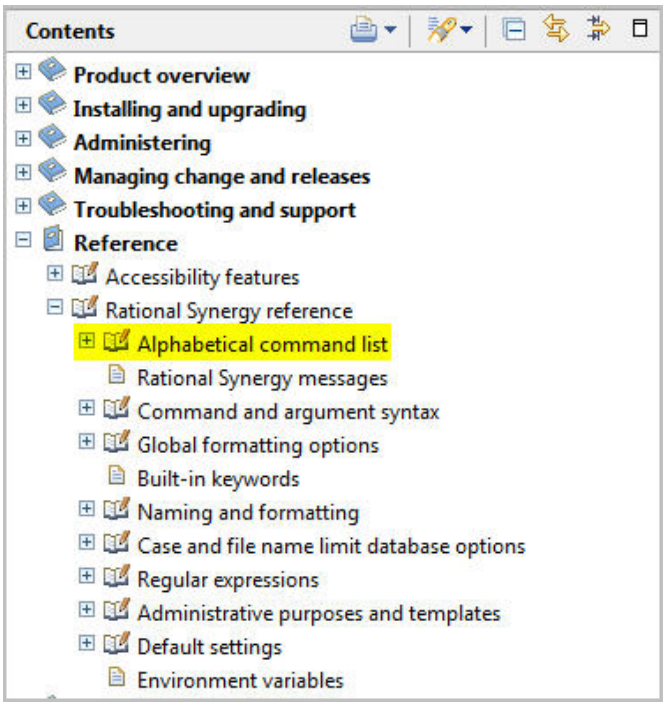
IBM® Rational® Synergy

CLI Alphabetical command list

You can print a section of topics from the IBM® Rational® Synergy 7.2 Information center. Also, with a local PDF distiller, you can print your topics directly to a PDF file. For detailed steps, see "Printing topics and creating PDFs."

However, the function to print a PDF of the topics contained in the CLI Alphabetical command list section does not work at this time. This PDF document contains the information under the **Alphabetical command list** section of the Information center.



In addition to the information in this document, you will find the following sections of the Synergy 7.2 Information center helpful. These sections can be printed or made into a PDF by following the steps in the "Printing topics and creating PDFs" topic.

- Command and argument syntax
- Naming and formatting
- Regular expressions
- Default settings
- Global formatting options

alias command

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

The alias command supports these subcommands:

- Defining an alias
- Showing aliases
- Showing an alias

Defining an alias

You can define a new alias. The alias_name is the name of the new alias to create.

About this task

```
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.

Example

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

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

When you use the new alias, it uses this form:

```
ccm getf myversion main.c
```

- Change the value of an alias.

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

For example, assume that 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`"] is 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
```

Showing aliases

You can show the defined aliases for the current session.

About this task

```
ccm alias
```

Example

- List all defined aliases.

```
ccm alias
```

Showing an alias

You can show a specified alias.

About this task

```
ccm alias alias_name
```

alias_name

Specifies the name of the alias to show.

attribute command

You can manipulate the attributes associated with objects in various ways.

The `attribute` command supports these subcommands.

- Copying attributes
- Creating an attribute
- Deleting an attribute
- Listing attributes
- Modifying an attribute
- Showing attributes

Copying attributes

You can copy attributes from an object to a specified object, a project to a specified project, and from a project to specified subprojects. You can also append attributes from an object to a specified object.

About this task

```
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_pr`*object*`_spec` and multiple objects for `to_pr`*object*`_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

The attributes of an object, also known as its properties, enable you to track various information. You can create attributes and set properties for the attributes, such as *type*and *value*.

About this task

```
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 these situations 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 exists.

-t|-type *attr_type*

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

- 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
```

Deleting an attribute

You can delete an attribute on an object or project by specifying the name of the attribute and specifying the object or project.

About this task

```
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.

Example

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

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

Listing attributes

You can list local, inherited, or all attributes of an object or project.

A local attribute is an attribute stored locally on and owned by an object. An inherited attribute is inherited from the type definition of an object or the recursive super-types of the type definition.

About this task

```
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 all inherited attributes.

Modifying an attribute

You can change the values of attributes.

About this task

```
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.

-v|-value *value*

Specifies the value of the attribute.

Example

Change the release attribute of `main.c` to `4.2_int`.

```
ccm attr -m release -v 4.2_int main.c
```

Showing attributes

You can show the value of an attribute on an object or project.

About this task

```
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
```

baseline command

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, a baseline is used 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 Rational Change, you can use baselines to generate change request reports.

Typically, a build manager creates a baseline; developers do not create a baseline because they do not make builds available to other users.

You might 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 the database in case it is needed later to create a fix for a build.

Creating a baseline for each `Integration Testing` and `System Testing` build helps testers and developers to refer to the set of changes that were used to create the build. Typically, you 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 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 might create a baseline named `Integration Build 20040913` containing static projects `toolkit-int_20040913`, `calculator-int_20040913`, and so on. The numeric designation is the date (`yyyymmdd`) the baseline was created.

A process rule specifies 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, the `Integration Testing` process rule for the current release specifies to use the `Integration Build 20040913` baseline. A development project called `calculator-bob` selects `calculator-20040913` as its baseline project.

Using baselines has many benefits.

- Build managers have a lightweight way to save a set of projects that were successfully built and tested.
- Process rules are flexible. They can specify a particular baseline or the latest baseline with certain characteristics so build managers can control the team process precisely. If problems

were discovered in a newer baseline, the build manager can reset the team baseline to a previous baseline.

- The update operation uses the baseline to streamline which tasks are evaluated, which improves 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 the projects 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 to see what features to test, and whether to expect a known problem to be fixed in a particular build.
- Specify to update a project to match the latest successful build.

How a new baseline is created

Both *prep* state projects and static projects can 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. 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 `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 (and 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 build management projects are checked out from the new baseline projects. Project histories are updated to 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, enable work area maintenance after the baseline is 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 fails. 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 might not have been copied to the database. In such a case, the baseline contains what is in the database, not what is in the non-visible work area. To avoid this problem, the build manager must synchronize changes to all non-visible work areas of projects that are added to a baseline. Complete this operation before adding such projects to a baseline.

You must work as a build manager to create or release a baseline. You must work 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, 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 are reversioned.

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 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 makes 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. A warning is also 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 correct file permissions, 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. A `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:

`%{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.

The `baseline` command supports these 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

You can compare two baselines.

About this task

```
ccm baseline -compare [-tasks] [-objects] [-projects] [-change_requests]
             baseline_spec1 baseline_spec2
```

baseline_spec1

Specifies the first baseline to be compared. See Baseline specification for more information.

baseline_spec2

Specifies the first baseline to be compared. See Baseline specification for more information.

-cr|-change_request|-change_requests

Specifies the baseline comparison to 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
```

Creating or previewing a baseline

You can create or preview 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.

About this task

```
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])
             ([-cpu|-check_project_update] | [-nocpu|-nocheck_project_update])
             [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.

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.

-cpu|-check_project_update

Specifies to check if any projects require an update before creating the baseline. The check represents projects that were not updated since the project grouping's baseline and tasks were last refreshed.

The default is determined by the `ccm.cli.baseline.create.checkprojectupdate` setting.

-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.

-nocpu|-nocheck_project_update

Specifies not to check if any projects require an update before creating the baseline.

The default is determined by the `ccm.cli.baseline.create.checkprojectupdate` setting.

-no_subprojects

Specifies that when a project is added, not to add its subprojects. This option impacts projects added using `-project`, `-project_grouping` and `-baseline`. A subproject is included if it is explicitly specified as a project or is a member of a specified project grouping or baseline.

If `-project_grouping` or `-baseline` is not specified, the default is `-subprojects`. If `-project_grouping` or `-baseline` is specified, the default is `-no_sub_projects`.

-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 the default 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.

The `–subprojects`, `-no_subprojects`, and `–all_subprojects` options 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 this precedence:

- If a baseline is specified, the first specified baseline purpose is used.
- If no baseline is specified, but a project grouping is specified, the first project grouping purpose is specified.
- If neither a baseline or project grouping is specified, the purpose of the first specified project is used.

A `purpose` is a setting that specifies the use of a project (for example, Insulated Development, Integration Testing, System Testing).

If you specify `-purpose`, you must also specify `-release`.

-rehearse

Lists the projects and products that make up the baseline and the name of the baseline to be created. Does not create the baseline.

If any version conflicts are found, a warning lists all the product and project versions that are in conflict. The conflicts exist because the resulting version 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 this precedence:

- If a baseline is specified, the first specified baseline release is used.
- If no baseline is specified, but a project grouping is specified, the first specified project grouping release is used.
- If neither a baseline or project grouping is specified, Synergy uses the release of the first specified project.

If you specify `-release`, you must also specify `-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 do not get it automatically as the latest baseline. SQE can use it for testing. After it passes testing, the build manager must move 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 match only 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 `-project_grouping` or

`-baseline` is not specified. If `-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 setting 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 makes 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 do not specify `–version_template`, the default template is used. For more information, see "Version template specification" in baseline command.

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 -
  purpose "Integration Testing" -project proj1-sqa_3 -subprojects
  ```

Creating component tasks for a baseline

You can create component tasks matching the release and purpose of specified baselines.

Before you begin

You must be in the *build_mgr* or *ccm_admin* role to create component tasks for a baseline.

About this task

```
ccm baseline -cct|-create_component_tasks baseline_spec...
```

baseline_spec...

Specifies the baselines to create component tasks for. See Baseline specification for more information.

Example

- Create component tasks for a baseline called `2.0_build_34`.

```
ccm baseline -create_component_tasks 2.0_build_34
```

Deleting a baseline

You can delete the specified baselines, optionally deleting the baseline projects and products.

Before you begin

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.

About this task

```
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 projects and products are not 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 are deleted.

Example

- Delete a baseline named `20090213`.

```
ccm baseline -delete 20090213
```

Listing baselines

You can list the baselines that match the specified criteria. If no criteria are specified, all baselines are listed.

About this task

```
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_header for details.

-f|-format "*format_string*"

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_header for details.

-nf|-noformat

Specifies not to use column alignment. See -nf|-noformat for details.

-ns|-no_sort

Specifies not to sort the command output. See -ns|-nosort for details.

-purpose *purpose*

Specifies to list only baselines with the specified purposes. If purposes are not specified, then baselines with any purpose are listed.

-release *release_spec*

Specifies to list only baselines with the specified releases. If releases are not specified, then baselines with any release are listed.

-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 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.

Marking a baseline for deletion

You can mark a baseline for deletion. This mark moves the baseline to the *deleted_baseline* state, which prevents the baseline from being used by the update members operation.

Before you begin

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.

About this task

```
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.

Modifying a baseline

You can modify the name or build of the specified baselines. You can also update the projects and products to use versions that match the current default baseline version template.

Before you begin

Build managers and users in the *ccm_admin* role can modify a baseline.

About this task

```
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 setting 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 are not 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 is not visible or because it cannot 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" in baseline command.

Publishing a baseline

You can publish the specified baselines and move them to the *published_baseline* state. Publication makes the baseline eligible for selection during an update members operation.

Before you begin

Build managers or users in the *ccm_admin* role can complete this move.

About this task

```
ccm baseline -publish baseline_spec...
```

-publish *baseline_spec...*

Specifies the baselines to be published. Moves baselines in the *test_baseline* state to the *published_baseline* state. See Baseline specification for more information.

Releasing a baseline

This subcommand releases the specified baselines and moves them into the *published_baseline* state. This move makes the baseline eligible for selection during an update members operation and indicates that the baseline is suitable for release.

Before you begin

Build managers and users in the *ccm_admin* role can release a baseline.

About this task

```
ccm baseline -rb|-release_baseline [-c|-comment comment]
            [-ce|-commentedit] [-cf|-commentfile file_path] baseline_spec...
```

baseline_spec...

Specifies the baseline to be released. See Baseline specification for more information.

-c|-comment *comment*

Specifies to append a comment 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 to invoke the default text editor to compose and edit the comment. 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 to use the contents of the specified file for the comment. If you specified `-comment`, it is appended to that comment. You can use this option with the `-commentedit` option.

Restoring a deleted baseline

This subcommand reverses the mark for deletion operation on the specified baselines. This operation must be performed before the baselines are deleted from the database.

Before you begin

Build managers and users in the *ccm_admin* role can restore a deleted baseline.

About this task

```
ccm baseline -undelete baseline_spec...
```

baseline_spec...

Specifies the baseline to be restored. See Baseline specification for more information.

Showing a baseline property

This subcommand shows a specified property for the specified baselines.

About this task

```
ccm baseline -sh|-show ((r|release) | (p|purpose) | (o|owner) |
            (desc|description) | (b|build)) baseline_spec...
```

baseline_spec...

Specifies the baseline to be released. See Baseline specification for more 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.

About this task

```
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 to be shown. See Baseline specification for more information.

-ch|-column_header

Specifies to use a column header in the output format. See -ch|-column_header 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_header for details.

-nf|-noformat

Specifies not to use column alignment. See -nf|-noformat for details.

-ns|-no_sort

Specifies not 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.

-sby|-sortby *sortspec*

Specifies how to sort the command output. See -sby|-sortby for details.

-u|-unnumbered

Suppresses automatic numbering of the command output. See -u|-unnumbered for details.

Showing baseline information

This subcommand shows general information about the specified baselines.

About this task

```
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 to be shown. See Baseline specification for more information.

-ch|-column_header

Specifies to use a column header in the output format. See -ch|-column_header 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_header 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.

bom command

This subcommand shows the Bill-of-Materials for a product. The `bom` command supports the “Showing a bill of materials” subcommand:

Showing a bill of materials

Use this command to show a Bill-of-Materials (BOM).

About this task

```
ccm bom file_spec...
```

file_spec

Specifies the product for which the Bill-of-Materials is displayed. This information exists for a controlled product only. See File specification for details.

candidates command

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 the same attribute values 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.

The candidates command supports the “Showing candidates” subcommand.

Showing candidates

Candidates are all the file versions that are eligible to be used in a directory entry. You can show candidates for an object or directory entry, and format the output in various ways.

About this task

```
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 -nch|-nocolumn_header for details.

-nf|-noformat

Specifies not to use column alignment. See -nf|-noformat for details.

-ns|-nosort|-no_sort

Specifies not to sort the command output. See -ns|-nosort for details.

-r|-recommend

Results in the version being recommended. The recommended version is selected based on the 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 output (that is, the output is not numbered). 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 *
```

cat command

You can display 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.

The cat command supports the "Showing source contents" subcommand.

Showing source contents

Use the `cat` command to display the source of an object.

About this task

```
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 `main.c-9` object version, which is a `csrc` object.

```
ccm cat main.c-9:csrc:2
```

change_type command

You can change the type of a specific file or directory.

The change_type command supports the "Changing the type for an object" subcommand.

Changing the type for an object

You can change the type of a specific file or directory. When you change a type, a version of the object is created with the specified type. If the specified object is in the *working* state, it is replaced with a new object, and the specified object is deleted 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.

The command fails if the object is a member of more than one project. The command also fails if you do not execute the command within the project where the object is a member.

About this task

```
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 an automatically checked out directory and new object with the specified task. If the current task is set and you do not specify a different task, the checked out directory and new 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 for the object. You can set *new_type* to a single object.

checkin command

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, and assign task numbers to objects. Additionally, you can add, modify, or replace a comment for the object you to check in.

Note: Work in one work area at a time, and perform check in operations with that work area visible.

The `checkin` command supports these subcommands:

- Checking in a project
- Checking in a task
- Checking in an object

Checking in a project

You can check 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*, the members of a project must also be non-writable. If the work area is being maintained, run the command on a client that is visible to the work area for the project. Changes made in a visible work area can be automatically synchronized back to the database.

About this task

```
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_string*

Specifies to append a comment 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 to invoke the default text editor to compose and edit the comment. 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 to use the contents of the specified file 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 comment 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 you do not enter a comment for any of the comment options, you are prompted for a comment. The comment 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, `-products` applies to all product members in the hierarchy.

-p|-project

Specifies that any subprojects within the specified project are checked in. If `-h|-hierarchy` is also specified, all subprojects in the hierarchy are also checked in.

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. The product state applies to hierarchical and non-hierarchical check-ins (that is, the `-product_state` option does not require the `-state` 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, all source objects in the hierarchy are checked in.

-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.

Attention: To check in a project version to a non-modifiable state, be sure that all members are in a non-modifiable state already. You cannot check in a project to a non-modifiable state if it has modifiable members.

Example

- 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
```

Checking in a task

You can complete a task, which checks in the objects associated with the task to a non-modifiable state and transitions the task to the *completed* state.

The `ccm ci -task` command is equivalent to the `ccm task -complete` command. You must be the resolver of the task or an administrator to use this command.

About this task

```
ccm ci|checkin -task (task_spec|(current|default))
               [-time|-time_actual task_duration]
               [-parallels|-check_parallels (none | (i|info|information) |
               (check|enforce))]
               [-c|-comment comment_string] [-ce|-commentedit]
               [-cf|-commentfile file_path] [-cr|-commentreplace]
```

-c|-comment *comment_string*

Specifies to append a comment 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 to invoke the default text editor to compose and edit the comment. The result saved from the text editor is used as the final comment. You can use this option with the `-comment` and `-commentfile` options.

-parallels|-check_parallels (none | (i|info|information) | (check|enforce))

Specifies the kind of parallel checking to be performed when checking in the task.

- `none` means no checking for parallels on the task.
- `i|info|information` means check and show any parallels on the task.
- `check|enforce` means check for parallels on the task. If parallels exist, do not complete the task and return with a non-zero exit status.

-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.

Checking in an object

You can check in specific objects, such as files and directories. Run this command on a client that is visible to the work area associated with the project, if the work area is maintained. Changes made in a visible work area can be automatically synchronized to the database.

This subcommand applies when you specify one or more arguments, and you do not specify the `-project` option.

About this task

```
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*

Specifies to append a comment 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 to invoke the default text editor to compose and edit the comment. 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 to use the contents of the specified file 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 comment is appended to an existing comment. Use the `-cr` option to replace an existing comment. You can replace a comment only on writable objects.

-nc|-nocomment

Do not prompt for comments. If you do not enter a comment for any of the comment options, you are prompted for a comment. The comment is used for all 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.

Example

- Check in the current version of `main.c` with a state of *visible*.

```
ccm checkin -s visible main.c
```

- Check in three files (`clear.c`, `concat.c`, and `display.c`).

```
ccm ci -nc clear.c concat.c display.c
```

- Check in the directory `utils` without any new comments.

```
ccm ci -nc utils
```

- Check in the `c_includes` symbolic link to the *checkpoint* state (UNIX only).

```
ccm ci -c "let others edit" -state checkpoint c_includes
```

checkout command

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. The default state is *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 check out an object and use the `-t` option to specify a new version, 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.

- File specification

The specified path must be in a project's maintained work area.

- File specification

For example, `sub_proj\main.c@my_proj-1` (Windows) or `sub_proj/main.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. Specify valid context projects by using a selection set reference form (for example, "`@1`") or an object name form (for example, `main.c-1:csrc:1`).

The `checkout` command supports these subcommands:

- Checking out an object
- Checking out a project

Checking out an object

You can check out files to own them and make them writable. You can check out directories, but directories are checked out automatically when you add a file to or move a file from the directory.

About this task

```
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 to append a comment on baseline projects and their members when they are checked in to the *released* state. The `-comment` option 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 to start the default text editor to compose and edit the comment. 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 to use the contents of the specified file 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 Folder 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 this command:

```
ccm co main.c -to bar
```

The new object version is `main.c-bar`.

To change the name, you must include the object name and the version in the destination argument. For example, run this command:

```
ccm co main.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 built-in algorithm.

If you are checking out a version of an object that is used in your current project, the new version (the "to" version) is also 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. The directory is then checked in to the integrate state. You can disable this feature by setting shared_project_directory_checkin to FALSE in your initialization file.

-task *task_spec*

Specifies the task to associate the newly checked out objects with. 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.

Example

- Check out version patch1 from version 1 of `main.c` (version 3 of `main.c` is in the current directory).

```
ccm co -c "patch1: fix symbol table bug" -to patch1 main.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
```

Checking out a project

You can make a copy of a project, which sets up a work area. In the work are, you can change project members. This command is now called the copy_project command operation.

About this task

```
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_string*

Specifies to append a comment on all baseline projects and their members when they are checked in to the *released* state. The *comment_string* 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 to start the default text editor to compose and edit the comment. 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 to use the contents of the specified file 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 have 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*. The default is `-not_modifiable_wa`.

-no_u|-no_update

Specifies that the checked-out project is not updated when it is copied. The default is `-no_update`.

-ntl|-no_translate

Specifies that ASCII files in the work area are copied between Windows and UNIX without newline translation. The default is `-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. The default is `-maintain_wa`.

-nwat|-no_wa_time

Specifies that the files in the work area for the project use timestamps. The timestamps show the modification time rather than the time they were copied to the work area. The default is `no_wa_time`.

-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 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 release of the project. 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 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 is 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 built-in algorithm.

-tl|-translate|-translation

Specifies that ASCII files in the work area to copy between Windows and UNIX with newline translation. The default is `-translation`.

-u|-update

Specifies that the checked-out project is updated when it is copied. The specified project is checked out without a work area and is updated according to the setting for the project grouping that indicates whether to refresh the baseline and tasks. 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 built-in algorithm.

-wa|-maintain_wa

Specifies that the project has a maintained work area. The default is `-maintain_wa`.

-wat|-wa_time

Specifies that the files in the project work area use timestamps that show the time they were copied into the work area. The default is `-no_wa_time`.

Example

- 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 HP -purpose
"System Testing" -versions "1.0:sqa,hpos1_1.0:hpos1_sqa,hpos2_1.0:hpos2_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
```

checkpoint command

The `checkpoint` command saves a personal version of an object for your use only. Checkpointing an object preserves 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`. To checkpoint an object, it must be in the *working* state. 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.

The `checkpoint` command supports these subcommands:

- Checkpointing a project
- Checkpointing an object

Checkpointing a project

You can save a personal version of a project for your use only. Checkpointing a project preserves it in a state that is not modifiable, but that you can delete later when you no longer need it. You must own the project to perform a `checkpoint`. To checkpoint a project, it must be in the *working* state.

About this task

```
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 to append a comment 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 to start the default text editor to compose and edit the comment. 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 to use the contents of the specified file 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 replaces any existing comment.

-p|-project *project_spec*

Checkpoints a project. See Project specification for details.

-t|-to *version*

Sets the version of the newly checked-out object. You can also add the version to the object name.

Checkpointing an object

You can save a personal version of an object for your use. Checkpointing an object preserves 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`. To checkpoint an object, it must be in the *working* state.

About this task

```
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 to append a comment 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 to start the default text editor to compose and edit the comment. 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 is 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 replaces 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.

Example

- Checkpoint the current *working* version of `main.c`, and add a comment.

```
ccm ckpt -c "Phase 1 works." main.c

Adding 'release' attribute with value '2.0' to object main.c-3:csrc:11
Associated object main.c-3:csrc:11 with task 36
Checkpointed object version: 'main.c-2:csrc:11'
```

- Checkpoint the current working version of `main.c`. Add a comment and specify the new *working* object version to be *joe*.

```
ccm ckpt -c "Trying Joe's algorithm." -t joe main.c

Adding 'release' attribute with value '2.0' to object main.c-joe:csrc:11
Associated object main.c-joe:csrc:11 with task 36.
Checkpointed object version: 'main.c-3:csrc:11'
```

cmdhistory command

You can obtain a record of commands run during a session.

The following list show ways you might use this command:

- You can test executing a set of commands, and then see the command history to capture that sequence in a script.
- If something unexpected happens, you can see the sequence of commands leading to the last command that was executed. This might be useful when discussing issues with IBM® Rational® Software Support.

The `cmdhistory` command supports these subcommands:

- Clearing entries from history
- Showing command history
- Setting a maximum number of commands to record

Clearing entries from history

You can clear the command history of all commands that were executed in the current Rational® Synergy session.

About this task

```
ccm cmdhistory -clear
```

-clear

Clears the command history of all commands executed in the current session.

Showing command history

The `cmdhistory -set` command shows the last *x*commands executed in the current session, up to the maximum that you have specified.

About this task

```
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 have entered for the session is displayed. For example, you request a count of 10 and have set a maximum command history of 50. You have executed 200 commands in that session, but the count shows the last 10 commands. However, if you have 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
```

Setting a maximum number of commands to record

You can set the command history to record a maximum number of commands.

About this task

```
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
```

conflicts command

You can display the conflicts for a project with update properties that 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 project membership.

The `conflicts` command supports these subcommands:

- Showing object conflicts for a project
- Showing task conflicts for a project

Showing object conflicts for a project

You can show the object conflicts in a project.

See Resolving membership conflicts in a project or project grouping to learn about membership conflicts found in projects or project groupings and how to resolve the conflicts.

