

---

## Creating a release

This subcommand creates a new release definition. You must be in the *build\_mgr* or *ccm\_admin* role to use this subcommand.

To create a release for a new application or component, you can either create a release with a unique component name or not use a component name.

To create a release based on a previous release, use the *-from* option. By default, the new release is created using process rules and other properties that correspond to those from the previous release. It is also a successor of the release on which it is based. The previous release is used as the baseline release.

```
ccm release -c|-create [-from release_spec] [-bl|-baseline release_spec]
    [-desc|-description description]
    [-desc_edit|-descriptionedit|-description_edit]
    [-desc_file|-descriptionfile|-description_file file_path]
    [-manager manager] ([-active] | [-inactive])
    ([-allow_dcm_transfer] | [-noallow_dcm_transfer])
    [-allow_parallel_check_out] [-noallow_parallel_check_out]
    [-allow_parallel_check_in] [-noallow_parallel_check_in]
    [-groups groups] ([-included_releases included_releases] |
    [-included_releases_file included_releases_file])
    [-phase phase] ([-process process_spec] |
    [(-process_rule process_rule_spec)...])
    ([-cct|-create_component_tasks] | [-nocct|-nocreate_component_tasks])
    release_spec
```

*-active*

Specifies that the release is active. This is the default.

*-allow\_dcm\_transfer*

Specifies that the release is eligible for DCM replication if included by a transfer set's release scope and query. When creating a release for a new component, this defaults to *true*. When creating a release based on a previous release, the setting for the release, on which the new release is based, is the default.

*-allow\_parallel\_check\_in*

Specifies that parallel check in for objects with this release is permitted. This is the default when creating a new release. When creating a release based on a previous release, the setting for the release, on which the new release is based, is the default. Note that the combination of parallel check in but no parallel check out is invalid.



---

`-allow_parallel_check_out`

Specifies that parallel check out for objects with this release is permitted. This is the default when creating a new release. When creating a release based on a previous release, the setting for the release, on which the new release is based, is the default. Note that the combination of parallel check in, but no parallel check out is invalid.

`-baseline release_spec`

Specifies the release used as the baseline for the new release. When creating a release based on a previous release, that previous release is used as the baseline by default. When creating a release for a new component, the default baseline release is blank.

`-cct|-create_component_tasks`

Specifies to create corresponding component tasks when baselines are created for the release definition being created. For example, use this option if you are creating a release definition and know that your team will require component tasks. When you create the baseline, component tasks will be created automatically. This is the default.

`-desc|-description description`

Specifies the description for the release. You can use escape sequences to include newlines and other characters. Alternatively, use the `-description_file` or `-description_edit` for specifying multi-line descriptions. If `-description`, `-description_file` and `-description_edit` are all used together, the description is formed by taking the `-description` option value, appending the description read from the file specified by `-description_file`, and then the current default text editor is launched showing that comment. The text saved from the editor is then used for setting the description.

`-desc_edit|-description_edit`

Invokes the current text editor to allow the release description to be interactively edited or composed. The saved result from the text editor is used to set the description. See [-desc|-description description](#).

`-desc_file|-description_file file_path`

Specifies a path to a file containing a description.

---

`-from release_spec`

Specifies the release on which the new release is based. When creating a release based on a previous release, many of the new release's settings are copied from the previous release. The previous release is used as the baseline release by default.

`-groups groups`

Specifies the groups that may modify the new release or create following releases from it. When creating a release based on a previous release, the new release uses the same groups as the release on which it is based by default. The groups value is a list of one or more group names separated by spaces commas.

`-inactive`

Specifies that the new release is inactive. Inactive releases cannot be used by developers for development work. By default, new releases are created as active releases.

`-included_releases included_releases`

Specifies one or many releases to be included in the release. This string supports multiple releases separated by a comma, and optionally, spaces. The comma is required; however, releases with leading or trailing spaces are not supported. Alternatively, you can use the `included_releases_file` option and enter data from a file.

Included releases are only used by default for object status-based updated. They are used for weighting the selection scoring while update members is running.

`-included_releases_file file_path`

Specifies a path to a file containing the releases to be included.

`-manager manager`

Specifies the product or component manager for the release. The default on create is the user who is creating the release definition, and can be only a one-line string.

`-noallow_dcm_transfer`

Specifies that the release is not eligible for DCM replication. When creating a release for a new component, the release is eligible for replication by default. When creating a release based on a previous release, the setting for the release, on which the new release is based, is the default.

---

`-noallow_parallel_check_in`

Specifies that parallel check in for objects with this release is not permitted. Parallel check in is allowed by default when creating a new release. When creating a release based on a previous release, the setting for the release, on which the new release is based, is the default. Note that the combination of parallel check in but no parallel check out is invalid.

`-noallow_parallel_check_out`

Specifies that parallel check out for objects with this release is not permitted. Parallel check out is permitted by default when creating a new release. When creating a release based on a previous release, the setting for the release, on which the new release is based, is the default. Note that the combination of parallel check in, but no parallel check out is invalid.

`-nocct | -nocreate_component_tasks`

Specifies that component tasks will not be automatically created when baselines are published for the release definition being created. You can create component tasks manually after a baseline is published for the release by using the [Creating component tasks for a baseline](#) subcommand.

`-phase phasename`

Specifies the release phase for the new release. By default, a new release is created with release phase `New`. The valid release phases are defined in the model attribute [release phase list](#). The factory default values are `New`, `Requirements Definition`, `Function Definition`, `Implementation`, `Validation`, and `Released`. The specified value must match one of the valid release phase values and is case sensitive.

`-process process_spec`

Specifies a process for a release as it is being created. The release-specific process rules associated with the generic process rules for the specified process are associated with the new release. If any of the release-specific process rules do not exist, they will be created.

`release_spec`

Specifies the name of the new release to create.

---

## Examples

- Create a new release `alphabets/2.0`, using the properties from `alphabets/1.0`.

Windows:

```
ccm release -create "alphabets/2.0" -from "alphabets/1.0" -  
description_file c:\alphabets_2\features.txt
```

UNIX:

```
ccm release -create "alphabets/2.0" -from "alphabets/1.0" -  
description_file /usr/john/alphabets_2/features.txt
```

- Create a release for a new component (not based on an existing release) named `harmony/1.0`.

```
ccm release -create "harmony/1.0" -desc "new product line to integrate  
X and Y" -manager "sue" -active -noallow_dcm_transfer
```

---

## ***Deleting a release***

This subcommand deletes one or more release definitions. To delete a release used by projects, files, folders or baselines, use the `-force` option. If the `-force` option is omitted, the command only succeeds if the release is only referenced by other releases or by process rules. If the release has any successor releases, the history for the release is collapsed. You must be in the `build_mgr` or `ccm_admin` role to use this subcommand.

```
ccm release -d|-delete [-force] release_spec...
```

`-force`

Specifies that the release should be deleted even if the release is referenced from objects other than releases or process rules. If projects or files use that release and `-force` is not specified, the deletion will fail.

`release_spec`

Specifies the releases to delete. See [Release specification](#) for details.

## ***Example***

- Delete the release definition for `sue/6.5`, whether or not objects use the specified release.

```
ccm release -delete -force sue/6.5
```

---

## Listing releases

This subcommand lists the releases matching any specified criteria. If no criteria are specified, all releases are listed.

```
ccm release -l|-list ([-active] | [-inactive])
                [(-component component_name)...] [-f|-format format] [-nf|-noformat]
                ([-ch|-column_header] | [-nch|-nocolumn_header])
                [-sep|-separator separator] ([-sby|-sortby sortspec] |
                [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat] [-u|-unnumbered]
```

**-active**

Specifies that only active releases are listed. If neither `-active` nor `-inactive` are specified, both active and inactive releases are listed.

**-ch|-column\_header**

Specifies to use a column header in the output format. See [-ch|-column headers](#) for details.

**-component *component\_name***

Specifies that only releases for a specific component name are listed. If a component name is not specified, no releases are listed.

**-f|-format *format***

Specifies the command output format. See [-f|-format](#) for details.

A keyword can be built-in (`%fullname`, `%displayname`, `%objectname`) or the name of any existing attribute such as `%modify_time` or `%status`.

See [Built-In keywords](#) for a list of keywords.

**-gby|-groupby *groupformat***

Specifies how to group the command output. See [-gby|-groupby](#) for details.

**-inactive**

Specifies that only inactive releases are listed. If neither `-active` nor `-inactive` are specified, then both active and inactive releases are listed.

**-nch|-nocolumn\_header**

Specifies not to use a column header in the output format. See [-nch|-nocolumn headers](#) for details.

---

`-nf|-noformat`

Specifies not to use column alignment. See [-nf|-noformat](#) for details.

`-ns|-nosort|-no_sort`

Specifies that the command's output will not be sorted. See [-ns|-nosort](#) for details.

`-sby|-sortby sortspec`

Specifies how to sort the command output. See [-sby|-sortby](#) for details.

`-sep|-separator separator`

Used only with the `-f|-format` option. Specifies a different separator character. See [-sep|-separator](#) for details.

`-u|-unnumbered`

Suppresses automatic numbering of the command's output (that is, the output is unnumbered). See [-u|-unnumbered](#) for details.

### **Example**

- List releases.

```
ccm release -list -active -component a
```

```
1) a/1.0
```

```
ccm release -list -inactive -component b
```

```
1) b/1.0
```

---

## Modifying a release

This subcommand modifies one or more releases. You must be in the *build\_mgr* or *ccm\_admin* role to use this subcommand.

```
ccm release -m|-modify [-bl|-baseline release_spec]  
    [-desc|-description description]  
    [-desc_edit|-descriptionedit|-description_edit]  
    [-desc_file|-descriptionfile|-description_file file_path]  
    [-manager manager] ([-active] | [-inactive])  
    ([-allow_dcm_transfer] | [-noallow_dcm_transfer])  
    [-allow_parallel_check_out] [-noallow_parallel_check_out]  
    [-allow_parallel_check_in] [-noallow_parallel_check_in]  
    [-groups groups] ([-included_releases included_releases] |  
    [-included_releases_file included_releases_file])  
    [-phase phase]  
    ([-apr|-add_process_rule|-add_process_rules process_rule_spec)...  
    [-cpr|-clear_process_rules]] | [(-rpr|-remove_process_rule|  
    -remove_process_rules process_rule_spec)...])  
    ([-cct|-create_component_tasks] | [-nocct|-nocreate_component_tasks])  
    release_spec...
```

-active

Sets the release to be active.

-allow\_dcm\_transfer

Specifies that the releases should be set as eligible for DCM replication if included by a transfer set's release scope and query.

-allow\_parallel\_check\_in

Specifies that parallel check in for objects with this release is permitted. Note that the combination of parallel check in but no parallel check out is invalid.

-allow\_parallel\_check\_out

Specifies that parallel check out for objects with this release is permitted. Note that the combination of parallel check in but no parallel check out is invalid.

-apr|-add\_process\_rule|-add\_process\_rules *process\_rule\_spec*

Adds the specified process rule to each of the releases specified by the arguments. See [Process rule specification](#) for details.



---

`-baseline release_spec`

Sets the baseline release for the releases being modified. See [Release specification](#) for details.

`-cct|-create_component_tasks`

Specifies to create corresponding component tasks when baselines are created for the release definition being modified. For example, use this option if you are modifying a release definition and know that your team will require component tasks. When you modify the baseline, component tasks will be created automatically.

`-cpr|-clear_process_rules`

Clears any existing process rules before adding a process rule (see [-apr|-add process rule|-add process rules process rule spec](#)), and sets an absolute set of process rules.

`-desc|-description description`

See [-desc|-description description](#).

`-desc_edit|-description_edit`

See [-desc\\_edit|-description\\_edit](#).

`-desc_file|-description_file file_path`

See [-desc\\_file|-description\\_file file\\_path](#).

`-groups groups`

Specifies the groups that may modify the new release or create following releases from it. The *groups* value is a list of one or more group names separated by spaces commas.

`-inactive`

Specifies that the releases being modified shall be set to inactive. Inactive releases cannot be used by developers for development work.

`-included_releases included_releases`

See [-included\\_releases included\\_releases](#).

---

`-included_release_file file_path`

See [-included\\_releases file file\\_path](#).

`-manager manager`

See [-manager manager](#).

`-m|-modify`

Modifies the specified release.

`-noallow_dcm_transfer`

Specifies that the releases should be set as ineligible for DCM replication.

`-noallow_parallel_check_in`

Specifies that parallel check in for objects with this release is not permitted. Note that the combination of parallel check in but no parallel check out is invalid.

`-noallow_parallel_check_out`

Specifies that parallel check out for objects with this release is not permitted. Note that the combination of parallel check in but no parallel check out is invalid.

`-nocct|-nocreate_component_tasks`

Specifies that component tasks will not be automatically created when baselines are published for the release definition being modified. You can create component tasks manually after a baseline is published for the release by using the [Creating component tasks for a baseline](#) subcommand.

`-phase phasename`

Specifies that the release phase should be set for the specified releases. The valid release phases are defined in the model attribute [release\\_phase\\_list](#). The default values are **New**, **Requirements Definition**, **Function Definition**, **Implementation**, **Validation**, and **Released**.

`release_spec`

Specifies the releases to modify. See [Release specification](#) for details.

---

`-rpr | -remove_process_rule | -remove_process_rules process_rule_spec`

Removes the specified process rule from each of the releases specified by the arguments. See [Process rule specification](#) for details.

### **Example**

- Modify the release information to set a new description, a new manager and a release in the **Implementation** phase.

```
ccm release -modify -description "version a of release 1.0 without
graphics capability" -manager jane -phase Implementation client/1.0a
```

### **Related topics**

- [Showing the process rules for a release](#)
- [Showing release information](#)

---

## Setting the controlling database for a release

This subcommand sets the controlling database for one or more releases. You must be in the *build\_mgr*, *dcm\_mgr*, or *ccm\_admin* role to use this subcommand.

```
ccm release -cdb|-controlling_database -local -component component_name
ccm release -cdb|-controlling_database -handover database_spec
           -component component_name
ccm release -cdb|-controlling_database -accept database_spec
           -component component_name
ccm release -cdb|-controlling_database -local release_spec...
ccm release -cdb|-controlling_database -handover database_spec
           release_spec...
ccm release -cdb|-controlling_database -accept database_spec
           release_spec...
```

-component *component\_name*

Specifies to set the controlling database for releases with the specified component name. An empty string applies the change to releases with the null component name.

*database\_spec*

Specifies that DCM updates are accepted from a specific database. See [Database specification](#) for details. See "Setting up a DCM database" in [Rational Synergy Distributed](#).

-handover *database\_spec*

Specifies to hand over control of the release to the specified database. You can set the *database\_spec* to a single database definition. See [Database specification](#) for details. You can only use this option when the release is locally controlled. See "Setting up a DCM database" in *Rational Synergy Distributed*.

-local

Specifies that control of the database is to be handled by the local database. The object is no longer updated by DCM replication from another database. See "Setting up a DCM database" in *Rational Synergy Distributed*.

### Example

- Hand over control of a locally-controlled release definition to a database whose ID is A1.

```
ccm release -controlling_database -local -handover A1 -component
releasename
```

---

## Showing the process rules for a release

This subcommand shows the process rules for one or more releases.

```
ccm release -s|-sh|-show ((pr|prs|process_rules) |
    (apr|aprs|available_process_rules) | (upr|uprs|unused_process_rules))
    [-f|-format format] [-nf|-noformat]
    ([-ch|-column_header] | [-nch|-nocolumn_header])
    [-sep|-separator separator] ([-sby|-sortby sortspec] |
    [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat] [-u|-unnumbered]
    release_spec...
```

[-ch|-column\\_header](#)

[-f|-format \*format\*](#)

[-gby|-groupby \*groupformat\*](#)

[-nch|-nocolumn\\_header](#)

[-nf|-noformat](#)

[-ns|-nosort|-no\\_sort](#)

[-sby|-sortby \*sortspec\*](#)

[-sep|-separator \*separator\*](#)

`-show process_rules|pr|prs`

Shows the current valid process rules for each of the specified arguments. This is numbered by default for each argument and sets the selection set. (text inset)

`-show available_process_rules|aprs|apr`

Show the available process rules for each release. The available process rules are all the process rules that are available for use in the release including those that may already be in use. This is numbered by default for each argument and sets the selection set.

---

`-show upr|uprs|unused_process_rules`

Show the available unused process rules for the release. These are the available process rules for a release excluding those currently used as valid process rules. This is numbered by default for each argument and sets the selection set.

[-u|-unnumbered](#)

### Related topics

- [Listing releases](#)
- [Showing release information](#)

---

## Showing release information

```
ccm release -s|-sh|-show (i|info|information) -f|-format format
    [-nf|-noformat] ([-ch|-column_header] | [-nch|-nocolumn_header])
    [-sep|-separator separator] release_spec...
ccm release -s|-sh|-show (i|info|information) [-v|-verbose]
    release_spec...
```

[-ch|-column\\_header](#)

[-f|-format \*format\*](#)

[-nch|-nocolumn\\_header](#)

[-nf|-noformat](#)

[-sep|-separator \*separator\*](#)

[-v|-verbose](#)

Shows the verbose information format.

