Rational. Synergy

IBM

CLI ヘルプ

IBM Rational Synergy

CLI ヘルプ

リリース 7.1a

本書をご使用になる前に、特記事項に記載されている情報をお読みください。

本書は、Rational Synergy（製品番号 5724V66）バージョン 7.1 および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

目次

全般的な使用方法 6

IBM Rational Synergy Readme 7
CLI の使用方法：Windows ユーザー 8
CLI の使用方法：UNIX ユーザー 10
Rational Synergy インターフェイス 11
用語と名称の変更 13
Rational Synergy CLI ヘルプ 14
IBM Rational ソフトウェア サポートへの問い合わせ 16
コマンドと引数の構文 19
グローバル フォーマット オプション 41
命名制限 55
大文字／小文字とファイル名制限のデータベース オプション 58
組み込み済みキーワード 59
正規表現 62
目的とテンプレートの管理 64

デフォルト設定 65

デフォルトの設定方法 66
デフォルト オプション 68
初期設定ファイル — Windows 99
初期設定ファイル — UNIX 99
環境変数 100
モデル オブジェクト属性オプションの設定 102
新しい属性のリスト ボックスの作成 104
オブジェクト タイプ属性オプションの設定 105
システム用 ini ファイルまたは個人用の ini ファイルでのオプションの設定 106
ccm set コマンドを使用したオプションの設定 107

コマンド 108

alias コマンド 109
attribute コマンド 114
baseline コマンド 124

bom コマンド 151
candidates コマンド 154
cat コマンド 158
change_type コマンド 161
checkin コマンド 164
checkout コマンド 172
checkpoint コマンド 182
cmdhistory コマンド 188
conflicts コマンド 193
copy_project コマンド 199
copy_to_file_system コマンド 206
create コマンド 209
dcm コマンド 218
delete コマンド 288
delimiter コマンド 294
dir コマンド 297
edit コマンド 303
finduse コマンド 306
folder コマンド 320
folder_template コマンド 346
groups コマンド 367
history コマンド 373
ln コマンド 378
ls コマンド 382
merge コマンド 387
move コマンド 391
process コマンド 398
process_rule コマンド 414
project コマンド 436
project_grouping コマンド 440
project_purpose コマンド 473
properties コマンド 480
query コマンド 485
reconcile コマンド 491
relate コマンド 503

release コマンド 508
save_offline_and_delete コマンド 528
soad_scope コマンド 538
set コマンド 552
show コマンド 557
show_servers コマンド 564
start コマンド 568
stop コマンド 574
sync コマンド 578
task コマンド 582
unalias コマンド 612
undo_update コマンド 615
unrelate コマンド 620
unset コマンド 624
unuse コマンド 627
update コマンド 634
use コマンド 640
view コマンド 646
work_area コマンド 649

Rational Synergy ヘルプへのリンク **657**

特記事項 **658**

全般的な使用方法

本章は IBM® Rational® Synergy の使用方法 ® を説明します。以下のトピックについて説明します。

- IBM Rational Synergy Readme
- CLI の使用方法：Windows ユーザー
- CLI の使用方法：UNIX ユーザー
- Rational Synergy インターフェイス
- 用語と名称の変更
- Rational Synergy CLI ヘルプ
- IBM Rational ソフトウェア サポートへの問い合わせ
- コマンドと引数の構文
- グローバル フォーマット オプション
- 命名制限
- 大文字／小文字とファイル名制限のデータベース オプション
- 組み込み済みキーワード
- 正規表現
- 目的とテンプレートの管理

特記事項

IBM Rational Synergy Readme

Rational Synergy の使用または管理を開始する前に、最新の Readme ファイルをお読みになることを推奨します。*Rational Synergy Readme* には、製品リリースに関する情報と以下の項目についての説明があります。

- システム要件
- 他の Rational 製品およびリリースとの互換性
- 本リリースの新機能
- 将来の変更の通知

CLI の使用方法：Windows ユーザー

Rational Synergy は、対応するすべての Windows プラットフォームにおけるコマンドライン インターフェイス（CLI）をサポートしています。

どの Rational Synergy コマンドも、Windows コマンド プロンプトから実行できます。

ただし､Rational Synergy インターフェイスからはコマンドライン インターフェイスを起動できません。CLI は別に起動する必要があります。

オプション区切り文字

デフォルトで、Windows クライアントはオプションの区切り文字としてスラッシュ（/）をサポートしていますが、ダッシュ（-）も使用可能です。このヘルプの例では、区切り文字としてダッシュ（-）を使用しています。

汎用名前付け規則

管理コマンドにパスを入力するときは、必ず汎用名前付け規則（UNC）を使用します。UNC により、ファイル、マシンその他のデバイスへのネットワーク アクセスが容易になります。UNC によって一定のフォーマットを使用することにより、リモート マシンやファイルを特定できます。フォーマットは以下のとおりです。

`¥¥`*`computer_name`*`¥`*`share_name`*`¥`*`path`*

以下の例で、`¥¥loon¥ccmdb¥tstgonzo` は UNC 形式のパスです。

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

すべての Rational Synergy コマンドで、UNC パスとドライブ文字を使用するパスの**両方を**使用できます（例、`c:¥users¥ccmdb¥base`）。ただし、`ccmdb create`、`ccmdb copy`、`ccmdb unpack` の 3 つのコマンドでは、データベースを作成するために UNC パスを指定する必要があります。

ファイル パス

Windows クライアントは、標準 Windows ファイル指定をサポートします。これは、以下のように記述します。

`drive``:¥`*`directory`*`¥`*`filename`*

Rational Synergy オンライン ヘルプでは、ファイル パスを以下のように示します。

`c:¥`*`directory`*`¥`*`filename`*

ファイルが別のドライブにある可能性もありますが､Rational Synergy ヘルプでは一貫してドライブ `c:` を使用します。

CCM_HOME の場所

`$CCM_HOME` は Rational Synergy 製品がインストールされているディレクトリです。たとえば、Rational Synergy インストール エリアの `etc` ディレクトリにある `remexec.cfg` ファイルを変更する場合、ディレクトリを *`CCM_HOME`*`¥etc` に変更する必要があります。

クライアント インストールでのデフォルト インストール ディレクトリは以下のとおりです。

```
C:¥Program Files¥IBM¥Rational¥Synergy¥7.1a
```

CLI の使用方法：UNIX ユーザー

Rational Synergy は、対応するすべての UNIX プラットフォームにおけるコマンドライン インターフェイス（CLI）をサポートしています。

どの Rational Synergy コマンドも、UNIX シェルから実行できます。

ただし、Rational Synergy インターフェイスからはコマンドライン インターフェイスを起動できません。CLI は別に起動する必要があります。

オプション区切り文字

デフォルトで、UNIX クライアントはオプションの区切り文字としてスラッシュ（/）をサポートしています。

CCM_HOME の場所

`$CCM_HOME` は Rational Synergy 製品がインストールされているディレクトリです。たとえば、Rational Synergy インストール エリアの `etc` ディレクトリにある `remexec.cfg` ファイルを変更する場合、ディレクトリを `$CCM_HOME/etc` に変更する必要があります。

Rational Synergy インターフェイス

Rational Synergy では以下のインターフェイスを提供しています。

- Synergy GUI

  このインターフェイスは、開発者およびビルド マネージャにすべての機能を提供します。ただし、管理操作はサポートしないため、管理以外の用途に使用します。Synergy GUI は、ウェブ モードまたはトラディショナル モードのいずれかで実行できます。これらのモードについては、以下のウェブ モードとトラディショナル モードで説明しています。

- Synergy CLI

  このインターフェイスは、一部の管理者操作を除いて開発者およびビルド マネージャにすべての機能を提供します。ウェブ モードでのみ実行されます。ウェブ モードについては、以下のウェブ モードとトラディショナル モードで説明しています。

- Synergy Classic GUI

  このインターフェイスは、主に管理者操作用です。Classic GUI を使用するユーザーは、Synergy GUI へ切り替えを考慮してください。Classic GUI は、ウェブ モードでは使用できません。このインターフェイスは、将来のリリースでは廃止される予定です。

- Synergy Classic CLI

  このインターフェイスは、主に管理者操作用であり、既存のスクリプトを Synergy CLI での使用に向けて変換するための移行期間を与えるために用意されています。Classic CLI は、ウェブ モードでは使用できません。このインターフェイスは、将来のリリースでは廃止される予定です。

ウェブ モードとトラディショナル モード

Synergy 7.1a では、ウェブ モードという新しい、より高速なモードを導入しています。ウェブ モードでは、クライアントとサーバー間の通信に新しい基礎アーキテクチャを使用しています。これは主に、ワイド エリア ネットワーク（WAN）を介して使用することを意図していますが、ローカル エリア ネットワーク（LAN）上でも同様に使用できます。自分がどのモードを使用すべきかについては、Synergy アドミニストレータにご確認ください。

ウェブ モードとトラディショナル モードの違いは、以下のとおりです。

	トラディショナル モード	ウェブ モード
パフォーマンス	旧リリースと同じ	より高速、特に WAN 経由時
Synergy GUI	旧リリースと同じ	ウェブ モードで使用可能
Classic GUI	リリース 6.5a と同じ	ウェブ モードでは使用不可
Synergy CLI	トラディショナル モードでは使用不可	リリース 7.1a では、管理コマンドの一部が未サポート
Classic CLI	旧リリースと同じ	ウェブ モードでは使用不可
ActiveCM	旧リリースと同じ	ウェブ モードでは使用不可
管理	リリース 6.5a と同じ	リリース 6.5a と同じ、プラス Synergy サーバー構成と RDS（下記参照）
インストール	旧リリースと同じ	トラディショナル モードと同じ
ネットワーク プロトコル	自社開発（RFC）	HTTP または HTTPS
ユーザー認証	オペレーティング システム（OS）	Rational Directory® Server™（RDS）、（LDAP）

トラディショナルモードは、現在のウェブモードの GUI および CLI ではまだ利用できない機能を提供しています。トラディショナルモードは将来のリリースでは廃止される予定です。区切り文字の変更、タイプ定義の追加や修正、アップグレードの実行、データベースのバックアップと整合性チェック、移行ユーティリティを使用したデータの移行など、ほとんどの管理機能にはトラディショナル モードを使用する必要があります。

用語と名称の変更

リリース 7.1a では、Rational Synergy インターフェイスはウェブ モードまたはトラディショナル モードのいずれかで実行できます。他のグラフィカル インターフェイスは、Rational Synergy Classic と呼ばれています。対応するコマンドライン インターフェイスは Rational Synergy Classic CLI と呼ばれています。ActiveCM はサポートされていません。

Rational Synergy Classic CLI と Rational Synergy Classic GUI インターフェイスでは、使用している用語は変わりません。以下の表に、Classic インターフェイスで使用している用語と、対応する Synergy インターフェイスの用語を示します。

Classic GUI と Classic CLI	Synergy GUI と Synergy CLI
メンバーのリコンフィギュア／更新	更新
リコンフィギュア テンプレート	プロセス ルール
リコンフィギュア プロパティ	更新プロパティ
リコンフィギュアの取り消し	更新の取り消し
チェックアウト（プロジェクト）	プロジェクトのコピー
ワークエリアのスナップショット	ファイル システムへのコピー
デフォルト タスク	カレント タスク

Rational Synergy CLI ヘルプ

現在表示されているヘルプは、Rational Synergy リリース 7.1a のウェブ モードで実行するコマンドライン インターフェイス（CLI）用のヘルプです。このヘルプ システム内の情報は、リリース 7.1a 用の現在入手可能な最新の内容です。必ず Readme を調べて、本書に含まれる情報に対する最新の変更を確認してください。

Rational Synergy Classic GUI と CLI のヘルプ

Rational Synergy Classic GUI を使用する場合、そのインターフェイスで開くヘルプ システムにも CLI に関する情報があります。ただし、GUI ヘルプと同じヘルプ システムに入っている CLI ヘルプは、最後の更新がリリース 6.3 用であり、最新のコマンドライン情報ではありません。Rational Synergy Classic CLI（トラディショナルモード）実行時に使用できる最新のコマンドとオプションについては、*CCM_HOME*¥jetty¥webapps¥help_cli（Windows）または $CCM_HOME/jetty/webapps/help_cli（UNIX）からヘルプを起動してください。最上位ファイルは、synergy_cli.html です。

Rational Synergy CLI（ウェブ モード）実行時に使用できる最新のコマンドとオプションについては、*CCM_HOME*¥jetty¥webapps¥help_cli_web（Windows）または $CCM_HOME/jetty/webapps/help_cli_web（UNIX）からヘルプを起動してください。最上位ファイルは、synergy_web_cli.html です。

現在どのヘルプ システムが表示されているのかが分からない場合は、各 HTML ページのフッターを見れば製品とリリースが確認できます。

Rational Synergy Classic CLI、トラディショナル モードのヘルプ

Rational Synergy Classic CLI を実行している場合、ヘルプ システムへのリリース 6.5 からの変更は微小です。記述されているコマンドの変更は非常にわずかです。

Rational Synergy CLI、ウェブ モードのヘルプ

Rational Synergy CLI を実行している場合、そのヘルプ システムは大幅に更新されており、サブコマンドの情報が再編成されています。いくつかのコマンドはこのリリースではサポートされていませんが、将来のリリースでサポートされます。Rational Synergy CLI は開発者およびビルド マネージャの機能の大部分をサポートしますが、一部の管理コマンドについては Rational Synergy Classic CLI を使用する必要があります。

ccm reconfigure_properties コマンドは廃止され現在サポートしていません。

以下のリストに、Rational Synergy CLI リリース 7.1a ウェブ モードを実行する場合にサポートされていないコマンドを示します。

- ccm archive_fix（IBM® Rational® ソフトウェア サポートが使用）
- ccm clean_cache
- ccm cleanup
- ccm collapse
- ccm dcm (-init)

- `ccm db_update`
- `ccm delimiter`（バージョン区切り文字の変更）
- `ccm depend`
- `ccm expand`
- `ccm export`
- `ccm fs_check`
- `ccm groups` (`-create`, `-list`)
- `ccm import`
- `ccm migrate`
- `ccm release` (`-delimiter`, `-rename`)
- `ccm resync`
- `ccm show -mar`
- `ccm source`
- `ccm typedef`
- `ccm update_properties`
- `ccm users`
- `ccm work_area` (`-find`, `-dbpath`)
- `ccm win_fixup`（IBM Rational ソフトウェア サポートが使用）

IBM Rational ソフトウェア サポートへの問い合わせ

お手持ちのリソースで、問題が解決されない場合は、IBM®Rational® ソフトウェア・サポートに連絡してください。IBM® Rational® ソフトウェア・サポートでは、製品の問題解決に関する支援を行っています。

前提条件

IBM Rational ソフトウェア・サポートに問題を送信するには、有効な Passport Advantage® ソフトウェア保守契約が必要です。パスポート・アドバンテージは、IBM の包括的ソフトウェア・ライセンスおよびソフトウェア保守 (製品のアップグレードおよび技術支援) オファリングです。次のサイトからオンラインでパスポート・アドバンテージに登録できます。http://www.ibm.com/software/lotus/passportadvantage/howtoenroll.html

- パスポート・アドバンテージについて詳しくは、パスポート・アドバンテージ FAQ（http://www.ibm.com/software/lotus/passportadvantage/brochures_faqs_quickguides.html）にアクセスしてください。
- さらに支援が必要な場合は、IBM 担当員に連絡してください。

問題をオンラインで (IBM Web サイトから) IBM Rational ソフトウェア・サポートに送信するには、さらに以下が必要です。

- IBM Support Web サイトの登録ユーザーであること。登録について詳しくは、http://www-01.ibm.com/software/support/ を参照してください。
- 許可された呼び出し元としてサービス要求ツールにリストされていること。

問題報告について

次のようにして、IBM Rational ソフトウェア・サポートに問題を送信します。

1. お客さまの問題のビジネス・インパクトを判別します。IBM へ問題を報告する際は、重大度レベルを問われます。そのため、報告する問題とそのビジネス・インパクトを理解して、評価する必要があります。

   重大度のレベルを決めるにあたっては、下表を参照してください。

重大度	説明
1	問題は**危機的な**ビジネス・インパクトを持ちます。プログラムを使用できず、業務に重大な影響が出ています。この状況には、即時に解決策が必要とされます。
2	問題は、**重大な**ビジネス・インパクトを持ちます。プログラムは使用可能ですが、非常に限定されています。

重大度	説明
3	問題は**部分的な**ビジネス・インパクトを持ちます。プログラムは使用可能ですが、比較的重要でない（業務に大きな影響はない）機能が利用できません。
4	問題は**わずかな**ビジネス・インパクトを持ちます。問題による業務への影響がほとんどないか、問題に対する有効な回避策が実施済みです。

2. 問題を説明して、背景情報を収集します。IBM に問題を説明する際は、なるべく具体的に説明してください。IBM Rational ソフトウェア・サポートの専門家が、問題を解決するために効果的な支援をできるように、関連するすべての背景情報を含めてください。時間を節約するために、以下の質問の答えを用意してください。
   - 問題の発生時に実行していたソフトウェア（複数可）のバージョンは何ですか？

     次のオプションを使用して、正確な製品名とバージョンを判別することができます。

     IBM Installation Manager を始動して、「ファイル」>「インストール済みパッケージの表示」を選択します。パッケージ・グループを展開し、パッケージを選択して、パッケージ名およびバージョン番号を確認します。

     製品を始動して、「ヘルプ」>「製品情報」をクリックし、オファリング名とバージョン番号を確認します。
   - オペレーティング・システムおよびバージョン番号（サービス・パックまたはパッチを含む）は何ですか？
   - 問題の症状に関連するログ、トレース、およびメッセージはありますか？
   - 問題を再現できますか？再現できる場合は、問題を再現するための手順は何ですか？
   - システムに変更を加えましたか？例えば、ハードウェア、オペレーティング・システム、ネットワーキング・ソフトウェア、またはその他のシステム・コンポーネントに変更を加えましたか？
   - 現在、問題に対する何らかの回避策を使用していますか ？ 使用している場合は、問題の報告時にその回避策も説明する準備をお願いします。
3. IBM Rational ソフトウェア・サポートに問題を送信します。次の方法で、IBM ソフトウェア・サポートに問題の送信ができます。
   - **オンラインの場合：** IBM Rational ソフトウェア・サポートの Web サイト (https://www.ibm.com/software/rational/support/) にアクセスして、Rational サポート・タスク・ナビゲーターで「**サービス要求を開く (Open Service Request)**」をクリックします。エレクトロニック問題報告ツールを選択し、「問題管理レコード (PMR) (Problem Management Record (PMR))」を開き、問題についてご自身の言葉で正確に記述してください。

サービス要求を開く方法について詳しくは、http://www.ibm.com/software/support/help.html にアクセスしてください。

IBM Support Assistant を使用してオンラインのサービス要求を開くこともできます。詳しくは、http://www-01.ibm.com/software/support/isa/faq.html を参照してください。

- **電話の場合**：国または地域別の電話番号を調べるには、http://www.ibm.com/planetwide/ の「IBM directory of worldwide contacts」で、お住まいの国名または地域名をクリックします。
- **IBM 担当員に依頼する場合**：オンラインまたは電話で IBM Rational ソフトウェア・サポートにアクセスできない場合は、IBM 担当員に連絡してください。必要な場合は、お客さまに代わって、IBM 担当員がサービス要求を開くことができます。http://www.ibm.com/planetwide/ で、各国への詳しい連絡先情報を検索できます。

送信した問題が、ソフトウェアの障害に関するものか、資料の欠落や不正確な記述によるものである場合は、IBM ソフトウェア・サポートはプログラム診断依頼書 (APAR) を作成します。APAR には、問題の詳細が記述されます。IBM ソフトウェア・サポートは可能な限り、APAR が解決されてフィックスが提供されるまでの間に実施できる回避策を提供します。IBM は、同一の問題を経験している他のユーザーが同じ解決方法を利用できるように、ソフトウェア・サポート Web サイトに解決済みの APAR を公開し、毎日更新しています。

コマンドと引数の構文

以下のように、コマンドを入力して Rational Synergy ツールを実行できます。

- 各ユーザー コマンドの先頭には、コマンド接頭辞 `ccm` が付きます。以下のように、ユーザー コマンドを個々に入力します。

  ```
  ccm dir
  ```

- Windows および UNIX では、一部の管理コマンドには接頭辞 `ccmdb` と `ccmsrv` を使用します。

  Windows の管理コマンドについては、『Rational Synergy 管理者ガイド Windows 版』を参照してください。UNIX 管理コマンドは、以下のマニュアルで説明しています。

- UNIX では、一部の管理コマンドには接頭辞 `ccm_` を使用します。管理コマンドは以下のように入力します。

  ```
  ccm_install
  ```

  UNIX の管理コマンンドトについては、『Rational Synergy 管理者ガイド UNIX 版』を参照してください。Oracle データベースのユーザーは『Rational Synergy Administration Guide for UNIX (Oracle)』を参照してください。

多くのコマンドでは、Synergy データベース内の複数のオブジェクトの指定を行うことができます。最も一般的な指定はファイルの指定で、データベース内のプロジェクト、ディレクトリ、またはファイルを指定します。

このセクションでは以下の指定について説明します。

- ベースラインの指定
- 変更依頼の指定
- データベースの指定
- ファイルの指定
- フォルダの指定
- フォルダ テンプレートの指定
- オブジェクトの指定
- プロセスの指定
- プロセス ルールの指定
- プロジェクトの指定
- プロジェクト グルーピングの指定
- タスクの指定
- 転送セットの指定

それぞれの指定には、以下のグローバル形式を使用できます。

- オブジェクト名形式
- クエリ選択セット参照形式
- Cvid 参照形式
- ファイル内容形式

オブジェクト名形式

データベース内のオブジェクトは、4 部名称を使用して表すことができます。

- *name:version:type:instance*

  たとえば、`ClientSessionContext.java:23:java:J#1` は、`ClientSessionContext.java` という名前のオブジェクトで、バージョン `23`、タイプ `java`、インスタンス `J#1` のものを意味します。

- *name version_delimiter version:type:instance*

  version_delimiter は、データベース内で使用している現在のバージョン区切り文字です。たとえば、デフォルトのバージョン区切り文字 `-`（ハイフン）を使用した指定 `ClientSessionContext.java-23:java:J#1` は、`ClientSessionContext.java` という名前のオブジェクトで、バージョン `23`、タイプ `java`、インスタンス `J#1` のものを意味します。

  > **注記**: allow_delimiter_in_name を有効にして上記の 2 番目の形式を使用した場合、Rational Synergy は、一番右のバージョン区切り文字に続く部分から **version** フィールドを決定します。たとえば、区切り文字としてハイフンを使用した場合、指定 `my-file-23:ascii:1` は、名前 `my-file` とバージョン `23` を表示します。

関連トピック

- クエリ選択セット参照形式
- Cvid 参照形式
- ファイル内容形式

クエリ選択セット参照形式

クエリやオブジェクトの一覧表示を行うコマンドでは、クエリ選択セットが設定されます。選択セットは、以前に実行したクエリの結果や以前に実行したコマンドの結果の表示順序を保持しています。出力には、デフォルトで番号が付けられています。選択セット内のオブジェクトは以下の方法で参照できます。

- *@n*

  ここで、*n* は、前のクエリで表示されたオブジェクトの番号です。たとえば、@1 は最初のオブジェクト、@2 は 2 番目のオブジェクトとなります。

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

  ここで、*n* と *m* は前のクエリで表示されたオブジェクトの番号で、*m* は *n* と等しいか大きい値です。これは、n 番目から m 番目のオブジェクトを意味します。たとえば、@2-5 は、2 番目から 5 番目のオブジェクトを意味し、@3-@6 は 3 番目から 6 番目のオブジェクトを意味します。

- *@*

  クエリ選択セット内のすべてのオブジェクトを意味します。

関連トピック

- オブジェクト名形式
- Cvid 参照形式
- ファイル内容形式

Cvid 参照形式

Synergy データベース内の各オブジェクトは、*cvid* という一意の整数インデックスを持ちます。これらのオブジェクトについては、以下の形式でその cvid を使用して参照できます。

- @=*n*

  ここで *n* はオブジェクトの cvid です。たとえば、@=12345 は、cvid が 12345 のオブジェクトを意味します。

関連トピック

- オブジェクト名形式
- クエリ選択セット参照形式
- ファイル内容形式

ファイル内容形式

ファイル内容形式を使うと、指定を行う際に 0 個、1 個または複数の適切な指定を含んだファイルを使用できます。

- `@:file_path`

  ここで、*`file_path`* は、ゼロ、1 つまたは複数の指定を含むファイルへのパスです。たとえば、`@:myobjects.txt` は、現在のディレクトリ内の `myobjects.txt` という名前のファイルから指定を得ます。

  たとえば、@:myobjects.txt では、現在のディレクトリ内の myobjects.txt という名前のファイルから指定を取り出すことを意味します。

  > **注記**：ファイル内容形式で指定したファイルには、ファイル内容形式による指定を含めることはできません。

関連トピック

- オブジェクト名形式
- クエリ選択セット参照形式
- Cvid 参照形式

ベースラインの指定

baseline_spec は、Rational Synergy データベース内の 1 つまたは複数のベースライン オブジェクトを示します。*baseline_spec* は、以下のいずれかの形式をとることができます。

- ベースライン オブジェクトに設定されている以下のグローバル形式のいずれか。
  - オブジェクト名形式
  - クエリ選択セット参照形式
  - Cvid 参照形式
  - ファイル内容形式
- *baseline_name*

  現在のデータベースで作成されたベースラインの名前です。たとえば、`Client build 46` は、現在のデータベースで作成された `Client build 46` という名前のベースラインを示します。
- *datasase_id dcm_delimiter baseline_name*

  ここで、*database_id* はベースラインが作成された DCM データベース ID、*dcm_delimiter* は現在の DCM 区切り文字 (デフォルトは `#`)、*baseline_name* はベースラインの名前です。たとえば、デフォルトの DCM 区切り文字 `#` を使用した場合、ベースライン指定 `A#api build 67` は、データベース `A` で作成された `api build 67` という名前のベースラインを示します。

変更依頼の指定

change_request_spec は、1 つまたは複数の変更依頼の参照です。*change_request_spec* は以下の任意の形式をとることができます。

- 変更依頼オブジェクトに設定されている以下のグローバル形式のいずれか。
  - オブジェクト名形式
  - クエリ選択セット参照形式
  - Cvid 参照形式
  - ファイル内容形式
- *change_request_id*

  変更依頼識別子によって識別される 1 つの変更依頼を示します。DCM 用に初期化されたデータベースでは、以下のいずれかの形式になります。DCM 用に初期化されていないデータベースでは、1 番目の形式のみ有効です。

  - *change_request_number*

    現在のデータベース内に作成された指定番号の変更依頼を示します。たとえば、34 は、現在のデータベース内に作成された変更依頼 34 を示します。

  - *database_id dcm_delimiter change_request_number*

    ここで、*database_id* は変更依頼が作成された DCM データベース ID、*dcm_delimiter* は現在の DCM 区切り文字（デフォルトは #）、*change_request_number* は変更依頼番号です。たとえば、B#543 は、データベース B 内に作成された変更依頼番号 543 を示します。

- *change_request_id(,change_request_id)...*

  カンマで区切られた *change_request_id* 要素のリストとして指定した、一連の変更依頼を示します。たとえば、B#543,23 は、データベース B 内に作成された変更依頼 543 と現在のデータベース内に作成された変更依頼 23 を示します。

- *change_request_id-change_request_id*

  範囲指定した一連の変更依頼を示します。範囲は、最初の *change_request_id* で指定した変更依頼から 2 番目の *change_request_id* で指定した変更依頼までを含みます。

  範囲の最初の変更依頼と最後の変更依頼は、同じデータベース内にある必要があります。たとえば、34-37 は、現在のデータベース内に作成された変更依頼 34、35、36、37 を示します。しかし、範囲 A#34-B#37 は、範囲の最初の変更依頼がデータベース A にあり、最後の変更依頼がデータベース B にあるため無効です。

データベースの指定

database_spec は、1 つまたは複数の DCM データベース定義を示します。DCM と DCM データベース定義の詳細については、Rational Synergy Distributed のドキュメントを参照してください。以下のいずれかの形式をとることができます。

- DCM データベース定義オブジェクトに設定されている以下のグローバル形式のいずれか。
  - オブジェクト名形式
  - クエリ選択セット参照形式
  - Cvid 参照形式
  - ファイル内容形式
- *database_id*

  ここで、*database_id* は、DCM データベース定義に関連付けられた DCM データベース ID です。たとえば、`IRVJ` はデータベース ID `IRVJ` の DCM データベース定義を示します。

ファイルの指定

file_spec は、Rational Synergy データベース内の 1 つまたは複数のプロジェクト、ディレクトリ、またはファイルを示します。以下のいずれかの形式をとることができます。

- プロジェクト、ディレクトリ、またはファイルに設定されている以下のグローバル形式のいずれか。
  - オブジェクト名形式
  - クエリ選択セット参照形式
  - Cvid 参照形式
  - ファイル内容形式
- ワークエリア参照形式

  管理プロジェクト ワークエリア内のオブジェクトをそのワークエリア内のパスで参照します。
- プロジェクト参照形式

  これは、オブジェクトを指定プロジェクト内の相対パスで参照します。

オブジェクト名には最大 155 文字、オブジェクト バージョンには最大 32 文字を指定できます。

ワークエリア参照形式

オブジェクト バージョンがプロジェクトのメンバーであり、プロジェクトがファイル システム内のワークエリア ディレクトリで同期している場合は、プロジェクトのディレクトリ構造内のパスによってオブジェクト バージョンを参照できます。

以下に Windows パスを使用した例を示します。UNIX の場合も機能は同じです。たとえば、オブジェクト バージョン `foo.c-4` がディレクトリ `dir1` 内のプロジェクト `jobA-1` のメンバーであり、プロジェクトのワークエリアが `c:¥users¥joe¥ccm_tutorial` である場合、`foo.c-4` のワークエリア参照形式は以下のようになります。

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

現在の作業ディレクトリが `c:¥users¥joe¥ccm_tutorial¥jobA-1¥jobA` の場合は、以下の相対パスを使用してオブジェクト バージョン `foo.c-4` を参照できます。

```
dir1¥foo.c
```

以下のようにワークエリア参照にバージョンを追加して、オブジェクトの他のバージョンを参照するよう変更できます。

path`¥`*object_name*`[:`*version*`]`

path`¥`*object_name*`[`*version_delimiter version*`]`

このファイル指定を使用して、オブジェクト バージョンの他のバージョンを参照します。

たとえば、foo.c-4 が現在のプロジェクトのメンバーであり、先行バージョン foo.c-3 がある場合、foo.c-4 に属する相対パスを使用してその先行バージョンを参照できます。

```
dir1¥foo.c-3
```

> **注記**：allow_delimiter_in_name を有効にして、version_delimiter を使用する 2 番目の形式を使用すると、バージョン区切り文字を含む全フィールドが名前と見なされ、バージョンはプロジェクト内で使用されている現在のバージョンになります。たとえば、バージョン区切り文字 - を使用すると、指定 c:¥users¥joe¥ccm_tutorial¥jobA-1¥jobA¥dir1¥foo.c-23 によって名前 foo.c-23 が得られます。これは、ワークエリア内で使われているオブジェクトの現在のバージョンを示します。
> allow_delimiter_in_name を使用するときは、パスと名前の後に :version 形式を使用する任意の明示的なバージョンを指定して混乱を避けてください。

プロジェクト参照形式

プロジェクト参照形式は、プロジェクト、ディレクトリ、またはファイルを指定プロジェクト内の相対パスで参照します。これは、プロジェクトが管理ワークエリアを持たない場合、あるいはそのワークエリアがクライアントから見えない場合でも使用できます。

relative_path@project_spec

relative_path:version@project_spec

relative_path version_delimiter version@project_spec

ここで、*relative_path* ファイル区切り文字として **/** または **¥** を使用できる相対パス、*project_spec* はプロジェクトの指定、*version_delimiter* は現在のバージョン区切り文字（デフォルトはハイフン -）、*version* はオブジェクトのバージョン（オプション）です。バージョンを指定しない場合、参照されるプロジェクトはプロジェクトで使用されるバージョンとなります。

相対パスは、プロジェクトの最上位を基準とします。たとえば、指定 myproject¥src¥ClientContext.java@myproject:1 は、プロジェクト myproject:1 のルート ディレクトリ myproject の下の src というディレクトリの下の ClientContext.java というファイルの現在のバージョンを示します。

相対パスの後にオプションとしてバージョンを指定できます。たとえば、指定 myproject¥src¥ClientContext.java:23@myproject:1 は、プロジェクト myproject:1 のルート ディレクトリ myproject の下の src というディレクトリの下の ClientContext.java というファイルのバージョン 23 を示します。

注記：allow_delimiter_in_name を有効にして、*version_delimiter* を使用する最後の形式を使用すると、バージョン区切り文字を含む全フィールドが名前と見なされ、バージョンはプロジェクト内で使用されている現在のバージョンになります。たとえば、バージョン区切り文字 **-** を使用した場合、指定 `myproject¥src¥ClientContext.java-23@myproject:1` によって名前 `ClientContext.java-23` が得られます。これは、プロジェクト内で使われているオブジェクトの現在のバージョンを示します。`allow_delimiter_in_name` を使用するときは、パスと名前の後に `:version` 形式を使用する任意の明示的なバージョンを指定して混乱を避けてください。

フォルダの指定

folder_spec は、Rational Synergy データベース内の 1 つまたは複数のフォルダを示します。以下のいずれかの形式をとることができます。

- フォルダ オブジェクトに設定されている以下のグローバル形式のいずれか。
  - オブジェクト名形式
  - クエリ選択セット参照形式
  - Cvid 参照形式
  - ファイル内容形式
- *folder_id*

  フォルダ識別子によって識別される 1 つのフォルダを示します。DCM 用に初期化されたデータベースでは、以下のいずれかの形式になります。DCM 用に初期化されていないデータベースでは、1 番目の形式のみ有効です。

  - *folder_number*

    現在のデータベース内に作成された指定番号のフォルダを示します。たとえば、34 は、現在のデータベース内に作成されたフォルダ 34 を示します。

  - *database_id dcm_delimiter folder_number*

    ここで、*database_id* はフォルダが作成された DCM データベース ID、*dcm_delimiter* は現在の DCM 区切り文字（デフォルトは #）、*folder_number* はフォルダの番号です。たとえば、B#543 は、データベース B 内に作成されたフォルダ番号 543 を示します。

- *folder_id(,folder_id)...*

  カンマで区切られた *folder_id* 要素のリストとして指定された、一連のフォルダを示します。たとえば、B#543,23 は、データベース B 内に作成されたフォルダ 543 と現在のデータベース内に作成されたフォルダ 23 を示します。

- *folder_id-folder_id*

  範囲指定した一連のフォルダを示します。範囲は、最初の *folder_id* で指定したフォルダから 2 番目の *folder_id* で指定したフォルダまでを含みます。範囲の最初のフォルダと最後のフォルダは、同じデータベース内にある必要があります。たとえば、34-37 は、現在のデータベース内に作成されたフォルダ 34、35、36、37 を示します。しかし、A#34-B#37 は、最初のフォルダがデータベース A にあり、最後のフォルダがデータベース B にあるため無効です。

フォルダ テンプレートの指定

`folder_template_spec` は、Rational Synergy データベース内の 1 つまたは複数のフォルダ テンプレートを示します。以下のいずれかの形式をとることができます。

- フォルダ テンプレート オブジェクトに設定されている以下のグローバル形式のいずれか。
  - オブジェクト名形式
  - クエリ選択セット参照形式
  - Cvid 参照形式
  - ファイル内容形式
- *`folder_template_name`*

  フォルダ テンプレートをその名前で指定します。たとえば `All completed tasks for %release` は、**All completed tasks for %release** という名前のフォルダ テンプレートを示します。

オブジェクトの指定

object_spec は、Rational Synergy データベース内の 1 つまたは複数のオブジェクトを示します。以下のいずれかの形式をとることができます。

- 以下のいずれかのグローバル形式。
  - オブジェクト名形式
  - クエリ選択セット参照形式
  - Cvid 参照形式
  - ファイル内容形式
- ワークエリア参照形式

  ブジェクト 管理プロジェクト ワークエリア内のオブジェクトをそのワークエリア内のパスで参照します。
- プロジェクト参照形式

  これは、オブジェクトを指定プロジェクト内の相対パスで参照します。

object_spec は、ファイルの指定と同じ構文をサポートしますが、プロジェクト、ディレクトリ、ファイルのみならず、Synergy データベース内の任意のオブジェクトを参照できます。

プロセスの指定

process_spec は、Rational Synergy データベース内の 1 つまたは複数のプロセス定義を示します。以下のいずれかの形式をとることができます。

- プロセス定義オブジェクトに設定されている以下のグローバル形式のいずれか。
  - オブジェクト名形式
  - クエリ選択セット参照形式
  - Cvid 参照形式
  - ファイル内容形式
- *process_name*

  プロセスをその名前で指定します。たとえば `Standard` は、**Standard** という名前のプロセスを示します。

プロセス ルールの指定

process_rule_spec は、Rational Synergy データベース内の 1 つまたは複数のプロセス ルールを示します。以下のいずれかの形式をとることができます。

- プロセス ルール オブジェクトに設定されている以下のグローバル形式のいずれか。
  - オブジェクト名形式
  - クエリ選択セット参照形式
  - Cvid 参照形式
  - ファイル内容形式
- *generic_process_rule_name*

  汎用プロセス ルールをその名前で指定します。たとえば `Integration Testing` は、**Integration Testing** という名前の汎用プロセス ルールを示します。
- *release:generic_process_rule_name*

  リリース固有プロセス ルールを指定します。*release* は有効なリリースで、*generic_process_rule_name* はその作成元の汎用プロセス ルールの名前です。たとえば、`1.0:Integration Testing` は **Integration Testing** 汎用プロセス ルールから作成されたリリース `1.0` のリリース固有プロセス ルールを示します。

プロジェクトの指定

project_spec は、Rational Synergy データベース内の 1 つまたは複数のプロジェクトを示します。以下のいずれかの形式をとることができます。

- プロジェクト オブジェクトに設定されている以下のグローバル形式のいずれか。
  - オブジェクト名形式
  - クエリ選択セット参照形式
  - Cvid 参照形式
  - ファイル内容形式
- *name:version*
  name version_delimiter version

  ここで、*name* はプロジェクトの名前、*version* はプロジェクトのバージョン、*version_delimiter* は現在のバージョン区切り文字（デフォルトはハイフン **-**）です。これは、現在のデータベース内に作成されたデフォルトのインスタンスを持つプロジェクトを指定します。DCM 用に初期化されていないデータベースでは、デフォルト インスタンスは `1` です。たとえば、`myproject:23` はプロジェクト `myproject:23:project:1` を示します。DCM データベース ID `A` と DCM 区切り文字 `#` の、DCM 用に初期化されたデータベースでは、デフォルト インスタンスは `A#1` になります。たとえば、`myproject:23` はプロジェクト `myproject:23:project:A#1` を示します。
- *name:version:project:instance*
  name version_delimiter version:project:instance

  ここで、*name* はプロジェクトの名前、*version* はプロジェクトのバージョン、*instance* はプロジェクトのインスタンス、*version_delimiter* は現在のバージョン区切り文字（デフォルトはハイフン **-**）です。たとえば、`myproject:23:project:A#1` はバージョン `23` とインスタンス `A#1` を持つ `myproject` という名前のプロジェクトを示します。

プロジェクト グルーピングの指定

project_grouping_spec は、Rational Synergy データベース内の 1 つまたは複数のプロジェクト グルーピングを示します。以下のいずれかの形式をとることができます。

- プロジェクト グルーピング オブジェクトに設定されている以下のグローバル形式のいずれか。
  - オブジェクト名形式
  - クエリ選択セット参照形式
  - Cvid 参照形式
  - ファイル内容形式
- *project_grouping_displayname*

  これは、プロジェクト グルーピングをその表示名で指定します。現在の所有者のプロジェクト グルーピングでは、以下の形式が可能です。

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

  たとえば、My client/2.0 Integration Testing Projects は、自分が所有するリリース **client/2.0** の目的 **Integration Testing** を持つプロジェクト グルーピングを示します。

  任意のユーザーが所有するプロジェクト グルーピングでは、以下の形式が可能です。

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

  たとえば、Linda's client/2.0 Integration Testing Projects は、*Linda* が所有するリリース **client/2.0** の目的 **Integration Testing** を持つプロジェクト グルーピングを示します。

リリースの指定

release_spec は、Rational Synergy データベース内の 1 つまたは複数のリリース定義を示します。以下のいずれかの形式をとることができます。

- リリース定義オブジェクトに設定されている以下のグローバル形式のいずれか。
  - オブジェクト名形式
  - クエリ選択セット参照形式
  - Cvid 参照形式
  - ファイル内容形式
- *release_name*

  リリースをそのリリース値で指定します。たとえば、リリース指定 `client/2.0` は、リリース **client/2.0** のリリース定義を示します。

タスクの指定

task_spec は、Rational Synergy データベース内の 1 つまたは複数のタスクを示します。以下のいずれかの形式をとることができます。

- タスクに設定されている以下のグローバル形式のいずれか。
  - オブジェクト名形式
  - クエリ選択セット参照形式
  - Cvid 参照形式
  - ファイル内容形式
- *task_id*

  タスク識別子によって識別される 1 つのタスクを示します。DCM 用に初期化されたデータベースでは、以下のいずれかの形式になります。DCM 用に初期化されていないデータベースでは、1 番目の形式のみ有効です。

  - task_number

    現在のデータベース内に作成された指定番号のタスクを示します。たとえば , 34 は、現在のデータベース内に作成された タスク 34 を示します。

  - database_id dcm_delimiter task_number

    ここで、*database_id* はタスクが作成された DCM データベース ID、*dcm_delimiter* は現在の DCM 区切り文字（デフォルトは #）、*task_number* はタスク番号です。たとえば、B#543 は、データベース B 内に作成されたタスク番号 543 を示します。
- *task_id(,task_id)...*

  カンマで区切られた *task_id* 要素のリストとして指定された、一連のタスクを示します。たとえば、B#543,23 は、データベース B 内に作成されたタスク 543 と現在のデータベース内に作成されたタスク 23 を示します。
- *task_id-task_id*

  範囲指定した一連のタスクを示します。範囲は、最初の *task_id* で指定したタスクから 2 番目の *task_id* で指定したタスクまでを含みます。範囲の最初のタスクと最後のタスクは同じデータベース内にある必要があります。たとえば、34-37 は、現在のデータベース内に作成されたタスク 34、35、36、37 を示します。しかし、A#34-B#37 は、最初のタスクがデータベース A にあり、最後のタスクがデータベース B にあるため無効です。

転送セットの指定

transfer_set_spec は、Rational Synergy データベース内の 1 つまたは複数の転送セットを示します。以下のいずれかの形式をとることができます。

- 転送セットに設定されている以下のグローバル形式のいずれか。
  - オブジェクト名形式
  - クエリ選択セット参照形式
  - Cvid 参照形式
  - ファイル内容形式
- *transfer_set_name*

  転送セットをその名前で示します。たとえば,`Entire Database` は、定義済み **Entire Database** 転送セットを示します。

グローバル フォーマット オプション

Rational Synergy CLI コマンドでは、情報の提示、フォーマット、およびグループ化の方法を定義できます。以下に、すべてのフォーマット オプションを示します。

- フォーマット文字列の高度な使用方法
- カラム配置
- カラム フォーマット要素
- カラムヘッダー
- キーワード
- 番号付け
- プロパティのフォーマット
- ソートとグループ化

すべてのコマンドがすべてのフォーマット オプションをサポートしているわけではありません。各サブコマンドのコマンド構文で、サポートされているフォーマット オプションを示しています。

これらのオプションを盛り込んだ出力の例については、このセクションの最後にあるフォーマットの使用例を参照してください。以下のフォーマット オプションを使用できます。

- -ch|-column_headers
- -nch|-nocolumn_headers
- -f|-format
- -nf|-noformat
- -sby|-sortby
- -ns|-nosort
- -gby|-groupby
- -sep|-separator
- -u|-unnumbered

フォーマット文字列

フォーマット文字列は、いくつかの文字列リテラル、いくつかのキーワード、または文字列リテラルとキーワードの組み合わせから構成されます。処理の結果、フォーマット文字列はコマンド出力中に展開された固定の文字列として表されます。たとえば、フォーマット文字列 `"Release=%release"` は、文字列リテラル `"Release="` とキーワード `%release` の 2 つの部分で構成されています。各オブジェクトについて、文字列リテラルが出力に表示され、キーワードはオブジェクトの release プロパティの文字列表現で置き換えられます。たとえば、3 つのオブジェクトについてレポートするコマンドの出力は以下のようになります。

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

カラム配置

ほとんどのコマンドの出力は、デフォルトでカラム配置が行われます。フォーマット文字列は、デフォルトの区切り文字（空白）を使用してカラムに配分されます。

たとえば、`"%name Release=%release"` のフォーマット文字列は、以下のカラムに配分されます。

- 最初のカラムは 1 つの要素、`%name` キーワードを含む。
- 2 番目のカラムは以下の 2 つの要素を含む。

  * 文字列リテラル `"Release="`

  * キーワード `%release`

3 つのオブジェクトについてレポートするコマンドにより、たとえば以下の出力が得られます。

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

出力例では、最初のカラムに空白を入れて 2 番目のカラムを揃えて配置しています。配置は、端末ディスプレイ装置を想定して計算され、等幅フォントを使用しています。カラムが正しく配置されて見えない場合、ディスプレイ装置でプロポーショナル フォントではなく固定スペース フォントを使用していることを確認してください。配置を正しくするには、`ccm.cli.format.font` をディスプレイ装置の固定スペース フォントに設定する必要があるかもしれません。

復帰改行文字により、フォーマット文字列は現在のカラムで終り、復帰改行文字に続くテキストは次の行の最初のカラムになります。

-sep|-separator

`-sep|-separator` *`separator`* オプションでは異なる区切り文字を指定できます。たとえば、フォーマット文字列 `"%name|Release %release"` があるとします。デフォ

ルトの区切り文字では、フォーマット文字列には以下のように 2 つのカラムがあります。

- 最初のカラムは以下の 2 つの要素を含む。

  * キーワード `%name`

  * 文字列リテラル `"|Release"`

- 2 番目のカラムはキーワード `%release` を含む。

3 つのオブジェクトについてレポートするコマンドにより、たとえば以下の出力が得られます。

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

次に、同じフォーマット文字列でオプション `-sep "|"` を指定した例を示します。フォーマット文字列は、パイプ区切り文字を使用してカラムに配分されます。

- 最初のカラムはキーワード `%name` を含む。
- 2 番目のカラムは以下の 2 つの要素を含む。

  * 文字列リテラル `"Release "`

  * キーワード `%release`

同じデータですが、この結果は以下のように出力されます。

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

-nf|-noformat

`-nf|-noformat` オプションは、カラム配置を使用しないよう指定します。このオプションを指定すると、フォーマット文字列はすべての要素を含む 1 つのカラムとして解釈されます。このオプションにより、区切り文字（カンマなど）で区切られた出力を作成します。

フォーマット文字列の高度な使用方法

これまで説明した例では、フォーマット文字列は文字列リテラルか `%release` などの単純なキーワードでした。この方法でも必要なフォーマットを得るのに十分ですが、Synergy CLI クライアントは、よりきめ細かい制御と柔軟性を提供する高度な形式もサポートします。

すでに説明したように、フォーマット文字列は -nf|-noformat オプションを指定しない限りまずカラムに配分されます。各カラムは、以下の任意の要素の集合で構成されます。

- 文字列リテラル

これは、表示どおりに出力されます。出力に 1 つのパーセント記号 **%** を入れたい場合は、**%%** のようにペアで指定します。

- キーワード

  これは、オブジェクトに指定されたプロパティの値を表す文字列表現に置き換わります。

- カラム プロパティ フォーマット要素

  そのカラムの内容の全体的なフォーマットを制御します。詳細については、カラム フォーマット要素を参照してください。

キーワード

前の例で最も単純な形式のキーワードは、パーセント記号の後にプロパティの名前を付けたものでした。たとえば、`%release` は、`release` とい名前のプロパティの値の文字列表現に置き換わります。Synergy データベース内のオブジェクトの場合、プロパティ名は属性の名前か組み込み済みキーワードになります。

また、オブジェクトがデータベース エンティティではない場合もあります。たとえば、プロジェクト メンバーシップ コンフリクトは、コンフリクトを起こしているオブジェクト、親プロジェクト、コンフリクトのタイプ、関連するタスクなどのデータで構成されます。このデータを指定するには、`objectkey` キーワードを使用できます。

```
%[objectkey]propertyname
```

`objectkey` は、指定したプロパティが示すオブジェクトの一部を示します。たとえば、プロジェクト メンバーシップ コンフリクトでは、`object` オブジェクト キーはコンフリクトを起こしているファイルまたはディレクトリを示し、`project` オブジェクト キーは、コンフリクトに関連する親プロジェクトを示します。フォーマット文字列 `%[project]displayname %[object]release` は、各メンバーシップ コンフリクトに関連するプロジェクトに組み込まれている `displayname` プロパティとコンフリクトを起こしているファイルまたはディレクトリの `release` プロパティを示します。

どのオブジェクト キーがサポートされているかについては、各コマンドで説明しています。

Synergy CLI は、以下の任意の形式の上級キーワードをサポートしています。

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

ここで、*keywordspec* は以下のいずれかの形式です。

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

propertyname[*propertyformat*]
+*propertyname*[*propertyformat*]
-*propertyname*[*propertyformat*]
[*objectkey*]*propertyname*[*propertyformat*]
+[*objectkey*]*propertyname*[*propertyformat*]
-[*objectkey*]*propertyname*[*propertyformat*]

上記のそれぞれの形式で、*objectkey* は参照するオブジェクトを示し、*propertyname* は属性や組み込み済みキーワードなどプロパティの名前です。

propertyformat は、プロパティ値を文字列に変換してそれをフォーマットする方法を示します。詳細については、プロパティのフォーマットを参照してください。

substitutionstring を使用する形式は、指定したプロパティまたは属性が存在するかどうかに関わらず、展開したキーワードに代替文字列リテラルを使用します。

- -*substitutionstring* 形式は、プロパティが存在しない場合は指定した置き換え文字列を使用することを意味します。プロパティが存在する場合は、その値を使用します。
- +*substitutionstring* 形式は、プロパティが存在する場合は指定した置き換え文字列を使用し、存在しない場合はヌル値の文字列表現を使用することを意味します。

オプションの先頭の + または - は、デフォルトのソート順序を示します。詳細については、ソートとグループ化を参照してください。

例

キーワード `"%{release:-No release}"` はオブジェクトのリリース プロパティを示します。プロパティが存在する場合、その値が表示されます。プロパティが存在しない場合、文字列 `"No release"` が表示されます。

キーワード `"%{[project]release[truncate='20']}"` は、オブジェクトに関連するプロジェクトの `release` プロパティを示します。プロパティが存在する場合、その値が表示され、幅が 20 を超えると末尾が省略記号になります。

プロパティのフォーマット

キーワードがオブジェクトのプロパティを示す場合、その値は文字列に変換され、プロパティ フォーマット オプションを使用してフォーマットされます。たとえば、ブール値属性は、ブール値の TRUE または FALSE の値を表す文字列に変換されます。プロパティ フォーマットを指定しない場合、以下の表に示すように、プロパティ値は文字列に変換されます。

値の種類	変換
null	ほとんどのコマンドで、指定したプロパティが存在しない場合、"<Not available>" と表示されます。
boolean	ブール値の TRUE は "TRUE" と表示され、FALSE は "FALSE" と表示されます。
文字列（文字列またはテキスト属性）	文字列値は、末尾の復帰改行を削除して表示されます。
日付（時刻属性）	日付は日付文字列に変換されます。ccm.cli.format.date が定義されている場合、これがフォーマット文字列として使用されます。定義されていない場合、ユーザーの地域のデフォルト日付フォーマットが使用されます。
整数	整数は、10 進整数文字列として表示されます。
浮動小数点	浮動小数点は、10 進浮動小数点文字列として表示されます。
集合	プロパティ値が要素の集合である場合、値のフォーマットは各要素をプロパティとして、カンマで区切りられます。

キーワードまたはカラム フォーマット要素では、各プロパティ値をフォーマットするコントロールを指定できます。このためには、空白文字で区切ったゼロ、1 つまたは複数の下記項目からなる *propertyformat* を指定します。

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

ここで：

propertyFormatName は、サポートされるプロパティ フォーマット オプションの名前。

propertyFormatValue は、空白文字を含まない値。この値は、Java 正規表現規則で定義されている "word" 文字で構成されている必要があります。

singleQuotedPropertyFormatValue は、一重引用符（アポストロフィー）以外の任意の文字からなる文字列。

doubleQuotedPropertyFormatValue は、二重引用符以外の任意の文字からなる文字列。

例

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

これは、値が最大 100 文字に切り捨てられ、幅 20 のカラムに入るよう折り返されるよう指定します。

Rational Synergy は、以下のプロパティ フォーマット名をサポートします。

プロパティ フォーマット名	説明
null	プロパティが存在しないときに `null` 値を表すために使用する文字列を指定します。
false	ブール値の FALSE 値を表すために使用する文字列を指定します。
true	ブール値の TRUE 値を表すために使用する文字列を指定します。
format	プロパティ値をフォーマットするために使用するフォーマット文字列を指定します。文字列が "{0" を含む場合、フォーマットは標準 Java フォーマットおよびローカライゼーション機能である *`MessageFormat`* を使用します。文字列が "{0" を含まない場合、フォーマットは標準 Java 文字列フォーマット機能である `printf` を使用します。
dateformat	日付プロパティ値をフォーマットするために使用する日付フォーマット文字列を指定します。日付フォーマットは、標準 Java 日付フォーマッタ である *`SimpleDateFormat`* の有効な文字列でなければなりません。
list_begin	プロパティ値が集合である場合に使用する開始文字列を指定します。
list_end	プロパティ値が集合である場合に使用する終了文字列を指定します。
separator	プロパティ値が集合である場合に使用する項目区切りを指定します。
truncate	表示される文字列の最大長を指定します。文字列が切り捨てられる場合、文字列は切り捨て表示で終わり、指定された最大長となります。デフォルトで、切り捨て表示は省略記号（...）です。
truncate_each	集合である値の各要素の最大長を指定します。文字列が切り捨てられる場合、文字列は切り捨て表示で終わり、指定された最大長となります。デフォルトで、切り捨て表示は省略記号（...）です。
truncate_indicator	切り捨てられた表示値を示すために使用する文字列を指定します。デフォルトで、この値は省略記号（...）です。
wordwrap	最大カラム長を指定します。指定した幅より広い文字を折り返すことも指定します。
wrap	最大カラム長を指定します。指定した幅より広い文字を折り返すことも指定します。
keep_trailing_newlines	値が TRUE なら、出力中の改行に続く値を保持します。

プロパティ フォーマット名	説明
nocolumn	値が TRUE なら、カラム配置を無視することを指定します。
indent	プロパティ値の行インデント幅を指定します。値は、1 から 1000 までの整数でなければなりません。

カラム フォーマット要素

フォーマット文字列では、カラム フォーマット要素を指定してカラム内のデータのフォーマット方法を制御できます。カラム フォーマット要素を指定すると、カラム全体が指定したオプションを使用してフォーマットされます。カラム フォーマット要素は以下の形式をとります。

`%[`*`propertyformat`*`]`

ここで、*`propertyformat`* は、プロパティのフォーマットで説明しているプロパティ フォーマットです。

たとえば、パイプ文字 **|** を区切り文字に使用したフォーマット文字列 `"%name|%[wrap=20]%version %{release[truncate=30]}"` があるとします。フォーマット文字列は 2 つのカラムで構成されています。

最初のカラムは、`%name` キーワードを含みます。

2 番目のカラムは以下の 4 つの要素を含みます。

- カラムの文字を幅 20 で折り返すカラム フォーマット要素 `"%[wrap=20]"`
- キーワード `%version`
- 文字列リテラル `" "`
- リリース値を幅 30 に切り捨てて表示するキーワード `%{release[truncate=30]}`。

2 番目のカラムを処理すると、表示する文字列は以下のように決定されます。

リリース プロパティ値は、必要に応じて文字列に変換されます。

1. 文字列リテラル `" "` が追加されます。
2. リリース プロパティ値は、必要に応じて文字列に変換されます。
3. 必要に応じて、リリース文字列は最大幅 30 に切り捨てられます。
4. 切り捨てられる可能性のある文字列はカラムに追加されます。
5. カラム値全体は、幅 20 で折り返されます。

-f|-format

`-f|-format` *`format`* オプションは、使用するフォーマット文字列を指定します。これには、前のセクションで説明したフォーマット オプションを使用できます。省略す

ると、コマンドはサブコマンドに固有のデフォルト フォーマット文字列を使用します。

ソートとグループ化

ほとんどのコマンドでは、ソートまたはグループ化オプションを指定せずにソートを行うと、オブジェクトはデフォルトのソート アルゴリズムによってソートされます。選択セットを更新した場合、ソートした出力でのオブジェクトの報告順序が反映されます。

デフォルトのソート アルゴリズムは、フォーマット文字列とそのフォーマット文字列内のキーワードで示された実際のプロパティ値を使用します。たとえば、Synergy データベース内のオブジェクトにフォーマット文字列 `"%create_time %release %displayname"` を使用すると、以下のようにソートされます。

1. `create_time` 属性を昇順第 1 キーとして、実際の日付値（表示される日付文字列ではなく）を使用
2. `release` 属性を昇順第 2 キーに使用
3. `displayname` 組み込みキーワードを昇順第 3 キーに使用

デフォルトでは、各ソート キーによってそのプロパティ値が昇順にソートされます。これを変更するには、上級キーワード形式のいずれかを使用します。詳細については、キーワードを参照してください。たとえば、フォーマット文字列 `"%{-create_time} %{-release} %displayname"` は、以下のようにソートを行います。

1. `create_time` 属性を降順第 1 キーとして、実際の日付値（表示される日付文字列ではなく）を使用
2. `release` 属性を降順第 2 キーに使用
3. `displayname` 組み込みキーワードを昇順第 3 キーに使用

指定順に引数を処理するコマンドは、一般にソートを行わず、ソートおよびグループ化関連オプションをサポートしません。そのようなコマンドの例に `ccm properties` があります。

-ns|-nosort

`-ns|-nosort` オプションは、すべてのソートとグループ化を行わないよう指定します。このオプションは、オブジェクトの元の順序を維持するために使用します。