About this task

```
ccm conf|conflicts [-r|-recurse] [-v|-verbose][-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. The default is `-ch|-column_header`.

See -ch|-column_header for details.

-f|-format *format*

Specifies the command output format. See -f|-format for details.

The format string specified by the command is processed by a data formatter. If there are multiple kinds of objects to be formatted based on a CLI command, then the format string must specify the properties to be displayed. Use an `objectkey` keyword with the following form: `%[objectkey]propertyname`.

The `objectkey` specifies the part of the object that the named property refers to. The `propertyname` is an advanced keyword. The following list shows the supported object keys and property names for the conflicts command:

- `objectkey`: object

`propertyname`: Any attribute or built-in keyword for versioned objects.

A specified attribute or built-in keyword value for the versioned object for the conflict. For a task conflict, the attribute is a null value.

- `objectkey`: project

`propertyname`: Any attribute or built-in keyword for projects.

A specified attribute or built-in keyword value of the context project in which the conflict was detected.

- `propertyname`: category

The category of the conflict is an abbreviated name that is consistent with the values shown in the GUI. You do not need to specify an object key to use this property name. Use the property name in the format string as `%category`.

- `propertyname`: description

The full description of the type of the conflict is the name that, in prior releases, was shown as the only form of conflict description. You do not need to specify an object key to use this property name. Use the property name in the format string as `%description`.

-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_header for details.

-nf|-noformat

Specifies not to use column alignment. See -nf|-noformat for details.

-ns|-no_sort

Specifies not to sort the output. 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.

-v|-verbose

Specifies to display detailed messages showing the steps and analysis of the membership conflicts.

Example

Show the conflict detection information for the `etc-1` project.

```
ccm conflicts etc-1
```

```
Project: etc-1
Objectname              Task    Conflict                          Category
abc.java-1:ascii:di#1   di#207  Included by 'use' operation?      Extra changes
def.java-1:ascii:di#1   di#207  Included by 'use' operation?      Extra changes
etc-1:dir:di#1          di#207  Included by 'use' operation?      Extra changes
```

Showing task conflicts for a project

You can show the task conflicts in a project.

See Resolving membership conflicts in a project or project grouping to learn about membership conflicts found in projects or project groupings and how to resolve the conflicts.

About this task

```
ccm conf|conflicts -t|-tasks [-r|-recurse] [-v|-verbose] [-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. The default is `-ch|-column_header`.

See -ch|-column_header for details.

-f|-format *format*

Specifies the command output format. See -f|-format for details.

The format string specified by the command is processed by a data formatter. If there are multiple kinds of objects to be formatted based on a CLI command, then the format string must specify the properties to be displayed. Use an `objectkey` keyword with the following form: `%[objectkey]propertyname`.

The `objectkey` specifies the part of the object that the named property refers to. The `propertyname` is an advanced keyword. The following list shows the supported object keys and property names for the conflicts command:

- `objectkey`: task

`propertyname`: Any attribute or built-in keyword for tasks.

A specified attribute or built-in keyword value for the task or tasks associated with the conflict. For a task conflict, this is a single value from the task in conflict. For an object conflict, this is a collection of values, one from each of the tasks associated with the conflicting object.

- `propertyname`: category

The category of the conflict is an abbreviated name that is consistent with the values shown in the GUI. You do not need to specify an object key to use this property name. Use the property name in the format string as `%category` itself.

- `propertyname`: description

The full description of the type of the conflict is the name that, in prior releases, was shown as the only form of conflict description. You do not need to specify an object key to use this property name. Use the property name in the format string as `%description` itself.

-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_header for details.

-nf|-noformat

Specifies not to use column alignment. See -nf|-noformat for details.

-ns|-no_sort

Specifies not to sort the output. 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.

-v|-verbose

Specifies to display detailed messages showing the steps and analysis of the membership conflicts.

Example

Show the task conflict detection information for the `etc-1` project.

```
ccm conflicts -tasks etc-1
```

```
ccm conf|conflicts -t|-tasks [-r|-recurse] [-v|-verbose] [-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
```

copy_project command

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. 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.

The `copy_project` command supports the "Copying a project" subcommand.

Copying a project

You can create 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.

About this task

```
ccm copy_project|cp [-purpose purpose] [-platform platform]
    [-release (release_spec|as_is)]
    [-subprojects] [-scope ((all|all_subprojects) |
    (project_only|nosubprojs|nosubprojects) |
    (same_component|same_component_subprojects) |
    (same_release|same_release_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 to append a comment to 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 to invoke the default text editor to compose and edit the comment. 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 to use the contents of the specified file 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 have 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*. The default is `-nmod|-not_modifiable_wa`.

-no_u|-no_update

Specifies not to update the project after it is copied. The default is `-no_u|-no_update`.

-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. The default is `-wa|-maintain_wa`.

-nwat|-no_wa_time

Specifies that the files in the project work area use timestamps. The timestamps show the modification time rather than the time they were copied into the work area. The default is `-nwat|-no_wa_time`.

-platform *platform*

Specifies the platform to be used for the new checked out project. Set the *platform* to a valid platform name. 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 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 do not specify this option, and you are in the *build_mgr* or *ccm_admin* role, the default is `Integration Testing`.

-rel|-relative

Specifies to locate the work area on a path relative to the parent project 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.

Projects must have a release value because project groupings and their corresponding process rules must always be associated with a release. Manual update properties are not supported.

-scope (all|all_subprojects) | (project_only|nosubprojs|nosubprojects) | (same_component|same_component_subprojects) | (same_release|same_release_subprojects)

Specifies the scope for copying subprojects. You can copy only the subprojects for the same release, only the subprojects for the same component, all subprojects, or no subprojects.

-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 project_subdir_template. See Setting the work area path template for shared access for more information.

-subprojects

Specifies to copy all subprojects in the specified project hierarchy.

-tl|-translate|-translation

Indicates to translate ASCII files 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 to update the checked out project when it is copied. If specified, the project is checked out without a work area. The project is updated according to the project grouping setting, which indicates whether to refresh the baseline and tasks. 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. The default is `-wa|-maintain_wa`.

-wat|-wa_time

Specifies that the files in the project work area use timestamps. The timestamps show the time the files were copied into the work area, rather than their modification time. The default is `-nwat|-no_wa_time`. The default is `-nwat|-no_wa_time`.

Example

- 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
```

copy_to_file_system command

A project copied to your work area is always copy-based, never link-based, and the files are read-only. You can copy a project to the file system even if the project does not have a work area. File modification time is set to the time the copy is created.
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. Although you can open directories and files in the copied project, you cannot maintain and reconcile the project.

About this task

```
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, it is set 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 and the selected project (`ccm_wa\database_name` on Windows, or `ccm_wa/database_name` on UNIX in your home directory).

Note: This option creates 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
```

create command

You can create an object and add it to the current project. You can create a project or a project member (directory or file), both of which are created as part of a project, or a floating object. A floating object is a new project that is not created as a part of a project.

- When you create a 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 an 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 specified task. The directory is then 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.

- When you create a project, it is created as a floating object, but you can make it a subproject in an existing project by using `use -p`.
- When you create a project, a work area is created 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, the directory is checked out automatically. Check in the directory and the new object to make the new object available to other users.

The `create` command supports these subcommands:

- Creating a top-level project
- Creating a project using an existing directory as its root directory
- Creating an object

Creating a top-level project

You can create a new top-level project. When you create a project, a work area is created for it automatically. By default, the work area is formed by expanding the default work area path template.

The default setting is`%HOMEPATH%\My Documents\Synergy\ccm_wa\databaseName\projectName-projectVersion` on Windows and `$HOME/ccm_wa/databaseName/projectName-projectVersion` on UNIX.

To make a project a member in an existing project, use the `ccm use -p` command after you have created the project.

About this task

```
ccm create -t|-type project [-platf|-platform platform]
        [-purp|-purpose purpose] [-release release_spec]
        [-set|-path|-setpath absolute_path] [-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])
        [-c|-comment comment_string] [-ce|-commentedit]
        [-cf|-commentfile file_path] [-task task_spec] new_project_spec...
```

-c|-comment *comment*

Specifies to append a comment 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 to invoke the default text editor to compose and edit the comment. 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 to use the contents of the specified file 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 to use the contents of the specified file 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 have permissions set so they are modifiable even if they are not checked out. The default is `-nmod|-not_modifiable_wa`.

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 name forms:

- A name, colon, and version
- A File contents forms that contains a name, colon, or version

The *`new_project_spec`* option is not a general project specification. You cannot use forms such as an object name form or query selection set reference form.

-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*. The default is `-nmod|-not_modifiable_wa`.

-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.

-nwat|-no_wa_time

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. The default is `-nwat|-no_wa_time`.

-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. Set the purpose to the name of a defined purpose that is valid for the specified release. Use the `project_purpose -show` command to list valid purposes.

-rel|-relative

Specifies to locate the work area on a path relative to the parent project path.

-release *release_spec*

Specifies the release 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 for the project. Set the *absolute_path* to an absolute path that you can see and modify.

-task *task_spec*

Specifies the task to associate with the new project root directory. You can set the *task_spec* to a single task. By default, the project root directory is associated with the current task. See Task specification for details.

-tl|-translate|-translation

Specifies to perform newline translation of ASCII files when the files are copied between a Windows client and UNIX server, or between a UNIX client and a Windows server.

-wa|-maintain_wa

Specifies that the new project has a maintained work area. The default is `-wa|-maintain_wa` if a work area option is not specified. The work area is updated with changes made to the new project. Use the `work area` command to turn off work area maintenance.

-wat|-wa_time

Specifies that the files in the project work area use timestamps. The timestamps show the time the files were copied into the work area, rather than their modification time. The default is `-nwat|-no_wa_time`.

Example

- 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
```

Creating a project using an existing directory as its root directory

You can create a project by reusing a specified directory as its root directory. The project name is the same as the name of the specified root directory. This command is useful for dividing a large directory hierarchy within a single project into one or more subprojects.

About this task

```
ccm create -r|-root -t|-type project [-v|-version version]
        [-platf|-platform platform] [-purp|-purpose purpose]
        [-release release_spec]
        ([-cb|-copy_based] | [-lb|-link_based|-ncb|-not_copy_based])
        ([-rel|-relative] | [-nrel|-not_relative])
        ([-mod|-modifiable] | [-nmod|-not_modifiable])
        [-set|-path|-setpath absolute_path]
        [-wat|-wa_time] [-nwat|-no_wa_time]
        ([-tl|-translate|-translation] | [-ntl|-no_translate|-no_translation])
        [-c|-comment comment_string] [-ce|-commentedit]
        [-cf|-commentfile file_path] dir_spec
```

-c|-comment *comment*

Specifies to append a comment 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 to start the default text editor to compose and edit the comment. 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 to use the contents of the specified file 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 to use the contents of the specified file 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 have permissions set so they are modifiable even if they are not checked out. The default is `-nmod|-not_modifiable_wa`.

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 name forms:

- A name, colon, and version
- A File contents forms that contains a name, colon, or version

The *`new_project_spec`* option is not a general project specification. You cannot use forms such as an object name form or query selection set reference form.

-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*. The default is `-nmod|-not_modifiable_wa`.

-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.

-nwat|-no_wa_time

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. The default is `-nwat|-no_wa_time`.

-platf|-platform *platform*

Specifies the platform for the new project. The platform must be a valid platform name. 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 not to set a platform.

-purpose *purpose*

Specifies the purpose for the new project. Set the purpose to the name of a defined purpose that is valid for the specified release. Use the `project_purpose -show` command to list valid purposes.

If this option is not specified, and you are in the *developer* role, the default is Insulated Development. If you do not specify this option, and you are in the *build_mgr* or *ccm_admin* role, the default is Integration Testing.

-rel|-relative

Specifies to locate the work area on a path relative to the parent project path.

-release *release_spec*

Specifies the release for the new project. You can set the *`release_spec`* to a single release that is defined and active. See Release specification for details.

Projects must have a release value because project groupings and corresponding process rules must be associated with a release. Manual update properties are not supported.

-set|-path|-setpath *absolute_path*

Specifies the work area path for the copied project. Set the *`absolute_path`* to an absolute path that you can see and modify. If not specified, the default work area path uses the current *`wa_path_template`* and *`project_subdir_template`*.

-tl|-translate|-translation

Specifies to perform newline translation of ASCII files when the files are copied between a Windows client and UNIX server, or between a UNIX client and a Windows server.

-v|-version *version*

Specifies the version to use for the new project.

-wa|-maintain_wa

Specifies that the new project has a maintained work area. The default is `-wa|-maintain_wa`.

-wat|-wa_time

Specifies that the files in the project work area use timestamps. The timestamps show the time the files were copied into the work area, rather than their modification time. The default is `-nwat|-no_wa_time`.

Example

- 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 -purp "Integration Testing" -wa
ccm create -t project SubPrj-1 -release 1.0 -purp "Integration Testing" -wa
cd WAPATH\MainPrj-1\MainPrj (Windows) OR cd WAPATH/MainPrj-1/MainPrj (Unix)
ccm use -p SubPrj-1
```

- 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 -purp "Integration Testing" -nwa
ccm create -t project SubPrj-1 -release 1.0 -purp "Integration Testing" -nwa
ccm use -p SubPrj-1 -dir MainPrj@MainPrj-1
```

Creating an object

You can create an object and add it to the project associated with the specified object. If you use a work area reference form, the context project is associated with the specified work area path. If you use a project reference specification form, the context project is 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 an object in a non-writable directory, a new directory version is checked out automatically. 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. The directory is automatically associated with the specified task and checked in to the *integrate* state. You can disable automatic check-in by setting shared_project_directory_checkin to FALSE in your initialization file.

About this task

```
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 to append a comment to 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 to invoke the default text editor to compose and edit the comment. 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 to use the contents of the specified file 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 these 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. Optionally, the relative path can end with the version delimiter and version or colon and version located under a directory in the specified project.
  - A File contents forms that contains either a work area reference form or a project reference form.

The forms provide a context project and a context parent directory. The object is created under the parent directory in the specified context project. The *new_file_spec* is not a general *file_spec*. You cannot use forms such as an object reference 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 option 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.

Example

- On Windows, create a 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 C source object called `sort.c` in the `utils/sym_tool` directory.

```
ccm create -type csrc utils/sym_tool/sort.c
```

- Create a directory object called `testcase` under the current directory.

```
ccm create -t dir testcase
```

delete command

You can delete an object version if it is not a member of a project. You can also delete an object version if it is only a member of the current project and has no successors.

When you delete an object from 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 non-writable, the directory is checked out and associated automatically with the current task. The directory is also 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.

The delete operation is permanent.

The `delete` command supports the "Deleting objects from the database" subcommand.

Deleting objects from the database

You can delete a specific version of a file, directory, or project from a directory and from the database. Additionally, you can delete a project hierarchy.

About this task

```
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 setting 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 is recursive for 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.

Example

- Delete the `sort.c` file and replace it with the previous version (output might differ from the following examples).

```
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 hierarchically and recursively.

```
ccm delete -p Project_Top-int:project:W#1 -recurse -h
```

delimiter command

The `delimiter` command shows the value of the delimiter character. The default is the dash character. Set the delimiter to any nonrestricted character. When you set the delimiter, you set it for a database.

The main reason for changing a delimiter is to avoid using a character that is used in object names in the database. The CM administrator ensures that the default delimiter does not conflict with objects that contain the dash character as a part of their name. If a conflict exists, change the default delimiter **before** users begin to work in the database.

Note: Change the delimiter for a database before migrating software under Rational® Synergy control. You can change the delimiter at any time. Changing the delimiter when projects are in a database forces you to change the work area paths of the projects that include the delimiter character.
After you change the delimiter and restart the interface, any new project that you create uses the new delimiter that you set. Work area paths for existing projects are not affected.

CAUTION:

Changing your delimiter might affect your work areas. See work_area command for more information.

Naming restrictions

Some characters are forbidden for use as the delimiter by the `ccm delimiter` command. These characters are listed in Naming and formatting.

You can control whether the delimiter is a restricted character. See allow_delimiter_in_name for information about changing restrictions for non-project object names. The delimiter is still restricted for versions, types, instances, and projects.

The `delimiter` command supports these subcommands:

- Setting the version delimiter
- Showing the current version delimiter

Setting the 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 creating the initial work area path for a project.

Before you begin

You must be in the *ccm_admin* role to set the delimiter.

About this task

```
ccm delim|delimiter version_delimiter
```

version_delimiter

Specifies the new version delimiter to be used.

Example

- The display name, object name, and work area path use the delimiter to separate the name from the version. For example, your project work area might be in the following location.

Windows: `c:\users\sue\ccm_wa\ccmint22\hello-sue`

UNIX: `linda/ccm_wa/ccmint22/hello-sue`

`ccm delimiter`

Change the delimiter between the file name and version to a comma by using the `ccm delim` command.

`ccm delim ","`

When you create a project (for example, `goodbye,sue`), your project work area is in the following location.

Windows: `c:\users\sue\ccm_wa\ccmint22\goodbye,sue`

UNIX: `~linda/ccm_wa/ccmint22/goodbye,sue`

- Suppose you want to reference the work area version of a `csrc` file, `poly.c,2`, even though you have version 3 (`poly.c,3`) in your work area. If your delimiter is set to a comma, you might specify the file as follows.

```
ccm properties poly.c,2
ccm properties poly.c:2
```

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.

About this task

```
ccm delim|delimiter
```

Example

Show the current version delimiter.

```
ccm delimiter
```

diff command

The `diff` command shows the differences between files, directories, or projects. Use this command to do two types of comparisons: a source compare (the default) and a version compare.

Source compare shows the differences between source files, directories, or projects. If you perform the `diff` command on directories, a comparison of the lists of non-versioned members is done. For projects, the lists of versioned member files are compared.

Version compare compares other attributes of the files, directories, and projects that are not considered to be the source; for example, the `create_time`, `modify_time`, `name`, and `version`.

The `diff` command supports the following subcommands:

- Comparing objects
- Comparing projects

Comparing objects

Each type of object for which you can compare source has default compare tools predefined by Rational® Synergy for both the CLI and the GUI. The default compare tool for the GUI is interactive, and the one for the CLI is automatic. The compare tool shows the differences between different versions of files or directories.

The default compare tool is specified in the `ccm.properties` file for UNIX and Windows. You can specify a different compare tool for each object type. The specified compare tool applies to the specified type and any subtype without a specified compare tool. By default, only the ASCII file type has a compare tool defined.

For example, If you define a "japanese" file type, and you want a comparison of Japanese files to be done in a different encoding than ASCII files, you can specify it as follows.

```
ccm.cli.tools.compare.japanese.windows="%ccm_home\\\\bin\\\\util\\\\cc_
dff.bat" "%ccm_home" %{encoding[null='SJIS']} %outfile %file1 %file2
```

By default, a compare tool is not set for the binary type or its subtypes, but you can set a compare tool manually.

Additionally, you can set encoding rules in the `ccm.properties` file so that compared files display the correct language, as follows:

```
// Command to compare source objects on UNIX and its checkstatus.
ccm.cli.tools.compare.ascii.unix=%ccm_home/bin/util/cc_dff %ccm_home %{
encoding[null='CP1252']} %outfile %file1 %file2^M
```

```
// Command to compare source objects on Windows and its checkstatus.
ccm.cli.tools.compare.ascii.windows="%ccm_home\\\\bin\\\\util\\\\cc_dff
.bat" "%ccm_home" %{encoding[null='CP1252']} %outfile %file1 %file2
```

The second parameter to the `ccm_diff` command allows you to specify an encoding for the file being compared. The syntax "`%{encoding[null='CP1252']}`" is interpreted as follows:

If the object type specifies a work area encoding with the `encoding_rules` attribute, use that encoding for the compare. If not, use the CP1252 encoding. (See File encodings for a discussion of the `encoding_rules` attribute.)

You can specify a default encoding other than CP1252 for this parameter. For example, the following syntax indicates to use the UTF8 encoding if an encoding is not specified on the object type:

```
ccm.cli.tools.compare.ascii.windows="%ccm_home\\\\bin\\\\util\\\\cc_dff
.bat" "%ccm_home" %{encoding[null='UTF8']} %outfile %file1 %file2
```

The following syntax indicates to always use the CP1252 encoding:

```
ccm.cli.tools.compare.ascii.windows="%ccm_home\\\\bin\\\\util\\\\cc_dff
.bat" "%ccm_home" CP1252 %outfile %file1 %file2
```

The valid encodings are CP1252, UTF8, BIG5, eucJP, EUC-KR, SJIS, and GB18030.

The files being compared and the ancestor file must have the same encoding.

For example, in a Chinese-language database, you might set the following encoding rules for the `ascii` type:

```
Server-encoding: GB18030
Unix-wa-encoding: GB18030
Windows-wa-encoding: GB18030
```

Assuming the default encoding parameters for the `ccm_diff` command were used, the CLI and GUI compare tools are then invoked with the GB18030 encoding.

Alternatively, if your site contains only Chinese-language databases, you can change the default compare command as follows, and then `encoding_rules` attributes are not required on the `ascii` type.

```
ccm.cli.tools.compare.ascii.windows="%ccm_home\\\\bin\\\\util\\\\cc_dff
.bat" "%ccm_home" %{encoding[null='GB18030']} %outfile %file1 %file2
```

About this task

```
ccm diff  [-vc|-versioncompare] object_spec1 [object_spec2]
```

object_spec1

object_spec1 specifies the first file or directory to be compared. You can set *object_spec1* to be a Object specification for one file or directory.

object_spec2

object_spec2 specifies the second file or directory to be compared. You can set *object_spec2* to be a Object specification for one file or directory. If *object_spec1* is a file, then *object_spec2* must also be a file. If *object_spec1* is a directory, then *object_spec2* must also be a directory.

Comparing projects

Default compare tools are predefined by Rational® Synergy for both the CLI and the GUI. The default compare tool for the GUI is interactive, and the one for the CLI is automatic. The compare tool shows the differences between different versions of projects.

The default compare tool is specified in the `ccm.properties` file as follows, for UNIX and Windows, respectively.

Additionally, you can set encoding rules in the `ccm.properties` file so that compared projects display in the correct encoding, as follows:

```
// Command to compare projects on UNIX and its checkstatus.
ccm.cli.tools.compare.project.unix=%ccm_home/bin/util/cc_dff %ccm_home
%{encoding[null='CP1252']} %outfile %project1 %project2^M

// Command to compare projects on Windows and its checkstatus.
ccm.cli.tools.compare.project.windows="%ccm_home\\\\bin\\\\util\\\\cc_d
ff.bat"
"%ccm_home" %{encoding[null='CP1252']} %outfile %project1 %project2
```

The second parameter to the `ccm_diff` command allows you to specify an encoding for the project being compared. The syntax "`%{encoding[null='CP1252']}`" is interpreted as follows:

If the project specifies a work area encoding with the `encoding_rules` attribute, use that encoding for the compare. If not, use the CP1252 encoding. (See File encodings for a discussion of the `encoding_rules` attribute.)

You can specify a default encoding other than CP1252 for this parameter. For example, the following syntax indicates to use the UTF8 encoding if an encoding is not specified on the project:

```
ccm.cli.tools.compare.project.windows="%ccm_home\\\\bin\\\\util\\\\cc_d
ff.bat"
"%ccm_home" %{encoding[null='UTF8']} %outfile %project1 %project2
```

The following syntax indicates to always use the CP1252 encoding:

```
ccm.cli.tools.compare.project.windows="%ccm_home\\\\bin\\\\util\\\\cc_d
ff.bat" "%ccm_home" CP1252 %outfile %project1 %project2
```

The valid encodings are CP1252, UTF8, BIG5, eucJP, EUC-KR, SJIS, and GB18030.

The projects being compared and the ancestor project must have the same encoding.

For example, in a Chinese-language database, you might set the following encoding rules for the `project` type:

```
Server-encoding: GB18030
Unix-wa-encoding: GB18030
Windows-wa-encoding: GB18030
```

Assuming the default encoding parameters for the `ccm_diff` command were used, the CLI and GUI compare tools are then invoked with the GB18030 encoding.

Alternatively, if your site contains only Chinese-language databases, you can change the default compare command as follows, and then `encoding_rules` attributes are not required on the `project` type.

```
ccm.cli.tools.compare.project.windows="%ccm_home\\\\bin\\\\util\\\\cc_d
ff.bat"
"%ccm_home" %{encoding[null='GB18030']} %outfile %project1 %project2
```

About this task

```
ccm diff  [-vc|-versioncompare -p|-project
    project_spec1 [project_spec2]
```

project_spec1

> ***project_spec1*** specifies the first project to be compared. You can set ***project_spec1*** to a Project specification for one project.

project_spec2

> ***project_spec2*** specifies the second project to be compared. You can set ***project_spec2*** to a Project specification for one project.

dir command

You can display two categories of files: objects under Rational Synergy control and files that exist in the file system only. By default, the command shows only controlled objects. Use the `-m` option to display uncontrolled objects as well as controlled objects.