## Examples

- View information about release `client/3.5`.  

```
ccm release -show information client/3.5
```
- View information about release `a/1.0`:  

```
ccm release -show information a/1.0
```

---

## Description and uses

Use the `release` command to create, modify, delete, and show release information.

A release is used for managing how projects are updated and to support parallel development. Each release has a unique name and changes are associated with that name. A name may consist of a component name, a release delimiter, and a component release. For example, the name **webapp/3.0** has a component name of **webapp** and represents release **3.0** of that component. The default release delimiter is `/`. The maximum length of a component name is 64 characters. The maximum length of a component release is 32 characters. An alternative form of a name is just to use the component release, such as **2.0**. Such releases are said to use a null component name. The advantage of using a component name in your names is that it keeps releases for the same component logically related and avoids the need for manual naming conventions.

A release definition has a number of important properties. The baseline release is used when updating projects. If a process rule uses the `%baseline_release` keyword in its baseline release-purpose list, this refers to the baseline release of the release associated with the project or project grouping being updated. The included releases are used for selection rule scoring, primarily with object status-based CM. A release may be marked as inactive in order to prevent the release being used by developers for further development.

Component names and component releases must **not** start with the following characters:

`/ \ ' " : * ? [ ] @ % - + ~ space, tab`

Second and subsequent characters **cannot** include the following:

`/ \ ' " : * ? [ ] @ %`

Note that the component name and component release can contain the version delimiter character (by default `-`) if it is not one of the restricted characters.

Whenever an object is checked out, Rational Synergy automatically copies the release from the current task to the new object.

You must be working in the required role to perform a release operation:

- Any user can show or list releases.
- A build manager or a user in the `ccm_admin` role can create, modify, or delete a release definition.
- A user in the `ccm_admin` role can change the release delimiter.
- A build manager or a user in the `ccm_admin` role can rename a release if only the release definition and its associated process rules will be updated, and you must be in the `ccm_admin` role if other associated objects will be updated.



---

---

## save\_offline\_and\_delete command

See [Description and uses](#) for details. The `save_offline_and_delete` command supports the following subcommands:

- [Creating a preview object list for deletion](#)
- [Deleting using a preview object list](#)
- [Deleting using a scope](#)

### ***Prerequisites***

To save objects offline, the current database must be initialized for DCM and a DCM license must be available.

---

## Creating a preview object list for deletion

This subcommand performs a preview of objects to be deleted by a soad scope and creates a preview object list that deletes the objects found by the preview. The query selection set matches the results from the preview object list. When you run this command, the preview object list is cleared. It is also cleared when you delete an object by using the preview object list or by using a scope command.

This subcommand populates the object list with results of the preview so you can use it with subsequent commands, such as `ccm query`. You can use the results for the current session only.

The role required to run the command depends on the specified scope. Each scope defines the roles that are permitted to execute or preview that scope.

**Note** The preview results are not overwritten if you perform a query after a preview and before you use the object list for a `ccm soad -delete` command.

```
ccm soad|save_offline_and_delete -preview -scope scope_name
  [-so|-save_offline] [-f|-format format] [-nf|-noformat]
  ([-ch|-column_header] | [-nch|-nocolumn_header])
  [-sep|-separator separator] ([-sby|-sortby sortspec] |
  [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat]
  [-u|-unnumbered] [-v|-verbose]
  [argument...]
```

`-ch|-column_header`

Specifies to use a column header in the output format. See [-ch|-column\\_headers](#) for details.

`-f|-format format`

Specifies the command output format. See [-f|-format](#) for details.

`-gby|-groupby groupformat`

Specifies how to group the command output. See [-gby|-groupby](#) for details.

`-nch|-nocolumn_header`

Specifies not to use a column header in the output format. See [-nch|-nocolumn\\_headers](#) for details.

`-nf|-noformat`

Specifies not to use column alignment. See [-nf|-noformat](#) for details.

---

`-ns|-no_sort`

Specifies that the command's output will not be sorted. See [-ns|-nosort](#) for details.

`-sby|-sortby sortspec`

Specifies how to sort the command output. See [-sby|-sortby](#) for details.

`-sep|-separator separator`

Used only with the [-f|-format](#) option. Specifies a different separator character. See [-sep|-separator](#) for details.

`-scope scope_name [argument...]`

Specifies the scope (modified query) used to save offline or delete objects.

Arguments are required only if appropriate for the specified scope. For example, the scope "My working projects and products for a specified release" requires that you specify a release value, which is an argument.

You must specify the arguments in the order used in the scope definition.

For a detailed discussion of scopes, see "SOAD scopes" in [Synergy CLI Help, Traditional mode](#).

`-so|-save_offline`

Creates an object list that preserves in the database any object that would be the last remaining version of that object name and instance.

`-u|-unnumbered`

Suppresses automatic numbering of the command's output. See [-u|-unnumbered](#) for details.

`-v|-verbose`

Generates messages detailing why objects are being included in or excluded from the list.

## **Example**

- Preview the deletion of objects found by the scope **Baselines marked for deletion and related projects and products**.

```
ccm soad -preview -scope "Baselines marked for deletion and related
projects and products"
```

---

## Related topics

- [Deleting using a preview object list](#)
- [Deleting using a scope](#)

---

## **Deleting using a preview object list**

This subcommand deletes the objects in the preview object list that was created by a previous command to create a preview object list for deletion. The preview object list is cleared on completion.

The objects are saved before being deleted if you used the `-so` option on a previous `ccm soad -preview` command.

The deletion preserves any object in the database that would be the last remaining version of that object name and instance.

```
ccm soad|save_offline_and_delete -delete [-path soad_path]
                                         [-pn|-package_name package_name] [-v|-verbose]
```

`-path soad_path`

Specifies the path to the DCM package using the previously specified path.

If you have not saved a package previously, you must specify the path.

**Note** The path must be visible to the engine and writable by `ccm_root`.

`-pn|-package_name package_name`

Specifies the name of the DCM package to which the objects are saved. The default name is "Save Offline and Delete saved on %date."

**Note** Include the `%date` keyword in the name if you define your own package name. This helps ensure that you can differentiate between packages created using the same scope.

`-v|-verbose`

Generates messages detailing why objects are being included in or excluded from the list.

## **Example**

- Delete the objects found by a previous `soad` preview.

```
ccm soad -delete
```

## **Related topics**

- [Creating a preview object list for deletion](#)
- [Deleting using a scope](#)

---

## Deleting using a scope

Use this subcommand to delete objects specified by a scope without performing a preview. Any previous preview object list is cleared on completion.

**Caution** Performing a preview before deleting the objects means that you can check which objects will be deleted before deleting them. This is essential if you are working in the *ccm\_admin* role. This command does not perform a preview; you will not know which objects will be deleted until the command has started.

```
ccm soad|save_offline_and_delete -delete -scope scope_name
    [-so|-save_offline] [-path soad_path]
    [-pn|-package_name package_name] [-v|-verbose]
    [argument...]
```

`-path soad_path`

Specifies the path to the DCM package using the previously specified path.

If you have not saved a package previously, you must specify the path.

**Note** The path must be visible to the engine and writable by *ccm\_root*.

`-pn|-package_name package_name`

Specifies the name of the DCM package to which the objects are saved. The default name is "Save Offline and Delete saved on %date."

**Note** Include the `%date` keyword in the name if you define your own package name. This helps ensure that you can differentiate between packages created using the same scope.

`-scope scope_name [argument...]`

Specifies the scope (modified query) used to save offline or delete objects.

Arguments are required only if appropriate for the specified scope. For example, the scope "My working projects and products for a specified release" requires that you specify a release value, which is an argument.

You must specify the arguments in the order used in the scope definition.

For a detailed discussion of scopes, see "SOAD scopes" in [Synergy CLI Help, Traditional mode](#).

---

-v | -verbose

Generates messages detailing why objects are being included in or excluded from the list.

## Examples

- Delete objects found by the **Baselines marked for deletion and related projects and products** scope without performing a preview.

```
ccm soad -scope "Baselines marked for deletion and related projects and products"
```

## Related topics

- [Creating a preview object list for deletion](#)
- [Deleting using a preview object list](#)



---

## Description and uses

The Save Offline and Delete (SOAD) utility deletes objects from a Synergy database by using a scope. The scope determines what objects are considered for deletion, which associated objects can also be deleted, and rules for when objects should not be deleted. Synergy ships with predefined scopes that represent typical reasons for deleting objects. Users can also create their own custom scopes. See the [soad\\_scope command](#) for details.

You can delete objects from a Synergy database to:

- reduce the size of the database as part of disk space management, or to speed database backups.
- remove *working* or *prep* projects that are no longer in use.
- remove unwanted history or baselines because they are not required.

The following are examples of the types of data that you can delete:

- Unwanted **Insulated Development** projects for specific developers
- Unwanted **Integration Testing** projects
- Old, static project hierarchies and the old files associated with them, where the hierarchies have been superseded by later released versions
- Unused old products
- Baselines that have been marked for deletion
- **Integration Testing** baselines for an old release.

SOAD also saves objects offline before they are deleted so that the objects can be restored in a Synergy database at a later date. Users in the *ccm\_admin* role can use this feature in databases that are DCM initialized. The objects to be saved are stored in a save offline package that is similar to a DCM package. The objects are restored using a DCM receive operation. See [Receiving packages](#) for details.

The scope defines which role a user must be in before they can use that scope. In addition, normal Synergy security rules apply about when an object can be deleted. You can delete any *working*-state project or object owned by you. If you are working as a build manager, you can also delete *prep* projects and objects. If you are in the *ccm\_admin* role, you can delete objects in any non-working state or objects owned by other users. SOAD also has a number of built-in safety measures to prevent certain types of objects, such as type definitions and other administrative data, from being deleted.

Use the `soad` command to do the following.

- Preview a delete by using a scope. The preview shows what objects would be selected for deletion without making any changes to the database. The advantage is that you can check that the preview results are what you expect before deleting the objects.
- Delete the objects found by a previous preview.

- 
- Delete using a scope without a preview. This allows you to perform the deletion without having to see a preview of the results. The advantage is speed, but you must be certain that you do not need to preview the objects. The disadvantage is that you will not be able to verify the objects to be deleted.

### **Related topics**

- [soad\\_scope command](#)



---

## soad\_scope command

See [Description and uses](#) for details. The `soad_scope` command supports the following subcommands:

- [Creating a scope](#)
- [Deleting a scope](#)
- [Listing scopes](#)
- [Modifying a scope](#)
- [Showing a scope](#)

---

## Creating a scope

This subcommand creates a soad scope.

**Caution** When you create a new scope, test the scope by performing a preview using that scope. Ensure that the scope has the exclusion rules set to prevent deleting data that should be kept.

```
ccm soad_scope|save_offline_and_delete_scope -c|-create [-roles role]
    [-parameters parameters] ([-object four_part_object_name] |
    [-query query]) [-expand|-expansion_rules expansion_rules]
    [-exclude|-exclusion_rules exclusion_rules]
    [-exclude_query|-exclusion_query exclusion_query]
    [-pn|-package_name package_name] scope_name
```

`-exclude|-exclusion_rules exclusion_rules`

Specifies one or more exclusion rules. Exclusion rules remove related objects from the initial object list.

For example, if your query retrieves all objects for a specified release, with the release name as the first parameter (`release=%1`), you can restrict the scope by adding exclusion rules to remove from the scope folders and tasks used by other projects; tasks used by other folders or associated with other objects; baselines used by other non-static projects; and objects that are part of other saved baselines.

`-expand|-expansion_rules expansion_rules`

Specifies one or more expansion rules. Expansion rules add related objects to the initial object list.

For example, if your query retrieves all objects for a specified release, with the release name as the first parameter (`release=%1`), you can expand the scope by adding expansion rules to include the folder and tasks for a project; the tasks for a folder; and the objects for a task.

`-exclude_query|-exclusion_query exclusion_query`

Specifies a query used to remove objects from the scope.

For example, to exclude from the scope objects that have an attribute named `requirements`, specify the following query expression:

```
has_attr('requirements')
```

SOAD will add the following negated clause, wherever it evaluates an object name, query, or rule:

```
and not has_attr('requirements')
```

---

`-object four_part_name`

Specifies the name of the object used for the initial object list (for example, %1). The resulting expanded string must be a valid four-part object name.

For example, you can use the project object name, entered as the first parameter (%1), to set the initial object list to that project object name.

`-parameters parameters`

Supplies labels for arguments for the `-object`, `-query`, and `-exclude_query` and definitions.

For example, define a scope such as the following for one parameter label, Release Value, for the query used in the "All objects for specified release" scope:

```
ccm soad_scope -create "All objects for specified release"
-parameters "Release Value" -query "release='%1'" other_options
```

Next, use the scope in the following `ccm soad -delete` command, where **2.3** is the release value:

```
ccm soad -delete -scope "All objects for specified release" 2.3
```

`-pn|-package_name package_name`

Specifies the name of the DCM package to which objects are saved for the scope. The package name can include keywords.

`-query query`

Specifies the query expression that defines the initial object list.

For example, to make the initial object list include all the projects and products for a specified release for the current user, specify the following query expression:

```
(cvtype='project' or is_product=TRUE) and owner='%user' and
status='working' and release='%1'
```

`-roles role`

Specifies the role that can use the scope. By default, only users working in the `ccm_admin` role can change the scope.

`scope_name`

Specifies the scope for Save Offline and Delete.

Use only characters not restricted by the OS.

---

This name is also the file name for the scope, including spaces and other characters, converted to a URL. For example, if you name the scope `This is my test scope`, the file name created is `This%20is%20my%20test%20scope.xml`.

This name is used as the file name for the scope. Spaces and punctuation characters are converted into underscore hex encoding. For example, if you name the scope **This is my test scope**, the file name created is **This\_0020is\_0020my\_0020\_test\_0020scope.xml**.

## Related topics

- [Deleting a scope](#)
- [Listing scopes](#)
- [Modifying a scope](#)
- [Showing a scope](#)

---

### ***Deleting a scope***

This subcommand deletes a soad scope.

```
ccm soad_scope|save_offline_and_delete_scope -d|-delete scope_name
```

*scope\_name*

Specifies the name of the scope to delete.

### **Related topics**

- [Creating a scope](#)
- [Listing scopes](#)
- [Modifying a scope](#)
- [Showing a scope](#)



---

### ***Listing exclusion rules***

This subcommand lists the names of the valid exclusion rules that are available for use by a soad scope.

```
ccm soad_scope|save_offline_and_delete_scope -l|-list  
      -exclude|-exclusion_rules
```

```
-exclude|-exclusion_rules
```

Lists the exclusion rules.

### **Related topics**

- [Creating a scope](#)
- [Showing a scope](#)

---

### ***Listing expansion rules***

This subcommand lists the names of the valid expansion rules that are available for use by a soad scope.

```
ccm soad_scope|save_offline_and_delete_scope -l|-list  
-expand|-expansion_rules
```

```
-expand|-expansion_rules
```

Lists the expansion rules.

### **Related topics**

- [Creating a scope](#)
- [Showing a scope](#)

---

## ***Listing scopes***

This command lists the names of the defined soad scopes.

```
ccm soad_scope|save_offline_and_delete_scope -l|-list -scope
```

-scope

Specifies the scope that you want to list.

## **Related topics**

- [Creating a scope](#)
- [Showing a scope](#)

---

## Modifying a scope

This subcommand changes the properties of a soad scope.

```
ccm soad_scope|save_offline_and_delete_scope -m|-modify [-roles role]
    [-parameters parameters] ([-object four_part_object_name] |
    [-query query]) [-expand|-expansion_rules expansion_rules]
    [-exclude|-exclusion_rules exclusion_rules]
    [-exclude_query|-exclusion_query exclusion_query]
    [-pn|-package_name package_name] scope_name
```

`-exclude|-exclusion_rules exclusion_rules`

Specifies one or more exclusion rules. Exclusion rules remove related objects from the initial object list.

For example, if your query retrieves all objects for a specified release, with the release name as the first parameter (`release=%1`), you can restrict the scope by adding exclusion rules to remove from the scope folders and tasks used by other projects; tasks used by other folders or associated with other objects; baselines used by other non-static projects; and objects that are part of other saved baselines.

`-expand|-expansion_rules expansion_rules`

Specifies one or more expansion rules. Expansion rules add related objects to the initial object list.

For example, if your query retrieves all objects for a specified release, with the release name as the first parameter (`release=%1`), you can expand the scope by adding expansion rules to include the folder and tasks for the project; the folders' tasks; and the tasks' objects.

`-exclude_query|-exclusion_query "query_expression"`

Specifies a query used to remove objects from the scope.

For example, to exclude from the scope objects that have an attribute named `requirements`, specify the following query expression:

```
has_attr('requirements')
```

SOAD will add the following negated clause, wherever it evaluates an object name, query, or rule:

```
and not has_attr('requirements')
```

`-object four_part_name`

Specifies the name of the object used for the initial object list (for example, `%1`). The resulting expanded string must be a valid four-part object name.