-sby|-sortby

`-sby|-sortby` *`sort_spec`* オプションは、フォーマット文字列に基づいてデフォルトのソート アルゴリズムを変更する出力の明示的なソートを指定します。*`sort_spec`* は、オプションでコマンド区切り文字を使用した以下の形式の 1 つ以上の項目のリストです。

propertyname
+propertyname
-propertyname
[*objectkey*]*propertyname*
+[*objectkey*]*propertyname*
-[*objectkey*]*propertyname*

先頭に **+** の付いた項目は昇順ソート キーとして使用されます。先頭に **-** の付いた項目は降順ソート キーとして使用されます。どちらも付いていない項目は昇順ソート キーとして使用されます。

例

- `create_time,release`

  昇順 `create_time` を第 1 キー、昇順 `release` を第 2 キーとしてソートを行います。

- `+create_time,release`

  降順 `create_time` を第 1 キー、昇順 `release` を第 2 キーとしてソートを行います。

- `+release-create_time`

  昇順 `release` を第 1 キー、降順 `create_time` を第 2 キーとしてソートを行います。

sort_spec にリストされるプロパティは、フォーマット文字列に表れる必要はありません。`-sby|-sortby` を指定すると、ソートはフォーマット文字列で参照しているプロパティに依存しません。

-gby|-groupby

グループ化はソートの一形式です。`-gby|-groupby` *groupformat* オプションを指定すると、*groupformat* フォーマット文字列で指定する値によって初期レベルのソートが行われます。同じ一連のグループ化データ値を持つオブジェクトは、1 つにまとめられ、出力で先頭にグループ ヘッダーが付きます。グループ ヘッダーは、指定された *groupformat* フォーマット文字列を展開して作成されます。詳細については、フォーマット文字列を参照してください。

たとえば、`-groupby "Release %release:"` を指定すると、すべてのオブジェクトはリリース プロパティ値を第 1 キーとしてソートされます。これは、他のすべてのソート キーに優先します。たとえば、コマンドがリリース値 **1.0** であるものを 2 つ、リリース値 **2.0** であるものを 3 つ、合計 5 つのオブジェクトを報告する場合、オブジェクトは以下のようにグループ化されます。

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

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

グループ化とソートの両方を使用する場合、まず groupformat で指定したソートキーを使用し、次に任意の *sort_spec* のソート キー、あるいは指定していない場合は *format* 文字列を使用します。たとえば、`-groupby "Release %release:"-sortby create_time` を指定した場合、ソートは release を昇順第 1 キーに使用し、`create_time` を昇順第 2 キーに使用して行われます。

グループ化、ソート、およびカラム ヘッダーの詳細な例については、フォーマットの使用例の 2 番目の例を参照してください。

カラムヘッダー

カラム ヘッダーは、フォーマット文字列から派生したタイトル文字列です。ヘッダー、出力の各カラムの上に見出しとして表示されます。たとえば、フォーマット 文字列 `"%name %release %create_time"` を指定すると、カラム 1、2、3 に対応するカラム ヘッダーはそれぞれ `Name`、`Release`、`Create_time` となります。

グループ化、ソート、およびカラム ヘッダーの詳細な例については、フォーマットの使用例の 2 番目の例を参照してください。

-ch|-column_headers

出力にカラム ヘッダーを使用するよう指定します。このオプションは、`-nf|-noformat` オプションと一緒には使用できません。

デフォルトでは、カラム ヘッダーを使用しません。

-nch|-nocolumn_headers

出力にカラム ヘッダーを使用しないよう指定します。ほとんどのコマンドで、これがデフォルト設定です。

番号付け

出力が Synergy データベース内のオブジェクトを表し、選択セットを設定するコマンドでは、デフォルトで番号付き出力を使用します。番号付けは、クエリ選択セット参照形式を使用してそのオブジェクトを示すために使用する、選択セット参照番号を反映します。

-u|-unnumbered

出力に番号を付けないよう指定します。このオプションは、出力および結果として得られる選択セットの順序には影響しません。

フォーマットの使用例

このセクションの例は、特定のコマンドの場合を示しているわけではありません。以下の表に、例で出力として使われているオブジェクトとプロパティのデータを示します。

displayname	task_synopsis	owner	release
101	Fix defect M#1234（不具合 M#1234 の修正）	fred	1.0
102	Implement sorting（ソートの実装）	joe	1.0
900	Fix defect M#12345（不具合 M#12345 を修正）	susan	1.0
901	Implement date formatting（データ フォーマットの実装）	john	1.0_patch
1000	Fix defect M#1357（不具合 M#1357 の修正）	fred	1.1
1001	Fix defect M#6523（不具合 M#6523 の修正）	fred	1.1
1002	Implement new property formats（新しいプロパティ フォーマットの実装）	susan	2.0
1003	Add grouping and extend sorting（グループ化の追加によりソートを拡張）	joe	2.0
1004	Fix defect J#1234（不具合 J#1234 の修正）	susan	1.1
1005	Extend grouping feature（グループ化機能の拡張）	john	2.0

- 以下に、番号付けをせず（`-u`）、ソートを行わない（`-nosort`）データの例を示します。したがって、出力はデータの元の順序で表示されます。`-format` 値によって、データは displayname、tasuk_synpsis、owner、release の 4 つのカラムに整理されます。`-noformat` を使用していないのでカラムは揃えて配置されます。

```
-u -nosort -format "%displayname %task_synopsis %owner %release"
```

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

- 以下に、デフォルトで番号付けされた出力の例を示します。また、以下のようになっています。
  - フォーマット文字列は、displayname と task_synopsis の 2 つのカラムを定義する。
  - グループ化は、owner プロパティを示すフォーマットを使用する。
  - ソートは、*release* 値の昇順で行う。*release* 値は出力には表示されず、ソートにのみ使用する。ソート順序は、owner を昇順第 1 キー（グループ化が優先）に使用し、release を昇順第 2 キーに使用する。
  - 同じ owner 値を持つオブジェクトは同じグループ ヘッダーの下にグループ化される。
  - カラム ヘッダーが表示される。

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

Owner:fred
 1) 101  Fix defect M#1234
 2) 1000 Fix defect M#1357
 3) 1001 Fix defect M#6523

Owner:joe
 4) 102  Implement sorting
 5) 1003 Add grouping and extending sorting

Owner:john
 6) 901  Implement date formatting
 7) 1005 Extend grouping feature

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

- 以下に、デフォルトで番号付けされた出力の例を示します。フォーマット文字列は、displayname、task_synopsis（値は、20 文字以降**切り捨てられる**）、release の 3 つのカラムを定義します。ソートは、`owner`（表示されないプロパティ）と `release` の昇順で行います。

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

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

  20 文字以降のテキストを切り捨てずに**折り返す**場合には、上記の例の `truncate=20` の代わりに `wrap=20` を使用します。出力は以下のようになります。

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

命名制限

このセクションでは、Rational Synergy のオブジェクト、リリース、データベースおよび DCM の命名制限について説明します。

オブジェクト名の制限

オブジェクト名には、制限された文字を**除き**、英数字と記号を自由に組み合わせて使用できます。

制限文字をバージョン区切り文字として使用することはできません。詳細については、Rational Synergy CLI ヘルプ、トラディショナル モードを参照してください。

以下に、Rational Synergy オブジェクトの命名制限の一部を示します。

- オブジェクト名には、8 ビット文字と 2 バイト文字（最上位ビットをセット）は使用できない。
- プロジェクト名にはタブは使用できない。Makefile 名には、タブと空白は使用できない。

下表に、その他の制限文字と制限の理由を示します。

制限文字	制限の理由
/	UNIX のパス区切り文字、内部区切り文字
¥	Windows のパス区切り文字、エスケープ文字
'	UNIX の引用符（開始）
"	Windows の引用符
:	Windows のドライブ名区切り文字、Rational Synergy のオブジェクト指定区切り文字
?	INFORMIX の 1 文字のワイルドカード、正規表現
*	INFORMIX の複数文字のワイルドカード、正規表現
[	INFORMIX の一致構文、正規表現
]	INFORMIX の一致構文、正規表現
@	Rational Synergy のオブジェクト指定区切り文字
-	Rational Synergy のバージョン区切り文字

以下の文字は、オブジェクト名の先頭文字には使用できません。

- ,（カンマ）
- +（プラス記号）
- -（ダッシュ）
- ~（チルダ）

リリース値の制限

Rational Synergy のリリース値は、以下の規則に従う必要があります。

- リリース値には、上記の表に示した制限文字は使用できない。
- 値 `none` と `as_is` は、`ccm checkout` など一部のコマンドでキーワードとして使用されるため、リリース値には使用できません。
- コンポーネント名は 64 文字以内で指定する。
- コンポーネント リリース値は 32 文字以内で指定する。

データベース名の制限

Rational Synergy のデータベース名は、以下の規則に従う必要があります。

- 同じデータベース サーバーを使用する 2 つのデータベースは、同じ名前を持つことはできない。名前は、完全データベース パスのリーフ ディレクトリです。
- データベース名に許される文字は、英字、数字、および下線のみ。
- 最初の文字は英字であること。
- データベース名は 18 文字以内で指定する。

> **注記**：Rational Synergy のデータベース名では、大文字と小文字は区別されません。

ベースライン名の制限

Rational Synergy のベースライン名は、以下の規則に従う必要があります。

ベースライン名は、デフォルトでは # 文字である DCM 区切り文字を含むことはできない。これは、DCM 初期化時またはその後にのみ変更できます。

DCM の制限

DCM データベースの名前は、以下の規則に従う必要があります。

DCM データベースのデータベース ID には以下の文字を使用できない。

制限文字	制限の理由
/	UNIX のパス区切り文字、内部区切り文字
¥	Windows のパス区切り文字、エスケープ文字
'	UNIX の引用符（開始）
"	Windows の引用符
:	Windows のドライブ名区切り文字、Rational Synergy のオブジェクト指定区切り文字
$	INFORMIX の 1 文字のワイルドカード、正規表現
?	INFORMIX の 1 文字のワイルドカード、正規表現
*	INFORMIX の複数文字のワイルドカード、正規表現
[	INFORMIX の一致構文、正規表現
]	INFORMIX の一致構文、正規表現
@	Rational Synergy のオブジェクト指定区切り文字
<space>	データベース ID を引用符で囲むことはできません。
#	Rational Synergy DCM の区切り文字、GNU makefile 内のコメント

上記にリストした制限文字のほか、データベース ID は 8 文字以内とし、名前 "probtrac" は使用できません。デフォルト設定では、バージョン区切り文字を含むデータベース ID は使用できません。ただし、これは allow_delimiter_in_name 属性で変更できます。

DCM データベース ID では、大文字と小文字が区別されます。小文字のデータベースを使用する DCM クラスタでは、DCM データベース ID は大文字／小文字にかかわらず一意でなければなりません。つまり、大文字／小文字だけで区別されるデータベース ID は使用できません。

英数字の a-z、A-Z、0-9 は DCM 区切り文字には使用できません。代替区切り文字として、"!"、"~"、"=" は使用できます。デフォルト DCM 区切り文字は **#** です。

大文字／小文字とファイル名制限のデータベース オプション

次に示す 2 つのデータベース オプション 大文字／小文字 と ファイル名の制限 は、Rational Synergy データベース内のオブジェクトに付与する名前に影響します。

注記： これらのオプションは、Rational Synergy CLI のトラディショナル モードを使用して変更できます。

大文字／小文字

Rational Synergy は、ファイル名で大文字／小文字を区別します。このオプション対応のキーワードにより、オブジェクト名の大文字／小文字を保持したり、Rational Synergy データベース内のオブジェクト名を小文字に変更できます。

データベースの大文字／小文字の設定を確認する場合は、以下のコマンドを入力します。

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

case オプションの変更方法については、該当する『Rational Synergy 管理者ガイド』の `ccmdb_info` コマンドの説明を参照してください。

ファイル名の制限

ファイル名の制限は、ファイル システムと Rational Synergy の両方の制限に依存します。デフォルト設定では、Rational Synergy データベースに最大 155 文字の名前を持つオブジェクト（ファイル、ディレクトリ、プロジェクト）を作成できます（使用できない記号については、コマンドと引数の構文を参照してください）。

データベースのファイル名制限のキーワードを確認する場合は、以下のコマンドを入力します。

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

ファイル名制限のキーワードの変更は、ユーザー *ccm_root* からのみ可能です。ファイル名制限モードの変更方法の詳細については、該当する『Rational Synergy 管理者ガイド』の `ccmdb info` コマンドの説明を参照してください。

組み込み済みキーワード

Rational Synergy には以下のキーワードが組み込まれています。これらのキーワードを使用して、コマンドラインのクエリ、リスト、表示操作、および GUI のクエリ操作からの出力フォーマットを制御できます。

> **注記**：属性名をキーワードとして使用することもできます。オブジェクトに関連する属性を一覧表示するには、`ccm attr` コマンドと `-list` オプションを使用します。

キーワード	説明
`%baseline`	プロジェクトのベースライン プロジェクトを返します。ベースラインが存在しない場合は、`<No baseline>` を返します。
`%change_request`	オブジェクトに関連する 1 つ以上の変更依頼を表示します。ファイルの場合、これらの変更依頼は関連タスク、およびそのタスクに関連付けられている変更依頼に基づいて決まります。
`%change_request_duplicates`	1 つの変更依頼と重複する変更依頼のリストを返します。
`%change_request_original`	重複している変更依頼の場合は、重複元の変更依頼を返します。
`%change_request_release`	オブジェクトに関連付けられている変更依頼のリリース プロパティを表示します。
`%change_request_status`	オブジェクトに関連付けられている 1 つ以上の変更依頼の状態を表示します。
`%change_request_synopsis`	オブジェクトに関連付けられている 1 つ以上の変更依頼の概要を表示します。
`%displayname`	ファイル、ディレクトリ、プロジェクトのデフォルトは *`name-version`* です。
`%fullname`	4 部名称を *`subsystem/cvtype/name/version`* 形式で返します。
`%has_`*`relationship`*	クエリ内のオブジェクトからの関係があるオブジェクトを表示します。
`%in_baseline`	プロジェクトがベースラインにある場合は、プロジェクトのベースラインの表示名を返します。
`%in_build`	プロジェクトがベースラインにある場合は、ベースラインのメンバーであるプロジェクトのベースライン ビルド番号を返します。

キーワード	説明 (続き)
%instance	オブジェクト名の %subsystem 部分の別名。
%is_*relationship*_of	クエリ内のオブジェクトへの関係があるオブジェクトを表示します。
%model	現在のモデル オブジェクトの %fullname を返します。
%objectname	オブジェクト名を *name-version*:*cvtype*:*subsystem* 形式で返します。
%problem_duplicates	1 つの問題と重複する問題のリストを返します。
%problem_original	重複している問題の場合は、重複元の問題を返します。
%purpose	プロジェクトの目的を表示します。
%requirement_id	タスクまたはオブジェクトに関連付けられたタスクに関連付けられた変更依頼に格納されている要求 ID を表示します。
%root	オブジェクトがルート ディレクトリ以外の場合は <no_root> 文字列を返し、ルート ディレクトリの場合はプロジェクトの %fullname を返します。
%sourcename	デフォルトはオブジェクト名です。
%states	空白で区切られた正式なオブジェクト状態を返します。
%task	このオブジェクトに関連付けられたタスク番号の、カンマで区切されたリストを返します。関連付けられたタスクがない場合は、<void> を返します。
%task_platform	このオブジェクトに関連付けられたタスクのプラットフォーム値の、カンマで区切られたリストを返します。関連付けられたタスクがない場合は、<void> を返します。
%task_release	このオブジェクトに関連付けられたタスクのリリース値の、カンマで区切られたリストを返します。関連付けられたタスクがない場合は、<void> を返します。
%task_status	このオブジェクトに関連付けられたタスク状態の、カンマで区切られたリストを返します。関連付けられたタスクがない場合は、<void> を返します。

キーワード	説明 (続き)
%task_subsystem	このオブジェクトに関連付けられたタスクのサブシステム（task_subsys）値の、カンマで区切られたリストを返します。関連付けられたタスクがない場合は、<void> を返します。
%task_synopsis	オブジェクトがタスクの場合は、task_synopsis 属性を返します。それ以外の場合は、このオブジェクトに関連付けられたタスクの、セミコロンで区切られた task_synopses を返します。関連付けられたタスクがない場合は、<void> を返します。
%type	オブジェクトのタイプを返します（cvtype 属性に格納されている）。

正規表現

- 通常文字

  正規表現内の通常文字はそれ自身と一致します。通常文字とは、以下に示す特殊文字以外の文字のことです。

  ```
  (  )  [  ]  ^  $  .  *  +  ?  |  ¥
  ```

- 特殊文字

  特殊文字は、下表に示すように正規表現の照合動作に作用します。ただし、`* + ?` など、任意の長さの文字列と一致させるために使用する文字は、一致可能な最長文字列の左端と常に一致します。

  下表に、特殊文字とその用途を示します。

文字	用途
^	文字列の先頭と一致します。例： `str ? * "^abc"` は、`abc` で始まる `str` と一致します。
$	文字列の末尾と一致します。例： `str ? * "^abc$"` は、`abc` で終わる `str` の値と一致します。
.	任意の 1 文字と一致します。例： `str ?* "a.c$"` は、`abc`、`axc` などを含む `str` の値と一致します。
*	直前の長さゼロ以上の表現と一致します。例： `str ? * "ab*c$"` は、`ac`、`abc`、または `abbc` などを含む `str` の値と一致します。
+	直前の長さ 1 以上の表現と一致します。例： `str ?* "a+c$"` は、`abc`、`abbc`、または `agggc` などを含む `str` の値と一致します。ただし、`ac` とは一致しません。
?	直前の長さゼロまたは 1 の表現と一致します。例： `str ? * "ab?c"` は、`ac` または `abc` を含む `str` の値と一致します。
|	直前または直後の表現と一致します。 例： `str ?* "a|b|c"` は、`a`, `b`, または `c` を含む `str` と一致します。
[ ]	[ ] 内にリストされた任意の 1 文字と一致します。 例： `str ?* "[ab]c"` は、`ac` または `bc` を含む `str` と一致します。
[^]	この文字の組み合わせは、[^] 内にリストされている以外の任意の 1 文字と一致します。例： `str ? * "a[^b]c"` は、`x` が `b` 以外の任意の文字の場合に、`axc` を含む `str` の値と一致します。

文字	用途
¥	直後の文字のエスケープ文字です。例： `str ? * "a¥.c$"` は、`str` が `a.c, and str ?* "a¥¥c$"` を含む場合に一致し、また［str ? * "a¥¥c$"］は `str` が `a¥c` を含む場合に一致します。 文字列に円記号（¥）を含めるためには、2 つ続けて円記号を入れる必要があります。
( )	サブ表現を区切ります。例： `str ? * "a(b|c)*d*"` が一致するのは、`a` が含まれていて、それに任意の数の `b` または `c` が続き、さらに `d` が続く `str` です。たとえば、`"ad"` または `"acbbccd"` などと一致します。

ワイルドカード一致の正規表現

クエリ演算子 MATCHES と共に、以下に示す文字を使用できます。

文字	用途
*	長さゼロ以上の文字と一致します。
?	任意の 1 文字と一致します。
¥	後ろに続く文字の正規表現表記としての意味をなくします。たとえば、¥* または ¥? と指定して * または ? と一致させる場合などに使用します。

目的とテンプレートの管理

CM アドミニストレータ（*ccm_admin* ロール）は、プロジェクトの目的とプロセス ルールの管理操作を行うためのロールを定義できます。

プロジェクト目的マネージャ

プロジェクト目的マネージャは、権限 PRIVILEGE_MANAGE_PROJECT_PURPOSES を含むロールを持つユーザーです。デフォルト設定では、この権限は *build_mgr* と *ccm_admin* の 2 つのロールに含まれています。サイトで任意のロールにこの権限を追加したり、削除できます。

プロジェクト目的マネージャは、データベースのプロジェクト目的を作成、削除できます。しかし、ビルド マネージャが目的を変更する際に、その変更があるプロジェクトの変更を伴い、かつそのプロジェクトの変更の権限をビルドマネージャがもたない場合は、目的の変更は失敗します。

この権限を編集できるのは *ccm_admin* ロールのユーザーのみです。

プロセス ルール マネージャ

プロセス ルール マネージャは、プロセス ルールを使用するデータベースに対して存在します。

> **注記**：Rational Synergy CLI のトラディショナル モードを使用してプロセス ルールを管理してください。

プロセス ルール マネージャは、権限 PRIVILEGE_MANAGE_PROCESS_RULES を含むロールを持つユーザーです。デフォルト設定では、この権限は *build_mgr* と *ccm_admin* の 2 つのロールに含まれており、*ccm_admin* ロール内のユーザーのみがこの権限を編集できます。サイトで任意のロールにこの権限を追加したり、削除できます。

プロセス ルール マネージャはプロセス ルールを作成または編集できます。しかし、ビルド マネージャがプロセス ルールを変更する際に、その変更があるプロジェクトの変更を伴い、かつそのプロジェクトの変更の権限をビルド マネージャが持たない場合は、ルールの変更は失敗します。また、ビルド マネージャは、開発者が作業中のプロジェクトで使用しているプロセス ルールを削除できません。プロセス ルールを削除できるのは *ccm_admin* ロールのユーザーのみです。

ロールの設定については、role_definitions を参照してください。

リリース マネージャ

リリース マネージャは、権限 PRIVILEGE_MANAGE_RELEASES を含むロールを持つユーザーです。デフォルト設定では、この権限は *build_mgr* と *ccm_admin* の 2 つのロールに含まれています。サイトで任意のロールにこの権限を追加したり、削除できます。

リリース マネージャは、リリース情報を作成または編集できます。ただし、たとえば使用中のリリースの名前変更や削除など、動作によっては *ccm_admin* ロールを必要とする場合があります。

デフォルト設定

デフォルト設定は、Rational Synergy の出荷時に用意されている一連の定義済みの値または設定です。これらのデフォルト設定は、大部分のユーザーが選択できる設定として定義されています。ただし、これらの設定は個々のニーズに合わせて修正できます。このセクションでは、デフォルト値とその格納場所を示し、設定の変更方法を解説し、設定間の相互関係について説明します。以下のトピックについて説明します。

- デフォルトの設定方法
- デフォルト オプション
- 初期設定ファイル — Windows
- 初期設定ファイル — UNIX
- 環境変数

デフォルトの設定方法

以下に、デフォルト値を設定または変更する標準的な方法を示します。

- システム全体の設定
- データベース全体の設定
- 個人設定
- コマンドライン設定

Rational Synergy は、最初にシステム全体またはデータベース全体の設定を、次に個人的な設定を、その次にコマンドラインから設定された値を読み取ります。最後に読み込まれた値がその前の設定に優先します。以下に、デフォルト値を設定する標準的な方法を説明します。

システム全体の設定

システム全体のデフォルトの設定はインストール エリアの全ユーザーに影響します。通常、これらのデフォルトはシステム 初期設定（`ini`）ファイルで設定されます。

初期設定ファイルは `ccm.ini` と呼ばれ、*`CCM_HOME`* の `etc` ディレクトリにあります。インストール エリアの全ユーザーは、システム初期設定ファイル内の新しいデフォルト設定を有効にするためには、セッションを再起動する必要があります。

データベース全体の設定

データベース全体の設定は該当データベースの全ユーザーに影響します。通常、これらのデフォルトはモデル オブジェクトの属性か、または特定タイプのオブジェクトの属性で設定されます。モデルの属性を修正して設定を変更した場合、新しい設定を有効にするためには、セッションを再起動する必要があります。

個人設定

個人設定は個人のセッションとデータベースにのみ影響します。これらのデフォルトは、個別の選択肢に応じて、以下の 3 つのいずれかの場所で設定します。

- 個人用初期設定（`ini`）ファイルの `[Options]` セクション

  Windows では、初期設定ファイルは `ccm.ini` という名前で Windows の Documents and Settings ディレクトリ（例、`C:¥Documents and Settings¥`*`user_name_directory`*）に作成されます。

  UNIX では、初期設定ファイルは `.ccm.ini` という名前で、`$HOME` ディレクトリに作成されます。

  初期設定ファイルの新しいデフォルト設定を有効にするためには、セッションを再起動する必要があります。

- コマンドライン

  いくつかの個人設定はコマンドラインを使用して設定されます。

コマンドライン設定

`ccm set` コマンドを使用して、コマンドラインから変数を設定することで、多くの Rational Synergy オプションを設定できます。この場合、新しいデフォルト設定は直ちに使用可能となるため、設定を有効にするためにセッションを再起動する必要はありません。オプションによっては、現在のセッションのみに設定が適用される場合と、セッション間で持続する場合があります。`set` コマンドの構文は以下のとおりです。

`ccm set` *`variable_name variable_value`*

`ccm set` コマンドで設定できるオプションのほとんどは、現在のセッションのみに適用されます。恒久的なオプションは、恒久的であることが示されます。

デフォルト オプション

ここでは、Rational Synergy オプション、デフォルト値、およびそれを設定する場所について説明します。オプションはアルファベット順にリストされています。オプション名は、コマンドラインまたは個人用の初期設定ファイルで変数として定義される場合、大文字と小文字の区別はありません。ただし、モデル属性内で指定されるオプションの場合、属性名は小文字で記述する必要があります。

Rational Synergy は、デフォルト オプションに対して行われた設定を以下の優先順位で使用します。

1. システム レベルまたはデータベース レベル（すなわち、システムの `ccm.ini` ファイルまたはモデル属性）

   Rational Synergy はシステムの `ini` ファイルまたは該当するモデル属性に設定されているオプションを最初に読みます。

2. 個人レベル（すなわち、個人の `ccm.ini` ファイル）

   個人の `ini` ファイルに設定されるオプションはシステム `ini` レベルの設定に優先します。

3. `ccm set` コマンドの使用

   `ccm set` コマンドを使用して行われた変更は、システムと個人の `ini` レベルの設定に優先します。

初期設定ファイル内のデフォルトの行継続文字は、Windows ではプラス記号（+）、UNIX ではバックスラッシュ（\）です。

モデル属性オプションの設定方法については、モデル オブジェクト属性オプションの設定を参照してください。

システムまたは個人の `ini` ファイルでオプションを設定する方法については、システム用 ini ファイルまたは個人用の ini ファイルでのオプションの設定を参照してください。

set コマンドを使用してオプションを設定する方法については、ccm set コマンドを使用したオプションの設定を参照してください。

activecm.disable_sync_at_startup

設定オプション： システムまたは個人の ini ファイル

ActiveCM が起動時にワークエリア内の全プロジェクトの同期操作を行うかどうかを指定します。これを有効にしていない場合、タスクバー インターフェイス セッション起動後に、ワークエリアとデータベースを手動で同期させる必要があります。

activecm.disable_sync_at_startup のデフォルトは FALSE です。

システムまたは個人の ini ファイルでオプションを設定する方法については、システム用 ini ファイルまたは個人用の ini ファイルでのオプションの設定を参照してください。

add_object_task_assoc

設定オプション： モデル オブジェクト属性

プロジェクトに追加される既存のオブジェクトとカレント タスクを関連付けます。このオプションは、貼り付け操作（GUI から）または use コマンド（CLI から）で使用します。

デフォルトは TRUE です。

モデルを Rational Synergy リリース 4.5 またはそれ以前のリリースに合致させたい場合は、このオプションを FALSE に設定する必要があります。

変更を有効にするには、セッションを再起動する必要があります。モデル属性オプションの設定方法については、モデル オブジェクト属性オプションの設定を参照してください。

add_used_subcomponents_to_task

設定オプション： モデル オブジェクト属性

プロジェクトに追加される既存のサブコンポーネントとカレント タスクを関連付けます。このオプションは、貼り付け操作（GUI から）または use コマンド（CLI から）で使用します。

デフォルトは TRUE です。

モデルを Rational Synergy リリース 4.5 またはそれ以前のリリースに合致させたい場合は、このオプションを FALSE に設定する必要があります。

変更を有効にするには、セッションを再起動する必要があります。モデル属性オプションの設定方法については、モデル オブジェクト属性オプションの設定を参照してください。

allow_delimiter_in_name

設定オプション： モデル オブジェクト、タイプ固有属性

区切り文字を制限文字とするかどうかを制御します。

TRUE に設定した場合、現在の区切り文字は非プロジェクト オブジェクト名に対する制限文字ではなくなります。ただし、バージョン、タイプ、インスタンス、プロジェクトについては、区切り文字は制限文字のままです。

この機能を有効にした場合、右端の区切り文字が区切り文字と認識され、オブジェクト解析が右から左に実行されます。実際には、まず指定された文字列を名前として識別するこ

とでオブジェクトの識別を行い、失敗した場合はその文字列を *name<delimiter>version* として識別します。この機能は、特に create、move、use では問題になります。

> **注記**：同じ DCM クラスタ内のすべてのデータベースは、この属性に対しては同じ値を使用する**必要があります**。同じ値を使用していない場合、オブジェクトに「~」を含めたり、区切り文字を「~」に変えてしまうような不具合が生じることがあります。

この機能を有効にした場合、バージョンを持つ非プロジェクト オブジェクトを作成できます。ただし、`ccm move` を使用して、名前変更したファイルにバージョンを設定することはできません（この制約に対処するには、`ccm attr` コマンドか、または**プロパティ**ダイアログを使用してバージョンを変更します）。

この属性は個々のタイプにも設定できます。この場合、データベースの設定が `FALSE` で、タイプ固有の設定が `TRUE` の場合、タイプ固有の設定が優先されます。

製品に組み込まれている以下のタイプでは、`allow_delimiter_in_name` は `TRUE` に設定されています。

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

デフォルトは `FALSE` です。

このオプションには以下の制限と効果があります。

- プロジェクト名には区切り文字を含むことはできない。ユーザーが名前に区切り文字を含んでいるプロジェクトを作成または移動しようとした場合、その試みは失敗し、エラー メッセージが表示されます。
- このオプションを有効にした場合、GUI または CLI での作成操作でバージョンの指定ができなくなり、常に名前が `object_spec` であるとみなされます（この変更を行う前は、オブジェクト作成時に名前とバージョンの両方を指定できます。たとえば、`foo-one` を指定して、名前が `foo`、バージョンが `one` のオブジェクトが作成されます。区切り文字の変更後は、名前が `foo-one` で、バージョンが `1` のオブジェクトが作成されます）。それ以外には、名前に区切り文字を含むオブジェクトを作成する方法はありません。作成してから名前を変更する必要があります。
- このオプションを有効にした後では、オブジェクト参照形式 *name<delim>version* を使用する CLI コマンドは、その名前を持つオブジェクトを最初に検索し、見つからなかった場合は、区切り文字の右側がない名前を持つオブジェクトを検索します。たとえば、`foo-one` というファイルと `foo` というファイルがともにワークエリアに存在し、`foo-one` を指定した場合、CLI コマンドは最初に `foo-one` という名前のファイルを検索します。その名前を持つファイルが見つからない場合のみ、名前が `foo` で、

バージョンが one のファイルを検索します。もう一方のファイル（foo version one）は、その 4 部名称または選択セット参照フォームを使用して識別できます。

- このオプションを有効にした場合、名前の先頭に区切り文字を持つオブジェクトで、その区切り文字がオプションの -（ダッシュ、マイナス）である場合、そのオブジェクトに対する CLI コマンドは失敗します。たとえば、コマンド `ccm create -foo.c` は失敗します。

モデル属性オプションの設定方法については、モデル オブジェクト属性オプションの設定を参照してください。

allow_prep

設定オプション： プロジェクト、プロジェクトのタイプ属性

プロジェクトの更新時に *prep* 状態のサブプロジェクトを含めます。ただし、このプロジェクトを含むタスクまたはフォルダを更新プロパティに含めていること、が必要です。

このオプションは、標準手法の代替手法をサポートするために用意されています。ただし、このオプションの使用には、*prep* 製品を別の（不適切な）内容で上書きするリスクが伴います。以下に発生する可能性がある状況を説明します。

-- プロジェクトに *prep* サブプロジェクトが含まれ、プロジェクトの所有者（または、ビルド管理プロジェクトのビルド マネージャ）が *build_mgr* ロールで実行する場合、次にプロジェクトをビルドするとき、*prep* サブプロジェクト内のプロジェクトが期限切れであると判断された場合、そのプロジェクトを再ビルドできます。再ビルドの後、プロジェクトは、プロジェクトが最後にビルドされたソフトウェアを構成する残りの製品と同期がずれることがあります。

デフォルトは FALSE で、プロジェクト更新時には、*prep* サブプロジェクトは使用できません。

プロジェクト タイプ属性は、モデル属性と同じ方法で設定されます。モデル属性オプションの設定方法については、モデル オブジェクト属性オプションの設定を参照してください。

auto_create_component_tasks

設定オプション： Releasedef タイプ属性

リリース用の新しいベースラインの作成時にデフォルトでコンポーネント タスクを作成するかどうかを指定します。個別リリースの対応する設定が、このオプションに優先します。

値が TRUE の場合、新しいベースライン用にコンポーネント タスクが作成されます。値が FALSE の場合は作成されません。この属性のデフォルト値は TRUE です。

個別リリースでこの設定を変更する方法については、リリースの修正を参照してください。

baseline_template

設定オプション： モデル オブジェクト属性、`ccm set` コマンド、または **オプション** ダイアログボックス

ベースラインの作成またはベースラインの修正操作で何も明示的に指定していない場合に、ベースライン内のプロジェクトと製品に使用されるバージョン テンプレートを指定します。

デフォルトは `%{version}_%date` です。

使用するテンプレートを変更する場合は、`ccm set` コマンドを使用します。この設定は恒久的で、特定データベース内の特定ユーザーの全クライアントの全セッションに適用されます。

このオプションは、**オプション** ダイアログボックスからも指定できます。この設定も恒久的で、特定データベース内の特定ユーザーの全クライアントの全セッションに適用されます。

ベースライン テンプレートの構文は baseline コマンドで定義されます。

モデル属性オプションの設定方法については、モデル オブジェクト属性オプションの設定を参照してください。

set コマンドを使用してオプションを設定する方法については、ccm set コマンドを使用したオプションの設定を参照してください。

baseline_template_date_format

設定オプション： モデル オブジェクト属性、`ccm set` コマンド、または **オプション** ダイアログボックス

ベースライン作成時に、`baseline_template` 内で `date` キーワードを展開するとき、使用する日付形式を指定します。

デフォルトは `= %Y%m%d` です。

使用する日付形式を変更する場合は、`ccm set` コマンドを使用します。この設定は、特定データベース内の特定ユーザーの全クライアントの全セッションに適用されます。

このオプションは、**オプション** ダイアログボックスからも指定できます。この設定も恒久的で、特定データベース内の特定ユーザーの全クライアントの全セッションに適用されます。

モデル属性オプションの設定方法については、モデル オブジェクト属性オプションの設定を参照してください。

set コマンドを使用してオプションを設定する方法については、ccm set コマンドを使用したオプションの設定を参照してください。

baseline_template_repl_char

設定オプション： モデル オブジェクト属性、`ccm set` コマンド、または **オプション** ダイアログボックス

ベースラインのプロジェクトまたは製品のインスタンス化された *`version_template`* に、バージョン文字列に許されない文字が含まれていた場合に使用される、デフォルトのバージョン文字列置換文字を設定します。

デフォルトは下線（_）です。

たとえば、`%platform` がプロジェクト バージョン テンプレートの一部であり、ビルド管理プロジェクトのプラットフォームが `SPARC-solaris` である場合、バージョン文字列には文字列 `SPARC_solaris` が入ります。または、`%release` が製品バージョン テンプレートの一部であり、prep 製品のリリースが `CM/6.5` である場合、このバージョン文字列には文字列 `CM_6.5` が入ります。

使用する文字を変更する場合は、`ccm set` コマンドを使用します。この設定は、特定データベース内の特定ユーザーの全クライアントの全セッションに適用されます。

このオプションは、**オプション** ダイアログボックスからも指定できます。この設定も恒久的で、特定データベース内の特定ユーザーの全クライアントの全セッションに適用されます。

モデル属性オプションの設定方法については、モデル オブジェクト属性オプションの設定を参照してください。

set コマンドを使用してオプションを設定する方法については、ccm set コマンドを使用したオプションの設定を参照してください。

check_release

設定オプション： モデル オブジェクト属性

オブジェクトとその関連タスクのリリース値を比較し、これらが同じものであることを確認します。値が一致しない場合、メッセージ ビュー（ccm_ui.log）にメッセージが書き込まれ、一致していないことを示します。

デフォルトは TRUE です。

モデル属性オプションの設定方法については、モデル オブジェクト属性オプションの設定を参照してください。

cli_compare_cmd
cli_proj_compare_cmd
cli_symlink_compare_cmd
cli_merge_cli

設定オプション： システムの ini ファイル、個人の ini ファイル、オブジェクト、オブジェクト タイプ属性、ccm set コマンド

cli_compare_cmd は、CLI から 2 つの通常のファイルを比較するときに実行されるデフォルトのコマンドです。通常のファイルとは、プロジェクト、ディレクトリ、シンボリック リンクのいずれでもないオブジェクトのことです。このオプションの値は比較のために選択された 1 番目のオブジェクトの cli_compare_cmd 属性となり、この値はデフォルトで ccm_dff -o %outfile %file1 %file2 です。

cli_proj_compare_cmd は、CLI から 2 つのプロジェクトを比較するときに実行されるデフォルトのコマンドです。このデフォルトは選択された最初のオブジェクトの cli_compare_cmd 属性となり、この属性はデフォルトで sdiff -w 80 %file1 %file2 となります。

cli_dir_compare_cmd は、CLI から 2 つのディレクトリを比較するときに実行されるデフォルトのコマンドです。このデフォルトは選択された最初のディレクトリの cli_compare_cmd 属性となり、この属性はデフォルトで %ccm_merge となります。

cli_symlink_compare_cmd は、CLI から 2 つのシンボリック リンクを比較するときに実行されるデフォルトのコマンドです。このデフォルトは選択された最初のシンボリック リンクの cli_compare_cmd 属性となり、この属性はデフォルトで ccm_dff -o %outfile %file1 %file2 となります。

cli_merge_cmd は、CLI から 2 つの通常のファイルをマージするときに実行されるデフォルトのコマンドです。通常のファイルとは、プロジェクト、ディレクトリ、シンボリック リンクのいずれでもないオブジェクトのことです。このデフォルトは選択された最初のディレクトリの cli_compare_cmd 属性となり、この属性はデフォルトで %ccm_merge となります。

Windows では、これら 4 つのオプションのすべてのデフォルトはシステムの初期設定ファイルで指定されており、オブジェクト タイプのデフォルトに優先します。4 つのオプションの Windows デフォルトはすべて同じで、ccm_dff -o %outfile %file1 %file2 です。

初期設定ファイル内で ccm_dff コマンドを参照するすべての行は、-o %outfile を使用して、比較結果を標準出力ではなく %outfile に書き込みます (%outfile は、2 つのファイルの比較結果が含まれるファイルです)。

システムの ini ファイルまたは個人の ini ファイルでオプションを設定する方法については、システム用 ini ファイルまたは個人用の ini ファイルでのオプションの設定を参照してください。

オブジェクト タイプ オプションの設定方法については、オブジェクト タイプ属性オプションの設定を参照してください。

set コマンドを使用してオプションを設定する方法については、ccm set コマンドを使用したオプションの設定を参照してください。

cli.text_editor

設定オプション: システムの ini ファイル、個人の ini ファイル、ccm set コマンド

オブジェクトのソースの修正に使用するテキスト エディタを指定します。接頭辞 cli は、デフォルトがコマンドライン用であることを示します。ユーザー インターフェイスはこの変数を使用して、テキスト属性の編集に使用するツールを決定します。必ずプログラムのフル パス名を含めてください。含めない場合はディレクトリをパスに含める必要があります。

GUI および CLI のデフォルト テキスト エディタは、Windows ではメモ帳、UNIX では vi です。

システムの ini ファイルまたは個人の ini ファイルでオプションを設定する方法については、システム用 ini ファイルまたは個人用の ini ファイルでのオプションの設定を参照してください。

set コマンドを使用してオプションを設定する方法については、ccm set コマンドを使用したオプションの設定を参照してください。

cli.text_viewer

設定オプション: システムの ini ファイル、個人の ini ファイル、ccm set コマンド

オブジェクトのソースの表示に使用するテキスト エディタを指定します。接頭辞 cli は、デフォルトがコマンドライン用であることを示します。

CLI のデフォルト テキスト ビューアは、Windows ではメモ帳、UNIX では vi です。

システムの ini ファイルまたは個人の ini ファイルでオプションを設定する方法については、システム用 ini ファイルまたは個人用の ini ファイルでのオプションの設定を参照してください。

set コマンドを使用してオプションを設定する方法については、ccm set コマンドを使用したオプションの設定を参照してください。

conflict_exclude_rules

設定オプション： モデル オブジェクト属性

この属性は、コンフリクトのあるオブジェクトの属性値に応じて、新しいコンフリクトを除外します。以下の構文をサポートします。

構文	説明
`attrname=attrvalue`	`attrname` オブジェクト属性値が `attrvalue` に一致するコンフリクトを除外します。
`attrname!=attrvalue`	`attrname` オブジェクト属性値が `attrvalue` に一致しないコンフリクトを除外します。
`EXISTS(attrname)`	`attrname` という名前の属性を持つオブジェクトのコンフリクトを除外します。
`NOT_EXISTS(attrname)`	`attrname` という名前の属性を持たないオブジェクトのコンフリクトを除外します。
`MATCHING(attrname)`	オブジェクトの `attrname` 属性値がプロジェクトの属性値に一致するコンフリクトを除外します。
`NOT_MATCHING(attrname)`	オブジェクトの `attrname` 属性値が自身のバージョンの属性値と一致しないコンフリクトを除外します。

`conflict_exclude_rules` のデフォルト値はありません。

追加情報：

- ! ルールおよび != ルールは、タイプが string と boolean の値をサポートします。
- このルールで指定された値には復帰改行文字を含むことはできません。
- 等号式／不等号ルールの区切り文字の場合を除いて、ルールに文字シーケンス = または != を含めることはできません。
- パーサーが理解できない行はすべて無視されます。
- 属性名または文字列値を引用符で囲む必要はなく、使用すべきではありません。引用符が使用されている場合、リテラル、すなわち名前または値の一部と見なされます。
- `conflict_exclude_rules` 属性の値はモデル コードにキャッシュされます。したがって、ルールが変更された場合、アクティブ セッションのユーザーは、新しい値を取得するためにはセッションを再起動する必要があります。
- この属性の設定は、手動で行うか、またはモデル インストールを通して行う必要があります。カスタマイズ用のインターフェイスはありません。

モデル属性オプションの設定方法については、モデル オブジェクト属性オプションの設定を参照してください。

conflict_parameters

設定オプション： モデル オブジェクト属性

プロジェクトのコンフリクトを表示する際、このデータベース内のユーザーに表示すべきコンフリクトのタイプを指定します。属性値のデフォルトでは、全タイプのコンフリクトを一覧表示し、ユーザーの要求に応じてプロジェクトのコンフリクトを表示するかどうかを示します。

デフォルトのエディタは、属性の設定を、一行に 1 つのコンフリクト設定の形式で表示します。この行のフォーマットは `conflict_number: TRUE|FALSE` となります。シャープ記号（#）で始まる行はコマンドとして扱われます。

このオプションのコンフリクト デフォルト値は以下のとおりです。

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

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

モデル属性オプションの設定方法については、モデル オブジェクト属性オプションの設定を参照してください。

copy_db_always

設定オプション： システムの ini ファイル、個人の `ini` ファイル

Windows では、`TRUE` に設定したとき、データベースのコピーを実行します。

UNIX では、`TRUE` に設定したとき、`ccm start -rc` でデータベースのコピーを実行します。

`copy_db_always` のデフォルトは設定なしです。したがって、`_timetag` ファイルが作用したときだけ、データベース コピーが実行されます。

システムの ini ファイルまたは個人の `ini` ファイルでオプションを設定する方法については、システム用 ini ファイルまたは個人用の ini ファイルでのオプションの設定を参照してください。

date_modified

設定オプション： モデル オブジェクト属性

ファイルが最後に修正された時間を示すキーワードを作成します。デフォルト モデルには `date_modified` キーワードは含まれません。`date_modified current_time` の行を選択セットに追加することにより、このキーワードを作成して現在の時間にその値を設定できます。チェックイン時にこのキーワードが展開されると、ファイルをチェックインした時間を示します。

モデル属性オプションの設定方法については、モデル オブジェクト属性オプションの設定を参照してください。

dcm_broadcast_dbid

設定オプション： モデル オブジェクト属性

正しいデータベースの転送パッケージを受け取るために、識別子として使われるデータベース ID を作成します。dcm_broadcast_dbid が空白以外の文字列に設定された場合、DCM 初期化によって、その属性値を DCM データベース識別子として使用するブロードキャスト データベースの DCM データベース定義が自動的に作成されます。dcm_broadcast_dbid が空白以外の文字列に設定されると、DCM は、一致する DCM ブロードキャスト データベース ID で生成された DCM 転送パッケージを受け取ります。

デフォルトの設定は TRUE です。

モデル属性オプションの設定方法については、モデル オブジェクト属性オプションの設定を参照してください。

dcm_log_enabled

設定オプション： モデル オブジェクト属性

DCM 受取りのインポート フェーズ後に、dcm_log 属性の作成と更新が行われるように指定します。これは、DCM がインポートまたは XML インポートの処理を指示する各オブジェクトを表示します。各行の形式を以下に示します。

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

ここで、<action> は以下のいずれかです。

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

A は更新が適用できる属性
R は更新が適用できる関係性
I はイメージ処理
<tset> は転送セット名
<dbid> はデータベース ID

dcm_log 属性は、エクスポートと XML エクスポートでは除外され、インポートと XML インポートでは無視されます。これはチェックアウト時にコピーされません。

デフォルト設定は FALSE です。このオプションは、DCM の問題のデバッグ時にお客様を支援するために、IBM Rational ソフトウェア サポートが使用するためのものです。DCM 問題のデバッグが必要ない場合、このオプションを無効にしておくことができます。

モデル属性オプションの設定方法については、モデル オブジェクト属性オプションの設定を参照してください。

dcm_time_sync_tolerance

設定オプション： モデル オブジェクト属性

データベースにアクセスするマシン間での時間の差を補正するために、サーバーの現在の時刻から差し引く時間の量（単位：秒）を指定します。DCM 転送で使用されるサーバーの同期の詳細については、『Rational Synergy Distributed』ドキュメントの「エンジンとサーバーの同期」の項を参照してください。

デフォルトの設定は、60 秒です。

モデル属性オプションの設定方法については、モデル オブジェクト属性オプションの設定を参照してください。

default_task_query

設定オプション： システムの ini ファイル、個人の `ini` ファイル

フォルダのクエリの指定に使用できるユーザー定義クエリを指定します。このデフォルトのクエリは、Rational Synergy CLI ヘルプ、トラディショナル モードの「クエリ式」で説明しているように、クエリ値を使用して修正できます。

`ini` ファイルで `default_task_query` オプションを指定すると、`ccm folder` コマンドの `task_scope` オプションに `user_defined` 値が含まれます。`User Defined` 値を選択すると、`default_task_query` オプションで指定したクエリがフォルダのクエリの一部となります。

`ini` ファイルで `default_task_query` を指定すると、`ccm folder` コマンドの `-task_scope` オプションは `user_defined` 値をサポートします。`ccm folder` コマンドで `task_scope user_defined` を使用すると、`default_task_query` オプションで指定したクエリがフォルダのクエリの一部として使用されます。

システムの ini ファイルまたは個人の `ini` ファイルでオプションを設定する方法については、システム用 ini ファイルまたは個人用の ini ファイルでのオプションの設定を参照してください。

default_version

設定オプション： モデル オブジェクト属性

オブジェクトの最初のバージョンのデフォルトの文字列を指定します。このオプションを使用して、`0001` などの最初のバージョンの代替文字列を指定します。

`default_version` のデフォルトは `1` です。

モデル属性オプションの設定方法については、モデル オブジェクト属性オプションの設定を参照してください。

engine_host

設定オプション： システムの ini ファイル、個人の `ini` ファイル

Rational Synergy エンジンを実行するマシンを指定します。

`engine_host` のデフォルトは設定なしです。

このオプションは Rational Synergy GUI では使用されません。

システムの ini ファイルまたは個人の `ini` ファイルでオプションを設定する方法については、システム用 ini ファイルまたは個人用の ini ファイルでのオプションの設定を参照してください。

expand_on_checkin

設定オプション： cvtype に設定

任意の cvtype に属性を追加し、指定された `cvtype` についてキーワードが展開されます。通常、キーワード展開はチェックアウト時に行われますが、このオプションを有効にした場合はチェックイン時に行われます。このオプションを使用するには、データベース アドミニストレータ として作業している必要があります。

特定タイプのオブジェクトに対してチェックイン時にキーワード展開をさせるには、任意のタイプに `expand_on_checkin` 属性を追加します。たとえば、すべてのテキスト タイプのオブジェクトについてキーワード展開を有効にするには、`ascii` cvtype に `expand_on_checkin` 属性を追加します。ASCII 階層内のすべてのオブジェクトがこの値を継承します。

`expand_on_checkin` のデフォルトは設定なしです。このオプションはブール値なので、`TRUE` か、また `FALSE` に設定する必要があります。

以下に、`ascii` タイプに対してこのオプションを使用する方法の例を示します。

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

html_browser

設定オプション： システムの ini ファイル、個人の `ini` ファイル

Rational Synergy の HTML ヘルプの表示に使用する HTML ブラウザを指定します。この値は、実行形式ファイルへの完全に記述されたパスである必要があります。たとえば以下のようになります。

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

デフォルトは、Windows では `ccm_exec`、UNIX では `ccm_browser` です。

このオプションは Rational Synergy GUI では使用されません。

システムの ini ファイルまたは個人の `ini` ファイルでオプションを設定する方法については、システム用 ini ファイルまたは個人用の ini ファイルでのオプションの設定を参照してください。

html_location

設定オプション： システムの ini ファイル、個人の `ini` ファイル

Rational Synergy HTML ヘルプ ファイルの代替ロケーションを指定します。この値には、インターネット上の場所（たとえば、`www.rational.com/new_help`）も指定できます。

デフォルトは `$CCM_HOME/help` です。

システムの ini ファイルまたは個人の `ini` ファイルでオプションを設定する方法については、システム用 ini ファイルまたは個人用の ini ファイルでのオプションの設定を参照してください。

include_required_tasks

設定オプション： モデル オブジェクト属性

プロジェクト グルーピングの Added Tasks にタスクを追加するときに、そのタスクが依存する必須タスクを計算してそれも追加するように指定します。

デフォルトは設定なしです。

モデル属性オプションの設定方法については、モデル オブジェクト属性オプションの設定を参照してください。

initial_role

設定オプション： システムの ini ファイル、個人の `ini` ファイル

Rational Synergy CLI を起動するときのロールを指定します。ロールとユーザー名によって、システムのオブジェクトに対するアクセス権が決定されます。

`ini` ファイルでロールを設定するほか、`ccm set` コマンドによってロールを変更できます（role で説明）。`ccm set` コマンドを使用するとき、変数名は `role` です。

`intial_role` のデフォルトは設定なしです。

> **注記：** `ccm set` コマンドを使用してロールを変更する場合、変更後のロールに対する権限を確保しておく必要があります。権限がないとこのコマンドは正しく機能しません。

システムの ini ファイルまたは個人の `ini` ファイルでオプションを設定する方法については、システム用 ini ファイルまたは個人用の ini ファイルでのオプションの設定を参照してください。

initials

設定オプション： システムの ini ファイル、個人の `ini` ファイル

プロジェクト オブジェクトまたは製品オブジェクトのデフォルトの次のバージョン値として、指定したイニシャルを使用するよう指定します。新規のプロジェクトまたは製品が個人用の場合、プロジェクトまたは製品をチェックアウトするとき、次のバージョン値のデフォルトは `initials` です。このオプションを設定していない場合、個人用のプロジェクトおよび製品の次のバージョン値のデフォルトはユーザー名です（他のすべての目的の場合、`initials` オプションの設定に関係なく、次のバージョン値のデフォルトは数値です）。

デフォルトの次のバージョン値にユーザーのイニシャルを使用するよう変更するには、システムまたは個人の初期設定ファイルに以下を入力します。

`initials=`*`your_initials`*

例：

```
initials=leb
```

`initials` のデフォルトは設定なしです。

システムの ini ファイルまたは個人の `ini` ファイルでオプションを設定する方法については、システム用 ini ファイルまたは個人用の ini ファイルでのオプションの設定を参照してください。

mail_cmd

設定オプション： システムの ini ファイル、個人の `ini` ファイル

Rational Synergy は、DCM 電子メール通知にデフォルトのメール ツールを使用します。Rational Synergy のメーラーではなくユーザー自身のメーラーを使用する場合は、`ini` ファイルの `[Options]` セクションに以下の行を入力します。

```
mail_cmd = user-defined_mail_command
```

`user-defined_mail_command` の構文は、ユーザーが使用するメーラーによって異なります。ただし、一般的にメーラーは受取り人、件名、および内容オプションと引数を必要とします。たとえば、以下に `ccmail` の `mail_cmd` 定義を示します。

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

`%recipients`、`%subject`、および `%content` の引数は、ユーザーがダイアログで与える情報をもとに Rational Synergy が自動的に展開します。

システムの ini ファイルまたは個人の `ini` ファイルでオプションを設定する方法については、システム用 ini ファイルまたは個人用の ini ファイルでのオプションの設定を参照してください。

multiple_local_proj_instances

設定オプション： モデル オブジェクト属性

プロジェクト作成時の動作を設定します。通常、任意のバージョンのプロジェクトを作成するとき、そのプロジェクトの別のインスタンスが存在し、かつ、その作成するプロジェクトがデータベースに対してローカルである場合は、作成が失敗します。同じプロジェクトの複数のインスタンスをローカルの場所には作成できません。ただし、非ローカル プロジェクトを別のデータベースから受け取った場合は、このことは、ローカル プロジェクトが同じ名前で作成されることを妨げません。

この属性を `TRUE` に設定した場合、複数のローカル プロジェクト インスタンスの作成が可能となります。DCM 用に初期化されていないデータベースでプロジェクトが作成された場合は、プロジェクトは必ずインスタンス 1 から開始されます。このインスタンスがすでに存在する場合、次に利用できるインスタンス番号が使用されます。ユーザーがインスタンス番号なしでプロジェクトを指定すると、デフォルトではインスタンス 1 が指定されたものと見なされます。

デフォルトは `FALSE` です。

モデル属性オプションの設定方法については、モデル オブジェクト属性オプションの設定を参照してください。

parallel_exclude_rules

設定オプション： モデル オブジェクト属性

パラレル通知を受けた場合に、どのバージョンを除外するかを定義する一連のルールを含みます。

以下の構文をサポートします。

構文	説明
attrname=attrvalue	attrname 属性値が attrvalue に一致するパラレル CV を除外します。 例：status=rejected
attrname!=attrvalue	attrname 属性値が attrvalue に一致しないパラレル CV を除外します。 例：is_product!=TRUE
EXISTS(attrname)	attrname という名前の属性を持つパラレル CV を除外します。 例：EXISTS(is_product)
EXCLUDE_NON_LEAF_NODES	パラレル バージョンの非リーフ ノードを除外します。
NOT_EXISTS(attrname)	attrname という名前の属性を持たないパラレル CV を除外します。 例：NOT_EXISTS(is_product)
MATCHING(attrname)	attrname 属性の値が自身のバージョンのものと一致するパラレル バージョンを除外します。 例：MATCHING(release)
NOT_MATCHING(attrname)	attrname 属性値が自身のバージョンのものと一致しないパラレル バージョンを除外します。 例：NOT_MATCHING(release)

この属性のデフォルト値は以下のとおりです。

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

追加情報：

- ! ルールおよび != ルールは、タイプが string と boolean の値をサポートします。
- このルールで指定された値には復帰改行文字を含むことはできません。
- 等号式／不等号ルールの区切り文字の場合を除いて、ルールに文字シーケンス = または != を含めることはできません。
- パーサーが理解できない行はすべて無視されます。
- 属性名または文字列値を引用符で囲む必要はなく、使用すべきではありません。引用符が使用されている場合、リテラル、すなわち名前または値の一部と見なされます。
- この属性の設定は、手動で行うか、またはモデル インストールを通して行う必要があります。カスタマイズ用のインターフェイスはありません。
- お客様によっては、デフォルト値に以下のルールを追加して、パラレル　バリアント ブランチを通知から除去することを望む場合もあります。

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

- ルール `MATCHING(owner)` を使用すべきではありません。このルールはチェックアウトには機能しません。このルールでは、派生元とするバージョンを使用してパラレルを検出するからです。

モデル属性オプションの設定方法については、モデル オブジェクト属性オプションの設定を参照してください。

proj_idx_wa_cache

設定オプション：　　システムの ini ファイル、個人の `ini` ファイル

2 つ目のワークエリア パス キャッシュのサイズを指定します。デフォルト値は `2500` です。この値を大きくすると、大規模プロジェクトのファイル アクセスのパフォーマンスを向上させることができます。

システムの ini ファイルまたは個人の `ini` ファイルでオプションを設定する方法については、システム用 ini ファイルまたは個人用の ini ファイルでのオプションの設定を参照してください。

project_subdir_template

設定オプション：　　モデル オブジェクト属性、ccm set コマンド、または **オプション** ダイアログボックス

ワークエリア パスのプロジェクト固有のディレクトリを定義するデフォルト テンプレートを変更します。このオプション設定は、設定して保存した後で作成されるプロジェクトのワークエリア パスに対して有効です。この設定によって、既存のプロジェクトのワークエリア パスが変更されることはありません。

このオプションの値をコマンドラインから変更する場合は、`project_subdir_template` 変数を設定します。これによって、インターフェイスが実行されているプラットフォーム

（UNIX または Windows のいずれか）のオプションが自動的に設定されます。この設定は恒久的で、特定データベース内の特定ユーザーの全クライアントの全セッションに適用されます。

モデル全体のデフォルト設定を変更する場合は、属性の名前に `_unix` または `_windows` を追加して、テンプレートを Windows または UNIX のどちらのワークエリアに適用するかどうかを指定する必要があります。たとえば、UNIX ワークエリア用のモデル全体のテンプレートを設定する場合は、`project_subdir_template_unix` という名前の属性を作成します。