The `dir` command supports the "Listing files" subcommand.

Listing files

You can list 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.

A new pseudo-property named `relative_path` is available for all controlled objects listed by a `ccm dir` command. This property is the relative path within the context project for that object, using a directory field separator of / on all platforms. If the object is not a member of the specified context project, by default the property is shown as an empty string.

About this task

```
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_header 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.

- 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. The situation also happens 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_header for details.

-nf|-noformat

Specifies not to use column alignment. See -nf|-noformat for details.

-ns|-nosort|-no_sort

Specifies not to sort the command 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 path 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 output. See -u|-unnumbered for details.

-w

Specifies to use the default short-form. This setting shows the display name of each object.

Example

- List the files that are not controlled

```
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
```

edit command

When you edit a file in a project, the editor is started in the work area if the work area is visible to the client. If the specified file is not in a project or if the work area for the project is not visible, the editor is started 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. You must reconcile your work area regularly after you edit and save a file.

The `edit` command supports the "Editing a file" subcommand.

Editing a file

You can edit the specified file by using the default editor.

About this task

```
ccm edit file_spec...
```

file_spec

Specifies the file to edit. See File specification for details.

Attention: 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.

Example

Edit version 8 of the `log.c` file. To edit an object, it must be writable by you.

```
ccm edit log.c-8
```

export command

The `ccm export` command exports specified objects to a package that might be later imported into the same or another Rational® Synergy database at the same release and patch level. The format of the packages that is produced by the `ccm export` command is implementation private and only intended for use with the `ccm import` command.

Export and import restrictions

Using the `ccm export` command to export data from one database and the `ccm import` command to import that data into another database is subject to the following restrictions.

CAUTION:
Any use outside of these restrictions is not supported. If you choose to work in an unsupported mode, loss of data or data inconsistencies in the database into which the data is imported might occur.

- The exporting database and importing database must be at the same Rational Synergy release, iFix, and patch level.
- If the exporting database is DCM initialized, the importing database must also be DCM initialized.
- If the exporting database is not DCM initialized, the importing database must also not be DCM initialized.
- The importing database must have type definitions for each object that is to be imported into the database.
- If the user is in the `ccm_admin` role, the import might overwrite and update existing objects, including objects in static states. The use of `ccm export` and `ccm import` is not intended as a replacement for DCM. If you are trying to implement replication of data between Rational Synergy databases, use DCM.
- The use of import or export to copy `model`, `cvtype`, `attype`, `admin`, or other similar types of objects that make up the Rational Synergy model is not supported. Incorrect usage in the `ccm_admin` role can make the importing database unusable.

The `export` command supports the "Exporting objects" subcommand.

Exporting objects

Use the `ccm export` command to export objects.

About this task

```
ccm export      -p|-project [-serverdir server_path] -to compressedfile
                [-r|-recurse] project_spec...
ccm export      [-serverdir server_path] -to compressedfile object_spec...
```

object_spec

Specifies the object to be exported. There is no expansion of the object to include other objects. See Object specification.

project_spec

Specifies the project to be exported. If the -r|-recurse option is not specified, only the project object is exported.

-r|-recurse

Specifies that for each project, the recursive members of the project are also exported.

-serverdir *server_path*

Specifies that an alternate temporary server file system location is to be used to prepare the exported data and compressed file. By default, the command uses the dbpath/export directory. To use this option, you must be in the ccm_admin role. This option is helpful when there is insufficient space on the file system hosting the Rational Synergy database.

-to compressedfile

Specifies the import package compressed file that is to be created. The *compressedfile* must be a valid client file system path.

finduse command

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 these kinds 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 File specification.

- 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.

The finduse command supports these 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
- Scopes

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.

About this task

```
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 which baseline to use. See Baseline specification for details.

Example

- Find project grouping that uses the baseline A#Base_One.

`ccm finduse` -baseline `A#Base_One`

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.

About this task

```
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 which change request uses to show. 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.

About this task

```
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
```

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.

About this task

```
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.

About this task

```
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
```

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.

About this task

```
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
```

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.

About this task

```
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.

Example

- 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
```

Finding where objects found by a query are used

This subcommand queries for objects matching a query expression or criteria. For each, the query then finds which projects, project groupings, process rules, folders, or baselines use them.

You can modify the query expression in the following ways.

- Use a query expression.
- Use 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.

About this task

```
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')""
```

Scopes

The scopes available for use with the `ccm finduse` command specify the scope of the search.

The following scopes are available.

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 to show all uses in development projects.

- `-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 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

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.

The `folder` command supports these 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`.

About this task

```
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_header 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 the folder comparison to show the tasks that are in both folders.

-nch|-nocolumn_header

Specifies not to use a column header in the output format. See -nch|-nocolumn_header for details.

-nf|-noformat

Specifies not to use column alignment. See -nf|-noformat for details.

-ns|-no_sort

Specifies not to sort the command output. See -ns|-nosort for details.

-not|-not_in

Specifies the folder comparison to 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 output. See -u|-unnumbered for details.

Example

- 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
```

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.

About this task

```
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 `-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 an existing folder specified by *folder_spec*. The folder specification for the existing folder must identify one 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

Example

- 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
```

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')`.

About this task

```
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 `-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 do not 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 do not specify any of these options, the default is to add tasks to the folder manually.

If you specify `-mode use_query` or `-query`, but do not specify `-custom`, `-dbid`, `-platform`, `-release`, `-subsystem`, or `-task_scope`, a default task query is used in this way:

- If you have defined a default task query, it is used.
- If you have not 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 to include a query for the specified platform.

-purpose *purpose*

Specifies to include a query for the specified purpose. See the `project_purpose` command Description and uses for a detailed description of purposes.

This option typically applies to queries for component tasks that are specified with one of these scopes: `component_task_projects`, `component_task_products`, or `component_task_projects_products`.

-qu|-query

Makes the new folder query-based, which 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 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 a task query that includes a query expression for task subsystem.

-ts|-scope|-task_scope

Specifies to use a task query. The task query includes 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 includes tasks for which you are the task resolver and that are modifiable 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.

Example

- 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 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.
```

Deleting a folder

This subcommand deletes the specified folders. The folders must be modifiable by you.

About this task

```
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.
```

Finding where a folder is in use

This subcommand finds where the specified folder is in use in the current database.

About this task

```
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_header for details.

folder_spec

Specifies the folders to search during the find operation. 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 -nch|-nocolumn_header for details.

-nf|-noformat

Specifies not to use column alignment. See -nf|-noformat for details.

-ns|-no_sort

Specifies not to sort the command output. 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 output (that is, the output is not numbered). See -u|-unnumbered for details.

Example

- Find the projects that use folders `123` and `234`.

```
ccm folder -finduse 123 234
```

Listing folders

About this task

This subcommand lists the folders that satisfy the specified criteria. If you do not specify options or arguments, the command lists all folders; otherwise, the command lists:

- 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_header 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_header for details.

-nf|-noformat

Specifies not to use column alignment. See -nf|-noformat for details.

-ns|-no_sort

Specifies not to sort the command output. 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 output. See -u|-unnumbered for details.

Example

- 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
```

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.

About this task

```
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)...]
          [-related] [-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 `-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 have not 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 for a detailed description of purposes.

This option typically applies to queries for component tasks that are specified with one of these scopes: `component_task_projects`, `component_task_products`, or `component_task_projects_products`.

-related

Use this option with the `-at|-add_task|-add_tasks` option or the `-rt|-remove_task|-remove_tasks` option only. When used with the `-at|-add_task|-add_tasks` option, the related tasks of the specified tasks are added. When used with the `-rt|-remove_task|-remove_tasks` option, the related tasks of the specified tasks are removed.

-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 includes 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.

Example

- Add tasks **5-9** to folder **95**.

```
ccm folder -modify -at 5-9 95
```

- Remove tasks **5-9** from folder **95**.

```
ccm folder -modify -rt 5-9 95
```

- 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
```

- 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
```

Showing a folder property

This subcommand shows the specified folder property.

About this task

```
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.

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.

About this task

```
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_header 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 -nch|-nocolumn_header for details.

-nf|-noformat

Specifies not to use column alignment. See -nf|-noformat for details.

-ns|-no_sort

Specifies not to sort the command output. 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 output. See -u|-unnumbered for details.

-v|-verbose

Specifies to use the verbose format for folder information.

Example

- 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
```

Showing folder information

This subcommand shows information about the specified folders.

About this task

```
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_header 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 -nch|-nocolumn_header for details.

-sep|-separator *separator*

Specifies a different separator character. See -sep|-separator for details.

-v|-verbose

Specifies that you want additional folder information.

Example

- Show information for folder **2**.

```
ccm folder -show info 2
```

folder_template command

Folder templates provide a pattern used to create folders. Folders created from a folder template are controlled by that folder template. Therefore, when you change the folder template, the folders controlled by it are updated.

You can give a folder template any description, using any character, except the pipe (|) character. The folder template description can include these 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 have the same description.

Keyword %release

For example, you create a folder template with a description of **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, 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 is added to the update properties for the project with 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, 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, a project owned by *jsmith* with a release of **3.1** is in a DCM database called `Bristol`. The project 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.

The `folder_template` command supports these 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.

When you use an option more than once, the query expression relating to each usage is combined with an "or". For example, specify `-release 1.0 -release 2.0` for a query expression of `(release='1.0' or release='2.0)`.

Contributions from different options are combined with "and". For example, specify `-release 1.0 -platform windows` for a query expression of `(release='1.0') and (platform='windows')`.

Before you begin

You must be in the *build_mgr* or *ccm_admin* role to use this subcommand.

About this task

```
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 command.

-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, the 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 option 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 Showing a project purpose for a detailed description of purposes.

This option typically applies to queries for component tasks that are specified with one of these 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 includes 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 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.

Example

- 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"
```

- Define a default query for a folder template to use to populate its folders with tasks.
  - Set the scope.
  - Set the release.

For parallel development and folder template management reasons, set this attribute.

- Set the subsystem, if necessary.
- Set the platform, if necessary.

If a folder applies to multiple platforms, you do not need to set the platform value.

- Set the database, if it is initialized to use DCM. For example, create a folder template. Folders created from this template collect all completed tasks for the current release, and makes them writable by build managers.

```
ccm folder_template -create -desc "All Completed Tasks for Release %release"
-task_scope all_completed -release "%" -writable build_manager
```

Deleting folder templates

This subcommand deletes a folder template. System predefined folder templates cannot be delete by a build manager.

Before you begin

You must be in the *build_mgr* or *ccm_admin* role to delete folder templates.

About this task

```
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.

Finding where a folder template is in use

This subcommand finds the process rule that uses the specified folder template.

About this task

```
ccm ft|folder_temp|folder_template -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_header for details.

folder_template_spec

Specifies the folder templates to search during the find operation. 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 -nch|-nocolumn_header for details.

-nf|-noformat

Specifies not to use column alignment. See -nf|-noformat for details.

-ns|-no_sort

Specifies not to sort the command output. 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 output. See -u|-unnumbered for details.

Example

- Find the process rules that use the folder template `All completed tasks for release %release`.

```
ccm folder -finduse "All completed tasks for release %release"
```

Listing folder templates

Use this subcommand to list folder templates.

About this task

```
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_header 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_header for details.

-nf|-noformat

Specifies not to use column alignment. See -nf|-noformat for details.

-ns|-no_sort

Specifies not to sort the command output. 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 output. See -u|-unnumbered for details.

Example

- View all personal folder templates.

```
ccm folder_template -list -template all_personal
```

- View all folder templates.

```
ccm folder_template -list
```

Modifying a folder template

Use this subcommand to modify a folder template. 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, specify `-release 1.0 -release 2.0` for a query expression of `(release='1.0' or release='2.0)`.

Contributions from different options are combined with "and". For example, specify `-release 1.0 -platform windows` for a query expression of `(release='1.0') and (platform='windows')`.

Before you begin

You must be in the *build_mgr* or *ccm_admin* role to modify a folder template.

About this task

```
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 Showing folder template information.

-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 these 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 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 Showing a project purpose for a detailed description of purposes.

This option typically applies to queries for component tasks that are specified with one of these 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 includes `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 includes `task_subsys='subsystem'`.

-ts|-scope|-task_scope

Specifies to use a task query. The task query includes 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 are updated to reflect the new permissions.

Setting controlling database for a folder template

This subcommand takes `local` control, `handover` control, or `accept` control for a folder template.

About this task

```
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*.

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. When you hand over control to a spoke through 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 that permits a generate operation, or a blank string. A blank string means that control cannot be handed over to that database.

-local

Specifies the folder template to take over local control. The object is no longer updated by DCM replication from another database.

Example

- Handover 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"
```

Showing a folder template property

Use this subcommand to view a specific property of a folder template.

About this task

```
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.

Showing folder template information

Use this subcommand to view information about a folder template.

About this task

```
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_header for details.

folder_template_spec

Specifies the folder template 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 -nch|-nocolumn_header for details.

-nf|-noformat

Specifies not to use column alignment. See -nf|-noformat for details.

-ns|-no_sort

Specifies not to sort the command output. 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 output (that is, the output is not numbered).

See -u|-unnumbered for details.

groups command

You can implement and define security for objects. A database can contain many different collections of objects. Group security restricts check out and modifies permissions to a specified group of users for files, directories, projects, tasks, and folders. In addition, you can specify read security, which limits visibility of the source contents of objects to designated groups.

Access rights for a file are not inherited from a parent directory or project. Such security is not secure in Rational® Synergy because users can add objects to their working projects. Unlike most operating system file systems, a Rational Synergy file can be used by any number of directories or projects. Therefore, access rights to collections of files must be established on each file.

If you work as the group manager, use group security for the following reasons.

- 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 can restrict access of an object to specified groups. The user can restrict access if the 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 that can be versioned, and does not apply to directories and projects. However, if you apply read security to an object that is currently in user work areas, those files are still readable by the users.

You can define the following different levels of read access security:

- 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 allows access to the source if the user is a member of at least one of those groups. All other users are denied access to the source contents of that object.
- An object with the highest level of security (no access to the source) cannot' 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. Read security can be used with copy-based work areas only.

The following examples illustrate how security is applied and used for an object.

- When no groups exist, or no groups are assigned to an object, there are no restrictions. Everyone can view, check out, and modify source files.
- 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 view the source objects only. In other words, check out and modify security is implemented, but read security does not yet exist.
- When one or more groups are created, and one group is given read security access, all other groups do not have read access to the files. After you start using the read security option, access to source contents is denied by default.

If you have a directory in the *public* state that uses group security and the user is not a member of any of the groups for the directory, the user can still create objects, add them to, or remove them from the directory. Users can easily overwrite changes if you use public directories. Exercise caution when using *public* directories.

See Database read security for additional information about setting up databases for read security. If you have a directory in the *public* state that uses group security and the user is not a member of any of the groups for the directory, the user can create objects, add them to, or remove them from the directory. Users can easily overwrite changes if they use *public* directories; exercise caution when using *public* directories.

The `groups` command supports these subcommands:

- Assigning groups to objects
- Creating a user group
- Listing user groups
- Modifying a user group
- Unassigning groups from objects

Assigning groups to objects

You can assign one or more user-group based restrictions to an object, which adds the specified entries to the `groups` attribute for the object.
For a file, directory, or project, one of the following must be true.

- The object must be the first version, and the user must have write access to the object.

- The user must be in the *group_mgr* role and be a member of the current groups for the object.
- 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 current groups for the object.
- The user must be in the *ccm_admin* role.

About this task

```
ccm group|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 this 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
```

Creating a user group

You can create a named user group.

About this task

```
ccm group|groups -c|-create (-v|-value
        [(-user|-users username_list)...] group_name
```

group_name...

Specifies the name of the user group to create.

-user|-users *username_list*

Specifies the names of users to be members of the new group. The *username_list* contains one or more user names separated by commas or spaces.

Listing user groups

Use this command to list defined user groups.

About this task

```
ccm group|groups -l|-list
```

Modifying a user group

Use this command to modify a specified user group.

About this task

```
ccm group|groups -m|-modify
         (-user|-users username_list)... group_name
ccm group|groups -m|-modify
         ([(-add|-add_user username_list)... group_name
         [(-remove|-remove_user username_list)... group_name
```

-add|-add_user *username_list*...

Specifies the names of users to be added as members of the group. The *username_list* contains one or more user names separated by commas or spaces.

group_name

Specifies the name of the user group to modify.

-remove|-remove_user *username_list*

Specifies the names of users that to be removed as members of the group. The *username_list* contains one or more user names separated by commas or spaces.

-user|-users *username_list*

Specifies the names of users to be members of the group. The *username_list* contains one or more user names separated by commas or spaces.

Unassigning groups from objects

You can remove assignment of one or more group restrictions from an object, which removes the specified entries to the `groups` attribute for the object.

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 current groups for the object.
- 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 current groups for the object.
- The user must be in the *ccm_admin* role.

About this task

```
ccm group|groups -unassign (-v|-value group_item_list)... object_spec...
```

-c|-create *group_name*

Creates a group name. Uses the default text editor to define the group membership.

-e|-edit *group_name*

Edits the user membership of an existing named group.

-l|-list

Lists all the defined groups.

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 this 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

- Create a group named `docs`.

```
ccm groups -c docs
```

- List all defined groups.

```
ccm groups -l
```

history command

You can show the version history of a project, directory, file, or release.

The `history` command supports the "Showing history for an object" subcommand.

Showing history for an object

You can show the version history for an object. For example, perhaps a developer has checked out a version of a file that you are about to work on. You might want to discuss your changes with the other developer to find out if the parallel versions must be merged.

About this task

```
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_header 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_header for details.

-nf|-noformat

Specifies not to use column alignment. See -nf|-noformat for details.

-ns|-nosort|-no_sort

Specifies not to sort the command output. 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 to list. 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:
************************************************************
```

import command

The `ccm import` command imports objects contained in an import package compressed file. The main use of this command is to import the data from an import package created by the `ccm migrate` command.

For this usage, the user must be in a role that is consistent with the data in the import package. Specifically, if none of the migrate objects have been previously imported, and can be created and transitioned to their wanted states as a developer, then a user in the `developer` or `build_mgr` role can use the command. If any of the objects previously exist and are in a *static* state, you must be in the `ccm_admin` role to update those objects.

You can also use this command to import objects that were previously exported using the `ccm export` command. This method is primarily intended for use by administrators and is subject to the export and import restrictions.

Note: The `ccm import` command is governed by the same security rules as other types of actions that create or modify data in a Rational® Synergy database. Also, if you use the `ccm_admin` role, you have the privilege to update static objects. Ensure that you abide by the export and import restrictions.

The `import` command supports the "Importing objects" subcommand.

Importing objects

Use the `ccm import` command to import objects contained in an import package compressed file. The main use of this command is to import the data from an import package that the `ccm migrate` command created.

About this task

```
ccm import      [-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]
                [-log import_log] [-s|-sh|-show] [-serverdir server_path]
                import_package_compressedfile
```

-ch|-column_header

Specifies to use a column header in the output format. See -ch|-column_header 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.

import_package_compressedfile

Specifies a client file system location of an import package compressed file.

-log *import_log*

Specifies a client file system location for the import log file. The import log file contains messages from the import operation. If a location is not specified, an import log is created in the current working directory.

-nch|-nocolumn_header

Specifies not to use a column header in the output format. See -nch|-nocolumn_header for details.

-nf|-noformat

Specifies that columns not be aligned. See -nf|-noformat for details.

-ns|-no_sort

Specifies not to sort the output. 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.

-serverdir *server_path*

Specifies a server file system location that is used as an alternative temporary directory for processing the import data. You must be in the ccm_admin role to use this option. By default, import uses the directory *dbpath*/import as a temporary directory. This option is helpful when there is insufficient disk space on the file system that hosts the Rational® Synergy database.

-s|-sh|-show

Shows details of the objects that were created or updated by import. By default, no such details are shown.

-u|-unnumbered

Suppresses automatic numbering of the output. The output is not numbered. See -u|-unnumbered for details.

ln command

You can create a symbolically linked object in the database. You must execute the command within the context of a project with a maintained UNIX work area on a UNIX client. Alternatively, you can create a controlled symbolic link from *file_spec* to *path_name*. The link can point to any path, and does not have to be a controlled object or a path within a maintained project work area.

Note: When you create a 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 current (or specified) task, and checked in to the *integrate* state. Set shared_project_directory_checkin to FALSE in your initialization file to disable the shared project feature.

The `ln` command supports the "Creating a symbolic link" subcommand.

Creating a symbolic link

You can create a symbolically linked object in the database or a controlled symbolic link from *file_spec* to *path_name*. This subcommand is available to UNIX users only.

About this task

```
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 to 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 to invoke the default text editor to compose and edit the comment. 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 to use the contents of the specified file 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 can use a work area reference form or a 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. The path does not have to be to 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
```

ls command

You can list 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.

The `ls` command supports the "Listing files" subcommand.

Listing files

You can display two categories of files. Objects under Rational® Synergy control are discussed in the `-l` command option. Files that exist only in the file system are discussed in the `-m` command option.

A new pseudo-property named `relative_path` is available for all controlled objects listed by a `ccm ls` command. This property is the relative path within the context project for that object, using a directory field separator of / on all platforms. If the object is not a member of the specified context project, by default the property is shown as an empty string.

About this task

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_header 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 to use a long default format. Use this format when a user-defined format is not specified with `-f|-format`.

-m

Shows both uncontrolled files and directories and 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_header for details.

-nf|-noformat

Specifies not to use column alignment. See -nf|-noformat for details.

-ns|-nosort|-no_sort

Specifies not to sort the command output. See -ns|-nosort 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 path 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
```

merge command

When you merge source files or directories, the merge tool compares the versions you selected. The tool then compares the differences of each version to the closest common ancestor. A new, merged, controlled version is created automatically when you exit your merge tool.

The `merge` operation works with both non-writable and writable 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 a task to be created automatically at the merge, the `release` value is obtained for the task from the project in which*file_spec1* resides.

The `merge` command supports the "Merging files or directories" subcommand.

Merging files or directories

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 default merge tool for the GUI is interactive, and the one for the CLI is automatic. The merge tool creates a new, controlled version of your file. If the merge is successful, the merge results are written to the new file.

When merging directories, the merge tool automatically accounts for both additions to and deletions from both directories in a new, controlled, merged directory.

If an object to be merged is a member of your project, Rational Synergy uses the new merged object in the project.

An area "in conflict" occurs when both versions have changes in the same place relative to the common ancestor. If your merged file contains any conflicts, the tool marks the conflicts so you can find them quickly and easily. The tool then writes the merge results to the new file.

The following example shows how the merged file is marked:

```
<<<<<<<file1 filename1
conflicting lines in file1
=======
conflicting lines in file2
>>>>>>>file2 filename2
```

The default merge tool is specified in the `ccm.properties` file for UNIX and Windows. You can specify a different merge tool for each object type. The specified merge tool applies to the specified type and any subtype without a specified merge tool. By default, only the ASCII file type has a merge tool defined.

Additionally, you can set encoding rules in the `ccm.properties` file so that merged files display in the correct encoding, as follows:

```
// Command to perform merge of source objects on UNIX and its
checkstatus.
ccm.cli.tools.merge.ascii.unix=%ccm_home/bin/util/cc_merge %ccm_home %{
encoding[null='CP1252']} %outfile %file1 %ancestor %file2^M

// Command to perform merge of source objects on Windows and its
checkstatus.
ccm.cli.tools.merge.ascii.windows="%ccm_home\\\\bin\\\\util\\\\cc_merge
.bat"
"%ccm_home" %{encoding[null='CP1252']} %outfile %file1 %ancestor %file2
```

The second parameter to the `ccm_merge` command allows you to specify an encoding for the file being merged. The syntax "`%{encoding[null='CP1252']}`" is interpreted as follows:

If the object type specifies a work area encoding with the `encoding_rules` attribute, use that encoding for the merge. If not, use the CP1252 encoding. (See File encodings for a discussion of the `encoding_rules` attribute.)

You can specify a default encoding other than CP1252 for this parameter. For example, the following syntax indicates to use the UTF8 encoding if an encoding is not specified on the object type:

```
ccm.cli.tools.merge.ascii.windows="%ccm_home\\\\bin\\\\util\\\\cc_merge
.bat"
"%ccm_home" %{encoding[null='UTF8']} %outfile %file1 %ancestor %file2
```

The following syntax indicates to always use the CP1252 encoding:

```
ccm.cli.tools.merge.ascii.windows="%ccm_home\\\\bin\\\\util\\\\cc_merge
.bat" "%ccm_home" CP1252 %outfile %file1 %ancestor %file2
```

The valid encodings are CP1252, UTF8, BIG5, eucJP, EUC-KR, SJIS, and GB18030.

The files being merged and the ancestor file must have the same encoding.

For example, in a Chinese-language database, you might set the following encoding rules for the `ascii` type:

```
Server-encoding: GB18030
Unix-wa-encoding: GB18030
Windows-wa-encoding: GB18030
```

Assuming the default encoding parameters for the `ccm_merge` command were used, the CLI and GUI merge tools are then invoked with the GB18030 encoding.