---

For example, you can use the project object name, entered as the first parameter (%1), to set the initial object list to that project object name.

`-parameters parameters`

Supplies labels for arguments for the `-object`, `-query`, and `-exclude_query` and definitions.

For example, define a scope such as the following for one parameter label, Release Value, for the query used in the All objects for specified release scope:

```
ccm soad_scope -create "All objects for specified release"
-parameters "Release Value" -query "release='%1'" other_options
```

Next, use the scope in the following `ccm soad -delete` command, where **2.3** is the release value:

```
ccm soad -delete -scope "All objects for specified release" 2.3
```

`-pn|-package_name package_name`

Specifies the name of the DCM package to which objects are saved for the scope. The package name can include keywords.

`-query query`

Specifies the query expression that defines the initial object list.

For example, to make the initial object list include all the projects and products for a specified release the current user, specify the following query expression:

```
(cvtype='project' or is_product=TRUE) and owner='%user' and
status='working' and release='%1'
```

`-roles role`

Specifies the role that can use the scope. By default, only users working in the `ccm_admin` role can change the scope.

`scope_name`

Specifies the scope for Save Offline and Delete.

Use only characters not restricted by the OS.

This name is used as the file name for the scope. Spaces and punctuation characters are converted into underscore hex encoding. For example, if you name the scope

**This is my test scope**, the file name created is  
**This\_0020is\_0020my\_0020\_test\_0020scope.xml**.

---

## Related topics

- [Deleting a scope](#)
- [Listing scopes](#)
- [Showing a scope](#)

---

## Showing a scope

This subcommand shows details for soad scopes.

```
ccm soad_scope|save_offline_and_delete_scope -s|-sh|-show scope_name...
```

*scope\_name...*

Shows all scopes with the following details:

- Roles
- Parameter labels
- Object
- Query
- Expansion rules
- Exclusion rules
- Exclusion query
- Package name

You can specify multiple scope names to show.

## Example

- Show the details for the scope **Baselines marked for deletion and related projects and products**.

```
ccm soad_scope -show "Baselines marked for deletion and related  
projects and products"
```

## Related topics

- [Creating a scope](#)
- [Deleting a scope](#)
- [Listing scopes](#)
- [Modifying a scope](#)

---

## Description and uses

The `soad_scope` command edits, creates, modifies, and deletes scopes used to save objects offline and to delete objects.

You can edit, create, or delete a scope only when working in the `ccm_admin` role.



## set command

See [Description and uses](#) for details. The `set` command supports the following subcommands:

- [Setting an option](#)
- [Showing an option](#)
- [Showing options](#)

---

## Setting an option

This subcommand sets an option value. For example, to set your role to *developer*, use **ccm set role developer**. The option value must be valid for the option you are setting. See [Default options](#) for details.

```
ccm set option value
```

*option*

Specifies the name of the option to be set. See [Default settings](#) for details.

*value*

Specifies the value of the option to be set. See [Default settings](#) for details.

## Examples

- Set your role to *developer*.

```
ccm set role developer
```

- Display your current role.

```
ccm set role  
developer
```

## Related topics

- [Showing an option](#)
- [Showing options](#)
- [Unsetting an option](#)

---

## **Showing an option**

This subcommand shows the value of an option.

```
ccm set option
```

```
option
```

Specifies the name of the option to be set. See [Default settings](#) for details.

### **Example**

- Show the value of `text_editor`.

Windows:

```
ccm set text_editor  
notepad %filename
```

UNIX:

```
ccm set text_editor  
vi %filename
```

### **Related topics**

- [Setting an option](#)
- [Showing options](#)
- [Unsetting an option](#)

---

## ***Showing options***

This subcommand shows available options.

```
ccm set
```

## **Related topics**

- [Setting an option](#)
- [Showing an option](#)
- [Unsetting an option](#)

---

## Description and uses

An option represents a control over the behavior of specific Synergy operations. Some options only apply to the current CLI session and are not saved from session to session. Some options are persistent user preferences stored in the database. Some options are predefined and read-only and cannot be modified.

Some of the options you can set include: `text_editor`, `text_viewer`, `role`, `verbosity`, and many more. See [Default settings](#) for a comprehensive list of options, and how and where to set them.

## show command

See [Description and uses](#) for details. The `show` command supports the following subcommand:

- [Showing projects](#)
- [Showing types](#)

---

## Showing projects

This subcommand shows all projects in the database that match specific criteria. If no query related options are specified, all projects are shown.

Each query related option is used to construct a query expression. For example, `-name name` results in a query expression of `name='name'`. If the same option is repeated, the query clauses are combined with an `or`. For example `-name example.txt -name another.txt` results in a query expression `"(name='example.txt' or name='another.txt')"`.

Query clauses for different options are combined with `and`. For example, `-n example.txt -s working` results in a query expression `"(name='example.txt') and (status='working')"`.

```
ccm show -p|-projects [(-o|-owner owner)...] [(-n|-name name)...]
          [(-v|-version version)...] [(-s|-state state)...]
          [(-task task_spec)...] [-f|-format format] [-nf|-noformat]
          [(-ch|-column_header) | (-nch|-nocolumn_header)]
          [-sep|-separator separator] ([-sby|-sortby sortspec] |
          [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat]
```

`-ch|-column_header`

Specifies to use a column header in the output format. See [-ch|-column\\_headers](#) for details.

`-f|-format format`

Specifies the command output format. See [-f|-format](#) for details.

A keyword can be built-in (`%fullname`, `%displayname`, `%objectname`) or the name of any existing attribute such as `%modify_time` or `%status`.

See [Built-In keywords](#) for a list of keywords.

`-gby|-groupby groupformat`

Specifies how to group the command output. See [-gby|-groupby](#) for details.

`-n|-name name`

Includes a query clause of the form `"name='name'"` to find projects with the specified name.

`-nch|-nocolumn_header`

Specifies not to use a column header in the output format. See [-ch|-column\\_headers](#) for details.

---

`-nf|-noformat`

Specifies not to use column alignment. See [-nf|-noformat](#) for details.

`-ns|-nosort|-no_sort`

Specifies that the command's output will not be sorted. See [-ns|-nosort](#) for details.

`-o|-owner owner`

Includes a query clause of the form "`owner='owner'`" to find projects with the specified owner.

`-sby|-sortby sortspec`

Specifies how to sort the command output. See [-ns|-nosort](#) for details.

`-sep|-separator separator`

Used only with the `-f|-format` option. Specifies a different separator character. See [-sep|-separator](#) for details.

`-s|-state state`

Includes a query clause of the form "`status='state'`" to find projects with the specified status.

`-task task_spec`

Includes a query clause of the form "`is_associated_cv_of(task('task_spec'))`" to find the associated projects of the specified task. You can set `task_spec` to one or more tasks. See [Task specification](#) for details.

`-v|-version version`

Includes a query clause of the form "`version='version'`" to find projects with the specified version.

## Examples

- Show projects in the database that have the status *integrate* and owner *john*.

```
ccm show -p -s integrate -o john
1) projY-1   integrate john project projY 1 2
2) projY-2   integrate john project projY 1 7
3) projY-2.1 integrate john project projY 1 8
```



- 
- Show projects in the database that have the status *integrate*, owner *john*, and are associated with task 8.

```
ccm show -p -s integrate -o john -task 8
1) projY-2.1 integrate john project projY 1 8
```

- Show the types defined in the database.

```
ccm show -t
```

```
ascii
binary
c++
csrc
dir
executable
incl
library
lsrc
makefile
project
relocatable_obj
shared_library
shsrc
symlink (UNIX)
ysrc
```

---

### ***Showing types***

This subcommand shows the object types defined in the database.

```
ccm show -t|-types
```

---

## **Description and uses**

The `show` command views the settings for certain attributes for projects, or views all types in the database.

## show\_servers command

See [Description and uses](#) for details. The `show_servers` command supports the [Showing servers](#) subcommand.

---

### ***Showing servers***

This subcommand shows the databases and associated Rational Synergy servers known to the router. You do not need to run a Rational Synergy CLI session to use this command. Use this command to see which databases are served by Rational Synergy servers and the server URL that should be specified to start a Synergy CLI session for each database.

```
ccm show_servers
```

---

## Description and uses

The `show_servers` command is useful for viewing the databases and associated Synergy servers known to the router. You might need this type of information to see the database you need to use and to find the server URL to specify when starting a session.

---

## start command

See [Description and uses](#) for details. The `start` command supports the [Starting a CLI session](#) subcommand.

---

## Starting a CLI session

```
ccm start [-q] -d database_path -s server_url [-m]
          [-pw password] [-n username] [-r initial_role] [-fpw]
```

-d *database\_pathname*

Specifies the absolute database path. For a database hosted on a UNIX server, use an absolute UNIX path. For a database hosted on a Windows server, use a UNC path.

-fpw

Specifies the path to the password file to be used instead of the default `.ccmrc` file.

-m

Specifies that multiple sessions will be used. You must set the `CCM_ADDR` environment variable to the session you will use. If omitted, `CCM_ADDR` is not defined for the different session, and the session information is written to a `.ccm.addr` file on the client. This is used as the default address.

-n *user\_name*

On Windows, specifies a Synergy user name to use. On UNIX, the Synergy user name is the same as the operating system user name and you don't need to specify `-n`. If you do specify `-n`, the user name must match the current user.

-pw *password*

Specifies the password for the Synergy user. If you do not specify a password, a default password is obtained from the `.ccmrc` file. If you specify the `-fpw` option, that file is used as a password file. If you do not specify the `-fpw` option, then `$HOME/.ccmrc` is used on UNIX, and `%HOMEPATH%\ccmrc` is used on Windows. (The default password is defined by using the `ccm set_password` command. See the [Rational Synergy Administration Guide for UNIX](#) for details on setting this command.)

-q

Starts a session in Quiet mode. When you use this option, Rational Synergy shows the `CCM_ADDR` for the CLI session. If you don't use this the option, the startup shows additional information, such as a copyright notice, before `CCM_ADDR`.

-r *initial\_role*

Specifies the initial role to be used. If you don't specify a role, your default initial role is used. It is based on the first role defined for you in the database user list.



---

`-s server_url`

Specifies the server to connect to. Ensure that the `server_url` is a valid URL for a compatible server, starting with either `http://` or `https://`

See "About the CCM server" in the appropriate [Rational Synergy Administration Guide](#) for a detailed discussion about the Synergy server you're connecting to.

## Examples

- Start a Rational Synergy session using a server URL.

```
ccm start -d /data/db1 -n bob -pw **** -s http://unixXYZ:8400
```

- Start a Rational Synergy session using a server URL in *quiet* mode while running another session.

```
ccm start -d \\ccmdb\db1 -n bob -pw **** -s http://winXYZ:8400 -q -m
```

## Caveats

If you start an additional session and you plan to use the command line, a warning message is displayed. Set the `CCM_ADDR` variable for the new session to the address displayed by Rational Synergy start, for example:

```
set CCM_ADDR=prefect.cwi.com:1368
```

This causes your Rational Synergy commands to be executed by the new session rather than by the session you were already running.

When running as user `ccm_root`, always use the `-m` option and always set `CCM_ADDR` in the environment. This distinguishes your `ccm_root` session from sessions where other users are running as `ccm_root`.

## Environment Variables

`CCM_ADDR`

## Files

`ccm.properties`

`ccm.user.properties`

`.ccm.rc`

## Related topics

- [stop command](#)

---

## Description and uses

The start command begins a Rational Synergy CLI session. After the session starts, the Rational Synergy address (`CCM_ADDR`), a unique identifier for this CLI session, displays in your command window (Windows) or in the shell where you started the session (UNIX).

If you run multiple Rational Synergy sessions, set the `CCM_ADDR` environment variable to specify which session will run your Rational Synergy commands. If you don't set `CCM_ADDR`, a default address is used. This default address is read from a `.ccm_addr` file. This file is generated if you start a CLI session without the `-m` (multiple sessions) option.

---

## stop command

See [Description and uses](#) for details. The `stop` command supports the [Stopping a CLI session](#) subcommand.

---

## ***Stopping a CLI session***

Use the stop command to stop a CLI session.

```
ccm stop|quit
```

### ***Example***

- Stop the current Rational Synergy CLI session.

```
ccm stop
```

### **Related topics**

- [start command](#)

---

## Description and uses

The `stop` command ends a Rational Synergy session.

## sync command

See [Description and uses](#) for details. The `sync` command supports the [Rewriting the work area](#) subcommand.

---

## Rewriting the work area

The `sync` command completely rewrites a work area for a project. The default directory in which all project work areas are created is `ccm_wa` followed by the database name in your home directory. Use the `sync` command to manually synchronize the work area.

**Note** Only a build manager or a user in the `ccm_admin` role can sync a non-writable project.

```
ccm sync [-r|-recurse] [-nr|-norecurse|-no_recurse] [-s|-static]
         [-p|-project] project_spec...
```

`-nr|-norecurse|-no_recurse`

Specifies not to recurse the project hierarchy during the project sync. Synchronize only the specified project.

This is the default.

`-p|-project project_spec...`

Specifies the project that you want to synchronize.

You can specify one or more arguments that are project specifications; each can specify one or more projects. See [Project specification](#) for details.

`-r|-recurse`

Causes all objects in the project hierarchy to be sync'd along with the specified project.

`-s|-static`

Updates an existing static work area with current data from the database. (A static work area is a local copy of the work area for a static subproject.) In addition, updates all static work areas in the hierarchy for which the `ccm sync` command was issued. This fully synchronizes all static work areas in the hierarchy by using one command. If no static work area exists in the hierarchy, this option is ignored.

## Examples

- Synchronize the work area for `toolkit-linda` and its subprojects.

```
ccm sync -recurse -project toolkit-linda
```

- Create a work area for the specified project.

```
ccm sync -p ico_aug1-1
```



---

## Defaults

You can set the following related options in your `ccm.ini` file (Windows) or `.ccm.ini` file (UNIX):

- [wa\\_path\\_template](#)

## Related topics

- [Modifying work area properties](#)
- [Showing work area properties](#)

---

## Description and uses

Your work area is created automatically when you create a project and when you check out a project using the check out commands. As you add new members to your project, your work area is updated automatically.

You will need to manually sync (force a sync) your work area in the following cases:

- You "clean out" (delete) any or all objects in your work area.  
When you force a sync, only the necessary (controlled) objects from your database are written out to your work area.
- The `work_area` command fails while changing your work area path.  
When you change your work area path either from the CLI or the GUI, Rational Synergy will try to update your work area path to the new location. If another application is using the old work area path, the move will fail and you will need to synchronize your work area.
- You change your work area type from one that uses local copies to one that uses symbolic links (or vice-versa).

If you want to change your work area type, do the following:

1. Reconcile the work area that you are currently using (either local copies or symbolic links).
2. Delete the work area objects from the file system.
3. Set your work area path and options.
4. Start a new session using the client option of choice (either local copies or symbolic links).
5. Re-create your work area by forcing a sync (execute the `sync` command).

**Note** To stop a sync from the CLI, enter `CTRL+C` at any time.

If you stop the sync, you will receive an error message stating that errors may occur in your work area. The errors will not occur until you try to use the work area; to avoid problems, perform a complete synchronization of the work area before you use it.

## task command

See [Description and uses](#) for details. The `task` command supports the following subcommands:

- [Assigning a task](#)
- [Associating a task with objects, tasks, or change requests](#)
- [Completing a task](#)
- [Copying a task](#)
- [Creating a task](#)
- [Disassociating a task from objects, tasks, or change requests](#)
- [Fixing a task](#)
- [Modifying a task](#)
- [Querying for tasks](#)
- [Setting or clearing the current task](#)
- [Showing a task property](#)
- [Showing the associated objects, change requests, and tasks for a task](#)
- [Showing task information](#)
- [Transitioning a task to a different state](#)

---

## Assigning a task

This subcommand assigns the specified task to the specified resolver. This operation can be performed by a user with task assignment privileges and by a user who can modify the task.

```
ccm task -as|-assign -t|-to resolver [-q|-quiet] task_spec...
```

```
-t|-to resolver
```

Specifies the resolver to which the tasks will be assigned. The *resolver* must be a valid task resolver.

```
-q|-quiet
```

Specifies that the confirmation messages include only the task identifier for each task assigned.

```
task_spec...
```

Specifies the tasks to be assigned. You can set the *task\_spec* to multiple tasks. For more details, see [Task specification](#).

## Example

- Assign tasks **54**, **60-63**, and **74** to user *joe*.

```
ccm task -as 54,60-63,74 -to joe
Assigned task 54
Assigned task 60
Assigned task 61
Assigned task 62
Assigned task 63
Assigned task 74
```

## Related topics

- [query command](#)

---

## **Associating a task with objects, tasks, or change requests**

This subcommand associates the specified task with specified objects, specified change requests, or with a specified task being fixed. For association with objects, any user that has modify access to the task can perform this operation. For association with a change request, any user that has modify access to the change request can perform this operation if the change request is in a state that allows task association.