このオプションは、**オプション** ダイアログボックスでは**プロジェクト固有ディレクトリの追加** オプションとなります。この設定も恒久的で、特定データベース内の特定ユーザーの全クライアントの全セッションに適用されます。

以下のキーワードが有効です。

`%project_name` は `%project_name` を新しいプロジェクト名で置き換えます。
`%project_version` は `%project_version` を新しいバージョンで置き換えます。
`%release` は `%release` を新しいリリース値で置き換えます。
`%platform` は `%platform` を新しいプラットフォーム名で置き換えます。
`%delimiter` は `%delimiter` を新しい区切り文字で置き換えます。

デフォルトは `%project_name%delimiter%project_version` です。

ワークエリア　　パスのプロジェクト固有ではない部分の変更が必要な場合は、wa_path_template を参照してください。

モデル属性オプションの設定方法については、モデル オブジェクト属性オプションの設定を参照してください。

set コマンドを使用してオプションを設定する方法については、ccm set コマンドを使用したオプションの設定を参照してください。

range_for_keyword_expand

設定オプション：　　システムの ini ファイル、個人の `ini` ファイル

オブジェクトの作成時または派生時に、キーワード用としてファイルの先頭から読み取る文字数を規定します。

ファイルをチェックアウトするとき、このオプションはファイルをスキャンして、キーワードを値に置き換えます。ファイルの全部分で定義されているキーワードを持つ大きなファイルがある場合、ファイル全体をスキャンするために時間がかかり、作成操作またはチェックアウト操作が非常に遅くなる可能性があります。

デフォルト値は `20480` です。この数値は、スキャンされるキーワードの最大文字数を示しています（ファイルが 1 行 80 文字に設定されている場合、デフォルトの設定では、1 ファイルあたり最低最初の 256 行のスキャンが可能です）。

すべてのキーワードがヘッダー領域にある場合、デフォルト設定でうまく機能します。キーワードがファイル全体にわたっている場合、キーワードをファイル全体に展開できるように、この優先順位を再設定する必要があります。

システムの ini ファイルまたは個人の `ini` ファイルでオプションを設定する方法については、システム用 ini ファイルまたは個人用の ini ファイルでのオプションの設定を参照してください。

reconcile.control_files_below_new_project

設定オプション： システムの ini ファイル、個人の ini ファイル

リコンサイル操作時にディレクトリから派生した新しいプロジェクトに、非管理ファイルを追加するかどうかを指定します。

デフォルトは FALSE です。

システムの ini ファイルまたは個人の ini ファイルでオプションを設定する方法については、システム用 ini ファイルまたは個人用の ini ファイルでのオプションの設定を参照してください。

reconcile.save_uncontrolled

設定オプション： システムの ini ファイル、個人の ini ファイル

コンフリクトの解決によってワークエリアから削除される非管理ファイルを、ワークエリアのゴミ箱に格納するかどうかを指定します。このオプションを TRUE に設定すると、データベースからのワークエリアの更新操作によってワークエリアから非管理ファイルが削除された場合、そのファイルはゴミ箱に格納されます。

デフォルトは FALSE です。

システムの ini ファイルまたは個人の ini ファイルでオプションを設定する方法については、システム用 ini ファイルまたは個人用の ini ファイルでのオプションの設定を参照してください。

reconf_consider_all_cands

設定オプション： モデル オブジェクト属性

プロジェクトの更新プロパティに候補が存在しないとき、最も適切なデータをディレクトリに配置するように指定します。この属性が存在しない場合、または値が FALSE の場合、プロジェクトの更新プロパティに候補が存在しないときは、ディレクトリ エントリは空のままとなります。

デフォルトは FALSE です。

モデル属性オプションの設定方法については、モデル オブジェクト属性オプションの設定を参照してください。

reconf_release_score

設定オプション： モデル オブジェクト属性

プロジェクトのリリースに最も一致するリリースを持つオブジェクトの選択に、リリース スコアを使用するように指定します。リリース スコアはデフォルトではタスクベースの更新には使用されませんが、パラレル リリースを開発し、1 つのリリースにもう 1 つのリリースの変更が含まれる場合には考慮されます。このオプションの使用は注意が必要なので、よく検討してから使用する必要があります。

デフォルトは FALSE です。

モデル属性オプションの設定方法については、モデル オブジェクト属性オプションの設定を参照してください。

reconf_stop_on_fail

設定オプション： システムの ini ファイル、個人の ini ファイル

個別の操作が失敗したとき、更新プロセスを停止します。TRUE に設定すると、更新に含まれる個別の操作が失敗した場合、更新が停止します。FALSE に設定すると、個別の操作が失敗した場合も更新プロセスが続行し、すべてのエラーを一度に見ることができます。

デフォルトは TRUE です。

システムの ini ファイルまたは個人の ini ファイルでオプションを設定する方法については、システム用 ini ファイルまたは個人用の ini ファイルでのオプションの設定を参照してください。

reconfigure_parallel_check

設定オプション： システムの ini ファイル、個人の ini ファイル

更新時にパラレル バージョン通知を表示するかどうかを指定します。

このオプションの値は FALSE、TRUE、または FULL に設定できます。FALSE に設定するか、または指定しない場合、パラレル検出は行われません。TRUE に設定した場合、更新選択ルールによって選ばれた候補内だけでパラレル検出が行われます。この設定は、保存されたベースラインとタスクが指定するパラレル バージョンを表示します。FULL に設定した場合、選択したオブジェクトの全バージョン内でパラレル検出が行われます。

デフォルトは FALSE（通知なし）です。

システムの ini ファイルまたは個人の ini ファイルでオプションを設定する方法については、システム用 ini ファイルまたは個人用の ini ファイルでのオプションの設定を参照してください。

reconfigure_using_tasks

設定オプション： モデル オブジェクト属性

プロジェクトの更新にタスクベースの Rational Synergy を使用するかどうかを指示します。この設定はデータベース全体に適用されます。

デフォルトは TRUE です。

モデル属性オプションの設定方法については、モデル オブジェクト属性オプションの設定を参照してください。

release_phase_list

設定オプション： モデル オブジェクト属性

リリースの開発または展開の各種フェーズを定義します。この機能により、開発プロセス期間のリリースの状況を追跡できます。製品の開発フェーズに合わせてこのリストをカスタマイズするか、あるいはデフォルトのリストを使用できます。デフォルトのフェーズリストには、`New`、`Requirements Definition`、`Function Definition`、`Implementation`、`Validation`、および `Released` の各フェーズがあります。

モデル属性は、各行に 1 エントリの形式で記述されます。リリース作成時のデフォルト値がリスト内の初期値となります。

モデル属性オプションの設定方法については、モデル オブジェクト属性オプションの設定を参照してください。

replace_subproj

設定オプション： システムの ini ファイル、個人の ini ファイル

更新操作が、サブプロジェクトの置き換えをデフォルトの動作として行うかどうかを指定します。

このオプションの値は、`TRUE`（更新時にサブプロジェクトを置き換える）、または `FALSE`（サブプロジェクトを置き換えない）に設定できます。

デフォルトは `TRUE`（サブプロジェクトを置き換える）です。

このオプションは CLI と Rational Synergy Classic で使用されますが、Rational Synergy GUI では使用されません。

オプション ダイアログボックスには、GUI でサブプロジェクトを置き換える別のオプションがあります。このオプションは GUI のみに適用されます。

システムの ini ファイルまたは個人の `ini` ファイルでオプションを設定する方法については、システム用 ini ファイルまたは個人用の ini ファイルでのオプションの設定を参照してください。

required_attributes

設定オプション： オブジェクト タイプ属性

あるタイプのオブジェクトを静的状態へ遷移する前に、フィールドへの入力をユーザーに要求するかどうかを指定します。必須フィールドの 1 つが未入力の場合、あるいは不正な値が入っている場合、オブジェクトは静的状態には遷移しません。

この属性の内容は、1 行に 1 つ、必須属性の名前が記述されている必要があります。たとえば、リリース、タスクの説明および優先度をタスクの必須フィールドにする場合、以下のように指定します。

```
release
task_description
priority
```

各属性に有効な値を指定しないと、タスクを完了できません。

デフォルトは空の文字列です。

オブジェクト タイプ属性オプションの設定方法については、オブジェクト タイプ属性オプションの設定を参照してください。

restrict_reconf_setting

設定オプション： モデル オブジェクト属性

開発者がプロジェクトの更新プロパティを "object status" から "tasks" へ、またはその逆に変更できるかどうかを指定します。またこのオプションでは、プロジェクト作成時に更新オプションを設定可能にするかどうかも設定できます。

> **注記：**FALSE に設定した場合、各ユーザーがいつでも更新プロパティを変更できるようになります。そのために、予期せぬビルド結果が生じることがあります。FALSE に設定した場合は、使用する更新プロパティのタイプについてチームが必ず合意する必要があります。

デフォルトでは、このオプションは TRUE に設定されており、開発者は更新プロパティの設定を変更できません。このオプションはモデル オブジェクト属性です。このオプションを設定または変更するには、ユーザーはビルド マネージャであるか、または *ccm_admin* ロールを持っている必要があります。

モデル属性オプションの設定方法については、モデル オブジェクト属性オプションの設定を参照してください。

role

設定オプション: システムの ini ファイル、個人の `ini` ファイル、`ccm set` コマンド

Rational Synergy CLI を使用するためのデフォルトのロールを指定します。

初期設定ファイルのデフォルトのロールを変更する場合は、initial_role オプションを使用します。

`ccm set` コマンドを使用してロールを変更する場合、変更後のロールに対する権限を確保しておく必要があります。権限がないとこのコマンドは正しく機能しません。

デフォルトは *developer* です。

このオプションは Rational Synergy GUI には影響ありません。

システムの ini ファイルまたは個人の `ini` ファイルでオプションを設定する方法については、システム用 ini ファイルまたは個人用の ini ファイルでのオプションの設定を参照してください。

set コマンドを使用してオプションを設定する方法については、ccm set コマンドを使用したオプションの設定を参照してください。

role_definitions

設定オプション： モデル オブジェクト属性

モデル オブジェクトのこの属性は、各ロールのユーザーがどの権限を利用できるかを指定します。

また、以下の目的でもこの属性を修正することがあります。

- プロセス ルールを修正できるデフォルト ロールを変更する。このためには、ロールに PRIVILEGE_MANAGE_PROCESS_RULES 権限を与えるか削除します。
- リリース管理のための新しいロールを追加する。このためには、新しいロールに PRIVILEGE_MANAGE_RELEASES 権限を与えます。
- タスクの作成と割り当てを行うことができるデフォルト ロールを変更する。このためには、ロールに PRIVILEGE_CREATE_AND_ASSIGN_TASKS 権限を与えるか削除します。
- DCM タスクを割り当てることができるデフォルト ロールを変更する。このためには、ロールに PRIVILEGE_ASSIGN_FOREIGN_TASKS 権限を与えるか削除します。

この属性の修正後は、セッションを再起動する必要があります。

モデル属性オプションの設定方法については、モデル オブジェクト属性オプションの設定を参照してください。

save_to_wastebasket

設定オプション： システムの ini ファイル、個人の `ini` ファイル

削除の必要なワークエリア内の非管理ファイルを、ゴミ箱ディレクトリに移動するかどうかを指定します。`update_db_from_workarea` オプションを `TRUE` に設定した場合、管理ファイルと衝突するファイルはゴミ箱にではなく、データベースにコピーされます。

デフォルトは `TRUE` です。

システムの ini ファイルまたは個人の `ini` ファイルでオプションを設定する方法については、システム用 ini ファイルまたは個人用の ini ファイルでのオプションの設定を参照してください。

shared_project_directory_checkin

設定オプション： システムの ini ファイル、個人の `ini` ファイル

共有プロジェクトの書き込み禁止ディレクトリにオブジェクトを追加または削除するとき、そのディレクトリを自動的に *integrate*（統合）状態にチェックインします。

デフォルトは `TRUE` です。

共有プロジェクトの詳細については、Rational Synergy CLI ヘルプ、トラディショナル モードの「共有プロジェクト」を参照してください。

システムの ini ファイルまたは個人の `ini` ファイルでオプションを設定する方法については、システム用 ini ファイルまたは個人用の ini ファイルでのオプションの設定を参照してください。

start_day_of_week

設定オプション： モデル オブジェクト属性

相対時間キーワード `%this_week_begin`、`%this_week_end`、`%last_week_begin`、`%last_week_end` を使用するクエリの、計算に用いる開始曜日を指定します。有効なエントリは 0 ～ 6、0 が日曜日、1 が月曜日というようになります。

デフォルトは `0` です。

モデル属性オプションの設定方法については、モデル オブジェクト属性オプションの設定を参照してください。

system_filename_filters

設定オプション： モデル オブジェクト属性

ユーザーがワークエリアの同期をとるときに無視するデータベースのデフォルトのファイル パターンを指定します。これは CM アドミニストレータ（*ccm_admin* ロール）が設定し、Rational Synergy によって使用されます。

あるファイルがその拡張子を基準にワークエリア内で無視されており、それを含める必要がある場合、`system_filename_filters` 属性からその拡張子を削除することで解決できます。

下表に、デフォルのフィルタを示します。

デフォルトのフィルタ	
`*.APS`	`*.BAK`
`*.BSC`	`*.class`
`*.CLW`	`*.ENC`
`*.EXP`	`*.IDB`
`*.ILK`	`*.INCR`
`*.MD#`	`*.MD~`
`*.MD%`	`*.NCB`
`*.OBJ`	`*.OPT`
`*.PCH`	`*.PDB`
`*.PLG`	`*.PROJDATA`
`*.RES`	`*.SBR`
`*.SUO`	`*.TLH`
`*.TLI`	`*.TMP`

デフォルトのフィルタ	
*.USER	*.WBK
_vti*	_vti**
~*	*~
Copy of *	New Folder
timestamp.inf	

モデル属性オプションの設定方法については、モデル オブジェクト属性オプションの設定を参照してください。

update_on_checkin_if_equal

設定オプション： システムの ini ファイル、個人の ini ファイル

エディタを使用するか、あるいはチェックアウトおよびチェックインのスクリプトを実行する場合、データベース バージョンとワークエリア バージョンのファイルのタイムスタンプを同じにできます。TRUE に設定すると、ワークエリアのファイルのタイムスタンプがデータベース バージョンのものより新しくないと示された場合でも、Rational Synergy は update_on_checkin_if_equal オプションによってワークエリアからデータベースにこのファイルをコピーします。

デフォルトは FALSE です。

システムの ini ファイルまたは個人の ini ファイルでオプションを設定する方法については、システム用 ini ファイルまたは個人用の ini ファイルでのオプションの設定を参照してください。

verbosity

設定オプション： システムの ini ファイル、個人の ini ファイル

ccm update コマンドからのメッセージ出力について、デフォルトの詳細度を指定します。このレベルを 5 以上に設定すると、更新操作によって詳細な情報が表示されます。またデータベースが使用するモデルでも詳細レベルを使用できます。

デフォルトは最低の設定の 0（ゼロ）です。

オプション ダイアログボックスにも詳細度の設定がありますが、効果はこの設定と同じです。

システムの ini ファイルまたは個人の ini ファイルでオプションを設定する方法については、システム用 ini ファイルまたは個人用の ini ファイルでのオプションの設定を参照してください。

wastebasket

設定オプション： システムの ini ファイル、個人の ini ファイル

wastebasket オプションを使用して、ゴミ箱ディレクトリの場所を指定します。

%database が、ゴミ箱を使用するデータベースの名前となります（ゴミ箱ディレクトリは非表示です）。

%database と %user は、同じテンプレートを使用するユーザーまたはデータベースあるいはユーザーとデータベースごとに異なるディレクトリ名を作成するため、ゴミ箱パスの指定に使用できるキーワードです。これらのキーワードは起動時に置き換えられます。ディレクトリが存在しない場合、Rational Synergy がこのディレクトリを作成します。

デフォルト パスはホーム ディレクトリにあり、以下のとおりです。

Windows：HOME¥%*user*¥ccm_wa¥.moved¥%database
UNIX：$HOME/*%user_name*/ccm_wa/.moved/%database

%*user* は %*user* をユーザー名に置き換えます。

システムの ini ファイルまたは個人の ini ファイルでオプションを設定する方法については、システム用 ini ファイルまたは個人用の ini ファイルでのオプションの設定を参照してください。

wa_path_cache_size

設定オプション： システムの ini ファイル、個人の ini ファイル

ワークエリア パス キャッシュのサイズを指定します。デフォルト値は 500 です。この値を大きくすると、大規模プロジェクトのファイル アクセスのパフォーマンスを向上させることができます。

システムの ini ファイルまたは個人の ini ファイルでオプションを設定する方法については、システム用 ini ファイルまたは個人用の ini ファイルでのオプションの設定を参照してください。

wa_path_template

設定オプション： モデル オブジェクト属性

ワークエリア パスのプロジェクト固有ではないディレクトリを定義しているデフォルト テンプレートを変更します。このオプション設定は、設定して保存した後で作成されるプロジェクトのワークエリア パスに対して有効です。この設定によって、既存のプロジェクトのワークエリア パスが変更されることはありません。

このオプションの値をコマンドラインから変更する場合は、wa_path_template 変数を設定します。これによって、インターフェイスが実行されているプラットフォーム（UNIX または Windows のいずれか）のオプションが自動的に設定されます。この設定は恒久的で、特定データベース内の特定ユーザーの全クライアントの全セッションに適用されます。

モデル全体のデフォルト設定を変更する場合は、属性の名前に _unix または _windows を追加して、テンプレートを Windows または UNIX のどちらのワークエリアに適用するかどうかを指定する必要があります。たとえば、UNIX ワークエリアのモデル全体のテンプレートを設定する場合は、wa_path_template_unix という名前の属性を作成します。

ccm set コマンドを使用して、以下のパスを設定します。

`ccm set wa_path_template %home¥%database¥`*`location`*

以下のキーワードが有効です。

`%database` は `%database` を新しいデータベース名で置き換えます。
`%user` は `%user` を新しいユーザー名で置き換えます。
`%owner` は `%owner` をプロジェクト所有者名で置き換えます。
`%home` は `%home` をホーム ディレクトリで置き換えます。

Windows のデフォルトは `%home¥ccm_wa¥%database`、`%home` はホーム ディレクトリです（ホーム ディレクトリを **Startup** ダイアログの **Home Directory** テキスト ボックス内に指定した場合）。

UNIX のデフォルトは `%home/ccm_wa/%database, where`、`%home` は UNIX ホーム ディレクトリです。

このオプションは、**オプション** ダイアログボックスでは**全てのワークエリアにデフォルトパスを追加**となります。この設定も恒久的で、特定データベース内の特定ユーザーの全クライアントの全セッションに適用されます。

ワークエリア　　パスのプロジェクト固有部分の変更が必要な場合は、project_subdir_template を参照してください。モデル属性オプションの設定方法については、モデル オブジェクト属性オプションの設定を参照してください。set コマンドを使用してオプションを設定する方法については、ccm set コマンドを使用したオプションの設定を参照してください。

初期設定ファイル — Windows

PC サーバーからの Rational Synergy の実行

通常、ccm.ini ファイルは 2 つの場所に格納されています。システム ファイルは製品のインストール エリア *CCM_HOME*¥etc、個人用ファイルは通常は各ユーザーの Windows の Documents and Settings ディレクトリにあります。ccm.ini ファイルが Windows の Documents and Settings ディレクトリにない場合は、システム ファイルからコピーして、そのファイルを設定したいオプションで変更します。

個人用の ccm.ini ファイルは、システム ファイルに優先します。

任意のテキスト エディタを使用して、ccm.ini ファイルを編集できます。

ユーザーの PC での Rational Synergy の実行

Rational Synergy をユーザーの PC のインストール ディレクトリから実行する場合、ccm.ini ファイルは *CCM_HOME*¥etc に格納されています。このファイルはユーザーのマシンにのみ有効であり、直接変更できます。

初期設定ファイル — UNIX

デフォルトの ccm.ini ファイルは $CCM_HOME/etc/ccm.ini に格納されています。このデフォルトのファイルをユーザーのホーム ディレクトリにコピーして、それを修正して利用できます。修正した場合、ユーザーの .ccm.ini ファイルの設定が、システムの ..ccm.ini ファイルの設定に優先します。設定を何も変更しなかった場合でも、まだ個人用の .ccm.ini ファイルを持っていなければ、セッションを終了したときに自動的に個人用のファイルが作成されます。

任意のテキスト エディタを使用して、個人用の ccm.ini ファイルを作成、編集できます。.ccm.ini ファイル はユーザーのホーム ディレクトリに格納されている必要があります。*Motif* という語句が先頭に付くオプションは、グラフィカル ユーザー インターフェイスにのみ有効です。インターフェイスの指定がない場合、このオプションは該当するすべてのインターフェイスに適用されます。

環境変数

Rational Synergy の実行方法に影響を与える以下の変数を定義できます。下表に、設定可能な環境変数を示します。

環境変数	必須／任意	使い方
AUTOMOUNT_FIX （UNIX のみ）	任意	自動マウンタの利用をサポートするため、取り除くパス名の部分を決定するために使用します。自動マウンタが /tmp_mnt 接頭辞を使用している場合は不要です。この変数は SunOS 自動マウンタ パッチでも使用されます。 詳細については、『Rational Synergy 管理者ガイド UNIX 版』を参照してください。
CCM_ADDR	任意	Rational Synergy インターフェイスの RFC（Remote Function Call）アドレス（host:*socket*）を指定します。
CCM_ENGLOG	任意	エンジン ログのリダイレクトに使用します。設定しない場合は、Windows インストール ディレクトリまたは UNIX ホーム ディレクトリにある ccm_eng.log ファイルが使用されます。エンジン ログ ファイルは可視ファイルで、*ccm_root* によって書き込み可能である必要があります。
CCM_HOME	任意	Rational Synergy インストール ディレクトリを指定します。通常、Windows クライアントの場合は C:¥Program Files¥IBM¥Rational¥ Synergy¥7.1a、UNIX の場合は、 /usr/local/ccm　です。
CCM_PAGER （UNIX のみ）	任意	UNIX 上で、ccm monitor　を使用してファイルまたはレポート出力を表示するときに、使用する実行形式ファイルの名前を指定します。設定した場合、これは PAGER に優先します。
CCM_UILOG	任意	ユーザー インターフェイス ログのリダイレクトに使用します。設定しない場合は、Windows インストール ディレクトリまたは UNIX ホーム ディレクトリにある ccm_ui.log が使用されます。
DISPLAY （UNIX のみ）	必須	X ディスプレイ サーバーの名前。たとえば、unix:0.0。
HOME （UNIX のみ）	必須	UNIX のホーム ディレクトリを指定します。
LD_LIBRARY_PATH （Sun のみ）	必須	ダイナミック オブジェクト ライブラリの検索に使用されるディレクトリのリストを指定します。たとえば、 /usr/lib/X11:/usr/openwin/lib

環境変数	必須／任意	使い方
PAGER （UNIX のみ）	任意	UNIX 上で、ccm monitor を使用してファイルまたはレポート出力を表示するときに、使用する実行形式ファイルの名前を指定します。
PATH	必須	実行形式ファイルの検索に使用されるディレクトリのリストを指定します。
PRINT_EDIT_CMD	任意	この変数に値を設定すると、エディタ コマンドの実行時に、使用するモデル定義エディタ メソッドが表示されます。
PRINT_TOOL_CMD	任意	この変数に値を設定すると、ツール コマンドの実行時に、モデル定義ツール メソッド（たとえば、デバッグ、印刷または実行）が表示されます。
RECONF_TIME	任意	update コマンドの実行時間の指定と表示を行います。
SHELL （UNIX のみ）	必須	サブプロセスを起動するための UNIX コマンド インタープリタ シェルの名前を指定します。たとえば、/bin/csh
TERM （UNIX のみ）	必須	UNIX コマンドの入力に使用するターミナル タイプを指定します。たとえば、xterm
UIDPATH （UNIX のみ）	任意	ファイルの検索に使用されるディレクトリのリストを指定します。
USER （UNIX のみ）	必須	ユーザー名を指定します。

注記：Rational Synergy は、CCM_ または AC_ で始まるその他の変数を、内部診断に使用します。IBM Rational Synergy サポートからの指示がない限り、INFORMIXDIR、INFORMIXSERVER、ONCONFIG などの変数を**設定しないでください**。

モデル オブジェクト属性オプションの設定

モデル オブジェクトに対してモデル全体の属性を設定できます。これらの設定は、モデルのインストール先のデータベースの全ユーザーに影響を与えます。モデル属性オブジェクトを変更するには、*ccm_admin* ロールを持っている必要があります。

最初の例は、属性の作成方法を示します。属性の作成はある特定の場合に必要となります。属性は、作成した後に値を設定できます。2 番目の例では、設定済みの属性の変更方法を示します。

変更するオプションに応じて、適切なオプション名と構文で例を置き換えてください。

属性の作成

この例では `allow_delimiter_in_name` 属性を使用します。この属性は、バージョン区切り文字の使用をオブジェクト名に対して許すかどうかを指定します。デフォルトでは、この属性は存在しません。すなわち、区切り文字は使用できません。

このオプションを初めて設定するときは、以下のように属性を作成する必要があります。

1. ロールを *ccm_admin* に設定します。

   ```
   ccm set role ccm_admin
   ```

2. Rational Synergy データベースでモデル オブジェクトをクエリします。

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

3. 属性を作成します。

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

4. 値を設定します。

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

5. セッションを再起動します (このデータベースの全ユーザーが自分のセッションを再起動する必要があります)。

属性の修正

この例では `wa_path_template_unix` 属性を使用します。この属性は、デフォルトのプロジェクト非固有ワークエリア ディレクトリを指定します。デフォルトでは、このディレクトリは `%home/ccm_wa/%database` です。

この属性を変更するには以下の手順を行います。

1. ロールを *ccm_admin* に設定します。

   ```
   ccm set role ccm_admin
   ```

2. Rational Synergy データベースでモデル オブジェクトをクエリします。

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

3. 新しいパスを指定します。

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

4. 新しい属性の内容を表示します。

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

5. 前のロールに戻ります。

   ```
   ccm set role previous_role
   ```

6. セッションを再起動します (このデータベースの全ユーザーが自分のセッションを再起動する必要があります)。

新しい属性のリスト ボックスの作成

新しく作成された属性を使ってリスト ボックスを作成できます。新しいリスト ボックスを作成する構文は以下のとおりです。

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

ここで、values_ref は別の属性で新しい値定義エントリとして定義されているものです。

各 values_ref 値の定義エントリは、あるオブジェクトについての別個のテキスト属性で定義するか、info_attrs.values_ref という型で定義する必要があります。ここで、values_ref は info_attrs 定義で参照できるように値のリストに付けられた名前です。この形式を使うと、外部ツールを使って簡単にリストに値を取り込めます。

values_ref は、属性名の一部となるため、正式な属性名である必要があります。属性名の最大長は 32 文字ですが、そのうち 11 文字を文字列 info_attrs が使用するため、values_ref 属性名は 21 文字以内にする必要があります。

info_attrs.values_ref 属性の内容は、リストボックスで設定可能な値を復帰改行で区切ったリストです。

このリスト内の値には、空白文字を含む ASCII 文字列を使用できます。ただし、文字列の先頭または末尾の空白文字は、復帰改行区切り文字の一部見なされるため使用できません。

例：

approval_level というカスタム属性をタスク タイプに追加したい場合を考えます。この属性への設定値は、以下のとおりとします。

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

タスク タイプに対する info_attrs 属性に以下のようなエントリを作成します。

approval_level:string:Approval Level::approval_values

次に、タスク タイプに info_attrs.approval_level を以下の内容で作成します。

new
pending
approved level 1
approved level 2

オブジェクト タイプ属性オプションの設定

オブジェクトに対してオブジェクト タイプ属性を設定できます。この設定は、そのオブジェクト タイプを持つデータベースの全ユーザーに影響を与えます。オブジェクト タイプを変更するには、*ccm_admin* ロールを持っている必要があります。

変更するオプションに応じて、適切なオプション名と構文で例を置き換えてください。

1. ロールを *ccm_admin* に設定します。

   ```
   ccm set role ccm_admin
   ```

2. 設定を変更するタイプをクエリします。

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

3. 属性を変更します。

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

   属性に対するエディタが呼び出されます。変更して、その値を保存します。

4. 前のロールに戻ります。

   ```
   ccm set role previous_role
   ```

5. セッションを再起動します (このデータベースの全ユーザーが自分のセッションを再起動する必要があります)。

システム用 ini ファイルまたは個人用の ini ファイルでのオプションの設定

一部のオプションは、個人用の `ini` ファイルを変更するか、あるいは `ccm set` コマンドを使用して設定できます。個人用の `ini` ファイルを使用してオプションを変更した場合、セッションの起動時に変更が有効になります。`ccm set` コマンドを使用してオプションを変更した場合、変更は実行中レベルで行われます（すなわち、変更を有効にするためにセッションを再起動する必要はありません）。

以下の例では `cli.text_editor` オプションを変更します。最初の例では Windows の `ccm.ini` ファイル内のオプションを、次の例では UNIX の `.ccm.ini` ファイル内のオプションを変更しています。変更するオプションに応じて、適切なオプション名と構文で例を置き換えてください。

- デフォルトのエディタとしてメモ帳を使用するには、`ccm.ini` ファイルに以下の設定を記述します。

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

- デフォルトのエディタとして `vi` を使用するには、`ccm.ini` ファイルに以下の設定を記述します。

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

ccm set コマンドを使用したオプションの設定

一部のオプションは、個人用の ini ファイルを変更するか、あるいは ccm set コマンドを使用して設定できます。個人用の ini ファイルを使用してオプションを変更した場合、セッションの起動時に変更が有効になります。ccm set コマンドを使用してオプションを変更した場合、変更は実行中レベルで行われます（すなわち、変更を有効にするためにセッションを再起動する必要はありません）。

以下の例では、wa_path_template オプションを変更します。

変更するオプションに応じて、適切なオプション名と構文で例を置き換えてください。

以下を入力してワークエリア パス テンプレートを変更します。

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

コマンド

alias コマンド

詳細については、説明と用途を参照してください。`alias` コマンドは、以下のサブコマンドをサポートします。

- 別名の定義
- 別名の表示
- 特定の別名の表示

別名の定義

このサブコマンドにより、新しい別名を定義します。*alias_name* は、作成する新しい別名の名前です。

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

alias_name

定義する新しい別名の名前を指定します。

alias_string

別名で置き換える対象を指定します。空白を含む *alias_string* を二重引用符で囲んで指定すると、空白で分割されたままの形で別名定義されます。*alias_name* の後ろに二重引用府なしで複数の引数を指定すると、各引数が連結された形で別名定義されます。

例

- 新バージョンを持つファイルをチェックアウトする別名を作成する。

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

  新しい別名を使用する場合は、以下のフォーマットになります。

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

- 別名の値を変更する。

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

  たとえば、オブジェクトのクエリを行う my_query という別名が定義されているとします。この my_query の値を、タスクのクエリを行うように変更したい場合は、alias コマンドを実行して、別名 my_query の値を変更します。

- 2 つの項目 properties と -verbose を持つ vprop という名の別名を定義する (*alias_string* ["properties -verbose"] は、別名定義の項目を作るため空白で分割されます)。

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

- 2 つの項目 properties と verbose を持つ vprop という名の別名を定義する (各引数 [properties -verbose] は別名の項目と見なされますが分割はされません)。

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

関連トピック

- 別名の表示
- 特定の別名の表示
- unalias コマンド

別名の表示

このサブコマンドにより、現在のセッションで定義されている別名を表示します。

```
ccm alias
```

例

- 定義済みのすべての別名を一覧表示する。

  ```
  ccm alias
  ```

関連トピック

- 別名の定義
- 特定の別名の表示
- unalias コマンド

特定の別名の表示

このサブコマンドにより、特定の別名を表示します。

```
ccm alias alias_name
```

`alias_name`

表示する別名の名前を指定します。

関連トピック

- 別名の定義
- 別名の表示
- unalias コマンド

説明と用途

別名は、既存のコマンドまたは他の別名に対して、別の名前を作成するために使用するマクロを示します。別名は現在のセッションでのみ存在します。別名が他の別名を参照し、その別名がまた他の別名を参照することもありますが、循環して参照することはできません。Rational Synergy はコマンドを別名として認識すると、別名を展開してそれが定義している項目で置き換えます。この展開は、コマンドが別名ではなくなるか循環参照が見つかるまで繰り返されます。

attribute コマンド

詳細については、説明と用途を参照してください。`attribute` コマンドは、以下のサブコマンドをサポートします。

- 属性のコピー
- 属性の作成
- 属性の削除
- 属性の修正
- 属性の表示
- 属性の一覧表示

属性のコピー

```
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`

指定された属性値を指定されたオブジェクトに付加します。このオプションを使用しない場合は、指定された属性が持つ既存の値がすべて新しい値に置き換わります。

`-cp|-copy` *`attr_names`*

選択されたオブジェクトまたはオブジェクト バージョンに、1 つまたは一連の属性をまとめて一度にコピーします。複数の属性名を指定する場合は、区切り文字としてコロンまたは空白文字、またはコロンと空白文字を使用できます。

`from_object_spec to_object_spec`

属性のコピー元のファイルが *`from_object_spec`* であり、属性のコピー先のファイルが to_object_spec であることを指定します。*`from_object_spec`* には 1 つのオブジェクト、*`to_object_spec`* には複数のオブジェクトを指定できます。

`from_project_spec to_project_spec`

属性のコピー元のプロジェクトが *`from_project_spec`* であり、属性のコピー先のプロジェクトが to_project_spec であることを指定します。*`from_project_spec`* には 1 つのプロジェクト、*`to_project_spec`* には複数のプロジェクトを指定できます。

`-subproj` または *`-suball`* を使用した場合、プロジェクトは *`to_proj_spec`* に適用されます。

`-suball`

指定された属性を、サブプロジェクトのオブジェクトおよび指定されたプロジェクトのすべてのメンバーに再帰的にコピーします。このオプションは、*`to_proj_spec`* に適用され、*`-p`* オプションを必要とします。`-subproj` オプションと一緒には使用できません。

`-subproj`

指定された属性または一連の属性を、指定されたプロジェクト内のサブプロジェクト オブジェクトに再帰的にコピーします。このオプションは、*`to_proj_spec`* に適用され、*`-p`* オプションを必要とします。`-suball` オプションと一緒には使用できません。

例

- プロジェクト attr_test-1 の version 属性をそのサブプロジェクトにコピーする。
  ```
  ccm attr -copy version -project attr_test-1 -subproj attr_test-1
  ```

属性の作成

```
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*

属性を作成します。

`-f|-force`

作成しようとする属性が存在しているか、同じタイプを持っているかをチェックします。その結果、以下のようになります。

- 作成しようとする属性が存在しており同じタイプを持っている場合は、属性の値が変更される（`-value` オプションを使用する場合）。
- 属性が存在しない場合は、新しい属性が作成される。
- 同じ名前を持つ属性が存在するがタイプが異なる場合、操作は失敗する。

`ccm attr -c` *attr_name* `-t` *type* と `ccm attr -c` *attr_name* `-f -t` *type* の違いは、`-force` オプションが指定されていないコマンドは、属性がすでに存在している場合に失敗することです。

`-p|-project`

属性を作成するプロジェクトを指定します。

`-t|-type` *attr_type*

属性のタイプを指定します。このオプションは、属性を作成する場合にのみ使用します。組み込まれている有効な値には、以下のようなものがあります。

- string（1 行の `ascii` 属性に使用）
- boolean
- text（複数行の `ascii` 属性に使用）

`-v|-value` *attr_value*

属性の値を指定します。

例

- `driver.c` オブジェクトの文字列属性 `new_attr` を作成する。

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

関連トピック

- 属性のコピー
- 属性の一覧表示
- 属性の表示

属性の削除

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

`-d|-delete attr_name`

属性を削除します。

`-p|-project`

属性を削除するプロジェクトを指定します。

例

- `driver-1` プロジェクトの `new_attr` 属性を削除する。

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

関連トピック

- 属性の一覧表示
- 属性の表示

属性の一覧表示

```
attr|attribute -p|-project ([-l|-list] | [-la] | [-li]) project_spec...
attr|attribute ([-l|-list] | [-la] | [-li]) object_spec...
```

`-l`

すべてのローカル属性を一覧表示します。

`-la`

すべての属性を一覧表示します。

`-li`

継承された属性を一覧表示します。

`-p|-project`

属性を一覧表示したいプロジェクトを指定します。

関連トピック

- 属性のコピー
- 属性の削除
- 属性の修正
- 属性の表示

属性の修正

```
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`

属性を修正します。`-v` オプションを指定しない場合、デフォルト エディタは適切な属性タイプを使用して指定オブジェクトの属性を更新します。

`-p|-project`

属性を修正するプロジェクトを指定します。

`-v|-value value`

-v|-value attr_value を参照してください。

例

- `foo.c` のリリース属性を `4.2_int` に変更する。

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

関連トピック

- 属性の一覧表示
- 属性の表示

属性の表示

```
attr|attribute -s|-sh|-show attr_name -p|-project project_spec...
attr|attribute -s|-sh|-show attr_name object_spec...
```

`-p|-project`

属性を表示したいプロジェクトを指定します。

`-s|-show attr_name`

属性の値を表示します。

例

- `driver.c` オブジェクトの `comment` 属性の値を表示する。

  ```
  ccm attr -s comment driver.c
  ```

関連トピック

- 属性のコピー
- 属性の削除
- 属性の一覧表示
- 属性の修正

説明と用途

`attribute` コマンドにより、Rational Synergy のオブジェクトに関連する属性を操作します。

baseline コマンド

詳細については、説明と用途を参照してください。`baseline` コマンドは、以下のサブコマンドをサポートします。

- ベースラインの比較
- ベースラインの作成またはプレビュー
- ベースラインのコンポーネント タスクの作成
- ベースラインの削除
- ベースラインの一覧表示
- ベースラインの削除のマーク付け
- ベースラインの修正
- ベースラインの公開
- ベースラインのリリース
- 削除した ベースライン のリストア
- ベースライン プロパティの表示
- ベースラインのプロジェクト、オブジェクト、タスク、または変更依頼の表示
- ベースライン情報の表示

ベースラインの比較

このサブコマンドにより、2 つのベースラインを比較します。

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

baseline_spec1

比較する最初のベースラインを指定します。詳細については、ベースラインの指定を参照してください。

baseline_spec2

比較する 2 番目のベースラインを指定します。詳細については、ベースラインの指定を参照してください。

`-cr|-change_request|-change_requests`

ベースライン比較は、2 つのベースラインに部分的および完全に含まれる変更依頼（CR）の詳細を含む必要があることを指定します。

デフォルトのフォーマットは `%displayname: %problem_synopsis` です。

`-objects`

ベースライン内のオブジェクト メンバーの比較を含めるよう指定します。

`-projects`

2 つのベースライン間のプロジェクトの比較を含めるよう指定します。

`-tasks`

2 つのベースライン間のタスクの比較を含めるよう指定します。

例

- ベースライン `20020401_1` 内のプロジェクトと、ベースライン `20020401_2` 内のプロジェクトを比較する。

  ```
  ccm baseline -compare 20020401_1 20020401_2 -projects
  ```

関連トピック

- ベースラインの一覧表示
- ベースラインの修正
- ベースライン プロパティの表示

ベースラインの作成またはプレビュー

このサブコマンドにより、ベースラインの作成または作成のプレビューを行います。ビルド マネージャまたは *ccm_admin* ロールのユーザーは、1 つまたは複数のプロジェクト、ベースライン、またはプロジェクト グルーピングからベースラインを作成できます。

```
ccm baseline -c|-create [(-p|-project project_spec)...]
             [(-bl|-baseline baseline_spec)...]
             [(-pg|-project_grouping project_grouping_spec)...]
             [-rehearse] [-r|-release release_spec] [-purpose purpose]
             [-d|-desc|-description description]
             [-vt|-version_template version_template] [-b|-build build]
             [-s|-state state] ([-subprojects] | [-all_subprojects] |
             [-no_subprojects]) [baseline_name]
```

`-all_subprojects`

追加するプロジェクトのメンバーであるすべてのサブプロジェクトを含めるよう指定します。このオプションは、`-project`、`-project_grouping`、および `-baseline` オプションで追加するプロジェクトに適用されます。

`-b|-baseline` *`baseline_spec`*

このオプションを `-create` と一緒に使用し、ベースラインの指定を指定すると、指定した既存のベースライン内のプロジェクトが新しいベースラインに追加されます。

`–subprojects`、`-no_subprojects` および `–all_subprojects` を使用すると、どのサブプロジェクトが追加されるかに影響します。

デフォルトでは、`-baseline` を使用して `-project` を使用しない場合、サブプロジェクトは含まれません。ただし、`-project` と `-baseline` の両方を使用すると、`-project` が暗黙的に指示する `–subprojects` デフォルトが、`-baseline` が暗黙的に指示する `-no_subprojects` デフォルトに優先します。

`baseline_name`

ベースラインに割り当てる名前を指定します。ベースラインを作成するとき、正式なベースライン名に割り当てることができます。

`baseline_name` を指定しないと、一意な名前が自動的にベースラインに割り当てられます。デフォルト名の形式は `yyyymmdd` です。必要に応じて、デフォルト名の後ろに下線と増分番号を付けて一意の名前を作成します。たとえば、2002 年 4 月 1 日に作成した最初のベースラインのデフォルト名は `20020401` となります。同じ日に作成した 2 番目の当ベースラインのデフォルト名は `20020401_1` となります。

`-b|-build` *`build`*

新しいベースラインのビルド番号または ID を指定します。ビルド番号または ID には任意の 1 行のテキスト値を使用できます。通常、ビルド値は何らかの形のビルド番号を含みます。

`-d|-desc|-description` *`description`*

新しいベースラインに使用する説明を指定します。説明は 1 行のテキストで、復帰改行文字を含めることはできません。

`-no_subprojects`

プロジェクトを追加するとき、そのサブプロジェクトを追加しないよう指定します。このオプションは、`-project`、`-project_grouping`、および `-baseline` オプションによって追加されるプロジェクトに影響します。サブプロジェクトは、プロジェクトとして明示的に指定した場合、あるいは指定したプロジェクト グルーピングまたはベースラインのメンバーである場合に含められます。

`-project_grouping` と `-baseline` のどちらも指定しない場合、デフォルトは `-subprojects` です。`-project_grouping` または `-baseline` を指定した場合、デフォルトは `-no_sub_projects` です。

`-p|-project` *`project_spec`*

新しいベースラインに 1 つまたは複数のプロジェクトを追加するよう指定します。デフォルトでは、プロジェクトが追加されると、その階層全体も追加されます。これは、`-no_subprojects` オプションを使用して変更できます。

`-pg|-project_grouping` *`project_grouping_spec`*

新しいベースラインに指定したプロジェクト グルーピング内のプロジェクトを追加するよう指定します。デフォルトでは、プロジェクト グルーピングを追加すると、プロジェクト グルーピング内のプロジェクトのみが追加されます。プロジェクト グルーピングに属さないサブプロジェクトは追加されません。この動作を変更するには `-all_subprojects` オプションを使用します。

`–subprojects`、`-no_subprojects` および `-all_subprojects` は、プロジェクト グルーピングと共にどのサブプロジェクトが追加されるかに影響します。

`-purpose` *`purpose`*

新しいベースラインに使用する目的を指定します。指定しない場合、以下の順にデフォルトの目的が設定されます。

- ベースラインを指定した場合、Synergy は最初に指定したベースラインの目的を使用する。
- ベースラインを指定しないが、プロジェクト　　グルーピングを指定した場合、Synergy は最初に指定したプロジェクト グルーピングの目的を使用する。
- ベースラインとプロジェクト グルーピングのどちらも指定しない場合、Synergy は最初に指定したプロジェクトの目的を使用する。

`purpose` はプロジェクトの用途（例、Insulated Development （個別開発）、Integration Testing （統合テスト）、System Testing （システム テスト）など）を指定する設定です。

`-purpose` を指定した場合、-release release_spec も指定する必要があります。

`-rehearse`

ベースラインを構成するプロジェクトと製品および作成するベースラインの名前を一覧表示するだけで、実際にはベースラインを作成しないよう指定します。

バージョンのコンフリクトが見つかった場合、コンフリクトのあるすべての製品とプロジェクトのバージョンを一覧表示するする警告が表示されます。コンフリクトは、生成されるバージョンが新しいベースラインにすでに存在するか、あるいは生成されるバージョンが正当なバージョン文字列ではない場合に発生します。

`-release` *`release_spec`*

新しいベースラインに使用するリリースを指定します。ベースラインを作成するとき、1 つのアクティブ リリースを指定できます。指定しない場合、以下の順にデフォルトのリリースが設定されます。

- ベースラインを指定した場合、Synergy は最初に指定したベースラインのリリースを使用する。
- ベースラインを指定しないが、プロジェクト　　グルーピングを指定した場合、Synergy は最初に指定したプロジェクト グルーピングのリリースを使用する。
- ベースラインとプロジェクト グルーピングのどちらも指定しない場合、Synergy は最初に指定したプロジェクトのリリースを使用する。

`-release` を指定した場合、-purpose purpose も指定する必要があります。

`-state` *state*

ベースライン作成時のベースラインの状態を指定します。ベースラインの作成時に有効な状態は、*test_baseline*、*published_baseline*、および *released* です。ベースラインのデフォルトの状態は *test_baseline* です。開発者はこの状態でベースラインを表示し、ベースラインを手動で使用できます。しかし、これを最新のベースラインとして自動的に取得することはできません。SQE はこれをテストのために使用できます。ベースラインがテストに合格すると、ビルド マネージャはテスト ベースラインを *published_baseline* 状態に遷移させて、開発者が使用できるようにする必要があります。

ベースラインを *released* 状態で作成することは、ベースラインを *published_baseline* 状態で作成してリリースすることと同じです。

`-subprojects`

追加するプロジェクトのコンポーネント名と一致するリリースのサブプロジェクトを含めよう指定します。コンポーネント名は完全に一致する必要があります。コンポーネント名を持たないリリースは、コンポーネント名を持たない他のリリースとのみ一致します。

このオプションは、`-project`、`-project_grouping`、および `-baseline` オプションで追加するプロジェクトに適用されます。これは、`-project_grouping` と `-baseline` のどちらも指定しない場合のデフォルトの動作です。`-project_grouping` または `-baseline` を指定した場合、デフォルトは `-no_subprojects` です。

`-vt|-version_template` *version_template*

新しいベースライン内のプロジェクトまたは製品に使用するバージョン テンプレートを指定します。コマンド実行時に作成される新しいバージョンのプロジェクトと製品は、そのバージョンに *version_template* を使用します。

version_template はオプションのキーワードを持つ任意の文字列で、その形式は *%keyword* または *%{keyword}* です。このキーワードには、任意の Rational Synergy 属性または組み込み済みキーワードを使用できます。

属性を展開するとき、検査対象のビルド管理プロジェクトまたは製品の対応する属性値が使用されます。指定したキーワード名について属性または組み込み済みキーワードが見つからない場合は、そのキーワードは空の文字列に置き換わります。

ベースライン内のプロジェクトまたは製品についてインスタンス化した `version_template` にバージョン文字列に使用できない文字が含まれている場合、それらの文字はデフォルトのバージョン文字列の置換文字に置き換わります。この

文字は、オプション baseline_template_repl_char を使用して、ccm.ini ファイルに指定されています。この文字のデフォルトは下線（_）です。たとえば、%platform がプロジェクト バージョン テンプレートの一部であり、ビルド管理プロジェクトのプラットフォームが SPARC-solaris の場合、バージョン文字列には文字列 SPARC_solaris が含まれます。あるいは、%release が製品バージョン テンプレートの一部であり、prep 製品が CM/6.6a リリースの場合、このバージョン文字列には文字列 CM_6.6a が含まれます。

ベースライン内のプロジェクトまたは製品についてインスタンス化した *version_template* がそのプロジェクトまたは製品の別のバージョンですでに使用されている場合、そのバージョンは、下線（_）と そのバージョンを一意にする 1 から始まる最初の整数を追加することによって、一意のバージョンとなります。これによってバージョン文字列が長くなりすぎる場合は、そのプロジェクトまたは製品には現在の日付をベースとしたバージョンが使用され、警告が表示されます。

-version_template を指定しない場合、デフォルトのテンプレートが使用されます。詳細については、バージョン テンプレートの指定を参照してください。

プロジェクトのワークエリア テンプレートにバージョンが含まれる場合、ワークエリアが更新されます。ワークエリアが見えないために更新不可能であり、また -skip_nonvisible_projects が使用されていない場合、操作は継続しますがすべてのエラーが報告されます。ワークエリアが見える状態であっても、適切なファイル アクセス権限の欠如やディスク領域の不足などの他の理由で更新できない場合、操作は継続しますがすべての失敗が報告されます。

例

- リリース **2.0** について、**Integration Testing（統合テスト）**の目的で、Build_1234_int という名前のベースラインを作成する。このベースラインは、proj1-sqa_3 という名前のプロジェクトとそのサブプロジェクトを含みます。

```
ccm baseline -c Build_1234_int -d "Integration build 1234" -r 2.0
-purpose "Integration Testing" -projects proj1-sqa_3 -subprojects
```

関連トピック

- ベースラインの比較
- ベースラインの削除
- ベースラインの一覧表示
- ベースラインの公開
- ベースラインのリリース
- ベースライン プロパティの表示

ベースラインのコンポーネント タスクの作成

このサブコマンドにより、指定したベースラインのリリースと目的と一致するコンポーネント タスクを作成します。ベースラインのコンポーネントタスクを作成するには *build_mgr* または *ccm_admin* ロールを持っている必要があります。

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

baseline_spec...

コンポーネント タスクを作成するベースラインを指定します。詳細については、ベースラインの指定を参照してください。

例

- ベースライン `2.0_build_34` のコンポーネント タスクを作成する。

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

関連トピック

- ベースラインの作成またはプレビュー
- ベースラインの一覧表示
- ベースライン プロパティの表示
- ベースラインのプロジェクト、オブジェクト、タスク、または変更依頼の表示
- ベースライン情報の表示

ベースラインの削除

このサブコマンドにより、指定したベースラインを削除し、オプションでベースラインのプロジェクトおよび製品を削除します。*ccm_admin* ロールのユーザーだけが、ベースラインを削除できます。*ccm_admin* ロール以外のユーザーがベースラインを使用するのを防ぐには、ベースラインの削除のマーク付けを参照してください。

```
ccm baseline -delete ([-wp|-with_projects_and_products] |
            [-np|-no_projects_and_products]) baseline_spec...
```

baseline_spec...

削除するベースラインを指定します。詳細については、ベースラインの指定を参照してください。

`-np|-no_projects_and_products`

ベースラインのみを削除するよう指定します。ベースラインのプロジェクトと製品は削除されません。

`-wp|-with_projects_and_products`

ベースライン内のプロジェクトと製品を削除するよう指定します。これはデフォルト設定です。

ベースライン作成前に静的状態になかったプロジェクトと製品のみ削除されます。

例

- `20090213` という名前のベースラインを削除する。

  ```
  ccm baseline -delete 20090213
  ```

関連トピック

- ベースラインの比較
- ベースラインの一覧表示
- ベースラインの削除のマーク付け
- ベースラインの修正
- ベースライン プロパティの表示

ベースラインの一覧表示

このサブコマンドにより、指定した条件と一致するベースラインを一覧表示します。条件を指定しない場合、すべてのベースラインを一覧表示します。

```
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`

出力フォーマットでカラム ヘッダーを使用するよう指定します。詳細については、-ch|-column_headers を参照してください。

`-f|-format format`

コマンドの出力フォーマットを指定します。詳細については、-f|-format を参照してください。

`-gby|-groupby groupformat`

コマンドの出力をグループ化する方法を指定します。詳細については、-gby|-groupby を参照してください。

`-nch|-nocolumn_header`

出力フォーマットでカラム ヘッダーを使用しないよう指定します。詳細については、-nch|-nocolumn_headers を参照してください。

`-nf|-noformat`

カラム配置を使用しないよう指定します。詳細については、-nf|-noformat を参照してください。

`-ns|-no_sort`

コマンドの出力をソートしないよう指定します。詳細については、-ns|-nosort を参照してください。

`-purpose purpose`

指定した目的のベースラインのみ一覧表示するよう指定します。目的を指定しない場合、Synergy は任意の目的のベースラインを一覧表示します。

`-release release_spec`

指定したリリースのベースラインのみ一覧表示するよう指定します。リリースを指定しない場合、Synergy はすべてのリリースのベースラインを一覧表示します。

`-sep|-separator separator`

-f|-format オプションと一緒にのみ使用します。異なる区切り文字を指定します。詳細については、-sep|-separator を参照してください。

`-sby|-sortby sortspec`

コマンドの出力をソートする方法を指定します。詳細については、-sby|-sortby を参照してください。

`-u|-unnumbered`

コマンド出力の自動番号付けを抑止します。詳細については、-u|-unnumbered を参照してください。

例

- リリースが **2.2** 、目的が **Integration Testing**（統合テスト）であるベースラインを一覧表示する。

  ```
  ccm baseline -list -release 2.2 -purpose "Integration Testing"
  ```

  詳細なフォーマットの例については、フォーマットの使用例を参照してください。

関連トピック

- ベースラインの比較
- ベースラインの削除
- ベースラインの削除のマーク付け
- ベースラインの修正
- ベースラインの公開
- ベースラインのリリース
- 削除した ベースライン のリストア

ベースラインの削除のマーク付け

このサブコマンドにより、ベースラインの削除のマークを付けます。これにより、ベースラインを *deleted_baseline* 状態に遷移させ、ベースラインがメンバーの更新操作で使われるのを防ぎます。

ビルド マネージャまたは *ccm_admin* ロールのユーザーのみが、この操作を行うことができます。削除のマークを付けたベースラインは、*ccm_admin* ロールのユーザーが後で削除することができます。

```
ccm baseline -mfd|-mark_for_deletion baseline_spec...
```

`-mfd|-mark_for_deletion`

削除のマークを付けるベースラインを指定します。詳細については、ベースラインの指定を参照してください。

関連トピック

- ベースラインの一覧表示
- 削除した ベースライン のリストア
- ベースラインのプロジェクト、オブジェクト、タスク、または変更依頼の表示
- ベースライン情報の表示

ベースラインの修正

このサブコマンドにより、指定したベースラインの名前またはビルドを変更するか、プロジェクトおよび製品を更新して現在のデフォルト ベースライン バージョン テンプレートと一致するバージョンを使用します。ビルド マネージャと *ccm_admin* ロールのユーザーは、ベースラインを変更できます。

```
ccm baseline -modify [-n|-name name] [-b|-build build]
            [-v|-versions [-vt|-version_template version_template]
            [-skip_nonvisible_projects]] baseline_spec...
```

baseline_spec...

修正するベースラインを指定します。詳細については、ベースラインの指定を参照してください。

`-b|-build` *build*

指定したベースラインのビルド番号または ID を *build* 値に設定するよう指定します。**Turn3** など任意の 1 行のテキスト値を使用できます。通常、*build* 値は何らかの形のビルド番号を含みます。

ccm_admin ロールのユーザーは、任意の状態のベースラインのビルドを変更できます。ビルド マネージャは、*released* 状態を除くすべての状態のベースラインのビルドを変更できます。

`-name` *name*

指定したベースラインの名前を *name* に設定するよう指定します。*ccm_admin* ロールのユーザーは、任意の状態のベースラインの名前を変更できます。ビルド マネージャは、*released* 状態を除くすべての状態のベースラインの名前を変更できます。

`-skip_nonvisible_projects`

見えるワークエリアを持たないプロジェクトは変更しないよう指定します。

見えないために更新できない各ワークエリアについては、警告が表示されます。他に問題がなく、また `-skip_nonvisible_projects` を使用していない場合、操作は継続し、正常に行われます。`-skip_nonvisible_projects` を使用した場合、エラーが返され、操作は継続しますが、完了しません。最後にすべてのエラーが報告されます。メッセージによって、ワークエリアが見えないために失敗したのか、あるいはワークエリアを変更できなかったために失敗したのかがわかります。

`-v|-versions`

指定したベースラインのプロジェクトと製品のバージョンを変更するよう指定します。

`-version_template` を指定した場合、新しいバージョンにそのバージョン テンプレートが使用されます。指定しない場合、デフォルトのベースライン バージョン テンプレートが使用されます。詳細については、baseline_template を参照してください。

`-vt|-version_template` *`version_template`*

修正したベースライン内のプロジェクトまたは製品に使用するバージョン テンプレートを指定します。詳細については、**バージョン テンプレートの指定**を参照してください。

関連トピック

- ベースラインの一覧表示
- ベースラインのリリース
- ベースラインのプロジェクト、オブジェクト、タスク、または変更依頼の表示

ベースラインの公開

このサブコマンドにより、指定したベースラインを公開し、*published_baseline* 状態へ遷移させます。これにより、メンバーの更新操作時にそのベースラインが選択可能になります。

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

`-publish baseline_spec...`

公開するベースラインを指定します。`test_baseline` 状態のベースラインを *published_baseline* 状態に遷移させます。ビルド マネージャまたは *ccm_admin* ロールのユーザーのみが、この遷移を完了できます。詳細については、ベースラインの指定を参照してください。

関連トピック

- ベースラインのリリース
- ベースライン プロパティの表示
- ベースラインのプロジェクト、オブジェクト、タスク、または変更依頼の表示
- ベースライン情報の表示

ベースラインのリリース

このサブコマンドにより、指定したベースラインをリリースし、*released* 状態へ遷移させます。これにより、メンバーの更新操作時にそのベースラインが選択可能となり、ベースラインがリリースに適していることを示します。ビルド マネージャと *ccm_admin* ロールのユーザーがベースラインをリリースできます。

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

baseline_spec...