Alternatively, if your site contains only Chinese-language databases, you can change the default merge command as follows, and `encoding_rules` attributes are not required on the `ascii` type.

```
ccm.cli.tools.merge.ascii.windows="%ccm_home\\\\bin\\\\util\\\\cc_merge
.bat"
"%ccm_home" %{encoding[null='GB18030']} %outfile %file1 %ancestor %file
2
```

About this task

```
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 to add a comment on the new object containing the merged result. 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 to invoke the default text editor to compose and edit the comment. 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 to use the contents of the specified file 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 do not specify `-task` but you have set a current task, the current task is used by default. See Task specification for further details.

migrate command

The term *migration* describes the process of taking data located outside of a Rational® Synergy database and creating that data in a Rational Synergy database in a way that represents the external data.

For example, you might have files in some directory organization in a file system location and want to create projects, directories, and files that represent that data in a Rational Synergy database.

In Rational Synergy, migration is performed in two separate phases:

1. Use the `ccm migrate` command to create an import package that describes the data to be created.
2. Use the `ccm import` command to import the data created in the previous step and create the objects in the Rational Synergy database.

The advantage of this two-phase approach is that if you are migrating several separate applications or components, you can perform the first step on each piece. Then, you can perform the more time consuming import of all the packages afterward, for example, overnight.

The ccm migrate operation has two types of migration:

- A *full migrate*, which results in a new set of projects containing new instances of directories and files representing the data to be migrated. The generated import package represents all of the objects that are to be migrated.
- An *incremental migrate*, which updates an existing project hierarchy with only the changes required to match the data to be migrated. The existing project hierarchy is used as a reference to determine the changes. The generated import package represents the new, changed, or removed objects compared with the reference project hierarchy.

If you have several releases of an application to be migrated, here is a typical usage pattern:

1. Perform a *full migrate* and import of the first release, creating a project hierarchy representing that release.
2. Perform a *copy project* operation to create a version of the project for the next release.
3. Perform an *incremental migrate* and import of the next release into the existing project created by the previous step. The import must be performed before proceeding to the next release to ensure that you have the correct history of changes.
4. Repeat steps 2 and 3 as required for each subsequent release.

The `ccm migrate` operation uses mapping rules to determine which files to process or ignore. The operation also uses mapping rules to determine object properties, including their Rational Synergy type. Many mapping rules are determined by data on type definitions. For example, if a java type definition is associated with a suffix matching pattern for `.java`, then typically a file to be migrated with a name such as `ContextFactory.java` is represented by a Rational Synergy object of type `java`.

Previewing and showing information about migrate

The `ccm migrate` command supports showing the details of the object, either as a preview without creating an import package, or in addition to creating the import package. You can use standard formatting options to specify what data is shown and the format of the presentation.

Two types of data can be shown in migrate format strings:

- Special migrate properties.
- Migrate properties that correspond to Rational Synergy attributes after being imported.

The following special migrate property keywords are supported:

Table 1. Special migrate property keywords

Property name	Description
`ACTION`	The type of migrate action being performed: • New An instance of the object is created. • Modified An existing object is modified by checking out a new version. • Ignored Anew object is ignored. • Deleted An existing object is removed from the project because it is absent from the source path, or a migrate rule is ignoring it. • Unchanged An existing object is unchanged.
`INDENTED_NAME`	The name of the object prefixed with zero, one, or more leading spaces. The number of leading spaces is determined by the indentation level of directory nesting with respect to the top-level project.
`SOURCE_PATH`	The absolute path of the source object being migrated.
`RELATIVE_PATH`	The relative path of the source object being migrated with respect to the top-level project.

Any property name that is not one of these special cases is treated as a property of the migrate object. In most cases, a Rational Synergy attribute is created when the object is imported. For example, the `name` property corresponds to the `name` attribute after the data is imported.

The `migrate` command supports the following subcommands.

- Previewing a full migrate from a plain file system
- Previewing an incremental migrate from a plain file system
- Performing a full migrate from a plain file system
- Performing an incremental migrate from a plain file system

Previewing a full migrate from a plain file system

The `ccm migrate -preview` command performs a preview of a full migrate from a plain file system location. The preview contains objects that would be included in an import package. However, running this command does not generate an import package.

About this task

```
ccm migrate     -preview -p|-project new_project_spec
                [-dt|-default_type default_type] [-rules file_path]
                [-results results_file] [-f|-format format] [-nf|-noformat]
                ([-ch|-column_header] | [-nch|-nocolumn_header])
                [-sep|-separator separator]
                ([-sby|-sortby sortspec] | [-ns|-nosort|-no_sort])
                [-gby|-groupby groupformat] directory_path
```

-ch|-column_header

Specifies to use a column header in the output format. See -ch|-column_header for details.

directory_path

Specifies a client file system directory containing the data to be migrated.

-dt|-default_type *default_type*

Specifies the default Rational® Synergy type that is used if no mapping rules define the type of the object. If not specified, the default is ascii.

-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.

new_project_spec

Specifies that the new project is created if the data was imported.

-nch|-nocolumn_header

Specifies not to use a column header in the output format. See -nch|-nocolumn_header for details.

-nf|-noformat

Specifies not to use a column alignment. See -nf|-noformat for details.

-ns|-no_sort

Specifies not to sort the output. See -ns|-nosort for details.

-rules *file_path*

Specifies to use a mapping rule instead of the default rules.

-results *results_file*

Specifies a results file that contains the migrate preview results. If not specified, the preview results are written to the *stdout* stream.

-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.

Previewing an incremental migrate from a plain file system

The `ccm migrate -incremental -preview` command performs a preview of an incremental migrate from a plain file system location. This preview shows what objects would be included in an import package. However, the import package is not generated.

About this task

```
ccm migrate -preview -inc|-incremental -p|-project project_spec
        [-dt|-default_type default_type] [-rules file_path]
        [-results results_file] [-f|-format format] [-nf|-noformat]
        ([-ch|-column_header] | [-nch|-nocolumn_header])
        [-sep|-separator separator]
        ([-sby|-sortby sortspec] | [-ns|-nosort|-no_sort])
        [-gby|-groupby groupformat] directory_path
```

-ch|-column_header

Specifies to use a column header in the output format. See -ch|-column_header for details.

directory_path

Specifies a client file system directory containing the data to be migrated.

-dt|-default_type *default_type*

Specifies the default Rational® Synergy type that is used if no mapping rules define the type of the object. If not specified, the default is ascii.

-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_header for details.

-nf|-noformat

Specifies not to use a column alignment. See -nf|-noformat for details.

-ns|-no_sort

Specifies not to sort the output. See -ns|-nosort for details.

project_spec

Specifies that the existing project to be used as a reference and updated if the data was imported. See Project specification for details.

-rules *file_path*

Specifies to use a mapping rule instead of the default rules.

-results *results_file*

Specifies a results file that contains the migrate preview results. If not specified, the preview results are written to the *stdout* stream.

-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.

Performing a full migrate from a plain file system

The `ccm migrate` command performs a full migrate from a plain file system location. It also produces a compressed file of the import package that can be imported using the `ccm import` command.

About this task

```
ccm migrate -p|-project new_project_spec [-task task_spec]
        [-dt|-default_type default_type] [-rules file_path]
        [-results results_file] [-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] [-output compressedfile]
        directory_path
```

-ch|-column_header

Specifies to use a column header in the output format. See -ch|-column_header for details.

directory_path

Specifies a client file system directory containing the data to be migrated.

-dt|-default_type *default_type*

Specifies the default Rational® Synergy type that is used if no mapping rules define the type of the object. If not specified, the default is ascii.

-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.

new_project_spec

Specifies the new project to be created.

-nch|-nocolumn_header

Specifies not to use a column header in the output format. See -nch|-nocolumn_header for details.

-nf|-noformat

Specifies not to use a column alignment. See -nf|-noformat for details.

-ns|-no_sort

Specifies not to sort the output. See -ns|-nosort for details.

-output *compressedfile*

Specifies the client path and name of the compressed file to be created containing the import package. The *compressedfile* must have a .zip suffix. If this option is not specified, a compressed file is created in the current working directory. The default name of the compressed file is constructed by using the name of the project, an underscore, and a date string in the form "YYYYMMDD." The YYYY is a four-digit year, MM is a two-digit month, and DD is a two-digit day. If a file of that name exists, an additional underscore and a unique integer number are appended. For example, on 16th June 2010, the first default compressed file name for migrating a project named example would be example_20100616.zip. The second one performed on that date from the same project name would be example_20100616_1.zip.

-rules *file_path*

Specifies to use a mapping rule instead of the default rules.

-results *results_file*

Specifies a results file that contains the migrate preview results. If not specified, the preview results are written to the *stdout* stream.

-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*|-*sh*|-*show*

Specifies that the objects contained in the import package are shown in the results file or the command output.

-*task* task_spec

Specifies the task that is associated with all of the directories and files created when the import package is imported. If not specified, the current task is used by default.

Performing an incremental migrate from a plain file system

The `ccm migrate -incremental` command performs an incremental migrate from a plain file system location using a specified project as a reference. It produces an import package compressed file that might be imported later using the `ccm import` command.

About this task

```
ccm migrate -p|-project project_spec -inc|-incremental
        [-task task_spec] [-dt|-default_type default_type]
        [-rules file_path] [-results results_file] [-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] [-output compressedfile]
        directory_path
```

-ch|-column_header

Specifies to use a column header in the output format. See -ch|-column_header for details.

directory_path

Specifies a client file system directory containing the data to be migrated.

-dt|-default_type *default_type*

Specifies the default Rational® Synergy type that is used if no mapping rules define the type of the object. If not specified, the default is ascii.

-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_header for details.

-nf|-noformat

Specifies not to use a column alignment. See -nf|-noformat for details.

-ns|-no_sort

Specifies not to sort the output. See -ns|-nosort for details.

-output *compressedfile*

Specifies the client path and name of the compressed file to be created containing the import package. The *compressedfile* must have a .zip suffix. If this option is not specified, a compressed file is created in the current working directory. The default name of the compressed file is constructed by using the name of the project, an underscore, and a date string in the form "YYYYMMDD." The YYYY is a four-digit year, MM is a two-digit month, and DD is a two-digit day. If a file of that name exists, an additional underscore and a unique integer number are appended. For example, on 16th June 2010, the first default compressed file name for migrating a project named

example would be example_20100616.zip. The second one performed on that date from the same project name would be example_20100616_1.zip.

project_spec

Specifies the project to be used as a reference and updated when the import package is imported. See Project specification for details.

-rules *file_path*

Specifies to use a mapping rule instead of the default rules.

-results *results_file*

Specifies a results file that contains the migrate preview results. If not specified, the preview results are written to the *stdout* stream.

-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|-sh|-show

Specifies that the objects contained in the import package are shown in the results file or the command output.

-task *task_spec*

Specifies the task that is associated with all of the directories and files created when the import package is imported. If not specified, the current task is used by default.

move command

You can rename a file or project, and move files, directories, projects, subprojects, and the contents with the `move` command.

The following lists more specific uses of the `move` command:

- 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.

The `move` command supports these subcommands:

- Renaming a project
- Renaming or moving an object

Renaming a project

You can rename 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 after you rename the project, the root directory is automatically renamed.

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.

About this task

```
ccm mv|move|ren|rename -p|-project [-task task_spec] project_spec new_project_spec
```

new_project_spec

Specifies the properties to use for the renamed project, as follows:

- 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

You can rename a file or directory, or move one or more files, directories, or projects to another directory (which might be in a different project). If you enter two arguments and the last argument is not set to an existing directory, the command is interpreted as a rename operation. Otherwise, the command is interpreted as a move operation.

When you rename a file or directory and the parent directory is not writable, a version is checked out automatically and associated with task. If a task is not specified, the current task is used by default. You must check in the directory to make it available to other users. The directory is automatically checked in when you complete the task.

When you move an object to or from a non-modifiable directory and the parent directory is non-modifiable, a version is checked out automatically and associated with the specified task. If a task is not specified, the current task is used by default. You must check in the directory to make it available to other users, which is automatically done when you complete the task.

If you are working in a shared project and the parent directory is non-modifiable, the directory is automatically checked out and associated with the current task. The directory is also checked in to the *integrate* state. You can disable this feature by setting shared_project_directory_checkin to FALSE in your initialization file.

You do not need to be in a work area to use this command if as you use the Folder specification.

Note: If you rename a file or directory used in other directories, review the other locations that use the object. Likewise, if you rename a file or directory used in directory versions other than the current or specified directory, review the other locations that use the object. The parent directory is checked out, but not other parent directories. You must check out the other directories to use the renamed object.

About this task

```
ccm mv|move|ren|rename [-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 *file_spec* is not set to a 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 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.

Example

- 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
```

process command

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.

Table 1. Built-in processes, process rules, and purposes

Process	Process Rule	Purpose
standard	Collaborative Development	Collaborative Development
	Custom Development	Custom Development
	Insulated Development	Insulated Development
	Integration Testing	Integration Testing
	Shared Development	Shared Development
	System Testing	System Testing
	Visible Development	Visible Development
distributed	Custom Development	Custom Development
	Insulated Development	Insulated Development
	Local Collaborative Development	Collaborative Development
	Local Integration Testing	Integration Testing
	Master Integration Testing	Master Integration Testing
	Shared Development	Shared Development
	System Testing	System Testing
	Visible Development	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.

The `process` command supports these 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
- Setting the controlling database for a process
- Showing process information

- Showing a property for a process
- 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.

Before you begin

You must be in the *build_mgr* or *ccm_admin* role to use this subcommand.

About this task

```
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.

generic_process_rule_spec

Specifies the generic process rule to copy. See Process rule specification for details.

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. See Process rule specification for details about generic process rules.

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
```

Copying a process to a new process

This subcommand copies the properties and process rules from a specified process to a new process.

Before you begin

You must be in the *build_mgr* or *ccm_admin* role to use this subcommand.

About this task

```
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.

new_process_spec

Specifies the name of the process to create. See Process specification for details.

process_spec

Specifies the process to copy. See Process specification for details.

process_rule_spec

Specifies the process rule to copy. See Process rule 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"
```

Creating a process

This subcommand creates a process definition, which includes specified process rules and an optional specified diagram uniform resource locator (URL).

Before you begin

You must be in the *build_mgr* or *ccm_admin* role to create a process.

About this task

```
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 are running a browser, see IBM Rational Synergy Standard process diagrams to view the various diagrams available for your use. If you do not have the diagram ready, later you can set the URL to the path where the descriptive file resides.

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. See Process rule specification for details about generic process rules.

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"
```

Deleting a process

This subcommand deletes processes. By default, processes can be deleted only if they are not used by any project grouping.

Before you begin

You must be in the *build_mgr* or *ccm_admin* role to delete a process.

About this task

```
ccm process -d|-delete process_spec...
```

-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
```

Listing processes

This subcommand lists the processes matching the specified criteria. If no options are specified, it lists all processes.

About this task

```
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_header 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_header for details.

-nf|-noformat

Specifies not to use column alignment. See -nf|-noformat for details.

-ns|-nosort|-no_sort

Specifies not to sort the command output. 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 output. See -u|-unnumbered for details.

Example

- List all currently defined processes.

```
ccm process -list

1)Distributed
2)Standard
```

Modifying a process

This subcommand modifies the specified processes. You must specify at least one of the options for the command to succeed.

Before you begin

You must be in the *build_mgr* or *ccm_admin* role to use this subcommand.

About this task

```
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. See Process rule specification for details about generic process rules.

-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 do not have the diagram ready, later you can set the URL to the path where the descriptive file resides.

-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. See Process rule specification for details about generic process rules.

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. See Process rule specification for details about generic process rules.

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"
```

Setting the controlling database for a process

This subcommand modifies the controlling database for a process definition. Use `ccm process -cdb` to grant local control of the process. You can also use the subcommand to hand over control to another database or to accept the process from another controlling database.

About this task

```
ccm process -cdb|-controlling_database -local process_spec...

ccm process -cdb|-controlling_database
            -handover database_spec process_spec...

ccm process -cdb|-controlling_database
            -accept database_spec process_spec...
```

-accept *database_spec*

Specifies to accept control of the object from the specified database. See Database specification for details.

-handover *database_spec*

Specifies to hand over control of the object to the specified database. See Database specification for details.

-local

Specifies to take local control of the specified process.

process_spec

Specifies the process to modify. See Process specification for details.

Showing process information

This subcommand shows information about the specified processes.

About this task

```
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_header 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 -nch|-nocolumn_header 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.

Showing a property for a process

This subcommand shows the property of the specified process rules.

About this task

```
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.

Creating a diagram URL is optional; not all processes have one to view.

process_spec

Specifies the process that you want to show properties for. See Process specification for details.

Showing process rules for a process

This subcommand shows the process rules for the specified process.

About this task

```
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_header 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_header for details.

-nf|-noformat

Specifies not to use column alignment. See -nf|-noformat for details.

-ns|-nosort|-no_sort

Specifies not to sort the command output. 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 -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 output. See -u|-unnumbered for details.

process_rule command

A process rule specifies how a baseline is chosen for the update properties for a project grouping and project. Use this command to display and set process rules.

Process rules support these 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 manually select a baseline for the update properties for a project grouping. 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. Set up rules, and then select rules to apply to a release and purpose, and switch among the set of process rules during 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 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 release value, a process rule is created for the combination of 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 purpose, a general process rule is created for that purpose. The build manager must then edit the 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 these exceptions:

- In the standard process, Collaborative Development collects all completed tasks from all databases. 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. 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 the CLI when a purpose is specified.

You can specify which process rules to use when you create a release. For more information, see the release command.

The `process_rule` command supports these 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 and folder templates to the specified process rules. You can add folder templates to a generic process rule only. You can add folders or folder templates to release specific process rules.

Before you begin

You must be working as a Process rules manager to use this command.

About this task

```
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 can have folder templates only. 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 template specification for details.

Example

- 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"
```

Comparing process rules

This subcommand compares two process rules.

About this task

```
ccm pr|process_rule|ut|update_temp|update_template|
       rt|recon_temp|reconfigure_template
       -comp|-compare process_rule_spec1 process_rule_spec2
```

process_rule_spec1

Specifies the process rules to be compared. See Process rule specification for details.

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"
```

Copying a process rule

This subcommand copies a process rule to another process rule.

The following rules apply when you copy a process rule.

- 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.

- Generic or release-specific to release-specific copies

You cannot copy either a generic or a release-specific process rule to a new release-specific process rule.

Before you begin

You must be working as a Process rules manager to use this command.

About this task

```
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"
```

Deleting a process rule

This subcommand deletes a process rule. By default, you can delete a process rule when it is not used by any project grouping. Use the `-force` option to delete a process rule that is in use.

Before you begin

You must be working as a Process rules manager to use this command.

About this task

```
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"
```

Listing process rules

This subcommand lists the process rules matching the specified criteria. If no options are specified, it lists all process rules.

About this task

```
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_header 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_header for details.

-nf|-noformat

Specifies not to use column alignment. See -nf|-noformat for details.

-ns|-nosort|-no_sort

Specifies not to sort the command output. 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 output. See -u|-unnumbered for details.

Example

- List all currently defined process rules and suppress numbering.

```
ccm process_rule -list -u
```

```
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
```

Modifying a process rule

This subcommand modifies the specified process rules.

Before you begin

You must be working as a Process rules manager to use this command.

About this task

```
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 to append the releases 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 of 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 can have folder templates only.

-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 for the process rule to 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 for the process rule to 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 for the process rule to use the latest static projects. This option cannot be used with the `-pb|-prep_baseline` option.

-lsbmp|-latest_static_or_build_management_projects

Specifies for the process rule to 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, any additional criteria is matched 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, 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. Set the Baseline Versions Matching option to **1.0_gr** to use the project with a version of **1.0_gr** 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 might 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 selects 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. The project checks for a matching platform, then uses **1.0_gr_unix**. This choice is made because the database is set up to support development of parallel platforms by default.)

-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 to consider prep state projects 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.

Example

- 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"
```

Removing folders and folder templates from a process rule

This subcommand removes folders and folder templates from the specified process rules. Generic process rules can have folder templates only.

Before you begin

You must be working as a Process rules manager to use this command.

About this task

```
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 can have folder templates only.

-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.

Setting the controlling database for a process rule

This subcommand sets DCM to either handover to a specified database, accept updates only from a specified database, or take local control.

About this task

```
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*.

-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. When you hand over control to a spoke from 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 operation is permitted, or a blank string. A blank string means that control cannot be handed over to that database.

-local

Specifies to take local control of the specified process rules.

process_rule_spec

Specifies the process rule to set the controlling database for. See Process rule specification for details.

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.

About this task

```
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

Specifies to use a column header in the output format. See -ch|-column_header 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_header for details.

-ns|-no_sort

Specifies not to sort the command output. See -ns|-nosort for details.

process_rule_spec

Specifies the process rule for the objects to be shown. See Process rule 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|-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

Suppresses automatic numbering of the command output. See -u|-unnumbered for details.

Showing a property for a process rule

This subcommand shows the property of the specified process rules.

About this task

```
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...
```

process_rule_spec

Specifies to show the property for the process rule. See Process rule specification for details.

-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.

Showing process rule information

This subcommand shows information about the specified process rules.

About this task

```
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

Specifies to use a column header in the output format. See -ch|-column_header 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_header for details.

-ns|-no_sort

Specifies not to sort the command output. See -ns|-nosort for details.

process_rule_spec

Specifies which process rule to show information for. See Process rule 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.

-u|-unnumbered

Suppresses automatic numbering of the command output. See -u|-unnumbered for details.

Example

- Show properties of the `2.1:Insulated Development` process rule.

```
ccm process_rule -show info "2.1:Insulated Development"
```

product_info command

You can show release and patch level information for the Rational® Synergy client, server, database schema, or database server.

If you run the command outside of a CLI session, you can show client information only. You must run the command in a CLI session to report on the server, database, or database server.

The `product_info` command supports the “Showing product version information” subcommand.

Showing product version information

Use this command to show Rational® Synergy release and patch level information.

About this task

```
ccm product_info -all
```

```
ccm product_info [-client_release] [-client_patch_level]
[-client_build_number] [-client] [-server_release]
[-server_patch_level] [-server_build_number] [-server]
[-database_schema] [-database_version] [-model_info]
[-database_info] [-database]
```

-all

Specifies to show all product version information.

-client

Specifies to show all client product information. Using this option is equivalent to specifying `-client_release`, `-client_patch_level`, and `-client_build_number`.

-client_release

Specifies to show the client release.

-client_patch_level

Specifies to show the client patch level.

-client_build_number

Specifies to show the client build number.

-database

Specifies to show all the database information. Using this option is equivalent to specifying `-database_schema`, `-database_version`, `-model_info`, and `-database_info`. This option requires that you run the command in a CLI session.

-database_info

Specifies to show database information. This option requires that you run the command in a CLI session.

-database_schema

Specifies to show database schema version information. This option requires that you run the command in a CLI session.

-database_version

Specifies to show database version and patch level information. This option requires that you run the command in a CLI session.

-model_info

Specifies to show model information for the database. This option requires that you run the command in a CLI session.

-server

Specifies to show all Synergy server information. Using this option is equivalent to specifying `-server_release`, `-server_patch_level`, `-server_build_number`. This option requires that you run the command in a CLI session.

-server_build_number

Specifies to show the server build number. This option requires that you run the command in a CLI session.

-server_patch_level

Specifies to show the server patch level. This option requires that you run the command in a CLI session.

-server_release

Specifies to show the server release level. This option requires that you run the command in a CLI session.

project command

The `project` command determines the project associated with a specified *file_spec* or the current working directory. The *file_spec* is typically a folder specification within a maintained work area.

The `project` command supports the "Displaying project information" subcommand.

Displaying project information

Use this command to identify a project associated with the current working directory or a user-specified path.

About this task

```
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_header for details.

file_spec

Specifies an object whose parent directory is 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_header for details.

-nf|-noformat

Specifies not to use column alignment. See -nf|-noformat for details.

-ns|-no_sort

Specifies not to sort the output. 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 output (the output is not numbered). 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
```

project_grouping command

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 at project update so that member selection is consistent across all projects. A project can be a member of only one project grouping. A project grouping is created 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 these ways:

- `My release purpose Projects`

The current user owns the project grouping and the database is not DCM-enabled, or the project grouping was created in the local database. The following is an example of a private project grouping,`My CM/6.5 Insulated Development Projects`.

- `owner's release purpose Projects`

A different user owns the project grouping and the database is not DCM-enabled, or the project grouping was created in the local database. The following is an example of a private project grouping, `John's CM/6.5 Insulated Development Projects`.

- `My` *`release purpose`* `Projects from Database` *`dbid`*

The current user owns the project grouping, the database is DCM-enabled, and the project grouping was not created in the local database. The following is an example of a private project grouping, `My CM/6.5 Insulated Development Projects from Database D`.