The following outlines task requirements for associating a task with a task to be fixed:

- Tasks related to each other can be from different databases.
- Tasks to be fixed must be in either the *completed* or *excluded* state.
- A fix task must be modifiable by the user establishing the relationship.
- A task can only fix one task.

```
ccm task -a|-associate|-relate task_spec -obj|-object file_spec...
ccm task -a|-associate|-relate task_spec -fixes task_spec
ccm task -a|-associate|-relate task_spec
        -prob|-problem|-change_request change_request_spec...
```

*change\_request\_spec*

Specifies the change request(s) that the task will be associated with. You can set the *change\_request\_spec* to multiple change requests. For details, see [Change request specification](#).

*file\_spec*

Specifies the object to be associated with the task. The object can be a project, directory, or file. For details, see [File specification](#).

-fixes *task\_spec*

Specifies the task to be fixed. You can set the *task\_spec* to one task. For more details, see [Task specification](#).

-prob|-problem|-change\_request

Specifies the change request to be associated with the task.

## **Examples**

- Associate task **17** with the object `MAIN.C-3:csrc:1`.  

```
ccm task -a 17 -obj MAIN.C-3:csrc:1
```
- Associate task **54** with change request **D#1231**.  

```
ccm task -associate 54 -change_request D#1231
```

---

## Related topics

- [query command](#)

---

## Completing a task

This subcommand completes a task, checking in the task's associated objects to a non-modifiable state, and transitioning the task to the *completed* state. This subcommand can be performed by the resolver of a task or by an administrator. The argument keyword (*current* or *default*) means to complete the current task.

```
ccm task -ci|-checkin|-complete [-time|-time_actual task_duration]  
    [-c|-comment comment_string] [-ce|-commentedit]  
    [-cf|-commentfile file_path] [-commentreplace]  
    (task_spec|(current|default))...
```

`-c|-comment comment`

Specifies a comment that will be appended on all baseline projects and their members when they are checked in to the *released* state. The *comment* can contain more than one line and accepts backslash encoded values.

You can use this option with `-commentedit` and `-commentfile`. If you use the `-commentedit` option, the comment displays in the default text editor.

`-ce|-commentedit`

Specifies that the default text editor will be invoked to allow the comment to be composed and edited. The result saved from the text editor is used as the final comment. You can use this option with the `-comment` and `-commentfile` options.

`-cf|-commentfile file_path`

Specifies that the contents of the specified file will be used for the comment. If you specified `-comment`, it is appended to that comment. You can use this option with the `-commentedit` option.

`-commentreplace`

Specifies that the comment will be replaced.

`-time|-time_actual task_duration`

Specifies the time required to complete the task. The *task\_duration* can be any string. However, to help with reporting and metrics, be sure to adopt a consistent convention for format and units.

---

`task_spec` | (current | default)

Specifies the task(s) to be completed. The `current` or `default` keyword means complete the current task. You can set the `task_spec` to multiple tasks. See [Task specification](#) for more information.

### **Examples**

- Check in all of the objects associated with task **40**.

```
ccm task -complete 40 -comment "The problem is fixed."
```

- Complete the current task.

```
ccm task -complete default
```

### **Related topics**

- [query command](#)



---

## Copying a task

This subcommand creates new tasks by copying specified tasks. Copy a task when you need to apply a task that you fixed for the release to a different release. The copied task and the original task might have the same associated objects, different associated objects, or a combination. By default, the objects associated with the existing task will also be associated with the corresponding copied task.

```
ccm task -cp|-copy [-no_objects] -s|-synopsis synopsis
    [-prob|-problem|-change_request change_request_spec]
    ([-def|-default|-current] | [-register])
    [-desc|-description description]
    [-desc_edit|-descriptionedit|-description_edit]
    [-desc_file|-descriptionfile|-description_file file_path]
    [-p|-priority priority] [-plat|-platform platform]
    [-r|-resolver resolver] [-rel|-release release_spec]
    [-sub|-subsystem subsystem] [-time|-time_estimate time_estimate]
    [-date|-date_estimate date_estimate] [-q|-quiet] task_spec...
```

`-date|-date_estimate date_estimate`

Specifies the estimated completion date of the tasks you're creating. If you don't specify the date estimate, it is set to the date estimate of the task you're copying. The *date\_estimate* must be a valid date.

`-def|-default|-current`

Specifies that the first task created from the task copy will become the current task for this CLI session.

`-desc|-description description`

Specifies a single-line description. The description cannot contain newline characters.

`-desc_edit|-description_edit`

Specifies to start the default text editor so you can edit or compose a multi-line description.

`-desc_file|-description_file file_path`

Specifies a path to a file containing a multi-line description.

`-no_objects`

Specifies that the objects associated with the task you're copying won't be associated with the task created by the copy.

---

`-p|-priority priority`

Specifies the priority of the new tasks. If you don't specify the priority, it is set to the priority of the task you're copying. The priority must be a valid task priority. The default valid priorities are `High`, `Medium`, and `Low`.

`-plat|-platform platform`

Specifies the platform of the tasks being created. If you don't specify the platform, it is set to the platform of the task you're copying. The platform must be a valid platform.

`-prob|-problem|-change_request change_request_spec`

Specifies to associate the new task with the specified change request. The change request must be modifiable by you and in a state that permits task association. You can set `change_request_spec` to one change request. For more details, see [Change request specification](#). If you don't specify a change request and the task you're copying is associated with a change request that is in the `assigned` state, the new task is also associated with that change request.

`-quiet`

Specifies that the confirmation messages include only the task identifier for each task created.

`-register`

Specifies to create the task in the `registered` state. (A task in this state is entered in Synergy, but is not assigned to anyone.) If you don't specify `-register`, the task is assigned to the same user as the source task or to the user specified by the `-resolver` option.

`-rel|-release release_spec`

Specifies the release for the created tasks. If you don't specify a release, it is set to the release of the task you're copying. You can set the `release_spec` to one release. For details, see [Release specification](#)

`-r|-resolver resolver`

Specifies which user is responsible for resolving the tasks. If not specified, it is set to the resolver of the task you're copying. The `resolver` must be a valid task resolver.

---

`-sub|-subsystem subsystem`

Specifies the task subsystem for the created tasks (for example, Any, GUI code, CLI code, or documentation). If you don't specify a subsystem, it is set to the subsystem of the task you're copying. The subsystem must be a valid task subsystem.

`-s|-synopsis synopsis`

Specifies the synopsis of the task you're copying. The synopsis can be any string without newline characters.

`task_spec...`

Specifies the tasks to be copied. You can set the `task_spec` to multiple tasks. For more details, see [Task specification](#).

`-time|-time_estimate time_estimate`

Specifies the estimated duration or effort to complete the created tasks. If you don't specify a time estimate, it is set to the time estimate of the task you're copying. The `time_estimate` can be any string. However, to help with reporting and metrics, be sure to adopt a consistent convention for format and units.

### **Example**

- Copy task **40**, and specify a different synopsis, release, resolver, and description. Do not to copy the objects associated with it.

```
ccm task -copy 40 -synopsis "Fix GUI color problem" -release 2.0 -  
resolver donho -no_objects -description "check RGB module"  
Task hawaii#50 created.
```

### **Related topics**

- [query command](#)

---

## Creating a task

This subcommand creates a task. If you specify a resolver, the task is assigned to the specified person. If you do not specify a resolver, or if you specify `-register` when creating the task, the task is registered in Synergy, but not assigned to a person.

```
ccm task -cr|-create -s|-synopsis synopsis
        [-prob|-problem|-change_request change_request_spec]
        ([-def|-default|-current] | [-register])
        [-desc|-description description]
        [-desc_edit|-descriptionedit|-description_edit]
        [-desc_file|-descriptionfile|-description_file file_path]
        [-p|-priority priority] [-plat|-platform platform]
        [-r|-resolver resolver] [-rel|-release release_spec]
        [-sub|-subsystem subsystem] [-time|-time_estimate time_estimate]
        [-date|-date_estimate date_estimate] [-q|-quiet]
```

`-date|-date_estimate date_estimate`

Specifies the estimated completion date. The *date\_estimate* must be a valid date.

`-def|-default|-current`

Specifies that the task you're creating will be set as the current task for this CLI session.

[-desc|-description \*description\*](#)

[-desc\\_edit|-description\\_edit](#)

[-desc\\_file|-description\\_file \*file\\_path\*](#)

`-plat|-platform platform`

Specifies the platform. The platform must be a valid platform.

`-p|-priority priority`

Specifies the priority. The priority must be a valid task priority. The default valid priorities are High, Medium, and Low.

`-prob|-problem|-change_request change_request_spec`

Specifies to associate the new task with the specified change request. The change request must be modifiable by you and in a state that permits task association. You

---

can set *change\_request\_spec* to one change request. For more details, see [Change request specification](#).

`-quiet`

Specifies that the confirmation messages include only the task identifier for each task created.

`-register`

Specifies that the task should be created in the registered state.

`-rel|-release release_spec`

Specifies the release. You can set *release\_spec* to one release. For details, see [Release specification](#)

`-r|-resolver resolver`

Specifies which user is responsible for resolving the tasks. If not specified, it is set to the resolver of the task you're copying. The *resolver* must be a valid task resolver.

`-sub|-subsystem subsystem`

Specifies the task subsystem. The subsystem must be a valid task subsystem.

`-s|-synopsis synopsis`

Specifies the synopsis of the task you're creating. The synopsis can be any string without newline characters.

`-time|-time_estimate time_estimate`

Specifies the estimated time it will take to complete the task(s). The *time\_estimate* can be any string. However, to help with reporting and metrics, be sure to adopt a consistent convention for format and units.

### **Example**

- Create a task with the synopsis name, Entanglement methods.

```
ccm task -create -synopsis "Entanglement methods"
Task 44 created.
```

---

## **Disassociating a task from objects, tasks, or change requests**

This subcommand breaks the association between two objects. It can disassociate the specified task from specified objects and change requests, and from a specified task being fixed. If you can modify the task, then you can disassociate it from objects or a task being fixed. If you need to disassociate the task from a change request, you must be able to modify both the change request and the task.

```
ccm task -d|-disassociate|-unrelate task_spec -obj|-object file_spec...
ccm task -d|-disassociate|-unrelate task_spec -fixes task_spec
ccm task -d|-disassociate|-unrelate task_spec -prob|-problem|
    -change_request change_request_spec...
```

*change\_request\_spec*

Specifies the change request(s) that the task will be associated with. You can set the *change\_request\_spec* to multiple change requests. For details, see [Change request specification](#).

-fixes *task\_spec*

Specifies the task that was fixed. You can set *task\_spec* to one task. For more details, see [Task specification](#).

-obj|-object *file\_spec*...

Specifies the name of the file or directory that you want to disassociate from the specified task.

-prob|-problem|-change\_request

Specifies the change request to be associated with the task.

## **Examples**

- Disassociate task **35** from object version MAIN.C-3:csrc:1.  

```
ccm task -d 34 -obj MAIN.C-3:csrc:1
```

Disassociated object version from task 34: MAIN.C-3:csrc:1
- Disassociate task **10668** from change request **6569**.  

```
ccm task -d 10668 -change_request 6569
```

## **Related topics**

- [query command](#)

---

## Fixing a task

This subcommand creates a task and establishes a relationship between it and the task to be fixed. This relationship detects when a project is using one task without the other. (This is called a *conflict*.) For information about conflicts, see [conflicts command](#).

To fix a task with an existing task, see [Associating a task with objects, tasks, or change requests](#). If the fix task needs to be fixed or enhanced, you can create a new fix task to fix the first fix task.

The following outlines task requirements for creating a fix relationship:

- Tasks related to each other can be from different databases.
- Tasks to be fixed must be in either the *completed* or *excluded* state.
- A fix task must be modifiable by the user establishing the relationship.
- A task can only fix one task.

```
ccm task -fix [-exclude] -s|-synopsis synopsis
            [-prob|-problem|-change_request change_request_spec]
            ([-def|-default|-current] | [-register])
            [-desc|-description description]
            [-desc_edit|-descriptionedit|-description_edit]
            [-desc_file|-descriptionfile|-description_file file_path]
            [-p|-priority priority] [-plat|-platform platform]
            [-r|-resolver resolver] [-rel|-release release_spec]
            [-sub|-subsystem subsystem] [-time|-time_estimate time_estimate]
            [-date|-date_estimate date_estimate] [-q|-quiet] task_spec...
```

[-date|-date\\_estimate \*date\\_estimate\*](#)

[-def|-default|-current](#)

Specifies that the fix task you're creating will be set as the current task for this CLI session.

[-desc|-description \*description\*](#)

[-desc\\_edit|-description\\_edit](#)

[-desc\\_file|-description\\_file \*file\\_path\*](#)

[-exclude](#)

Specifies to transition the tasks being fixed to the *excluded* state. Use this option to exclude them from being automatically included in future builds.

---

[-plat|-platform platform](#)

[-pl|-priority priority](#)

`-prob|-problem|-change_request change_request_spec`

Specifies to associate the fix task with the specified change request. The change request must be modifiable by you and in a state that permits task association. If you don't specify a change request and the task being fixed is associated with a change request that is in the *assigned* state, the new task is also associated with that change request.

You can set *change\_request\_spec* to one change request. For more details, see [Change request specification](#).

`-quiet`

Specifies that the confirmation messages include only the task identifier for the fix task.

[-register](#)

[-rel|-release release\\_spec](#)

`-r|-resolver resolver`

Specifies which user is responsible for resolving the tasks. If not specified, it is set to the resolver of the task you're fixing. The *resolver* must be a valid task resolver.

[-sub|-subsystem subsystem](#)

[-s|-synopsis synopsis](#)

`task_spec...`

Specifies the tasks to be fixed. You can set *task\_spec* to multiple tasks. For more details, see [Task specification](#).



---

[-time|time\\_estimate\\_time\\_estimate](#)

### **Examples**

- Create a relationship between the fix task (19) and the task to be fixed (4).  

```
ccm task -relate 19 -fixes 4
```
- Break a relationship between the fix task (25) and the task it fixed (12).  

```
ccm task -unrelate 25 -fixes 12
```
- Create a fix task for task 4.  

```
ccm task -fix -s "Create a fix task for task 4" 4
```

Task 17 created to fix Task 4.
- Create a fix task and transition the task being fixed to the *excluded* state  

```
ccm task -fix -exclude -s "exclude task 1 and create new for release 1.0" 1
```

Task 16 created to fix Task 1.

### **Related topics**

- [query command](#)

---

## Modifying a task

```
ccm task -mod|-modify [-s|-synopsis synopsis]  
                [-desc|-description description]  
                [-desc_edit|-descriptionedit|-description_edit]  
                [-desc_file|-descriptionfile|-description_file file_path]  
                [-desc_replace|-descriptionreplace|-description_replace]  
                [-p|-priority priority] [-plat|-platform platform]  
                [-r|-resolver resolver] [-rel|-release release_spec]  
                [-sub|-subsystem subsystem] [-time|-time_estimate time_estimate]  
                [-date|-date_estimate date_estimate] task_spec...
```

[-date|-date\\_estimate \*date\\_estimate\*](#)

[-desc|-description \*description\*](#)

[-desc\\_edit|-description\\_edit](#)

[-desc\\_file|-description\\_file \*file\\_path\*](#)

[-desc\\_replace|-descriptionreplace|-description\\_replace](#)

Specifies to replace the existing task description with the specified description. By default, the description is appended to the existing task description. The description cannot contain newline characters.

[-plat|-platform \*platform\*](#)

[-p|-priority \*priority\*](#)

[-rel|-release \*release\\_spec\*](#)

[-r|-resolver \*resolver\*](#)

Specifies which user is responsible for resolving the tasks. If not specified, it is set to the resolver of the task you're modifying. The *resolver* must be a valid task resolver.

[-sub|-subsystem \*subsystem\*](#)

[-s|-synopsis \*synopsis\*](#)

---

*task\_spec...*

Specifies the tasks to be modified. You can set the *task\_spec* to multiple tasks. For more details, see [Task specification](#).