ベースライン名が使用できる場合は、*baseline_name* またはクエリ選択セット参照形式を使用します。詳細については、ベースラインの指定を参照してください。

`-c|-comment` *comment*

すべてのベースライン プロジェクトとそのメンバーを *released* 状態にチェックインするとき、それらに追加するコメントを指定します。*comment* には、複数行を含むことができ、円記号でコード化した値を使用できます。

このオプションは、`-commentedit` および `-commentfile` と一緒に使用できます。`-commentedit` オプションを使用した場合、コメントはデフォルトのテキスト エディタで表示されます。

`-ce|-commentedit`

コメントの作成と編集用にデフォルトのテキスト エディタを起動するよう指定します。テキスト エディタで保存した結果が最終的なコメントとして使用されます。このオプションは、`-comment` および `-commentfile` オプションと一緒に使用できます。

`-cf|-commentfile` *file_path*

指定したファイルの内容をコメントとして使用するよう指定します。`-comment` を指定した場合、それがコメントに追加されます。このオプションは、`-commentedit` オプションと一緒に使用できます。

`-rb|-release_baseline`

指定した *baseline_spec* を持つベースラインをリリースします。また、ベースラインのすべてのプロジェクトとそのメンバーを *released* 状態にチェックインします。

関連トピック

- ベースライン プロパティの表示
- ベースラインのプロジェクト、オブジェクト、タスク、または変更依頼の表示
- ベースライン情報の表示

削除した ベースライン のリストア

このサブコマンドにより、指定したベースラインの削除マークを元に戻します。この操作は、ベースラインをデータベースから削除する前に行った場合にのみ成功します。

ビルド マネージャと *ccm_admin* ロールのユーザーは、ベースラインを削除できます。

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

baseline_spec...

ベースライン名が使用できる場合は、*baseline_name* またはクエリ選択セット参照形式を使用します。詳細については、ベースラインの指定を参照してください。

関連トピック

- ベースラインの比較
- ベースラインの修正
- ベースラインの公開
- ベースライン プロパティの表示
- ベースラインのプロジェクト、オブジェクト、タスク、または変更依頼の表示

ベースライン プロパティの表示

このサブコマンドにより、指定したベースラインの指定したプロパティを表示します。

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

baseline_spec...

ベースライン名が使用できる場合は、*baseline_name* またはクエリ選択セット参照形式を使用します。詳細については、ベースラインの指定を参照してください。

関連トピック

- ベースラインのプロジェクト、オブジェクト、タスク、または変更依頼の表示
- ベースライン情報の表示

ベースラインのプロジェクト、オブジェクト、タスク、または変更依頼の表示

このサブコマンドにより、ベースラインのプロジェクト、ファイル、ディレクトリ、タスク、コンポーネント タスク、または変更依頼を表示します。変更依頼については、ベースラインに完全に含まれているか部分的に含まれているかを示します。

```
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...

ベースライン名が使用できる場合は、*baseline_name* またはクエリ選択セット参照形式を使用します。詳細については、ベースラインの指定を参照してください。

-ch|-column_header

出力フォーマットでカラム ヘッダーを使用するよう指定します。詳細については、-ch|-column_headers を参照してください。

-f|-format *format*

コマンドの出力フォーマットを指定します。詳細については、-f|-format を参照してください。

-gby|-groupby *groupformat*

コマンドの出力をグループ化する方法を指定します。詳細については、-gby|-groupby を参照してください。

-nch|-nocolumn_header

出力フォーマットでカラム ヘッダーを使用しないよう指定します。詳細については、-nch|-nocolumn_headers を参照してください。

`-nf|-noformat`

カラム配置を使用しないよう指定します。詳細については、-nf|-noformat を参照してください。

`-ns|-no_sort`

コマンドの出力をソートしないよう指定します。詳細については、-ns|-nosort を参照してください。

`-sep|-separator` *`separator`*

-f|-format オプションと一緒にのみ使用します。異なる区切り文字を指定します。詳細については、-sep|-separator を参照してください。

`-sby|-sortby` *`sortspec`*

コマンドの出力をソートする方法を指定します。詳細については、-sby|-sortby を参照してください。

`-u|-unnumbered`

コマンド出力の自動番号付けを抑止します。詳細については、-u|-unnumbered を参照してください。

関連トピック

- ベースラインの修正
- ベースラインの公開
- ベースラインのリリース

ベースライン情報の表示

このサブコマンドにより、指定したベースラインの一般情報を表示します。

```
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...

ベースライン名が使用できる場合は、*baseline_name* またはクエリ選択セット参照形式を使用します。詳細については、ベースラインの指定を参照してください。

-ch|-column_header

出力フォーマットでカラム ヘッダーを使用するよう指定します。詳細については、-ch|-column_headers を参照してください。

-f|-format *format*

コマンドの出力フォーマットを指定します。詳細については、-f|-format を参照してください。

-nch|-nocolumn_header

出力フォーマットでカラム ヘッダーを使用しないよう指定します。詳細については、-nch|-nocolumn_headers を参照してください。

-nf|-noformat

カラム配置を使用しないよう指定します。詳細については、-nf|-noformat を参照してください。

-sep|-separator *separator*

-f|-format オプションと一緒にのみ使用します。異なる区切り文字を指定します。詳細については、-sep|-separator を参照してください。

関連トピック

- ベースラインの削除
- ベースラインの削除のマーク付け
- ベースラインの修正
- ベースラインの公開

- ベースラインのリリース

説明と用途

ベースラインは、特定の時点でデータを表すために使われるプロジェクトとタスクのセットです。ベースラインにはいろいろな用途があります。更新を行うとき、Rational Synergy は新規変更を探す開始点としてベースラインを使用します。また、2 つのベースラインを比較して、特定のビルドを基準にどのような変更が行われたかを確認できます。Rational Change を使用していれば、ベースラインを使用して変更依頼レポートを作成できます。

通常はビルド マネージャがベースラインを作成します。開発者は自分のビルドを他の開発者にも使用できるようにはしないので、ベースラインを作成する必要がありません。

ビルドを行ったら直ちにベースラインを作成すると便利です。ベースラインを作成し、すべての開発者に公開することなくテスト グループに公開できます。ビルドを行うと同時にベースラインを作成すると、後にそのビルドの修正をする必要があるときに利用できるビルドの詳細が Rational Synergy に保存されます。

`Integration Testing` と `System Testing` ごとにビルドを作成しておくと、テスターおよび開発者はそのビルドに盛り込まれた一連の変更点を参照できます。一般的に同じリリースと目的を持つすべてのプロジェクトのベースラインを作成します。たとえば、`Integration Testing` ビルド用には、そのリリースのすべての `Integration Testing` プロジェクトを使用してベースラインを作成します。

> **注記**：ベースラインを作成するとき、ベースラインに含めるプロジェクトのリストを指定します。変更を参照するための完全なセットとなるように、必ずベースラインに関連するすべてのプロジェクトを含めてください。

ベースラインは、テンプレートを使用するプロジェクトのベースラインを定義するため、プロセス ルールで使用できます。たとえば、ビルド マネージャは、静的プロジェクト `toolkit-int_20040913`、`calculator-int_20040913` などを含む `Integration Build 20040913` という名前のベースラインを作成できます。表示されている数字は、ベースラインが作成された日付（*yyyymmdd*）です。

プロセス ルールは、そのプロジェクトが特定のベースラインを使用することを指定できます。プロセス ルールを参照するプロジェクトは、そのベースラインを使用して、更新時に使用するベースライン プロジェクトを識別します。たとえば、現在のリリースの `Integration Testing` プロセスが `Integration Build 20040913` ベースラインを使用するように指定した場合、開発者の `calculator-bob` プロジェクトは `calculator-20040913` をそのベースライン プロジェクトとして選択します。

ベースラインの使用には、以下の利点があります。

- ビルド マネージャがビルドとテストに成功したプロジェクト セットを保存するための、手軽な方法が提供される。
- プロセス ルールがより柔軟になる。特定のベースラインまたは特定の特性を持つ最新のベースラインを指定できるので、ビルド マネージャはチームのプロセスをより精密に管理できます。新しいベースラインで問題が見つかった場合は、ビルド マネージャはチームのベースラインをビルドに成功した前のベースラインに戻すことができます。

- 更新操作では、ベースラインを使用してどのタスクを評価するかを簡素化するので、更新のパフォーマンスを改善できる。更新の候補を算出するとき、このベースライン以降のタスクのみを考慮すればよくなります。ベースラインの作成時に、手動で更新するプロジェクトの場合には、一連のタスクをプロジェクト グルーピングまたはプロジェクト自体から取り込みます。また、リリースが異なる場合は、プロジェクト グルーピングのベースラインからタスクを新しいベースラインに追加します。
- チーム メンバーはベースラインを比較して、どのタスクが最新のベースラインに導入されたか、またはベースラインに特定のタスクが含まれているかどうかを識別することができる。これは、どの機能をテストすべきか、また、既知の問題が特定のビルドで解決されているのかどうかを認知する必要のあるテスターにとっては有用な仕組みです。
- 成功した最新のビルドと一致するように、プロジェクトの更新を指定できる。

新しいベースラインの作成方法

prep 状態のプロジェクトと静的プロジェクトの両方を新しいベースラインに追加できます。ただし、ビルド管理プロジェクトをベースラインに追加する場合、実際のプロジェクトは追加されません。代わりに、プロジェクトのコピーを作成してベースラインに追加し、チェックインします。不要な再ビルドを発生させないようにするために、ビルド管理プロジェクトとそのワークエリアはそのまま維持されます。さらに、ビルド管理プロジェクトのメンバーである非静的製品すべてについて、新しいバージョンをチェックアウトおよびチェックインします。それ以外の場合、新しいプロジェクトはビルド管理プロジェクトと同じメンバーを持ちます。新しいプロジェクトと製品を、ベースラインの目的に関連する `member_status` としてチェックインします。この `member_status` が有効な状態ではない場合、プロジェクトと製品を *integrate*（統合）状態としてチェックインします。

たとえば、`Integration Testing`（統合テスト）用ベースラインには、*integrate*（統合）状態にあるプロジェクトと製品が含まれます。

ビルド管理プロジェクトにプロジェクトまたは製品以外の非静的メンバーが含まれている場合、そのビルド管理プロジェクトはベースラインに追加できません。このようなプロジェクトをベースラインに追加するためには、その前にその非静的メンバーをチェックインする必要があります。さらに、更新プロパティに完了していないタスクが含まれるプロジェクトは、ベースラインに追加できません。

新しいプロジェクトまたは新しい製品のバージョンは、ビルド管理プロジェクトのバージョン、日付、および必要に応じてバージョンを一意にするために追加される増分番号をベースに作成されます。たとえば、プロジェクト `ccm_gui-sol_int` をベースラインの一部として保存する場合、新しいベースライン プロジェクトは `ccm_gui-sol_int_20040709` のようになります。既存の文字列に下線、日付および増分番号を追加できない場合（かつ 32 文字の限度を超えない場合）、日付と番号だけが使われます。

ベースラインを作成すると、履歴表示リンクが変更され、既存のビルド管理プロジェクトが新しいベースライン プロジェクトからチェックアウトされたときに表示されるようになります。さらに、ベースライン プロジェクトで作成される製品から既存の prep 製品がチェックアウトされたように見えるよう、プロジェクトの履歴が更新されます。

ベースラインの一部として作成される新しいプロジェクトにはワークエリアがありません。このプロジェクトにワークエリアが必要な場合は、ベースライン作成後にワークエリア管理を有効にする必要があります。見えるワークエリアを持つプロジェクトをベースラインに追加すると、ワークエリア コンフリクトの検査が行われます。解決できないコンフリクトが見つかった場合、ベースライン作成操作は失敗します。この問題を解決するには、プロジェクトをリコンサイルする必要があります。

見えないワークエリアを持つプロジェクトをベースラインに追加した場合、最後にビルドした製品がデータベースにコピーされないことがあります。このような場合、ベースラインにはデータベースにあるものが入り、見えないワークエリアにあるものは入りません。この問題を防ぐためには、ビルド マネージャは、ベースラインに追加するプロジェクトの見えないワークエリアの変更をすべてデータベースに同期させる必要があります。これは、プロジェクトをベースラインに追加する前に行っておく必要があります。

`baseline` コマンドを使用して以下の操作を行います。

- 既存の prep 階層または一連の階層からベースラインを作成する。
- 最新のテスト済みの変更を開発者に公開するために、Tested Tasks フォルダに手動で実装する代わりに、ベースラインを保存する。
- 特定のベースラインに関する情報または関連プロジェクト、オブジェクトおよびタスクを表示する。
- ベースラインを一覧表示する。
- ベースラインを修正または名前を変更する。
- ベースラインをリリースするか、2 つのベースラインを比較する。
- 既存のベースラインを削除するか、ベースラインに削除のマークを付ける。
- 削除したベースラインをリストアする。

ベースラインの作成またはリリースは、ビルド マネージャとして行う必要があります。ベースラインを削除、またはリリースしたベースラインのビルドを修正するには、*ccm_admin* ロールを持っている必要があります。

バージョン テンプレートの指定

`version_template` は任意の文字列で、*`%keyword`* または *`%{keyword}`* 形式のオプションのキーワードを持ちます。キーワードとしては、任意の Rational Synergy 属性、特殊キーワード `%baseline_name`、`%date`、`%build` を使用できます。

属性を展開するとき、検査対象のビルド管理プロジェクトまたは製品の対応する属性値が使用されます。指定したキーワード名について属性または組み込み済みキーワードが見つからない場合は、そのキーワードの代わりに空の文字列が使用されます。

ベースライン作成時に静的となったプロジェクトと製品のバージョンは、*`version_template`* と一致するように更新されます。ただし、ベースライン作成前に静的状態にあったプロジェクトのバージョンは更新されません。たとえば、CM/6.5 SP2 ベースラインが CM/6.5 SP2 ベースラインの 20 の既存の静的プロジェクトと、CM/6.5 SP2 の 5 つの新しいプロジェクトで作成された場合、5 つの新しいプロジェクトのバージョンのみが更新されます。

ベースライン内のプロジェクトまたは製品についてインスタンス化した version_template にバージョン文字列に使用できない文字が含まれている場合、それらの文字はデフォルトのバージョン文字列の置換文字に置き換わります。この文字は、オプション baseline_template_repl_char を使用して、ccm.ini ファイルに指定されています。この文字のデフォルトは下線（_）です。たとえば、%platform がプロジェクト バージョン テンプレートの一部であり、ビルド管理プロジェクトのプラットフォームが SPARC-solaris の場合、バージョン文字列には文字列 SPARC_solaris が含まれます。あるいは、%release が製品バージョン テンプレートの一部であり、prep 製品が CM/6.6a リリースの場合、このバージョン文字列には文字列 CM_6.6a が含まれます。

ベースライン内のプロジェクトまたは製品についてインスタンス化した *version_template* がそのプロジェクトまたは製品の別のバージョンですでに使用されている場合、そのバージョンは、下線（_）と そのバージョンを一意にする 1 から始まる最初の整数を追加することによって、一意のバージョンとなります。これによってバージョン文字列が長くなりすぎる場合は、そのプロジェクトまたは製品には現在の日付をベースとしたバージョンが使用され、警告が表示されます。

-version_template を指定しない場合、デフォルト（すなわち、保存されている）テンプレートが使用されます。

プロジェクトのワークエリア テンプレートにバージョンが含まれる場合、ワークエリアが更新されます。ワークエリアが見えないために更新不可能であり、また -skip_nonvisible_projects が使用されていない場合、操作は継続しますがすべてのエラーが報告されます。ワークエリアが見える状態であっても、適切なファイル アクセス権限の欠如やディスク領域の不足などの他の理由で更新できない場合、操作は継続しますがすべての失敗が報告されます。

バージョン テンプレートの代替キーワード構文

キーワード構文により、使用したキーワードに基づいて展開動作を制御できます。

- %{*keyword:-string*} *keyword* にヌル以外の値が設定されている場合は正常に展開され、それ以外の場合は *string* に展開されます。ただし、キーワードが見つからないときに何も表示しないようにするには *string* を空白にします。
- %{*keyword:+string*} *keyword* にヌル以外の値が設定されている場合は *string* に展開され、それ以外の場合は空白の文字列に展開されます（置き換えなし）。

  例として、platform が存在するかどうかにしたがって“solaris_7.0”または“7.0”を得るには、以下のように指定します。

  %{platform:-}%{platform:+_}6.6a

  - %{platform:-} は、プラットフォームが solaris の場合は solaris に展開され、それ以外の場合は空白の文字列に展開されます。
  - %{platform:+_} は、プラットフォームが存在する場合は _ に展開され、それ以外の場合は空白の文字列に展開されます。

bom コマンド

詳細については、説明と用途を参照してください。bom コマンドは、BOM（部品構成表）の表示サブコマンドをサポートします。

BOM（部品構成表）の表示

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

`file_spec`

BOM を表示する製品を指定します。通常、この情報は管理製品に対してのみ存在します。詳細については、ファイルの指定を参照してください。

説明と用途

このコマンドにより、製品の BOM を表示します。

candidates コマンド

詳細については、説明と用途を参照してください。`candidates` コマンドは、候補の表示サブコマンドをサポートします。

候補の表示

```
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`

出力フォーマットでカラム ヘッダーを使用するよう指定します。詳細については、-ch|-column_headers を参照してください。

`file_spec`

候補のバージョンを一覧表示する対象のオブジェクトまたはディレクトリ エントリの名前を指定します。詳細については、ファイルの指定を参照してください。

`-f|-format` *`format`*

出力フォーマットでカラム ヘッダーを使用するよう指定します。詳細については、-f|-format を参照してください。

キーワードには、組み込見込み済みのもの（`%fullname`、`%displayname`、`%objectname`）、あるいは `%modify_time`、`%status` などの既存の属性の名前を使用できます。

キーワードのリストについては、組み込み済みキーワードを参照してください。

`-gby|-groupby` *`groupformat`*

コマンドの出力をグループ化する方法を指定します。詳細については、-gby|-groupby を参照してください。

`-nch|-nocolumn_header`

出力フォーマットでカラム ヘッダーを使用しないよう指定します。詳細については、-ch|-column_headers を参照してください。

`-nf|-noformat`

カラム配置を使用しないよう指定します。詳細については、-nf|-noformat を参照してください。

`-ns|-nosort|-no_sort`

コマンドの出力をソートしないよう指定します。詳細については、-ns|-nosort を参照してください。

`-r|-recommend`

推奨されるバージョンが表示されます。このバージョンは、Rational Synergy が選択ルールに基づいて選択するバージョンです。デフォルトのフォーマットでは、推奨バージョンにはアスタリスク（*）が付きます。ユーザー定義フォーマットでは、`%recommended` キーワードを使用して算出した推奨バージョンを示します。

`-sby|-sortby` *`sortspec`*

コマンドの出力をソートする方法を指定します。詳細については、-sby|-sortby を参照してください。

`-sep|-separator` *`separator`*

`-f|-format` オプションと一緒にのみ使用します。異なる区切り文字を指定します。詳細については、-sep|-separator を参照してください。

`-u|-unnumbered`

コマンドの出力の自動番号付けを抑止します（すなわち、出力に番号付けがされません）。詳細については、-u|-unnumbered を参照してください。

例

- ディレクトリ内の現在のプロジェクトのメンバーとなり得る `Xincls.h` のバージョンを一覧表示し、使用するバージョンを推奨する。

```
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 *
```

関連トピック

- update コマンド
- use コマンド

説明と用途

`candidates` コマンドにより、ディレクトリ エントリ内で使用操作または更新操作を実行するときに選択可能なオブジェクトの全バージョンを一覧表示します。オブジェクトの名前、タイプおよびオブジェクト インスタンス属性の値がディレクトリ エントリのものと一致する場合、そのオブジェクトが使用対象の候補となります。

出力には、各オブジェクト バージョンの名前、バージョン、状態、所有者、それが作成されたプロジェクト、インスタンス、関連付けられているタスク番号が表示されます。`cand` コマンドは、番号フォーマットのオプションをサポートし、クエリ選択リストを設定します。フォーマット文字列のプロパティ *keyword recommended_version* は、推奨バージョンを表します。これにより、推奨バージョンは「*」、非推奨バージョンは空白の文字列に展開されます。

cat コマンド

詳細については、説明と用途を参照してください。cat コマンドは、ソース内容の表示サブコマンドをサポートします。

ソース内容の表示

cat コマンドにより、オブジェクトのソースを表示します。

```
ccm cat|type file_spec...
```

file_spec

表示するファイルを指定します。詳細については、ファイルの指定を参照してください。

例

- csrc オブジェクトである foo.c-9 オブジェクトのバージョンの 2 つ目のインスタンスを表示する。

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

関連トピック

- view コマンド

説明と用途

このサブコマンドは、現在ディレクトリのメンバーでないオブジェクトの内容を表示する場合に役立ちます。コンテキスト プロジェクト内のファイルを指定し、対応するワークエリアが見える場合、ワークエリア ファイルが表示されます。見えない場合は、データベースから一時コピーが表示されます。

change_type コマンド

詳細については、説明と用途を参照してください。change_type コマンドは、オブジェクトのタイプの変更サブコマンドをサポートします。

オブジェクトのタイプの変更

このサブコマンドにより、指定したファイルまたはディレクトリのタイプを変更します。

```
ccm change_type -t|-type new_type [-task task_spec} file_spec
```

file_spec

変更するファイルまたはディレクトリを指定します。詳細については、ファイルの指定を参照してください。

-task *task_spec*

チェックアウトしたディレクトリと新規作成したオブジェクトを、指定したタスクに関連付けます。カレント タスクが設定されており、別のタスクを指定しない場合は、チェックアウトしたディレクトリと新規作成するオブジェクトはカレント タスクに自動的に関連付けられます。

task_spec には、1 つのタスクを指定できます。*task_spec* を空白の文字列に設定すると、サポートしないタスクがないことを意味します。

-type *new_type*

オブジェクトの新しいタイプを指定します。*new_type* には、1 つのタスクを指定できます。

説明と用途

このサブコマンドにより、ファイルまたはディレクトリのタイプを変更します。

タイプを変更すると、Rational Synergy は指定されたタイプでオブジェクトの新しいバージョンを作成します。指定したオブジェクトが *working*（作業中）状態にある場合、Rational Synergy はそのオブジェクトを新しいオブジェクトに置き換え、指定されたオブジェクトをデータベースから削除します。

そのオブジェクトがプロジェクトのメンバーであり、`change_type` コマンドがそのプロジェクト内で実行された場合、Rational Synergy はプロジェクト内で旧オブジェクトを新しいオブジェクトに置き換えます。親ディレクトリが修正可能ではない場合、自動的にチェックアウトされます。プロジェクトは書き込み可能である必要があります。

オブジェクトが複数のプロジェクトのメンバーである場合、またはオブジェクトがメンバーとなっているプロジェクト内でコマンドを実行しない場合、コマンドは失敗します。

checkin コマンド

詳細については、説明と用途を参照してください。checkin コマンドは、以下のサブコマンドをサポートします。

- プロジェクトのチェックイン
- タスクのチェックイン
- オブジェクトのチェックイン

プロジェクトのチェックイン

このサブコマンドにより、プロジェクトをチェックインし、オプションとしてプロジェクトのメンバーであるソースと製品をチェックインします。プロジェクトを *integrate* または *released* などの静的状態にチェックインするときは、プロジェクトのメンバーも静的状態にある必要があります。プロジェクトを管理して、ワークエリアに対して行った変更が自動的にデータベースと同期されるようにするには、このコマンドをプロジェクトのワークエリアが見えるクライアントで実行する必要があります。

```
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`*

すべてのベースライン プロジェクトとそのメンバーを *released* 状態にチェックインするとき、それらに追加するコメントを指定します。*`comment`* には、複数行を含むことができ、円記号でコード化した値を使用できます。

このオプションは、`-commentedit` および `-commentfile` と一緒に使用できます。`-commentedit` オプションを使用した場合、コメントはデフォルトのテキスト エディタで表示されます。

`-ce|-commentedit`

コメントの作成と編集用にデフォルトのテキスト エディタを起動するよう指定します。テキスト エディタで保存した結果が最終的なコメントとして使用されます。このオプションは、`-comment` および `-commentfile` オプションと一緒に使用できます。

`-cf|-commentfile` *`file_path`*

指定したファイルの内容をコメントとして使用するよう指定します。`-comment` を指定した場合、それがコメントに追加されます。このオプションは、`-commentedit` オプションと一緒に使用できます。

`-cr|-commentreplace`

通常、新しく指定したコメントは既存のコメントに追加されます。`-cr` オプションを使用すると、既存のコメントが置き換えられます。ただし、置き換えられるのは書き込み可能オブジェクトのコメントのみです。

`-h|-hierarchy`

プロジェクト階層に（ソース、製品、プロジェクトに対する）チェックイン範囲を適用します。

`-nc|-nocomment`

コメントを尋ねるプロンプトを表示しないよう指定します。`-c|-comment`、`-cf|-commentfile` または `ce|-commentedit` のいずれのオプションにもコメントを指定せず、このオプションも指定しなかった場合、このコマンドでチェックインするすべてのオブジェクトに使用するコメントを入力するよう要求されます。コメントの入力要求を抑止するには `-nc|-nocomment` オプションを使用します。

`-products`

現在のプロジェクトのすべての製品メンバーをチェックインします。`-h|-hierarchy` も指定した場合、階層内のすべてのプロジェクト メンバーに適用されます。

`-p|-project`

指定したプロジェクト内のサブプロジェクトもチェックインするよう指定します。`-h|-hierarchy` も指定した場合、階層内のすべてのサブプロジェクトに適用されます。

`project_spec`

チェックインするプロジェクトを指定します。詳細については、プロジェクトの指定を参照してください。

`-ps|-product_state`

チェックインする任意の製品オブジェクトの状態を指定します。指定しない場合、デフォルトの次の状態が自動的に決定されます。

製品チェックイン時の、製品オブジェクトの状態を指定します。この設定は、階層および非階層チェックインの両方に適用されます（すなわち、このオプションは `-s` オプションを必要としません）。

`-s|-state` *`state`*

チェックインするプロジェクトの状態を指定します。指定しない場合、デフォルトの次の状態が自動的に決定されます。

`-source|-sources`

ソース オブジェクトである現在のプロジェクトのすべてのメンバーをチェックインします。ソース オブジェクトとは、製品としてマークされていないファイルまたはディレクトリのことです。`-h|-hierarchy` も指定した場合、階層内のすべてのソース オブジェクトに適用されます。

`-ss|-source_state` *`source_state`*

ソース オブジェクトのチェックイン時の状態を指定します。指定しない場合、デフォルトの次の状態が自動的に決定されます。

`-task` *`task_spec`*

チェックインする任意のソース オブジェクトに関連付けるタスクを指定します。ソース オブジェクトとは、製品としてマークされていないファイルまたはディレクトリのことです。*`task_spec`* には、1 つのタスクを指定できます。詳細については、タスクの指定を参照してください。

警告

プロジェクト バージョンを修正不可状態にチェックインする場合は、すべてのメンバーが修正不可状態にあることを確認してください。修正可能メンバーを含むプロジェクトは、修正不可状態にチェックインできません。

例

- `projB-3` プロジェクトをチェックインする。

  ```
  ccm ci -c "configuration sent to customer A" -p projB-3
  ```

- `tools-5` プロジェクトのすべてのメンバーをチェックインする。製品メンバーは *checkpoint*（チェックポイント）状態、ソース（非製品）メンバーは *integrate*（統合）状態に。

  ```
  ccm ci -p tools-5 -products -s checkpoint
  ccm ci -p tools-5 -source -ss integrate
  ```

関連トピック

- checkout コマンド
- task コマンド

タスクのチェックイン

このサブコマンドにより、タスクに関連するオブジェクトを修正不可状態にチェックインし、タスクを完了状態へ遷移させてタスクを完了します。このサブコマンドは `ccm task -complete` コマンドと同じです。このサブコマンドを使用するには、タスクの担当者または管理者である必要があります。

```
ccm ci|checkin -task (task_spec|(current|default))
               [-c|-comment comment_string] [-ce|-commentedit]
               [-cf|-commentfile file_path] [-cr|-commentreplace]
```

`-c|-comment` *`comment_string`*

-c|-comment comment を参照してください。

`-ce|-commentedit`

-ce|-commentedit を参照してください。

`-task (task_spec|(current|default))`

チェックインまたは完了するタスクを指定します。*`task_spec`* には、1 つのタスクを指定できます。詳細については、タスクの指定を参照してください。

関連トピック

- checkout コマンド
- task コマンド

オブジェクトのチェックイン

このサブコマンドにより、ファイルやディレクトリなど特定のオブジェクトをチェックインします。オブジェクトに関連しているプロジェクトを管理して、ワークエリアに対して行った変更が自動的にデータベースと同期されるようにするには、このコマンドをプロジェクトのワークエリアが見えるクライアントで実行する必要があります。

このサブコマンドは、1 つまたは複数の引数を指定し、かつ `-project` オプションを指定しない場合に適用されます。

```
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`*

-c|-comment comment を参照してください。

`-ce|-commentedit`

-ce|-commentedit を参照してください。

`-cf|-commentfile` *`file_path`*

-cf|-commentfile file_path を参照してください。

`-cr|-commentreplace`

-cr|-commentreplace を参照してください。

`-nc|-nocomment`

コメントを尋ねるプロンプトを表示しないよう指定します。`-c|-comment`、`-cf|-commentfile` または `ce|-commentedit` のいずれのオプションにもコメントを指定せず、このオプションも指定しなかった場合、このコマンドでチェックインするすべてのオブジェクトに使用するコメントを入力するよう要求されます。コメントの入力要求を抑止するには `-nc|-nocomment` オプションを使用します。

`file_spec`

チェックインするファイルまたはディレクトリを指定します。詳細については、ファイルの指定を参照してください。

`-s|-state` *`state`*

チェックインするオブジェクトの状態を指定します。指定しない場合、デフォルトの次の状態が自動的に決定されます。

`-task` *`task_spec`*

チェックインする任意のソース オブジェクトに関連付けるタスクを指定します。ソース オブジェクトとは、製品としてマークされていないファイルまたはディレクトリのことです。*`task_spec`* には、1 つのタスクを指定できます。詳細については、タスクの指定を参照してください。

例

- 現在のバージョンの `foo.c` を、*visible* 状態でチェックインする。

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

- 3 つのファイル（`clear.c`、`concat.c`、`display.c`）をチェックインする。
- 新しいコメントを何も追加せずに、`utils` ディレクトリをチェックインする。

  ```
  ccm ci -nc utils
  ```

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

- `c_includes` シンボリック リンクを *checkpoint*（チェックポイント）状態にチェックインする（UNIX のみ）。

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

関連トピック

- checkout コマンド
- task コマンド

説明と用途

checkin により、1 つまたは複数のオブジェクトをチェックインし、必要に応じて次の状態を設定します。

ソース（非製品オブジェクト）、製品、およびプロジェクトの各オブジェクトをチェックインし、チェックインするオブジェクトにタスク番号を割り当てたり、チェックインするオブジェクトのコメントを追加、修正、または置換できます。

> **注記**：変更は 1 つのワークエリアからのみ行い、チェックインはそのワークエリアを見えるようにして行う必要があります。

checkout コマンド

詳細については、説明と用途を参照してください。checkout コマンドは、以下のサブコマンドをサポートします。

- オブジェクトのチェックアウト
- プロジェクトのチェックアウト

オブジェクトのチェックアウト

このサブコマンドにより、ファイルまたはディレクトリをチェックアウトします。

```
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`

すべてのベースライン プロジェクトとそのメンバーを *released* 状態にチェックインするとき、それらに追加するコメントを指定します。*comment* には、複数行を含むことができ、円記号でコード化した値を使用できます。

このオプションは、`-commentedit` および `-commentfile` と一緒に使用できます。`-commentedit` オプションを使用した場合、コメントはデフォルトのテキスト エディタで表示されます。

`-ce|-commentedit`

コメントの作成と編集用にデフォルトのテキスト エディタを起動するよう指定します。テキスト エディタで保存した結果が最終的なコメントとして使用されます。このオプションは、`-comment` および `-commentfile` オプションと一緒に使用できます。

`-cf|-commentfile file_path`

指定したファイルの内容をコメントとして使用するよう指定します。`-comment` を指定した場合、それがコメントに追加されます。このオプションは、`-commentedit` オプションと一緒に使用できます。

file_spec

チェックアウトするファイルまたはディレクトリを指定します。オブジェクトは、チェックアウトのコンテキスト プロジェクトを与えるためワークエリア参照形式またはプロジェクト参照形式を使用して指定する必要があります。詳細については、ファイルの指定を参照してください。

`-t|-to version|file_spec`

バージョンを指定し、新しい非プロジェクト オブジェクトのバージョンを変更するか、あるいは新しいプロジェクトまたはプロジェクト階層のバージョンを指定します。

デフォルトで、-to 引数は新しいバージョンとして解釈されます。たとえば、以下のコマンドを実行する場合、

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

新しいオブジェクト バージョンは foo.c-bar です。

名前を変更するには、変更後のオブジェクト名とバージョンを引数に含める必要があります。たとえば、以下のコマンドを実行する場合、

```
ccm co foo.c -to bar.c-1
```

新しいオブジェクト バージョンは bar.c-1 です。

プロジェクトをチェックアウトする場合は、バージョンのみを指定できます。プロジェクトの階層をチェックアウトする場合、新しいバージョンはプロジェクトとそのサブプロジェクトに使用されます。新しいバージョンを階層内のプロジェクトの旧バージョンに対応付ける場合は、-versions オプションを使用します。-to オプションと -versions オプションは一緒には使用できません。また、-to オプションまたは -version オプションを指定しない場合、デフォルトの次のバージョンは Rational Synergy に組み込まれたアルゴリズムを使用して自動的に計算されます。

現在のプロジェクトで使用しているオブジェクトの新しいバージョンをチェックアウトする場合、新しくチェックアウトしたバージョン（"to" バージョン）もプロジェクトで使用されます。

> **注記**：書込み禁止ディレクトリ内の新しいオブジェクト名にチェックアウトすると、新しいディレクトリ バージョンが自動的にチェックアウトされます。
>
> 共有プロジェクト内にいて、現在のディレクトリが書き込み禁止の場合、そのディレクトリはチェックアウトされ、デフォルト（または指定した）タスクと自動的に関連付けられ、*integrate*（統合）状態にチェックインされます。この機能を無効にする場合は、初期設定ファイル内の shared_project_directory_checkin を FALSE に設定します（shared_project_directory_checkin を参照してください）。

-task *task_spec*

新たにチェックアウトしたオブジェクトを関連付けするタスクを指定します。カレント タスクが設定されており、別のタスクを指定しない場合は、チェックアウトするオブジェクトはカレント タスクに自動的に関連付けられます。（詳細については、カレント タスクの設定または解除**を参照してください）。***task_spec* には、1 つのタスクを指定できます。詳細については、タスクの指定を参照してください。

例

- `foo.c` のバージョン 1 からバージョン patch1 をチェックアウトする（`foo.c` のバージョン 3 は現在のディレクトリ内にある）。

  ```
  ccm co -c "patch1:fix symbol table bug" -to patch1 foo.c-1
  ```

- 現在バージョン 4 である `utils¥tools`（Windows）ディレクトリまたは `utils/tools`（UNIX）ディレクトリをチェックアウトする。

  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
  ```

- コメントを設定し、チェックアウトするオブジェクト バージョンにタスクを関連付ける。

  ```
  ccm co -c "comment string" -task task_number object_name1 object_name2
  ```

関連トピック

- checkin コマンド
- copy_project コマンド

プロジェクトのチェックアウト

このサブコマンドにより、ワークエリアにプロジェクトのコピーを作成できます。このワークエリアで、プロジェクト メンバーを変更できます。このコマンドは、現バージョンの Rational Synergy では copy_project 操作と呼ばれています。詳細については、copy_project コマンドを参照してください。

```
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*

すべてのベースライン プロジェクトとそのメンバーを *released* 状態にチェックインするとき、それらに追加するコメントを指定します。*comment* には、複数行を含むことができ、円記号でコード化した値を使用できます。

このオプションは、`-commentedit` および `-commentfile` と一緒に使用できます。`-commentedit` オプションを使用した場合、コメントはデフォルトのテキスト エディタで表示されます。

`-cb|-copy_based`

ワークエリアをコピーベースに指定します。

`-ce|-commentedit`

コメントの作成と編集用にデフォルトのテキスト エディタを起動するよう指定します。テキスト エディタで保存した結果が最終的なコメントとして使用されます。このオプションは、`-comment` および `-commentfile` オプションと一緒に使用できます。

`-cf|-commentfile` *file_path*

指定したファイルの内容をコメントとして使用するよう指定します。`-comment` を指定した場合、それがコメントに追加されます。このオプションは、`-commentedit` オプションと一緒に使用できます。

`-lb|-link_based|-ncb|-not_copy_based`

ワークエリアをリンク ベースに指定します。このオプションは、UNIX ユーザーのみが使用できます。このオプションは、`-p` オプションと一緒に使用する必要があります。

詳細については、work_area コマンドを参照してください。

`-mod|-modifiable_wa`

ワークエリア内のファイルに、チェックアウトされていない場合でも修正可能となるようにアクセス許可を設定します。デフォルトは `-nmod|-not_modifiable_wa` です。

`-nmod|-not_modifiable_wa`

working などの書き込み可能状態の場合にのみ、デフォルトでワークエリア内のファイルが修正可能となるようにアクセス許可を設定します。これはデフォルト設定です。

`-no_u|-no_update`

プロジェクトのコピー時に、チェックアウトしたプロジェクトを更新しないよう指定します。これはデフォルト設定です。

`-ntl|-no_translate`

ワークエリア内の ASCII ファイルが、復帰改行の変換を行わずに Windows と UNIX 間でコピーされるよう指定します。デフォルトは `-tl|-translate` です。

`-nrel|-not_relative`

ワークエリアを絶対パス上に置くよう指定します。デフォルトで、新しいプロジェクトはチェックアウトするプロジェクトと同じ相対設定を使用します。

`-nwa|-no_wa`

プロジェクトが管理ワークエリアを持たないよう指定します。デフォルトは、`-wa|-maintain_wa` です。

`-nwat|-no_wa_time`

ファイルがワークエリアにコピーされた時刻ではなく、Rational Synergy に格納されている最後の変更時刻を反映するように、プロジェクト ワークエリア内にあるファイルのタイムスタンプを設定します。これはデフォルト設定です。

`-platform` *`platform`*

新しくチェックアウトするプロジェクトに使用するプラットフォームを指定します。プラットフォームは、有効なプラットフォームの名前である必要があります。プラットフォームの選択肢は、Rational Synergy のインストール エリアにある `CCM_HOME¥etc¥om_hosts.cfg file` ファイル（Windows）または `$CCM_HOME/etc/om_hosts.cfg` ファイル（UNIX）に定義されています。オプションを指定しない場合、デフォルトでチェックアウトするプロジェクトと同じプラットフォーム値が使用されます。

`project_spec`

コピーするプロジェクトを指定します。詳細については、プロジェクトの指定を参照してください。

`-purpose` *`purpose`*

新しくコピーするプロジェクトの目的を指定します。目的は、有効な定義済み目的の名前で、プロジェクトのリリースに有効である必要があります。詳細については、project_purpose コマンドを参照してください。

このオプションを指定せず、*developer* ロールを持っている場合、デフォルトは Insulated Development（個別開発）です。このオプションを指定せず、*build_mgr* または *ccm_admin* ロールを持っている場合、デフォルトは `Integration Testing`（統合テスト）です。

`-rel|-relative`

ワークエリア パスを親プロジェクトのパスに対する相対パスにします。デフォルトでは、新しいプロジェクトはチェックアウトするプロジェクトと同じ相対設定を使用します。

`-release (`*`release_spec`*`|as_is)`

新しくコピーするプロジェクトのリリース値を指定します。キーワード “`as_is`” を指定するか、オプションを指定しない場合、デフォルトでチェックアウトするプロジェクトのリリース値が使用されます。*`release_spec`* には、現在のデータベースで定義したリリース値を指定できます。

`-set|-path|-setpath` *`absolute_path`*

コピーするオブジェクトに使用するワークエリア パスを指定します。指定しない場合、デフォルトのワークエリア パスは現在のワークエリア パス テンプレートとプロジェクト サブディレクトリ テンプレートを使用して決定されます。

`-subprojects`

指定したプロジェクト階層内のすべてのサブプロジェクトをコピーするよう指定します。

`-t|-to` *`version|file_spec`*

チェックアウトするプロジェクトのバージョンを指定します。`-t|-to` または `-versions` を指定しない場合、デフォルトの次のバージョンは Rational Synergy に組み込まれたアルゴリズムを使用して自動的に計算されます。

`-tl|-translate|-translation`

ワークエリア内の ASCII ファイルが、復帰改行の変換を行って Windows と UNIX 間でコピーされるよう指定します。これはデフォルト設定です。

`-u|-update`

プロジェクトのコピー時に、チェックアウトしたプロジェクトを更新するよう指定します。指定した場合、プロジェクトはワークエリアなしでチェックアウトされ、ベースラインとタスクを更新するかどうかを指示するプロジェクト グルーピングの設定に従って更新されます。プロジェクトが管理ワークエリアを持つ場合、プロジェクトは同期されます。デフォルトは `-no_u|-no_update` です。

`-versions old_version:new_version,old_version:new_version,...`

プロジェクトまたはプロジェクト階層のコピーに使用する新しいバージョンを指定します。各マッピングは、現在その値を持つ階層内のすべてのプロジェクトに適用されます。`new_version` が `NoCheckOut` の場合、`old_version` に関連付けられているプロジェクトはコピーされません。

`-t|-to` と `-versions` のどちらも指定しない場合、デフォルトの次のバージョンは Rational Synergy に組み込まれたアルゴリズムを使用して計算されます。

`-wa|-maintain_wa`

プロジェクトが管理ワークエリアを持つよう指定します。これはデフォルト設定です。

`-wat|-wa_time`

プロジェクトのワークエリア内のファイルで、Rational Synergy での最新の修正時刻ではなく、ファイルがワークエリアにコピーされた時刻を反映したタイムスタンプを使用するよう指定します。デフォルトは、`-nwat|-no_wa_time` です。

例

- 既存のプロジェクト階層から新しい開発プロジェクト階層をチェックアウトする。プロジェクトのすべてのバージョンをユーザーの名前に設定します。

  ```
  ccm co -p toolkit-int -subprojects -to john
  ```

- システム テストのため、新しいビルド管理プロジェクト階層をチェックアウトする。リリース値、プラットフォーム値、バージョンを設定します。

  Windows :
  ```
  ccm co -p tool_top-1.0 -subprojects -release 2.0 -platform win32 -
  purpose "System Testing" -versions
  "1.0:sqa,win16_1.0:win16_sqa,win32_1.0:win32_sqa"
  ```

  UNIX :
  ```
  ccm co -p tool_top-1.0 -subprojects -release 2.0 -platform SunOS -
  purpose "System Testing" -versions
  "1.0:sqa,win16_1.0:win16_sqa,win32_1.0:win32_sqa"
  ```

- 最上位のプロジェクトのバージョンを修正し、その変更をサブプロジェクトのバージョンに反映させる。

  ```
  ccm co -p top_project_spec -subprojects -to version
  ```

関連トピック

- checkin コマンド
- copy_project コマンド

説明と用途

非共有プロジェクトのオブジェクトをチェックアウトするとき、そのデフォルトの状態は *working*（作業中）です。共有プロジェクトのファイルまたはディレクトリをチェックアウトするとき、そのデフォルトの状態は、それが非製品の場合は *visible*（可視）、それが製品の場合は shared（共有）です。

オブジェクトをチェックアウトするとき、オブジェクトの書き込み可能バージョンがディレクトリに置かれます（オブジェクトを確認するには、`ccm dir` または `ccm ls` コマンドを使用します）。ディレクトリをチェックアウトしても、ファイル システムには目に見える変更はありません。チェックアウト時に `-t` オプションを使用して新しいバージョンを指定すると、バージョンの指定と新しいオブジェクトの名前変更が可能になります。UNIX では、シンボリック リンクをチェックアウトすることにより、シンボリック リンクがポイントする位置を変更できます。

チェックアウトするオブジェクトは、コンテキスト オブジェクトと親ディレクトリを提供する形で指定する必要があります。

- ワークエリア参照形式

  指定したパスは、プロジェクトの管理ワークエリアにある必要があります。

- プロジェクト参照形式

  たとえば、`sub_proj¥foo.c@my_proj-1`（Windows）または `sub_proj/foo.c@my_proj-1`（UNIX）。

プロジェクトが管理ワークエリアを持たないときでもプロジェクト参照形式を使用できます。

選択セットの参照形式（例 "`@1`"）やオブジェクト名形式（例 `foo.c-1:csrc:1`）など、コンテキスト プロジェクトを与えない ***`file_spec`*** を使用してオブジェクトをチェックアウトすることはできません。

checkpoint コマンド

詳細については、説明と用途を参照してください。`checkpoint` コマンドは、以下のサブコマンドをサポートします。

- プロジェクトのチェックポイント処理
- オブジェクトのチェックポイント処理

プロジェクトのチェックポイント処理

このサブコマンドにより、プロジェクトの個人バージョンをユーザー専用として保存します。このとき、その個人バージョンは修正不可状態に保護され、必要がなくなったときには後で削除できます。checkpoint を実行するには、プロジェクトを所有している必要があります。作業中のプロジェクトのみチェックポイント処理できます。

```
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`

すべてのベースライン プロジェクトとそのメンバーを *released* 状態にチェックインするとき、それらに追加するコメントを指定します。*comment* には、複数行を含むことができ、円記号でコード化した値を使用できます。

このオプションは、-commentedit および -commentfile と一緒に使用できます。-commentedit オプションを使用した場合、コメントはデフォルトのテキスト エディタで表示されます。

`-ce|-commentedit`

コメントの作成と編集用にデフォルトのテキスト エディタを起動するよう指定します。テキスト エディタで保存した結果が最終的なコメントとして使用されます。このオプションは、-comment および -commentfile オプションと一緒に使用できます。

`-cf|-commentfile file_path`

指定したファイルの内容をコメントとして使用するよう指定します。-comment を指定した場合、それがコメントに追加されます。このオプションは、-commentedit オプションと一緒に使用できます。

`-cr|-commentreplace`

通常、指定されたコメントは既存のコメントに追加されます。ただし、-cr オプションを使用すると、既存のコメントが新しいコメントに置き換わります。

`-p|-project project_spec`

プロジェクトをチェックポイント処理します。詳細については、プロジェクトの指定を参照してください。

`-t|-to` *version*

新しくチェックアウトするオブジェクトのバージョンを設定します。この操作は、オブジェクト名にバージョンを追加することによっても実行できます。

関連トピック

- オブジェクトのチェックポイント処理
- プロジェクトのチェックアウト

オブジェクトのチェックポイント処理

このサブコマンドにより、オブジェクトの個人バージョンをユーザー専用として保存します。このとき、その個人バージョンは修正不可状態に保護され、必要がなくなったときには後で削除できます。checkpoint を実行するには、オブジェクトを所有している必要があります。作業中のオブジェクトのみチェックポイント処理できます。

```
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`

すべてのベースライン プロジェクトとそのメンバーを *released* 状態にチェックインするとき、それらに追加するコメントを指定します。*comment* には、複数行を含むことができ、円記号でコード化した値を使用できます。

このオプションは、-commentedit および -commentfile と一緒に使用できます。-commentedit オプションを使用した場合、コメントはデフォルトのテキスト エディタで表示されます。

`-ce|-commentedit`

コメントの作成と編集用にデフォルトのテキスト エディタを起動するよう指定します。テキスト エディタで保存した結果が最終的なコメントとして使用されます。このオプションは、-comment および -commentfile オプションと一緒に使用できます。

`-cf|-commentfile file_path`

指定したファイルの内容をコメントとして使用するよう指定します。-comment を指定した場合、それがコメントに追加されます。このオプションは、-commentedit オプションと一緒に使用できます。

`-cr|-commentreplace`

通常、指定されたコメントは既存のコメントに追加されます。ただし、-cr オプションを使用すると、既存のコメントが新しいコメントに置き換わります。

file_spec

チェックポイント処理するファイル、ディレクトリ、またはプロジェクトを指定します。詳細については、ファイルの指定を参照してください。

`-p|-project`

プロジェクトの履歴を表示します。

`project_spec`

一覧表示するプロジェクトを指定します。詳細については、プロジェクトの指定を参照してください。

`-t|-to version|`*`file_spec`*

バージョンを指定し、新しい非プロジェクト オブジェクトのバージョンを変更するか、あるいは新しいプロジェクトまたはプロジェクト階層のバージョンを指定します。

`-task` *`task_spec`*

新しくチェックアウトするオブジェクトに関連付けるタスクを指定します。詳細については、タスクの指定を参照してください。

タスクの指定を行わないが、カレント タスクが設定されている場合は、新しく作成されるバージョンはカレント タスクに関連付けられます。チェックポイント オブジェクト バージョンに関連付けられるタスクには変更はありません。

例

- `foo.c` の現在の *working*（作業中）バージョンをチェックポイント処理し、コメントを追加する。

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

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

- `foo.c` の現在の作業中バージョンをチェックポイント処理する。コメントを追加し、新しい *working*（作業中）オブジェクトのバージョンを *joe* にする。

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

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

関連トピック

- オブジェクトのチェックアウト
- プロジェクトのチェックポイント処理

説明と用途

`checkpoint` コマンドにより、オブジェクトの個人バージョンをユーザー専用として保存します。このとき、その個人バージョンは修正不可状態に保護され、必要がなくなったときには後で削除できます。`checkpoint` を実行するには、オブジェクトを所有している必要があります。作業中のオブジェクトのみチェックポイント処理できます。

チェックポイント処理を実行すると、オブジェクトの現在のバージョンは *checkpoint*（チェックポイント）状態に移行し、オブジェクトの新しいバージョンが作成されます。`checkpoint` コマンドで指定するすべてのコメントは、チェックポイント処理されたオブジェクトに適用されます。

cmdhistory コマンド

詳細については、説明と用途を参照してください。`cmdhistory` コマンドは、以下のサブコマンドをサポートします。

- 履歴項目のクリア
- コマンド履歴の表示
- 記録するコマンドの最大数の設定

履歴項目のクリア

```
ccm cmdhistory -clear
```

`-clear`

現在のセッションで実行したすべてのコマンドの履歴をクリアします。

関連トピック

- 記録するコマンドの最大数の設定
- コマンド履歴の表示
- オブジェクトの履歴の表示

コマンド履歴の表示

```
ccm cmdhistory -s|-sh|-show [count]
```

`-s|-sh|-show`

`cmdhistory` コマンドを除いて、現在のセッションで実行したコマンドを指定した最大数まで表示します。

`count`

この引数を指定すると、セッション中に入力したコマンドが指定数分表示されます。たとえば、count 10 を要求し、最大コマンド履歴を 50 に設定し、そのセッション中に 200 個のコマンドを実行した場合、最後の 10 個のコマンドが表示されます。そのセッション中に 5 個のコマンドしか実行していない場合、その 5 個のコマンドが表示されます。

例

- このセッション中に実行した最後の 3 個のコマンドを表示する。

```
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
```

関連トピック

- 履歴項目のクリア
- 記録するコマンドの最大数の設定
- オブジェクトの履歴の表示

記録するコマンドの最大数の設定

`ccm cmdhistory -set` *`maximum`*

`-set`

履歴に保存するコマンドの数を指定します。

保存するコマンドのデフォルトの最大値は 100 です。

`maximum`

保存するコマンドの最大値を指定した数に変更します。

例

- コマンド履歴を最大 60 コマンド記録するように設定する。

  ```
  ccm cmdhistory -set 60
  ```

関連トピック

- 履歴項目のクリア
- コマンド履歴の表示
- オブジェクトの履歴の表示

説明と用途

`cmdhistory` により、セッション中に実行したコマンドの記録を取得します。このコマンドの用途として、以下のような例が挙げられます。

- SQE がアドホック テストを行った後で、使用した一連のコマンドを新規のテスト用スクリプトに取り込む。
- IBM Rational ソフトウェア サポートは、顧客の CLI コマンド履歴を取得して問題の調査と分析に役立てる。

conflicts コマンド

詳細については、説明と用途を参照してください。`conflicts` コマンドは、以下のサブコマンドをサポートします。

- プロジェクトのオブジェクト コンフリクトの表示
- プロジェクトのタスク コンフリクトの表示

プロジェクトのオブジェクト コンフリクトの表示

このサブコマンドにより、プロジェクトのオブジェクト コンフリクトを表示します。

```
ccm conflicts [-r|-recurse] [-f|-format format] [-nf|-noformat]
            ([-ch|-column_header] | [-nch|-nocolumn_header])
            [-sep|-separator separator] ([-sby|-sortby sortspec] |
            [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat] project_spec
```

`-ch|-column_header`

出力フォーマットでカラム ヘッダーを使用するよう指定します。これはデフォルト設定です。詳細については、-ch|-column_headers を参照してください。

`-f|-format format`

コマンドの出力フォーマットを指定します。詳細については、-f|-format を参照してください。

`-gby|-groupby groupformat`

コマンドの出力をグループ化する方法を指定します。詳細については、-gby|-groupby を参照してください。

`-nch|-nocolumn_header`

出力フォーマットでカラム ヘッダーを使用しないよう指定します。詳細については、-nch|-nocolumn_headers を参照してください。

`-nf|-noformat`

カラム配置を使用しないよう指定します。詳細については、-nf|-noformat を参照してください。

`-ns|-no_sort`

コマンドの出力をソートしないよう指定します。詳細については、-ns|-nosort を参照してください。

`project_spec`

メンバーシップ コンフリクトを分析するプロジェクトを指定します。*`project_spec`* は、1 つのプロジェクトにのみ設定できます。詳細については、プロジェクトの指定を参照してください。

`-r|-recurse`

指定されたプロジェクトが最上位に位置するプロジェクト階層内の、すべてのプロジェクトのメンバーシップ コンフリクトを表示するよう指定します。

`-sep|-separator` *`separator`*

-f|-format オプションと一緒にのみ使用します。異なる区切り文字を指定します。詳細については、-sep|-separator を参照してください。

`-sby|-sortby` *`sortspec`*

コマンドの出力をソートする方法を指定します。詳細については、-sby|-sortby を参照してください。

関連トピック

- グローバル フォーマット オプション
- フォーマットの使用例

プロジェクトのタスク コンフリクトの表示

このサブコマンドにより、プロジェクトのタスク コンフリクトを表示します。

```
ccm conflicts -t|-tasks [-r|-recurse] [-f|-format format] [-nf|-noformat]
            ([-ch|-column_header] | [-nch|-nocolumn_header])
            [-sep|-separator separator] ([-sby|-sortby sortspec] |
            [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat] project_spec
```

`-ch|-column_header`

出力フォーマットでカラム ヘッダーを使用するよう指定します。これはデフォルト設定です。詳細については、-ch|-column_headers を参照してください。

`-f|-format format`

コマンドの出力フォーマットを指定します。詳細については、-f|-format を参照してください。

`-gby|-groupby groupformat`

コマンドの出力をグループ化する方法を指定します。詳細については、-gby|-groupby を参照してください。

`-nch|-nocolumn_header`

出力フォーマットでカラム ヘッダーを使用しないよう指定します。詳細については、-nch|-nocolumn_headers を参照してください。

`-nf|-noformat`

カラム配置を使用しないよう指定します。詳細については、-nf|-noformat を参照してください。

`-ns|-no_sort`

コマンドの出力をソートしないよう指定します。詳細については、-ns|-nosort を参照してください。

`project_spec`

メンバーシップ コンフリクトを分析するプロジェクトを指定します。*`project_spec`* は、1 つのプロジェクトにのみ設定できます。詳細については、プロジェクトの指定を参照してください。

`-r|-recurse`

指定されたプロジェクトが最上位に位置するプロジェクト階層内の、すべてのプロジェクトのメンバーシップ コンフリクトを表示するよう指定します。

`-sep|-separator` *`separator`*

-f|-format オプションと一緒にのみ使用します。異なる区切り文字を指定します。詳細については、-sep|-separator を参照してください。

`-sby|-sortby` *`sortspec`*

コマンドの出力をソートする方法を指定します。詳細については、-sby|-sortby を参照してください。

例

- `ProjectTwo-newVer` プロジェクトに関するコンフリクト検出情報を表示する。

```
ccm conflicts ProjectTwo-newVer
```

```
Project:ProjectTwo-newVer
Objectname              Task Conflicts                    Category
NewFile.txt-one:ascii:1 9    Implicitly included         Extra Changes
File1.txt-ab:ascii:1    9    Included by 'use' operation?Extra Changes
File2.txt-cd:ascii:1    9    Included by 'use' operation?Extra Changes
File2.txt-ef:ascii:1    9    Implicitly included         Extra Changes
ProjectTwo-2:dir:1      9    Included by 'use' operation?Extra Changes
```

関連トピック

- グローバル フォーマット オプション
- フォーマットの使用例

説明と用途

`conflicts` コマンドは、更新プロパティがタスクとベースラインを使用するように設定されているプロジェクトのコンフリクトを表示します。

コンフリクトは、プロジェクトの更新プロパティに関連する変更と、プロジェクトのメンバーシップに含まれる変更との間に生じた不整合です。

コンフリクトとその確認方法の詳細については、Rational Synergy CLI ヘルプ、トラディショナル モードの「コンフリクト検出」を参照してください。

copy_project コマンド

詳細については、説明と用途を参照してください。`copy_project` コマンドは プロジェクトのコピー サブコマンドをサポートします。

プロジェクトのコピー

このサブコマンドにより、プロジェクトまたはプロジェクト階層の修正可能バージョンを作成します。デフォルトでは、プロジェクトをコピーすると、そのコピーがデータベースに作成され、ワークエリアが自動的に作成されます。プロジェクトをコピーするとき、ワークエリア プロパティを設定できます。

```
ccm copy_project|cp [-purpose purpose] [-platform platform]
    [-release (release_spec|as_is)] [-subprojects] ([-t|-to version] |
    [(-versions old_version:new_version,old_version:new_version...)...])
    ([-u|-update] | [-no_u|-no_update]) ([-cb|-copy_based] |
    [-lb|-link_based|-ncb|-not_copy_based])
    ([-rel|-relative] | [-nrel|-not_relative])
    [-set|-path|-setpath absolute_path] ([-mod|-modifiable] |
    [-nmod|-not_modifiable]) ([-tl|-translate|-translation] |
    [-ntl|-no_translate|-no_translation]) ([-wa|-maintain_wa] |
    [-nwa|-no_wa]) ([-wat|-wa_time] | [-nwat|-no_wa_time])
    [-c|-comment comment_string] [-ce|-commentedit]
    [-cf|-commentfile file_path] project_spec...
```

`-c|-comment comment`

すべてのベースライン プロジェクトとそのメンバーを *released* 状態にチェックインするとき、それらに追加するコメントを指定します。*comment* には、複数行を含むことができ、円記号でコード化した値を使用できます。

このオプションは、`-commentedit` および `-commentfile` と一緒に使用できます。`-commentedit` オプションを使用した場合、コメントはデフォルトのテキスト エディタで表示されます。

`-ce|-commentedit`

コメントの作成と編集用にデフォルトのテキスト エディタを起動するよう指定します。テキスト エディタで保存した結果が最終的なコメントとして使用されます。このオプションは、`-comment` および `-commentfile` オプションと一緒に使用できます。