- `owner's release purpose Projects from Database dbid`

A different user owns the project grouping, the database is DCM-enabled, and the project grouping was not created in the local database. The following is an example of a private project grouping,`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 these ways:

- `All` *`release purpose`* `Projects from Database` *`dbid`*

For DCM-enabled databases, the `dbid` is the ID of the database where the project grouping was created. The following is an example of a non-private project grouping, `All CM/6.5 Integration Testing Projects from Database D`.

- `All` *`release purpose`* `Projects`

For databases not DCM-enabled, 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 might not have update properties specified by the project grouping. Projects that use process rules 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. Additionally, 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 set of tasks 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.

The `project_grouping` command supports these 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
- Removing tasks from the update properties for a project grouping
- Setting the auto-update mode for a project grouping
- Showing a project grouping property
- Setting a custom baseline for project groupings
- Showing the associated projects, baseline, folders, tasks, or objects for a project grouping
- Showing project grouping information

- Updating the baseline and tasks for a project grouping

Adding tasks to the update properties for a project grouping

You can add one or more specified tasks to the specified project groupings, or add all previously removed tasks back into each specified project grouping. Any user who can modify the specified project groupings can use this command.

About this task

```
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 add tasks to. See Project grouping specification for details.

project_grouping_spec2

Specifies the project groupings to add tasks to.

Example

- 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

You can compare projects for a project grouping with projects for a specified baseline.

About this task

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_header 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 -nch|-nocolumn_header for details.

-nf|-noformat

Specifies not to use column alignment. See -nf|-noformat for details.

-ns|-no_sort

Specifies not to sort the output. 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 output (the output is not numbered). 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"
```

Comparing projects for two project groupings

You can compare projects in two project groupings.

Before you begin

You must specify `-union`, `-intersection`, or `-not_in`.

About this task

```
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
```

```
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
```

Comparing tasks for a project grouping with tasks for a baseline

You can compare all tasks belonging to a project grouping with all tasks belonging to a baseline.

About this task

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_header 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 -nch|-nocolumn_header for details.

-nf|-noformat

Specifies not to use column alignment. See -nf|-noformat for details.

-ns|-no_sort

Specifies not to sort the output. 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 output (the output is not numbered). 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"
```

Comparing tasks for project groupings

You can compare all tasks for two specified project groupings.

Before you begin

You must specify `-union`, `-intersection`, or `-not_in`.

About this task

```
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_header 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 -nch|-nocolumn_header for details.

-nf|-noformat

Specifies not to use column alignment. See -nf|-noformat for details.

-ns|-no_sort

Specifies not to sort the output. 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 output (the output is not numbered). 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"
```

Copying tasks from one project grouping to another

You can copy tasks from one project grouping to another. Any user who can modify the destination project grouping can copy tasks to it.

About this task

```
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. Copying tasks gives you a way to add a set of tasks to a different project grouping.

project_grouping_spec1

Specifies which project groupings to copy tasks from. See Project grouping specification for details.

project_grouping_spec2

Specifies which project groupings to copy tasks to. See Project grouping specification for details.

Example

- Copy tasks from on 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
```

Deleting a project grouping and members

You can delete a project grouping, either with or without its member projects. To delete a project grouping, you must be able to modify it, and the project grouping must not have member projects.

About this task

```
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 to delete the associated projects and project grouping for 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 to delete the associated projects for the project grouping. The operation succeeds if the specified project grouping has no associated projects. The default is `-nm|-no_members` if options are not specified.

project_grouping_spec

Specifies the project groupings to delete. See Project grouping specification for details.

Listing project groupings

You can list the project groupings that match the specified criteria. If you do not specify options, all project groupings are listed.

About this task

```
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_header 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_header for details.

-nf|-noformat

Specifies not to use column alignment. See -nf|-noformat for details.

-ns|-nosort|-no_sort

Specifies not to sort the output. See -ns|-nosort for details.

-o|-owner *owner*

Specifies to list only project groupings with the specified owner. 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. Set the purpose to the name of a valid defined purpose. If not specified, project groupings for all purposes are listed.

r|-release *release_spec*

Specifies to list only project groupings for the specified release. 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 output (the output is not numbered). 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
```

Previewing update baseline and tasks for a project grouping

You can perform 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.

About this task

```
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 is processed in the specified order.

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"
```

Removing tasks from the update properties for a project grouping

You can remove tasks from specified project groupings. The command supports removing specified tasks, all tasks currently in the project grouping, and all tasks that were manually added to the project grouping. Any user who can modify the project grouping can remove tasks from the update properties for a project grouping.

About this task

```
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.

Setting the auto-update mode for a project grouping

You can define whether the baseline and tasks are automatically updated for specified project groupings. By default, when members of a project or project grouping are updated, the baseline and tasks for the project grouping are also updated. However, you can clear or set the auto-update feature.

About this task

```
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. Updating the baseline and tasks 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.

Showing a project grouping property

You can show a specific property for selected project groupings.

About this task

```
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

Specifies to display the name of the database where the project grouping was created.

o|owner

Specifies to display 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
```

Setting a custom baseline for project groupings

You can set a custom baseline for specified project groupings.

About this task

```
ccm pg|project_grouping -scb|-set_custom_baseline baseline_spec project_grouping_spec...
```

baseline_spec...

Specifies the baseline to be used by the project grouping. See Baseline specification for more information.

project_grouping_spec

Specifies which project groupings to set the custom baseline for. See Project grouping specification for details.

Showing the associated projects, baseline, folders, tasks, or objects for a project grouping

You can show the associated projects, baseline, folders, tasks, or objects for the specified project groupings.

About this task

```
ccm pg|project_grouping -s|-sh|-show ((proj|projects) | (bl|baseline) |
        (fo|folders) | (at|added_tasks) | (rt|removed_tasks)
        (tob|tasks_on_top_of_baseline) | | all_tasks |
        (obj|objs|objects) | automatic_tasks) [-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

Specifies to use a column header in the output format. See -ch|-column_header 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 -nch|-nocolumn_header for details.

-nf|-noformat

Specifies not to use column alignment. See -nf|-noformat for details.

-ns|-nosort|-no_sort

Specifies not to sort the command output. See -ns|-nosort for details.

project_grouping_spec

Specifies the project grouping to show. For more information, see Project grouping specification.

-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|-sh|-show

These keywords are supported with `-show:`

- `all_tasks`

If specified, shows the set of tasks used by the update operation

- `at|added_tasks`

If specified, all tasks that are in Added Tasks for the project grouping are displayed.

- `automatic_tasks`

If specified, shows the automatic tasks in the project grouping. An automatic task is automatically created by Rational Synergy and related to a nonstatic project or nonstatic product.

- `bl|baseline`

If specified, the baseline name is displayed.

- `fo|folders`

If specified, the folder name is displayed.

- `obj|objs|objects`

If specified, all objects that are included in all projects in the project grouping are displayed.

- `proj|projects`

If specified, all projects that are included in the project grouping are displayed.

- `rt|removed_tasks`

If specified, all tasks that are in Removed Tasks for the project grouping are displayed.

- `tob|tasks_on_top_of_baseline`

If specified, displays the tasks that are in the project grouping but are not in the baseline.

-u|-unnumbered

Suppresses automatic numbering of the command output. See -u|-unnumbered for details.

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
```

Showing project grouping information

You can show information about the specified project groupings.

About this task

```
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

Specifies to use a column header in the output format. See -ch|-column_header 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.

-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

Specifies not to use a column header in the output format. See -nch|-nocolumn_header for details.

-nf|-noformat

Specifies not to use column alignment. See -nf|-noformat for details.

project_grouping_spec

Specifies the project groupings to show. See Project grouping specification for details.

-sep|-separator *separator*

Used only with the `-f|-format` option. Specifies a different separator character. See -sep|-separator for details.

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
```

Updating the baseline and tasks for a project grouping

You can update the baseline and tasks for 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.

About this task

```
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.

Example

- Update the baselines and tasks for the project grouping.

```
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"
```

project_purpose command

The `project_purpose` command creates or shows (depending on your user role) the project purposes for a 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 following project purposes are included in each database.

Purpose name

This name reflects the purpose, for example, performance testing, personal use, and so on

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 can define three unique levels of system testing called `sqa1`, `sqa2`, and `sqa3`.

Status of the project

The status shows what state projects (*working*, *prep*, and so on) of this purpose can use.

The project purpose table affects:

- Options you can specify in these 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 are associated with
- Effects the synopses of corresponding automatic tasks

Each database contains one project purpose list only. You can define project purpose lists for each release.

The project purpose table defines these purposes:

```
Integration Testing:         prep:              integrate
System Testing:              prep:              sqa
Insulated Development:       working:           working
Collaborative Development:   working:           collaborative
Shared Development:          shared:            shared
Visible Development:         visible:           visible
Master Integration Testing:  master_integrate:  prep
```

The `project_purpose` command supports these 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.

Before you begin

You must be in the *build_mgr* or *ccm_admin* role to use this subcommand.

About this task

```
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.

Make the purpose and the member status 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 do not 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.

If you are using a DCM initialized database, you can create the same purpose in other databases in the DCM cluster.

-stat|-status *status*

Specifies the state of the new purpose. Make the state modifiable, 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 occurs.

- 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.

Before you begin

You must be in the *build_mgr* or *ccm_admin* role to use this subcommand.

About this task

```
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

About this task

- 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.

Before you begin

You must be in the *build_mgr* or *ccm_admin* role to use this subcommand.

About this task

```
ccm project_purpose -m|-modify [-n|-name purpose_name]
[-ms|-member_status member_status] [-force] purpose_name
```

-force

Specifies to modify the project purpose even if the specified new *member_status* value is used by existing objects in the database.

-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.

Make the purpose and the member status similar. For example, if you are creating a purpose of `Test Integration`, set the member status 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

Use this subcommand to view a project purpose.

About this task

```
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_header 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_header 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 not to sort the command output. 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. The setting 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
```

properties command

Use the `properties` command to find information about one or more objects. You can display the attribute values of a group of model-defined attributes for the specified objects to standard output.

The `properties` command supports these subcommands:

- Showing properties
- Showing properties with a specified format

Showing properties

You can show 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.

About this task

```
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 grouping specification for details.

-v|-verbose

Shows a more detailed set of information for certain types of objects, such as folders. The option is ignored for object types that do not have a verbose information form.

Example

- 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

You can show 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.

About this task

```
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_header 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 -nch|-nocolumn_header for details.

-nf|-noformat

Specifies not to use column alignment. See -nf|-noformat for details.

-ns|-nosort|-no_sort

Specifies not to sort the output. See -ns|-nosort for details.

-p|-project

Shows the history of a project.

project_spec

Specifies the project to list. See Project grouping specification for details.

-sep|-separator *separator*

Used only with the -format option. Specifies a different separator character. See -sep|-separator for details.

query command

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 do not specify a query expression using query-related options or a `query_string` argument, the command shows the current selection set and applies any sorting, then updates the selection set.

Query functions and sorting

To use a query function with sorting (for example, `recursive_is_member_of`), the query function sorting order is applied to the final 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 forms 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())`.

For detailed information about query expressions, see Rational Synergy Query Expressions.

The `query` command supports the "Querying for objects or showing the query selection set" subcommand.

Querying for objects or showing the query selection set

You can search for objects in the database.

About this task

```
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_header 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 -nch|-nocolumn_header for details.

-nf|-noformat

Specifies not to use column alignment. See -nf|-noformat for details.

-ns|-no_sort

Specifies not to sort the command output. 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.

-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 output. See -u|-unnumbered for details.

Example

- List all objects named `main.c` owned by *jane*.

```
ccm query -n main.c -o jane
1) main.c-1 integrate jane nasub1 csrc 1 1
2) main.c-1.2 working jane nasub1 csrc 1 4
3) main.c-2 working jane nasub2 csrc 1 5
```

- To look at the source contents of item 3 in the selection set, enter.

```
ccm cat @3
```

- List all objects named `main.c`, owned by *ann*, for task 4.

```
ccm query -n main.c -o ann -task 4
1) main.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"
```

reconcile command

You can compare 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 these cases:

- You modified a file in your work area, regardless of whether you checked it 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 unavailable 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 unavailable 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, use the `reconcile` command to sync your work area and the database.
- On UNIX, if a tool you are using breaks the links between objects you are modifying and the database, the `reconcile` command reconciles the changes and reestablishes the links.

For example, if you are not running a session but you must modify a non-modifiable object, you can change the object in your work area. You can update the database later by resetting the Read Only attribute on the file and modifying it. Later, when you start a session, 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, a message displays stating that errors might occur in your work area. The errors do not occur until you try to use the work area. To avoid problems, reconcile the work area before you use it.

Some operations perform some reconcile actions automatically:

- A file is checked in and a context project was available.

The corresponding work area is examined for work area conflicts. Where possible, changes to the work area are used to update the database automatically.

- A modifiable file is changed in a work area, and a different version is used as a result of a `ccm use` or `ccm update_members` command.

The database is updated with the changed file contents from the work area.

- 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.

- 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 are updated with the new contents. The update occurs when the database is either updated explicitly by a `ccm reconcile` command, or automatically as part of other operations.

The `reconcile` command supports these 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

You can identify and show work area conflicts; however, you cannot use this command to perform any action to resolve conflicts.

About this task

```
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_header 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. Use an option value with a list of one or more file extensions, separated by a comma.

-imwf|-ignore_missing_wa_file

Specifies to ignore files that are missing from the work area and not report them as work area conflicts. The default is to ignore missing work area files.

-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 to report missing work area files 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_header 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 subprojects for a project or the files for a directory or subdirectories when a project or directory is reconciled. The default is not to reconcile recursively.

-ns|-nosort|-no_sort

Specifies not to sort the output. See -ns|-nosort for details.

project_spec

Specifies the project to be reconciled.

-r|-recurse

Specifies to recursively reconcile subprojects, files, and subdirectories for a project. The default is not to reconcile recursively.

This option controls the depth of a reconcile operation when you synchronize a project. The depth of a reconcile is important to consider. If you synchronize a top-level project with many nested subprojects, the operation takes a substantial amount of time and resources to recursively reconcile every subproject beneath your specified top-level project.

If you specify a directory and `-recurse`, subprojects are not reconciled recursively 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. The default is to show the work area conflicts.

-udb|-update_db

Updates the database with versions in your work area. Uses of this option include:

- If you modify a file that is not checked out, reconcile creates a version by default, and the database is updated with your changes.
- If you update the database copy of a file from another work area and change the same file from this work area, the database is updated from the 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.

Example

- Reconcile the `ico_june16-1` project, but do not reconcile files whose file name contains any of these 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, you must update **move.c**, which was in the working state, and **colname.c**, which was in the integrate state. After you copied and modified the objects in your work area, the direction of the project changed and you ended up not needing these changes after all.

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.

```
% cd ~john/ccm_wa/ccmint15% lsico_june16-1$ ccm reconcile -p
ico_june16-1 -no_recurseExamining work area for conflicts...not
recursing hierarchy, conflicts will be automatically discardedUpdating
'/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.
```

Synchronizing a work area with changes from the database

You can update 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.

About this task

```
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 Showing work area conflicts.

-imwaf|-ignore_missing_wa_file

Specifies to ignore files that are missing from the work area and not to recreate the files from the database copies. The default is to ignore missing work area files.

-iu|-ignore_uncontrolled

Specifies to ignore uncontrolled files during reconcile. The default is to ignore uncontrolled files.

-mwaf|-missing_wa_file

Specifies to process missing work area files by unusing the corresponding members from the project in the database. The objects are not deleted from the database. The default is to ignore missing work area files.

-nr|-no_recurse

See Showing work area conflicts.

project_spec

Specifies the project to be reconciled.

-r|-recurse

See Showing work area conflicts.

-uwa|-update_wa

See Showing work area conflicts.

Example

- 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
```

Synchronizing the database with changes from a work area

You can update the database with 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.

About this task

```
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 Showing work area conflicts.

-imwaf|-ignore_missing_wa_file

Specifies to ignore files that are missing from the work area and not to remove or delete the corresponding members of the project. The default is to ignore missing work area files.

-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 to create missing work area files from corresponding objects in the database. The default is to ignore missing work area files.

-nr|-no_recurse

See Showing work area conflicts.

-p|-project *project_spec*

Specifies the project to be reconciled.

-r|-recurse

See Showing work area conflicts.

-t|-task *-task_spec*

Specifies to associate the task with 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 Showing work area conflicts.

Example

Reconcile the file `main.c` by updating the database from the work area.

```
ccm reconcile -update_db main.c-1:csrc:1
```

relate command

Use this command to add a relationship (`relation_name`) between `file_spec1` and `file_spec2`, or to show the relationship with the specified data.

The `relate` command supports these subcommands:

- Creating a relationship from one object to another
- Showing relationships to and from an object

Creating a relationship from one object to another

You can create a specified relationship from an object to one or more destination objects, or from specified objects to a single destination object. If you create a relationship that exists, the operation succeeds although the command does nothing.

About this task

```
ccm relate -n|-name relationship_name [-f|-from from_object_spec]...
           [-t|-to to_object_spec]...
```

-f|-from *from_object_spec*...

Specifies the originating object or objects from which the relationship is created. If more than one origin object is specified, you cannot specify multiple destination objects.

-n|-name *relationship_name*

Specifies the name of the relationship to create.

t|-to *to_object_spec*...

Specifies the destination object or objects to which the relationship is created. If more than one destination object is specified, you cannot specify multiple origin objects.

Example

- 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 6.

```
ccm relate -name successor -from print.c-5.1.1:csrc:1 -to print.c-6:csrc:1
```

Showing relationships to and from an object

You can show relationships from an object to one or more destination objects, or from specified objects to a single destination object.

About this task

```
ccm relate -s|-sh|-show [-n|-name relationship_name]
           [(-f|-from from_object_spec)...]
           [(-t|-to to_object_spec)...]
           ([-l] [-fmt|-format format]) [-nf|-noformat]
           [-sep|-separator separator] |
           ([-ch|-column_header] | [-nch|-nocolumn_header])
           ([-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_header 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 originating object or objects from which the relationship is shown. If not specified, relationships from any origin object are shown. You must specify either `-from` or `-to`.

-gby|-groupby *groupformat*

Specifies how to group the command output. See -gby|-groupby for details.

-l

Specifies to use the default long format.

-n|-name

Specifies the name of the relationship to show. If not specified, shows the names of all relationships.

-nch|-nocolumn_header

Specifies not to use a column header in the output format. See -nch|-nocolumn_header for details.

-nf|-noformat

Specifies not to use column alignment. See -nf|-noformat for details.

-ns|-nosort|-no_sort

Specifies not to sort the command output. 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 destination object or objects to which the relationship is shown. If not specified, relationships to any object are shown. You must specify either `-from` or `-to`.

Example

- Show the relationships between `clear-2` and `clear-1`.

```
ccm relate -s -f clear-1 -t clear-2
```

- Show every relationship in the database from `clear-1` to any other object version.

```
ccm relate -s -f clear-1
```

release command

Use this command to create, modify, delete, show, and rename 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 can 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 uses the component release only, such as **2.0**. Such releases are said to use a null component name. Use a component name to keep releases for the same component logically related and to avoid the need for manual naming conventions.

A release definition has a number of important properties. The baseline release is used when updating projects. For example, a process rule uses the `%baseline_release` keyword in its baseline release-purpose list. The process rule denotes the baseline release of the release associated with the project or project grouping being updated. A release can be marked as inactive to prevent the release being used by developers for further development.

Component names and component releases must **not** start with these characters:

/ \ ' " : * ? [ ] @ % - + ~ space, tab

Second and subsequent characters **cannot** include:

/ \ ' " : * ? [ ] @ %

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, the release is automatically copied 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 and rename the release. For example, if the release is in active development but the release name must change, then the *ccm_admin* user must use a session that was started in Single user admin mode before changing the release name.

- 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 are updated. You must be in the *ccm_admin* role if other associated objects are updated.

The `release` command supports these subcommands:

- Creating a release
- Deleting a release
- Listing releases
- Modifying a release
- Renaming a release
- Setting the controlling database for a release
- Showing a property for a release
- Showing the process rules for a release
- Showing release information

Creating a release

This subcommand creates a release definition.

To create a release for a new application or component, you can use 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 the properties used in 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.

Before you begin

You must be in the *build_mgr* or *ccm_admin* role to use this subcommand.

About this task

```
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 setting is the default.

-allow_dcm_transfer

Specifies that the release is eligible for DCM replication if included by the release scope and query for the transfer set. 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 setting is the default when creating a 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. You cannot combine `parallel check in` without `parallel check out`.

-allow_parallel_check_out

Specifies that parallel check-out for objects with this release is permitted. This setting is the default when creating a 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. You cannot combine `parallel check in` without `parallel check out`.

-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 requires component tasks. When you create the baseline, component tasks are created automatically. This setting 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 takes the `-description` option value, appends the description read from the file specified by `-description_file`, and starts the current default text editor to show the comment. The text saved from the editor is then used for setting the description.

-desc_edit|-description_edit

Starts 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`.

-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 settings for the release are copied from the previous release. The previous release is used as the baseline release by default.

-groups *groups*

Specifies the groups that can 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 used by default for object status-based updates only. Included releases are used to weight 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 a one-line string only.

-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 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. You cannot combine parallel check in without parallel check out.

-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 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. You cannot combine parallel check in without parallel check out.

-nocct|-nocreate_component_tasks

Specifies that component tasks are not 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. 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 are created.

release_spec

Specifies the name of the new release to create.

Example

- 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 ore 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 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.

Before you begin

You must be in the *build_mgr* or *ccm_admin* role to delete a release.

About this task

```
ccm release -d|-delete [-force] release_spec...
```

-force

Specifies to delete the release 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 fails.

release_spec

Specifies the releases to delete. See Release specification for details.

Example

- Delete the release definition for `sue/6.5`, even if 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.

About this task

```
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` or `-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_header 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` or `-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_header for details.

-nf|-noformat

Specifies not to use column alignment. See -nf|-noformat for details.

-ns|-nosort|-no_sort

Specifies not to sort the command output. 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 output. 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.

Before you begin

You must be in the *build_mgr* or *ccm_admin* role to use this subcommand.

About this task

```
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 to set the releases as eligible for DCM replication if included by the release scope and query for a transfer set.

-allow_parallel_check_in

Specifies that parallel check-in for objects with this release is permitted. You cannot combine `parallel check in` without `parallel check out`.

-allow_parallel_check_out

Specifies that parallel check-out for objects with this release is permitted. You cannot combine `parallel check in` without `parallel check out`.

-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 requires component tasks. When you modify the baseline, component tasks are created automatically.

-cpr|-clear_process_rules

Clears any existing process rules before adding a process rule, and sets an absolute set of process rules.

-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 takes the `-description` option value, appends the description read from the file specified by `-description_file`, and starts the current default text editor to show the comment. The text saved from the editor is then used for setting the description.

-desc_edit|-description_edit

Starts 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`.

-desc_file|-description_file *file_path*

Specifies a path to a file containing a description.

-groups *groups*

Specifies the groups that can 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 to set the releases being modified to inactive. Inactive releases cannot be used by developers for development work.

-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 used by default for object status-based updates only. Included releases are used to weight the selection scoring while update members is running.

-included_release_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 a one-line string only.

-m|-modify

Modifies the specified release.

-noallow_dcm_transfer

Specifies to set the releases as ineligible for DCM replication.

-noallow_parallel_check_in

Specifies that parallel check in for objects with this release is not permitted. You cannot combine `parallel check in` without `parallel check out`.

-noallow_parallel_check_out

Specifies that parallel check-out for objects with this release is not permitted. You cannot combine `parallel check in` without `parallel check out`.

-nocct|-nocreate_component_tasks

Specifies not to automatically create component tasks when baselines are published for the release definition being modified. You can create component tasks manually after a baseline is published for the release. See Creating component tasks for a baseline for details.

-phase *phasename*

Specifies to set the release phase for the specified releases. The valid release phases are defined in the model attribute. 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
  ```

Renaming a release

You can rename a release to a new name.

Before you begin

You must be in the *build_mgr* or *ccm_admin* role to rename a release. You might need to use a CLI session that was started in Single user admin mode first.

About this task

```
ccm release -d|-delete -rename ([-all] | [-local] |
             [(-dbid|-database_id database_spec)...]) [-force] [-preview]
             [-nocheck] oldrelease_spec newreleasename
```

-all

Specifies to update objects from all databases referencing the old release name to reference the new release name.

-dbid|-database_id *database_spec*...

Specifies to update objects created in the specified database referencing the old release name to reference the new release name. See Database specification for details.

-force

Specifies to rename the release even if there are existing objects referencing the new release name.

-local

Specifies to update objects locally created in this database referencing the old release name to reference the new release name.

-nocheck

Specifies to rename the release even if the user is not running in single user admin mode. If objects other than the release definition and its process rules require update, and the user is not in the *ccm_admin* role using single user admin mode, the command reports an error and the release is not renamed.

newreleasename

Specifies the new name for the release.

oldrelease_spec

Specifies the old release name. See Release specification for details.

-preview

Specifies to preview a summary of the changes to be made. No data is modified.

Setting the controlling database for a release

This subcommand sets the controlling database for one or more releases.

Before you begin

You must be in the *build_mgr*, *dcm_mgr*, or *ccm_admin* role to use this subcommand.

About this task

```
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.