[-time|time\\_estimate](#) *time\_estimate*

### **Example**

- Change the release for task 68 to 4.1.  

```
ccm task -modify -release 4.1 68
```

### **Related topics**

- [query command](#)

---

## Querying for tasks

This subcommand queries for tasks that are in a release, not in a release, or that match the specified query criteria or query expression. Use the tasks found by the query to set the query selection set.

```
ccm task -qu|-query -in_rel|-in_release [-f|-format format]
        [-nf|-noformat] ([-ch|-column_header] | [-nch|-nocolumn_header])
        [-sep|-separator separator] ([-sby|-sortby sortspec] |
        [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat]
        [-u|-unnumbered] old_project_spec project_spec
ccm task -qu|-query -in_rel|-in_release [-f|-format format]
        [-nf|-noformat] ([-ch|-column_header] | [-nch|-nocolumn_header])
        [-sep|-separator separator] ([-sby|-sortby sortspec] |
        [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat]
        [-u|-unnumbered] project_spec
ccm task -qu|-query -not_in_rel|-not_in_release [-f|-format format]
        [-nf|-noformat] ([-ch|-column_header] | [-nch|-nocolumn_header])
        [-sep|-separator separator] ([-sby|-sortby sortspec] |
        [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat]
        [-u|-unnumbered] project_spec
ccm task -qu|-query [-cus|-custom custom_query]
        [(-db|-dbid|-database_id database_spec)...]
        [(-plat|-platform platform)...] [(-purpose purpose)...]
        [(-rel|-release release_spec)...]
        [(-sub|-subsystem subsystem)...]
        [-ts|-scope|-task_scope (user_defined |
        (all_my_assigned|all_owners_assigned) |
        (all_my_assigned_or_completed|all_owners_assigned_or_completed) |
        (all_my_completed|all_owners_completed) |
        (all_my_tasks|all_owners_tasks) | all_completed | all_tasks)]
        (ct_projs|ct_projects|component_task_projects) |
        (ct_prods|ct_products|component_task_products) |
        (ct_projs_prods|ct_projects_products |
        component_task_projects_products))
        [-f|-format format] [-nf|-noformat] ([-ch|-column_header] |
        [-nch|-nocolumn_header]) [-sep|-separator separator]
        ([-sby|-sortby sortspec] | [-ns|-nosort|-no_sort])
        [-gby|-groupby groupformat] [-u|-unnumbered]
```

**-ch|-column\_header**

Specifies to use a column header in the output format. See [-ch|-column headers](#) for details.

**-cus|-custom *custom\_query***

Specifies to include the specified custom query expression in the query.

---

`-db|-dbid|-database_id database_spec`

When used with the `-task_scope` option, specifies a database identifier that modifies the query generated from the task scope. See [Database specification](#) for further details.

`-f|-format format`

Specifies the command output format. See [-f|-format](#) for details.

`-gby|-groupby groupformat`

Specifies how to group the command output. See [-gby|-groupby](#) for details.

`-in_rel|-in_release old_project_spec project_spec`

Shows all tasks that are in the project hierarchy with `project_spec` as its root. This is determined by getting all tasks for all objects in the hierarchy, and for all their ancestors, subtracting the similar list of tasks for the hierarchy with `old_project_spec` as its root.

If you do not specify `old_project_spec`, no tasks are subtracted. You should specify `old_project_spec` except for the first release of the product, when there is no baseline release.

`-nch|-nocolumn_header`

Specifies not to use a column header in the output format. See [-ch|-column\\_headers](#) for details.

`-nf|-noformat`

Specifies not to use column alignment. See [-nf|-noformat](#) for details.

`-ns|-no_sort`

Specifies that the command's output will not be sorted. See [-ns|-nosort](#) for details.

`-not_in_rel|-not_in_release project_spec`

Shows all tasks that are not in the project hierarchy with `project_spec` as its root.

`-plat|-platform platform`

Specifies the platform. The platform must be a valid platform.

---

`-purpose purpose`

Specifies to create the folder with a task query that includes a query for the specified purpose. See the `project_purpose` command [Description and uses](#) section for a detailed description of purposes.

This option typically applies to queries for component tasks that are specified with one of the following scopes: `component_task_projects`, `component_task_products`, or `component_task_projects_products`.

`-rel|-release release_spec`

Specifies the release. You can set `release_spec` to one release. For details, see [Release specification](#)

`-sep|-separator separator`

Specifies a different separator character. See [-sep|-separator](#) for details.

`-sby|-sortby sortspec`

Specifies how to sort the command output. See [-sby|-sortby](#) for details.

`-sub|-subsystem subsystem`

Specifies the task subsystem. The subsystem must be a valid task subsystem.

`-ts|-scope|-task_scope`

Specifies to use a task query. The task query will include a query expression that depends on the specified scope. The query expression associated with the specified scope also depends on the `-database_id` option. You can use the following scopes:

- `user_defined`  
This scope is defined by the [default task query](#) option. If you specify `-database_id`, the query also includes a query expression for tasks modifiable in or completed in the specified database.
- `all_my_assigned|all_owners_assigned`  
This scope queries for all tasks assigned to you. If you specify `-database_id`, the query is for all tasks assigned to you that are modifiable in the specified database.
- `all_my_assigned_or_completed|all_owners_assigned_or_completed`

---

This scope queries for all tasks assigned to you or completed by you. If you specify `-database_id`, the query is for all tasks assigned to you and modifiable in the specified database, or completed by you in the specified database.

- `all_my_completed|all_owners_completed`

This scope queries for all tasks completed by you. If you specify `-database_id`, the query is for all tasks completed by you in the specified database.

- `all_my_tasks|all_owners_tasks`

This scope queries for all tasks for which you are the task resolver. If you specify `-database_id`, the query is for all tasks for which you are the task resolver and that are modifiable in the specified database or were completed in the specified database.

- `all_completed`

This scope queries for all completed tasks. If you specify `-database_id`, the query is for all tasks completed in the specified database.

- `all_tasks`

This scope queries for all tasks. If you specify `-database_id`, the query is for all tasks that are modifiable in the specified database or that were completed in the specified database.

- `component_task_projects|component_task_products|  
component_task_projects_products`

This scope queries for component tasks for projects, products, or projects and products. If you specify `-database_id`, the query is for all component tasks that were created in the specified database. If you specify `-purpose`, the query is for component tasks with the specified purpose.

`-u|-unnumbered`

Suppresses automatic numbering of the command's output (that is, the output is unnumbered). See [-u|-unnumbered](#) for details.

## Example

- Query for the tasks that have a release value set to 3.0. Format the output so that it shows only the task synopsis.

```
ccm task -qu -rel 3.0 -f "%priority %task_synopsis"
```

```
1) high Correct formatting of calculating number
2) high Redesign gui for file open dialog
3) high Performance improvement for file close
4) low Enhance message text
```

---

## Related topics

- [query command](#)



---

## Setting or clearing the current task

This subcommand sets or clears the current task.

By default, if you perform an operation that associates changes with a task, but you don't specify a task, Rational Synergy uses the current task.

```
ccm task -def|-default|-current [(task_spec|none)]
```

Sets the specified task as the current task for this CLI session. You can set *task\_spec* to one task that is assigned to you. For details, see [Task specification](#).

Specifying the `none` keyword clears the current task.

If you don't specify arguments, the command shows the current task, but doesn't set it.

## Examples

- Show the current task.

```
ccm task -current
The current task is not set.
```

- Set the current task.

```
ccm task -current 26
The current task is set to:
26: Close box no longer active
```

- Clear the current task.

```
ccm task -current none
The current task has been cleared.
```

## Related topics

- [query command](#)

---

## ***Showing a task property***

This subcommand shows the specified task property.

```
ccm task -s|-sh|-show ((p|priority) | (plat|platform) | (r|resolver) |  
    (rel|release) | (s|synopsis) | (sub|subsystem) |  
    ( time|time_estimate) | (date|date_estimate) |  
    (desc|description) | status_log) task_spec...
```

*task\_spec...*

Specifies the task(s) whose properties you want to view. See [Task specification](#) for details.

## **Related topics**

- [query command](#)

---

## **Showing the associated objects, change requests, and tasks for a task**

This subcommand shows the following:

- Associated objects
- Change requests for the specified tasks
- Tasks fixed by the specified task(s)
- Tasks that fix the specified task(s)

The query selection displays the associated objects.

```
ccm task -s|-sh|-show ((obj|objs|objects) |
    (cr|change_request|change_requests|prob|problem|problems) |
    (fix|fixes) | fixed_by)
    [-f|-format format] [-nf|-noformat [-ch|-column_header] |
    [-nch|-nocolumn_header]] [-sep|-separator separator]
    ([-sby|-sortby sortspec] | [-ns|-nosort|-no_sort])
    [-gby|-groupby groupformat] [-u|-unnumbered] task_spec...
```

[-ch|-column\\_header](#)

[-f|-format \*format\*](#)

[-gby|-groupby \*groupformat\*](#)

[-nch|-nocolumn\\_header](#)

[-nf|-noformat](#)

[-ns|-no\\_sort](#)

[-sep|-separator \*separator\*](#)

[-sby|-sortby \*sortspec\*](#)

*task\_spec...*

Specifies the task(s) whose properties you want to view. See [Task specification](#) for details.

---

`-u` | `-unnumbered`

Suppresses automatic numbering of the command's output (that is, the output is unnumbered). See [-u|-unnumbered](#) for details.

### **Examples**

- Show the change requests associated with task **68**.

```
ccm task -show change_request 68
```

```
1) Change request 5  
2) Change request 6
```

- Show the objects associated with tasks **4** and **5**.

```
ccm task -show objects 4,5
```

```
1) MAIN.C-2:csrc:1 integrate ann  
2) MAIN.H-4:incl:1 integrate ann  
3) UTIL.C-7:csrc:1 integrate ann  
4) MSGS.H-9:incl:1 integrate ann
```

### **Related topics**

- [query command](#)

---

## Showing task information

This subcommand shows information, such as task synopsis, description, state, resolver, and more, about the specified task(s).

```
ccm task -s|-sh|-show (i|info|information) -f|-format format
                    [-nf|-noformat] ([-ch|-column_header] | [-nch|-nocolumn_header])
                    [-sep|-separator separator] task_spec...
ccm task -s|-sh|-show (i|info|information) [-v|-verbose] task_spec...
```

[-ch|-column\\_header](#)

[-f|-format \*format\*](#)

[-nch|-nocolumn\\_header](#)

[-nf|-noformat](#)

[-sep|-separator \*separator\*](#)

*task\_spec...*

Specifies the task(s) whose properties you want to view. See [Task specification](#) for details.

[-v|-verbose](#)

Specifies to display output using a verbose default format for the task information.

## Examples

- Show task information for release 1.

```
ccm task -s release 1
Task 1: a/1.0
```

- Show formatted information about tasks 30 - 33.

```
ccm task -show info 30-34 -format "%priority %30-33 %task_synopsis"
-ns
1) high 33 Date field not validated on Inventory Form
2) high 41 Wrong window receives message
3) high 22 Saving a file takes forever
4) low 39 Button icons are rather obscure
5) low 4 OK button not default
```

- 
- Show information for the objects associated with task **14**.

```
ccm task -sh obj 14
```

```
Task 14:
```

```
1) a.txt-1.1:ascii:1 integrate jane
```

```
2) b.txt-1.1:ascii:1 integrate jane
```

## Related topics

- [query command](#)

---

## Transitioning a task to a different state

```
ccm task -st|-state task_state [-r|-resolver resolver]  
        [-desc|-description description]  
        [-desc_edit|-descriptionedit|-description_edit]  
        [-desc_file|-descriptionfile|-description_file file_path]  
        task_spec...
```

[-desc|-description \*description\*](#)

[-desc\\_edit|-description\\_edit](#)

[-desc\\_file|-description\\_file \*file\\_path\*](#)

*task\_state*

Specifies the state you want to transition the task to.

*-r|-resolver resolver*

Specifies that the resolver of the specified tasks should be set. This option can only be used when transitioning a task to the *task\_assigned* state. The *resolver* must be a valid task resolver.

*task\_spec...*

Specifies the task(s) you want to transition. See [Task specification](#) for details.

## Examples

- Transition a task from the *completed* state to *excluded*.

```
ccm task -state excluded 94
```

- Transition a task from the *excluded* state to *completed*.

```
ccm task -state completed 94
```

## Related topics

- [query command](#)

---

## Description and uses

Use the `task` command to perform the following task-based operations:

- Assign a task
- Associate a task with objects, another task, or a change request
- Complete (check in) a task
- Copy a task
- Create a task
- Disassociate a task from objects, another task, or a change request
- Fix a task
- Modify a task
- Query for tasks
- Create (relate) or break (unrelate) relationships between a task and tasks, or objects
- Set or clear the current (default) task
- Show task information
- Transition a task to a different state



## unalias command

See [Description and uses](#) for details. The `unalias` command supports the [Removing an alias](#) subcommand.

---

## **Removing an alias**

```
ccm unalias alias_name
```

*alias\_name*

Specifies the name of the alias you want to remove.

### **Example**

- Remove the alias for the `getf` command.

```
ccm unalias getf
```

### **Related topics**

- [alias command](#)

---

## **Description and uses**

The `unalias` command removes a defined alias.

Using the `unalias` command removes an alias for the current session only.

## undo\_update command

See [Description and uses](#) for details. The `undo_update` command supports the following subcommands:

- [Reversing an update for a directory](#)
- [Reversing an update for a project](#)
- [Reversing an update for a project grouping](#)

---

## Reversing an update for a directory

```
ccm unupd|undo_update|unreconf|undo_reconfigure [-r|-recurse]
          [-v|-verbose] dir_spec...
```

*dir\_spec...*

Specifies the directory where the update will be reversed. You can set *dir\_spec* to multiple directory objects. The *dir\_spec* takes the forms described in [File specification](#).

`-r|-recurse`

Specifies to include subprojects.

`-v|-verbose`

Displays detailed undo update messages.

## Related topics

- [Reversing an update for a project](#)
- [Reversing an update for a project grouping](#)

---

## Reversing an update for a project

```
ccm unupd|undo_update|unreconf|undo_reconfigure -p|-project  
[-r|-recurse] [-v|-verbose] project_spec...
```

*project\_spec*

Specifies the project where the update will be reversed. You can set *project\_spec* to multiple projects. The *project\_spec* takes the forms described in [Project specification](#).

-r|-recurse

Specifies to include subprojects.

-v|-verbose

Displays detailed undo update messages.

### Examples

- Reverse the update on the `proj1-1` project.  

```
ccm unupd -p proj1-1
```
- Reverse the update on a project named `toolkit-jane`, which also has subprojects.  

```
ccm undo_update -recurse -project toolkit-jane
```

### Related topics

- [Reversing an update for a directory](#)
- [Reversing an update for a project grouping](#)

---

## Reversing an update for a project grouping

```
ccm unupd|undo_update|unreconf|undo_reconfigure  
    -pg|-project_grouping [-r|-recurse] [-v|-verbose]  
    project_grouping_spec...
```

*project\_grouping\_spec*

Specifies the project grouping where the update will be reversed. You can set *project\_grouping\_spec* to multiple projects.

No changes are made to the baseline and tasks for the project grouping. See [Project grouping specification](#) for details.

-r|-recurse

Specifies to include subprojects.

-v|-verbose

Displays detailed undo update messages.

## Related topics

- [Reversing an update for a directory](#)
- [Reversing an update for a project](#)

---

## Description and uses

The `undo_update` command reverses the update operation for a specified directory or project object.

By default and for performance purposes, the `undo_update` command does not provide parallel version notification when it encounters parallel object versions. You can enable parallel version notification by setting the [reconfigure\\_parallel\\_check](#) user option to `TRUE` in your initialization file.

The undo update process stops if an individual operation within the undo fails. For example, if the current version of an object has a work area conflict, the process stops and the new version is not automatically used. This is done to protect the data in the user's work area.

The default setting to stop the `undo_update` command can be changed by modifying your initialization file. Some users may want to continue with the `undo_update` process, even though an individual failure has occurred. You can set update to continue by setting the [reconf\\_stop\\_on\\_fail](#) option to `False`.

The `undo_update` command can be used to reverse the last undo update operation. In other words, if two or more undo updates are performed, only the last one will be reversed.



## unrelate command

See [Description and uses](#) for details. The `unrelate` command supports the [Deleting a relationship between two objects](#) subcommand.

---

## Deleting a relationship between two objects

```
ccm unrelate -n|-name relationship_name -f|-from from_object_spec  
              -t|-to to_object_spec
```

`-f|-from object_spec`

Specifies the source object of the relationship. You can set the *object\_spec* to a single object of any type.

`-n|-name relationship_name`

Specifies the name of the relationship to delete.

`-t|-to object_spec`

Specifies the destination object of the relationship. You can set the *object\_spec* to a single object of any type.

### Example

- Delete the **successor** relationship from `clear.c-2` to `clear.c-1`:

```
ccm unrelate -n successor -f clear.c-1:csrc:1 -t clear.c-2:csrc:1
```

### Related topics

- [Creating a relationship from one object to another](#)
- [Showing relationships to and from an object](#)

---

## Description and uses

The `unrelate` command deletes a relationship, *rel\_name*, between *file\_spec1* and *file\_spec2*.

More relationships are predefined in Rational Synergy. See "Relationships" in [Synergy CLI Help, Traditional mode](#) for a table of these relationships. However, you can define new relationships using the `relate` command.

To delete the relationship between two objects, you must include all three object specifications (defined above).

The command does not update the query selection set.

## unset command

See [Description and uses](#) for details. The `unset` command supports the [Unsetting an option](#) subcommand.

---

## ***Unsetting an option***

```
ccm unset option
```

*option*

Specifies the name of the option being unset.

### ***Example***

- Unset the `proj_log` option.

```
ccm unset proj_log
```

### **Related topics**

- [Setting an option](#)
- [Showing an option](#)
- [Showing options](#)

---

## Description and uses

The `unset` command removes any settings to the value of an option. This reverts the option to the factory default value. Some options are predefined and read-only, and cannot be unset. See [Default options](#) for details about options.