`-cf|-commentfile file_path`

指定したファイルの内容をコメントとして使用するよう指定します。`-comment` を指定した場合、それがコメントに追加されます。このオプションは、`-commentedit` オプションと一緒に使用できます。

`-cb|-copy_based`

ワークエリアをコピーベースに指定します。

`-lb|-link_based|-ncb|-not_copy_based`

ワークエリアをリンク ベースに指定します。このオプションは、UNIX ユーザーのみが使用できます。

詳細については、work_area コマンドを参照してください。

`-mod|-modifiable_wa`

ワークエリア内のファイルに、チェックアウトされていない場合でも修正可能となるようにアクセス許可を設定します。デフォルトは `-nmod|-not_modifiable_wa` です。

`-nmod|-not_modifiable_wa`

working などの書き込み可能状態の場合にのみ、デフォルトでワークエリア内のファイルが修正可能となるようにアクセス許可を設定します。これはデフォルト設定です。

`-no_u|-no_update`

プロジェクトのコピー時に、チェックアウトしたプロジェクトを更新しないよう指定します。これはデフォルト設定です。

`-ntl|-no_translate|-no_translation`

ワークエリア内の ASCII ファイルが、復帰改行の変換を行わずに Windows と UNIX 間でコピーされるよう指定します。デフォルトは `-tl|-translate` です。

`-nrel|-not_relative`

ワークエリアを絶対パス上に置くよう指定します。デフォルトでは、新しいプロジェクトはチェックアウトするプロジェクトと同じ相対設定を使用します。

`-nwa|-no_wa`

プロジェクトが管理ワークエリアを持たないよう指定します。デフォルトは `-wa|-maintain_wa` です。

`-nwat|-no_wa_time`

ファイルがワークエリアにコピーされた時刻ではなく、Rational Synergy に格納されている最後の変更時刻を反映するように、プロジェクト ワークエリア内にあるファイルのタイムスタンプを設定します。これはデフォルト設定です。

`-platform` *`platform`*

新しくチェックアウトするプロジェクトに使用するプラットフォームを指定します。*platform* は、有効なプラットフォームの名前である必要があります。プラットフォームの選択肢は、Rational Synergy のインストール エリアにある `CCM_HOME¥etc¥om_hosts.cfg` ファイル（Windows）または `$CCM_HOME/etc/om_hosts.cfg` ファイル（UNIX）に定義されています。オプションを指定しない場合、デフォルトでチェックアウトするプロジェクトと同じプラットフォーム値が使用されます。

`project_spec`

コピーするプロジェクトを指定します。詳細については、プロジェクトの指定を参照してください。

`-purpose` *`purpose`*

新しくコピーするプロジェクトの目的を指定します。*purpose* は、有効な定義済み目的の名前で、プロジェクトのリリースに有効である必要があります。詳細については、project_purpose コマンドを参照してください。

このオプションを指定せず、developer ロールを持っている場合、デフォルトは Insulated Development （個別開発）です。このオプションを指定せず、*build_mgr* または *ccm_admin* ロールを持っている場合、デフォルトは `Integration Testing`（統合テスト）です。

`-rel|-relative`

ワークエリア パスを親プロジェクトのパスに対する相対パスにします。デフォルトでは、新しいプロジェクトはチェックアウトするプロジェクトと同じ相対設定を使用します。

`-release` *`release_spec`*

新しくコピーするプロジェクトのリリース値を指定します。キーワード “`as_is`” を指定するか、オプションを指定しない場合、デフォルトでチェックアウトするプロジェクトのリリース値が使用されます。*`release_spec`* には、現在のデータベースで定義したリリース値を指定できます。詳細については、リリースの指定を参照してください。

手動更新プロパティはサポートされておらず、プロジェクト グルーピングと対応するプロセス ルールは常にリリースに関連付ける必要があるため、Rational Synergy プロジェクトはリリース値を持つ必要があります。

`-set|-path|-setpath` *`absolute_path`*

コピーするオブジェクトに使用するワークエリア パスを指定します。指定しない場合は、現在の wa_path_template と project_subdir_template を使用してデフォルトのワークエリア パスが決定されます。

`-subprojects`

指定したプロジェクト階層内のすべてのサブプロジェクトをコピーするよう指定します。

`-tl|-translate|-translation`

プロジェクトのワークエリア内で、Windows と UNIX 間で ASCII ファイルをコピーするとき、ASCII ファイルを変換するよう指定します。

`-t|-to` *`version`*

チェックアウトするプロジェクトのバージョンを指定します。`-t|-to` と `-versions` のどちらも指定しない場合、デフォルトの次のバージョンは自動的に計算されます。

`-u|-update`

プロジェクトのコピー時に、チェックアウトしたプロジェクトを更新するよう指定します。指定した場合、プロジェクトはワークエリアなしでチェックアウトされ、ベースラインとタスクを更新するかどうかを指示するプロジェクト グルーピングの設定に従って更新されます。プロジェクトが管理ワークエリアを持つ場合、プロジェクトは同期されます。デフォルトは `-no_u|-no_update` です。

`-versions "`*`old_ver`*`:`*`new_ver`*`,`*`old_ver`*`:`*`new_ver`*`,..."`

プロジェクトまたはプロジェクト階層のコピーに使用する新しいバージョンを指定します。各マッピングは、現在その値を持つ階層内のすべてのプロジェクトに適用されます。`new_version` が `NoCheckOut` の場合、`old_version` に関連付けられているプロジェクトはコピーされません。

`-t|-to` と `-versions` のどちらも指定しない場合、デフォルトの次のバージョンは自動的に計算されます。

`-wa|-maintain_wa`

プロジェクトが管理ワークエリアを持つよう指定します。これはデフォルト設定です。

`-wat|-wa_time`

プロジェクトのワークエリア内のファイルで、Rational Synergy での最新の修正時刻ではなく、ファイルがワークエリアにコピーされた時刻を反映したタイムスタンプを使用するよう指定します。デフォルトは `-nwat|-no_wa_time` です。

例

- `projA-3` プロジェクトの新しいバージョンをコピーする。

  ```
  ccm copy_project -c "test projA" projA-3
  ```

- 既存のプロジェクト階層から新しい開発プロジェクト階層をコピーする。プロジェクトのすべてのバージョンをユーザーの名前に設定します。

  ```
  ccm copy_project toolkit-int -subprojects -to john
  ```

- システム テストのため、新しいビルド管理プロジェクト階層をコピーする。リリース値、プラットフォーム値、バージョンを設定します。

  ```
  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"
  ```

- 最上位のプロジェクトのバージョンを修正し、その変更をサブプロジェクトのバージョンに反映させる。

  ```
  ccm copy_project top_project_spec -subprojects -to version
  ```

関連トピック

- プロジェクトのチェックイン
- プロジェクトのチェックアウト

説明と用途

`copy_project` コマンドは `-project` オプションの付いた `checkout` コマンドと機能は同じです。旧リリースでは、`copy_project` 操作は `checkout -project` 操作と呼ばれていました。

静的（修正不可状態の）プロジェクトからプロジェクトをコピーして、サブプロジェクトをコピーしなかった場合、そのサブプロジェクトが相対ワークエリアを持っていれば、コピーされるプロジェクトのワークエリア内の適切な位置にこれらのサブプロジェクトのワークエリアの新しいコピーが作成されます。開発者は、この方法で、相対ワークエリアを持つ静的サブプロジェクトを再利用できます。

静的ワークエリアは管理されず、データベースとのリコンサイルはできません。リコンサイル時には静的ワークエリアは無視されます。静的ワークエリアの同期をとると、変更されたすべてのファイルがデータベースからのファイルに置き換わります。静的ワークエリアを持つプロジェクトをコピーすると、元のワークエリアはそのまま残ります。元のワークエリアをリコンサイルして、変更を放棄するかまたは維持する必要があります。

copy_to_file_system コマンド

詳細については、説明と用途を参照してください。`copy_to_file_system` コマンドはファイル システムへのプロジェクトのコピー サブコマンドをサポートします。

ファイル システムへのプロジェクトのコピー

`copy_to_file_system` コマンドにより、書き込み禁止のプロジェクトのコピーをワークエリア内に作成します。

作成後は、そのプロジェクトの維持およびリコンサイルはできません。

```
ccm cfs|copy_to_file_system|wa_snapshot [-p|-path path] [-r|-recurse]
        project_spec...
```

`-p|-path path`

コピーされるプロジェクトの書き込み先のパスを指定します。デフォルトのパスは展開したデフォルト ワークエリア パスです（Windows では `ccm_wa¥database_name`、UNIX ではホーム ディレクトリの `ccm_wa/database_name`）。

> **注記**：パスを指定しないと、パスは展開したデフォルト ワークエリア パスに設定されます。また、このパスは空で、ディレクトリにファイルが含まれていてはなりません。

`project_spec`

コピーされるプロジェクトを指定します。詳細については、プロジェクトの指定を参照してください。

`-r|-recurse`

選択したプロジェクト（Windows では `ccm_wa¥database_name`、UNIX ではホーム ディレクトリの `ccm_wa/database_name`）だけでなく、サブプロジェクトにもコピーしたプロジェクトを作成します。

> **注記**：このオプションは、指定したプロジェクトとすべてのサブプロジェクトのワークエリア コピーを作成します。このオプションを指定しなかった場合、サブプロジェクトは無視されます。

例

- プロジェクト リスト `proj1-1 proj2-2` についてコピーしたプロジェクトをワークエリアに作成する。

  ```
  ccm copy_to_file_system -path C:¥ccm_wa¥ccm_docs proj1-1 proj2-1
  ```

説明と用途

ワークエリアにコピーされるプロジェクトは以下の特性を持ちます。

- リンクベースではなく、常にコピーベース
- ファイルは読み取り専用
- ファイルの変更時刻がコピーの作成時刻に設定される
- ワークエリアを持たないプロジェクトで作成可能

create コマンド

詳細については、説明と用途を参照してください。`create` コマンドは、以下のサブコマンドをサポートします。

- 最上位プロジェクトの作成
- オブジェクトの作成

最上位プロジェクトの作成

このサブコマンドにより、新しい最上位プロジェクトを作成します。既存 Rational Synergy プロジェクトのサブプロジェクトとするには、プロジェクトを作成した後で `ccm use -p` コマンドを使用します。プロジェクトを作成すると、Rational Synergy はそのワークエリアを自動的に作成します。デフォルトでは、ワークエリアはデフォルトワークエリア パス テンプレートを展開して作成されます。デフォルト設定では、Windows では `%HOMEPATH%¥My Documents¥Synergy¥ccm_wa¥databaseName¥projectName-projectVersion` 、UNIX では `$HOME/ccm_wa/databaseName/projectName-projectVersion` となります。

```
ccm create -t|-type project [-platf|-platform platform]
        [-purp|-purpose purpose] [-rel|-release release_spec]
        ([-cb|-copy_based] | [-lb|-link_based|-ncb|-not_copy_based])
        [-set|-path|-setpath absolute_path]
        [-wa|-maintain_wa] [-nwa|-no_wa]
        [-task task_spec] [-c|-comment comment_string] [-ce|-commentedit]
        [-cf|-commentfile file_path] new_project_spec...
```

`-c|-comment` *`comment`*

すべてのベースライン プロジェクトとそのメンバーを *released* 状態にチェックインするとき、それらに追加するコメントを指定します。*`comment`* には、複数行を含むことができ、円記号でコード化した値を使用できます。

このオプションは、`-commentedit` および `-commentfile` と一緒に使用できます。`-commentedit` オプションを使用した場合、コメントはデフォルトのテキスト エディタで表示されます。

`-ce|-commentedit`

コメントの作成と編集用にデフォルトのテキスト エディタを起動するよう指定します。テキスト エディタで保存した結果が最終的なコメントとして使用されます。このオプションは、`-comment` および `-commentfile` オプションと一緒に使用できます。

`-cf|-commentfile` *`file_path`*

指定したファイルの内容をコメントとして使用するよう指定します。`-comment` を指定した場合、それがコメントに追加されます。このオプションは、`-commentedit` オプションと一緒に使用できます。

`-cf|-commentfile` *`file_path`*

指定したファイルの内容をコメントとして使用するよう指定します。`-comment` を指定した場合、それがコメントに追加されます。このオプションは、`-commentedit` オプションと一緒に使用できます。

`-cb|-copy_based`

ワークエリアをコピーベースに指定します。

`-lb|-link_based|-ncb|-not_copy_based`

ワークエリアをリンク ベースに指定します。このオプションは、UNIX ユーザーのみが使用できます。

詳細については、work_area コマンドを参照してください。

`new_project_spec`

作成するプロジェクトの名前とバージョン（オプション）を指定します。*`new_project_spec`* は、以下のいずれかの形式をとる必要があります。

- 名前
- 名前、コロン、バージョン
- 上記のいずれかを含む ファイル内容形式

`new_project_spec` は一般的なプロジェクト指定ではありません。オブジェクト名形式やクエリ選択セット参照形式などの形式は使用できません。

`-nwa|-no_wa`

新しいプロジェクトが管理ワークエリアを持たないよう指定します。後でプロジェクトに管理ワークエリアを持たせる場合は、ワークエリア コマンドを使用します。デフォルトでは、管理ワークエリアを持つプロジェクトが作成されます。

`-platf|-platform` *`platform`*

新しいプロジェクトのプラットフォームを指定します。プラットフォームは、有効なプラットフォーム名である必要があります。

`-purpose` *`purpose`*

新しいプロジェクトの目的を指定します。目的は、指定したリリースに有効な定義済み目的の名前である必要があります。有効な目的を一覧表示するには、`project_purpose -show` コマンドを使用します。

`-release` *`release_spec`*

新しいプロジェクトに使用するリリースを指定します。*`release_spec`* には、定義済みでアクティブな 1 つのリリースを指定できます。詳細については、リリースの指定を参照してください。

`-set|-path|-setpath` *`absolute_path`*

プロジェクトに使用するワークエリア パスを指定します。*`absolute_path`* は、可視で修正可能な絶対パスである必要があります。

`-task` *`task_spec`*

新しいプロジェクトのルート ディレクトリを関連付けるタスクを指定します。*`task_spec`* には、1 つのタスクを指定できます。デフォルトでは、プロジェクトのルート ディレクトリは現在のタスクと関連付けられています。詳細については、タスクの指定を参照してください。

`-wa|-maintain_wa`

新しいプロジェクトが管理ワークエリアを持つよう指定します。これは、`-wa|-maintain_wa` と `-nwa|-no_wa` のどちらも指定しない場合のデフォルトです。ワークエリアは、新しいプロジェクトに対する変更によって更新されます。ワークエリアの管理をオフにするには、`work area` コマンドを使用します。

例

- proj1 という名前の初期プロジェクトをワークエリアに作成する。

  ```
  ccm create -t project proj1
  ```

- 初期プロジェクトを作成し、ワークエリアを管理する。

  ```
  ccm create -t project -c "test" -wa -set "/tmp" testwa-1.0
  ```

- -wa を使用して MainPrj-1 および SubPrj-1 を作成する。MainPrj-1 ルート ディレクトリ内の SubPrj-1 を使用します。

  ```
  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）または cd WAPATH/MainPrj-1/MainPrj（Unix）

  ```
  ccm use -p SubPrj-1 -task 13
  ```

- -nwa を使用して MainPrj-1 および SubPrj-1 を作成する。MainPrj-1 ルート ディレクトリ内の SubPrj-1 を使用します。

  ```
  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
  ```

関連トピック

- 現在のディレクトリへのプロジェクトの追加
- データベースからのオブジェクトの削除
- プロジェクトの表示

オブジェクトの作成

このサブコマンドにより、新しいオブジェクトを作成し、そのオブジェクトをコンテキスト プロジェクトとコンテキスト親ディレクトリに関連付けます。コンテキスト プロジェクトとコンテキスト親ディレクトリは、ワークエリア参照形式を使用する場合は、指定されたワークエリア パスと関連したプロジェクトであり、プロジェクト参照形式を使用する場合は、その指定されたプロジェクトです。

非共有プロジェクトのオブジェクトを作成する場合、デフォルトの状態は working（作業中）です。共有プロジェクトのファイルまたはディレクトリを作成する場合、そのデフォルトの状態は、非製品の場合は visible（可視）、製品の場合は shared（共有）です。

書込み禁止ディレクトリ内に新しいオブジェクトを作成する場合は、新しいディレクトリ バージョンが自動的にチェックアウトされます。他のユーザーが新しいオブジェクトを利用できるようにするには、そのディレクトリと新しいオブジェクトをチェックインする必要があります。

共有プロジェクト内にいて、現在のディレクトリが修正不可の場合、そのディレクトリはチェックアウトされ、自動的に現在の（または指定した）タスクと関連付けられ、integrate（統合）状態にチェックインされます。初期設定ファイル内の shared_project_directory_checkin を FALSE に設定して、自動チェックイン機能を無効にできます。shared_project_directory_checkin を参照してください。

```
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`

すべてのベースライン プロジェクトとそのメンバーを *released* 状態にチェックインするとき、それらに追加するコメントを指定します。`comment` には、複数行を含むことができ、円記号でコード化した値を使用できます。

このオプションは、`-commentedit` および `-commentfile` と一緒に使用できます。`-commentedit` オプションを使用した場合、コメントはデフォルトのテキスト エディタで表示されます。

`-ce|-commentedit`

コメントの作成と編集用にデフォルトのテキスト エディタを起動するよう指定します。テキスト エディタで保存した結果が最終的なコメントとして使用されます。このオプションは、`-comment` および `-commentfile` オプションと一緒に使用できます。

`-cf|-commentfile` *`file_path`*

指定したファイルの内容をコメントとして使用するよう指定します。`-comment` を指定した場合、それがコメントに追加されます。このオプションは、`-commentedit` オプションと一緒に使用できます。

`new_file_spec`

新規作成するファイルまたはディレクトリを指定します。*`new_file_spec`* は、以下のいずれかの形式をとる必要があります。

- ワークエリア参照形式。新しいオブジェクトの名前、およびオプションでバージョン区切り文字とバージョンまたはコロンとバージョンで終わる相対パスを持つ。親ディレクトリは、管理ワークエリア内の管理ディレクトリを参照する必要があります。
- プロジェクト参照形式。指定したプロジェクト内にあるディレクトリ下にあり、新しいオブジェクトの名前、およびオプションでバージョン区切り文字とバージョンまたはコロンとバージョンで終わる相対パスを持つ。
- ファイル内容形式。上記のいずれかを含む。

これらの形式により、コンテキスト プロジェクトとコンテキスト親ディレクトリを与えます。オブジェクトは、指定したコンテキスト プロジェクト内の親ディレクトリの下に作成されます。*`new_file_spec`* は一般的な *`file_spec`* ではありません。オブジェクト名形式やクエリ選択セット参照形式などの形式は使用できません。

バージョンを指定しなかった場合、デフォルトのバージョンとして 1 が使用されます。

allow_delimiter_in_name を TRUE に設定し、*`new_file_spec`* が 1 つのバージョン区切り文字を含む場合、その文字列がオブジェクトの名前として使用されます。たとえば、`newfile-2` のデフォルト バージョン名は `newfile-2` となります。この設定では、バージョン `2` を持つ `newfile` というファイルを作成するには、`newfile` の *`new_file_spec`* に `-version 2` を使用します。`allow_delimiter_in_name` を FALSE に設定している場合は、*`new_file_spec`* 内の任意のバージョン区切り文字がバージョン区切り文字として処理されるので、バージョンを指定できます。

`-task` *`task_spec`*

新しいオブジェクトを関連付けるタスクを指定します。新しいオブジェクトを作成するディレクトリが修正不可の場合、ディレクトリが自動的にチェックアウトされてそのタスクに関連付けられます。*`task_spec`* には、1 つのタスクを指定できます。デフォルトでは、新しいオブジェクトと任意の自動的にチェックアウトされたディレクトリが、カレント タスクと関連付けられます。詳細については、タスクの指定を参照してください。

`-t|-type` *`type`*

新しいオブジェクトのタイプを指定します。タイプを指定しなかった場合、デフォルトは拡張子から計算されます（たとえば、`.c` オブジェクトのデフォルトは `csrc` タイプとなります）。

`-v|-version` *`version`*

指定した場合、*`new_file_spec`* で指定したすべてのバージョンに優先します。この方法は、主に `allow_delimiter_in_name` を TRUE に設定した場合に使用します。*`new_file_spec`* が 1 つのバージョン区切り文字を含む場合、その文字列がオブジェクトの名前として使用されます。たとえば、`newfile-2` のデフォルト バージョン名は `newfile-2` となります。この設定では、バージョン `2` を持つ `newfile` というファイルを作成するには、`newfile` の *`new_file_spec`* に `-version 2` を使用します。`allow_delimiter_in_name` を FALSE に設定している場合は、*`new_file_spec`* 内の任意のバージョン区切り文字がバージョン区切り文字として処理されるので、バージョンを指定できます。

例

- Windows 上で、`sort.c` という名前の新しい C ソース オブジェクトを `utils¥sym_tool` ディレクトリに作成する。

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

- UNIX 上で、`sort.c` という名前の新しい C ソース オブジェクトを `utils/sym_tool` ディレクトリに作成する。

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

- `testcase` という名前の新しいディレクトリ オブジェクトを現在のディレクトリの下に作成する。

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

関連トピック

- 現在のディレクトリへのプロジェクトの追加
- データベースからのオブジェクトの削除
- プロジェクトの表示

説明と用途

`create` コマンドにより、以下のように、新しいオブジェクトを作成して現在のプロジェクトに追加します。

- 新しい ファイルまたはディレクトリを作成すると、プロジェクトを構成する現在のディレクトリに追加される。
- 非共有プロジェクトのオブジェクトを作成する場合、デフォルトの状態は *working*（作業中）となる。共有プロジェクトのファイルまたはディレクトリを作成する場合は、デフォルトの状態は、非製品の場合は *visible*（可視）、製品の場合は *shared*（共有）となる。
- 書込み禁止ディレクトリ内に新しいオブジェクトを作成する場合は、新しいディレクトリ バージョンが自動的にチェックアウトされる。

  共有プロジェクト内にいて、現在のディレクトリが変更禁止の場合、ディレクトリがチェックアウトされ、デフォルトの（または指定した）タスクと自動的に関連付けられ、*integrate*（統合）状態にチェックインされる。初期設定ファイル内の `shared_project_directory_checkin` を `FALSE` に設定して、自動チェックイン機能を無効にできます（shared_project_directory_checkin を参照してください）。
- 新しいプロジェクトを作成する場合は、フローティング オブジェクトとして作成される。ただし、`use -p` コマンドを使用すれば、既存のプロジェクト内のサブプロジェクトとして作成できる。
- プロジェクトを作成すると、Rational Synergy はそのワークエリアを自動的に作成する。デフォルトでは、ワークエリアは `My Documents¥Synergy¥ ccm_wa¥` *`database¥project_name-version`*（Windows）、または `ccm_wa/`*`database/project_name-version`*（UNIX）ユーザーのホーム ディレクトリに置かれます（詳細については、ワークエリア プロパティの修正を参照してください）。
- ディレクトリにメンバーを追加するには、そのディレクトリが書き込み可能（チェックアウト可能）である必要がある。修正不可ディレクトリにオブジェクトを作成しようとすると、Rational Synergy はそのディレクトリを自動的にチェックアウトします。他のユーザーが新しいオブジェクトを利用できるようにするには、そのディレクトリと新しいオブジェクトをチェックインする必要があります。

dcm コマンド

詳細については、説明と用途を参照してください。dcm コマンドは、以下のサブコマンドをサポートします。

- 転送セットへのオブジェクトの追加
- データベース定義の作成
- 転送セットの作成
- データベース定義の削除
- 転送セットの削除
- パッケージの削除
- 転送パッケージの生成
- データベースへの最新のマーク付け
- データベース定義の修正
- 転送セットの修正
- DCM 設定の修正
- パッケージの受取り
- 転送セットの間接変更依頼メンバーの再計算
- 転送セットの間接リリース メンバーの再計算
- 転送セットのメンバーの再計算
- 転送セットからのオブジェクトの削除
- データベース定義情報の表示
- DCM プロパティの表示
- データベース ID の表示
- DCM イベントの概要の表示
- 設定の表示
- 受取りロックの表示
- DCM イベントの概要の表示
- 指定 DCM イベントの表示
- 最新生成時刻の表示
- 現在の DCM データベース ID の表示
- 転送セット情報の表示
- 転送セット メンバーの表示

- 転送セットの表示
- 生成パッケージの転送

前提条件

現在のデーベースが、DCM 用に初期化されている必要があります。

転送セットへのオブジェクトの追加

このサブコマンドにより、指定したオブジェクトを指定した転送セットに追加します。プロジェクト、ディレクトリ、およびファイルを、履歴つきにまたは履歴なしで追加できます。プロジェクト、ディレクトリ、またはファイルを履歴つきで追加した場合、そのオブジェクトのすべてのバージョンが転送セットに追加されます。オブジェクトを転送セットに追加すると、関連するオブジェクトも自動的に間接メンバーとして追加されることがあります。たとえば、プロジェクトを追加するとそのメンバーが含まれ、タスクを追加した場合はタスクに関連するオブジェクトが追加されます。オブジェクトの展開方法の詳細については、『Rational Synergy Distributed』を参照してください。

> **注記**：オブジェクトを転送セットに追加しても、それが必ず送信されるわけではありません。一部のオブジェクトは、デフォルトの除外ルールによって除外されます。転送セットの設定によって一部のオブジェクトが除外されることもあります。
>
> 反対に、定義済みの **Entire Database** 転送セットには、自動的にすべてのプロジェクト、ディレクトリ、ファイル、タスク、フォルダ、およびベースラインが含まれます。**Entire Database** 転送セットにオブジェクトを追加することはできません。

DCM マネージャまたはアドミニストレータが、転送セットにオブジェクトを追加することができます。

```
ccm dcm -add -ts|-transfer_set transfer_set_spec
       ([-h|-history] | [-nh|-no_history]) object_spec...
```

`-h|-history`

ファイル、ディレクトリ、またはプロジェクトにこのオプションを使用することで、オブジェクトの履歴内のすべてのバージョンを履歴メンバーとして追加するよう指定します。このオプションは、タスクとフォルダには影響しません。履歴メンバーであるオブジェクトの新しいバージョンをチェックアウトすると、新しいバージョンは自動的に履歴メンバーになります。

デフォルトは、現在の DCM 設定によって決まります。デフォルトでは、オブジェクトの履歴内のバージョンを履歴メンバーとして追加しません。

`object_spec...`

転送セットに追加するオブジェクトを指定します。詳細については、オブジェクトの指定を参照してください。

`-ts|-transfer_set` *`transfer_set_spec`*

オブジェクトをメンバーとして追加する転送セットを指定します。*`transfer_set_spec`* で 1 つの転送セットを指定する必要があります。詳細については、転送セットの指定を参照してください。

例

- `infotec-23` プロジェクトを **InfoServer source** 転送セットに追加する。

  ```
  ccm dcm -add -ts "InfoServer source" infotec-23:project:1
  ```

  クエリ選択セットを使用してオブジェクト名を指定することもできます。

関連トピック

- 転送セットの作成
- 転送セットの削除
- 転送パッケージの生成
- 転送セットの修正
- パッケージの受取り
- 転送セットのメンバーの再計算
- 転送セットからのオブジェクトの削除
- DCM プロパティの表示
- データベース ID の表示
- 転送セット情報の表示
- 転送セット メンバーの表示
- 転送セットの表示

データベース定義の作成

このサブコマンドにより、DCM クラスタ内の別のデータベースを表す DCM データベース定義を作成します。これは、そのデータベースにデータを複製する前に必要です。データベース定義は、そのデータベースの DCM データベース ID と一致するデータベース ID を使用する必要があります。データベース定義は、DCM がパッケージをそのデータベースに転送する方法、自動受取りの有無、自動受取りの場合その受取り方法を定義します。

Any データベース定義は、「ブロードキャスト パッケージ」を作成するための特殊な定義済みデータベースを表します。これは、DCM 生成によって作成され、対応する Synergy バージョンの任意のデータベースで受け取られます。

DCM マネージャまたはアドミニストレータが、データベース定義を作成できます。

```
ccm dcm -c|-create -dbid|-database_id new_database_id
       [-desc|-description description]
       [-tm|-transfer_mode ((manual|manual_copy) | direct |
       (cp|copy|local_copy) | (rcp|remote_copy) |
       (ftp|file_transfer_protocol) | (user|user_defined))]
       ([-ar|-automatic_receive] | [-noar|-noautomatic_receive])
       ([-rb|-run_in_background] | [-norb|-norun_in_background])
       [-host host] [-os|-operating_system (unix | (windows|windows_nt))]
       [-path path] [-tp|-transfer_path path] [-ccm_home ccm_home]
       ([-zip] | [-nozip]) ([-ga|-generate_allowed] |
       [-noga|-nogenerate_allowed]) ([-handover_allowed] |
       [-nohandover_allowed]) ([-hidden] | [-nohidden])
       [-location location] [-admin_info admin_info]
```

`-admin_info` *`admin_info`*

データベース アドミニストレータの連絡先情報を指定します。*`admin_info`* 値には、復帰改行文字を含まない任意の文字列を指定できます。たとえば、電話番号や電子メール アドレスは有効な項目です。

`-ar|-automatic_receive`

DCM がこのデータベースへ生成した後、転送パッケージが自動的に受け取られるよう指定します。詳細については、『Rational Synergy Distributed』の「自動受取り設定と使用」を参照してください。デフォルトは、`-noar|-noautomatic_receive` です。

`-ccm_home` *`path`*

Synergy `$CCM_HOME` インストール パスを指定します。Rational Synergy は、このパスを使用してデータベース定義用に生成されるパッケージの自動受取りを行います。対応するデータベースのホストが UNIX サーバーである場合は、絶対パスを入力します。Windows サーバーの場合は UNC パスを入力します。

`-dbid|-database_id` *`new_database_id`*

新しいデータベース定義の DCM データベース ID を指定します。*`new_database_id`* は、8 文字以内で DCM クラスタ内で一意である必要があります。小文字データベースを使用するクラスタでは、ID は大文字／小文字にかかわらず一意である必要があります。たとえば、2 つの異なる定義に「a」と「A」を使用しないでください。

DCM データベースの命名制限の詳細については、DCM の制限を参照してください。

`-desc|-description` *`description`*

作成するデータベース定義の説明を指定します。*`description`* には、復帰改行文字を含めることはできません。

`-ga|-generate_allowed`

新しいデータベースが DCM 生成を使用できることを指定します。これはデフォルト設定です。

`-handover_allowed`

新しいデータベースにオブジェクトの管理を付与することを指定します。デフォルトは、`-nohandover_allowed` です。

`-hidden`

データベース ID を一覧表示するダイアログで、新しいデータベース定義を表示しないよう指定します。

デフォルトは、`-nohidden` です。

`-host` *`host`*

データベース ホストとするマシンの名前を指定します。これは、リモート コピーおよびファイル転送プロトコル転送モードに必要です。*`host`* 名が有効であり、IP アドレスに設定されていることを確認してください。

`-location` *`location`*

データベースの地理的な位置を指定します（例、`Irvine, California`）。位置には、復帰改行文字を含まない任意の文字列を使用できます。

`-noar|-noautomatic_receive`

DCM がこのデータベースへ生成した後、転送パッケージを自動的に受け取らないよう指定します。パッケージは手動で受け取る必要があります。これはデフォルト設定です。

`-noga|-nogenerate_allowed`

新しいデータベースが DCM 生成を使用できないことを指定します。デフォルトは、`-ga|-generate_allowed` です。

`-nohandover_allowed`

新しいデータベースにオブジェクトの管理を付与しないよう指定します。これはデフォルト設定です。

`-nohidden`

データベース ID を一覧表示するダイアログで、新しいデータベース定義を表示するよう指定します。これはデフォルト設定です。

`-norb|-norun_in_background`

自動受取りを使用している場合、バックグランドで自動受取りを実行しないよう指定します。

自動受取りを使用している場合、DCM 生成がパッケージの生成と転送を完了したとき、デスティネーション データベースでセッションを開始してパッケージの受取りを開始します。`-norun_in_background` を使用している場合、生成は受取りの完了を待ってデスティネーション データベースでの受取りの結果を表示します。

これはデフォルト設定です。

`-nozip`

データベースに対して生成する転送パッケージを圧縮しないよう指定します。これは、`direct` 転送モードのデフォルトです。このオプションは、`file transfer protocol` 転送モードでは使用できません。

`-rb|-run_in_background`

自動受取りを使用している場合、バックグランドで自動受取りを実行するよう指定します。

自動受取りを使用している場合、DCM 生成がパッケージの生成と転送を完了したとき、デスティネーション データベースでセッションを開始してパッケージの受取りを開始します。`-run_in_background` を使用している場合、生成は受取りの完了を待た

ずに直ちに戻ります。これは、転送パッケージの受取りを待たずにセッションを継続できることを意味しますが、受取りが成功したかを確認できません。後で、デスティネーション データベースの DCM イベント ログで受取りの詳細を表示することができます。

デフォルトは、`-norb|-norun_in_background` です。

`-os|-operating_system (unix|(windows|windows_nt))`

データベース ホストとするマシンのオペレーティング システムを指定します。

デフォルトでは、現在のデータベースのサーバーと同じ設定を使用します。

`-path` *`path`*

データベースへのパスを指定します。UNIX サーバーでは絶対パス、Windows サーバーでは UNC パスを使用します。このオプションは、手動以外の転送モードを使用する場合データベース パスを必要とします。

`-tp|-transfer_path` *`transfer_path`*

データベースへの転送パスを指定します。転送パスは、転送パッケージが置かれている場所です。これは、UNIX サーバーでは絶対パスで、Windows サーバーでは UNC パスです。転送パスを指定しないか空の文字列を使用した場合、Rational Synergy はパッケージをデータベース パスの下の **dcm/receive** ディレクトリの下に置きます。

`-tm|-transfer_mode` *`value`*

新しいデータベース定義に使用する転送モードを指定します。デフォルトは手動です。転送モードは、転送パッケージをデスティネーション データベースへ転送する方法を定義します。

transfer_mode オプションは、以下のいずれかの値をとる必要があります。

- `manual | manual_copy`

  パッケージは、DCM によってコピーされず、手動で転送する必要があります。

- `cp | copy | local_copy`

  パッケージは生成後にデスティネーション データベースの転送パスにコピーされます。

- `direct`

  パッケージは直接デスティネーション データベースの転送パスに生成されます。

- `ftp | file_transfer_protocol`

  パッケージは、生成後に *ftp* を使用してデスティネーション データベースの転送パスに転送されます。

- `rcp | remote_copy`

  パッケージは、生成後に *rcp* を使用してデスティネーション データベースの転送パスにリモート コピーされます。

- `user | user_defined`

  パッケージは、生成後にユーザーがカスタマイズしたシェル スクリプト（UNIX）またはバッチ ファイル（Windows）を起動して転送されます。

転送モードの詳細については、『Rational Synergy Distributed』を参照してください。

`-zip`

データベースに対して生成する転送パッケージを圧縮するよう指定します。これは、`direct` 以外のすべての転送モードのデフォルトです。このオプションは `direct` と一緒には使用できません。

関連トピック

- データベース定義の削除
- データベース定義の修正
- データベース定義情報の表示
- 現在の DCM データベース ID の表示

転送セットの作成

このサブコマンドにより、転送セットを作成します。転送セットは、他のデータベースに複製するオブジェクトの集まりを表します。複製するオブジェクトは転送セットに追加することができます。詳細については、転送セットへのオブジェクトの追加を参照してください。**Entire Database** 転送セットは、自動的にすべてのプロジェクト、ディレクトリ、ファイル、タスク、フォルダ、およびベースラインを含む、特殊な定義済みデータベースです。

DCM マネージャまたはアドミニストレータが、転送セットを作成できます。

```
ccm dcm -c|-create -ts|-transfer_set new_transfer_set_name
       ([-email email_address] | [-noemail])
       [-ep|-email_policy (generate | transfer | always)]
       [-crsc|-change_request_scope|-ps|-problem_scope (none |
       (crs|crs only|change_requests|problems) |
       (crs_and_tasks|crs and tasks|
       change_requests_and_tasks|problems_and_tasks) |
       (crs_tasks_and_objects|crs, tasks and objects|
       change_requests_tasks_and_objects|problems_tasks_and_objects))]
       [-crq|-change_request_query|-pq|-problem_query cr_query]
       ([-cumcrsc|-cumulative|-cumulative_change_request_scope] |
       [-nocumcrsc|-nocumulative|-nocumulative_change_request_scope])
       [-rsc|-release_scope (none | releases |
       (releases_templates|releases_and_templates|releases and templates))]
       [-rq|-release_query release_query]
       ([-cumrsc|-cumulative_release_scope] |
       [-nocumrsc|-nocumulative_release_scope]) ([-exclude_products] |
       [-noexclude_products]) ([-exclude_imported_objects] |
       [-noexclude_imported_objects])
       ([-exclude_nct|-exclude_non_completed_tasks] |
       [-noexclude_nct|-noexclude_non_completed_tasks])
       ([-exclude_typedefs] | [-noexclude_typedefs]) ([-exclude_db_info] |
       [-noexclude_db_info]) ([-ib|-include_baselines] |
       [-noib|-noinclude_baselines]) [-exclude_types type1,type2,...]
       ([-ferp] | [-noferp]) ([-local_parallel] | [-nolocal_parallel])
       [-dir|-directory generate_dir]
```

`-crq|-change_request_query|-pq|-problem_query cr_query`

転送セットの変更依頼範囲と共に使用する変更依頼クエリを指定します。変更依頼クエリは、変更依頼範囲を none 以外の値に設定した場合にのみ使用できます。

値は、空白か有効なクエリ式のいずれかです。空白値は、すべての変更依頼をクエリすることを意味します。これはデフォルト値です。

`-crsc|-change_request_scope|-ps|-problem_scope` *scope*

変更依頼と関連するオブジェクトを、転送セットを使用して生成した転送パッケージに含める方法を指定します。以下の範囲から選択します。

- `none`

  変更依頼は自動的に含まれません。これはデフォルト設定です。

- `crs|crs only|change_requests|problems`

  変更依頼クエリによって検出された変更依頼はその添付と共に含まれます。

- `crs_and_tasks|crs and tasks|`
  `change_requests_and_tasks|problems_and_tasks`

  変更依頼クエリによって検出された変更依頼はその添付および関連タスクと共に含まれます。

- `crs_tasks_and_objects|crs, tasks and objects|`
  `change_requests_tasks_and_objects|problems_tasks_and_objects`

  変更依頼クエリによって検出された変更依頼はその添付と関連タスク、および各タスクの関連オブジェクトと共に含まれます。

転送セットの直接メンバーとして明示的に追加された変更依頼、タスク、および他のオブジェクトはこのオプションの影響を受けません。

`-cumcrsc|-cumulative|-cumulative_change_request_scope`

新しい転送セットの変更依頼範囲を累積に指定します。転送セットの変更依頼範囲とクエリは、各生成操作または生成プレビュー操作時に評価されます。ただし、`-cumulative` を指定した場合、現在のクエリで検出されなくても、以前のクエリで検出された古いメンバーは永遠に削除されません。つまり、変更依頼の間接（クエリベース）メンバーシップは、追加されて累積されます。

`-cumrsc|-cumulative_release_scope`

新しい転送セットのリリース範囲を累積に指定します。転送セットのリリース範囲とクエリは、各生成操作または生成プレビュー操作時に評価されます。ただし、`-cumulative_release_scope` を指定した場合、現在のクエリで検出されなくても、以前のクエリで検出された古いメンバーは永遠に削除されません。つまり、リリースの間接（クエリベース）メンバーシップは、追加されるのみで、累積されていきます。

`-dir|-directory` *generate_dir*

新しい転送セットに対して、直接以外の転送モードでの転送パッケージの生成を準備するために指定した生成ディレクトリを使用することを指定します。*generate_dir* 値はサーバー パスを表します。UNIX サーバーでは絶対パス、Windows サーバーでは UNC パスを使用します。空白文字列を使用すると Rational Synergy はデータベー

ス パスの下の **dcm/generate** ディレクトリの下にあるデフォルトの生成ディレクトリを使用します。これはデフォルト設定です。

`-email` *`email_address`*

転送セットの生成、受取りまたは転送の後に、電子メールを受け取る 1 人または複数の電子メール アドレスを指定します。

アドレスを空白またはカンマで区切ることにより、転送セットの複数の電子メール受信者を定義できます。電子メール リストを定義したい場合は、自分のメール サーバーの機能を使用して、電子メールの別名または配布先リストを設定できます。この操作方法については、使用するメール サーバーとオペレーティング システムを調べてください。

デフォルトは、`-noemail` です。

`-ep|-email_policy` *`policy`*

生成操作時および転送操作時に使用する電子メール ポリシーを指定します。このオプションは、以下の電子メール ポリシーをサポートします。

- `Transfer`：デスティネーション データベースに空でないパッケージを転送されたときのみ、電子メール メッセージが送られるよう指定します。DCM 生成操作が実行されたときに含まれるオブジェクトがない場合、メッセージは送出されません。
- `Generate`：空でない転送パッケージが生成または転送されたときに、電子メール メッセージが送られるよう指定します。DCM 生成操作が実行されたときに含まれるオブジェクトがない場合、メッセージは送出されません。これはデフォルト設定です。
- `Always`：空ではない転送パッケージを生成または転送するときは、電子メール メッセージが必ず送出されるよう指定します。DCM 生成操作の実行後に含まれるオブジェクトがない場合、あるいはデスティネーション データベースに自動配信しなパッケージを生成する場合が、これに当てはまります。

`-exclude_db_info`

新しい転送セットから生成した転送パッケージから、データベース定義に関する情報を除外するよう指定します。デフォルトは、`-noexclude_db_info` です。

`-exclude_imported_objects`

新しい転送セットから生成した転送パッケージから他のデータベースで作成したオブジェクトを除外するよう指定します。デフォルトは、`-noexclude_imported_objects` です。

`-exclude_nct|-exclude_non_completed_tasks`

新しい転送セットから生成した転送パッケージから、完了していないタスクを除外するよう指定します。

デフォルトは、`-noexclude_non_completed_tasks` です。

`-exclude_products`

新しい転送セットから生成した転送パッケージから、製品を除外するよう指定します。デフォルトは、`-noexclude_products` です。

`-exclude_types type1,type2,...`

新しい転送セットから生成した転送パッケージから、指定したタイプのオブジェクトを除外するよう指定します。この値は、ゼロ個以上のタイプ名をカンマとオプションの空白で区切ったリストです。デフォルトは空のリストです。

`-exclude_typedefs`

新しい転送セットから生成した転送パッケージから、タイプ定義を除外するよう指定します。デフォルトは、`-noexclude_typedefs` です。

`-ferp`

転送セットのメンバーであるプロジェクトの更新プロパティを完全に展開するよう指定します。このオプションを指定した結果は以下のとおりです。

- メンバーであるタスクに関連するすべてのオブジェクトは、プロジェクト階層のメンバーではない場合でも含まれる。
- プロジェクトの更新プロパティ内のすべてのフォルダ、タスク、およびベースライン プロジェクトは、静的状態のプロジェクトの場合にも含まれる。
- 各ベースラインプロジェクトのすべてのサブプロジェクトは、プロジェクト階層の更新プロパティで直接使用されていない場合でも含まれる。

これにより転送セットの間接メンバーの数が大幅に増え、間接メンバーを計算する時間が長くなります。詳細については、『Rational Synergy Distributed』の「リコンフィギュア プロパティの全展開」を参照してください。

デフォルトは、`-noferp` です。

`-ib|-include_baselines`

転送のメンバーであるオブジェクトに関連するベースラインを含めるよう指定します。デフォルトは、`Default Include Baselines` の DCM 設定によって決まります。デフォルトは、`-noinclude_baselines` です。

`-local_parallel`

新しい転送セットから生成された転送パッケージから受け取ったパラレル オブジェクト バージョンのローカル所有者に、電子メールによりパラレル通知を送信するよう指定します。これはデフォルト設定です。

`-nocumcrsc|-nocumulative|-nocumulative_change_request_scope`

新しい転送セットの変更依頼範囲を累積なしに指定します。転送セットの変更依頼範囲とクエリは、各生成操作または生成プレビュー操作時に評価されます。このオプションを使用した場合、以前のクエリで検出された古いメンバーで現在のクエリで検出されなかったものは、転送セットの間接（クエリベース）メンバーなので削除されます。

これはデフォルト設定です。

`-nocumrsc|-nocumulative_release_scope`

新しい転送セットのリリース範囲を累積なしに指定します。転送セットのリリース範囲とクエリは、各生成または生成プレビュー操作時に評価されます。このオプションを使用した場合、以前のクエリで検出された古いメンバーで現在のクエリで検出されなかったものは、転送セットの間接（クエリベース）メンバーなので削除されます。これはデフォルト設定です。

`-noemail`

転送セットの生成、受取り、または転送後に電子メールを送信しないよう指定します。これはデフォルト設定です。

`-noexclude_db_info`

新しい転送セットから生成した転送パッケージにデータベース定義に関する情報を含めるよう指定します。これはデフォルト設定です。

`-noexclude_imported_objects`

新しい転送セットから生成した転送パッケージに他のデータベースで作成したオブジェクトを含めるよう指定します。これはデフォルト設定です。

`-noexclude_nct|-exclude_non_completed_tasks`

新しい転送セットから生成した転送パッケージに完了していないタスクを含めるよう指定します。これはデフォルト設定です。

`-noexclude_products`

新しい転送セットから生成した転送パッケージに製品を含めるよう指定します。これはデフォルト設定です。

`-noexclude_typedefs`

新しい転送セットから生成した転送パッケージにタイプ定義を含めるよう指定します。これはデフォルト設定です。

`-noferp`

転送セットのメンバーであるプロジェクトの更新プロパティを完全に展開しないよう指定します。このオプションを指定した結果は以下のとおりです。

- プロジェクトの更新プロパティ内にあるタスクの関連オブジェクトは、プロジェクト階層のメンバーである場合にのみ含まれる。
- 静的状態のプロジェクトのフォルダ、タスク、およびベースライン プロジェクトは含まれない。
- ベースライン　プロジェクトのサブプロジェクトは、プロジェクトの更新プロパティで直接使用されている場合でも含まれる。

これはデフォルトです。詳細については、『Rational Synergy Distributed』の「リコンフィギュア プロパティの全展開」を参照してください。

`-noib|-noinclude_baselines`

転送のメンバーであるオブジェクトに関連するベースラインを自動的に含めないよう指定します。デフォルトは、`Default Include Baselines` の DCM 設定によって決まります。これはデフォルト設定です。

`-nolocal_parallel`

新しい転送セットから生成された転送パッケージから受け取ったオブジェクトのパラレルのローカル所有者に、電子メールによりパラレル通知を送信しないよう指定します。デフォルトは、`-local_parallel` です。

`-nomail`

転送セットの生成、受取り、または転送に対して電子メールを送信しないよう指定します。これはデフォルト設定です。

`-rq|-release_query` *`release_query`*

新しい転送セットのリリース範囲と一緒に使用するリリース クエリを指定します。*`release_query`* は、空白文字列または有効なクエリ式のいずれかです。空白文字列は、すべてのリリースをクエリすることを意味します。リリース クエリは、リリース範囲を none 以外の値に設定した場合にのみ使用できます。これはデフォルト設定です。

`-rsc|-release_scope (none | releases | (releases_templates | releases_and_templates | releases and templates))`

新しい転送セットのリリース範囲を指定します。以下の範囲を使用します。

- `none`

  リリース定義は自動的に含まれません。

- `releases`

  リリース クエリによって検出されたリリース定義は、間接クエリ メンバーとして自動的に含まれます。ただし、対応するプロセス ルールとフォルダ テンプレートは自動的に含まれません。

- `releases_and_templates`

  リリース クエリによって検出されたリリース定義は、間接クエリ メンバーとして自動的に含まれます。各リリースに対して、そのプロセス ルールとプロセス ルールが使用する任意のユーザー定義フォルダ テンプレートも間接クエリ メンバーとして自動的に含まれます。これはデフォルト設定です。

`-ts|-transfer_set` *`transfer_set_name`*

作成する転送セットの名前を指定します。名前は任意の文字を含むことができますが、このデータベース内で一意である必要があります。

例

- オブジェクト タイプを転送セットから除外する（転送セット作成時に）。

  `ccm dcm -create -ts` *`transfer_set_spec`* `-exclude_types "`*`list_of_types`*`"`

- 製品を転送セットに含める（転送セット作成時に）。

  `ccm dcm -create -ts` *`transfer_set_spec`* `-noexclude_products`

- インポートしたオブジェクトを転送セットから除外する（転送セット作成時に）。

  `ccm dcm -create -ts` *`transfer_set_spec`* `-exclude_imported_objects`

- データベース情報を転送セットから除外する（転送セット作成時に）。

  `ccm dcm -create -ts` *`transfer_set_spec`* `-exclude_db_info`

関連トピック

- 転送セットへのオブジェクトの追加
- 転送セットの作成
- 転送セットの削除
- 転送パッケージの生成
- 転送セットの修正
- パッケージの受取り
- 転送セットの間接リリース メンバーの再計算
- 転送セットのメンバーの再計算
- 転送セットからのオブジェクトの削除
- DCM プロパティの表示
- 最新生成時刻の表示
- 転送セット情報の表示
- 転送セット メンバーの表示
- 転送セットの表示
- 生成パッケージの転送

データベース定義の削除

このサブコマンドにより、DCM データベース定義を削除します。データベース定義を削除すると、それに対して転送セットを使用していつ DCM 生成操作を行ったかについての情報も削除されます。これは、対応するデータベースがすでに DCM クラスタ内に存在しない場合に適切です。データベースを削除するのではなく退去させる場合は、データベース定義を非表示とするか、生成を許可しないように設定できます。

DCM マネージャまたはアドミニストレータが、データベース定義を削除できます。

```
ccm dcm -d|-delete -dbid|-database_id database_spec...
```

database_spec...