-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 use this option when the release is locally controlled.

-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.

Example

- Handover control of a locally controlled release definition to a database whose ID is A1.

```
ccm release -controlling_database -local -handover A1 -component releasename
```

Showing a property for a release

You can show release properties for a specified release.

About this task

```
ccm release -s|-sh|-show (active | allow_dcm_transfer | baseline |
    create_time | (desc|description) | groups | included_releases |
    manager | modifiable_in | owner | parallel_check_out |
    parallel_check_in | phase | phase_log) release_spec...
```

release_spec

Specifies the releases to display. See Release specification for details.

-s|-sh|-show (active)

Shows the active releases for the specified release.

-s|-sh|-show (allow_dcm_transfer)

Shows if the specified release allows DCM transfers.

-s|-sh|-show (baseline)

Shows the baseline for the specified release.

-s|-sh|-show (create_time)

Shows the time that the specified release was created.

-s|-sh|-show (desc|description)

Shows the description entered for the specified release.

-s|-sh|-show (groups)

Shows the groups set up for the specified release.

-s|-sh|-show (included_releases)

Shows any releases to be included in the specified release.

-s|-sh|-show (manager)

Shows the manager for the specified release.

-s|-sh|-show (modifiable_in)

Shows which databases the specified release is modifiable in.

-s|-sh|-show (phase)

Shows the phase for the specified release.

-s|-sh|-show (phase_log)

Shows phase log attributes for the specified release.

Showing the process rules for a release

This subcommand shows the process rules for one or more releases.

About this task

```
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

Specifies to use a column header in the output format. See -ch|-column_header 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_header for details.

-nf|-noformat

Specifies not to use column alignment. See -nf|-noformat for details.

-ns|-no_sort

Specifies not to sort the command output. 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.

-show process_rules|pr|prs

Shows the current valid process rules for each of the specified arguments. The output 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 process rules already in use. The output 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. The output shows the available process rules for a release excluding valid process rules in use. The output is numbered by default for each argument and sets the selection set.

-u|-unnumbered

Suppresses automatic numbering of the command output. See -u|-unnumbered for details.

Showing release information

You can view release information in various formats.

About this task

```
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

Specifies to use a column header in the output format. See -ch|-column_header 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_header for details.

-nf|-noformat

Specifies not to use column alignment. See -nf|-noformat for details.

-sep|-separator *separator*

Specifies a different separator character. See -sep|-separator for details.

-v|-verbose

Shows the verbose information format.

Example

- 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
```

save_offline_and_delete command

The Save Offline and Delete (SOAD) command deletes objects from a database by using a scope.

The scope determines what objects are considered for deletion, what associated objects to delete, and rules for what objects not to delete. Databases ship 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.

See Rational® Synergy Save Offline and Delete Scopes for details about SOAD scopes and how SOAD evaluates scopes.

You can delete objects from a 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 examples show 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 a transfer package 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:

- Preview a delete by using a scope. The preview shows what objects would be selected for deletion without 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 setting allows you to perform the deletion without seeing a preview of the results. The advantage is speed, but you must be certain that previewing the objects is unnecessary. The disadvantage is that you are not able to verify the objects to be deleted.

To save objects offline, the current database must be initialized for DCM and a DCM license must be available.

The save_offline_and_delete command supports these subcommands:

- Creating a preview object list for deletion
- Deleting using a preview object list
- Deleting using a scope

Creating a preview object list for deletion

Use this subcommand to perform a preview of objects to be deleted by a save offline and delete scope. Then, use the subcommand to create 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. The list is also cleared when you delete an object from 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.

About this task

```
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...]
```

argument...

Specifies the arguments required for the specified scope. For example, the scope "`My working projects and products for a specified release`" requires one argument, which is a release name. If a scope requires multiple arguments, the arguments must be specified in the order defined in the scope definition.

-ch|-column_header

Specifies to use a column header in the output format. See -ch|-column_header 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_header for details.

-nf|-noformat

Specifies not to use column alignment. See -nf|-noformat for details.

-ns|-no_sort

Specifies not to sort the command output. 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.

-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 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"
```

Deleting using a preview object list

This subcommand deletes the objects in the preview object list that were 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.

About this task

```
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 Save Offline 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. The date 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 save offline and delete preview.

```
ccm soad -delete
```

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.

Note: Performing a preview before deleting the objects means that you can check the objects to be deleted before deleting them. Knowing the objects to be deleted is essential if you are working in the *ccm_admin* role. This command does not perform a preview; you do not know the objects to be deleted until the command has started.

About this task

```
ccm soad|save_offline_and_delete -delete -scope scope_name
        [-so|-save_offline] [-path soad_path]
        [-pn|-package_name package_name] [-v|-verbose]
        [argument...]
```

argument...

Specifies the arguments required for the specified scope. For example, the scope "`My working projects and products for a specified release`" requires one argument, which is a release name. If a scope requires multiple arguments, the arguments must be specified in the order defined in the scope definition.

-path *soad_path*

Specifies the path to the Save Offline 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 Save Offline 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. The date 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.

-v|-verbose

Generates messages detailing why objects are being included in or excluded from the list.

Example

- Delete objects found by the Baselines marked for deletion and related projects and products scope without performing a preview.

```
ccm soad -delete -scope "Baselines marked for deletion and related projects and products"
```

select command

The ccm select command provides the ability to manipulate the query selection set in various ways. The command supports a stack of selection sets that might be used to save and restore selection sets.

For example, if you want a script to use queries but preserve the current selection set, you can push the current selection set onto the selection set stack, execute a query, and then pop the selection set stack to restore the selection set.

The select command supports the following subcommands:

- Adding specified objects to current selection set
- Clearing the current selection set
- Clearing the selection set stack
- Counting number of objects in current query selection set
- Counting number of selection sets in selection set stack
- Setting current selection set to intersection of its previous objects and popped selection set
- Setting current selection set to difference between its previous objects and popped selection set
- Popping selection set stack to set current selection set to the difference between popped selection set and previous selection set
- Popping the last pushed selection set from selection set stack
- Pushing current selection set onto selection set stack
- Removing specified objects from current selection set
- Setting current selection set to specified objects
- Showing a selection set on the selection set stack
- Setting current selection set to symmetric difference between previous objects and popped selection set
- Merging popped selection set with current selection set

Adding specified objects to current selection set

The `ccm select -add` command adds specified objects to the current query selection set if they are not already present.

About this task

```
ccm select -add object_spec...
```

object_spec

Specifies the object to be added. See Object specification for details.

Clearing the current selection set

The `ccm select -clear` command clears the current selection set.

About this task

```
ccm select -clear
```

Clearing the selection set stack

The `ccm select -clear -stack` command clears the current selection set stack.

About this task

```
ccm select -clear -stack
```

Counting number of objects in current query selection set

The `ccm select -count` command shows the number of objects in the current query selection set.

About this task

```
ccm select -count
```

Counting number of selection sets in selection set stack

The `ccm select -count -stack` command shows the number of select sets in the selection set stack.

About this task

```
ccm select -count -stack
```

Setting current selection set to intersection of its previous objects and popped selection set

The `ccm select -intersection` command pops the last pushed entry on the selection set. This action sets the current selection set to the intersection of its previous objects and the popped selection set. For example, if the top of the stack contains objects {a,b,c} and the current selection set contains {b,c,d}, then the current selection set after the command is {b,c,}.

About this task

```
ccm select -int|-intersection
```

Setting current selection set to difference between its previous objects and popped selection set

The `ccm select -not_in_stack` command pops the selection set stack, setting the current selection set to the difference between its previous objects and the popped selection set. That is, the new selection set contains objects that were in the previous selection set but not in the popped selection set. For example, if the top of the stack contains objects {a,b,c} and the current selection set contains {b,c,d}, then the current selection set after the command is {d}.

About this task

```
ccm select -not_in_stack|-not_in
```

Popping selection set stack to set current selection set to the difference between popped selection set and previous selection set

The `ccm select -only_in_stack` command pops the selection set stack, setting the current selection set to the difference between the popped selection set and the previous selection set. That is, the new selection set contains objects that were in the popped selection set but not in the previous selection set. For example, if the top of the stack contains objects {a,b,c} and the current selection set contains {b,c,d}, then the current selection set after the command is {a}.

About this task

```
ccm select -only_in_stack
```

Popping the last pushed selection set from selection set stack

The `ccm select -pop` command pops the last pushed entry on the selection set and places it in the current selection set.

About this task

```
ccm select -pop
```

Pushing current selection set onto selection set stack

The `ccm select -push` command pushes the current query selection set onto the selection set stack.

About this task

```
ccm select -push
```

Removing specified objects from current selection set

The `ccm select -remove` command removes the specified objects from the current query selection set. If an object is not in the selection set, it is ignored and no error is reported.

About this task

```
ccm select -remove object_spec...
```

object_spec

Specifies the object to be removed. See Object specification for details.

Setting current selection set to specified objects

The `ccm select -set` command sets the current selection set to the specified objects.

About this task

```
ccm select -set object_spec...
```

object_spec

Specifies the object to set in the selection set. See Object specification for details.

Showing the current selection set

The `ccm select -show` command shows the objects that are in the current selection set.

About this task

```
ccm select -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] [-u|-unnumbered]
```

-ch|-column_header

Specifies to use a column header in the output format. See -ch|-column_header 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_header for details.

-nf|-noformat

Specifies not to use column alignment. See -nf|-noformat for details.

-ns|-no_sort

Specifies not to sort the output. 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 output. The output is not numbered. See -u|-unnumbered for details.

Showing a selection set on the selection set stack

The `ccm select -show -stack` command shows the objects that are contained in a selection set in the selection set stack.

About this task

```
ccm select -s|-sh|-show -stack select_index
        [-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_header 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_header for details.

-nf|-noformat

Specifies not to use column alignment. See -nf|-noformat for details.

-ns|-no_sort

Specifies not to sort the output. 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.

-stack *select_index*

Specifies to show the index entry of the selection set stack.

Setting current selection set to symmetric difference between previous objects and popped selection set

The `ccm select -show -stack` command pops the selection set stack, setting the current selection set to the symmetric difference between its previous objects and the popped selection set. That is, the new selection set contains objects that were in either the previous selection set, or in the popped selection set, but not both. For example, if the top of the stack contains objects {a,b,c} and the current selection set contains {b,c,d}, the current selection set after the command is {d,a}.

About this task

```
ccm select -symdiff
```

Merging popped selection set with current selection set

The `ccm select -union` command pops the selection set stack, merging the popped selection set with the current selection set. For example, if the top of the stack contains objects {a,b,c} and the current selection set contains {b,c,d}, the current selection set after the command is {b,c,d,a}.

About this task

```
ccm select -un|-union
```

set command

You can set behaviors in Rational® Synergy. An option represents a control over the behavior of specific Rational Synergy operations. Some options apply to the current CLI session only 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.

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.

The `set` command supports these subcommands:

- Setting an option
- Showing an option
- Showing options

Setting an option

You can set an option value. The option value must be valid for the option you are setting. The ccm set command sets values for the current CLI session only. (The setting is not persistent.)

For example, to set your role to *developer*, use ccm set role developer.

About this task

```
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.

The CLI supports setting tools for the edit, view, compare, and merge actions dynamically for any type defined in the database. The following examples show how to use the ccm set command:

- `ccm set view_command.`*`type`*

Sets the tool to view objects of type *`type`*. This setting maps to the same preference as `ccm.cli.tools.view.`*`type`*`.windows` on Windows, and `ccm.cli.tools.view.`*`type`*`.unix` on UNIX.

- `ccm set edit_command.`*`type`*

Sets the tool to edit objects of type *type*. This setting maps to the same preference as `ccm.cli.tools.edit.`*type*`.windows` on Windows, and `ccm.cli.tools.edit.`*type*`.unix` on UNIX.

- `ccm set compare_command.`*type*

Sets the tool to compare objects of type *type*. If the objects have different types and are compatible, the first object type is used. This setting maps to the same preference as `ccm.cli.tools.compare.`*type*`.windows` on Windows, and `ccm.cli.tools.compare.`*type*`.unix` on UNIX.

- `ccm set merge_command.`*type*

Sets the tool to merge objects of type *type*. If the objects have different types and are compatible, the first object type is used. This setting maps to the same preference as `ccm.cli.tools.merge.`*type*`.windows` on Windows, and `ccm.cli.tools.merge.`*type*`.unix` on UNIX.

The following existing command options that set the tools for a CLI session are still valid.

- `ccm set text_editor`

This command option is equivalent to the new option, `ccm set edit_command.ascii`.

- `ccm set text_viewer`

This command option is equivalent to the new option, `ccm set view_command.ascii`.

- `ccm set cli_compare_cmd`

This command option is equivalent to the new option `ccm set compare_command.ascii`. Other compare command options, such as `cli_dir_compare_cmd`, `cli_proj_compare_cmd`, and `cli_symlink_compare_cmd`, are still available.

- `ccm set cli_merge_cmd`

This command option is equivalent to the new option, `ccm set merge_command.ascii`.

Example

- Set your session to view `java` objects using the EditPlus viewer.

```
ccm set view_command.java “\"D:/Program Files/EditPlus/editplus.exe\" %file”
```

- Set your role to *developer*.

```
ccm set role developer
```

- Display your current role.

```
ccm set role
developer
```

Showing an option

You can show the value of an option. Alternatively, if you want to show all options, do not specify an option.

About this task

```
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
```

Showing options

This subcommand shows listable options.

About this task

```
ccm set
```

Example

- Show all listable options.

```
ccm set
```

soad_scope command

The `soad_scope` command edits, creates, modifies, and deletes scopes used to save objects offline and to delete objects.

The `soad_scope` command supports these subcommands:

- Creating a scope
- Deleting a scope
- Listing exclusion rules
- Listing expansion rules
- Listing scopes
- Modifying a scope
- Showing a scope

Creating a scope

This subcommand creates a save offline and delete scope.

Note: When you create a 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 you want to keep.

See Rational® Synergy Save Offline and Delete Scopes for details about SOAD scopes and how SOAD evaluates scopes.

Before you begin

You can create a scope when working in the *ccm_admin* role.

About this task

```
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, 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 can also be removed.

-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'`). 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 this query expression:

```
has_attr('requirements')
```

SOAD adds this 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 this 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 `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 Save Offline 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 this 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_is_my_test_scope.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**.

Deleting a scope

This subcommand deletes a save offline and delete scope.

See Rational® Synergy Save Offline and Delete Scopes for details about SOAD scopes and how SOAD evaluates scopes.

Before you begin

You can delete a scope when working in the *ccm_admin* role.

About this task

```
ccm soad_scope|save_offline_and_delete_scope -d|-delete scope_name
```

scope_name

Specifies the name of the scope to delete.

Listing exclusion rules

This subcommand lists the names of the valid exclusion rules that are available for use by a save offline and delete scope.

About this task

```
ccm soad_scope|save_offline_and_delete_scope -l|-list
              -exclude|-exclusion_rules
```

-exclude|-exclusion_rules

Lists the exclusion rules.

Listing expansion rules

This subcommand lists the names of the valid expansion rules that are available for use by a save offline and delete scope.

About this task

```
ccm soad_scope|save_offline_and_delete_scope -l|-list
              -expand|-expansion_rules
```

-expand|-expansion_rules

Lists the expansion rules.

Listing scopes

This command lists the names of the defined save offline and delete scopes.

See Rational® Synergy Save Offline and Delete Scopes for details about SOAD scopes and how SOAD evaluates scopes.

About this task

```
ccm soad_scope|save_offline_and_delete_scope -l|-list -scope
```

-scope

Specifies the scope that you want to list.

Modifying a scope

This subcommand changes the properties of a save offline and delete scope.

See Rational® Synergy Save Offline and Delete Scopes for details about SOAD scopes and how SOAD evaluates scopes.

Before you begin

You can edit a scope when working in the *ccm_admin* role.

About this task

```
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, 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 can also be removed.

-expand|-expansion_rules *expansion_rules*

Specifies one or more expansion rules. Expansion rules add related objects to the initial object list.

For example, your query retrieves all objects for a specified release, with the release name as the first parameter (`release='%1'`). Expand the scope by adding expansion rules to include the folder and tasks for the project, the tasks for the folders, and the objects for the tasks.

-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 this query expression:

```
has_attr('requirements')
```

SOAD adds the 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 `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 Save Offline 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 this 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**.

Showing a scope

This subcommand shows details for save offline and delete scopes.

See Rational® Synergy Save Offline and Delete Scopes for details about SOAD scopes and how SOAD evaluates scopes.

About this task

```
ccm soad_scope|save_offline_and_delete_scope -s|-sh|-show scope_name...
```

scope_name...

Shows all scopes with these details.

- Roles
- Parameter labels
- Object
- Query
- Expansion rules
- Exclusion rules

- Exclusion query
- Package name

You can specify to show multiple scope names.

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"
```

show_servers command

Use this command to view the databases and associated Rational® Synergy servers known to the router.

The command output shows the server URL to be used by default when starting a GUI or CLI session without specifying the server URL.

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 are not required to run a CLI session to use this command.

Use this command to see which databases are served by Rational Synergy servers. Additionally, you can specify a server URL to start a CLI session.

About this task

```
ccm show_servers
```

start command

The start command begins a Rational® Synergy CLI session.

After the session starts, the Rational Synergy address (`CCM_ADDR`), displays. This unique identifier for the 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 to run your Rational Synergy commands. If you do not set `CCM_ADDR`, a default address is used. This default address is read from a `.ccm_`*`hostname`*`_7.2_addr` file. *`hostname`* is the name of the client machine. This file is generated if you start a CLI session without the `-m` (multiple sessions) option.

Single user admin mode

Some operations, such as initializing DCM and renaming a release, require the user performing the operation to have exclusive access to the database. Gain exclusive access by starting a CLI session with the `-single|-single_user` *`reason`* option. This option starts a session and sets the database to single user admin mode. You must be in the *ccm_admin* role to use single user admin mode.

After you start a session in single user admin mode, other users cannot start new client sessions or perform operations in the database using existing clients until the single user admin mode session is stopped with a `ccm stop` command.

If you have a session in single user admin mode, you can start other clients with the same user name. However, only one back-end session is available to process requests. If several clients all request operations requiring server processing, the requests are executed serially, one at a time.

The `-reset_single_user` option allows an administrator to release the single user admin mode if the session owning that mode does not exit with `ccm stop`, or if the user has unintentionally left that session running.

CAUTION:
Operations that require single user admin mode assume that once granted, the operation is started. If you use the `-reset_single_user` option before that operation has completed, users might create data that is inconsistent with the administrator changes being made. For example, during a DCM initialization, if users create data, the data might be incorrect for subsequent usage in a DCM initialized database. The incorrect data is not detected or prevented.

Before you reset single user admin mode, check with the original user who was granted that mode. Determine if they are finished using the mode. Reset single user admin mode only if the operations have completed.

The `start` command supports these subcommands:

- Starting a CLI session
- Starting the Rational Synergy GUI from the command line

Starting a CLI session

The start command begins a Rational® Synergy CLI session.

About this task

```
ccm start [-nogui][-q] -d database_path -s server_url [-m]
          [-pw password] | [-pwprompt]) [-fpw password_file]
          [-n username] [-r initial_role]
          [-single|-single_user reason] [-reset_single|-reset_single_user]
```

-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 to use multiple sessions. You must set the `CCM_ADDR` environment variable to the session to be used. 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 setting is used as the default address.

-n *user_name*

On Windows, specifies a Rational Synergy user name to use.

On UNIX, the user name is the same as the operating system user name and you do not need to specify `-n`. If you specify `-n`, the user name must match the current user.

-pw *password*

This option is available to Windows users only. (On UNIX, the user name is taken from the operating system user name.)

Specifies the password for the Rational 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/.ccmwc` is used on UNIX, and `%HOMEPATH%\.ccmrc` is used on Windows. (The default password is defined by using the `ccm set_password` command. See Setting up ccm set_password for details.)

-pwprompt

Prompts for a password to use to authenticate the user.

-q

Starts a session in Quiet mode. When you use this option, Rational Synergy shows the `CCM_ADDR` for the CLI session. If you do not use this option, the startup shows additional information, such as a copyright notice, before `CCM_ADDR`.

-reset_single|-reset_single_user

Specifies to start a new session by resetting single user admin mode. See Single user admin mode for details.

-r *initial_role*

Specifies the initial role to be used. If you do not specify a role, the default role is *developer*. To change your initial role, see `ccm.cli.start.defaultrole` in Rational Synergy Default Settings.

-s *server_url*

Specifies the URL of the Rational Synergy server to use. Specifying the server URL is optional. If you do not specify a server URL, a default URL registered with the Rational Synergy router for the database is used. View the registered databases and servers by using the `ccm show_servers` command.

If the default server URL does not resolve from a client machine, use the `-s` *`server_url`* option to specify a server URL with a name that resolves.

-single|-single_user *reason*

Specifies to start the session in single user admin mode. See Single user admin mode for details.

Example

- Start a Rational Synergy CLI session on a specified database.

```
ccm start -d /data/db1 -n bob -pw ****
```

- 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
```

Attention:

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 displayed address, for example:

```
set CCM_ADDR=prefect.cwi.com:1368
```

This setting causes 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. These settings distinguish your *ccm_root* session from sessions where other users are running as *ccm_root*.

Starting the Rational Synergy GUI from the command line

Use the `cmsynergy` command to start a session. The `cmsynergy` command opens the Start dialog box with any specified settings.

About this task

```
cmsynergy [-user | -u user_name] [-password | -pw password]
          [-host | -h engine_host] [-database | -d database_path]
          [-server | -s server_url]
```

-h | -host *engine_host*

Specifies the engine host for this session. This option and `-s` *`server_url`*are mutually exclusive.

-d | -database *database_path*

Specifies the database path for the Rational® Synergy database to connect to this session.

-pw | -password *password*

Specifies the user password for the session.

-s | server *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://` This option and `-host` *`engine_host`*are mutually exclusive.

See Administering the CCM server for a detailed discussion about the CCM server.

-u | -user *user_name*

This option, available for Windows users only, specifies the user name for the session.

Example

- Start Rational Synergy.

```
> cmsynergy -u linda -pw jupiter -h milkyway -s http://unixXYZ:8400
```

- Start Rational Synergy using three of the values (`user_name`, `engine_host`, and `server`).

```
> cmsynergy -u linda -h milkyway -s http://unixXYZ:8400
```

- Start a Rational Synergy session using a server URL.

```
> cmsynergy -u bob -pw **** -d /data/db1 -s http://unixXYZ:8400
```

status command

Use this command shows information about user CLI or GUI sessions matching specified criteria.

You can run the command in two modes: sessionless and in a session.

In **sessionless** mode, you can run the command without starting a session or with CCM_ADDR not set to the address of an existing session. By default, the command shows the sessions that belong to a Rational® Synergy user whose name is the operating system user. Use the `-u|-user` *`username`* option to specify a Rational Synergy user name if the operating system user is not your Rational Synergy user name. In sessionless mode, the project associated with the current working directory cannot be identified.

In **session** mode, run the command in an existing CLI session. By default, the command shows the sessions that are owned by the Rational Synergy user that started the session, and identifies the current session. The command attempts to identify the project associated with the current working directory.

The `status` command supports the “Showing sessions” subcommand.

Showing sessions

This subcommand shows information about user CLI or GUI sessions matching specified criteria.

About this task

```
ccm status -address ([-cli] | [-gui]) [-db|-database database_pattern]
           [-u|-user username]
ccm status ([-cli] | [-gui]) [-db|-database database_pattern]
           [-u|-user username] [-f|-format format] [-nf|-noformat]
           ([-ch|-column_header] | [-nch|-nocolumn_header])
           [-sep|-separator separator]
           ([-sby|-sortby sortspec] | [-ns|-nosort|-no_sort])
           [-gby|-groupby groupformat]
```

-address

Specifies to show the session address for each matching session only. If a CLI session is running on the current client machine, you might use this option as follows: `export CCM_ADDR='ccm status -address -cli'`

-cli

Specifies to show CLI sessions only.

-ch|-column_header

Specifies to use a column header in the output format. See -ch|-column_header for details.

-db|-database *database_pattern*

Specifies to show sessions running on databases that match the *`database_pattern`* only.

If the *database_pattern* does not contain any of the characters "/", "\", "*", "?", "[" or "]", the pattern is used to match the end of the full database path (without any trailing "/db" or "\db"). For example, for a database path of `/ccmdb/example`, a *database_pattern* of *example* is a match.

Otherwise, the supplied *database_pattern* is used as a *shell matching pattern*. A shell matching pattern allows the use of "*" to match any (possibly zero length) substring, and "?" to match any single character. With shell match patterns, "*" matches the corresponding part of the path only. For example, for a database path of `/ccmdb/example`, a shell pattern of */*/example* is a match. A shell pattern of *example* is not a match.

-f|-format *format*

Specifies the command output format. See -f|-format for details.

The following keywords are supported.

Table 1. Supported keywords for -format option

Keyword	Description
address	The address of the session.
current_session	Has a boolean value of TRUE if the session is the current session executing the command.
database	The path of the database used by the session.
session	The type of session such as "Command Interface" or "Graphical Interface".
username	The name of the Synergy user who started the session.

-gui

Specifies to show GUI sessions only.

-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_header for details.

-nf|-noformat

Specifies not to use column alignment. See -nf|-noformat for details.

-ns|-no_sort

Specifies not to sort the command output. 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 output. See -u|-unnumbered for details.

stop command

The `stop` command ends a session.

The `stop` command supports the "Stopping a CLI session" subcommand.

Stopping a CLI session

Use the stop command to stop a CLI session.

About this task

```
ccm stop|quit
```

Example

Stop the current CLI session.