## unuse command

See [Description and uses](#) for details. The `unuse` command supports the following subcommands:

- [Removing a project from a specified directory](#)
- [Removing a project from the current directory](#)
- [Removing an object from a project](#)

---

## Removing a project from a specified directory

This subcommand removes a project from a specified directory and its associated context project. The directory must be specified using a project reference spec form or a work area reference form in order to provide the context project. If the context project is in working state, you must be the owner of that project. If the context project is in prep state, you must be a build manager.

```
ccm unuse -p|-project -dir dir_spec [-t|-task task_spec]  
          [-d|-delete [-f|-force]] [-r|-replace] project_spec...
```

`-d|-delete`

Remove the object from the directory and its context project, then delete the object from the database.

`-dir dir_spec`

Specifies the directory from which the object will be unused. The *dir\_spec* is a *file\_spec* (see [File specification](#)) that resolves to a single directory object and provides a context project. A project reference spec form or a work area reference form provides such a context project.

`-force`

This option can only be used with the `-d|-delete` option. It specifies that when an object is deleted, that object is also removed from all projects that are modifiable by you before it is deleted from the database. When this option is not specified, the object is only unused from the context project for the object. The deletion of the object from the database occurs if the object is not a member of any project.

*project\_spec*

Specifies the projects to be unused. See [Project specification](#) for details.

`-r|-replace`

Replace the object in the directory with its predecessor. When this option is specified, the list of files in the directory remains unchanged; only the version of the specified object changes.

`-t|-task task_spec`

Specifies the task that will be associated with any directory that is automatically checked out. If `-r|-replace` is not specified, the directory that contains the object is updated to remove the entry for that object. If the directory is not in a static state, the



---

directory is automatically checked out. If the `-t|-task` option is not specified, then the current task is used by default. See [Task specification](#) for details.

### **Examples**

- Remove the `ico_jan5` and `ico_jan6` subprojects from the `ico_jan4-1` top-level project.  

```
ccm unuse ico_jan5 ico_jan6
```

```
Member ico_jan5-1 removed from project ico_jan4-1  
Member ico_jan6-1 removed from project ico_jan4-1
```
- Unuse the subproject `SubProject_One-1` under the `Dir` of the project `Project_One-1`.  

```
ccm unuse -p -dir Project_One\Dir@Project_One-1 SubProject_One-1:project:1
```

### **Related topics**

- [use command](#)
- [delete command](#)

---

## Removing a project from the current directory

This subcommand removes a project from the current working directory. The current working directory must be within a maintained work area whose project is modifiable by you. If the context project is in working state, you must be the owner of that project. If the context project is in prep state, you must be a build manager.

```
ccm unuse -p|-project [-t|-task task_spec] [-d|-delete [-f|-force]]
           [-r|-replace] project_spec...
```

-d|-delete

See [-d|-delete](#).

-force

See [-force](#).

*project\_spec*

See [project\\_spec](#).

-r|-replace

See [-r|-replace](#).

-t|-task *task\_spec*

See [-t|-task task\\_spec](#).

### Example

- Cut `sort.c` from the current directory.

```
ccm unuse sort.c-1:csrc:1
```

### Related topics

- [use command](#)
- [delete command](#)

---

## Removing an object from a project

This subcommand removes an object from a project. If the `-dir` option is specified, the object will be removed from the directory and its associated context project. If the `-dir` option is not specified, it removes the object from the current working directory. The current working directory must be within a maintained work area whose project is modifiable by you. If the context project is in *working* state, you must be the owner of that project. If the context project is in *prep* state, you must be a build manager.

```
ccm unuse [-dir dir_spec] [-t|-task task_spec] [-d|-delete [-f|-force]]
          [-r|-replace] file_spec...
```

`-d|-delete`

See [-d|-delete](#).

`-dir dir_spec`

See [-dir dir\\_spec](#).

`file_spec`

Specifies the object or objects to unuse.

`-force`

See [-force](#).

`-r|-replace`

Replace the object in the directory with its predecessor. When this option is specified, the list of files in the directory remains unchanged; only the version of the specified object changes.

`-t|-task task_spec`

See [-t|-task task\\_spec](#).

## Examples

- Cut the `sort.c` object from the current project.

```
ccm unuse sort.c
```

```
Member sort.c-1 removed from project ico-1
```

- Cut the `ico_jan5-1` under `Dir` folder of the project `Project_One-1`:

```
ccm unuse -dir Project_One\Dir@Project_One-1 ico_jan5-1:ascii:1 -task
10
```

---

## Related topics

- [use command](#)
- [delete command](#)

---

## Description and uses

Removes an existing file, directory, root directory, or project from the current project or directory. The directory must be checked out to remove members from it; however, if you try to remove an object from a non-modifiable directory, Rational Synergy checks out the directory automatically (unless you specify the `-r` option). You must check **in** the directory to make the changes in the directory available to other users. Note that `unuse` is now called `cut`.

The root directory can be the target of this command, but only when specified with the `-d` and `-r` options. If you want to use a different version of the root directory, use the `use` command. You cannot cut the root directory without replacing it because a project must always have a root directory.

**Note** When you cut an object in a non-modifiable directory, a new directory version is checked out automatically unless you replace the object with a different version.

If you are in a shared project and your current directory is non-modifiable, the directory is checked out and associated automatically with the current (or specified) task and is checked in to the *integrate* state. You can disable the automatic check-in feature by setting `shared_project_directory_checkin` to `FALSE` in your initialization file. (See [shared project directory checkin](#).)

If you want to delete a project, see the [delete command](#) (`ccm delete -p project_name-version`).

You do not need to be in a work area to use this command if you use the [Project reference form](#):

**Windows:** `relative_path\object_name@project_name-project_version`

**UNIX:** `relative_path/object_name@project_name-project_version`

The following is an example of the project reference form and how to use it to delete the root directory, `ico/hi_world.c@final-1`:

```
ccm unuse -d -r final@final-1
```

You will receive a message telling you the *object\_version* that was removed and which version replaced it.

## update command

See [Description and uses](#) for details. The `update_members` command supports the following subcommands:

- [Updating members for a directory](#)
- [Updating members for a project grouping](#)
- [Updating project members](#)

---

## Updating members for a directory

```
ccm u|update|update_members|reconf|reconfigure
      ([-ks|-keep_subprojects] | [-rs|-replace_subprojects])
      [-r|-recurse] [-v|-verbose] dir_spec...
```

*dir\_spec*

Specifies the directory to be updated. You can set *dir\_spec* to multiple directory objects. The *dir\_spec* takes the forms described in [File specification](#).

`-ks|-keep_subprojects`

Specifies to keep subprojects in their current place. If you don't specify `-keep_subprojects` or `-replace_subprojects`, the default depends on how you've set [replace subproj](#).

`-rs|-replace_subprojects`

Specifies to replace subprojects with new subprojects. Subprojects are replaced only if the selection rules choose other versions of those subprojects. If you don't specify `-keep_subprojects` or `-replace_subprojects`, the default depends on how you've set [replace subproj](#).

`-r|-recurse`

Specifies to include subprojects.

`-v|-verbose`

Displays detailed undo update messages.

## Related topics

- [undo update command](#)
- [project grouping command](#)

---

## Updating members for a project grouping

```
ccm u|update|update_members|reconf|reconfigure -pg|-project_grouping  
      ([-ks|-keep_subprojects] | [-rs|-replace_subprojects])  
      [-r|-recurse] [-v|-verbose] project_grouping_spec...
```

*project\_grouping\_spec...*

Specifies the project grouping to be updated. You can set *project\_grouping\_spec* to multiple project groupings. The *project\_grouping\_spec* takes the forms described in [Project grouping specification](#).

`-ks|-keep_subprojects`

Specifies to keep subprojects in their current place. If you don't specify `-keep_subprojects` or `-replace_subprojects`, the default depends on how you've set [replace\\_subproj](#).

`-rs|-replace_subprojects`

Specifies to replace subprojects with new subprojects. Subprojects are replaced only if the selection rules choose other versions of those subprojects. If you don't specify `-keep_subprojects` or `-replace_subprojects`, the default depends on how you've set [replace\\_subproj](#).

`-r|-recurse`

Specifies to include subprojects.

`-v|-verbose`

Displays detailed undo update messages.

## Related topics

- [undo\\_update command](#)
- [project\\_grouping command](#)



---

## Updating project members

```
ccm u|update|update_members|reconf|reconfigure -p|-project
      ([-ks|-keep_subprojects] | [-rs|-replace_subprojects])
      [-r|-recurse] [-v|-verbose] project_spec...
```

*project\_spec*...

Specifies the projects to be updated. You can set *project\_spec* to multiple projects. The *project\_spec* takes the forms described in [Project specification](#).

`-ks|-keep_subprojects`

Specifies to keep subprojects in their current place. If you don't specify `-keep_subprojects` or `-replace_subprojects`, the default depends on how you've set [replace subproj](#).

`-rs|-replace_subprojects`

Specifies to replace subprojects with new subprojects. Subprojects are replaced only if the selection rules choose other versions of those subprojects. If you don't specify `-keep_subprojects` or `-replace_subprojects`, the default depends on how you've set [replace subproj](#).

`-r|-recurse`

Specifies to include subprojects.

`-v|-verbose`

Displays detailed undo update messages.

## Examples

- Update a project named `proj1-1` and replace its subprojects.  

```
ccm update -rs -p proj1-1
```
- Update all projects in the grouping named `All Fox/2.01 Integration Testing Projects`.  

```
ccm update -pg "All Fox/2.01 Integration Testing Projects"
```

## Related topics

- [undo update command](#)
- [project grouping command](#)

---

## Description and uses

The `update` command updates the specified directory, project object, or project grouping. It uses the baseline and tasks of project groupings to find the appropriate candidates and selection rules to select new versions of the members, if appropriate. You can also specify a project grouping to be updated. Note that the `update members` operation is now called `update` in Rational Synergy.

The update process stops if an individual operation within the update fails. For example, if the current version of an object has a work area conflict, the process stops and the new version is not automatically used. This is done to protect the data in the user's work area.

The default setting to stop the update can be changed by modifying your initialization file. Some users may want to continue with the update process, even though an individual failure has occurred. You can set update to continue by setting the [reconf\\_stop\\_on\\_fail](#) option to `False`.

Use project groupings to perform a multiphase build, where the set of projects in a project grouping is not updated all at the same time. In such a case, all the projects are updated using the same baseline and tasks. A developer or build manager can update additional projects in the same project grouping, using the same baseline and tasks, without having that baseline and those tasks refreshed. Thus, it is necessary to calculate and save this baseline and tasks, and to specify that subsequent project updates use this saved baseline and tasks.

You may need to change the update properties of a project. The update properties determine whether a project is updated using tasks and a baseline or object status, and sets parameters that control which objects are selected. Here are some of the properties you may need to change, and the commands to use to change them:

- Change the project's release or purpose with the `ccm attr` command.
- Add and remove tasks for process-rule-based projects, with the `ccm project_grouping` command.
- Set the baseline for custom development purpose process-rule-based projects, with the `ccm project_grouping` command.
- Add and remove tasks, and set the baseline, for manual projects, with the `ccm update_properties` command.

You can also change update properties using the Rational Synergy GUI.

By default and for performance purposes, the update command does not provide parallel version notification when it encounters parallel object versions. You can enable parallel version notification by setting the [reconfigure\\_parallel\\_check](#) user option.

## use command

See [Description and uses](#) for details. The `use` command supports the following subcommands:

- [Adding a project to the current directory](#)
- [Using different versions or adding objects to a specified directory](#)
- [Using different versions or adding objects to the current directory](#)

---

## **Adding a project to the current directory**

This subcommand adds one or more existing projects to the current working directory. The current working directory must be in a maintained work area whose project is modifiable by the user. Use this subcommand to add projects as subprojects. Special handling may be performed in shared projects (see [shared project directory checkin](#)). If the context project is in the *working* state, you must be the owner of that project. If the context project is in the *prep* state, you must be a build manager.

The following applies if you are using a different version of a subcomponent. If the subcomponent is a static subproject with a component that is different than the parent project, the different version of the subproject is associated with the current task.

```
ccm use -p|-project [-t|-task task_spec] project_spec...
```

*project\_spec*

Specifies the projects you are using. See [Project specification](#) for details.

*-t|-task -task\_spec*

Specifies the task that is associated with any directory checked out in order to add a new member. If omitted, the current task is used. When an object is added to a directory, if the directory is in a *static* state such as *integrate*, it is automatically checked out. If the directory is in a state that is writable by you, then the existing directory version is updated with the new member. See [Task specification](#) for details.

### **Example**

- Add the `SubPrj-one:project:1` project to the current directory:

```
ccm use -p -task 31 SubPrj-one:project:1
```

### **Related topics**

- [delete command](#)
- [unuse command](#)

---

## ***Using different versions or adding objects to a specified directory***

This subcommand uses different versions of an object or adds existing objects as new project members under the specified directory. The directory to which the objects are added must be specified in a form that provides a context project, such as a [Project reference form](#) or a [Work area reference form](#).

If a directory entry already exists for an object, then the command uses the specified object under its corresponding directory entry. If a directory entry does not exist for the object, then the directory is automatically checked out against the specified task, a new directory entry is created for the object, and the specified object is used in the context project associated with the specified directory. Special handling may be performed in shared projects (see [shared project directory checkin](#)). If the context project is in the *working* state, you must be the owner of that project. If the context project is in the *prep* state, you must be a build manager.

The following applies if you are using a different version of a subcomponent. If the subcomponent is a static product with a component that is different than the parent project, the different version of the product is associated with the current task.

```
ccm use -p|-project -dir dir_spec [-t|-task task_spec] project_spec...  
ccm use -dir dir_spec [-t|-task task_spec] file_spec...
```

*-dir dir\_spec*

Specifies the directory under which different versions of objects or existing objects are added. The *dir\_spec* is a *file\_spec* (see [File specification](#)) that you can set to a single directory object and provides a context project. A [Project reference form](#) or a [Work area reference form](#) provides such a context project.

*file\_spec*

Specifies the object versions you are using. See [File specification](#) for details.

*project\_spec*

See [project\\_spec](#).

*-t|-task -task\_spec*

See [-t|-task -task\\_spec](#).

## ***Examples***

- Use different version of the project `SubPrj-2`:  

```
ccm use -p SubPrj-2:project:1
```
- Use the version of `clear.c` chosen by the selection rules.

---

```
ccm use -rules clear.c
```

- Add a project 'SubPrj-one:project:1' to the root directory of the project 'TopPrj-top:project:1' (the current directory may be any directory or the projects may or may not have the maintained work areas):

```
ccm use -p -dir TopPrj@TopPrj-top -task 31 SubPrj-one:project:1
```

- Use the object 'a.txt-1.2:ascii:1' to the directory dir1 under the root directory of the project, TopPrj-top:project:1:

```
ccm use -dir TopPrj\dir1@TopPrj-top -task 31 a.txt-1.2:ascii:1
```

- Use a different version for the object a.txt-1.1:ascii:1

```
ccm use -dir TopPrj\dir1@TopPrj-top a.txt-1.1:ascii:1
```

## Related topics

- [delete command](#)
- [unuse command](#)

---

### ***Using different versions or adding objects to the current directory***

This subcommand uses a different version of an object in the current working directory, or adds existing objects as new project members under the current working directory. The current working directory must be in a maintained work area whose project is modifiable by the user. Special handling may be performed in shared projects (see [shared project directory checkin](#)). If the context project is in *working* state, you must be the owner of that project. If the context project is in *prep* state, you must be a build manager.

The following applies if you are using a different version of a subcomponent. If the subcomponent is a static product with a component that is different than the parent project, the different version of the product is associated with the current task.

```
ccm use [-r|-rules|-recommend] [-t|-task task_spec] file_spec...
```

*file\_spec*

See [file\\_spec](#).

`-r|-rules|-recommend`

Uses the version selected by the selection rules.

`-t|-task -task_spec`

See [-t|-task -task\\_spec](#).

### ***Examples***

- Add the `util-b2` and `tools-b2` projects to the current directory.  

```
ccm use -p util-b2 tools-b2
```
- Use the recommended version for `file_top_1.txt` under the current directory:  

```
ccm use -rules file_top_1.txt
```
- Add an existing member `file_sub_1.txt-1` to the current directory:  

```
ccm use -task 29 file_sub_1.txt-1:ascii:1
```

### **Related topics**

- [delete command](#)
- [unuse command](#)

---

## Description and uses

The `use` command performs either of the following operations:

- Replaces an existing file, directory, or project with another version
- Pastes an existing file, directory, or project that is not already in the current directory

When you paste an object to a non-writable directory, a new directory version is checked out automatically.

If you are in a shared project and your current directory is non-writable, the directory is checked out and associated automatically with the default (or specified) task and is checked in to the *integrate* state. You can disable the automatic check-in feature by setting `shared_project_directory_checkin` to `FALSE` in your initialization file. (See [shared project directory checkin](#).)