削除するデータベース定義を指定します。詳細については、データベースの指定を参照してください。

関連トピック

- データベース定義の作成
- データベース定義の修正
- データベース定義情報の表示
- データベース ID の表示

転送セットの削除

このサブコマンドにより、転送セットを削除します。転送セットを削除することにより、それに直接追加されたすべてのメンバーおよび対応する間接メンバーとの関連も解除します。転送セットを削除すると、その転送セットを使用していつ DCM 生成操作を行ったかについての情報も削除されます。定義済み `Entire Database` 転送セットは削除できません。

DCM マネージャまたはアドミニストレータが、転送セットを削除できます。

```
ccm dcm -d|-delete -ts|-transfer_set transfer_set_spec...
```

`transfer_set_spec...`

削除する転送セットを指定します。詳細については、転送セットの指定を参照してください。

例

- 1 つまたは複数の転送セットを削除する。

  ```
  ccm dcm -delete -ts "My transfer set"
  ```

関連トピック

- 転送セットへのオブジェクトの追加
- 転送セットの作成
- 転送パッケージの生成
- 転送セットの修正
- 転送セットの間接変更依頼メンバーの再計算
- 転送セットの間接リリース メンバーの再計算
- 転送セットのメンバーの再計算
- 転送セットからのオブジェクトの削除
- 転送セット情報の表示
- 転送セット メンバーの表示
- 転送セットの表示
- 生成パッケージの転送

パッケージの削除

このサブコマンドにより、指定したサーバー ファイル システム位置からパッケージを削除します。DCM マネージャまたはアドミニストレータが、パッケージを削除できます。-generate、-receive、または -serverdir のいずれかのオプションを指定する必要があります。

```
ccm dcm -d|-delete -packages [-from_dbid|-from_database_id database_spec]
        [-to_dbid|-to_database_id database_spec]
        [-ts|-transfer_set transfer_set_spec] ([-gen|-generate] |
        [-rec|-receive] | [-soad|-save_offline_and_delete] |
        [-serverdir server_path])
```

`-from_dbid|-from_database_id database_spec`

転送パッケージの生成元のデスティネーション　データベースを指定します。*database_spec* には、1 つのデータベース定義を指定できます。詳細については、データベースの指定を参照してください。

`-gen|-generate`

削除するパッケージをサーバー生成ディレクトリで検索するよう指定します。

`-rec|-receive`

削除するパッケージをサーバー受取りディレクトリで検索するよう指定します。

`-soad|-save_offline_and_delete`

削除するパッケージを、オフライン保存 パッケージのデフォルト サーバー ディレクトリで検索するよう指定します。

`-to_dbid|-to_database_id database_spec`

転送パッケージの生成先のデスティネーション　データベースを指定します。*database_spec* には、1 つのデータベース定義を指定できます。詳細については、データベースの指定を参照してください。

`-serverdir server_path`

削除するパッケージを含むファイル システム内のサーバー ディレクトリへのパスを指定します。

`-ts|-transfer_set` *`transfer_set_spec`*

パッケージの生成に使用した転送セット名でパッケージのフィルタリングを行うよう指定します。*`transfer_set_spec`* には、1 つの転送セットを指定できます。詳細については、転送セットの指定を参照してください。

例

- 転送セット `webapp` を使用して生成し、サーバー生成ディレクトリにあるすべてのパッケージを削除する。

  ```
  ccm dcm -delete -packages -ts webapp -generate
  ```

- サーバー受取りディレクトリにある、データベース `INDWEB` から生成したすべてのパッケージを削除する。

  ```
  ccm dcm -delete -packages -from_dbid INDWEB -receive
  ```

関連トピック

- 転送パッケージの生成
- パッケージの受取り
- パッケージの表示
- 生成パッケージの転送

転送パッケージの生成

このサブコマンドにより、指定した転送セットとデスティネーション データベースの転送パッケージを生成します。転送セットの間接メンバーが古い場合、このコマンドは転送セット内の現在の直接メンバーから計算しなおします。その後、そのデータベースの最後の生成時以降に転送セットに追加されたか、そのデータベースの最後の生成時以降に修正された直接メンバーと間接メンバーが転送パッケージに含まれます。オプションとして、デスティネーション データベースに対して定義されている転送モードを使用して、転送パッケージをそのデータベースに転送できます。パッケージを転送する場合、デスティネーション データベースにパッケージの自動受取りを設定できます。

DCM マネージャまたはアドミニストレータが、転送パッケージを生成できます。

```
ccm dcm -gen|-generate -dbid|-database_id database_spec
        -ts|-transfer_set transfer_set_spec
        [-lg|-last_generated last_generated_value]
        ([-email email_address] | [-noemail])
        ([-trn|-transfer [-rec|-receive ( [-wait] | [-nowait] )]] |
        [-notrn|-notransfer])
```

`-dbid|-database_id` *`database_spec`*

転送パッケージを生成するデスティネーション データベースを指定します。*`database_spec`* には、1 つのデータベース定義を指定できます。詳細については、データベースの指定を参照してください。

`-email` *`email_address`*

転送セットの生成の後に、電子メールを受け取る 1 人または複数の電子メール アドレスを指定します。

アドレスを空白またはカンマで区切ることにより、転送セットの複数の電子メール受信者を定義できます。電子メール リストを定義したい場合は、自分のメール サーバーの機能を使用して、電子メールの別名または配布先リストを設定できます。この操作方法については、使用するメール サーバーとオペレーティング システムを調べてください。

電子メール アドレスを設定しない場合、Rational Synergy は `dcm generate` コマンドで指定した転送セットで定義されている電子メール アドレスを使用します。

`-lg|-last_generated` *`last_generated_value`*

最後に生成を行った時刻を指定します。

注記：このオプションは上級ユーザー用です。

指定しない場合、パッケージは指定した転送セットとデータベースの転送セットが最後に生成された時刻を使用するよう生成されます。このオプションは、紛失転送

パッケージからのリカバリなど、以前の変更を含む転送パッケージの生成に使用します。

– never

– integer index。1 は最新の生成された転送パッケージを示します。

最新のタイムスタンプではないタイムスタンプを選択した場合、生成された転送パッケージには、その日付以降に変更があったかメンバーとなったすべてのオブジェクトが含まれます。また、最新のタイムスタンプがリストから削除されます。

> **注意！** never を使用すると、すべての過去のタイム スタンプがリストから削除されます。

last_generated_value が never の場合は、初めてであるかのように転送パッケージが生成されます。転送パッケージには、最後に修正されたのはいつか、あるいは転送セットのメンバーになったのはいつか、に関わらずすべてのオブジェクトが含まれます。

-noemail

転送セットの生成後に電子メールを送信しないよう指定します。デフォルトで、Rational Synergy は転送セットの電子メール設定を使用します。

-notrn|-notransfer

生成されたパッケージをデスティネーション データベースに転送しないよう指定します。このオプションは、デスティネーション データベースが direct 以外の転送モードを持つ場合に使用できます。これはデフォルト設定です。

-nowait

受取りデータベースが他の転送パッケージの受取りを完了していなくても、自動受取りを実行できるようにします。

> **注意！** 複数の転送パッケージを同時にデータベースで受け取るのは安全ではありません。デフォルトでは、DCM はパッケージを 1 つずつデータベースに受け取ります。このオプションは、転送パッケージが重複オブジェクトを含まない確証がある場合にのみ使用します。たとえば、それぞれ異なる転送セットから生成された 2 つのパッケージに同じタスク オブジェクトが存在する場合、それらのパッケージは重複オブジェクトを持ちます。この場合、デフォルトの -wait を使用してください。

`-rec|-receive`

生成されたパッケージをデスティネーション データベースで受け取るよう指定します。このオプションは、パッケージがデスティネーション データベースに転送され、データベース定義のホストとデータベース パスが定義されている場合に使用します。詳細については、『Rational Synergy Distributed』の「自動受取り設定と使用」を参照してください。

`-trn|-transfer`

生成されたパッケージをデスティネーション データベースに転送するよう指定します。このオプションは、デスティネーション データベースが `none` または `direct` 以外の転送モードを持つ場合に使用できます。デフォルトは `-notransfer` です。

`-ts|-transfer_set` *`transfer_set_spec`*

DCM 生成に転送セットを使用するよう指定します。*`transfer_set_spec`* には、1 つの転送セットを指定できます。詳細については、転送セットの指定を参照してください。

`-wait`

受取りデータベースが他の転送パッケージの受取りを完了するまで、自動受取りを実行できないようにします。これはデフォルト設定です。

例

- **Secure transformer layer** 転送セットと **BST** データベースの転送パッケージを作成し、後で転送するためにそれを保存する。

```
ccm dcm -gen -ts "Secure transformer layer" -dbid BST
```

```
Computing transfer package...
Computing transfer package for 'Secure transformer layer' going to
database 'BST'...
115 objects will be included in transfer package for 'Secure
transformer layer' going to database 'BST'...
```

```
Generating transfer package...
...
DCM data generated to file
 '¥¥ccmsrv¥ccmdbs¥appdevdb¥dcm¥generate¥CA#7#BST#865889312.tar.gz'
Updating database...
DCM Generate completed successfully.
```

関連トピック

- 転送セットへのオブジェクトの追加
- 転送セットの作成
- 転送セットの削除
- 転送パッケージの生成
- パッケージの受取り
- 最新生成時刻の表示
- 転送セット情報の表示
- 転送セット メンバーの表示
- 転送セットの表示
- 生成パッケージの転送

データベースへの最新のマーク付け

このサブコマンドにより、データベース定義に、指定した転送セットの最新のマークを付けます。現在のデータベースのコピーをアンパックし、そのデータベース ID を変更してデータベースを作成する場合、データベースをバックアップしたときに存在していたすべてのオブジェクトがすでに含まれています。このサブコマンドは、オブジェクトを不必要に送るのを防ぐため、既存のデータベースへの複製を設定するときに便利です。

```
ccm dcm -mark_up_to_date -dbid|-database_id database_spec
        -ts|-transfer_set transfer_set_spec [-force] [date]
```

`-database_id database_spec`

転送パッケージのデスティネーション データベースを生成するよう指定します。*database_spec* には、1 つのデータベース定義を指定できます。詳細については、データベースの指定を参照してください。

date

データベースが最新の状態である日付を指定します。デフォルトで、Rational Synergy は現在の日時を使用します。別のデータベースをアンパックしてデータベースを作成した場合、このコマンドの日付にバックアップの日時を使用することができます。その日時に行われた変更は、コピーしたデータベースには存在しません。

`-force`

指定した転送セットとデスティネーション データベースが以前パッケージを生成していた場合でも、操作が成功することを指定します。デフォルトでは、転送セットとデスティネーション データベースが一度も転送パッケージを生成したことがない場合にのみ操作は成功します。

`-ts|-transfer_set transfer_set_spec`

指定したデスティネーション データベースに対して最新とマークする転送セットを指定します。*transfer_set_spec* には、1 つの転送セットを指定できます。詳細については、転送セットの指定を参照してください。

関連トピック

- DCM プロパティの表示

データベース定義の修正

このサブコマンドにより、指定したデータベース定義を修正します。DCM マネージャまたはアドミニストレータが、この操作を行うことができます。

```
ccm dcm -m|-modify -dbid|-database_id [-desc|-description description]
        [-tm|-transfer_mode ((manual|manual_copy) | direct |
        (cp|copy|local_copy) | (rcp|remote_copy) |
        (ftp|file_transfer_protocol) | (user|user_defined))]
        ([-ar|-automatic_receive] | [-noar|-noautomatic_receive])
        ([-rb|-run_in_background] | [-norb|-norun_in_background])
        [-host host] [-os|-operating_system (unix | (windows|windows_nt))]
        [-path path] [-tp|-transfer_path path] [-ccm_home path]
        ([-zip] | [-nozip]) ([-ga|-generate_allowed] |
        [-noga|-nogenerate_allowed]) ([-handover_allowed] |
        [-nohandover_allowed]) ([-hidden] | [-nohidden])
        [-location location] [-admin_info admin_info]
        [-new_dbid|-new_database_id new_dbid] database_spec...
```

`-admin_info` *admin_info*

指定したデータベースの連絡先情報を修正するよう指定します。*admin_info* 値には、復帰改行文字を含まない任意の文字列を指定できます。たとえば、電話番号や電子メール アドレスは有効な項目です。

`-ar|-automatic_receive`

指定したデータベースの転送パッケージの自動受取りを指定します。詳細については、『Rational Synergy Distributed』の「自動受取り設定と使用」を参照してください。

`-ccm_home` *path*

指定したデータベースについて、自動受取りを行うために使用する Synergy `$CCM_HOME` インストール パスを変更します。対応するデータベースのホストが UNIX サーバーである場合は、絶対パスを入力します。Windows サーバーの場合は UNC パスを入力します。

database_spec

修正するデータベース定義を指定します。データベースの指定を参照してください。

`-desc|-description` *description*

指定したデータベース説明を修正するよう指定します。復帰改行文字を含めることはできません。

`-ga|-generate_allowed`

DCM 生成が使用できるように、データベース定義を修正することを指定します。

`-handover_allowed`

オブジェクトの管理をデータベース定義に渡すことができるように、データベース定義を修正することを指定します。

`-hidden`

指定したデータベース定義がデータベース ID を一覧表示するダイアログに表示されないように、非表示のマーク付けを行うことを指定します。このオプションは、データベース定義を削除せずに退去させるときに便利です。

`-host` *`host`*

指定したデータベース定義のホストを修正するよう指定します。これは、リモート コピーおよびファイル転送プロトコル転送モードに必要です。*`host`* 名が有効であり、IP アドレスに設定されていることを確認してください。

`-location` *`location`*

指定したデータベースの地理的位置を修正するよう指定します。位置には、復帰改行文字を含まない任意の文字列を使用できます。

`-new_dbid|-new_database_id` *`new_dbid`*

指定したデータベース定義に割り当てる新しいデータベース ID を指定します。制限される文字と制限される理由については、命名制限を参照してください。

`-noar|-noautomatic_receive`

転送パッケージを自動的に受け取らないよう指定します。パッケージは手動で受け取る必要があります。

`-nohandover_allowed`

オブジェクトの管理をデータベース定義に渡すことができないように、データベース定義を修正することを指定します。

`-nohidden`

データベース ID を一覧表示するダイアログで、新しいデータベース定義を表示するよう指定します。

`-noga|-nogenerate_allowed`

DCM 生成が使用できないように、データベース定義を修正することを指定します。

`-norb|-norun_in_background`

`-nozip`

指定したデータベースの転送パッケージを圧縮しないよう指定します。これは、`direct` 転送モードのデフォルトです。このオプションは、`file transfer protocol` 転送モードでは使用できません。

`-rb|-run_in_background`

`-os|-operating_system (unix|(windows|windows_nt))`

データベース ホストとするマシンのオペレーティン グシステムを、指定したデータベース定義用に修正するよう指定します。

`-path` *`path`*

指定したデータベース定義のデータベース パスを修正するよう指定します。UNIX サーバーでは絶対パス、Windows サーバーでは UNC パスを使用します。手動以外の転送モードを使用する場合は、データベース パスを入力する必要があります。

`-tm|-transfer_mode` *`value`*

指定したデータベース定義のために修正する転送モードを指定します。転送モードは、転送パッケージをデスティネーション データベースへ転送する方法を定義します。

`-transfer_mode` オプションは、以下のいずれかの値をとる必要があります。

- `manual | manual_copy`

  パッケージは、DCM によってコピーされず、手動で転送する必要があります。

- `cp | copy | local_copy`

  パッケージは生成後にデスティネーション データベースの転送パスにコピーされます。

- `direct`

  パッケージは直接デスティネーション データベースの転送パスに生成されます。

- `ftp | file_transfer_protocol`

パッケージは、生成後に *ftp* を使用してデスティネーション データベースの転送パスに転送されます。

- `rcp | remote_copy`

  パッケージは、生成後に `rcp` を使用してデスティネーション データベースの転送パスにリモート コピーされます。

- `user | user_defined`

  パッケージは、生成後にユーザーがカスタマイズしたシェル スクリプト（UNIX）またはバッチ ファイル（Windows）を起動して転送されます。

転送モードの詳細については、『Rational Synergy Distributed』を参照してください。

`-tp|-transfer_path` *`transfer_path`*

指定したデータベース定義のデータベースへの転送パスを修正するよう指定します。転送パスは、Rational Synergy が転送パッケージを置く場所です。これは、UNIX サーバーでは絶対パスで、Windows サーバーでは UNC パスです。転送パスを空白のままにした場合、Rational Synergy はパッケージをデータベース パスの下の **dcm/receive** ディレクトリの下に置きます。

`-zip`

指定したデータベースの転送パッケージを圧縮するよう指定します。これは、ファイル転送プロトコル転送モードのデフォルトです。このオプションは、`direct` と一緒には使用できません。

関連トピック

- データベース定義の作成
- データベース定義の削除
- データベース定義情報の表示
- DCM プロパティの表示
- データベース ID の表示
- 現在の DCM データベース ID の表示

転送セットの修正

このサブコマンドにより、指定した転送セットを修正します。DCM マネージャまたはアドミニストレータが、この操作を行うことができます。

```
ccm dcm -m|-modify -ts|-transfer_set ([-email email_address] |
       [-noemail]) [-ep|-email_policy (generate | transfer | always)]
       [-crsc|-change_request_scope|-ps|-problem_scope (none |
       (crs|crs only|change_requests|problems) |
       (crs_and_tasks|crs and tasks|change_requests_and_tasks|
       problems_and_tasks) | (crs_tasks_and_objects|crs,
       tasks and objects|change_requests_tasks_and_objects|
       problems_tasks_and_objects))]
       [-crq|-change_request_query|-pq|-problem_query cr_query]
       ([-cumcrsc|-cumulative|-cumulative_change_request_scope] |
       [-nocumcrsc|-nocumulative|-nocumulative_change_request_scope])
       [-rsc|-release_scope (none | releases |
       (releases_templates|releases_and_templates|releases and templates))]
       [-rq|-release_query release_query]
       ([-cumrsc|-cumulative_release_scope] |
       [-nocumrsc|-nocumulative_release_scope]) ([-exclude_products] |
       [-noexclude_products]) ([-exclude_imported_objects] |
       [-noexclude_imported_objects])
       ([-exclude_nct|-exclude_non_completed_tasks] |
       [-noexclude_nct|-noexclude_non_completed_tasks])
       ([-exclude_typedefs] | [-noexclude_typedefs]) ([-exclude_db_info] |
       [-noexclude_db_info]) ([-ib|-include_baselines] |
       [-noib|-noinclude_baselines]) [-exclude_types type1,type2,...]
       ([-ferp] | [-noferp]) ([-local_parallel] | [-nolocal_parallel])
       [-dir|-directory generate_dir]
       [-new_ts|-new_transfer_set new_transfer_set_name]
       transfer_set_spec...
```

`-crq|-change_request_query|-pq|-problem_query cr_query`

指定した転送セットの変更依頼クエリを修正するよう指定します。変更依頼クエリは、変更依頼範囲を none 以外の値に設定した場合にのみ使用できます。

値は、空白か有効なクエリ式のいずれかです。空白値は、すべての変更依頼をクエリすることを意味します。

`-crsc|-change_request_scope|-ps|-problem_scope scope`

指定した転送セットの変更依頼範囲を修正するよう指定します。範囲は、変更依頼と関連するオブジェクトを、転送セットを使用して生成した転送パッケージに含める方法を定義します。以下の範囲から選択します。

- none

  変更依頼は自動的に含まれません。これはデフォルト設定です。

- `crs|crs only|change_requests|problems`

  変更依頼クエリによって検出された変更依頼はその添付と共に含まれます。

- `crs_and_tasks|crs and tasks|`
  `change_requests_and_tasks|problems_and_tasks`

  変更依頼クエリによって検出された変更依頼はその添付および関連タスクと共に含まれます。

- `crs_tasks_and_objects|crs, tasks and objects|`
  `change_requests_tasks_and_objects|problems_tasks_and_objects`

  変更依頼クエリによって検出された変更依頼はその添付と関連タスク、および各タスクの関連オブジェクトと共に含まれます。

転送セットの直接メンバーとして明示的に追加された変更依頼、タスク、および他のオブジェクトはこのオプションの影響を受けません。

`-cumcrsc|-cumulative|-cumulative_change_request_scope`

指定した転送セットの変更依頼範囲を累積に修正するよう指定します。転送セットの変更依頼範囲とクエリは、各生成操作または生成プレビュー操作時に評価されます。ただし、`-cumulative` を指定した場合、現在のクエリで検出されなくても、以前のクエリで検出された古いメンバーは永遠に削除されません。つまり、変更依頼の間接（クエリベース）メンバーシップは、追加されて累積されます。

`-cumrsc|-cumulative_release_scope`

指定した転送セットのリリース範囲を累積に指定します。転送セットのリリース範囲とクエリは、各生成操作または生成プレビュー操作時に評価されます。ただし、`-cumulative_release_scope` を指定した場合、現在のクエリで検出されなくても、以前のクエリで検出された古いメンバーは永遠に削除されません。つまり、リリースの間接（クエリベース）メンバーシップは、追加されるのみで、累積されていきます。

`-dir|-directory` *`generate_dir`*

指定した転送セットの生成ディレクトリを修正するよう指定します。生成ディレクトリを使用して `direct` 以外の転送モードについて生成される転送パッケージを準備します。*`generate_dir`* 値はサーバー パスを表します。UNIX サーバーでは絶対パス、Windows サーバーでは UNC パスを使用します。空白文字列を使用すると Rational Synergy はデータベース パスの下の **dcm/generate** ディレクトリの下にあるデフォルトの生成ディレクトリを使用します。これはデフォルト設定です。

`-email` *`email_address`*

指定した転送セットの電子メール アドレスを修正するよう指定します。このオプションは、転送セットの生成、受取りまたは転送の後に、電子メールを受け取る 1 人または複数の電子メール アドレスを示します。

アドレスを空白またはカンマで区切ることにより、転送セットの複数の電子メール受信者を定義できます。電子メール リストを定義したい場合は、自分のメール サーバーの機能を使用して、電子メールの別名または配布先リストを設定できます。この操作方法については、使用するメール サーバーとオペレーティング システムを調べてください。

`-ep|-email_policy` *`value`*

指定した転送セットの電子メール ポリシーを修正するよう指定します。このオプションは、以下の電子メール ポリシーをサポートします。

- `Transfer`：デスティネーション データベースに空でないパッケージを転送されたときのみ、電子メール メッセージが送られるよう指定します。DCM 生成操作が実行されたときに含まれるオブジェクトがない場合、メッセージは送出されません。
- `Generate`：空でない転送パッケージが生成または転送されたときに、電子メール メッセージが送られるよう指定します。DCM 生成操作が実行されたときに含まれるオブジェクトがない場合、メッセージは送出されません。
- `Always`：空ではない転送パッケージを生成または転送するときは、電子メール メッセージが必ず送出されるよう指定します。DCM 生成操作の実行後に含まれるオブジェクトがない場合、あるいはデスティネーション データベースに自動配信しないパッケージを生成する場合が、これに当てはまります。

`-exclude_db_info`

指定した転送セットから生成した転送パッケージから、データベース定義に関する情報を除外するよう指定します。

`-exclude_imported_objects`

指定した転送セットから生成した転送パッケージから、他のデータベースで作成したオブジェクトを除外するよう指定します。

`-exclude_nct|-exclude_non_completed_tasks`

指定した転送セットから生成した転送パッケージから、完了していないタスクを除外するよう指定します。

`-exclude_products`

指定した転送セットから生成した転送パッケージから、製品を除外するよう指定します。

`-exclude_types type1,type2,...`

指定した転送セットから生成した転送パッケージから、指定したタイプのオブジェクトを除外するよう指定します。この値は、ゼロ個以上のタイプ名をカンマとオプションの空白で区切ったリストです。

`-exclude_typedefs`

指定した転送セットから生成した転送パッケージから、タイプ定義を除外するよう指定します。

`-ferp`

指定した転送セットのメンバーであるプロジェクトの更新プロパティを完全に展開するよう指定します。このオプションを指定した結果は以下のとおりです。

- メンバーであるタスクに関連するすべてのオブジェクトは、プロジェクト階層のメンバーではない場合でも含まれる。
- プロジェクトの更新プロパティ内のすべてのフォルダ、タスク、およびベースライン プロジェクトは、静的状態のプロジェクトの場合にも含まれる。
- 各ベースラインプロジェクトのすべてのサブプロジェクトは、プロジェクト階層の更新プロパティで直接使用されていない場合でも含まれる。

これにより転送セットの間接メンバーの数が大幅に増え、間接メンバーを計算する時間が長くなります。詳細については、『Rational Synergy Distributed』の「リコンフィギュア プロパティの全展開」を参照してください。

`-ib|-include_baselines`

指定した転送セットのメンバーであるオブジェクトに関連するベースラインを含めるよう指定します。このオプションの使用には注意が必要です。間違って使用すると、ベースラインの一部のみが複製されることになります。詳細については、『Rational Synergy Distributed』の「関連ベースラインの取り入れ」を参照してください。

`-local_parallel`

指定した転送セットから生成された転送パッケージから受け取ったパラレル オブジェクト バージョンのローカル所有者に、電子メールによりパラレル通知を送信するよう指定します。

-new_ts|-new_transfer_set *new_transfer_set_name*

転送セットの新しい名前を指定します。

-nocumcrsc|-nocumulative|-nocumulative_change_request_scope

指定した転送セットの変更依頼範囲を累積なしに修正するよう指定します。転送セットの変更依頼範囲とクエリは、各生成操作または生成プレビュー操作時に評価されます。このオプションを使用した場合、以前のクエリで検出された古いメンバーで現在のクエリで検出されなかったものは、転送セットの間接（クエリベース）メンバーなので削除されます。

-nocumrsc|-nocumulative_release_scope

指定した転送セットのリリース範囲を累積なしに指定します。転送セットのリリース範囲とクエリは、各生成または生成プレビュー操作時に評価されます。このオプションを使用した場合、以前のクエリで検出された古いメンバーで現在のクエリで検出されなかったものは、転送セットの間接（クエリベース）メンバーなので削除されます。

-noemail

指定した転送セットの生成、受取り、または転送後に電子メールを送信しないよう指定します。

-noexclude_db_info

指定した転送セットから生成した転送パッケージにデータベース定義に関する情報を含めるよう指定します。

-noexclude_imported_objects

指定した転送セットから生成した転送パッケージに他のデータベースで作成したオブジェクトを含めるよう指定します。

-noexclude_nct|-exclude_non_completed_tasks

指定した転送セットから生成した転送パッケージに完了していないタスクを含めるよう指定します。

-noexclude_products

指定した転送セットから生成した転送パッケージに製品を含めるよう指定します。

`-noexclude_typedefs`

指定した転送セットから生成した転送パッケージにタイプ定義を含めるよう指定します。

`-noferp`

転送セットのメンバーであるプロジェクトの更新プロパティを完全に展開しないよう指定します。このオプションを指定した結果は以下のとおりです。

- プロジェクトの更新プロパティ内にあるタスクの関連オブジェクトは、プロジェクト階層のメンバーである場合にのみ含まれる。
- 静的状態のプロジェクトのフォルダ、タスク、およびベースライン プロジェクトは含まれない。
- ベースライン　プロジェクトのサブプロジェクトは、プロジェクトの更新プロパティで直接使用されている場合でも含まれる。

これはデフォルトです。詳細については、『Rational Synergy Distributed』の「リコンフィギュア プロパティの全展開」を参照してください。

`-noib|-noinclude_baselines`

指定した転送セットのメンバーであるオブジェクトに関連するベースラインを自動的に含めないよう指定します。

`-nolocal_parallel`

指定した転送セットから生成された転送パッケージから受け取ったパラレル オブジェクト バージョンのローカル所有者に、電子メールによりパラレル通知を送信しないよう指定します。

`-rq|-release_query` *`release_query`*

指定した転送セットのリリース範囲と一緒に使用するリリース クエリを修正するよう指定します。*`release_query`* は、空白文字列または有効なクエリ式のいずれかです。空白文字列は、すべてのリリースをクエリすることを意味します。リリース クエリは、リリース範囲を `none` 以外の値に設定した場合にのみ使用できます。

`-rsc|-release_scope` *`value`*

指定した転送セットのリリース範囲を指定します。以下の範囲を使用します。

- `none`

  リリース定義は自動的に含まれません。

- `releases`

  リリース クエリによって検出されたリリース定義は、間接クエリ メンバーとして自動的に含まれます。ただし、対応するプロセス ルールとフォルダ テンプレートは自動的に含まれません。

- `releases_templates|releases_and_templates |releases and templates`

  リリース クエリによって検出されたリリース定義は、間接クエリ メンバーとして自動的に含まれます。各リリースに対して、そのプロセス ルールとプロセス ルールが使用する任意のユーザー定義フォルダ テンプレートも間接クエリ メンバーとして自動的に含まれます。

`transfer_set_spec...`

修正する転送セットを指定します。詳細については、転送セットの指定を参照してください。

例

- 転送セット **client** と **server** の変更依頼範囲とリリース範囲を変更する。

```
ccm dcm -modify -ts -crsc crs -rsc releases_and_templates client server
```

関連トピック

- 転送セットへのオブジェクトの追加
- 転送セットの作成
- 転送セットの削除
- 転送セットの修正
- 転送セットの間接変更依頼メンバーの再計算
- 転送セットの間接リリース メンバーの再計算
- 転送セットのメンバーの再計算
- 転送セットからのオブジェクトの削除
- 転送セット情報の表示
- 転送セット メンバーの表示
- 転送セットの表示
- 生成パッケージの転送

DCM 設定の修正

このサブコマンドにより、どの DCM 設定を変更するかを示します。このコマンドに関連するオプションを少なくとも 1 つ指定する必要があります。

```
ccm dcm -m|-modify -settings
        [-desc|-description description_of_database]
        [-location location]
        [-admin_info admin_info]
        [-default_add_history|-nodefault_add_history]
        [-default_include_baselines|-nodefault_include_baselines]
        [-ignore_maintain_wa|-noignore_maintain_wa]
        [-update_db_info|-noupdate_db_info]
        [-keep_typedefs|-nokeep_typedefs]
        [-event_log_size log_size]
        [-parallel_checking parallel_check_keyword]
        [-update_releases release_action_keyword]
        [-add_receive_control_transition transition]
        [-remove_receive_control_transition transition]
        [-no_of_generate_times generate_times]
        [-no_of_old_generate_times old_generate_times]
        [-old_generate_time_resolution old_generate_resolution]
        [-update_rft|-noupdate_rft]
```

`-add_receive_control_transition` *`transition`*

有効な状態遷移を Receive Control Transitions リストに追加します。追加された状態遷移は、現在のデータベースで管理されているオブジェクトを受け取るときに可能となります。*`transition`* に指定する値は以下の形式となります。

`from_state`:*`to_state`*

ここで：

`from_state` は有効な状態であること。

`to_state` は、指定した *`from_state`* からの遷移について、有効な状態であること。

`-admin_info` *`admin_info`*

データベース アドミニストレータの連絡先情報を指定します。*`admin_info`* 値には、復帰改行文字を含まない任意の文字列を指定できます。たとえば、電話番号や電子メール アドレスは有効な項目です。

`-default_include_baselines`

転送セット メンバーに関連付けられているベースラインが、転送セットに含まれるように指定します。

`-desc|-description` *`description`*

このデータベースの説明を指定します。*`description`* には、復帰改行文字を含めることはできません。

`-default_add_history`

転送セットに追加されるオブジェクトが、その直前バージョンと一緒に追加されるよう指定します。

`-event_log_size` *`log_size`*

イベント ログの最大エントリ数を指定します。*`log_size`* オプションには、ゼロ以外の任意の正の整数を指定できます。イベント ログ リストが *`log_size`* に指定した値に到達した場合、最も古いエントリから取り除かれ、新しいエントリが追加されます。

`-ignore_maintain_wa`

インポートまたは XML インポート（DCM 受取り操作時など）で作成するプロジェクトは、管理ワークエリアなしで作成されることを指定します。`ccm wa -wa` コマンドを使用してプロジェクトのワークエリアの管理を有効に戻すことができます。

`-keep_typedefs`

受取り操作の完了後にタイプ定義を維持するよう指定します。

`-location` *`location`*

データベースの地理的な位置を指定します（例、`Irvine, California`）。位置には、復帰改行文字を含まない任意の文字列を使用できます。

`-no_of_generate_times` *`generate_times`*

DCM が格納する生成時刻の数を指定します。*`generate_times`* に指定する値は、古い生成時刻の数に等しいかそれより大きいゼロ以外の正の整数です。

`-no_of_old_generate_times` *`old_generate_times`*

DCM が格納する古い生成時刻の数を指定します。*`old_generate_times`* に指定する値は、古い生成時刻の数に等しいかそれより小さいゼロ以外の正の整数です。

`-nodefault_add_history`

転送セットに追加されるオブジェクトが、その直前バージョンなしに追加されるよう指定します。これはデフォルト設定です。

`-nodefault_include_baselines`

転送セット メンバーに関連付けられているベースラインが、転送セットから除外されるよう指定します。これはデフォルト設定です。

`-noignore_maintain_wa`

インポートまたは XML インポートによって作成されたプロジェクトが管理ワークエリアを持つよう指定します。これはデフォルト設定です。

`-nokeep_typedefs`

受取り操作の完了後にタイプ定義を削除するよう指定します。これはデフォルト設定です。

`-noupdate_db_info`

受取り操作時に、DCM データベース情報を更新しないよう指定します。

`-noupdate_rtf`

受取り操作時に、プロセス ルールと関連するフォルダ テンプレートを更新しないよう指定します。

`-old_generate_time_resolution` *`old_generate_resolution`*

古い生成時刻の間隔（日数単位）を指定します。*`old_generate_resolution`* に指定する値は、ゼロ以外の任意の正の整数です。

`-parallel_checking (none|created|updated)`

DCM 受取り操作時に実行するパラレル チェックのタイプを指定します。指定できる値は `none`、`created`、`updated`、大文字と小文字が区別されます。

`-remove_receive_control_transition` *`transition`*

指定した *`transition`* を Receive Control Transitions リストから削除します。削除された状態遷移は、現在のデータベースで管理されているオブジェクトを受け取るときには可能となりません。*`transition`* に指定する値は以下の形式となります。

`from_state``:`*`to_state`*

ここで：

`from_state` は Receive Control Transitions リストで有効な状態であること。

`to_state` は、指定した *`from_state`* からの遷移について、有効な状態であること。

`-update_db_info`

受取り操作時に、DCM データベース情報が更新される可能性があることを指定します。この情報は、DCM Information ファイルにあるデータで更新されます。これはデフォルト設定です。

`-update_rtf`

受取り操作時に、プロセス ルールと関連するフォルダ テンプレートが更新される可能性があることを指定します。これはデフォルト設定です。

`-update_releases (none|active|inactive)`

DCM 受取り時に、リリース定義を更新する方法を指定します。指定できる値は `none`、`active`、`inactive`、大文字と小文字が区別されます。

- `none`

  リリース定義が作成も更新もされないよう指定します。

- `active`

  DCM 転送パッケージにリリース定義が含まれる場合、受取り側データベースに存在するリリース定義が更新されますが、新しいアクティブ リリース定義のみが作成されます。DCM 転送パッケージに（Telelogic Synergy 6.2 またはそれ以前からの）リリース テーブル情報のみが含まれる場合、リリースのリリース定義はアクティブ リリースとして作成されます。

- `inactive`

  DCM 転送パッケージにリリース定義が含まれる場合、これらは受取り側データベース内で作成または更新されます。DCM 転送パッケージに（Telelogic Synergy 6.2 またはそれ以前からの）リリース テーブル情報のみが含まれる場合、リリースのリリース定義は非アクティブ リリースとして作成されます。

例

- イベント ログ サイズと古い生成時刻の数の設定を変更する。

  ```
  ccm dcm -m -settings -event_log_size 25 -no_of_old_generate_times 2
  ```

関連トピック

- 設定の表示
- 最新生成時刻の表示
- DCM イベントの概要の表示
- 転送セット情報の表示

パッケージの受取り

このサブコマンドにより、このデータベース用に生成された転送パッケージ、あるいはブロードキャスト パッケージまたはオフライン保存パッケージである転送パッケージを受け取ります。生成データベースまたは転送セット、あるいは生成データベースと転送セットを指定して、どのパッケージを受け取るかを指定できます。デフォルトでは、任意の転送セットと共に任意の他のデータベースによってこのデータベース用に生成されたパッケージを受け取ります。パッケージは、生成されたときと同じ順序で受け取られます。

このサブコマンドは、アドミニストレータが実行できます。

```
ccm dcm -rec|-receive [-dbid|-database_id database_spec]
        [-ts|-transfer_set transfer_set_spec] [-a|-all] [-im|-ignore_missing]
        ([-wait] | [-nowait]) ([-ic|-ignore_checks] |
        [-noic|-noignore_checks]) ([-ivdc|-ignore_version_delimiter_check] |
        [-noivdc|-noignore_version_delimiter_check])
        ([-irdc|-ignore_release_delimiter_check] |
        [-noirdc|-noignore_release_delimiter_check])
        ([-itsc|-ignore_time_sync_check] |
        [-noitsc|-noignore_time_sync_check]) [-dir|-directory receive_dir]
```

`-a|-all`

Specifies to receive all transfer packages for all transfer sets. このオプションは、`-database_id` または `-transfer_set` と一緒には使用できません。

`-database_id` *`database_spec`*

指定したデータベースからのみ転送パッケージを受け取るよう指定します。*`database_spec`* には、1 つのデータベース定義を指定できます。詳細については、データベースの指定を参照してください。

`-dir|-directory` *`receive_dir`*

転送パッケージは、サーバー上の指定した *`receive_dir`* にあることを指定します。デフォルトでは、パッケージは現在のデータベース パスの下の **dcm/receive** ディレクトリから受け取ります。*`receive_dir`* は、サーバー パスを表します。UNIX サーバーでは絶対パス、Windows サーバーでは UNC パスを使用します。

`-ignore_checks`

以下のいずれかのチェックが失敗した場合、それらを無視して操作を続行するよう指定します。

- バージョン区切り文字のチェック
- リリース区切り文字のチェック
- 時刻同期のチェック

このオプションは、-ivdc|-ignore_version_delimiter_check、-irdc|-ignore_release_delimiter_check、および -itsc|-ignore_time_sync_check オプションを指定することと同じです。

`-irdc|-ignore_release_delimiter_check`

リリース区切り文字のチェックが失敗した場合、その状態を無視して操作を続行するよう指定します。

デフォルトでは、生成データベース内のリリース区切り文字が受取りデータベースのものと異なる場合、受取りは失敗します。DCM クラスタ内のすべてのデータベースは同じリリース区切り文字を使用する必要があります。

`-itsc|-ignore_time_sync_check`

転送パッケージの生成時刻が将来の時刻である場合、状態を無視して操作を続行するよう指定します。

デフォルトでは、この状態が検出されると受取りは失敗します。この状態は、パッケージを生成したコンピュータまたは受け取るコンピュータあるいはその両方のタイム ゾーンまたは時刻設定が誤っている場合に発生します。時刻を修正すると、DCM は複数のタイム ゾーンにまたがって正しく動作します。

`-ivdc|-ignore_version_delimiter_check`

バージョン区切り文字のチェックが失敗した場合、その状態を無視して操作を続行するよう指定します。

デフォルトでは、生成データベース内のバージョン区切り文字が受取りデータベースのものと異なる場合、受取りは失敗します。

`-im|-ignore_missing`

紛失転送パッケージを無視するように DCM に指示します。

> **注意！** このオプションを使用すると、空のディレクトリ エントリが作られたり、関係が失われることがあります。

`-noic|-noignore_checks`

以下のいずれかのチェックが失敗した場合、警告を出して操作を続行しないよう指定します。

- バージョン区切り文字のチェック
- リリース区切り文字のチェック
- 時刻同期のチェック

このオプションは、-noivdc|-noignore_version_delimiter_check、-noirdc|-noignore_release_delimiter_check、および -noitsc|-noignore_time_sync_check オプションを指定することと同じです。これはデフォルト設定です。

`-noirdc|-noignore_release_delimiter_check`

リリース区切り文字のチェックが失敗した場合、エラーを報告して操作を続行しないよう指定します。これはデフォルト設定です。リリース区切り文字の詳細については、release コマンドの「説明と用途」を参照してください。

`-noitsc|-noignore_time_sync_check`

転送パッケージの生成時刻が将来の時刻である場合、エラーを報告して操作を続行しないよう指定します。

デフォルトでは、この状態が検出されると受取りは失敗します。この状態は、パッケージを生成したコンピュータまたは受け取るコンピュータあるいはその両方のタイム ゾーンまたは時刻設定が誤っている場合に発生します。時刻を修正すると、DCM は複数のタイム ゾーンにまたがって正しく動作します。

`-noivdc|-noignore_version_delimiter_check`

バージョン区切り文字のチェックが失敗した場合、エラーを報告して操作を続行しないよう指定します。これはデフォルト設定です。バージョン区切り文字の制限については、DCM の制限を参照してください。

`-nowait`

受取りデータベースが他の転送パッケージの受取りを完了していなくても、受取りを実行できるようにします。

> **注意！** 複数の転送パッケージを同時にデータベースで受け取るのは安全ではありません。デフォルトでは、DCM はパッケージを 1 つずつデータベースに受け取ります。このオプションは、転送パッケージが重複オブジェクトを含まない確証がある場合にのみ使用します。たとえば、それぞ

れ異なる転送セットから生成された 2 つのパッケージに同じタスク オブジェクトが存在する場合、それらのパッケージは重複オブジェクトを持ちます。この場合、デフォルトの `-wait` を使用してください。

`-ts|-transfer_set` *`transfer_set_spec...`*

`transfer_set_spec...` は、DCM 受取りに使用する転送セットを指定します。*`transfer_set_spec`* には、1 つの転送セットを指定できます。詳細については、転送セットの指定を参照してください。

`-wait`

受取りデータベースが他の転送パッケージの受取りを完了していない場合に、受取りを実行できないようにします。この操作をキャンセルするには `Ctr. + C` キーを使用します。

これはデフォルト設定です。

例

- ソース データベースから転送パッケージを受け取る。

```
ccm dcm -receive -ts "Entire Database" -dbid USIRJA
```

関連トピック

- 転送パッケージの生成
- 転送セットの間接変更依頼メンバーの再計算
- 転送セットの間接リリース メンバーの再計算
- 転送セットのメンバーの再計算
- 最新生成時刻の表示
- 転送セット情報の表示
- 転送セット メンバーの表示
- 転送セットの表示
- 生成パッケージの転送

転送セットの間接変更依頼メンバーの再計算

このサブコマンドにより、現在の変更依頼範囲、変更依頼クエリ、および範囲が累積かどうかに基づいて、転送セットの間接変更依頼（CR）メンバーを再計算します。これは、DCM 生成を行うとき自動的に行われます。

このサブコマンドは、DCM マネージャまたはアドミニストレータが実行できます。

```
ccm dcm -recompute -crs|-change_requests|-problems
        [-dbid|-database_id database_spec]
        -ts|-transfer_set transfer_set_spec...
```

`-database_id database_spec`

転送セットの変更依頼クエリで `%to_dbid` キーワードに使用するデータベース ID を指定します。*`database_spec`* には、1 つのデータベース定義を指定できます。詳細については、データベースの指定を参照してください。

`transfer_set_spec...`

間接 CR メンバーの再計算に使用する転送セットを指定します。詳細については、転送セットの指定 を参照してください。

関連トピック

- 転送セットへのオブジェクトの追加
- 転送セットの作成
- 転送セットの削除
- 転送セットの修正
- 転送セットの間接リリース メンバーの再計算
- 転送セットのメンバーの再計算
- 転送セットからのオブジェクトの削除
- 転送セット情報の表示
- 転送セット メンバーの表示
- 転送セットの表示

転送セットの間接リリース メンバーの再計算

このサブコマンドにより、現在のリリース範囲、リリース クエリ、および範囲が累積かどうかに基づいて、転送セットの間接リリース メンバーを再計算します。これは、DCM 生成後に自動的に行われます。

このサブコマンドは、DCM マネージャまたはアドミニストレータが実行できます。

```
ccm dcm -recompute -rel|-release|-releases -ts|-transfer_set
        transfer_set_spec...
```

transfer_set_spec...

間接リリース メンバーの再計算に使用する転送セットを指定します。詳細については、転送セットの指定を参照してください。

関連トピック

- 転送セットへのオブジェクトの追加
- 転送セットの作成
- 転送セットの削除
- 転送セットの修正
- 転送セットのメンバーの再計算
- 転送セットからのオブジェクトの削除
- 転送セット情報の表示
- 転送セット メンバーの表示
- 転送セットの表示

転送セットのメンバーの再計算

このサブコマンドにより、転送セットの現在の直接メンバーに基づいて転送セットの間接メンバーを再計算します。これは、DCM 生成時に自動的に行われます。

このサブコマンドは、DCM マネージャまたはアドミニストレータが実行できます。

```
ccm dcm -recompute -ts|-transfer_set transfer_set_spec...
```

transfer_set_spec...

指定した転送セットの再計算を指定します。詳細については、転送セットの指定を参照してください。

関連トピック

- 転送セットへのオブジェクトの追加
- 転送セットの作成
- 転送セットの削除
- 転送セットの修正
- 転送セットの間接変更依頼メンバーの再計算
- 転送セットの間接リリース メンバーの再計算
- 転送セットからのオブジェクトの削除
- 転送セット情報の表示
- 転送セット メンバーの表示
- 転送セットの表示

転送セットからのオブジェクトの削除

このサブコマンドにより、指定したオブジェクトを指定した転送セットから削除します。オブジェクトが転送セットの直接メンバーであれば、削除できます。つまり、転送セットに明示的に追加したオブジェクトは削除できます。オブジェクトが履歴メンバーならば、そのオブジェクトのすべてのバージョンが削除されます。オブジェクトを削除しても、直ちに転送セットの間接メンバーの再計算は行われません。オブジェクトを削除して間接メンバーを見るには（現在のメンバーをベースに）、転送セットのメンバーを再計算します（転送セットのメンバーの再計算を参照してください）。

このサブコマンドは、DCM マネージャまたはアドミニストレータが実行できます。

```
ccm dcm -remove -ts|-transfer_set transfer_set_spec object_spec...
```

object_spec...

転送セットから削除するオブジェクトを指定します。詳細については、オブジェクトの指定を参照してください。

`-ts|-transfer_set` *transfer_set_spec...*

指定した転送セットからオブジェクトを削除するよう指定します。*transfer_set_spec* には、1 つの転送セットを指定できます。詳細については、転送セットの指定を参照してください。

関連トピック

- 転送セットへのオブジェクトの追加
- 転送セットの作成
- 転送セットの削除
- 転送セットの修正
- 転送セットのメンバーの再計算
- 転送セットからのオブジェクトの削除
- 転送セット情報の表示
- 転送セット メンバーの表示
- 転送セットの表示

データベース定義情報の表示

このサブコマンドにより、指定したデータベース定義の情報を表示します。

```
ccm dcm -s|-sh|-show -dbid|-database_id database_spec...
```

database_spec

表示するデータベース定義を指定します。詳細については、データベースの指定を参照してください。

関連トピック

- データベース定義の作成
- データベース定義の削除
- データベース定義の修正
- データベース定義情報の表示
- DCM プロパティの表示

パッケージの表示

このサブコマンドにより、指定した条件に従ってパッケージのサマリを表示するか、パッケージに含まれるオブジェクトのプレビューを表示します。パッケージは、任意のデータベースによって、任意の転送セットで、save_offline_and_delete コマンドを使用して生成できます。

同じ条件で異なる時刻に複数のパッケージが生成された場合、最も古いパッケージが先頭に、最後に生成されたパッケージ（最新）が最後に表示されます。

```
ccm dcm -s|-sh|-show -packages [-members]
        [-from_dbid|-from_database_id database_spec]
        [-to_dbid|-to_database_id database_spec]
        [-ts|-transfer_set transfer_set_spec]
        ([-gen|-generate] | [-rec|-receive] |
        [-soad|-save_offline_and_delete] | [-serverdir server_path])
```

`-from_dbid|-from_database_id database_spec`

パッケージの生成元のデスティネーション データベースを指定します。*database_spec* には、1 つのデータベース定義を指定できます。詳細については、データベースの指定を参照してください。

`-gen|-generate`

表示するパッケージをサーバー生成ディレクトリで検索するよう指定します。

`-members`

出力に DCM プレビュー ファイルとパッケージ サマリの両方の情報を含めるよう指定します。

`-rec|-receive`

サーバー受取りディレクトリでパッケージを検索するよう指定します。

`-serverdir server_path`

パッケージを含むファイル システム内のサーバー ディレクトリへのパスを指定します。

`-soad|-save_offline_and_delete`

soad パッケージのデフォルト サーバー ディレクトリからパッケージを検索するよう指定します。

`-to_dbid|-to_database_id` *`database_spec`*

パッケージの生成先のデスティネーション データベースを指定します。*`database_spec`* には、1 つのデータベース定義を指定できます。詳細については、データベースの指定を参照してください。

`-ts|-transfer_set` *`transfer_set_spec`*

パッケージの生成に使用した転送セット名でパッケージのフィルタリングを行うよう指定します。*`transfer_set_spec`* には、1 つの転送セットを指定できます。詳細については、転送セットの指定を参照してください。

例

- サーバー受取りディレクトリ内の、ukweb データベースから生成されたすべてのパッケージを表示する。

  ```
  ccm dcm -show -packages -receive -from_dbid ukweb
  ```

関連トピック

- パッケージの削除
- 生成パッケージの転送

DCM プロパティの表示

このサブコマンドにより、指定したオブジェクトの DCM プロパティを表示します。オブジェクトが任意の転送セットのメンバーであるかどうか、およびそれらは最後にデータベースへ送られた後に修正されたかどうかを示します。

```
ccm dcm -s|-sh|-show -prop|-properties object_spec...
```

object_spec...

指定したオブジェクトの DCM プロパティを表示するよう指定します。詳細については、オブジェクトの指定を参照してください。

例

- 前のクエリで検出された 2 番目の DCM プロパティを表示する。

```
ccm dcm -show -properties @2
Object foo.txt-2:ascii:test71#1

Transfer Sets
Member of  Member Type  Member Since
preptest   indirect     09/03/09 10:06

Transfer Status
Database ID  Modified since last sent
b7int        TRUE
```

関連トピック

- 最新生成時刻の表示
- 生成パッケージの転送

設定の表示

このサブコマンドは、DCM 設定を一覧表示します。出力には以下の項目が含まれます。

- 現在のデータベースの説明。
- 現在のデータベースの位置。
- 現在のデータベースの DCM 管理の責任を負う担当者に関する情報。
- DCM 追加操作に対する履歴設定のデフォルト（.ini ファイルが優先します）。
- イベント ログのエントリ数。
- `maintain_wa` をデフォルトで無効とするかどうか。
- 受取り操作時にリリース定義がどのように更新されるか。
- タイプ定義を受取り操作後に維持するかどうか。
- 受取り操作時にパラレル検査がどのように行われるか。
- 状態遷移が存在する場合、現在のデータベースで管理されるオブジェクトについてどの状態遷移を受け取ることができるか。
- 受取り操作時に、DCM データベース情報を更新するかどうか。
- 受取り操作時に、プロセス ルールを更新するかどうか。
- DCM が格納する生成時刻の数。
- DCM が格納する古い生成時刻の数。
- 古い生成時刻の間隔（日数単位）。

```
ccm dcm -show -settings
```

関連トピック

- 転送セットの修正
- 最新生成時刻の表示

データベース ID の表示

このサブコマンドに、既知データベース定義の DCM データベース ID を表示します。

```
ccm dcm -s|-sh|-show -dbid|-database_id -a|-all [-f|-format format]
       [-nf|-noformat] ([-ch|-column_header] | [-nch|-nocolumn_header])
       [-sep|-separator separator] ([-sby|-sortby sortspec] |
       [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat] [-u|-unnumbered]
```

`-a|-all`

既知データベース定義のすべての DCM データベース ID を表示するよう指定します。

`-ch|-column_header`

出力フォーマットでカラム ヘッダーを使用するよう指定します。詳細については、-ch|-column_headers を参照してください。

`-f|-format format`

コマンドの出力フォーマットを指定します。詳細については、-f|-format を参照してください。

`-gby|-groupby groupformat`

コマンドの出力をグループ化する方法を指定します。詳細については、-gby|-groupby を参照してください。

`-nch|-nocolumn_header`

出力フォーマットでカラム ヘッダーを使用しないよう指定します。詳細については、-nch|-nocolumn_headers を参照してください。

`-nf|-noformat`

カラム配置を使用しないよう指定します。詳細については、-nf|-noformat を参照してください。

`-ns|-no_sort`

コマンドの出力をソートしないよう指定します。詳細については、-ns|-nosort を参照してください。

`-sep|-separator` *separator*

異なる区切り文字を指定します。詳細については、-sep|-separator を参照してください。

`-sby|-sortby` *sortspec*

コマンドの出力をソートする方法を指定します。詳細については、-sby|-sortby を参照してください。

`-u|-unnumbered`

コマンドの出力の自動番号付けを抑止します（すなわち、出力に番号付けがされません）。詳細については、-u|-unnumbered を参照してください。

関連トピック

- 現在の DCM データベース ID の表示
- 生成パッケージの転送

DCM イベントの概要の表示

各イベントに対して DCM は以下の種類の情報を取り込みます。

- イベント情報

  イベント詳細の概要です。DCM の生成、転送、および受取り操作では、この情報は送信される電子メールに含まれる情報と同じです。

- メッセージ

  操作に対して表示されたすべてのメッセージを取り込みます。操作が失敗した場合、これらのメッセージを調べて失敗の詳細情報を確認してください。

```
ccm dcm -s|-sh|-show -el|-event_log [-dbid|-database_id database_spec
        [-ts|-transfer_set transfer_set_spec] [-f|-format format]
        [-nf|-noformat] ([-ch|-column_header] | [-nch|-nocolumn_header])
        [-sep|-separator separator] ([-sby|-sortby sortspec] |
        [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat]
```

`-ch|-column_header`

出力フォーマットでカラム ヘッダーを使用するよう指定します。詳細については、-ch|-column_headers を参照してください。

`-database_id` *`database_spec`*

指定したデータベース定義と転送セットのイベント ログ情報を表示するために使用するデータベース定義を指定します。*`database_spec`* には、1 つのデータベース定義を指定できます。詳細については、データベースの指定を参照してください。

`-el|-event_log`

取り込んだメッセージを、指定したイベントのイベント ログに表示するよう指定します。

`-f|-format` *`format`*

コマンドの出力フォーマットを指定します。詳細については、-f|-format を参照してください。

`-gby|-groupby` *`groupformat`*

コマンドの出力をグループ化する方法を指定します。詳細については、-gby|-groupby を参照してください。

`-nch|-nocolumn_header`

出力フォーマットでカラム ヘッダーを使用しないよう指定します。詳細については、-nch|-nocolumn_headers を参照してください。

`-nf|-noformat`

カラム配置を使用しないよう指定します。詳細については、-nf|-noformat を参照してください。

`-ns|-no_sort`

コマンドの出力をソートしないよう指定します。詳細については、-ns|-nosort を参照してください。

`-sep|-separator` *`separator`*

異なる区切り文字を指定します。詳細については、-sep|-separator を参照してください。

`-sby|-sortby` *`sortspec`*

コマンドの出力をソートする方法を指定します。詳細については、-sby|-sortby を参照してください。

`transfer_set_spec`

使用する転送セットを指定します。*`transfer_set_spec`* には、1 つの転送セットを指定できます。詳細については、転送セットの指定を参照してください。

例

- データベース `sdg1` の DCM イベントを表示する。

  ```
  ccm dcm -show -event_log -dbid sdg1
  ```

関連トピック

- 指定 DCM イベントの表示

指定 DCM イベントの表示

このサブコマンドにより、指定した DCM イベントのイベント ログ情報を表示します。

```
ccm dcm -s|-sh|-show -el|-event_log -index number [-info]
       [-messages] [-f|-format format] [-nf|-noformat]
       ([-ch|-column_header] | [-nch|-nocolumn_header])
       [-sep|-separator separator] ([-sby|-sortby sortspec] |
       [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat]
```

`-ch|-column_header`

出力フォーマットでカラム ヘッダーを使用するよう指定します。詳細については、-ch|-column_headers を参照してください。

`-f|-format format`

コマンドの出力フォーマットを指定します。詳細については、-f|-format を参照してください。

`-gby|-groupby groupformat`

コマンドの出力をグループ化する方法を指定します。詳細については、-gby|-groupby を参照してください。

`-index number`

イベントのインデックス番号を指定します。インデックス番号は、イベント エントリを識別する一意の番号です。ロギングされる最初のイベントはインデックス番号 1、ロギングされる 2 番目のイベントはインデックス番号 2、というように割り当てられます。

ゼロ以外の正のインデックス番号は、DCM イベント ログに含まれる特定のイベントの絶対インデックスです。ゼロまたは負のインデックス番号は相対インデックスを表します。0 は最後のエントリを、負数は最後のエントリから数えたエントリ数を意味します。

`-info`

サマリ ファイルに格納されている指定したイベントの情報を表示するよう指定します。

`-messages`

指定したイベントのログ ファイルから取り込んだメッセージを表示するよう指定します。

`-nch|-nocolumn_header`

出力フォーマットでカラム ヘッダーを使用しないよう指定します。詳細については、-nch|-nocolumn_headers を参照してください。

`-nf|-noformat`

カラム配置を使用しないよう指定します。詳細については、-nf|-noformat を参照してください。

`-ns|-no_sort`

コマンドの出力をソートしないよう指定します。詳細については、-ns|-nosort を参照してください。

`-sep|-separator` *`separator`*

異なる区切り文字を指定します。詳細については、-sep|-separator を参照してください。

`-sby|-sortby` *`sortspec`*

コマンドの出力をソートする方法を指定します。詳細については、-sby|-sortby を参照してください。

`-ts|-transfer_set` *`transfer_set_spec`*

イベント概要を転送セット名でフィルタリングするよう指定します。*`transfer_set_spec`* には、1 つの転送セットを指定できます。詳細については、転送セットの指定を参照してください。

関連トピック

- DCM イベントの概要の表示

最新生成時刻の表示

このサブコマンドにより、指定したデータベース定義と転送セットに対して最後に DCM 生成を実行した時刻を表示します。

```
ccm dcm -s|-sh|-show -ts|-transfer_set transfer_set_spec
        -dbid|-database_id -a|-all
ccm dcm -s|-sh|-show -ts|-transfer_set transfer_set_spec
        -dbid|-database_id database_spec
```

`-a|-all`

すべてのデータベースへの指定した転送セットの生成時刻を表示することを指定します。

`-database_id` *`database_spec`*

指定したデータベース定義と転送セットの最新 DCM 生成時刻を表示するために使用するデータベース定義を指定します。*`database_spec`* には、1 つのデータベース定義を指定できます。詳細については、データベースの指定を参照してください。

`transfer_set_spec`

使用する転送セットを指定します。*`transfer_set_spec`* には、1 つの転送セットを指定できます。詳細については、転送セットの指定を参照してください。

関連トピック

- 転送セットの修正
- 転送セット情報の表示
- 転送セット メンバーの表示
- 転送セットの表示

現在の DCM データベース ID の表示

このサブコマンドにより、現在のデータベースの DCM データベース ID を表示します。

```
ccm dcm -s|-sh|-show -dbid|-database_id
```

関連トピック

- データベース ID の表示
- 生成パッケージの転送

受取りロックの表示

受取りロックは、データベース内で 1 つの DCM 受取り操作のみ行われることを確実にする機能です。このサブコマンドにより、データベースの受取りロックが設定されているかどうかを表示します。設定されている場合、ロックが設定されている現在実行中の受取りの情報が表示されます。

```
ccm dcm -show -receive_lock
```

関連トピック

- パッケージの受取り
- 転送セットの表示

転送セット情報の表示

このサブコマンドにより、指定した転送セットの情報を表示します。

```
ccm dcm -s|-sh|-show -ts|-transfer_set transfer_set_spec...
```

transfer_set_spec...