```
ccm stop
```

sync command

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. A sync operation writes out the database files to the work area.

Manually sync (force a sync) your work area in these 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 from the CLI or the GUI, your work area path is updated to the new location. If another application is using the old work area path, the move fails. 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:

- Reconcile the work area that you are currently using (either local copies or symbolic links).
- Delete the work area objects from the file system.
- Set your work area path and options.
- Start a new session using the client option of choice (either local copies or symbolic links).
- 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.

  Note: If you stop the sync, an error message displays stating that errors might occur in your work area. The errors do 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.

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.

Before you begin

Only a build manager or a user in the *ccm_admin* role can sync a non-writable project.

About this task

```
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 setting 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 synchronized 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. 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.

Example

- 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
```

task command

You can use the task command to create, modify, and delete tasks, and to perform the following operations:

- 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 parallels for a task
- Showing task information
- Transitioning a task to a different state

Assigning a task

You can assign the specified task to the specified resolver. To assign a task, you must have task assignment privileges and be able to modify the task.

About this task

```
ccm task -as|-assign -t|-to resolver [-q|-quiet] task_spec...
```

-t|-to *resolver*

Specifies the resolver to assign the tasks to. 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
```

Associating a task with objects, tasks, or change requests

You can associate 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. The change request must be in a state that allows task association.

The task requirements for associating a task with a task to be fixed are as follows:

- 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.

About this 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 requests associated with the task. 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.

Example

- 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
```

- Create a relationship between the fix task (**19**) and the task to be fixed (**4**).

```
ccm task -relate 19 -fixes 4
```

Completing a task

When you complete a task, the objects associated with the task are checked in to a non-modifiable state. The task is then moved to the *completed* state. The resolver of a task or an administrator can perform this operation.

About this task

```
ccm task -ci|-checkin|-complete [-time|-time_actual task_duration]
        [-parallels|-check_parallels (none | (i|info|information) |
        (check|enforce))]
        [-c|-comment comment_string] [-ce|-commentedit]
        [-cf|-commentfile file_path] [-commentreplace]
        (task_spec|(current|default))...
```

-c|-comment *comment*

Specifies to append the specified comment 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 to start the default text editor to compose and edit the comment. 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 to use the contents of the specified file for the comment. If you specified `-comment`, it is appended to that comment. You can use this option with the `-commentedit` option.

-commentreplace

Specifies to replace the comment.

-parallels|-check_parallels (none | (i|info|information) | (check|enforce))

Specifies the parallel checking to be performed when completing the task.

- `none` means no checking for parallels on the task.
- `i|info|information` means check and show any parallels on the task.
- `check|enforce` means check for parallels on the task. If parallels exist, do not complete the task and return with a non-zero exit status.

-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 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.

Example

- 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
```

Copying a task

You can create tasks by copying specified tasks. Copy a task to apply a task that you fixed for one 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 are also associated with the corresponding copied task.

About this 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 are creating. If you do not specify the date estimate, it is set to the date estimate of the task you are copying. The *date_estimate* must be a valid date.

-def|-default|-current

Specifies to make the first task created from the task copy 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 copied task are not associated with the task created by the copy.

-p|-priority *priority*

Specifies the priority of the new tasks. If you do not specify the priority, it is set to the priority of the task you are 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 do not specify the platform, it is set to the platform of the task you are 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 do not specify a change request and the task you are 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 do not 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 do not specify a release, it is set to the release of the task you are 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 are 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 do not specify a subsystem, it is set to the subsystem of the task you are copying. The subsystem must be a valid task subsystem.

-s|-synopsis *synopsis*

Specifies the synopsis of the task you are 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 do not specify a time estimate, it is set to the time estimate of the task you are 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 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.
```

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, but not assigned to a person.

About this task

```
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 to set the task you are creating as 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.

-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 to create the task 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 are 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 are creating. The synopsis can be any string without newline characters.

-time|-time_estimate *time_estimate*

Specifies the estimated time to complete the task. 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

You can break the association between two objects. The break disassociates 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. To disassociate the task from a change request, you must be able to modify both the change request and the task.

About this 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 requests to associate the task 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 to disassociate from the specified task.

-prob|-problem|-change_request

Specifies the change request to be associated with the task.

Example

- Disassociate task **35** from object version `MAIN.C-3:csrc:1`.

```
ccm task -d 34 -obj MAIN.C-3:csrc:1
```

- Disassociate task **10668** from change request **6569**.

```
ccm task -d 10668 -change_request 6569
```

- Break a relationship between the fix task (**25**) and the task it fixed (**12**).

```
ccm task -unrelate 25 -fixes 12
```

Fixing a task

You can create a task and establish a relationship between it and the task to be fixed. The relationship detects when a project is using one task without the other, which is called a *conflict*.

To fix a task with an existing task, see Associating a task with objects, tasks, or change requests. To fix or enhance a fix task, create another 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 fix only one task.

About this 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...
```

-def|-default|-current

Specifies to set the fix task you are creating as 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.

-exclude

Specifies to move the tasks being fixed to the *excluded* state. Use this option to exclude them from being automatically included in future builds.

-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 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 do not 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

Specifies to create the task 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 are fixing. 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 are creating. The synopsis can be any string without newline characters.

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*

Specifies the estimated time to complete the tasks. 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 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.
```

Modifying a task

You can modify many attributes of a task, such as the synopsis, the platform, the priority, and the resolver. You must have task assignment privileges and be able to modify the task.

About this 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*

Specifies the estimated completion date of the tasks you are creating. If you do not specify the date estimate, it is set to the date estimate of the task you are copying. The *date_estimate* must be a valid date.

-desc|-description *description*

Specifies a single-line description. The description cannot contain new line characters.

-desc_edit|-description_edit

Specifies to start the default text editor so you can edit or compose a multiline description.

-desc_file|-description_file *file_path*

Specifies a path to a file containing a multiline description.

-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 new line characters.

-date|-date_estimate *date_estimate*

Specifies the estimated completion date of the tasks you are creating. If you do not specify the date estimate, it is set to the date estimate of the task you are 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 new line characters.

-desc_edit|-description_edit

Specifies to start the default text editor so you can edit or compose a multiline description.

-desc_file|-description_file *file_path*

Specifies a path to a file containing a multiline description.

-rel|-release *release_spec*

Specifies the release for the created tasks. If you do not specify a release, it is set to the release of the task you are 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 modifying. 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 do not specify a subsystem, it is set to the subsystem of the task you are copying. The subsystem must be a valid task subsystem.

-s|-synopsis *synopsis*

Specifies the synopsis of the task you are copying. The synopsis can be any string without new line characters.

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*

Specifies the estimated duration or effort to complete the created tasks. If you do not specify a time estimate, it is set to the time estimate of the task you are 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

Change the release for task **68** to `4.1`.

```
ccm task -modify -release 4.1 68
```

Querying for tasks

You can query 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.

About this task

```
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] 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
```

-ch|-column_header

Specifies to use a column header in the output format. See -ch|-column_header 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. The task list is determined by gathering all tasks for all objects in the hierarchy, and for all their ancestors, and then 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. 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 -nch|-nocolumn_header for details.

-nf|-noformat

Specifies not to use column alignment. See -nf|-noformat for details.

-ns|-no_sort

Specifies not to sort the output for the command. 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 Showing a project purpose for a detailed description of purposes.

This option typically applies to queries for component tasks that are specified with one of these 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 includes 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`

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 that you can resolve and that are modifiable or 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 output (that is, the output is not numbered). 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
```

Setting or clearing the current task

You can set or clear the current task. By default, if you perform an operation that associates changes with a task, but you do not specify a task, Rational® Synergy uses the current task.

About this 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 do not specify arguments, the command shows the current task but does not set it.

Example

- 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.
```

Showing a task property

You can show a specified task property.

About this task

```
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 tasks whose properties you want to view. See Task specification for details.

Showing the associated objects, change requests, and tasks for a task

You can show associated objects, change requests for the specified tasks, tasks fixed by specified tasks, and tasks that fix specified tasks. The query selection displays the associated objects.

About this task

```
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

Specifies to use a column header in the output format. See -ch|-column_header 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_header for details.

-nf|-noformat

Specifies not to use column alignment. See -nf|-noformat for details.

-ns|-nosort|-no_sort

Specifies not to sort the command output. 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.

task_spec...

Specifies the tasks whose properties you want to view. See Task specification for details.

-u|-unnumbered

Suppresses automatic numbering of the output (that is, the output is not numbered). See -u|-unnumbered for details.

Example

- 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
```

Showing parallels for a task

You can show parallel versions for the objects associated with a specified task.

About this task

```
ccm task -s|-sh|-show (parallel|parallels)
         [-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

Specifies to use a column header in the output format. See -ch|-column_header 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_header for details.

-nf|-noformat

Specifies not to use column alignment. See -nf|-noformat for details.

-ns|-nosort|-no_sort

Specifies not to sort the command output. See -ns|-nosort for details.

-sby|-sortby *sortspec*

Specifies how to sort the command output. See -sby|-sortby for details.

-sep|-separator *separator*

Specifies a different separator character. See -sep|-separator for details.

task_spec...

Specifies the task whose parallels you want to show. See Task specification for details.

-u|-unnumbered

Suppresses automatic numbering of the command output. See -u|-unnumbered for details.

Showing task information

You can show information, such as task synopsis, description, state, resolver, and more, about the specified tasks.

About this task

```
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

Specifies to use a column header in the output format. See -ch|-column_header 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 -nch|-nocolumn_header 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.

task_spec...

Specifies the tasks 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.

Example

- 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 Form2) high 41  Wrong
window receives message3) high 22 Saving a file takes forever 4) low 39
Button icons are rather obscure5) 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 jane2) b.txt-1.1:ascii:1 integrate  jane
```

Transitioning a task to a different state

You can change the state of a task if you have task assignment privileges and are able to modify the task.

About this task

```
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...
```


```
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*

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 multiline description.

-desc_file|-description_file *file_path*

Specifies a path to a file containing a multiline description.

task_state

Specifies the state that you want to move the task to.

-r|-resolver *resolver*

Specifies to set the resolver of the specified tasks. This option can only be used when moving a task to the *task_assigned* state. The `resolver` must be a valid task resolver.

task_spec...

Specifies the tasks you want to move. See Task specification for details.

Example

- Transition a task from the *completed* state to *excluded*.

  ```
  ccm task -state excluded 94
  ```

- Move a task from the *excluded* state to *completed*.

  ```
  ccm task -state completed 94
  ```

typedef command

Rational® Synergy supports type-dependent behaviors. For example, you can allow parallel versions for one type of object but not another. Or you can use a selected editor tool for objects of a specific type. When you use an appropriate type for your files, you can define the appropriate behaviors for that type. This method helps describe the purpose of objects of that type.

A type definition is an object that defines and represents the properties of a type. Objects of that type inherit certain properties and behaviors from the corresponding type definition.

Types can be categorized into the following groups:

- project -

  A container for directories, files, and symlinks.

- non project-member types

  These types cannot be used as members of a project and cannot display in work areas. Examples of such types are baseline, task, folder, releasedef, process_rule. These types are provided as *base model* types.

- dir

  An organizational type that corresponds to a directory and owns zero, one, or many directory entries. Within a project, each directory entry for a directory typically has a corresponding child object that matches the directory entry.

- file and symlink types

  These types can be used as members of a project and have a work area representation. Examples are dir, ascii, binary, csrc, and java.

Type definitions inherit some properties from a *super_type*. Type definitions have an inheritance hierarchy. Most file-based types use a *super_type* of either ascii or binary.

- The binary type is a predefined base model type that represents files whose contents are binary and are always represented "as is" without any keyword expansion or end-of-line translation.
- The ascii type is a predefined base model type that represents files whose contents are usually 7-bit ASCII. With files of type or subtype ascii, Rational Synergy looks at the file contents to determine whether they contain certain binary characters or a significant proportion of non 7-bit bytes. If they do, the system handles the source code for that file as if it were a binary type. That is, keyword expansion and end-of-line translation are not performed on such files.

Use the ccm typedef command to create file-based types and to modify dir, symlink, and file-based types. Only users that who are assigned the role type_developer or ccm_admin can create or modify type definitions.

Each type definition has a number of properties that describe the type:
Table 1. Type definition properties

Property	Description
name	The name of the type. For example, "java"
description	A one-line description of the meaning or purpose of the type.
super_type	The parent type for inheritance. It is invalid to define types with circular inheritance. Typical values are "ascii" or "binary".
source template	Defines a template for the initial source contents of the file when a new object of that type is created. The template can contain Rational Synergy keywords that are expanded on creation.
range keyword expand	If a file has ascii contents, keyword expansion is performed on check out and, optionally, on check in. This property defines how many bytes at the start of the file are processed for keyword expansion. The values might be any of the following: • -1 signifies that keyword expansion is enabled and the range is infinite. The entire file is processed. • 0 signifies that keyword expansion is disabled. • *N*>0 signifies that keyword expansion of the first *N* bytes is enabled.
parallel check out	Specifies when parallel objects of this type are allowed on check out. A value of FALSE means that users cannot check out parallel versions for objects of this type. A value of TRUE means that parallels might be allowed. The release definition for the associated release of the object determines whether parallels are allowed.
parallel check in	Specifies when parallel objects of this type are allowed on check in. A value of FALSE means that users cannot check in parallel versions for objects of this type. A value of TRUE means that parallels might be allowed. The release definition for the associated release of the object determines whether parallels are allowed.
active	Specifies whether new objects of this type can be created or checked in. A value of FALSE is used to retire a type definition. Existing objects of that type remain unchanged. However, users must use the *change type* operation if they want to check in new versions of such objects. In this way, over time, users gradually move to using some alternative type.

Property	Description
execute permission	Specifies whether on a UNIX work area updated by a UNIX client, the object has the UNIX 'x' execute permission.
windows ignore	On a Windows client, specifies whether reconcile and migrate should ignore files of this type.
unix ignore	On a UNIX client, specifies whether reconcile and migrate should ignore files of this type.
windows match	Specifies zero, one, or many regular expressions are used to match files on Windows clients. During create, reconcile and migrate operations on Windows, files that match any of these expressions use this type as the default. See Mapping rules for more details.
unix match	Specifies zero, one, or many regular expressions are used to match files on UNIX clients. During create, reconcile and migrate operations on UNIX, files that match any of these expressions use this type as the default. See Mapping rules for more details.

The typedef command supports the following subcommands.

- Creating a type
- Exporting a type
- Importing a type
- Listing types
- Modifying a type
- Showing information for a type
- Showing a property for a type

Creating a type

The `ccm typedef -create` command creates a type definition. The type must not exist, and you must have the `type_developer` or `ccm_admin` role available.

About this task

```
ccm typedef -c|-create -desc|-description description
        -st|-super_type super_type
        [-stf|-source_template_file template_file]
        [-rke|-range_keyword_expand (range_integer|inherit)]
        [-pco|-parallel_check_out (inherit | true | false)]
        [-pci|-parallel_check_in (inherit | true | false)]
        ([-active] | [-inactive])
        ([-x|-execute_permission] | [-nox|-noexecute_permission])
        ([-wi|-win_ignore] | [-nowi|-nowin_ignore])
        ([-ui|-unix_ignore] | [-noui|-nounix_ignore])
        [(-match match_expression)...]
        [(-wm|-win_match match_expression)...]
        [(-um|-unix_match match_expression)...]
        [(-suffix|-suffixes suffix)...]
        [(-ws|-win_suffix|-win_suffixes suffix)...]
        [(-us|-unix_suffix|-unix_suffixes suffix)...] type_name
```

-active

Specifies that type definition is active and available for use when creating new objects. This setting is the default setting.

-desc|-description *description*

Specifies a one line description for the type. Use to describe the meaning or purpose of the type.

-inactive

Specifies that the type definition is inactive. Users cannot create or check in objects of this type.

-match *match_expression*

Specifies a *match_expression* regular expression that is used to match files for this type definition on both Windows and UNIX. This option is equivalent to specifying -wm *match_expression* -um *match_expression*.

-nowi|-nowin_ignore

On Windows, specifies that files of this type are not ignored by default during reconcile or migrate.

-noui|-nounix_ignore

On UNIX, specifies that files of this type are not ignored by default during reconcile or migrate.

-nox|-noexecute_permission

On a UNIX client, specifies files of this type in a work area do not have the UNIX 'x' execute permission.

-pci|-parallel_check_in (inherit | true | false)

Specifies whether parallel objects of this type can be checked in. The value inherit means that this property is inherited from its super_type. The value true means that parallel check in might be

permitted, subject to the settings on the release definition associated with the release of the object. The value false means that parallel check in is not permitted.

-pco|-parallel_check_out (inherit | true | false)

Specifies whether parallel objects of this type can be checked out. The value inherit means that this property is inherited from its *super_type*. The value true means that parallel check out might be permitted, subject to the settings on the release definition associated with the release of the object. The value false means that parallel check out is not permitted.

-rke|-range_keyword_expand (*range_integer*|inherit)

Specifies the range for keyword expansion. The value inherit means that the setting is inherited from its *super_type*. See range_for_keyword_expand for details.

-st|-super_type *super_type*

Specifies the *super_type* for this type. The *super_type* must be a file-based type. Typical values are ascii or binary. It is invalid to create cyclic inheritance.

-stf|-source_template_file *template_file*

Specifies a source template. The template is read from the *template_file*. The template can use Rational® Synergy keywords that are expanded on create, check out, and, optionally, check in.

-suffix|-suffixes *suffix*)...

Specifies that Windows and UNIX file matching patterns be added to match the specified *suffix*. The *suffix* must be one or more suffixes separated by commas, each starting with '.'., which is equivalent to specifying -ws *suffix* -us *suffix*.

type_name

Specifies the name of the new type to create. The type name cannot contain any restricted characters. You cannot create a type if another type exists with the same name, even if the case is different.

-wi|-win_ignore

On Windows, specifies that files of this type are ignored by default during reconcile or migrate.

-ws|-win_suffix|-win_suffixes *suffix*

Specifies that Windows file-matching patterns be added to match the specified *suffix*. The *suffix* must be one or more suffixes separated by commas, each starting with'.'.

-ui|-unix_ignore

On UNIX, specifies that files of this type are ignored by default during reconcile or migrate.

-us|-unix_suffix|-unix_suffixes *suffix*

Specifies that UNIX file matching patterns be added to match the specified *suffix*. The *suffix* must be one or more suffixes separated by commas, each starting with '.'.

-x|-execute_permission

On a UNIX client, specifies that files of this type in a work area have the UNIX 'x' execute permission.

Exporting a type

The `ccm type -export` command exports a type definition. Then, a type definition can be imported into another Rational® Synergy database using the `ccm typedef -import` command.

About this task

```
ccm typedef -export [-to xml_file] type_name
```

CAUTION:
Do not export Rational Synergy base model type definitions and import them into another Rational Synergy database that is at a different Rational Synergy release, iFix, or patch level. Such usage is invalid and might result in patch or upgrade changes made to that base model type definition being overwritten or lost. Patch or upgrade changes might result in unpredictable and erroneous behavior.

Instead, export user-created custom type definitions from a Rational Synergy database and import them into another database at the same or different release, iFix, or patch level. However, features that might be available in the originating database might not be available in databases at previous Rational Synergy releases.

-to *xml_file*

Specifies the XML file that is created for the exported type definition. The file must use a '.xml' suffix. The default is a file in the current directory named *type_name*.xml.

type_name

Specifies the name of the type definition to be exported.

Importing a type

The `ccm type -import` command imports a type definition that was previously exported using the `ccm typedef -export` command.

About this task

```
ccm typedef -import xml_file
```

CAUTION:
Do not export Rational® Synergy base model type definitions and import them into another Rational Synergy database that is at a different Rational Synergy release, iFix, or patch level. Such usage is invalid and might result in patch or upgrade changes made to that base model type definition being overwritten or lost. Patch or upgrade changes can result in unpredictable and erroneous behavior.

Instead, export user-created custom type definitions from a Rational Synergy database and import them into another database at the same or different release, iFix, or patch level. However, features that might be available in the originating database might not be available in databases at previous Rational Synergy releases.

xml_file

Specifies the type definition export file to import.

Listing types

The `ccm typedef -list` command lists the file-based types, and the `symlink` and `dir` types that match the specified criteria.

About this task

```
ccm typedef -l|-list ([-active] | [-inactive])
```

-active

Specifies that only the active type definitions are listed. The default is to list all file-based types plus symlink and dir.

-inactive

Specifies that only the inactive type definitions are listed. The default is to list all file-based types plus symlink and dir.

Modifying a type

The `ccm typedef -modify` command modifies a specified file-based type, or the `symlink` or `dir` type.

About this task

```
ccm typedef -m|-modify [-desc|-description description]
        [-st|-super_type super_type]
        [-stf|-source_template_file template_file]
        [-rke|-range_keyword_expand (range_integer|inherit)]
        [-pco|-parallel_check_out (inherit | true | false)]
        [-pci|-parallel_check_in (inherit | true | false)]
        ([-active] | [-inactive])
        ([-x|-execute_permission] | [-nox|-noexecute_permission])
        ([-wi|-win_ignore] | [-nowi|-nowin_ignore])
        ([-ui|-unix_ignore] | [-noui|-nounix_ignore])
        [(-match match_expression)...]
        [(-wm|-win_match match_expression)...]
        [(-um|-unix_match match_expression)...]
        [(-add_suffix|-add_suffixes suffix)...]
        [(-add_win_suffix|-add_win_suffixes suffix)...]
        [(-add_unix_suffix|-add_unix_suffixes suffix)...]
        [(-remove_suffix|-remove_suffixes suffix)...]
        [(-remove_win_suffix|-remove_win_suffixes suffix)...]
        [(-remove_unix_suffix|-remove_unix_suffixes suffix)...]
        type_name
```

-active

Modifies the type definition to be active and available for use when creating objects.

-add_suffix|-add_suffixes *suffix*

Adds match expressions for both Windows and UNIX that match the specified *suffix*. The *suffix* must be one or more suffixes separated by commas, each starting with '.'. This option is equivalent to specifying -add_win_suffix *suffix* -add_unix_suffix *suffix*.

-add_win_suffix|-add_win_suffixes *suffix*

Adds match expressions for Windows that match the specified *suffix*. The *suffix* must be one or more suffixes separated by commas, each starting with '.'.

-add_unix_suffix|-add_unix_suffixes *suffix*

Adds match expressions for UNIX that match the specified *suffix*. The *suffix* must be one or more suffixes separated by commas, each starting with '.'.

-desc|-description *description*

Modifies the one-line description for the type. Use to describe the meaning or purpose of the type.

-inactive

Modifies the type definition to be inactive. Users are not able to create or check in objects of this type.

-match *match_expression*

Modifies the *match_expression* regular expression that is used to match files for this type definition on both Windows and UNIX. This option is equivalent to specifying -wm *match_expression* -um *match_expression*.

-nowi|-nowin_ignore

On Windows, modifies the type so that files of this type are not ignored by default during reconcile or migrate.

-noui|-nounix_ignore

On UNIX, modifies the type so that files of this type are not ignored by default during reconcile or migrate.

-nox|-noexecute_permission

On a UNIX client, modifies the type so that files of this type in a work area do not have the UNIX 'x' execute permission.

-pci|-parallel_check_in (inherit | true | false)

Specifies whether parallel objects of this type can be checked in. The value inherit means that this property is inherited from its *super_type*. The value true means that parallel check in might be permitted, subject to the settings on the release definition associated with the release of the object. The value false means that parallel check in is not permitted.

-pco|-parallel_check_out (inherit | true | false)

Specifies whether parallel objects of this type can be checked out. The value inherit means that this property is inherited from its *super_type*. The value true means that parallel check out might be permitted, subject to the settings on the release definition associated with the release of the object. The value false means that parallel check out is not permitted.

-remove_suffix|-remove_suffixes *suffix*

Removes match expressions for both Windows and UNIX that match the specified *suffix*. The *suffix* must be one or more suffixes separated by commas, each starting with '.'. This option is equivalent to specifying -remove_win_suffix *suffix* -remove_unix_suffix *suffix*.

-remove_win_suffix|-remove_win_suffixes *suffix*

Removes match expressions for Windows that match the specified *suffix*. The *suffix* must be one or more suffixes separated by commas, each starting with '.'.

-remove_unix_suffix|-remove_unix_suffixes suffix

Removes match expressions for UNIX that match the specified *suffix*. The *suffix* must be one or more suffixes separated by commas, each starting with '.'.

-rke|-range_keyword_expand (*range_integer*|inherit)

Modifies the range for keyword expansion. The value inherit means that the setting is inherited from its *super_type*. See range_for_keyword_expand for details.

-st|-super_type *super_type*

Modifies the *super_type* for this type. The *super_type* must be a file-based type. Typical values are ascii or binary. It is invalid to create cyclic inheritance.

-stf|-source_template_file *template_file*

Modifies the source template. The template is read from the *template_file*. The template might use Rational® Synergy keywords that are expanded on create, check out, and, optionally, check in.

-suffix|-suffixes *suffix*)...

Modifies the Windows and UNIX file matching patterns to match the specified *suffix*. The *suffix* must be one or more suffixes separated by commas, each starting with '.'. This option is equivalent to specifying -ws *suffix* -us *suffix*.

type_name

Specifies the name of the type to modify.

-wi|-win_ignore

On Windows, modifies the type so that files of this type are ignored by default during reconcile or migrate.

-ws|-win_suffix|-win_suffixes *suffix*

Modifies the Windows file-matching patterns to match the specified *suffix*. The *suffix* must be one or more suffixes separated by commas, each starting with '.'.

-ui|-unix_ignore

On UNIX, modifies the type so that files of this type are ignored by default during reconcile or migrate.

-us|-unix_suffix|-unix_suffixes *suffix*

Modifies the UNIX file matching patterns to match the specified *suffix*. The *suffix* must be one or more suffixes separated by commas, each starting with '.'.

-x|-execute_permission

On UNIX client, modifies the type so that files of this type in a work area have the UNIX 'x' execute permission.

Showing information for a type

The `ccm typedef -show information` command shows information about a specified type definition.

About this task