The following applies if you are using a different version of a subcomponent. If the subcomponent is a static subproject or product with a component that is different than the parent project, the different version of the subproject or product is associated with the current task. If a current task is not set, you will be prompted to set one. You can disable this feature by setting `add_used_subcomponents_to_task` to `FALSE` in your initialization file. (See [add used subcomponents to task](#).)

When you use a directory, the directory is updated automatically. You must check in the directory to make the changes in its content available to other users.



## view command

See [Description and uses](#) for details. The `view` command supports the [Viewing a file](#) subcommand.

---

## **Viewing a file**

```
ccm view file_spec...
```

*file\_spec*

Specifies the object to be displayed. See [File specification](#) for details.

## **Example**

- View version 8 of the `log.c` object.

```
ccm view log.c-8
```

## **Related topics**

- [cat command](#) (UNIX only)
- [edit command](#)

---

## Description and uses

The `view` command shows the specified file. The default viewer is used to view the file.

If the file is specified in a form that provides a context project, such as a [Work area reference form](#) or a [Project reference form](#), and the corresponding work area location is visible to the client, then the viewer is launched on that work area location. If a project context is not available or the corresponding work area is not visible, the viewer is launched with a temporary read-only copy of the file from the database.

## **work\_area command**

See [Description and uses](#) for details. The `work_area` command supports the following subcommands:

- [Modifying work area properties](#)
- [Showing work area properties](#)

---

## Modifying work area properties

This subcommand changes the work area properties of a project including whether the work area is maintained and the work area path. If no project is specified, the command updates the project whose work area is associated with the current working directory.

```
ccm wa|work_area ([-wa|-maintain_wa] | [-nwa|-no_wa])
    ([-cb|-copy_based] | [-lb|-link_based|-ncb|-not_copy_based])
    ([-rel|-relative] | [-nrel|-not_relative])
    ([-mod|-modifiable] | [-nmod|-not_modifiable])
    ([-wat|-wa_time] | [-nwat|-no_wa_time])
    ([-tl|-translate|-translation] | [-ntl|-no_translate|-no_translation])
    [-set|-path|-setpath absolute_path]
    [-pst|-project_subdir_template template_value]
    ([-r|-recurse] | [-nr|-norecurse|-no_recurse])
ccm wa|work_area ([-wa|-maintain_wa] | [-nwa|-no_wa])
    ([-cb|-copy_based] | [-ncb|-not_copy_based])
    ([-rel|-relative] | [-nrel|-not_relative])
    ([-mod|-modifiable] | [-nmod|-not_modifiable])
    ([-wat|-wa_time] | [-nwat|-no_wa_time])
    ([-tl|-translate|-translation] | [-ntl|-no_translate|-no_translation])
    [-set|-path|-setpath absolute_path]
    [-pst|-project_subdir_template template_value]
    ([-r|-recurse] | [-nr|-norecurse|-no_recurse]) [-p|-project]
    project_spec...
```

`-cb|-copy_based`

Specifies that any work area is copy based.

`-lb|-link_based|-ncb|-not_copy_based`

Specifies that any work area is link based. This option is available to UNIX users only.

`-mod|-modifiable_wa`

Specifies that files in the work area have permissions set so they are modifiable even if they have not been checked out. The default is `-nmod|-not_modifiable_wa`.

`-nmod|-not_modifiable_wa`

Specifies that files in the work area have permissions set so they are modifiable by default only if they are in a writable state such as *working*. This is the default.

`-nr|-no_recurse`

Do not recurse the project hierarchy when applying these options. Change only the specified project. This is the default.

---

`-nrel|-not_relative`

Specifies that any work area will be located on an absolute path.

`-ntl|-no_translate|-no_translation`

Specifies that ASCII files in the work area are copied between Windows and UNIX without newline translation. The default is `-tl|-translate`.

`-nwa|-no_wa`

Specifies that the project should not have a maintained work area. This default is `-wa|-maintain_wa`.

`-nwat|-no_wa_time`

Specifies that the files in the project's work area should use timestamps that are the show the Rational Synergy modification time rather than the time they were copied into the work area. This is the default.

`-p|-project`

It is not necessary to specify this option.

*project\_spec*

Specifies the project to be modified. See [Project specification](#) for details.

`-pst|-project_subdir_template] template_value`

Changes the specified work area path for the project (where the project is synchronized to the file system) to a new location. This parameter changes only the project-specific portion of the work area path. To change to a different part of the file system for your work area or synchronize your work area to a different platform, see [set|-path|-setpath absolute\\_path](#).

The default directory in which all project work areas are created is `ccm_wa` followed by the `database_name` in your home directory. By default, the project name and version are appended to the `database_name`. You can change the project-specific portion of the name to include `project_name`, `project_version`, `release`, `platform`, and `delimiter` by modifying the work area template.

If the previous path is visible to the interface host, it is moved to the new location. Otherwise, the work area is created when you execute the `work_area` command with this option.

---

`-r` | `-recurse`

Causes all projects in the project hierarchy to be updated along with the specified project. The default is `-nr` | `-norecurse`.

`-rel` | `-relative`

Specifies that any work area will be located on a path relative to the parent project's path.

`-set` | `-path` | `-setpath` *absolute\_path*

Changes the specified work area path for the project to the new location. This option changes the non-project-specific portion of the work area path. To change the project-specific portion of the name, such as `project_name`, `project_version`, `release`, `platform`, and `delimiter` by modifying the work area template, see [-pst|-project\\_subdir\\_template\]\\_template\\_value](#).

If the previous path is visible to the interface host, it is moved to the new location. Otherwise, the work area is created when you execute the `work_area` command with this option.

You can change the work area path of a read-only project only if you are a build manager or a user in the `ccm_admin` role.

`tl` | `-translate` | `-translation`

Specifies that ASCII files in the work area will be copied between Windows and UNIX with newline translation.

`-wa` | `-maintain_wa`

Maintain a work area. Setting this option synchronizes the work area and keeps it synchronized.

To stop a sync from the CLI, enter `<CTRL+C>` at any time.

However, if you stop the sync, you will receive an error message stating that errors may occur in your work area. The errors will not occur until you try to use the work area. To avoid problems, perform a complete synchronization of the work area before you use it.

You can use this option on a read-only project only if you are in the `ccm_admin` role.

`-wat` | `-wa_time`

Specifies that the files in the project's work area should use timestamps that show the time they were copied into the work area, rather than the Rational Synergy modification time. The default is `-no_wa_time`.

---

## Related topics

- [delimiter command](#)
- [reconcile command](#)



---

## Showing work area properties

This subcommand shows the work area properties of a project. If no project is specified, the command shows the work area properties of the project whose work area is associated with the current working directory.

```
ccm wa|work_area -s|-sh|-show [-r|-recurse] [-f|-format format]
    [-nf|-noformat] ([-ch|-column_header] | [-nch|-nocolumn_header])
    [-sep|-separator separator] ([-sby|-sortby sortspec] |
    [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat]
ccm wa|work_area -s|-sh|-show [-r|-recurse] [-p|-project]
    [-f|-format format] [-nf|-noformat]
    ([-ch|-column_header] | [-nch|-nocolumn_header])
    [-sep|-separator separator] ([-sby|-sortby sortspec] |
    [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat] project_spec...
```

`-ch|-column_header`

Specifies to use a column header in the output format. See [-ch|-column\\_headers](#) for details.

`-f|-format format`

Specifies the command output format. See [-f|-format](#) for details.

A keyword can be built-in (`%fullname`, `%displayname`, `%objectname`) or the name of any existing attribute such as `%modify_time` or `%status`.

See [Built-In keywords](#) for a list of keywords.

`-gby|-groupby groupformat`

Specifies how to group the command output. See [-gby|-groupby](#) for details.

`-nch|-nocolumn_header`

Specifies not to use a column header in the output format. See [-ch|-column\\_headers](#) for details.

`-nf|-noformat`

Specifies not to use column alignment. See [-nf|-noformat](#) for details.

`-ns|-nosort|-no_sort`

Specifies that the command's output will not be sorted. See [-ns|-nosort](#) for details.

---

`p|-project`

See [-p|-project](#).

`project_spec`

Specifies the project to be shown. See [Project specification](#) for details.

`-sby|-sortby sortspec`

Specifies how to sort the command output. See [-ns|-nosort](#) for details.

`-sep|-separator separator`

Used only with the `-f|-format` option. Specifies a different separator character. See [-sep|-separator](#) for details.

`-r|-recurse`

Causes all projects in the project hierarchy to be shown along with the specified project. The default is to only show the specified project.

### **Example**

- Show work area properties:  
`ccm wa -show -recurse project-2`

### **Related topics**

- [reconcile command](#)

---

## **Description and uses**

The `work_area` command shows and modifies work area options.

## Links to Rational Synergy Help

Use the following links to open Rational Synergy Help systems in HTML or PDF:

- Rational Synergy Classic CLI Help, [HTML](#)| [PDF](#)
- Rational Synergy Help, Developers [HTML](#)| [PDF](#)
- Rational Synergy Help, Build Managers [HTML](#)| [PDF](#)
- Explorer Interface Help [HTML](#)| [PDF](#)
- Taskbar Interface Help [HTML](#)| [PDF](#)

---

## Notices

© Copyright 2000, 2009

U.S. Government Users Restricted Rights - Use, duplication, or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

This information was developed for products and services offered in the U.S.A. IBM® may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information about the products and services currently available in your area. 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. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send written license inquiries to:

IBM Director of Licensing  
IBM Corporation  
North Castle Drive  
Armonk, NY 10504-1785  
U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send written inquiries to:

IBM World Trade Asia Corporation  
Licensing  
2-31 Roppongi 3-chome, Minato-ku  
Tokyo 106-0032, Japan

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions. Therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements changes in the products the programs described in this publication at any time without notice.

---

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

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:

Intellectual Property Dept. for Rational® Software  
IBM Corporation  
1 Rogers Street  
Cambridge, Massachusetts 02142  
U.S.A.

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, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information 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.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

## Trademarks

IBM, the IBM logo, and [ibm.com](http://ibm.com) are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at [www.ibm.com/legal/copytrade.html](http://www.ibm.com/legal/copytrade.html).

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

AIX and Informix are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos and Solaris are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

HP and HP-UX are registered trademarks of Hewlett-Packard Corporation.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows XP, Windows Vista other Microsoft products referenced herein are either trademarks or registered trademarks of Microsoft Corporation.

UNIX is a registered trademark of The Open Group in the United States and other countries.

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



# Index

## Symbols

%baseline, 59  
 %change\_request, 59  
 %change\_request\_duplicates, 59  
 %change\_request\_original, 59  
 %change\_request\_release, 59  
 %change\_request\_status, 59  
 %change\_request\_synopsis, 59  
 %displayname, 59  
 %fullname, 59  
 %in\_baseline, 59  
 %in\_build, 60  
 %instance, 60  
 %model, 60  
 %objectname, 60  
 %problem\_duplicates, 60  
 %problem\_original, 60  
 %purpose, 60  
 %requirement\_id, 60  
 %root, 60  
 %sourcename, 60  
 %states, 60  
 %task, 60  
 %task\_platform, 60  
 %task\_release, 61  
 %task\_status, 61  
 %task\_subsystem, 61  
 %task\_synopsis, 61  
 %type, 61  
 .ccm\_addr file, 560  
 @cvid, 21

## A

activecm.disable\_sync\_at\_startup, 69  
 add\_object\_task\_assoc, 69  
 alias, removing, 599  
 allow\_delimiter\_in\_name, 69  
 allow\_prep, 72  
 alphanumerics, 55  
 associating

project and purpose, 175  
 tasks with objects, 571  
 attributes  
   changing, 118  
   copying, 112  
   displaying, 119  
   editing, 118  
   listing, 117  
   modifying, 118  
   new, 114  
   setting required fields, 89  
   showing, 119  
   view settings, 548  
 AUTOMOUNT\_FIX, 97  
 auto-refresh mode, setting for a project  
   grouping, 456

## B

baseline properties, showing, 140  
 baseline\_template, 72  
 baseline\_template\_date\_format, 73  
 baseline\_template\_repl\_char, 73  
 baselines  
   adding projects, 125  
   changing, 134  
   comparing, 122  
   comparing projects, 438  
   creating, 124  
   defining state, 126  
   deleting, 130, 133  
   displaying properties, 140  
   editing, 134  
   listing, 131  
   marking for deletion, 133  
   modifying, 134  
   names, 24  
   naming restrictions, 56  
   previewing, 124  
   publishing, 136  
   recovering, 139  
   releasing, 137  
   restoring, 139  
   restoring deleted, 139  
   showing information, 143

- showing objects, tasks, CRs, 141
- showing properties, 140
- specification syntax, 24
- browser, setting default, 81

## C

- ccm query
  - examples, 482
  - showing object version, 21
- ccm.ini file, 66
  - location of personal file, UNIX, 96
  - location of personal file, Windows, 96
  - location of system file, Windows, 96
- CCM\_ADDR
  - set as ccm\_root, 559
  - usage explained, 97
  - where stored, 560
- ccm\_eng.log
  - location of, 97
  - using to redirect output, 97
- CCM\_ENGLOG, 97
- CCM\_HOME
  - setting variable, 97
  - UNIX location, 9
  - Windows location, 8
- CCM\_PAGER, 97
- ccm\_ui.log
  - location of, 97
  - using to redirect output, 97
- CCM\_UILOG, 97
- change requests
  - recomputing, 261
  - specification, 25
  - syntax, 25
- changing
  - process, 401
  - process rule, 419
  - task, 584
  - work area properties, 635
- check\_release, 74
- checkin
  - objects, 166
  - projects, 162
  - tasks, 165

- CLI, starting a session, 558
- column\_headers option, 50
- columns
  - alignment, 41
  - elements, format, 47
  - headers, format, 50
- command history, showing, 185
- command-line default settings, 67
- commands
  - alias, 106
  - baseline, 121
  - bom, 148
  - candidates, 151
  - cat, 155
  - change\_type, 158
  - checkin, 161
  - checkout, 169
  - cmdhistory, 185
  - conflicts, 190
  - copy\_project, 196
  - copy\_to\_file\_system, 203
  - dcm, 215
  - delete, 286
  - delimiter, 292
  - finduse, 304
  - folder, 318
  - folder\_template, 344
  - groups, 365
  - history, 370
  - ln, 375
  - move, 388
  - process, 394
  - process rule, 410
  - process\_rule, 410
  - project, 432
  - project\_grouping, 436
  - project\_purpose, 467
  - properties, 474
  - query, 479
  - reconcile, 485
  - release, 501
  - set, 543
  - show, 548
  - show\_servers, 554
  - soad, 520

- soad\_scope, 530
  - start, 557
  - stop, 562
  - sync, 565
  - syntax, 18
  - task, 569
  - undo\_update, 602
  - unset, 610
  - unuse, 613
  - update\_members, 620
  - use, 625
  - view, 631
  - work\_area, 634
  - compare\_cmd, 74
  - comparing
    - files to merge, 387
    - folders, 319
  - conflict\_parameters, 77
  - conflicts
    - in merge operation, 387
    - in reconcile operation, 494
    - in update process, 624
    - show task conflicts, 193
    - showing object conflicts, 191
  - controlling database, setting for a process rule, 424
  - copy\_db\_always, 78
  - copy\_project command, 196
  - copy\_to\_file\_system command, 203
  - current directory
    - adding a project, 626
    - removing a project, 616
  - current tasks
    - clearing, 591
    - defined, 591
    - setting, 591
- D**
- data, sending with DCM, 285
  - database definitions
    - creating, 219
    - deleting, 232
    - modifying, 241
  - database specification, syntax, 26
  - databases, 359
    - controlling, 514
    - marking as up-to-date, 240
    - naming restrictions, 56
    - showing IDs, 270
  - date formats
    - baseline template, 73
  - date\_modified, 78
  - DCM (Distributed Configuration Management)
    - restricted characters, 57
    - sending data, 285
    - showing database definition information, 265
    - showing database IDs, 277
    - showing event log information, 274
    - showing event summary information, 272
    - showing generate times, 276
    - showing properties, 268
    - transferring generated packages, 284
  - dcm\_broadcast\_dbid, 78
  - dcm\_time\_sync\_tolerance, 79
  - default\_task\_query, 80
  - default\_version, 80
  - defaults
    - activecm.disable\_sync\_at\_startup, 69
    - add\_object\_task\_assoc, 69
    - allow\_delimiter\_in\_name, 69
    - allow\_prep, 72
    - baseline\_template, 72
    - baseline\_template\_date\_format, 73
    - baseline\_template\_repl\_char, 73
    - check\_release, 74
    - command line, 67
    - compare\_cmd, 74
    - conflict\_parameters, 77
    - copy\_db\_always, 78
    - date\_modified, 78
    - dcm\_broadcast\_dbid, 78
    - dcm\_time\_sync\_tolerance, 79
    - default\_task\_query, 80
    - default\_version, 80