表示する転送セットを指定します。*transfer_set_spec* には、1 つの転送セットを指定できます。詳細については、転送セットの指定を参照してください。

関連トピック

- 転送セットへのオブジェクトの追加
- 転送セットの作成
- 転送セットの削除
- 転送セットの修正
- 転送セットのメンバーの再計算
- 転送セットからのオブジェクトの削除
- 転送セット情報の表示
- 転送セット メンバーの表示
- 転送セットの表示

転送セット メンバーの表示

このサブコマンドにより、指定した転送セットの直接メンバーと、オプションで間接メンバーを表示します。

```
ccm dcm -s|-sh|-show -ts|-transfer_set -members (direct | all)
       [-f|-format format] [-nf|-noformat] ([-ch|-column_header] |
       [-nch|-nocolumn_header]) [-sep|-separator separator]
       ([-sby|-sortby sortspec] | [-ns|-nosort|-no_sort])
       [-gby|-groupby groupformat] [-u|-unnumbered] transfer_set_spec...
```

-ch|-column_header

-f|-format format

-gby|-groupby groupformat

-members (direct|all)

表示するメンバーの種類を指定します。-members direct を指定すると、直接メンバーのみが表示されます。all を指定すると、直接メンバーと間接メンバーが表示されます。直接メンバーと間接メンバーの詳細については、『Rational Synergy Distributed』の「転送セットへのオブジェクトの追加」を参照してください。

-nch|-nocolumn_header

-nf|-noformat

-ns|-no_sort

-sep|-separator separator

-sby|-sortby sortspec

-u|-unnumbered

関連トピック

- 転送セットへのオブジェクトの追加
- 転送セットの作成
- 転送セットの削除
- 転送セットの修正

- 転送セットのメンバーの再計算
- 転送セットからのオブジェクトの削除
- 転送セット情報の表示
- 転送セット メンバーの表示
- 転送セットの表示

転送セットの表示

このサブコマンドにより、定義されているすべての転送セットの名前を表示します。

```
ccm dcm -s|-sh|-show -ts|-transfer_set -a|-all [-f|-format format]
       [-nf|-noformat] ([-ch|-column_header] | [-nch|-nocolumn_header])
       [-sep|-separator separator] ([-sby|-sortby sortspec] |
       [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat] [-u|-unnumbered]
```

`-a|-all`

指定したデータベースの定義されているすべての転送セットのすべての名前を表示するよう指定します。

`-ch|-column_header`

`-f|-format format`

`-gby|-groupby groupformat`

`-nch|-nocolumn_header`

`-nf|-noformat`

`-ns|-no_sort`

`-sep|-separator separator`

`-sby|-sortby sortspec`

`-u|-unnumbered`

関連トピック

- 転送セットへのオブジェクトの追加
- 転送セットの作成
- 転送セットの削除
- 転送セットの修正
- 転送セットのメンバーの再計算
- 転送セットからのオブジェクトの削除

- 転送セット情報の表示
- 転送セット メンバーの表示

生成パッケージの転送

このサブコマンドにより、このデータベースから生成されたパッケージをそのデスティネーション データベースに転送します。転送するパッケージは、デスティネーション データベース ID または転送セット名で指定できます。

このサブコマンドは、DCM マネージャまたはアドミニストレータが実行できます。

```
ccm dcm -trn|-transfer [-dbid|-database_id database_spec]
        ([-ts|-transfer_set transfer_set_spec] | [-a|-all])
```

`-a|-all`

すべての転送セットを使用して生成されたパッケージを指定したデスティネーション データベースへ転送することを指定します。

`-database_id` *`database_spec`*

指定したデータベースの最後の生成時刻を表示するよう指定します。*`database_spec`* には、1 つのデータベース定義を指定できます。詳細については、データベースの指定を参照してください。

`-ts|-transfer_set` *`transfer_set_spec`*`...`

指定した転送セットから生成したパッケージのみ転送するよう指定します。*`transfer_set_spec`* には、1 つの転送セットを指定できます。詳細については、転送セットの指定を参照してください。

関連トピック

- 転送パッケージの生成
- パッケージの受取り
- DCM プロパティの表示
- 最新生成時刻の表示

説明と用途

`dcm` コマンドにより、転送パッケージの生成、転送パッケージのデスティネーション データベースへの送出、転送パッケージの受取り、転送セットへのオブジェクトの追加を行います。`dcm` コマンドのオプションにより、これらの操作の 1 つまたは複数を実行できます。

`-add`、`-create`、`-gen`、`-modify`、`-delete`、`-remove` オプションを使用するには、DCM マネージャである必要があります。`-rec`、`-init`、`-change`、`-modify`、`-settings` オプションを使用するには、*ccm_admin* ロールを持っている必要があります。

delete コマンド

詳細については、説明と用途を参照してください。`delete` コマンドは、データベースからのオブジェクトの削除サブコマンドをサポートします。

データベースからのオブジェクトの削除

`delete` コマンドにより、ディレクトリとデータベースから特定バージョンのファイル、ディレクトリまたはプロジェクトを削除します。また、コマンドラインまたは GUI からプロジェクト階層を削除できます。

```
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`

プロジェクト階層全体を削除します。このオプションは `-recurse` オプションと一緒にのみ使用します。

`object_spec`

削除するオブジェクトを指定します。

`-p|-project`

プロジェクト形式のコマンドを指定します。

`project_spec`

削除するプロジェクトを指定します。詳細については、プロジェクトの指定を参照してください。

`-r|-recurse`

削除操作をディレクトリまたはサブプロジェクトにも再帰的に行うかどうかを指定します。オブジェクトがプロジェクトである場合、再帰的なサブプロジェクトも削除されます。オブジェクトがディレクトリである場合、再帰的な子ディレクトリも削除されます。その他のタイプのオブジェクトでは、このオプションは無効です。

このオプションを使用して階層的にオブジェクトを削除する場合、以下が適用されます。

- プロジェクト オブジェクトに対しては、`-recurse` は `-scope project_and_non-project_members` を指定することと同じで、プロジェクトと、プロジェクト内のサブプロジェクト以外のメンバーを削除する。

- プロジェクト オブジェクトに対しては、-recurse -hierarchy は -scope entire_project_hierarchy を指定することと同じで、プロジェクトと、プロジェクト内のサブプロジェクトを含む再帰的なメンバーを削除する。
- ディレクトリ オブジェクトに対しては、-recurse は -scope directory_and_non-project_members を指定することと同じで、ディレクトリと、ディレクトリ下のサブプロジェクト以外の再帰的な子ディレクトリを削除する。
- ディレクトリ オブジェクトに対しては、-recurse -hierarchy は -scope entire_directory_hierarchy を指定することと同じで、ディレクトリと、ディレクトリ下のサブプロジェクトを含む再帰的な子ディレクトリを削除する。
- その他のタイプのオブジェクトでは、このオプションは無効である。

`-repl|-replace`

オブジェクトを削除し、そのオブジェクトを先行バージョンに置き換えます。

`-scope (project_only | project_and_non-project_members | project_and_subproject_hierarchy | entire_project_hierarchy)`

プロジェクトの削除の範囲を指定します。project_only は、プロジェクトとそのルート ディレクトリのみを削除します。project_and_non-project_members は、プロジェクトと、プロジェクト内のサブプロジェクト以外のすべてのメンバーを削除します。project_and_subproject_hierarchy は、すべてのサブプロジェクトを含むプロジェクト階層全体を削除します。

`-scope (directory_only | directory_and_non-project_members | entire_directory_hierarchy)`

任意のディレクトリ オブジェクトの削除の範囲を指定します。directory_only は、ディレクトリ自体のみを削除します。directory_and_non-project_members の範囲は、ディレクトリと、ディレクトリ下のサブプロジェクト以外のすべての子ディレクトリを削除します。entire_directory_hierarchy は、ディレクトリと、ディレクトリ下のサブプロジェクトを含むすべての再帰的な子ディレクトリを削除します。

`-t|-task` *`task_spec`*

親ディレクトリが読み出し専用であるオブジェクトを削除すると、ディレクトリの新しいバージョンが自動的にチェックアウトされます。このオプションは、オブジェクトが読み出し専用ディレクトリから削除された場合、新しくチェックアウトしたディレクトリをそのタスクに関連付けます。カレント タスクが設定されており、別のタスクを指定しない場合は、新しく作成するディレクトリはカレント タスクに自動的に関連付けられます。詳細については、タスクの指定を参照してください。

例

- `sort.c` ファイルを削除し、旧バージョンのファイルに置き換える（実際の出力は以下の例とは異なる場合がある）。

  ```
  ccm delete sort.c
  Member sort.c-1 deleted from project ico_proj-1
  ```

- `sort.c` ファイルを削除する。

  ```
  ccm delete sort.c-1:csrc:J#1
  ```

- プロジェクトを削除する。

  ```
  ccm delete -p Project_delete-1:project:M#1
  ```

- プロジェクトをその階層とともに再帰的に削除する。

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

関連トピック

- create コマンド
- unuse コマンド
- use コマンド

説明と用途

オブジェクト バージョンがプロジェクトのメンバーではない場合、または現在のプロジェクトのみのメンバーで、後継バージョンがない場合、そのオブジェクト バージョンを削除できます。

注記：書き込み禁止ディレクトリからオブジェクトを削除すると、新しいディレクトリ バージョンが自動的にチェックアウトされます。

共有プロジェクト内にいて、現在のディレクトリが書き込み禁止の場合、そのディレクトリはチェックアウトされ、カレント（または指定した）タスクに自動的に関連付けられ、*integrate*（統合）状態にチェックインされます。初期設定ファイル内の `shared_project_directory_checkin` を FALSE に設定して、自動チェックイン機能を無効にできます（shared_project_directory_checkin を参照してください）。

注意！削除操作は恒久的です。

delimiter コマンド

詳細については、説明と用途を参照してください。`delimiter` コマンドは、現在のバージョンの区切り文字の表示サブコマンドをサポートします。

現在のバージョンの区切り文字の表示

区切り文字とは、プロジェクト名またはオブジェクト名とバージョン値を区切る文字のことです。プロジェクトをコピーするときに、プロジェクト名とバージョンを区切るために使用します。

```
ccm delim|delimiter
```

例

- 現在のバージョンの区切り文字を表示する。

  ```
  ccm delimiter
  ```

関連トピック

- プロジェクトのコピー
- ワークエリア プロパティの修正
- ワークエリア プロパティの表示

説明と用途

`delimiter` コマンドにより、区切り文字の値を表示します。

dir コマンド

詳細については、説明と用途を参照してください。`dir` コマンドは、ファイルの一覧表示サブコマンドをサポートします。

ファイルの一覧表示

`dir` コマンドは、ワークエリア内のプロジェクト オブジェクトまたはディレクトリ オブジェクト バージョンの内容を一覧表示します。デフォルトでは、出力は、大文字／小文字を区別せず名前順にソートされたファイル システム内のオブジェクトの一覧と、それらに関連付けられているプロジェクトから構成されます。

```
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`

出力フォーマットでカラム ヘッダーを使用するよう指定します。詳細については、-ch|-column_headers を参照してください。

`-f|-format format`

コマンドの出力フォーマットを指定します。詳細については、-f|-format を参照してください。

キーワードには、組み込み済みのもの（`%fullname`、`%displayname`、`%objectname`）、あるいは `%modify_time`、`%status` などの既存の属性の名前を使用できます。

キーワードのリストについては、組み込み済みキーワードを参照してください。

`-gby|-groupby groupformat`

コマンドの出力をグループ化する方法を指定します。詳細については、-gby|-groupby を参照してください。

`-m`

管理ファイルと非管理ファイルの両方、およびディレクトリを表示します。`-f|-format` オプションによってユーザー定義フォーマットが指定されていない場合、デフォルトのフォーマット（短いフォーマットまたは長いフォーマット）には、以下のように、ファイルの同期状態を示すカラムが含まれます。

- LC（ローカル コピー）：B プロジェクト内に存在し、ワークエリアにシンボリック リンクではなくローカル コピーが存在するファイルを意味します。表示されたファイルにこの記号が付いていて、ワークエリアがリンク ベースの場合は、リコ

ンサイル操作を行ってください。詳細については、reconcile コマンドを参照してください。

- NS（同期外れ）：B プロジェクト内には存在するが、ワークエリアには存在しないファイルを意味します。このような状態は、プロジェクトにファイルを追加したが、ワークエリアが見えない場合、またはファイルのリンクまたはローカル コピーが削除されている場合に発生します。ワークエリア内のほとんどのファイルがこの記号付きで表示された場合は、リコンサイル操作を行ってください。詳細については、reconcile コマンドを参照してください。
- UC（非管理）：B ワークエリア内には存在するが、プロジェクト内には存在しないファイルを意味します。非管理の記号の付いたファイルを表示するには、`-m` オプションに `-l` オプションをつけて使用する必要があります。ユーザー定義フォーマットでは、`%Sync` キーワードを使用して同期状態を表示します。

`-nch|-nocolumn_header`

出力フォーマットでカラム ヘッダーを使用しないよう指定します。詳細については、-nch|-nocolumn_headers を参照してください。

`-nf|-noformat`

カラム配置を使用しないよう指定します。詳細については、-nf|-noformat を参照してください。

`-ns|-nosort|-no_sort`

コマンドの出力をソートしないよう指定します。詳細については、-ns|-nosort を参照してください。

`path_or_file_spec`

パス リストを指定します。*`path_or_file_spec`* には、データベースに定義されているプロジェクト、ディレクトリ、またはファイルを指定できます。また、空のディレクトリ エントリも可能です。省略した場合、現在の作業ディレクトリが一覧表示されます。詳細については、ファイルの指定を参照してください。

`-p|-project`

プロジェクトを一覧表示するよう指定します。

`project_spec`

一覧表示するプロジェクトを指定します。詳細については、プロジェクトの指定を参照してください。

`-s`

サブディレクトリのメンバーを再帰的に表示します。このコマンドはサブプロジェクトには再帰しません。

`-sby|-sortby` *`sortspec`*

コマンドの出力をソートする方法を指定します。詳細については、-sby|-sortby を参照してください。

`-sep|-separator` *`separator`*

`-f|-format` オプションと一緒にのみ使用します。異なる区切り文字を指定します。詳細については、-sep|-separator を参照してください。

`-u|-unnumbered`

コマンドの出力の自動番号付けを抑止します（すなわち、出力に番号付けがされません）。詳細については、-u|-unnumbered を参照してください。

`-w`

デフォルトの短いフォーマットを使用するよう指定します。このオプションにより、各オブジェクトの表示名を表示します。

例

- Rational Synergy によって管理されていないファイルを一覧表示する。

```
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
```

- 現在のディレクトリを長いフォーマットで一覧表示する（先頭に LC が付いているファイルはローカル コピー ファイル）。

```
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
```

- 現在のディレクトリで、すべてのオブジェクトのファイル名とバージョンを一覧表示する。

```
ccm dir -w
ext_incl-1
incl-1
src-1
```

- 現在のディレクトリで、サブディレクトリを含むすべてのメンバーを表示する。

  ```
  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
  ```

- 現在のディレクトリで、すべてのオブジェクトの絶対パスを表示する。

  ```
  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
  ```

説明と用途

`dir` コマンドは、2 つのカテゴリのファイルを表示します。すなわち、Rational Synergy の管理下にあるオブジェクトと、ファイル システムのみに存在するファイルです。デフォルトでは、このコマンドは管理オブジェクトのみを表示します。`-m` オプションを使用すると、管理オブジェクトと非管理オブジェクトの両方を表示できます。

edit コマンド

詳細については、説明と用途を参照してください。`edit` コマンドは、ファイルの編集サブコマンドをサポートします。

ファイルの編集

このサブコマンドにより、指定したファイルを編集します。ファイルの編集には、デフォルトのエディタが使用されます。

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

file_spec

編集するファイルを指定します。詳細については、ファイルの指定を参照してください。

例

- `log.c` ファイルのバージョン 8 を編集する。オブジェクトを編集するには、そのオブジェクトが書き込み可能である必要があります。

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

警告

Windows では、修正可能ファイルはワークエリアが見えるプロジェクトからのみ編集できます。

UNIX では、現在のユーザーが変更可能なファイルのみを編集できます。

関連トピック

- view コマンド
- reconcile コマンド
- cli.text_editor
- cli.text_viewer

説明と用途

ワークエリア参照形式やプロジェクト参照形式のようなコンテキスト プロジェクトを提供する形式で編集対象ファイルを指定し、対応するワークエリアの場所がクライアントから見える場合、ワークエリアの場所でエディタが起動します。プロジェクト コンテキストが使用できないか、対応するワークエリアが見えない場合、起動したエディタでは、データベースのファイルから一時的な読み出し専用コピーが開きます。

> **注記**：コピーベースのワークエリアでファイルを編集すると、編集による変更は対応するデータベース オブジェクトには自動的に反映されません。最良の結果を得るには、ファイルの編集と保存を行うとき、ワークエリアを定期的にリコンサイルしてください。

finduse コマンド

詳細については、説明と用途を参照してください。`finduse` コマンドは、以下のサブコマンドをサポートします。

- ベースラインの使用箇所の検索
- 変更依頼の使用箇所の検索
- フォルダの使用箇所の検索
- フォルダ テンプレートの使用箇所の検索
- プロジェクトの使用箇所の検索
- タスクの使用箇所の検索
- オブジェクトの使用箇所の検索
- クエリで見つかったオブジェクトの使用箇所の検索

ベースラインの使用箇所の検索

このサブコマンドにより、ベースラインを使用するプロジェクト グルーピングおよびプロセス ルールを検索します。

> **注記**：このサブコマンドの適用範囲の定義については、適用範囲を参照してください。

```
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

使用箇所を表示するベースラインを指定します。詳細については、ベースラインの指定を参照してください。

例

- ベースライン **A#Base_One** を使用するプロジェクト グルーピングを検索する。

  ```
  ccm finduse -baseline A#Base_One
  ```

関連トピック

- プロジェクトの使用箇所の検索
- タスクの使用箇所の検索
- オブジェクトの使用箇所の検索
- クエリで見つかったオブジェクトの使用箇所の検索

変更依頼の使用箇所の検索

このサブコマンドにより、変更依頼を使用するベースラインまたはプロジェクト グルーピングを検索します。

> **注記**：このサブコマンドの適用範囲の定義については、適用範囲を参照してください。

```
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

change_request_spec で変更依頼を指定します。詳細については、変更依頼の指定を参照してください。

例

- 変更依頼 P#1265 を使用する、すべてのプロジェクト グルーピングを検索する。

  ```
  ccm finduse -my_project_grouping -cr P#1265
  ```

フォルダの使用箇所の検索

このサブコマンドにより、フォルダを使用するプロジェクト、プロジェクト グルーピング、またはプロセス ルールを検索します。

> **注記**：このサブコマンドの適用範囲の定義については、適用範囲を参照してください。

```
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

folder_spec が含まれる、すべてのオブジェクトを検索します。詳細については、フォルダの指定を参照してください。

例

- フォルダ 7 を使用する、すべてのプロジェクトを検索する。

  ```
  ccm finduse -folder 7

  Folder EAP#7: bill's Completed Tasks for Release 1.2
  draw_proj-bill
  ```

関連トピック

- フォルダの比較
- フォルダの一覧表示
- フォルダ プロパティの表示
- フォルダ情報の表示

フォルダ テンプレートの使用箇所の検索

このサブコマンドにより、フォルダ テンプレートを使用するプロセス ルールを検索します。

> **注記：**このサブコマンドの適用範囲の定義については、適用範囲を参照してください。

```
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

folder_template_spec が含まれる、すべてのオブジェクトを検索します。詳細については、フォルダ テンプレートの指定を参照してください。

例

- `All completed tasks for release %release` フォルダ テンプレートを使用する、すべてのプロセス ルールを検索する。

  ```
  ccm finduse -all_process_rules -ft "All completed tasks for release
  %release"
  ```

プロジェクトの使用箇所の検索

このサブコマンドにより、プロジェクトを使用するプロジェクト、プロジェクト グルーピング、またはベースラインを検索します。

> **注記**：このサブコマンドの適用範囲の定義については、適用範囲を参照してください。

```
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

project_spec が含まれる、すべてのプロジェクトを検索します。詳細については、プロジェクトの指定を参照してください。

例

- **project_sub2-win** プロジェクトを使用する、すべてのオブジェクトを検索する。

  ```
  ccm finduse -project project_sub2-win -all_projs -apg -all_folders -
  all_process_rules -all_baselines
  ```

関連トピック

- ベースラインの使用箇所の検索
- タスクの使用箇所の検索
- オブジェクトの使用箇所の検索
- クエリで見つかったオブジェクトの使用箇所の検索

タスクの使用箇所の検索

このサブコマンドにより、タスクを使用するプロジェクト、プロジェクト グルーピング、フォルダ、またはベースラインを検索します。

> **注記**：このサブコマンドの適用範囲の定義については、適用範囲を参照してください。

```
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

task_spec が含まれる、すべてのタスクを検索します。詳細については、タスクの指定を参照してください。

例

- **GA#1** タスクを使用する、すべてのオブジェクトを検索する。

  ```
  ccm finduse -task GA#1 -all_projs -all_baselines -all_folders -
  all_process_rules -apg
  ```

関連トピック

- ベースラインの使用箇所の検索
- プロジェクトの使用箇所の検索
- オブジェクトの使用箇所の検索
- クエリで見つかったオブジェクトの使用箇所の検索

オブジェクトの使用箇所の検索

このサブコマンドにより、オブジェクトを使用するプロジェクト、プロジェクト グルーピング、プロセス ルール、フォルダ、またはベースラインを検索します。

> **注記**：このサブコマンドの適用範囲の定義については、適用範囲を参照してください。

```
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

object_spec が含まれる、すべてのオブジェクトを検索します。詳細については、オブジェクトの指定を参照してください。

例

- プロジェクト内で、`display.c` という名前のオブジェクト バージョンの使用箇所をすべて検索する。

  ```
  ccm finduse -name display.c
  ```

- 現在のディレクトリで使用中の `draw.c` のバージョンのプロジェクトで、すべての使用箇所を検索する。

  ```
  ccm finduse draw.c
  ```

- オブジェクト `draw.c-2:csrc:EAP#1` が含まれる、すべての個人用のフォルダを検索する。

  ```
  ccm finduse -personal_folder draw.c-2:csrc:EAP#1
  ```

関連トピック

- ベースラインの使用箇所の検索
- プロジェクトの使用箇所の検索
- タスクの使用箇所の検索
- クエリで見つかったオブジェクトの使用箇所の検索

クエリで見つかったオブジェクトの使用箇所の検索

このサブコマンドにより、クエリ式またはクエリ条件と一致するオブジェクトをクエリした後、見つかったオブジェクトについて、それぞれ使用するプロジェクト、プロジェクトグルーピング、プロセス ルール、フォルダ、またはベースラインを検索します。クエリ式は以下のいずれかまたは両方を用いて修正できます。

- クエリ式
- クエリ文節を生成する 1 つまたは複数のクエリ関連オプション

各クエリ関連オプションは、クエリ文節を生成します。たとえば、`-name` *`name`* オプションは、`(name='`*`name`*`')` 形式でクエリ文節を生成します。

同じクエリ オプションを繰り返し使用すると、クエリ文節は `or` で結合されます。たとえば、`-name file1 -name file2` は、クエリ文節 `(name='file1' or name='file2')` を生成します。

複数のオプションまたはクエリ式から生成されるクエリ文節は、`and` で結合されます。たとえば、`-name file1 -owner joe` は、クエリ文節 `(name='file1') and (owner='joe')` を生成します。

> **注記：**このサブコマンドの適用範囲の定義については、適用範囲を参照してください。

```
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`*

`subsystem='`*`instance`*`'` 形式のクエリ文節を含めて、指定されたインスタンスを持つオブジェクトを検索します。

`-n|-name` *name*

`name='`*name*`'` 形式のクエリ文節を含めて、指定された名前を持つオブジェクトを検索します。

`-o|-owner` *owner*

`owner='`*owner*`'` 形式のクエリ文節を含めて、指定された所有者を持つオブジェクトを検索します。

`-release` *release_spec*

`release='`*releasename*`'` 形式のクエリ文節を含めて、指定されたリリース値を持つオブジェクトを検索します。*release_spec* には、1 つまたは複数のリリース定義またはリリース値を指定できます。詳細については、リリースの指定を参照してください。

`-st|-state` *state*

`status='`*state*`'` 形式のクエリ文節を含めて、指定された状態のオブジェクトを検索します。

`-t|-type` *type*

`type='`*type*`'` 形式のクエリ文節を含めて、指定されたタイプのオブジェクトを検索します。

`-v|-version` *version*

`version='`*version*`'` 形式のクエリ文節を含めて、指定されたバージョンのオブジェクトを検索します。

例

- クエリによって検索するオブジェクトのデフォルトの範囲を検索する。

```
ccm finduse -query ""cvtype='ascii' or (cvtype='task' and
status='completed')""
```

関連トピック

- ベースラインの使用箇所の検索
- プロジェクトの使用箇所の検索
- タスクの使用箇所の検索
- オブジェクトの使用箇所の検索

説明と用途

`finduse` コマンドは、データベースから、指定したオブジェクトの使用箇所を検索し、そのオブジェクトが使用されている箇所のリストを返します。

各 `finduse` コマンドは、返されるオブジェクトの範囲を定義するオプションをサポートします。範囲に関するオプションは、実行する検索の種類を定義します。以下の種類の検索を行うことができます。

- プロジェクト関連

  結果には、範囲に一致しており、指定されたオブジェクトを使用しているプロジェクトが含まれます。それぞれの結果はプロジェクト参照形式に示しています。

- プロジェクト グルーピング関連

  結果には、範囲に一致しており、指定されたオブジェクトを使用しているプロジェクト グルーピング含まれます。それぞれの結果はプロジェクト グルーピングの指定に示しています。

- プロセス ルール関連

  結果には、範囲に一致しており、指定されたオブジェクトを使用しているプロセス ルールが含まれます。それぞれの結果はプロセス ルールの指定に示しています。

- フォルダ テンプレート関連

  結果には、範囲に一致しており、指定されたオブジェクトを使用しているフォルダ テンプレートが含まれます。それぞれの結果はフォルダ テンプレートの指定に示しています。

- フォルダ関連

  結果には、範囲に一致しており、指定されたオブジェクトを使用しているフォルダが含まれます。それぞれの結果はプロジェクトの指定に示しています。

- 変更依頼関連

  結果には、範囲に一致しており、指定されたオブジェクトを使用している変更依頼が含まれます。それぞれの結果は変更依頼の指定に示しています。

- ベースライン関連

  結果には、指定されたオブジェクトを使用するベースラインが含まれます。それぞれの結果はベースラインの指定に示しています。

使用されるデフォルトの範囲は、`finduse` コマンドで使用するオブジェクトのタイプによって異なります。

- 変更依頼

  範囲オプションを指定しない場合、デフォルトの範囲はすべてのベースラインとなります。

- タスク

範囲オプションを指定しない場合、デフォルトの範囲はすべてのプロジェクトとなります。

- プロジェクト、ファイル、またはディレクトリ

  範囲オプションを指定しない場合、デフォルトの範囲はすべてのプロジェクトとなります。

- フォルダ

  プロジェクトまたはプロジェクト グルーピングの範囲オプションを指定しない場合、常にすべてのプロジェクトが範囲に含まれます。

- フォルダ テンプレート

  範囲を指定しない場合、デフォルトの範囲はすべてのプロセス ルールとなります。

- ベースライン

  プロジェクト グルーピングの範囲オプションを指定しない場合、すべてのプロジェクト グルーピングが範囲に含まれます。

適用範囲

適用範囲によって、以下のような検索の範囲を指定します。

ベースラインの範囲

- `-all_baseline|-all_baselines`

  ベースライン内のすべての使用箇所を表示するよう指定します。

変更依頼の範囲

変更依頼内のすべての使用箇所を表示するよう指定します。

フォルダの範囲

- `-all_fold|-all_folds|-all_folder|-all_folders`

  任意の状態のフォルダ内のすべての使用箇所を表示するよう指定します。

- `-personal_fold|-personal_folds|-personal_folder|-personal_folders`

  個人用フォル内のすべての使用箇所を表示するよう指定します。

- `-shared_fold|-shared_folds|-shared_folder|-shared_folders`

  共有フォルダ内のすべての使用箇所を表示するよう指定します。

- `-prep_fold|-prep_folds|-prep_folder|-prep_folders`

  prep（ビルド管理）フォルダ内のすべての使用箇所を表示するよう指定します。

- `-non_write_fold|-non_write_folds|-non_write_folder|-non_write_folders`

  書き込み禁止フォルダ内のすべての使用箇所を表示するよう指定します。

フォルダ テンプレートの範囲

フォルダ テンプレート内のすべての使用箇所を表示するよう指定します。

プロジェクトの範囲

- `-all_proj|-all_projs|-all_project|-all_projects`

  任意の状態のプロジェクト内のすべての使用箇所を表示するよう指定します。

- `-shared_proj|-shared_projs|-shared_project|-shared_projects`

  共有プロジェクト内のすべての使用箇所を表示するよう指定します。

- `-prep_proj|-prep_projs|-prep_project|-prep_projects`

  prep（ビルド管理）プロジェクト内のすべての使用箇所を表示するよう指定します。

- `-working_proj|-working_projs|-working_project|-working_projects`

  開発プロジェクト内のすべての使用箇所を表示するよう指定します。

- `-released_proj|-released_projs|-released_project|-released_projects`

  リリース済みプロジェクト内のすべての使用箇所を表示するよう指定します。

プロセス ルールの範囲

- `-all_process_rules`

  クエリを使用して指定または検索されたタイプを持つ、ベースライン、フォルダ、またはフォルダ テンプレート オブジェクトのプロセス ルール内の、すべての使用箇所を表示するよう指定します。

プロジェクト グルーピングの範囲

- `-apg|-all_project_grouping|-all_project_groupings`

  任意の状態または所有者の、プロジェクト グルーピング内のすべての使用箇所を表示するよう指定します。

- `-mpg|-my_project_grouping|-my_project_groupings`

  ユーザーが所有するプロジェクト グルーピング内のすべての使用箇所を表示するよう指定します。

- `-ppg|-prep_project_grouping|-prep_project_groupings`

  prep（ビルド管理）プロジェクト グルーピング内のすべての使用箇所を表示するよう指定します。

- `-spg|-shared_project_grouping|-shared_project_groupings`

  共有プロジェクト グルーピング内のすべての使用箇所を表示するよう指定します。

- `-wpg|-working_project_grouping|-working_project_groupings`

  作業中のプロジェクト グルーピング内のすべての使用箇所を表示するよう指定します。

folder コマンド

詳細については、説明と用途を参照してください。`folder` コマンドは、以下のサブコマンドをサポートします。

- フォルダの比較
- フォルダのコピー
- フォルダの作成
- **フォルダの削除**
- **フォルダの使用箇所の検索**
- フォルダの一覧表示
- **フォルダの修正**
- フォルダ プロパティの表示
- フォルダに関連付けられたタスクまたはオブジェクトの表示
- **フォルダ情報の表示**

フォルダの比較

このサブコマンドにより、2 つのフォルダの内容を比較します。

`-union`、`-intersection`、または `-not_in` を指定する必要があります。

```
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`

出力フォーマットでカラム ヘッダーを使用するよう指定します。詳細については、-ch|-column_headers を参照してください。

`folder_spec1`

比較する最初のフォルダを指定します。詳細については、フォルダの指定を参照してください。

`folder_spec2`

比較する 2 番目のフォルダを指定します。詳細については、フォルダの指定を参照してください。

`-f|-format` *`format`*

コマンドの出力フォーマットを指定します。詳細については、-f|-format を参照してください。

`-gby|-groupby` *`groupformat`*

コマンドの出力をグループ化する方法を指定します。詳細については、-gby|-groupby を参照してください。

`-int|-intersection`

フォルダ比較によって、両方のフォルダに含まれているタスクを表示するよう指定します。

`-nch|-nocolumn_header`

出力フォーマットでカラム ヘッダーを使用しないよう指定します。詳細については、-ch|-column_headers を参照してください。

`-nf|-noformat`

カラム配置を使用しないよう指定します。詳細については、-nf|-noformat を参照してください。

`-ns|-no_sort`

コマンドの出力をソートしないよう指定します。詳細については、-ns|-nosort を参照してください。

`-not|-not_in`

フォルダ比較によって、*folder_spec1* にあって *folder_spec2* にないタスクを表示するよう指定します。

`-sep|-separator` *separator*

異なる区切り文字を指定します。詳細については、-sep|-separator を参照してください。

`-sby|-sortby` *sortspec*

コマンドの出力をソートする方法を指定します。詳細については、-sby|-sortby を参照してください。

`-un|-union`

フォルダ比較によって、2 つのフォルダのいずれかに含まれているタスクを表示するよう指定します。

`-u|-unnumbered`

コマンドの出力の自動番号付けを抑止します（すなわち、出力に番号付けがされません）。詳細については、-u|-unnumbered を参照してください。

例

- フォルダ **154** または **155** のいずれかにあるタスクを表示する。

```
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
```

- フォルダ **154** と **155** が共通して持っているタスクを表示する。

  ```
  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
  ```

- フォルダ **154** にあって、**155** にないタスクを表示する。

  ```
  ccm folder -compare -not_in 154 155
  1) Task 12: System error when time zone changes
  2) Task 31: Wrong window receives message
  ```

関連トピック

- **フォルダの使用箇所の検索**
- フォルダの一覧表示
- フォルダ プロパティの表示
- **フォルダ情報の表示**

フォルダのコピー

このサブコマンドにより、フォルダの内容を新しいフォルダまたは既存のフォルダにコピーします。開発者またはビルド マネージャは、フォルダを新しいフォルダにコピーできます。フォルダを既存のフォルダにコピーする場合、コピー先のフォルダは自分が修正可能である必要があります。

```
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`

-e|-existing folder_spec オプションと一緒に使用することで、コピー先フォルダのタスクを置き換えることなく、コピー元フォルダのタスクをコピー先フォルダに追加するよう指定します。

`-e|-existing` *`folder_spec`*

フォルダを *`folder_spec`* によって指定した既存のフォルダにコピーするよう指定します。*`folder_spec`* には、1 つのフォルダを指定できます。詳細については、フォルダの指定を参照してください。

デフォルトでは、コピー先フォルダ内のタスクがコピー元フォルダ内のタスクに置き換えられます。タスクを置き換えるのではなくて追加するには、`-append` オプションを使用します。

`folder_spec`

コピー元のフォルダを指定します。詳細については、フォルダの指定を参照してください。

`-new` *`new_folder_name`*

フォルダを *`new_folder_name`* で指定した名前を持つ新しいフォルダにコピーするよう指定します。*`new_folder_name`* には復帰改行文字を含めることはできません。

`-q|-quiet`

更新または作成したフォルダの表示名を表示するよう指定します。表示名には有効なフォルダの指定が表示されます。

例

- フォルダ **95** を **Tasks Completed for Release 3.4 on September 15, 1997** という名前の新しいフォルダにコピーする。

  ```
  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'
  ```

- フォルダ 95 を既存のフォルダ 103 にコピーする。

  ```
  ccm folder -cp 95 -existing 103
  Folder '95: Tasks Completed for Release 3.4' copied to '103:  Tested
  Tasks for Release 3.4"
  ```

- フォルダ `95` を既存フォルダ `103` にコピーしてタスクを追加する。

  ```
  ccm folder -copy 95 -append -existing 103
  ```

関連トピック

- **フォルダの使用箇所の検索**
- フォルダの一覧表示
- フォルダ プロパティの表示
- **フォルダ情報の表示**

フォルダの作成

このサブコマンドにより、フォルダを作成します。

オプションを複数回使用すると、各使用オプションに関するクエリ式が「or」で結合されます。たとえば、`-release 1.0 -release 2.0` を指定すると、クエリ式 `(release='1.0' or release='2.0)` が得られます。

複数のオプションからのクエリ式は「and」で結合されます。たとえば、`-release 1.0 -platform windows` を指定すると、クエリ式 `(release='1.0') and (platform='windows')` が得られます。

```
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`

指定したカスタム クエリ式を新規フォルダ クエリに含めるように指定します。

`-db|-dbid|-database_id database_spec...`

-ts|-scope|-task_scope オプションと一緒に使用することで、タスク範囲から生成されるクエリを修飾するデータベース ID を指定します。詳細については、データベースの指定を参照してください。

`-mode ((man|manual) | (uq|use_query))`

新しいフォルダへのタスクの追加を、手動で行うかクエリを使用するかを指定します。

`-mode` または `-query` を指定しない場合、デフォルト モードはクエリ関連オプションを指定するかどうかによって変わります。`-custom`、`-dbid`、`-platform`、`-release`、`-subsystem`、`-task_scope` を指定した場合、デフォルト モードはクエリベースとなります。これらのいずれのオプションも指定しない場合、デフォルトでは手動でタスクをフォルダに追加します。

`-mode use_query` または `-query` を指定し、`-custom`、`-dbid`、`-platform`、`-release`、`-subsystem` または `-task_scope` を指定しない場合、デフォルトのタスク クエリは次のようになります。

- default_task_query を定義していた場合、このデフォルト タスク クエリが使用される。
- デフォルト クエリを定義していなかった場合、タスク範囲 **All my assigned and completed tasks** が使用される。

`-n|-name` *`folder_name`*

作成する新規フォルダの名前を指定します。*`folder_name`* には復帰改行文字を含めることはできません。

`-plat|-platform` *`platform`*

フォルダのタスク クエリに、指定したプラットフォームのクエリが含まれるよう指定します。

`-purpose` *`purpose`*

指定した目的のクエリを含むタスク クエリによってフォルダを作成するよう指定します。目的の詳細については、`project_purpose` コマンドの説明と用途を参照してください。

通常、このオプションは、`component_task_projects`、`component_task_products`、または `component_task_projects_products` のいずれかの範囲で指定されるコンポーネント タスクのクエリに適用されます。

`-qu|-query`

新規フォルダがクエリベースとなるように指定します。これは `-mode use_query` の使用と同じ意味になります。詳細については、-mode ((man|manual) | (uq|use_query)) を参照してください。

`-q|-quiet`

作成したフォルダの表示名のみを表示するよう指定します。表示名には有効なフォルダの指定が表示されます。

`-rel|-release` *`release_spec`*

指定したリリース値のクエリを含むタスク クエリによってフォルダを作成するよう指定します。*`release_spec`* には、複数のリリース値を指定できます。詳細については、リリースの指定を参照してください。

`-sub|-subsystem` *`subsystem`*

タスク サブシステムのクエリ式を含むタスク クエリによってフォルダを作成するよう指定します。

`-ts|-scope|-task_scope`

タスク クエリを使用するよう指定します。タスク クエリには指定した範囲によって決まるクエリ式が含まれます。指定した範囲に関連付けられるクエリ式は、`-database_id` オプションによっても異なります。以下の範囲を使用できます。

- *`user_defined`*

  この範囲は default_task_query オプションによって定義されます。`-database_id` を指定すると、クエリには、指定したデータベース内で修正可能または完了したタスクのクエリ式も含まれます。

- `all_my_assigned|all_owners_assigned`

  この範囲では、自分に割り当てられているすべてのタスクをクエリします。`-database_id` を指定すると、指定したデータベースで修正可能な、自分に割り当てられているすべてのタスクをクエリします。

- `all_my_assigned_or_completed|all_owners_assigned_or_completed`

  この範囲では、自分に割り当てられているすべてのタスク、または自分が完了したすべてのタスクをクエリします。`-database_id` を指定すると、指定したデータベース内で自分に割り当てられている修正可能なタスク、または指定したデータベース内で自分が完了したすべてのタスクをクエリします。

- `all_my_completed|all_owners_completed`

  この範囲では、自分が完了したすべてのタスクをクエリします。`-database_id` を指定すると、指定したデータベース内で自分が完了したすべてのタスクをクエリします。

- `all_my_tasks|all_owners_tasks`

  この範囲では、自分がタスク担当者であるすべてのタスクをクエリします。`-database_id` を指定すると、自分がタスク担当者であり、指定したデータベース内で修正可能または自分が完了したすべてのタスクをクエリします。

- `all_completed`

  この範囲では、完了したすべてのタスクをクエリします。`-database_id` を指定すると、指定したデータベースで完了したすべてのタスクをクエリします。

- `all_tasks`

  この範囲では、すべてのタスクをクエリします。`-database_id` を指定すると、指定したデータベース内で修正可能または完了したすべてのタスクをクエリします。

- `component_task_projects|component_task_products|`
  `component_task_projects_products`

この範囲では、プロジェクト、製品、またはプロジェクトと製品のコンポーネント タスクをクエリします。-database_id を指定すると、指定したデータベースで作成されたすべてのコンポーネント タスクをクエリします。-purpose を指定すると、指定した目的を持つコンポーネント タスクをクエリします。

-w|-writable (owner | (build_mgr|build_manager|buildmanager) | all | none)

新規フォルダを誰が修正可能かを指定します。

例

- 所有者が書き込み可能な **Tested Tasks for Release 3.5** という名前の新規フォルダを作成し、フォルダ ID 以外のすべてのコマンド出力を抑止する。

  ```
  ccm folder -cr -n "Tested Tasks for Release 3.5" -w Owner -q 159
  ```

- *query_spec* に *task_spec* と *release* 値を使用する、**My Tasks for Release 3.5** という名前の新規フォルダを作成する。

  ```
  ccm folder -cr -name "My Tasks for Release 3.5" -ts all_my_tasks -rel 3.5
  Created folder 160.
  ```

関連トピック

- フォルダの使用箇所の検索
- フォルダの一覧表示
- フォルダ プロパティの表示
- フォルダ情報の表示

フォルダの削除

このサブコマンドにより、指定したフォルダを削除します。フォルダは自分が修正可能である必要があります。

```
ccm folder -d|-delete folder_spec...
```

folder_spec

削除するフォルダを指定します。詳細については、フォルダの指定を参照してください。

例

- フォルダ **109**、**110**、および **158** を削除する。

```
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.
```

関連トピック

- フォルダの使用箇所の検索
- フォルダの一覧表示
- フォルダ プロパティの表示
- フォルダ情報の表示

フォルダの使用箇所の検索

このサブコマンドにより、指定したフォルダの現在のデータベースでの使用箇所を検索します。

```
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`

出力フォーマットでカラム ヘッダーを使用するよう指定します。詳細については、-ch|-column_headers を参照してください。

`folder_spec`

プロジェクトで使用箇所の検索を行うときに、検索対象とするフォルダを指定します。

詳細については、フォルダの指定を参照してください。

`-f|-format` *`format`*

コマンドの出力フォーマットを指定します。詳細については、-f|-format を参照してください。

`-gby|-groupby` *`groupformat`*

コマンドの出力をグループ化する方法を指定します。詳細については、-gby|-groupby を参照してください。

`-nch|-nocolumn_header`

出力フォーマットでカラム ヘッダーを使用しないよう指定します。詳細については、-ch|-column_headers を参照してください。

`-nf|-noformat`

カラム配置を使用しないよう指定します。詳細については、-nf|-noformat を参照してください。

`-ns|-no_sort`

コマンドの出力をソートしないよう指定します。詳細については、-ns|-nosort を参照してください。

`-sep|-separator` *`separator`*

異なる区切り文字を指定します。詳細については、-sep|-separator を参照してください。

`-u|-unnumbered`

コマンドの出力の自動番号付けを抑止します（すなわち、出力に番号付けがされません）。詳細については、-u|-unnumbered を参照してください。

例

- フォルダ `123` と `234` を使用するプロジェクトを検索する。

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

関連トピック

- フォルダのコピー
- **フォルダの削除**
- フォルダの一覧表示
- **フォルダの修正**
- フォルダ プロパティの表示
- フォルダに関連付けられたタスクまたはオブジェクトの表示
- **フォルダ情報の表示**

フォルダの一覧表示

このサブコマンドにより、指定した基準を満すフォルダを一覧表示します。オプションまたは引数を指定しない場合、すべてのフォルダを一覧表示します。オプションまたは引数を指定した場合は、以下のようになります。

- `all_personal` 範囲では、所有者が書き込み可能なフォルダを一覧表示する。
- `all_build_mgrs` 範囲では、ビルド マネージャが書き込み可能なフォルダを一覧表示する。
- `all_shared` 範囲では、全員が書き込み可能なフォルダを一覧表示する。
- `all_non_writable` 範囲では、読み出し専用のフォルダを一覧表示する。

```
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`

出力フォーマットでカラム ヘッダーを使用するよう指定します。詳細については、-ch|-column_headers を参照してください。

`-f|-format format`

コマンドの出力フォーマットを指定します。詳細については、-f|-format を参照してください。

`-gby|-groupby groupformat`

コマンドの出力をグループ化する方法を指定します。詳細については、-gby|-groupby を参照してください。

`-nch|-nocolumn_header`

出力フォーマットでカラム ヘッダーを使用しないよう指定します。詳細については、-ch|-column_headers を参照してください。

`-nf|-noformat`

カラム配置を使用しないよう指定します。詳細については、-nf|-noformat を参照してください。

`-ns|-no_sort`

コマンドの出力をソートしないよう指定します。詳細については、-ns|-nosort を参照してください。

`-sep|-separator` *`separator`*

異なる区切り文字を指定します。詳細については、-sep|-separator を参照してください。

`-sby|-sortby` *`sortspec`*

コマンドの出力をソートする方法を指定します。詳細については、-sby|-sortby を参照してください。

`-u|-unnumbered`

コマンドの出力の自動番号付けを抑止します（すなわち、出力に番号付けがされません）。詳細については、-u|-unnumbered を参照してください。

例

- 現在のデータベース内にあるすべてのビルド管理フォルダを一覧表示する。

```
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
```

- 自分のすべての個人用フォルダを一覧表示する。

```
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
```

関連トピック

- フォルダの比較
- フォルダの使用箇所の検索
- フォルダ プロパティの表示
- **フォルダ情報の表示**

フォルダの修正

このサブコマンドにより、指定したファイルを修正します。フォルダは自分が修正可能である必要があります。

フォルダ クエリを定義するとき、-custom、-platform、-release、-subsystem、および -task_scope オプションが、最終的に生成されるタスク クエリに影響します。-platform、-release および -subsystem オプションは、複数回使用できます。オプションを複数回使用すると、各使用オプションに関するクエリ式が「or」で結合されます。たとえば、-release 1.0 -release 2.0 を指定すると、クエリ式 (release='1.0' or release='2.0') が得られます。複数のオプションからのクエリ式は「and」で結合されます。たとえば、-release 1.0 -platform windows を指定すると、クエリ式 (release='1.0') and (platform='windows') が得られます。-task_scope オプションも指定した範囲に基づくタスク クエリを提供しますが、これは指定した -database_id オプションによって修飾されます。最終的に使用されるタスク クエリでは、これらすべての要素が 1 つのクエリ式にまとめられます。

```
ccm folder -m|-modify [-n|-name folder_name]
          [-mode ((man|manual) | (uq|use_query))]
          [-w|-writable (owner | (build_mgr|build_manager|buildmanager) |
          all | none)] [-cus|-custom custom_query]
          [(-db|-dbid|-database_id database_spec)...]
          [(-plat|-platform platform)...] [(-purpose purpose)...]
          [(-rel|-release release_spec)...]
          [(-sub|-subsystem subsystem)...] [-ts|-scope|-task_scope
          (user_defined | (all_my_assigned|all_owners_assigned) |
          (all_my_assigned_or_completed|all_owners_assigned_or_completed) |
          (all_my_completed|all_owners_completed) |
          (all_my_tasks|all_owners_tasks) | all_completed | all_tasks)]
          (ct_projs|ct_projects|component_task_projects) |
          (ct_prods|ct_products|component_task_products) |
          (ct_projs_prods|ct_projects_products|
          component_task_projects_products))]
          [(-at|-add_task|-add_tasks task_spec)...]
          [(-rt|-remove_task|-remove_tasks task_spec)...]
          [-up|-update] folder_spec...
```

-at|-add_task|-add_tasks *task_spec*

指定したタスクを指定したフォルダへ追加します。詳細については、タスクの指定 を参照してください。

-cus|-custom *custom_query*

フォルダ クエリを更新して指定したカスタム クエリ式を含むよう指定します。

`-db|-dbid|-database_id` *`database_spec`*

`-task_scope` オプションと一緒に使用することで、タスク範囲から生成されるクエリを修飾するデータベース ID を指定します。詳細については、-ts|-scope|-task_scope と データベースの指定 を参照してください。

`folder_spec`

修正するフォルダを指定します。詳細については、フォルダの指定を参照してください。

`-mode ((man|manual) | (uq|use_query))`

フォルダに手動でタスクを追加するか、クエリを使用するかを指定します。

フォルダを手動からクエリベースに変更し、タスク クエリを定義せず、使用可能なオプションを指定しなかった場合、default_task_query が次のように定義された形でフォルダがクエリベースとして作成されます。

- デフォルトのタスク クエリ を定義していた場合、それが使用される。
- デフォルト クエリを定義していなかった場合、タスク範囲 **All my assigned and completed tasks** が使用される。

`-n|-name` *`folder_name`*

指定したフォルダを指定したフォルダ名に変更するよう指定します。*`folder_name`* には復帰改行文字を含めることはできません。

`-plat|-platform` *`platform`*

フォルダ クエリを更新して指定したプラットフォームのクエリ式を使用するよう指定します。

`-purpose` *`purpose`*

指定した目的のクエリを含むタスク クエリによってフォルダを作成するよう指定します。目的の詳細については、`project_purpose` コマンドの説明と用途を参照してください。

通常、このオプションは、`component_task_projects`、`component_task_products`、または `component_task_projects_products` のいずれかの範囲で指定されるコンポーネント タスクのクエリに適用されます。

`-rel|-release` *`release_spec`*

フォルダ クエリを更新して指定したリリース値のクエリ式を使用するよう指定します。*`release_spec`* には、複数のリリース値を指定できます。詳細については、リリースの指定 を参照してください。

`-rt|-remove_task|-remove_tasks` *`task_spec`*

指定したフォルダから指定したタスクを削除します。詳細については、タスクの指定 を参照してください。

`-sub|-subsystem` *`subsystem`*

フォルダ クエリを更新してタスク サブシステムのクエリ式を使用するよう指定します。

`-ts|-scope|-task_scope`

タスク クエリを使用するよう指定します。タスク クエリには指定した範囲によって決まるクエリ式が含まれます。指定した範囲に関連付けられるクエリ式は、`-database_id` オプションによっても異なります。以下の範囲を使用できます。

- *`user_defined`*

  この範囲は default_task_query オプションによって定義されます。`-database_id` を指定すると、クエリには、指定したデータベース内で修正可能または完了したタスクのクエリ式も含まれます。

- `all_my_assigned|all_owners_assigned`

  この範囲では、自分に割り当てられているすべてのタスクをクエリします。`-database_id` を指定すると、指定したデータベースで修正可能な、自分に割り当てられているすべてのタスクをクエリします。

- `all_my_assigned_or_completed|all_owners_assigned_or_completed`

  この範囲では、自分に割り当てられているすべてのタスク、または自分が完了したすべてのタスクをクエリします。`-database_id` を指定すると、指定したデータベース内で自分に割り当てられている修正可能なタスク、または指定したデータベース内で自分が完了したすべてのタスクをクエリします。

- `all_my_completed|all_owners_completed`

  この範囲では、自分が完了したすべてのタスクをクエリします。`-database_id` を指定すると、指定したデータベース内で自分が完了したすべてのタスクをクエリします。

- `all_my_tasks|all_owners_tasks`

  この範囲では、自分がタスク担当者であるすべてのタスクをクエリします。`-database_id` を指定すると、自分がタスク担当者であり、指定したデータベース内で修正可能または自分が完了したすべてのタスクをクエリします。

- `all_completed`

  この範囲では、完了したすべてのタスクをクエリします。`-database_id` を指定すると、指定したデータベースで完了したすべてのタスクをクエリします。

- `all_tasks`

  この範囲では、すべてのタスクをクエリします。`-database_id` を指定すると、指定したデータベース内で修正可能または完了したすべてのタスクをクエリします。

- `component_task_projects|component_task_products|component_task_projects_products`

  この範囲では、プロジェクト、製品、またはプロジェクトと製品のコンポーネント タスクをクエリします。`-database_id` を指定すると、指定したデータベースで作成されたすべてのコンポーネント タスクをクエリします。`-purpose` を指定すると、指定した目的を持つコンポーネント タスクをクエリします。

`-up|-update`

フォルダのクエリを実行することによりクエリベース フォルダを更新するよう指定します。指定したフォルダがクエリベースでない場合、エラーが報告されます。

`-w|-writable (owner | (build_mgr|build_manager|buildmanager) | all | none)`

指定したフォルダを誰が修正可能かを指定します。

例

- フォルダ **95** にタスク **5-9** を追加する。

```
ccm folder -modify -at 5-9 95
Updating folder 95: Tested Tasks for Release 3.2 ...
     Added task 5
     Added task 6
     Added task 7
     Added task 8
     Task 9 is already in the folder
     Added 4 tasks.
```

- フォルダ **95** からタスク **5-9** を削除する。

```
ccm folder -modify -rt 5-9 95
Updating folder 95: Tested Tasks for Release 3.2 ...
   Removed task 5
   Removed task 6
   Removed task 7
   Removed task 8
   Removed task 9
   Removed 5 tasks.
```

- フォルダ **51** に複数のタスク（**5**、**12**、**14**）を追加する。

```
ccm folder -modify -add_task 5,12,14 51
```

- フォルダ **160** の内容を更新する。

  ```
  ccm folder -m -up 160
  Updated folder '160: My Tasks for Release 3.5'.
  ```

- フォルダ **111** のモードを変更してタスク追加のクエリを使用できるようにする。

  ```
  ccm folder -modify -mode use_query 111
  ```

- フォルダ **111** を変更して、`all_my_tasks` 範囲とリリース **3.5** を使用してタスクを追加できるようにする。

  ```
  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'
  ```

- フォルダ **85** の名前を **Completed tasks for release 3.5** に変更する。

  ```
  ccm folder -modify -name "Completed tasks for release 3.5" 85
  ```

関連トピック

- **フォルダの削除**
- フォルダの一覧表示
- フォルダ プロパティの表示
- フォルダに関連付けられたタスクまたはオブジェクトの表示
- **フォルダ情報の表示**

フォルダ プロパティの表示

このサブコマンドにより、指定したフォルダのプロパティを表示します。

```
ccm folder -s|-sh|-show (mode | (n|na|name) | (q|qu|query) | (w|wr|writable))
          folder_spec...
```

folder_spec

プロパティを表示したいフォルダを指定します。詳細については、フォルダの指定を参照してください。

関連トピック

- フォルダのコピー
- **フォルダの削除**
- フォルダの一覧表示
- **フォルダの修正**
- フォルダに関連付けられたタスクまたはオブジェクトの表示
- **フォルダ情報の表示**

フォルダに関連付けられたタスクまたはオブジェクトの表示

指定したフォルダについて、関連付けられているタスクまたは関連付けられているタスクのオブジェクトを表示します。

```
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`

出力フォーマットでカラム ヘッダーを使用するよう指定します。詳細については、-ch|-column_headers を参照してください。

`folder_spec`

表示するフォルダを指定します。詳細については、フォルダの指定 を参照してください。

`-f|-format` *`format`*

コマンドの出力フォーマットを指定します。詳細については、-f|-format を参照してください。

`-gby|-groupby` *`groupformat`*

コマンドの出力をグループ化する方法を指定します。詳細については、-gby|-groupby を参照してください。

`-nch|-nocolumn_header`

出力フォーマットでカラム ヘッダーを使用しないよう指定します。詳細については、-ch|-column_headers を参照してください。

`-nf|-noformat`

カラム配置を使用しないよう指定します。詳細については、-nf|-noformat を参照してください。

`-ns|-no_sort`

コマンドの出力をソートしないよう指定します。詳細については、-ns|-nosort を参照してください。

`-sep|-separator` *separator*

異なる区切り文字を指定します。詳細については、-sep|-separator を参照してください。

`-sby|-sortby` *sortspec*

コマンドの出力をソートする方法を指定します。詳細については、-sby|-sortby を参照してください。

`-u|-unnumbered`

コマンドの出力の自動番号付けを抑止します（すなわち、出力に番号付けがされません）。詳細については、-u|-unnumbered を参照してください。

`-v|-verbose`

フォルダ情報の表示に詳細フォーマットを使用するよう指定します。

例

- フォルダ **111** 内のタスクを表示する。

```
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
```

- フォルダ **160** に関連付けられたオブジェクトを表示する。

```
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
```

関連トピック

- フォルダのコピー
- **フォルダの削除**
- フォルダの一覧表示
- **フォルダの修正**
- フォルダ プロパティの表示
- **フォルダ情報の表示**

フォルダ情報の表示

このサブコマンドにより、指定したフォルダの情報を表示します。

```
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`

出力フォーマットでカラム ヘッダーを使用するよう指定します。詳細については、-ch|-column_headers を参照してください。

`folder_spec`

表示するフォルダを指定します。詳細については、フォルダの指定 を参照してください。

`-f|-format` *`format`*

コマンドの出力フォーマットを指定します。詳細については、-f|-format を参照してください。

`-gby|-groupby` *`groupformat`*

コマンドの出力をグループ化する方法を指定します。詳細については、-gby|-groupby を参照してください。

`-nch|-nocolumn_header`

出力フォーマットでカラム ヘッダーを使用しないよう指定します。詳細については、-ch|-column_headers を参照してください。

`-sep|-separator` *`separator`*

異なる区切り文字を指定します。詳細については、-sep|-separator を参照してください。

`-v|-verbose`

詳細なフォルダ情報が必要であることを指定します。

例

- フォルダ **2** の情報を表示する。

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

```
Folder 2:
  Description:   joe’s Assigned Or Completed Tasks for Release 1.0
  Owner:         joe
  Writable By:   Owner
  Mode:          Adds Tasks Using a Query
  Query:         owner='joe' and (status='task_assigned' or
  status='completed') and release='1.0'
  Modifiable In:  <not available>
```

関連トピック

- フォルダのコピー
- **フォルダの削除**
- フォルダの一覧表示
- **フォルダの修正**
- フォルダ プロパティの表示
- フォルダに関連付けられたタスクまたはオブジェクトの表示

説明と用途

フォルダを使用して、プロジェクトおよびプロジェクト グルーピングの更新プロパティを定義します。通常、フォルダは、プロセス ルールで定義されたフォルダ テンプレートから作成されます。

folder_template コマンド

詳細については、説明と用途を参照してください。`folder_template` コマンドは、以下のサブコマンドをサポートします。

- フォルダ テンプレートの作成
- フォルダ テンプレートの削除
- フォルダ テンプレートの使用箇所の検索
- フォルダ テンプレートの一覧表示
- フォルダ テンプレートの修正
- フォルダ テンプレートの管理データベースの設定
- フォルダ テンプレート プロパティの表示
- フォルダ テンプレート情報の表示

フォルダ テンプレートの作成

このサブコマンドにより、フォルダ テンプレートを作成します。このサブコマンドを使用するには、*build_mgr* または *ccm_admin* ロールを持っている必要があります。

オプションを複数回使用すると、各使用オプションに関するクエリ式が「or」で結合されます。たとえば、`-release 1.0 -release 2.0` を指定すると、クエリ式 `(release='1.0' or release='2.0)` が得られます。

複数のオプションからのクエリ式は「and」で結合されます。たとえば、`-release 1.0 -platform windows` を指定すると、クエリ式 `(release='1.0') and (platform='windows')` が得られます。

```
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`*

指定したカスタム クエリ式を新規フォルダ テンプレート クエリに含めるよう指定します。

`-desc|-description` *`description`*

フォルダ テンプレートからフォルダを作成するときに、キーワード展開後に使用される文字列を指定します。復帰改行文字を含めることはできません。*`description`* を指定しないと、デフォルト値としてフォルダ テンプレート名が使用されます。キーワード展開の詳細については、フォルダ テンプレートの説明を参照してください。

`-db|-dbid|-database_id` *`database_spec`*

作成するフォルダ テンプレートに関連付けられるデータベース ID を指定します。詳細については、データベースの指定を参照してください。

`-mode ((man|manual) | (uq|use_query))`

フォルダ テンプレートの内容を手動またはクエリベースのいずれかに指定します。

クエリを定義していなかった場合、default_task_query が使用されます。

`-must_be_local`

ローカルに作成したプロジェクトの更新プロパティに対して、フォルダ テンプレートがローカル フォルダを使用するよう指定します。デフォルトは `-nomust_be_local` です。

name

作成する新規フォルダ テンプレートの名前を指定します。*name* には復帰改行文字を含めることはできません。

`-nomust_be_local`

ローカルに作成したプロジェクトの更新プロパティに対して、フォルダ テンプレートが非ローカル フォルダを使用できるよう指定します。これはデフォルト設定です。

`-plat|-platform` *platform*

`platform='platform'` を含むフォルダ テンプレートから作成されたフォルダに対するクエリを指定します。プラットフォームの選択肢は、`CCM_HOME¥etc¥om_hosts.cfg` ファイル（Windows）または `$CCM_HOME/etc/om_hosts.cfg` ファイル（UNIX）に定義されます。フォルダ テンプレートが複数のプラットフォームに適用される場合は、プラットフォーム値は設定しません。

`-purpose` *purpose*

指定した目的のクエリを含むタスク クエリによってフォルダを作成するよう指定します。目的の詳細については、`project_purpose` コマンドの説明と用途を参照してください。

通常、このオプションは、`component_task_projects`、`component_task_products`、または `component_task_projects_products` のいずれかの範囲で指定されるコンポーネント タスクのクエリに適用されます。

`-rel|-release` *release_spec*

`release='releasename'` を含むフォルダ テンプレートから作成されたフォルダに対するクエリを指定します。*release_spec* には、複数のリリース値を指定できます。詳細については、リリースの指定を参照してください。

`-sub|-subsystem` *`subsystem`*

`task_subsys='`*`subsystem`*`'` を含むフォルダ テンプレートから作成されたフォルダに対するクエリを指定します。

`-ts|-scope|-task_scope`

タスク クエリを使用するよう指定します。タスク クエリには指定した範囲によって決まるクエリ式が含まれます。指定した範囲に関連付けられるクエリ式は、`-database_id` オプションによっても異なります。以下の範囲を使用できます。

- *`user_defined`*

  この範囲は default_task_query オプションによって定義されます。`-database_id` を指定すると、クエリには、指定したデータベース内で修正可能または完了したタスクのクエリ式も含まれます。

- `all_my_assigned|all_owners_assigned`

  この範囲では、自分に割り当てられているすべてのタスクをクエリします。`-database_id` を指定すると、指定したデータベースで修正可能な、自分に割り当てられているすべてのタスクをクエリします。

- `all_my_assigned_or_completed|all_owners_assigned_or_completed`

  この範囲では、自分に割り当てられているすべてのタスク、または自分が完了したすべてのタスクをクエリします。`-database_id` を指定すると、指定したデータベース内で自分に割り当てられている修正可能なタスク、または指定したデータベース内で自分が完了したすべてのタスクをクエリします。

- `all_my_completed|all_owners_completed`

  この範囲では、自分が完了したすべてのタスクをクエリします。`-database_id` を指定すると、指定したデータベース内で自分が完了したすべてのタスクをクエリします。

- `all_my_tasks|all_owners_tasks`

  この範囲では、自分がタスク担当者であるすべてのタスクをクエリします。`-database_id` を指定すると、自分がタスク担当者であり、指定したデータベース内で修正可能または自分が完了したすべてのタスクをクエリします。

- `all_completed`

  この範囲では、完了したすべてのタスクをクエリします。`-database_id` を指定すると、指定したデータベースで完了したすべてのタスクをクエリします。

- `all_tasks`

  この範囲では、すべてのタスクをクエリします。`-database_id` を指定すると、指定したデータベース内で修正可能または完了したすべてのタスクをクエリします。

- `component_task_projects|component_task_products|`
  `component_task_projects_products`

この範囲では、プロジェクト、製品、またはプロジェクトと製品のコンポーネント タスクをクエリします。-database_id を指定すると、指定したデータベースで作成されたすべてのコンポーネント タスクをクエリします。-purpose を指定すると、指定した目的を持つコンポーネント タスクをクエリします。

-w|-writable (owner | (build_mgr|build_manager|buildmanager) | all | none)

フォルダ テンプレートを使用して作成したフォルダを誰が修正可能かを指定します。指定しなかった場合、デフォルトは owner となり、フォルダの所有者のみが修正できます。

例

- 説明が「%owner's Completed Tasks for Release %release from Database X」であるフォルダ テンプレートを作成し、フォルダ テンプレートがクエリを使用するように設定し、フォルダ クエリを入力する。デフォルト設定が owner であるため、誰がフォルダ テンプレートの書き込みと使用が可能であるかを設定する必要はありません。

  ```
  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"
  ```

- 以下の操作を行い、フォルダ テンプレートがそのフォルダにタスクを含めるために使用するデフォルト クエリを定義する。

  1. 範囲を設定します。
  2. リリース値を設定します。

     パラレル開発とフォルダ テンプレート管理のため、この属性は必ず設定してください。
  3. 必要に応じて、サブシステムを設定します。
  4. 必要に応じて、プラットフォームを設定します。

     1 つのフォルダが複数プラットフォームに適用される場合は、プラットフォーム値を設定する必要はありません。
  5. データベースが DCM 用に初期化されている場合は、データベースを設定します。たとえば、新規フォルダ テンプレートを作成します。このテンプレートから作成されるフォルダは、現在のリリースのすべての完了タスクを収集し、ビルド マネージャによる書き込みが可能となります。