```
ccm typedef -s|-sh|-show (i|info|information) type_name
```

type_name

Specifies the name of the type to show.

Showing a property for a type

The `ccm typedef -show information` command shows a specified property of a specified type definition.

About this task

```
ccm typedef -s|-sh|-show ((desc|description) | (st|super_type) |
        (wm|win_match) | (um|unix_match) | (wi|win_ignore) |
        (ui|unix_ignore) | (rke|range_keyword_expand) | active |
        (pco|parallel_check_out) | (pci|parallel_check_in) |
        (x|execute_permission) | source_template)
        type_name
```

type_name

Specifies the name of the type to show.

unalias command

You can remove an alias for the current session only.

About this task

The `unalias` command supports the "Removing an alias" subcommand.

Removing an alias

You can remove a specified alias for the current session only.

About this task

```
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
```

undo_update command

You can reverse an update operation for a specified directory or project object. You can also reverse the last undo update operation. In other words, if two or more undo updates are performed, only the last one is reversed.

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. Stopping the undo protects the work area data.

By default and for performance reasons, the `undo_update` command does not provide parallel version notification when it encounters parallel object versions. You can set parallel version notification by configuring the reconfigure_parallel_check user option to TRUE in your initialization file.

If you want to continue with the `undo_update` process, even though an individual failure has occurred, you can set update to continue. Set the reconf_stop_on_fail option to `False`.

The `undo_update` command supports these 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

You can reverse an update for a directory.

About this task

```
ccm unupd|undo_update|unreconf|undo_reconfigure [-r|-recurse]
          [-v|-verbose] dir_spec...
```

dir_spec...

Specifies the directory to reverse the update in. 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

Shows detailed undo update messages.

Reversing an update for a project

You can reverse an update for a project.

About this task

```
ccm unupd|undo_update|unreconf|undo_reconfigure -p|-project
          [-r|-recurse] [-v|-verbose] project_spec...
```

project_spec

Specifies the project to reverse the update in. 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

Shows detailed undo update messages.

Example

- 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
```

Reversing an update for a project grouping

You can reverse an update for a project grouping.

About this task

```
ccm unupd|undo_update|unreconf|undo_reconfigure
          -pg|-project_grouping [-r|-recurse] [-v|-verbose]
          project_grouping_spec...
```

project_grouping_spec

Specifies the project grouping to reverse the update in. 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

Shows detailed undo update messages.

unrelate command

The `unrelate` command deletes a relationship, `rel_name`, between `file_spec1` and `file_spec2`.

You can define new relationships using the `relate` command.

To delete the relationship between two objects, you must include all three object specifications.

The command does not update the query selection set.

The `unrelate` command supports the "Deleting a relationship between two objects" subcommand.

Deleting a relationship between two objects

You can delete a specified relationship from an object to one or more destination objects, or from specified objects to a single destination object. If you delete a relationship that does not exist, the operation succeeds although the command does nothing.

About this task

```
ccm unrelate -n|-name relationship_name
             (-f|-from from_object_spec)...
             (-t|-to to_object_spec)...
```

-f|-from *object_spec*...

Specifies the originating object or objects from which the relationship is created. If more than one origin object is specified, you cannot specify multiple destination objects.

-n|-name *relationship_name*

Specifies the name of the relationship to delete.

-t|-to *object_spec*...

Specifies the destination object or objects to which the relationship is created. If more than one destination object is specified, you cannot specify multiple origin objects.

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
```

unuse command

You can remove an existing file, directory, root directory, or project from the current project or directory. The directory must be modifiable to remove members from it. If you try to remove an object from a non-modifiable directory, the directory is automatically checked out (unless you specify the `-r` option). You must check **in** the directory to make the files in the directory available to other users.

The root directory can be the command target, 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 working in a shared project and the directory is non-modifiable, the directory is checked out and associated automatically with the specified task. The directory is then checked in to the *integrate* state. You can disable automatic check-in by setting shared_project_directory_checkin to FALSE in your initialization file.

If you want to delete a project, see the delete command (`ccm delete -p project_name-version`).

You can start the command from any location if you use the Folder specification:

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
```

A message reports the `object_version` that was removed and which version replaced it.

The `unuse` command supports these 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

You can remove a project from a specified directory and its associated context project. The directory must be specified using a project reference specification form or a work area reference form to provide the context project.

Before you begin

If the context project is in the *working* state, you must be the owner of that project. If the context project is in *prep* state, you must be a build manager.

About this task

```
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 to remove the object from. 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 must be used with the `-d|-delete` option. Forcing a removal specifies that the deleted object is removed from all projects that are modifiable by you before it is deleted from the database. Without specifying `-force`, the object is removed from the context project. The object is removed from the database 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 to associate the specified task with the 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 nonmodifiable, 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.

Example

- 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_jan4-1Member 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
```

Removing a project from the current directory

You can remove 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.

About this task

```
ccm unuse -p|-project [-t|-task task_spec] [-d|-delete [-f|-force]]
          [-r|-replace] project_spec...
```

-d|-delete

Remove the object from the current directory and its context project, then delete the object from the database.

-force

This option must be used with the `-delete` option. Forcing a removal specifies that the deleted object is removed from all projects that are modifiable by you before it is deleted from the database. Without specifying `-force`, the object is removed from the database if it is not a member of any project.

project_spec

Specifies the projects to be unused. See Project specification for details.

-r|-replace

Replaces the object in the directory with its predecessor. When this option is specified, the list of files in the directory remains unchanged; only the specified object version changes.

-t|-task *task_spec*

Associates the specified task with the directory that is automatically checked out. If `-replace` is not specified, the directory that contains the object is updated to remove the entry for that object. If the directory is non-modifiable, the directory is automatically checked out. If the `-task` option is not specified, the current task is used by default. See Task specification for details.

Example

Cut `sort.c` from the current directory.

```
ccm unuse sort.c-1:csrc:1
```

Removing an object from a project

You can remove an object from a project. 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.

The `-dir` option serves two purposes. If you specify the `-dir` option, the object is removed from the directory and its associated context project. If you do not specify the `-dir` option, the command removes the object from the current working directory.

About this task

```
ccm unuse [-dir dir_spec] [-t|-task task_spec] [-d|-delete [-f|-force]]
          [-r|-replace] file_spec...
```

-d|-delete

Remove the object from the project, then delete the object from the database.

-dir *dir_spec*

Specifies to the directory to unuse the object from. 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 specification form or a work area reference form provides such a context project.

file_spec

Specifies the object or objects to unuse. (See File specification for details.)

-force

This option must be used with the `-delete` option. Forcing a removal specifies that the deleted object is removed from all projects that are modifiable by you before it is deleted from the database. Without specifying `-force`, the object is removed from the context project for the object. The object is deleted from the database only if it is not a member of any project.

-r|-replace

Replaces the object in the directory with its predecessor. When this option is specified, the list of files in the directory remains unchanged; only the specified object version changes.

-t|-task *task_spec*

Associates the specified task with the directory that is automatically checked out. If `-replace` is not specified, the directory that contains the object is updated to remove the entry for that object. If the directory is non-modifiable, the directory is automatically checked out. If the `-task` option is not specified, then the current task is used by default. See Task specification for details.

Example

- 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
```

update command

You can update a specified directory, project object, or project grouping. Update uses the baseline and tasks of project groupings to find candidates and selection rules to select appropriate versions of the members. You can also specify a project grouping to be updated.

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. Stopping the update protects the work area data.

The default setting to stop the update can be changed by modifying your initialization file. Some users might want to continue with the update process, even though an individual failure has occurred. You can set update to continue by configuring 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 refreshing the baseline and tasks. 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 might 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 might change, and the commands to use to change them:

- Change the release or purpose for the project 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.

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 set parallel version notification by configuring the reconfigure_parallel_check user option.

The `update_members` command supports these subcommands:

- Updating members for a directory
- Updating members for a project grouping
- Updating project members

Updating members for a directory

You can update members of a directory.

About this task

```
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 do not specify `-keep_subprojects` or `-replace_subprojects`, the default is used.

-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 have set `replace_subprojects`.

-r|-recurse

Specifies to include subprojects.

-v|-verbose

Displays detailed undo update messages.

Updating members for a project grouping

You can update members of a project grouping.

About this task

```
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 do not specify `-keep_subprojects` or `-replace_subprojects`, the default is used.

-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 do not specify `-keep_subprojects` or `-replace_subprojects`, the default depends on how you have set `replace_subprojects`.

-r|-recurse

Specifies to include subprojects.

-v|-verbose

Displays detailed undo update messages.

Updating project members

You can update project members from the directory or project level. Update uses the baseline and tasks of project groupings to find the candidates and selection rules to select appropriate object versions.

About this task

```
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 do not specify `-keep_subprojects` or `-replace_subprojects`, the default is used.

-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 do not specify `-keep_subprojects` or `-replace_subprojects`, the default depends on how you have set `replace_subprojects`.

-r|-recurse

Specifies to include subprojects.

-v|-verbose

Shows detailed undo update messages.

Example

- 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"
```

use command

You can replace an existing file, directory, or project with another version. Additionally, you can add a file, directory, or project that is in the database, but not in the current directory. When you add an object to a non-modifiable directory, a new directory version is checked out automatically.

If you are working in a shared project and the directory is non-modifiable, the directory is checked out and associated automatically with the default task. The directory is then checked in to the *integrate* state. You can disable automatic check-in 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 from the parent project, the different version of the subproject or product is associated with the current task. Disable this feature by setting `add_used_subcomponents_to_task` to FALSE in your initialization file. (See admin_user.)

When you use a directory, the directory is updated automatically. You must check in the directory to make the files available to other users.

The `use` command supports these 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

You can add 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. 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 from the parent project, the different version of the subproject is associated with the current task.

About this task

```
ccm use -p|-project [-t|-task task_spec] project_spec...
```

project_spec

Specifies the projects you are using. See Project specification for details.

Special handling might be required in shared projects (see shared_project_directory_checkin).

-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
```

Using different versions or adding objects to a specified directory

You can use different versions of an object or add existing objects as new project members under the specified directory. The directory objects added in 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 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 and associated with the specified task. A new directory entry is then created for the object, which is used in the context project for the directory. 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 nonmodifiable product with a component that is different from the parent project, the different version of the product is associated with the current task.

About this 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

Specifies the projects to be used. See Project specification for details.

Special handling might be required in shared projects (see shared_project_directory_checkin).

-t|-task *-task_spec*

Specifies the task that is associated with any directory that was checked out when a new member was added. 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

- 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 the `SubPrj-one:project:1` project to the root directory of the `TopPrj-top:project:1` project. (The current directory can be any directory; the projects might not have maintained work areas.)

```
ccm use -p -dir TopPrj@TopPrj-top -task 31 SubPrj-one:project:1
```

- Use the `a.txt-1.2:ascii:1` object to the `dir1` directory under the root directory of the `TopPrj-top:project:1` project.

```
ccm use -dir TopPrj\dir1@TopPrj-top -task 31 a.txt-1.2:ascii:1
```

- Use a different version for the `a.txt-1.1:ascii:1` object.

```
ccm use -dir TopPrj\dir1@TopPrj-top a.txt-1.1:ascii:1
```

Using different versions or adding objects to the current directory

You can use a different version of an object in the current working directory. You can also add 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. If the context project is in the *working* state, you must own the 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 from the parent project, the different version of the product is associated with the current task.

About this task

```
ccm use [-r|-rules|-recommend] [-t|-task task_spec] file_spec...
```

file_spec

Specifies the object versions you are using. See File specification for details.

-r|-rules|-recommend

Uses the version selected by the selection rules.

-t|-task *-task_spec*

Specifies the task that is associated with the current directory that was checked out when a new member was added. 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 `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
```

view command

You can show a 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 started from the 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.

The `view` command supports the "Viewing a file" subcommand.

Viewing a file

You can show a specified file. The default viewer is used to view the file.

About this task

```
ccm view file_spec...
```

file_spec

Specifies the object to be displayed. See File specification for details.

Example

View the `log.c` object with a version of 8.

```
ccm view log.c-8
```

work_area command

You can show and modify work area options.

The `work_area` command supports these subcommands:

- Modifying work area properties
- Showing work area properties
- Finding and showing all projects with the specified character string in their work area paths
- Finding and replacing a character string in a work area path
- Updating projects with an obsolete database path after moving a database
- Showing projects with an obsolete database path after moving a database
- Updating projects with an obsolete work area path after copying a database
- Showing projects with an obsolete work area path after copying a database

Modifying work area properties

You can change the work area properties of a project, such as the work area path and whether the work area is maintained. If you do not specify a project, the command updates the project whose work area is associated with the current working directory.

About this task

```
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*. The default is `-not_modifiable_wa`.

-nr|-no_recurse

Do not recurse the project hierarchy when applying these options. Change only the specified project. The default is `-no_recurse`.

-nrel|-not_relative

Specifies that any work area is 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 `-translate`.

-nwa|-no_wa

Specifies not to maintain the work area for the project. This default is `-maintain_wa`.

-nwat|-no_wa_time

Specifies to use timestamps for the files in the project's work are. The timestamps must show the modification time rather than the time they were copied into the work area. This default is `-no_wa_time`.

-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 the `setpath_absolute_path`option.

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 to locate any work area 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 the `-project_subdir_template`option.

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 to copy ASCII files in the work area 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.

If you stop the sync, an error message reports that errors might occur in your work area. However, the errors do not occur until you try to use the work area. To avoid problems, perform a synchronization of the entire work area before you use it.

You can use this option when using the `-recurse` option on a parent project of a read-only project only if you are in the *ccm_admin* role. Build managers can use the command directly on the released project.

-wat|-wa_time

Specifies to use timestamps for the files in the project's work area. The timestamps must show the time they were copied into the work area, rather than the modification time. The default is `-no_wa_time`.

Showing work area properties

You can show the work area properties of a project. If you do not specify a project, the command shows the work area properties of the project whose work area is associated with the current working directory.

About this task

```
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_header 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_header for details.

-nf|-noformat

Specifies not to use column alignment. See -nf|-noformat for details.

-ns|-nosort|-no_sort

Specifies not to sort the output. See -ns|-nosort for details.

p|-project

It is not necessary to specify this option.

project_spec

Specifies the project to be shown. See Project grouping 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 `-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.

-new

Used with `-find -replace` to indicate that the database is new. This option is intended for two situations: when the work areas specified by the `-find` option are not visible, or are visible but are ignored.

-visible

Indicates to consider only visible work areas for update.

A message is displayed for work areas skipped because the work area is not visible to the interface. The default is `-visible`.

See the `-new` option for information about updating work area paths of projects that are in a database in which work areas have not yet been created.

You can use this option only with the `-find` option.

Example

Show work area properties:

```
ccm wa -show -recurse project-2
```

Finding and showing all projects with the specified character string in their work area paths

Use this command to find all projects with a specified character string in their work area paths, and to show the projects.

About this task

```
ccm wa|work_area -find "find_string" [-reg|-regexp] [-replace "new_string"]
                 -show
                 [-scope working|prep|shared|checkpoint|STATIC|ALL|DB]
                 [-p|-project] project_spec [project_spec...]
```

p|-project

It is not necessary to specify this option.

project_spec...

Specifies the project to be shown. See Project grouping specification for details.

-reg|-regexp

Indicates that *new_string* and *find_string* are regular expressions.

You can use this option only with the `-find` option.

-replace "*new_string*"

Substitutes *new_string* for *find_string* in the work area paths of all of the projects found using the `-find` *find_string* option.

Use the `-reg` option if you want to use a regular expression (for example, ".*") for *new_string*.

You can use this option only with the `-find` option.

-scope

Establishes an initial criterion for which projects can be found using the `-find` or `-dbpath` options.

The scope option can be one of these states:

working (all *working*-state projects)
checkpoint (all *checkpoint*-state projects)
prep (all *prep* projects)
shared (all *shared* projects)

Alternatively, the scope option can be one of these case-sensitive keywords:

`STATIC` (all non-writable projects; for example, projects in the *integrate*, *test*, *sqa*, or *released* state)
`ALL` (all projects, regardless of status)
`DB` (all projects for the current database, regardless of ownership or status)

If the scope is *working*, *checkpoint*, or `ALL`, the projects must be owned by you.

The default scope is *working*.

You can use this option only with the `-find` or `-dbpath` option.

Finding and replacing a character string in a work area path

Use this command to find all projects with a specified character string in their work area paths, and to replace the character string.

About this task

```
ccm wa|work_area -find "find_string" [-reg|-regexp] -replace "new_string"
                 [-scope working|prep|shared|checkpoint|STATIC|ALL|DB]
                 [-new|-visible]
                 [-p|-project] project_spec [project_spec...]
```

p|-project

It is not necessary to specify this option.

project_spec...

Specifies the project. See Project grouping specification for details.

-reg|-regexp

Indicates that *new_string* and *find_string* are regular expressions.

You can use this option only with the `-find` option.

-replace "*new_string*"

Substitutes *new_string* for *find_string* in the work area paths of all of the projects found using the `-find` *find_string* option.

Use the `-reg` option if you want to use a regular expression (for example, ".*") for *new_string*.

You can use this option only with the `-find` option.

-scope

Establishes an initial criterion for which projects can be found using the `-find` or `-dbpath` options.

The scope option can be one of these states:

working (all *working*-state projects)
checkpoint (all *checkpoint*-state projects)
prep (all *prep* projects)
shared (all *shared* projects)

Alternatively, the scope option can be one of these case-sensitive keywords:

`STATIC` (all non-writable projects; for example, projects in the *integrate*, *test*, *sqa*, or *released* state)
`ALL` (all projects, regardless of status)
`DB` (all projects for the current database, regardless of ownership or status)

If the scope is *working*, *checkpoint*, or `ALL`, the projects must be owned by you.

The default scope is *working*.

You can use this option only with the `-find` or `-dbpath` option.

Updating projects with an obsolete database path after moving a database

You can update projects whose work area references an obsolete database path. Use this command when a database is moved to a new location that has similar visibility to Rational® Synergy clients.

About this task

```
ccm wa|work_area -dbpath old_database_path -replace [-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] [-u|-unnumbered]
            project_spec...
```

-ch|-column_header

Specifies to use a column header in the output format. See -ch|-column_header 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_header for details.

-nf|-noformat

Specifies not to use column alignment. See -nf|-noformat for details.

-ns|-no_sort

Specifies not to sort the command output. See -ns|-nosort for details.

old_database_path

Specifies the path of the database before it was moved to its current location.

project_spec...

Specifies the project to be shown. See Project grouping 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.

-u|-unnumbered

Suppresses automatic numbering of the command output. See -u|-unnumbered for details.

Showing projects with an obsolete database path after moving a database

You can show projects whose work area references an obsolete database path. Use this command when a database is moved to a new location that has similar visibility to Rational® Synergy clients.

About this task

```
ccm wa|work_area -dbpath old_database_path [-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] [-u|-unnumbered]
            project_spec...
```

-ch|-column_header

Specifies to use a column header in the output format. See -ch|-column_header 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_header for details.

-nf|-noformat

Specifies not to use column alignment. See -nf|-noformat for details.

-ns|-no_sort

Specifies not to sort the command output. See -ns|-nosort for details.

old_database_path

Specifies the path of the database before it was moved to its current location.

p|-project

It is not necessary to specify this option.

project_spec...

Specifies the project to be shown. See Project grouping 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.

-u|-unnumbered

Suppresses automatic numbering of the command output. See -u|-unnumbered for details.

Updating projects with an obsolete work area path after copying a database

You can update projects whose work area references an obsolete work area path. Use this command when a database is copied to a new location that has similar visibility to Rational® Synergy clients.

About this task

```
ccm wa|work_area -find find_string -replace new_string
            [-s|-sh|-show] [-reg|-regexp] [-visible]
            [-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...
```

-ch|-column_header

Specifies to use a column header in the output format. See -ch|-column_header for details.

find_string

Specifies the string to find in each project's work area path. If `-regexp` is not specified, the operation finds projects whose work area path contains the *`find_string`*. If `-regexp` is specified, *`find_string`* is treated as a regular expression.

-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_header for details.

-nf|-noformat

Specifies not to use column alignment. See -nf|-noformat for details.

-ns|-no_sort

Specifies not to sort the command output. See -ns|-nosort for details.

project_spec...

Specifies the project to be shown. See Project grouping specification for details.

-reg|-regexp

Indicates that *`new_string`* and *`find_string`* are regular expressions.

You can use this option only with the `-find` option.

-replace *new_string*

Specifies the new string to replace *`find_string`*. If `-regexp` is not specified, *`new_string`* replaces *`find_string`* in the work area path. If `-regexp` is specified, *`new_string`* is treated as a replacement regular expression. See Regular expressions 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 output. See -u|-unnumbered for details.

-visible

Specifies to update projects whose work area is visible to the current client only.

Showing projects with an obsolete work area path after copying a database

You can show projects whose work area references an obsolete work area path. Use this command when a database is copied to a new location that has similar visibility to Rational® Synergy clients.

About this task

```
ccm wa|work_area -find find_string -replace new_string
            [-s|-sh|-show] [-reg|-regexp] [-visible]
            [-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...
```

-ch|-column_header

Specifies to use a column header in the output format. See -ch|-column_header for details.

find_string

Specifies the string to find in each project's work area path. If `-regexp` is not specified, the operation finds projects whose work area path contains the *`find_string`*. If `-regexp` is specified, *`find_string`* is treated as a regular expression.

-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_header for details.

-nf|-noformat

Specifies not to use column alignment. See -nf|-noformat for details.

-ns|-no_sort

Specifies not to sort the command output. See -ns|-nosort for details.

project_spec...

Specifies the project to be shown. See Project grouping specification for details.

-reg|-regexp

Indicates that *new_string* and *find_string* are regular expressions.

You can use this option only with the `-find` option.

-replace *new_string*

Specifies the new string to replace *find_string*. If `-regexp` is not specified, *new_string* replaces *find_string* in the work area path. If `-regexp` is specified, *new_string* is treated as a replacement regular expression. See Regular expressions 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 output. See -u|-unnumbered for details.

-visible

Specifies to show projects whose work area is visible to the current client only.

Notices

Programming interfaces: Intended programming interfaces allow the customer to write programs to obtain the services of Rational® Change.
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 on 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 license inquiries, in writing, to:
IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.
For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:
Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 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 and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites 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
5 Technology Park Drive
Westford, MA 01886
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.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

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.

Copyright license

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Trademarks

A current list of IBM trademarks is available on the Web at www.ibm.com/legal/copytrade.shtml.