- engine\_host, 80
- expand\_on\_checkin, 81
- html\_browser, 81
- html\_location, 81
- include\_required\_tasks, 82
- initial\_role, 82
- initials, 82
- mail\_cmd, 83
- multiple\_local\_proj\_instances, 83
- personal, 66
- proj\_idx\_wa\_cache, 85
- project\_subdir\_template\_unix, 85
- range\_for\_keyword\_expand, 86
- reconcile.control\_files\_below\_new\_  
project, 87
- reconcile.save\_uncontrolled, 87
- reconf\_consider\_all\_cands, 87
- reconf\_stop\_on\_fail, 87, 88
- reconfigure\_parallel\_check, 88
- reconfigure\_using\_tasks, 88
- release\_phase\_list, 89
- required\_attributes, 89
- restrict\_reconf\_setting, 90
- role, 90
- save\_to\_wastebasket, 91
- shared\_project\_directory\_checkin,  
91
- start\_day\_of\_week, 91
- system\_filename\_filters, 92
- system-wide, 66
- text\_viewer, 75
- update\_on\_checkin\_if\_equal, 93
- verbosity, 93
- wa\_path\_cache\_size, 94
- wa\_path\_template, 94
- wastebasket, 93
- where stored, 66
- where to set, 66
- work area directory, 566
- defining required fields, 89
- delimiter
  - defined, 293
  - UNIX, 9
  - Windows, 8
- delimiter command, 292

- directories
  - adding a project, 626
  - creating modifiable version of project  
hierarchy, 197
  - merging, 387
  - removing files, 613
  - removing projects, 614
  - replacing, 625
  - updating, 620
  - updating members, 621
  - when checked out automatically, 171
  - where added when new, 214
- directory contents
  - listing objects in long format, 380
  - listing, UNIX, 379
  - listing, Windows, 295
- directory update
  - reversing, 603
  - undoing, 603
- DISPLAY, 97
- displaying properties with format, 476
- Distributed Configuration Management  
(DCM)
  - showing database definition  
information, 265
  - showing database IDs, 265
  - showing event log information, 274
  - showing event summary information,  
272

## E

- editing a process, 401
- editing a process rule, 419
- editing a task, 584
- editing work area properties, 635
- engine
  - log file, 97
- engine\_host, 80, 81
- environment variables
  - CCM\_ADDR, set as ccm\_root, 559
- expand\_on\_checkin, 81
- expression, query construction, 484

**F**

- fields, defining required, 89
- file names, 27
- file specification syntax, 27
- files
  - .ccm\_addr, 560
  - adding, 625
  - adding relationships, 496
  - case-sensitive names, 58
  - ccm.ini, 96
  - ccm\_eng.log, 97
  - ccm\_ui.log, 97
  - comparing/merging, 384
  - editing, 301
  - file contents form, 23
  - finding where used, 304
  - local copy marks, 380
  - merged, how annotated, 387
  - merging, 387
  - moving, 388
  - new, where added in project, 214
  - removing, 613
  - replacing, 625
  - showing not synchronized, 296
  - specification, 27
- floating objects, adding to project, 214
- folder templates
  - adding to a process rule, 411
  - changing, 355
  - creating, 345
  - deleting, 350
  - listing, 353
  - modifying, 355
  - removing, 423
  - setting the controlling database, 359
  - showing, 353
  - showing information, 362
  - showing properties, 361
  - specification, 31
- folders
  - adding to a process rule, 411
  - comparing, 319
  - copying, 322
  - creating new, 324
  - deleting, 328
  - listing, 331
  - modifying, 333
  - showing information, 341
  - showing objects, 339
  - showing properties, 338
  - showing tasks, 339
  - specification syntax, 30
- format options, 48
  - column alignment, 41
  - column format, 47
  - column headers, discussed, 50
  - column\_headers option, 50
  - defined, 48
  - examples, usage, 52
  - format strings, 41
  - format strings, advanced usage, 42
  - groupby, 49
  - groupby example, 53
  - keywords, 43
  - nocolumn\_headers option, 50
  - noformat option, 42
  - nosort example, 52
  - nosort option, 48
  - property formatting, 45
  - separator option, 41
  - sortby option, 49
  - sorting and grouping, 48
  - truncate example, 54
  - unnumbered option, 51
  - wrap example, 54
- format strings
  - advanced usage, 42
  - defined, 48
- forms
  - cvid reference, 22
  - file contents, 23
  - object name, 20
  - project reference, 28
  - query selection sets
    - reference, 21
- functions, querying and sorting, 484

## G

- general usage information, 6
- groupby
  - example, 53
  - option, 49
- groups
  - assigning to objects, 366
  - unassigning from objects, 367

## H

- help, specifying alternate location, 81
- hierarchies, deleting, 287
- history
  - clear entries, 186
  - command, clearing, 185
  - delete, 186
  - set maximum commands to record, 188
  - show current commands, 187
- history, showing, 370
- HOME, 97
- HTML
  - browser default, 81
  - help files location default, 81
- html\_browser, 81
- html\_location, 81

## I

- IBM Customer Support, 15
- include\_required\_tasks, 82
- initial\_role, 82
- initialization file, 96
  - location, 66
  - personal, 96
  - system, 96
  - where to make personal entries, 66
- initials, setting option, 82

## K

- keywords
  - built in, 59
  - change behavior, 147

- format options in, 43
- using attribute names, 59

## L

- LD\_LIBRARY\_PATH, 97
- legal notices, copyright information, 644
- links
  - removing, 613
  - replacing, 625
- listing
  - folder templates, 353
  - folders, 331
- local copies
  - defined, 380
  - marks, 296

## M

- mail\_cmd, 83
- managers
  - defined, 64
  - process rules, 64
  - project purpose manager, defined, 64
  - release, defined, 64
- marks
  - NS (not synchronized), 381
  - UN (uncontrolled), 381
- members, deleting, 451
- merging
  - and conflicts, 387
  - annotating a merged file, 387
  - directories, 387
  - files, 387
- modifiable version of project hierarchy, creating, 197
- modifying a process, 401
- modifying a process rule, 419
- modifying a task, 584
- modifying work area properties, 635
- multiple\_local\_proj\_instances, 83

## N

- naming restrictions

- databases, 56
- objects, 55
- releases, 56
- nocolumn\_headers option, 50
- noformat option, 42
- nosort, format example, 52
- notices, legal, 644
- NS, (not synchronized marks), 381
- numbered format, 48

## O

- object name form, 20
- object names
  - baseline, 24
  - file, 27
  - project reference form, 28
  - selection set reference form, 21
  - work area reference form, 27
- objects
  - adding, 625
  - adding to a specified directory, 627
  - adding to a transfer set, 217
  - adding to current directory, 629
  - assigning groups, 366
  - changing a version, 627, 629
  - checking in, 166
  - checkpoint, 179
  - controlling access to, 368
  - creating, 211, 214
  - creating relationships between, 497
  - displaying in selection set, 21
  - finding in projects, 28
  - finding uses of, 315
  - floating, adding to projects, 214
  - getting latest version, 624
  - instance of, 20
  - locating, 479
  - moving, 390
  - name length limit, 27
  - new, 211
  - querying, 480
  - querying for, 586
  - removing from a transfer set, 264
  - removing from projects, 617

- renaming, 390
- saving for personal use, 179
- searching for in database, 479
- showing relationships, 498
- showing source for, 631
- specification, 32
- specification syntax, 24, 27
- unassigning groups, 367
- using a different version, 627, 629
- option delimiter
  - UNIX, 9
  - Windows, 8
- options
  - groupby, 49
  - implicitly setting, 547
  - section in initialization file, 66
  - separator, 41
  - setting in ccm.ini files, 68
  - unnumbered, 51
  - verbosity, 93
  - where to set initial values, 547

## P

- PAGER, 98
- PATH, 98
- path
  - CCM\_HOME - UNIX, 9
  - CCM\_HOME - Windows, 8
  - defining non-project-specific directory, 94
  - defining project-specific directory, 85
- performance, improving, 94
- personal
  - ccm.ini file, location, UNIX, 96
  - ccm.ini file, location, Windows, 96
  - default settings, 66
- PRINT\_EDIT\_CMD, 98
- PRINT\_TOOL\_CMD, 98
- problem *See change request*
- process rules
  - adding folder templates, 411
  - adding folders, 411
  - changing, 419
  - copying, 413

- creating, 429
- defined, 429
- deleting, 415
- displaying, 416
- displaying properties, 427
- editing, 419
- modifying, 419
- removing folders, 423
- setting the controlling database, 424
- showing, 416, 515
- showing baseline projects, 425
- showing folders, 425
- showing information, 428
- showing members, 425
- showing properties, 427
- specification, 34
- standard behavior, 430
- using with project grouping, 465
- processes
  - changing, 401
  - copying to a new process, 396
  - copying to an existing process, 395
  - creating, 397
  - deleting, 398
  - displaying, 399
  - displaying properties, 405
  - editing, 401
  - listing, 399
  - modifying, 401
  - showing, 399
  - showing information, 403
  - showing properties, 405
  - specification, 33
  - Standard Process, 408
- proj\_idx\_wa\_cache, 85
- project grouping update
  - reversing, 605
  - undo, 605
- project groupings
  - adding tasks to update properties, 437
  - comparing projects, 438
  - copying tasks among projects, 450
  - defined, 465
  - deleting groupings and members, 451
  - displaying, 452
  - listing, 452
  - removing tasks from update properties, 455
  - reversing update of, 605
  - setting the auto-refresh mode, 456
  - showing baselines, 459
  - showing information, 462
  - showing objects, 459
  - showing objects, baselines, tasks, 459
  - showing projects, 459
  - showing properties, 457
  - showing tasks, 459
  - specification, 36
  - undoing update, 605
  - updating, 605, 620
  - updating baseline and tasks, 464
  - updating members, 622
- project members, updating, 623
- project purposes
  - creating, 468
  - manager, defined, 64
  - new, 468
  - showing, 471
- project reference form, 28
- project update
  - reversing, 604
  - undoing, 604
- project\_subdir\_template\_unix, 85
- projects
  - adding new objects to, 214
  - adding to a current directory, 626
  - checking in, 162
  - checking out, 197
  - comparing project grouping and baseline, 438
  - copying, 197
  - copying projects, 197
  - creating a top level project, 207
  - creating as floating object, 214
  - creating writable version of, 197
  - making a copy of, 203
  - names, 35



- new, 207
  - removing objects from, 617
  - renaming, 389
  - replacing, 625
  - show task conflicts, 193
  - showing object conflicts, 191
  - specification, 35
  - viewing, 549
  - properties
    - of objects, formatting, 45
    - showing, 475
    - showing with format, 476
- Q**
- queries
    - controlling format of, 59
    - expression construction, 484
    - for tasks, 586
    - functions and sorting, 484
    - selection set reference form, 21
  - querying for objects, 480
- R**
- range\_for\_keyword\_expand, 86
  - Readme contents, 7
  - reconcile.control\_files\_below\_new\_project, 87
  - reconcile.save\_uncontrolled, 87
  - reconf\_consider\_all\_cands, 87
  - reconf\_stop\_on\_fail, 87, 88
  - RECONF\_TIME, 98
  - reconfigure\_parallel\_check, 88
  - reconfigure\_using\_tasks, 88
  - recursively deleting, 287
  - relationships
    - defined, 496
    - deleting, 580, 607, 608
    - showing, 498
    - using relate command, 500
  - release\_phase\_list, 89
  - releases
    - creating, 502
    - deleting, 507
    - displaying information, 517
    - listing, 508
    - modifying, 510
    - naming restrictions, 56
    - new, 502
    - setting the controlling database, 514
    - showing information, 517
    - showing process rules, 515
    - specification, 37
  - removing
    - a project from a directory, 614, 616
    - symbolic links, 619
  - required fields, defining, 89
  - required\_attributes, 89
  - restrict\_reconf\_setting, 90
  - restricted
    - characters, 55
    - DCM characters, 57
    - names, 55
  - reversing a directory update, 603
  - reversing a grouping project update, 605
  - role, setting default, 90
- S**
- save offline and delete
    - command, 520
    - create, 521
    - preview, 521
  - save\_to\_wastebasket, 91
  - security
    - applying settings, 368
    - assigning levels, 368
    - set read, 368
  - selection set, showing, 480
  - selection sets
    - defined, 21
    - ordering and use, 484
    - reference form, 21
  - separator option, 41
  - servers, showing, 554
  - sessions
    - CCM\_HOME variable, 97
    - interface address for, 560
    - starting, 560
    - stopping, 562

- setting
    - file patterns, ignore on sync, 92
    - required fields at task completion, 89
  - settings, viewing attributes, 548
  - shared projects, states of files created in, 214
  - shared\_project\_directory\_checkin, 91
  - SHELL variable, 98
  - showing properties with format, 476
  - showing query selection set, 480
  - showing work area properties, 639
  - soad command, 520
  - soad\_scope command, 530
  - specifications
    - baseline, 24
    - change requests, 25
    - database, 26
    - file, 27
    - folder, 30
    - folder template, 31
    - object, 32
    - process, 33
    - process rules, 34
    - project, 35
    - project grouping, 36
    - release, 37
    - task, 38
    - transfer set, 39
  - Standard process, 408
  - start\_day\_of\_week, 91
  - strings format, defined, 48
  - symbolic links
    - creating, 376
    - removing, 619
  - sync
    - command, 565
    - stopping, 568
  - syntax
    - baseline specification, 24
    - change request specification, 25
    - cvid reference form, 22
    - database specification, 26
    - file contents form, 23
    - file specification, 27
    - folder specification, 30, 31
    - folder template specification, 31
    - for commands, 18
    - for projects, 35
    - object name form, 20
    - object specification, 32
    - process rules specification, 34
    - process specification, 33
    - project grouping specification, 36
    - project reference form, 28
    - project specification, 35
    - release specification, 37
    - selection sets, reference form, 21
    - task specification, 38, 39
    - transfer set specification, 39
    - work area reference form, 27
  - system ccm.ini file, location of, Windows, 96
  - system default settings, 66
  - system\_filename\_filters, 92
- ## T
- tasks
    - assigning, 570
    - associated change requests, 593
    - associated objects, 593
    - associating with objects, 571
    - change requests, 580
    - changing, 584
    - changing relationships, 581
    - changing states, 597
    - checking in, 165
    - clearing, 591
    - copying, 575
    - copying among projects, 450
    - creating, 578
    - creating and assigning, 591
    - current, 591
    - deleting relationships, 580
    - disassociating, 580
    - displaying information, 595
    - editing, 584
    - finishing, 573
    - fixing, 581
    - modifying, 584

- new, 578
- objects, 580
- objects, tasks
  - change requests, 571
- querying for, 586
- relationships, 571
- removing from update properties, 455
- searching, 586
- setting, 591
- setting current, 591
- setting required fields at completion, 89
- showing information, 595
- showing objects, change requests, tasks, 593
- showing properties, 592
- specification, 38
- transitioning to a different state, 597
- unrelate tasks, 580

TERM, 98

text\_viewer, 75

time update operation, 98

top level projects, creating, 207

trademarks, 646

transfer packages

- generating, 236
- receiving, 257

transfer sets

- adding objects, 217
- creating, 224
- deleting, 233
- editing, 245
- modifying, 245
- recomputing indirect CR members, 261
- recomputing indirect release members, 262
- recomputing members, 263
- removing objects from, 264
- showing, 282
- showing information, 279
- showing members, 280
- specification, 39

transferring generated packages, 284

## U

UC (uncontrolled), defined, 297

UIDPATH, 98

UNC (universal naming convention), defined, 8

uncontrolled (UC)

- defined, 297
- marks, 297

uncontrolled marks, 381

undo a project grouping update, 605

undoing a directory update, 603

universal naming convention (UNC), defined, 8

unnumbered option, 51

unrelate

- objects, 609
- tasks, 580

unset variables, 612

unuse command, using to delete, 613

update

- directory members, 621
- display and time, 98
- list candidates, 154
- project grouping setting for consistency, 465
- reversing, 602, 604, 605
- undo command, 605
- undoing, 604, 605

update properties

- adding tasks, 437
- removing, 455

update\_on\_checkin\_if\_equal, 93

updating work area, 565

USER, 98

users

- defining groups, 368
- restricting access to objects, 368

## V

variables

- AUTOMOUNT\_FIX, 97
- CCM\_ADDR, 97
- CCM\_ENGLOG, 97

- CCM\_HOME, 97
- CCM\_PAGER, 97
- CCM\_UILOG, 97
- DISPLAY, 97
- HOME, 97
- implicitly set, 612
- LD\_LIBRARY\_PATH, 97
- PAGER, 98
- PATH, 98
- PRINT\_EDIT\_CMD, 98
- RECONF\_TIME, 98
- SHELL, 98
- TERM, 98
- UIDPATH, 98
- USER, 98
- verbose, update messages, 93
- verbosity option, 93
- version
  - length limit, 27
  - showing history, 370
- viewing properties with format, 476

## W

- wa\_path\_cache\_size, 94
- wa\_path\_template, 94
- wastebasket, 93
- work area
  - changing options, 634
  - creating projects, 214
  - defining path, 85
  - properties, modifying, changing,
    - editing, 635
  - reconciling, 485
  - reference form, 27
  - showing properties, 639
  - update, 565
  - updating, 565
  - where created by default, 566
- work\_area command, 634

---