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

関連トピック

- フォルダ テンプレートの削除
- フォルダ テンプレートの一覧表示
- フォルダ テンプレートの管理データベースの設定

フォルダ テンプレートの削除

このサブコマンドにより、フォルダ テンプレートを削除します。このサブコマンドを使用するには、*build_mgr* または *ccm_admin* ロールを持っている必要があります。ただし、システムの定義済みフォルダ テンプレートは、ビルド マネージャが削除することはできません。

```
ccm ft|folder_temp|folder_template -d|-delete folder_template_spec...
```

folder_template_spec

削除するフォルダ テンプレートを指定します。詳細については、フォルダ テンプレートの指定を参照してください。

関連トピック

- フォルダ テンプレートの使用箇所の検索
- フォルダ テンプレートの一覧表示
- フォルダ テンプレート プロパティの表示
- フォルダ テンプレート情報の表示

フォルダ テンプレートの使用箇所の検索

このサブコマンドにより、指定したフォルダ テンプレートを使用するプロセス ルールを検索します。

```
ccm folder -fu|-finduse|-find_use
           [-f|-format format] [-nf|-noformat] ([-ch|-column_header]
           | [-nch|-nocolumn_header]) [-sep|-separator separator]
           ([-sby|-sortby sortspec] | [-ns|-nosort|-no_sort])
           [-gby|-groupby groupformat] [-u|-unnumbered]
           folder_template_spec...
```

`-ch|-column_header`

出力フォーマットでカラム ヘッダーを使用するよう指定します。詳細については、-ch|-column_headers を参照してください。

`folder_spec`

プロジェクトで使用箇所の検索を行うときに、検索対象とするフォルダ テンプレートを指定します。

詳細については、フォルダ テンプレートの指定 を参照してください。

`-f|-format` *`format`*

コマンドの出力フォーマットを指定します。詳細については、-f|-format を参照してください。

`-gby|-groupby` *`groupformat`*

コマンドの出力をグループ化する方法を指定します。詳細については、-gby|-groupby を参照してください。

`-nch|-nocolumn_header`

出力フォーマットでカラム ヘッダーを使用しないよう指定します。詳細については、-ch|-column_headers を参照してください。

`-nf|-noformat`

カラム配置を使用しないよう指定します。詳細については、-nf|-noformat を参照してください。

`-ns|-no_sort`

コマンドの出力をソートしないよう指定します。詳細については、-ns|-nosort を参照してください。

`-sep|-separator` *separator*

異なる区切り文字を指定します。詳細については、-sep|-separator を参照してください。

`-u|-unnumbered`

コマンドの出力の自動番号付けを抑止します（すなわち、出力に番号付けがされません）。詳細については、-u|-unnumbered を参照してください。

例

- フォルダ テンプレート `All completed tasks for release %release` を使用するプロセス ルールを検索する。

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

関連トピック

- フォルダ テンプレートの削除
- フォルダ テンプレートの修正
- フォルダ テンプレートの管理データベースの設定
- フォルダ テンプレート プロパティの表示
- フォルダ テンプレート情報の表示

フォルダ テンプレートの一覧表示

このサブコマンドにより、フォルダ テンプレートを一覧表示します。

```
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`

出力フォーマットでカラム ヘッダーを使用するよう指定します。詳細については、-ch|-column_headers を参照してください。

`-f|-format format`

コマンドの出力フォーマットを指定します。詳細については、-f|-format を参照してください。

`-gby|-groupby groupformat`

コマンドの出力をグループ化する方法を指定します。詳細については、-gby|-groupby を参照してください。

`-nch|-nocolumn_header`

出力フォーマットでカラム ヘッダーを使用しないよう指定します。詳細については、-ch|-column_headers を参照してください。

`-nf|-noformat`

カラム配置を使用しないよう指定します。詳細については、-nf|-noformat を参照してください。

`-ns|-no_sort`

コマンドの出力をソートしないよう指定します。詳細については、-ns|-nosort を参照してください。

`-sep|-separator separator`

異なる区切り文字を指定します。詳細については、-sep|-separator を参照してください。

`-sby|-sortby` *`sortspec`*

コマンドの出力をソートする方法を指定します。詳細については、-sby|-sortby を参照してください。

`-u|-unnumbered`

コマンドの出力の自動番号付けを抑止します（すなわち、出力に番号付けがされません）。詳細については、-u|-unnumbered を参照してください。

例

- すべての個人用フォルダ テンプレートを表示する。

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

- すべてのフォルダ テンプレートを表示する。

  ```
  ccm folder_template -list
  ```

関連トピック

- フォルダ テンプレートの削除
- フォルダ テンプレートの修正
- フォルダ テンプレートの管理データベースの設定

フォルダ テンプレートの修正

このサブコマンドにより、フォルダ テンプレートを修正します。このサブコマンドを使用するには、*build_mgr* または *ccm_admin* ロールを持っている必要があります。ただし、システムの定義済みフォルダ テンプレートは、ビルド マネージャが修正することはできません。

オプションを複数回使用すると、各使用オプションに関するクエリ式が「or」で結合されます。たとえば、`-release 1.0 -release 2.0` を指定すると、クエリ式 `(release='1.0' or release='2.0)` が得られます。

複数のオプションからのクエリ式は「and」で結合されます。たとえば、`-release 1.0 -platform windows` を指定すると、クエリ式 `(release='1.0') and (platform='windows')` が得られます。

```
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`*

指定したカスタム クエリ式を修正したフォルダ テンプレート クエリに含めるよう指定します。

`-desc|-description` *`description`*

フォルダ テンプレートからフォルダを修正するときに、キーワード展開後に使用される文字列を指定します。復帰改行文字を含めることはできません。*`description`* を指定しないと、デフォルト値としてフォルダ テンプレート名が使用されます。キーワード展開の詳細については、フォルダ テンプレートの説明を参照してください。

`-db|-dbid|-database_id` *`database_spec`*

修正するフォルダ テンプレートに関連付けられているデータベース ID を指定します。詳細については、データベースの指定を参照してください。

`folder_template_spec`

修正するフォルダ テンプレートを指定します。詳細については、フォルダ テンプレートの指定 を参照してください。

`-mode ((man|manual) | (uq|use_query))`

フォルダへのタスクの追加を、手動で行うかクエリを使用するかを指定します。Rational Synergy は、手動からクエリ ベースへのモード変更を以下のように行います。

- 手動に設定されたフォルダ テンプレートをクエリ ベースのフォルダ テンプレートに変更する場合で、modify コマンドでクエリも定義されている場合には、その指定されたクエリを使用する。
- 手動に設定されたフォルダ テンプレートをクエリ ベースのフォルダ テンプレートに変更する場合で、コマンドでクエリを指定しなかった場合には、以下のようになる。

  * そのフォルダ テンプレートが以前クエリ ベースであった場合には、最後のクエリを使用する。

  * そのフォルダ テンプレートが以前クエリ ベースであったことがなく、ユーザー定義クエリ (default_task_query) が存在する場合には、そのユーザー定義クエリを使用する。

  * そのフォルダ テンプレートが以前クエリ ベースであったことがなく、ユーザー定義クエリ (default_task_query) が存在しない場合には、**All Tasks Assigned to** *your_user_name* を使用する。

`-must_be_local`

ローカルに作成したプロジェクトの更新プロパティに対して、フォルダ テンプレートがローカル フォルダを使用するよう、フォルダ テンプレートを修正します。

`-nomust_be_local`

ローカルに作成したプロジェクトの更新プロパティに対して、フォルダ テンプレートが非ローカル フォルダを使用するよう、フォルダ テンプレートを修正します。

`-plat|-platform` *`platform`*

フォルダ テンプレート クエリを更新するよう指定します。新規クエリには `platform='platform'` が含まれます。プラットフォームの選択肢は、

CCM_HOME¥etc¥om_hosts.cfg ファイル（Windows）または $CCM_HOME/etc/om_hosts.cfg ファイル（UNIX）に定義されます。フォルダ テンプレートが複数のプラットフォームに適用される場合は、プラットフォーム値は設定しません。

-purpose *purpose*

指定した目的のクエリを含むタスク クエリによってフォルダを作成するよう指定します。目的の詳細については、project_purpose コマンドの説明と用途を参照してください。

通常、このオプションは、component_task_projects、component_task_products、または component_task_projects_products のいずれかの範囲で指定されるコンポーネント タスクのクエリに適用されます。

-rel|-release *release_spec*

新しいクエリによってフォルダ テンプレートを更新するよう指定します。クエリには release='*releasename*' が含まれます。*release_spec* には、複数のリリース値を指定できます。詳細については、リリースの指定 を参照してください。

-sub|-subsystem *subsystem*

新しいクエリによってフォルダ テンプレートを更新するよう指定します。クエリには task_subsys='*subsystem*' が含まれます。

-ts|-scope|-task_scope

タスク クエリを使用するよう指定します。タスク クエリには指定した範囲によって決まるクエリ式が含まれます。指定した範囲に関連付けられるクエリ式は、-database_id オプションによっても異なります。以下の範囲を使用できます。

- *user_defined*

  この範囲は default_task_query オプションによって定義されます。-database_id を指定すると、クエリには、指定したデータベース内で修正可能または完了したタスクのクエリ式も含まれます。

- all_my_assigned|all_owners_assigned

  この範囲では、自分に割り当てられているすべてのタスクをクエリします。-database_id を指定すると、指定したデータベースで修正可能な、自分に割り当てられているすべてのタスクをクエリします。

- all_my_assigned_or_completed|all_owners_assigned_or_completed

  この範囲では、自分に割り当てられているすべてのタスク、または自分が完了したすべてのタスクをクエリします。-database_id を指定すると、指定したデータベース内で自分に割り当てられている修正可能なタスク、または指定したデータベース内で自分が完了したすべてのタスクをクエリします。

- `all_my_completed|all_owners_completed`

  この範囲では、自分が完了したすべてのタスクをクエリします。`-database_id` を指定すると、指定したデータベース内で自分が完了したすべてのタスクをクエリします。

- `all_my_tasks|all_owners_tasks`

  この範囲では、自分がタスク担当者であるすべてのタスクをクエリします。`-database_id` を指定すると、自分がタスク担当者であり、指定したデータベース内で修正可能または自分が完了したすべてのタスクをクエリします。

- `all_completed`

  この範囲では、完了したすべてのタスクをクエリします。`-database_id` を指定すると、指定したデータベースで完了したすべてのタスクをクエリします。

- `all_tasks`

  この範囲では、すべてのタスクをクエリします。`-database_id` を指定すると、指定したデータベース内で修正可能または完了したすべてのタスクをクエリします。

- `component_task_projects|component_task_products|component_task_projects_products`

  この範囲では、プロジェクト、製品、またはプロジェクトと製品のコンポーネント タスクをクエリします。`-database_id` を指定すると、指定したデータベースで作成されたすべてのコンポーネント タスクをクエリします。`-purpose` を指定すると、指定した目的を持つコンポーネント タスクをクエリします。

`-w|-writable (owner | (build_mgr|build_manager|buildmanager) | all | none)`

フォルダ テンプレートの書き込み可能プロパティを更新するよう指定します。フォルダ テンプレートから作成され、フォルダ テンプレートによって管理されるすべてのフォルダが更新され、新しい権限が反映されます。

関連トピック

- フォルダ テンプレートの一覧表示
- **フォルダの修正**
- フォルダ テンプレート プロパティの表示
- フォルダ テンプレート情報の表示

フォルダ テンプレートの管理データベースの設定

このサブコマンドにより、フォルダ テンプレートのローカル管理、ハンドオーバ管理または受取り管理を行います。詳細については、『Rational Synergy Distributed』の「汎用プロセスとリリース固有プロセスの複製」を参照してください。

```
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`*

指定したデータベースからの DCM 更新を受け取るよう指定します。*`database_spec`* には、1 つの DCM データベース定義を指定できます。*`database_spec`* の使用の詳細については、データベースの指定を参照してください。DCM 更新の受取りの詳細については、『Rational Synergy Distributed』の「汎用プロセスとリリース固有プロセスの複製」を参照してください。

`folder_template_spec`

更新するフォルダ テンプレートを指定します。詳細については、フォルダ テンプレートの指定を参照してください。

`-handover` *`database_spec`*

オブジェクトの管理を現在のデータベースから指定したデータベースに渡すよう指定します。DCM データベース定義作成時のデフォルト値は空白の文字列です。これは、管理を特定のハブ データベースを通して特定のスポークに渡すとき、*`database_spec`* 値にハブ *`database_spec`* を指定する必要があることを意味します。指定した *`database_spec`* は、生成が許可されている既知の DCM データベース定義か、または空白の文字列のいずれかです。空白の文字列は、そのデータベースへの管理の譲渡を認めないことを意味します。

このオプションの詳細については、『Rational Synergy Distributed』の「データベースの管理と管理のハンドオーバ」を参照してください。

`-local`

フォルダ テンプレートをローカル管理するよう指定します。オブジェクトは、他のデータベースからの DCM 複製では更新されなくなります。このオプションの詳細については、『Rational Synergy Distributed』の「汎用プロセスとリリース固有プロセスの複製」を参照してください。

例

- フォルダ テンプレート **All system testing tasks for server** の管理をデータベース A1 に渡す。

  ```
  ccm folder_template -controlling_database -handover A1 "All system testing tasks for server"
  ```

関連トピック

- フォルダ テンプレートの使用箇所の検索
- フォルダ テンプレートの一覧表示
- フォルダ テンプレート プロパティの表示
- フォルダ テンプレート情報の表示

フォルダ テンプレート プロパティの表示

このサブコマンドにより、フォルダ テンプレートの特定のプロパティを表示します。

```
ccm ft|folder_temp|folder_template -s|-sh|-show ((desc|description) |
        mode | query | (w|wr|writable)) folder_template_spec...
```

folder_template_spec

表示するフォルダ テンプレートを指定します。詳細については、フォルダ テンプレートの指定を参照してください。

関連トピック

- フォルダ テンプレートの削除
- フォルダ テンプレートの修正

フォルダ テンプレート情報の表示

このサブコマンドにより、フォルダ テンプレートの情報を表示します。

```
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`

出力フォーマットでカラム ヘッダーを使用するよう指定します。詳細については、-ch|-column_headers を参照してください。

`folder_template_spec`

表示するフォルダ テンプレートを指定します。詳細については、フォルダ テンプレートの指定を参照してください。

`-f|-format` *`format`*

コマンドの出力フォーマットを指定します。詳細については、-f|-format を参照してください。

`-gby|-groupby` *`groupformat`*

コマンドの出力をグループ化する方法を指定します。詳細については、-gby|-groupby を参照してください。

`-nch|-nocolumn_header`

出力フォーマットでカラム ヘッダーを使用しないよう指定します。詳細については、-ch|-column_headers を参照してください。

`-nf|-noformat`

カラム配置を使用しないよう指定します。詳細については、-nf|-noformat を参照してください。

`-ns|-no_sort`

コマンドの出力をソートしないよう指定します。詳細については、-ns|-nosort を参照してください。

`-sep|-separator` *separator*

異なる区切り文字を指定します。詳細については、-sep|-separator を参照してください。

`-sby|-sortby` *sortspec*

コマンドの出力をソートする方法を指定します。詳細については、-sby|-sortby を参照してください。

`-u|-unnumbered`

コマンドの出力の自動番号付けを抑止します（すなわち、出力に番号付けがされません）。詳細については、-u|-unnumbered を参照してください。

関連トピック

- フォルダ テンプレートの修正
- フォルダ テンプレート プロパティの表示

説明と用途

フォルダ テンプレートは、フォルダ作成のために使用するパターンとなります。フォルダ テンプレートから作成されたフォルダは、そのフォルダ テンプレートによって管理されます。したがって、フォルダ テンプレートを変更すると、そのフォルダ テンプレートが管理しているフォルダが更新されます。

フォルダ テンプレートの説明

フォルダ テンプレートには、パイプ文字（|）以外の任意の文字を使用して任意の説明を付けることができます。フォルダ テンプレートの説明には、キーワード `%owner`、`%release`、`%database` を任意に組み合わせて入れることができます。フォルダ テンプレートの説明には、必ずしもキーワードを入れる必要はありません。フォルダ テンプレートの説明にキーワードを入れなかった場合、このフォルダ テンプレートから作成されるフォルダにはすべて同じ説明が付けられます。

キーワード %release

たとえば、説明が **Completed Tasks for Release %release** であるフォルダ テンプレートを作成すると、キーワード `%release` はこのフォルダ テンプレートを含むプロセス ルールを使用しているプロジェクトのリリース値に展開されます。キーワード `%release` は、このフォルダ テンプレートからフォルダを作成するときに展開されます。たとえば、**リリース 2.0** のプロジェクトが説明 **Completed Tasks for Release %release** を持つフォルダ テンプレートを含むプロセス ルールを使用すると、このテンプレートから作成されて更新プロパティに追加されるフォルダの説明は、**Completed Tasks for Release 2.0** となります。

キーワード %owner

`%owner` キーワードは、更新プロパティにフォルダ テンプレートから作成されるフォルダが含まれるプロジェクトの所有者に展開されます。たとえば、*jsmith* が所有するリリース **3.1** のプロジェクトが、説明 **%owner's Completed Tasks for Release %release** を持つフォルダ テンプレートを含むプロセス ルールを使用すると、説明が **jsmith's Completed Tasks for Release 3.1** であるフォルダが、このフォルダ テンプレートから作成されてプロジェクトの更新プロパティに追加されます。

キーワード %database

`%database` キーワードは、作成されるフォルダを使用しているプロジェクトが作成されたデータベースの DCM データベース ID に展開されます。たとえば、*jsmith* が所有するリリース **3.1** のプロジェクトが `Bristol` という DCM データベース内にあり、説明が **%owner's Completed Tasks for Release %release from Database %database** であるフォルダ テンプレートを含むプロセス ルールを使用していると、説明が **jsmith's Completed Tasks for Release 3.1 from Database Bristol** であるフォルダがこのフォルダ テンプレートから作成されてプロジェクトの更新プロパティに追加されます。

groups コマンド

詳細については、説明と用途を参照してください。groups コマンドは、以下のサブコマンドをサポートします。

- オブジェクトへのグループの割り当て
- オブジェクトのグループ割り当ての解除

オブジェクトへのグループの割り当て

このサブコマンドにより、1 つまたは複数のグループ制限をオブジェクトに割り当てます。これにより、指定したエントリをオブジェクトの *groups* 属性に追加します。

ファイル、ディレクトリまたはプロジェクトの場合、以下のいずれかに該当する必要があります。

- オブジェクトはそのオブジェクトの最初のバージョンであり、ユーザーはそのオブジェクトに書き込み権限を持っていること。
- ユーザーは *group_mgr* ロールを持ち、オブジェクトの現在のグループのメンバーであること。
- ユーザーは *ccm_admin* ロールを持っていること。

その他のオブジェクトのタイプの場合、以下のいずれかに該当する必要があります。

- ユーザーは *group_mgr* ロールを持ち、オブジェクトの現在のグループのメンバーであること。
- ユーザーは *ccm_admin* ロールを持っていること。

```
ccm groups -a|-assign (-v|-value group_item_list)... object_spec...
```

object_spec...

更新するオブジェクトを指定します。詳細については、オブジェクトの指定を参照してください。

`-v|-value` *group_item_list*

追加するグループ制限を指定します。*group_item_list* は、カンマまたは空白文字、あるいはカンマと空白文字で区切った 1 つまたは複数の項目のリストです。各項目の形式は以下のとおりです。

group_name

group_name:readsource

group_name はグループの名前であり、修正またはチェックアウトのアクセス制限を定義します。*group_name:readsource* はソース属性の可視性に対するアクセス制限を定義します。

例

- グループ **sqe_team** と **design_team** を オブジェクト **makefile.pc-1:makefile:tut70#4** に割り当てる。

  ```
  ccm groups -assign "sqe_team, design_team" makefile.pc-
  1:makefile:tut70#4
  ```

関連トピック

- オブジェクトのグループ割り当ての解除

オブジェクトのグループ割り当ての解除

このサブコマンドにより、オブジェクトから 1 つまたは複数のグループ制限の割り当てを解除します。これにより、オブジェクトの *groups* 属性から指定したエントリを削除します。

ファイル、ディレクトリまたはプロジェクトの場合、以下のいずれかに該当する必要があります。

- オブジェクトはそのオブジェクトの最初のバージョンであり、ユーザーはそのオブジェクトに書き込み権限を持っていること。
- ユーザーは *group_mgr* ロールを持ち、オブジェクトの現在のグループのメンバーであること。
- ユーザーは *ccm_admin* ロールを持っていること。

その他のオブジェクトのタイプの場合、以下のいずれかに該当する必要があります。

- ユーザーは *group_mgr* ロールを持ち、オブジェクトの現在のグループのメンバーであること。
- ユーザーは *ccm_admin* ロールを持っていること。

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

object_spec...

更新するオブジェクトを指定します。詳細については、オブジェクトの指定を参照してください。

`-v|-value` *group_item_list*

追加するグループ制限を指定します。*group_item_list* は、カンマまたは空白文字、あるいはカンマと空白文字で区切った 1 つまたは複数の項目のリストです。各項目の形式は以下のとおりです。

group_name

group_name:readsource

group_name はグループの名前であり、修正またはチェックアウトのアクセス制限を定義します。*group_name:readsource* はソース属性の可視性に対するアクセス制限を定義します。

関連トピック

- オブジェクトへのグループの割り当て

説明と用途

Rational Synergy データベースには、各種オブジェクトを多数含むことができます。該当するロールを持つすべてのユーザーがすべてのオブジェクトを表示、チェックアウトおよび修正するということは、必ずしも適切ではありません。グループ セキュリティの設定により、チェックアウト権限と修正権限を、指定したユーザーのグループに制限できます。さらに、読み出しセキュリティを指定して、オブジェクトのソース内容の可視性を指定したグループに制限することもできます。`groups` コマンドを使用して、オブジェクトのセキュリティを実装して定義します。

グループ マネージャとして作業している場合は、グループ セキュリティを使用して以下の操作が可能です。

- ユーザーのグループに名前を付けて定義する。
- 名前を付けたグループのメンバーであるユーザーを定義、修正する。
- あるオブジェクトに 1 つまたは複数のグループを割り当てることにより、そのオブジェクトへのチェックアウト、読み出し、および修正アクセスを特定のグループに制限する。
- 1 つまたは複数のグループにファイルの読み出しソース管理を割り当てることにより、ファイルのソース内容の可視性を指定した名前のグループに制限する。

developer、*writer*、*component_developer*、*build_mgr* など、オブジェクトを作成できるロールで作業しているユーザーは、そのオブジェクトが唯一のバージョンであり、かつユーザーがそのオブジェクトを修正できる場合、オブジェクトのアクセスを指定グループに制限できます。

読み出しセキュリティは、オブジェクトのソース属性にアクセス コントロールを与えることで実現します。ユーザーは、読み出し制限に関わらず、オブジェクトのクエリを行い、他の属性を見ることができます。読み出しセキュリティは、バージョン管理可能なソース オブジェクトに適用されます。ディレクトリやプロジェクトには適用されません。

> **注記：**ユーザーのワークエリアに存在しているオブジェクトに読み出しセキュリティを適用しても、ファイルはユーザーから読み出し可能です。

読み出しアクセス セキュリティは、3 つのレベルで定義できます。

- ソースへの読み出しアクセス制限がないオブジェクトには、誰でもアクセスできる。
- 1 つまたは複数のグループによる読み出しアクセスが定義されているオブジェクトについては、ユーザーが少なくともそのうちの 1 つのグループのメンバーである場合にのみ、ソースのアクセスが許可される。他のすべてのユーザーは、そのオブジェクトのソース内容へのアクセスを拒否されます。
- 最高レベルのセキュリティ（ソースへのアクセス禁止）が課せられているオブジェクトについては、表示、チェックアウト、および修正が禁止されるが、他の属性は表示できる。ただし、*ccm_admin* ロールで作業しているユーザーは、常にファイルのソース内容を表示できます。

チェックアウトされたすべてのオブジェクトは、読み出しセキュリティ制限も含めて、その祖先と同じグループ セキュリティ制限を継承します。

以下に、セキュリティをオブジェクトに対して適用し、使用する例を示します。

1. グループが存在しない、またはオブジェクトにグループが割り当てられていない場合、何も制限はなく、誰でもソース ファイルを表示、チェックアウト、修正できます。
2. 1 つまたは複数のグループが作成され、1 つのグループがそのオブジェクトに割り当てられた場合、指定されたグループのメンバーのみがファイルを表示、チェックアウト、修正できます。指定されたグループ以外のユーザーでも、ソース オブジェクトの表示のみ可能です。つまり、チェックアウトと修正にセキュリティが導入されますが、読み出しセキュリティはまだ存在していません。
3. 1 つまたは複数のグループが作成され、1 つのグループに読み出しセキュリティ アクセス権（ソース ファイルを表示する権限）が与えられた場合、他のすべてのグループはファイルへの読み出しアクセスができなくなります。このように、読み出しセキュリティ オプションを有効にすると、ソース内容へのアクセス権がデフォルトで定義されます。

読み出しセキュリティは、コピー ベースのワークエリアでのみ使用できます。データベースの読み出しセキュリティ設定の詳細については、該当する『管理者ガイド』を参照してください。Windows の場合は、『Rational Synergy 管理者ガイド Windows 版』を参照してください。UNIX の場合は、『Rational Synergy 管理者ガイド UNIX 版』を参照してください。

グループ セキュリティを使用する 公開状態のディレクトリがあり、ユーザーがどのディレクトリのグループのメンバーでもない場合でも、そのディレクトリでの新規オブジェクトの作成、追加、使用解除が可能です。公開ディレクトリは、ユーザーが相互に変更を簡単に上書きできてしまうことから、その使用には注意が必要です。

DCM でのグループ セキュリティの使用方法の詳細については、『Rational Synergy Distributed』を参照してください。

history コマンド

詳細については、説明と用途を参照してください。`history` コマンドは、オブジェクトの履歴の表示サブコマンドをサポートします。

オブジェクトの履歴の表示

```
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`

出力フォーマットでカラム ヘッダーを使用するよう指定します。詳細については、-ch|-column_headers を参照してください。

`-f|-format format`

コマンドの出力フォーマットを指定します。詳細については、-f|-format を参照してください。

キーワードには、組み込み済みのもの（`%fullname`、`%displayname`、`%objectname`）、あるいは `%modify_time`、`%status` などの既存の属性の名前を使用できます。

キーワードのリストについては、組み込み済みキーワードを参照してください。

`-gby|-groupby groupformat`

コマンドの出力をグループ化する方法を指定します。詳細については、-gby|-groupby を参照してください。

`-nch|-nocolumn_header`

出力フォーマットでカラム ヘッダーを使用しないよう指定します。詳細については、-nch|-nocolumn_headers を参照してください。

`-nf|-noformat`

カラム配置を使用しないよう指定します。詳細については、-nf|-noformat を参照してください。

`-ns|-nosort|-no_sort`

コマンドの出力をソートしないよう指定します。詳細については、-ns|-nosort を参照してください。

object_spec

表示するプロジェクト定義、ファイル定義、ディレクトリ定義またはリリース定義を指定します。

`-p|-project`

プロジェクトの履歴を表示します。

project_spec

履歴を表示するプロジェクトを指定します。詳細については、プロジェクトの指定を参照してください。

`-sby|-sortby` *sortspec*

コマンドの出力をソートする方法を指定します。詳細については、-sby|-sortby を参照してください。

`-sep|-separator` *separator*

`-f|-format` オプションと一緒にのみ使用します。異なる区切り文字を指定します。詳細については、-sep|-separator を参照してください。

例

- 親プロジェクトのワークエリア内部から `main.c` の履歴を確認する。

```
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:
************************************************************
```

関連トピック

- 履歴項目のクリア
- コマンド履歴の表示
- 記録するコマンドの最大数の設定

説明と用途

`history` コマンドにより、プロジェクト、ディレクトリ、ファイルまたはリリースのバージョン履歴を表示します。どのユーザーでもこの操作を実行できます。

ln コマンド

詳細については、説明と用途を参照してください。ln コマンドは、シンボリック リンクの作成サブコマンドをサポートします。

シンボリック リンクの作成

このサブコマンドは、UNIX ユーザーのみが使用できます。

```
ccm ln [-s] [-c|-comment comment_string] [-ce|-commentedit]
       [-cf|-commentfile file_path] [-t|-task task_spec] file_path file_spec
```

`-c|-comment comment`

すべてのベースライン プロジェクトとそのメンバーを *released* 状態にチェックインするとき、それらに追加するコメントを指定します。*comment* には、複数行を含むことができ、円記号でコード化した値を使用できます。

このオプションは、`-commentedit` および `-commentfile` と一緒に使用できます。`-commentedit` オプションを使用した場合、コメントはデフォルトのテキスト エディタで表示されます。

`-ce|-commentedit`

コメントの作成と編集用にデフォルトのテキスト エディタを起動するよう指定します。テキスト エディタで保存した結果が最終的なコメントとして使用されます。このオプションは、`-comment` および `-commentfile` オプションと一緒に使用できます。

`-cf|-commentfile` *file_path*

指定したファイルの内容をコメントとして使用するよう指定します。`-comment` を指定した場合、それがコメントに追加されます。このオプションは、`-commentedit` オプションと一緒に使用できます。

file_spec

シンボリック リンクの名前、または名前とバージョンを指定します。file_spec はコンテキスト プロジェクトと親ディレクトリを与える必要があります。ワークエリア参照形式またはプロジェクト参照形式を使用できます（詳細については、ファイルの指定を参照してください）。バージョンを指定しなかった場合、デフォルトのバージョンが使用されます。

file_path

シンボリック リンクが示す先のパスを指定します。これは管理ワークエリア内で管理されているファイルまたはパスである必要はありません。

`-s`

UNIX 形式の互換性を提供します。それ以外ではオプションは無視されます。

`-t|-task <task_spec>`

シンボリック リンクを指定タスクと関連付けるよう指定します。*task_spec* には、1 つのタスクを指定できます（詳細については、タスクの指定を参照してください）。このオプションを指定しない場合、シンボリック リンクは設定されているカレント タスクに関連付けられます。

例

- `ico_2-1` プロジェクト内の `sort.c` オブジェクトへのシンボリック リンク `sort.c` を作成する。

```
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
```

関連トピック

- work_area コマンド

説明と用途

このサブコマンドにより、データベース内にシンボリック リンク オブジェクトを作成します。管理ワークエリアを持つプロジェクトのコンテキスト内でコマンドを実行する場合、UNIX クライアント上のみで実行できます。シンボリック リンクは、UNIX クライアントによって更新される UNIX ワークエリア内のみで表すことができます。

`ln` コマンドにより、`file_spec` から `path_name` への管理されたシンボリック リンクを作成します。シンボリック リンクは任意のパスを指すことができます。必ずしもプロジェクトの管理ワークエリア内で管理されているオブジェクトまたはパスである必要はありません。

> **注記：**書き込み禁止ディレクトリに新しいリンクを作成する場合は、新しいディレクトリ バージョンが自動的にチェックアウトされます。

共有プロジェクト内で作業しており、現在のディレクトリが書き込み禁止の場合、そのディレクトリはチェックアウトされ、デフォルト（または指定した）タスクと自動的に関連付けられ、*integrate*（統合）状態にチェックインされます。この機能を無効にするには、初期設定ファイル内の shared_project_directory_checkin を FALSE に設定します。

ls コマンド

詳細については、説明と用途を参照してください。ls コマンドは、ファイルの一覧表示サブコマンドをサポートします。

ファイルの一覧表示

`ls` コマンドは、UNIX オペレーティング システムでのみ機能します。

```
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`

出力フォーマットでカラム ヘッダーを使用するよう指定します。詳細については、-ch|-column_headers を参照してください。

`-f|-format` *`format`*

コマンドの出力フォーマットを指定します。詳細については、-f|-format を参照してください。

キーワードには、組み込み済みのもの（`%fullname`、`%displayname`、`%objectname`）、あるいは `%modify_time`、`%status` などの既存の属性の名前を使用できます。

キーワードのリストについては、組み込み済みキーワードを参照してください。

`-gby|-groupby` *`groupformat`*

コマンドの出力をグループ化する方法を指定します。詳細については、-gby|-groupby を参照してください。

`-l`

デフォルトの長いフォーマットを使用するよう指定します。このオプションは、`-f|-format` によって定義フォーマットを指定していなかった場合のみ使用できます。

`-m`

管理ファイルとディレクトリのほか、非管理ファイルとディレクトリも表示します。`-f|-format` オプションによってユーザー定義フォーマットが指定されていない場合、デフォルトのフォーマット（短いフォーマットまたは長いフォーマット）には、以下のように、ファイルの同期状態を示すカラムが含まれます。

- LC（ローカル コピー）

プロジェクト内に存在し、ワークエリアにシンボリック リンクではなくローカル コピーが存在するファイルを意味します。

表示されたファイルにこの記号が付いていて、ワークエリアがリンク ベースの場合は、リコンサイル操作を行ってください。詳細については、reconcile コマンドを参照してください。

- NS（同期されていない）

  プロジェクト内には存在するが、ワークエリアには存在しないファイルを意味します。このような状態は、プロジェクトにファイルを追加したが、ワークエリアが見えない場合、またはファイルのリンクまたはローカル コピーが削除されている場合に発生します。

  ワークエリア内のほとんどのファイルがこの記号付きで表示された場合は、リコンサイル操作を行ってください。詳細については、reconcile コマンドを参照してください。

- UC（非管理）

  ワークエリア内には存在するが、プロジェクト内には存在しないファイルを意味します。非管理の記号の付いたファイルを表示するには、-m オプションに -l オプションをつけて使用する必要があります。

  ユーザー定義フォーマットでは、%Sync キーワードを使用して同期状態を表示できます。

  オブジェクトが作成から 6 ヶ月以上経過している場合、時刻ではなく年が表示されます。

`-nch|-nocolumn_header`

出力フォーマットでカラム ヘッダーを使用しないよう指定します。詳細については、-nch|-nocolumn_headers を参照してください。

`-nf|-noformat`

カラム配置を使用しないよう指定します。詳細については、-nf|-noformat を参照してください。

`-ns|-nosort|-no_sort`

コマンドの出力をソートしないよう指定します。詳細については、-sby|-sortby を参照してください。

`path_or_file_spec`

一覧表示するパスを指定します。*`path_or_file_spec`* には、データベースに定義されているプロジェクト、ディレクトリ、またはファイルを指定できます。また、空

のディレクトリ エントリも可能です。省略した場合、現在の作業ディレクトリが一覧表示されます。詳細については、ファイルの指定を参照してください。

`-p|-project`

プロジェクトの履歴を表示します。

`project_spec`

一覧表示するプロジェクトを指定します。詳細については、プロジェクトの指定を参照してください。

`-R`

サブディレクトリのメンバーを再帰的に表示します。このコマンドはサブプロジェクトには再帰しません。

`-sby|-sortby` *`sortspec`*

コマンドの出力をソートする方法を指定します。詳細については、-sby|-sortby を参照してください。

`-sep|-separator` *`separator`*

`-f|-format` オプションと一緒にのみ使用します。異なる区切り文字を指定します。詳細については、-sep|-separator を参照してください。

例

- 現在のディレクトリを長いフォーマットで一覧表示する。

```
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
```

説明と用途

`ls` コマンドはワークエリア内のディレクトリ オブジェクト バージョンの内容を一覧表示します。デフォルトでは、出力は、ファイル システム内のオブジェクトの一覧とそれらに関連付けられているプロジェクトから構成されます。

`ls` コマンドは、2 つのカテゴリのファイルを表示します。すなわち、Rational Synergy の管理下のオブジェクトとファイル システムのみに存在するファイルです。これらのファイルの表示方法については、`-l` オプションと `-m` オプションを参照してください。

merge コマンド

詳細については、説明と用途を参照してください。merge コマンドは、ファイル／ディレクトリのマージサブコマンドをサポートします。

ファイル／ディレクトリのマージ

このサブコマンドにより、ファイルまたはディレクトリをマージします。

```
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`

すべてのベースライン プロジェクトとそのメンバーを *released* 状態にチェックインするとき、それらに追加するコメントを指定します。`comment` には、複数行を含むことができ、円記号でコード化した値を使用できます。

このオプションは、`-commentedit` および `-commentfile` と一緒に使用できます。`-commentedit` オプションを使用した場合、コメントはデフォルトのテキスト エディタで表示されます。

`-ce|-commentedit`

コメントの作成と編集用にデフォルトのテキスト エディタを起動するよう指定します。テキスト エディタで保存した結果が最終的なコメントとして使用されます。このオプションは、`-comment` および `-commentfile` オプションと一緒に使用できます。

`-cf|-commentfile file_path`

指定したファイルの内容をコメントとして使用するよう指定します。`-comment` を指定した場合、それがコメントに追加されます。このオプションは、`-commentedit` オプションと一緒に使用できます。

`-create_task`

マージされた新規オブジェクト バージョンを作成するときに、Rational Synergy がタスクを自動的に作成し、新規オブジェクト バージョンをタスクに関連付けます。

タスクはマージを実行したユーザーに割り当てられます。タスクのリリース値は、新規オブジェクト バージョンが作成されたプロジェクトのリリース値に設定されます。オブジェクト バージョンがプロジェクトの外部で作成された場合、リリース値は設定されません。

file_spec1

file_spec1 は、マージされる最初のファイルまたはディレクトリの名前を指定します。これは、マージされるオブジェクトのデフォルトの次バージョンを決定するために使用されます。***file_spec1*** が 1 つのファイルまたはディレクトリの ファイルの指定 となるように設定できます。

file_spec2

file_spec2 は、マージされる 2 番目のファイルまたはディレクトリの名前を指定します。***file_spec2*** が 1 つのファイルまたはディレクトリの ファイルの指定 となるように設定できます。***file_spec1*** がファイルであれば、***file_spec2*** もファイルである必要があります。***file_spec1*** がディレクトリであれば、***file_spec2*** もディレクトリである必要があります。

`-t|-task` *task_spec*

マージされた新規バージョンを関連付けるタスクを指定します。`-task` を指定しなかった場合、カレント タスクが設定されていれば、デフォルトでカレント タスクが使用されます。詳細については、タスクの指定 を参照してください。

説明と用途

`merge` コマンドを使用してソース ファイルまたはディレクトリをマージする場合、マージ ツールは選択されたバージョンを比較し、次に最も近い共通祖先に対して各バージョンの相違を比較します。Rational Synergy は、マージ ツールの終了時にマージされた新規管理バージョンを自動的に作成します。

マージ操作は、静的オブジェクトと修正可能な非静的オブジェクトの両方で機能します。パラレル チェックインが抑止されている場合でも、ユーザーはパラレル バージョンをマージして、マージ ツールを使用できます。旧リリースでは、パラレル チェックインは許されず、マージ ツールも起動されませんでした。

マージで自動的に新規タスクを作成するように要求すると、Rational Synergy は *`file_spec1`* が存在するプロジェクトからタスクの `release` 値を取得します。

ファイルのマージ

ソースのマージが可能な各種オブジェクトには、CLI 用と GUI 用に、Rational Synergy の定義済みデフォルト マージ ツールがあります。

自動マージ ツールはファイルの新規管理バージョンを作成します。マージに成功すると、マージ結果が新規ファイルに書き込まれます。

両方のバージョンが祖先に対して相対的に同じ場所が変更されていると、コンフリクトが発生します。マージ済みのファイルにコンフリクトが存在する場合、警告が表示され、コンフリクトを容易に見つけられるようにツールがマークを付け、また自動マージのマージ結果が新規ファイルに書きまれます。

以下に、一時ファイルに対するマーク付けの例を示します。

```
<<<<<<<file1 (file1 changes recommended)
unique lines in file1
======= (common lines)
unique lines in file2
>>>>>>>file2 (file2 changes recommended)
```

ディレクトリのマージ

コマンドラインからは、マージ ツールは、両ディレクトリのすべてのメンバーを、マージされた新規管理ディレクトリに自動的に含めます。マージされるオブジェクトの 1 つが現在のプロジェクトのメンバーである場合、Rational Synergy はプロジェクト内でマージされた新規ディレクトリを使用します。これはルート ディレクトリとサブディレクトリの両方に適用されます。

move コマンド

詳細については、説明と用途を参照してください。move コマンドは、以下のサブコマンドをサポートします。

- プロジェクトの名前変更
- オブジェクトの名前変更／移動

プロジェクトの名前変更

このサブコマンドにより、プロジェクトの名前を変更します。プロジェクト名を変更すると、ルート ディレクトリ名がプロジェクトの新規名を反映して変更されます。

書き込み可能なプロジェクトの名前を変更しようとしたとき、そのルート ディレクトリが書き込み禁止の場合、この操作は失敗します。最初にルート ディレクトリをチェックアウトする必要があります。こうすることで、プロジェクト名を変更すると、Rational Synergy が自動的にルート ディレクトリ名を変更します。

> **注記**：プロジェクトをサブプロジェクトとして使用する場合、名前が変更されたプロジェクトを使用する親ディレクトリをチェックアウトし、名前が変更されたプロジェクトを使用するように更新する必要があります。

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

new_project_spec

名前の変更に使用する名前とバージョン（オプション）を指定します。以下の形式をとることができます。

- 名前
- 名前、バージョン区切り文字、バージョン
- 名前、コロン、バージョン

project_spec

名前を変更するプロジェクトを指定します。詳細については、プロジェクトの指定を参照してください。

`-task` *task_spec*

名前を変更したプロジェクトのルート ディレクトリを関連付けるタスクを指定します。指定しなかった場合、カレント タスクが設定されていれば、デフォルトでカレント タスクが使用されます。詳細については、タスクの指定を参照してください。

オブジェクトの名前変更／移動

このサブコマンドにより、ファイルまたはディレクトリの名前を変更するか、1 つまたは複数のファイル、ディレクトリまたはプロジェクトを別のディレクトリ（他のプロジェクトに属する場合あり）に移動します。

最後の引数に既存のディレクトリが指定されておらず、2 つの引数が存在する場合、このコマンドは名前変更操作と解釈されます。それ以外の場合、このコマンドは移動操作として解釈されます。

ファイルまたはディレクトリの名前を変更するとき、親ディレクトリが書き込み禁止の場合、Rational Synergy は新規の親ディレクトリ バージョンを自動的にチェックアウトし、指定したタスクに関連付けます。タスクが指定されていない場合、デフォルトでカレントタスクが使用されます。ディレクトリを他のユーザーが使用できるようにするには、ディレクトリのチェックイン が必要です。通常、操作に使用されるタスクを完了したときに行います。

オブジェクトを書き込み禁止ディレクトリとの間で移動する場合、親ディレクトリが書き込み禁止であれば、Rational Synergy は新規の親ディレクトリ バージョンを自動的にチェックアウトし、指定したタスクに関連付けます。タスクが指定されていない場合、デフォルトでカレント タスクが使用されます。ディレクトリを他のユーザーが使用できるようにするには、ディレクトリのチェックイン が必要です。通常、操作に使用されるタスクを完了したときに行います。

共有プロジェクト内にいて、親ディレクトリが書き込み禁止の場合、Rational Synergy はそのディレクトリをチェックアウトし、指定タスクまたはカレント タスクと自動的に関連付け、その後 *integrate*（統合）状態にチェックインします。この機能を無効にする場合は、初期設定ファイル内の shared_project_directory_checkin を FALSE に設定します（shared_project_directory_checkin を参照してください）。

プロジェクト参照形式を使用している限り、このコマンドをワークエリアから実行する必要はありません。

> **注記：**他のディレクトリで使用されているファイルまたはディレクトリを名前変更する場合、または、現在のディレクトリでも指定したディレクトリでもないディレクトリ バージョンで使用されているファイルまたは ディレクトリを名前変更する場合は、そのオブジェクトを使用している箇所を確認する必要があります。Rational Synergy は親ディレクトリを更新しますが、他の親ディレクトリは更新しません。新しく名前を変更したオブジェクトを使用するには、これらの他のディレクトリをチェックアウトする必要があります。

```
ccm move [-task task_spec] file_spec... file_spec
```

file_spec

名前変更または移動するファイルまたはディレクトリ、新しい名前または新しい場所を指定します。最後の *file_spec* 引数に既存の管理ディレクトリを指定すると、オブジェクトが現在の場所からそのディレクトリに移動されます。このディレクトリは、移動するオブジェクトと同じプロジェクト内または別のプロジェクト内のどちらでもかまいません。最後の *file_spec* 引数に既存の管理ディレクトリを指定せず、2 つの引数のみを指定した場合、最初の引数によって指定されたファイルまたはディレクトリが 2 番目（最後）の引数によって指定された名前とバージョン（オプション）に変更されます。詳細については、ファイルの指定を参照してください。

`-task` *task_spec*

自動的にチェックアウトされた親ディレクトリが関連付けられているタスクを指定します。タスクを指定しなかった場合、カレント タスクが設定されていれば、デフォルトでカレント タスクが使用されます。詳細については、タスクの指定を参照してください。

例

- ファイル `octagon.h` を `src` ディレクトリから現在のプロジェクト内の `incl` ディレクトリに移動する。

  Windows :
  ```
  ccm move src¥octagon.h incl/
  ```

  UNIX :
  ```
  ccm move src/octagon.h incl/
  ```

  現在のプロジェクトでファイル名を `turquoise.c` から `magenta.c` に変更する。

  ```
  ccm move turquoise.c magenta.c
  ```

- `ccm_aug8-1` プロジェクトを `test_a-1` に名前変更する。

  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’ ...
  ```

- ファイル `hello.c` を `hi_world.c` に名前変更し、次に別のプロジェクトのディレクトリに移動する。

  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
```

- ファイル `hello.c` を `beta-1` から新規プロジェクト `final-1` に移動する。

  Windows :
  ```
  ccm move beta-1¥hello.c@beta-1 final@final-1
  ```
  UNIX :
  ```
  ccm move beta-1/hello.c@beta-1 final@final-1
  ```

説明と用途

`move` コマンドには、以下のような使い方があります。

- ファイルまたはオブジェクトの名前を変更する。プロジェクト名を変更すると、ルート ディレクトリ名がプロジェクトの新規名を反映して変更されます。
- 1 つまたは複数のファイルを別のディレクトリに移動する。
- ファイルを新規プロジェクトに移動する（新規ワークエリア内）。
- 1 つまたは複数のディレクトリとその内容を特定のディレクトリに移動する。
- サブプロジェクトを新しい最上位プロジェクトに移動する。
- 1 つまたは複数のサブプロジェクトとその内容を別のディレクトリに移動する。

process コマンド

詳細については、説明と用途を参照してください。process コマンドは、以下のサブコマンドをサポートします。

- 既存のプロセスへのプロセスのコピー
- 新規プロセスへのプロセスのコピー
- プロセスの作成
- プロセスの削除
- プロセスの一覧表示
- プロセスの修正
- プロセスのプロパティの表示
- プロセス情報の表示
- プロセスのプロセス ルールの表示

既存のプロセスへのプロセスのコピー

このサブコマンドにより、指定したプロセスからプロパティとプロセス ルールを別のプロセスにコピーします。このサブコマンドを使用するには、*build_mgr* または *ccm_admin* ロールを持っている必要があります。

```
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...`*

既存のプロセスにコピーする前に、コピー元のプロセスからコピーした汎用プロセス ルールを追加するよう指定します。

`from_process_spec`

コピーするプロセスを指定します。*`from_process_spec`* には、1 つのプロセスを指定できます。詳細については、プロセスの指定を参照してください。

`to_process_spec`

更新するプロセスを指定します。*`to_process_spec`* には、1 つのプロセスを指定できます。詳細については、プロセスの指定を参照してください。

`-rpr|-remove_process_rule|-remove_process_rules` *`generic_process_rule_spec...`*

既存のプロセスにコピーする前に、コピー元のプロセス内の汎用プロセス ルールを削除するよう指定します。

例

- 標準プロセスを既存のプロセス `custom` にコピーし、その既存プロセスが標準プロセスのプロセス ルールを使用するようにする。

  ```
  ccm process -copy standard -existing custom
  ```

関連トピック

- プロセスの一覧表示
- プロセス情報の表示
- プロセスのプロパティの表示
- プロセスのプロセス ルールの表示

新規プロセスへのプロセスのコピー

このサブコマンドにより、指定したプロセスからプロパティとプロセス ルールを新規プロセスにコピーします。このサブコマンドを使用するには、*build_mgr* または *ccm_admin* ロールを持っている必要があります。

```
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...`

コピー元のプロセスからコピーしたプロセス ルールを新規プロセスに追加するよう指定します。

`-cp|-copy`

2 つの プロセスの指定引数をとります。それぞれを *1* つのオブジェクトを指定できます。

process_spec

コピーするプロセスを指定します。詳細については、プロセスの指定を参照してください。

new_process_spec

作成するプロセスの名前を指定します。詳細については、プロセスの指定を参照してください。

`-rpr|-remove_process_rule|-remove_process_rules process_rule_spec...`

新規のプロセスにコピーする前に、コピー元のプロセス内の汎用プロセス ルールを削除するよう指定します。

例

- Integration Testing（統合テスト）用の新規カスタムプロセスのため、標準プロセスをベースとしてカスタム プロセスを作成する。

  ```
  ccm process -copy standard -new custom -apr "Custom Integration
  Testing"
  ```

関連トピック

- プロセスの一覧表示

- プロセス情報の表示
- プロセスのプロパティの表示
- プロセスのプロセス ルールの表示

プロセスの作成

このサブコマンドにより、指定したプロセス定義とオプションとして指定したダイアグラム URL（uniform resource locator ）を含む、プロセス定義を作成します。このサブコマンドを使用するには、*build_mgr* または *ccm_admin* ロールを持っている必要があります。

```
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`*

新しく作成するプロセス用のダイアグラム URL を指定します。

ダイアグラム URL は、作成するプロセスの詳細情報をもつファイルの URL です。デフォルトで用意されているプロセスには、この詳細情報を含む説明ファイルの URL（ヘルプサーバー上）が設定されています。この説明ファイルの内容は、プロジェクトグルーピングのベースラインとタスクのフローを表示したダイアグラムです。ファイルを表示するには、`-show diagramurl` サブコマンドを使用します。

説明ファイル（ダイアグラム）を用意していない場合は、ファイルを作成した後でその場所への URL を設定できます。

`process_spec`

新しいプロセスの名前を指定します。詳細については、プロセスの指定を参照してください。

`-pr|-prs|-process_rule|-process_rules` *`generic_process_rule_spec...`*

新しいプロセスに追加する汎用プロセス ルールを指定します。

例

- **Insulated Development**（個別開発）、**Integration Testing**（統合テスト）、**Integration Testing**（システム テスト）用の汎用プロセス ルールでカスタム プロセスを作成する。

  ```
  ccm process -create custom -pr "Insulated Development" -pr
  "Integration Testing" -pr "System Testing"
  ```

関連トピック

- プロセスの一覧表示
- プロセス情報の表示
- プロセスのプロパティの表示
- プロセスのプロセス ルールの表示

プロセスの削除

このサブコマンドにより、プロセスを削除します。デフォルトでは、プロセスがどのプロジェクト グルーピングにも使用されていない場合にのみ、そのプロセスを削除できます。このサブコマンドを使用するには、*build_mgr* または *ccm_admin* ロールを持っている必要があります。

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

`-d|-delete`

このコマンドは 1 つまたは複数の *process_spec* の引数をとります。それぞれに複数のプロセス定義を指定できます。

`-force`

プロジェクト グルーピングによって使用されている場合でも、プロセス ルールを削除します。`-force` コマンドを省略すると、プロセス ルールが使用されていない場合にのみ削除されます。

process_spec

削除するプロセスを指定します。詳細については、プロセスの指定を参照してください。

例

- プロセス `custom` を削除する。

  ```
  ccm process -delete custom
  ```

関連トピック

- プロセスの一覧表示
- プロセス情報の表示
- プロセスのプロパティの表示
- プロセスのプロセス ルールの表示

プロセスの一覧表示

このサブコマンドにより、指定した条件と一致するプロセスを一覧表示します。オプションを指定しない場合、すべてのプロセスが一覧表示されます。

```
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`

出力フォーマットでカラム ヘッダーを使用するよう指定します。詳細については、-ch|-column_headers を参照してください。

`-f|-format format`

コマンドの出力フォーマットを指定します。詳細については、-f|-format を参照してください。

キーワードには、組み込み済みのもの（`%fullname`、`%displayname`、`%objectname`）、あるいは `%modify_time`、`%status` などの既存の属性の名前を使用できます。

キーワードのリストについては、組み込み済みキーワードを参照してください。

`-gby|-groupby groupformat`

コマンドの出力をグループ化する方法を指定します。詳細については、-gby|-groupby を参照してください。

`-nch|-nocolumn_header`

出力フォーマットでカラム ヘッダーを使用しないよう指定します。詳細については、-ch|-column_headers を参照してください。

`-nf|-noformat`

カラム配置を使用しないよう指定します。詳細については、-nf|-noformat を参照してください。

`-ns|-nosort|-no_sort`

コマンドの出力をソートしないよう指定します。詳細については、-ns|-nosort を参照してください。

`-sby|-sortby` *`sortspec`*

コマンドの出力をソートする方法を指定します。詳細については、-ns|-nosort を参照してください。

`-sep|-separator` *`separator`*

`-f|-format` オプションと一緒にのみ使用します。異なる区切り文字を指定します。詳細については、-sep|-separator を参照してください。

`-u|-unnumbered`

コマンドの出力の自動番号付けを抑止します（すなわち、出力に番号付けがされません）。詳細については、-u|-unnumbered を参照してください。

例

- すべての定義済みプロセスを一覧表示する。

```
ccm process -list
```

```
Distributed Process
GUI Retrofit Process
Standard Process
XML Source Team Process
```

関連トピック

- 既存のプロセスへのプロセスのコピー
- 新規プロセスへのプロセスのコピー
- プロセスの削除
- プロセスの修正
- プロセスのプロセス ルールの表示

プロセスの修正

このサブコマンドにより、指定したプロセスを修正します。コマンドを正しく実行するには、以下のオプションの少なくとも 1 つを指定する必要があります。

このサブコマンドを使用するには、*build_mgr* または *ccm_admin* ロールを持っている必要があります。

```
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...*

指定した各プロセスに汎用プロセス ルールを追加するよう指定します。このオプションは `-pr` オプションと一緒に使用できませんが、`-rpr` オプションと一緒に使用できます。

`-diagram|-diagramurl` *diagram_url_string*

新しく作成するプロセス用のダイアグラム URL を指定します。

ダイアグラム URL は、作成するプロセスの詳細情報をもつファイルの URL です。デフォルトで用意されているプロセスには、この詳細情報を含む説明ファイルの URL（ヘルプサーバー上）が設定されています。この説明ファイルの内容は、プロジェクトグルーピングのベースラインとタスクのフローを表示したダイアグラムです。ファイルを表示するには、`-show diagramurl` サブコマンドを使用します。

説明ファイル（ダイアグラム）を用意していない場合は、ファイルを作成した後でその場所への URL を設定できます。

generic_process_rule_spec

修正する汎用プロセス ルールを指定します。詳細については、プロセス ルールの指定 を参照してください。

`-n|-name` *name*

削除するプロセスの名前を指定します。

`-pr|-prs|-process_rule|-process_rules` *generic_process_rule_spec...*

各プロセスに設定する完全な汎用プロセス ルール セットを指定します。このオプションは `-apr` オプションまたは `-rpr` オプションと一緒には使用できません。

process_spec

修正するプロセスを指定します。詳細については、プロセスの指定を参照してください。

`-rpr|-remove_process_rule|-remove_process_rules` *generic_process_rule_spec...*

指定した各プロセスから削除する汎用プロセス ルールを指定します。このオプションは `-pr` オプションと一緒に使用できませんが、`-apr` オプションと一緒に使用できます。

例

- 標準プロセスをカスタマイズして、**Integration Testing**（統合テスト）汎用プロセス ルールを **Integration Testing**（統合テスト）プロセス ルールに置き換えます。

  ```
  ccm process -modify standard -rpr "Integration Testing" -apr "Custom
  Integration Testing"
  ```

関連トピック

- プロセスの一覧表示
- プロセス情報の表示
- プロセスのプロパティの表示
- プロセスのプロセス ルールの表示

プロセス情報の表示

このサブコマンドにより、指定したプロセスの情報を表示します。

```
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`

出力フォーマットでカラム ヘッダーを使用するよう指定します。詳細については、-ch|-column_headers を参照してください。

`-f|-format format`

コマンドの出力フォーマットを指定します。詳細については、-f|-format を参照してください。

キーワードには、組み込み済みのもの（`%fullname`、`%displayname`、`%objectname`）、あるいは `%modify_time`、`%status` などの既存の属性の名前を使用できます。

キーワードのリストについては、組み込み済みキーワードを参照してください。

`-nch|-nocolumn_header`

出力フォーマットでカラム ヘッダーを使用しないよう指定します。詳細については、-ch|-column_headers を参照してください。

`-nf|-noformat`

カラム配置を使用しないよう指定します。詳細については、-nf|-noformat を参照してください。

`process_spec`

情報を表示させるプロセスを指定します。詳細については、プロセスの指定を参照してください。

`-sep|-separator separator`

`-f|-format` オプションと一緒にのみ使用します。異なる区切り文字を指定します。詳細については、-sep|-separator を参照してください。

関連トピック

- プロセスの一覧表示
- プロセスのプロパティの表示
- プロセスのプロセス ルールの表示

プロセスのプロパティの表示

このサブコマンドにより、指定したプロセスのプロパティを表示します。

```
ccm process -s|-sh|-show (diagram|diagramurl) process_spec...
```

`-diagram|-diagramurl`

表示するダイアグラム URL を指定します。

ダイアグラム URL は、作成するプロセスの詳細情報をもつファイルの URL です。デフォルトで用意されているプロセスには、この詳細情報を含む説明ファイルの URL（ヘルプサーバー上）が設定されています。この説明ファイルの内容は、プロジェクトグルーピングのベースラインとタスクのフローを表示したダイアグラムです。

ダイアグラム URL の作成はオプションです。すべてのプロセスに表示するダイアグラム URL があるとは限りません。

`process_spec`

プロパティを表示させるプロセスを指定します。詳細については、プロセスの指定を参照してください。

関連トピック

- プロセスの一覧表示
- プロセス情報の表示
- プロセスのプロセス ルールの表示

プロセスのプロセス ルールの表示

このサブコマンドにより、指定したプロセスのプロセス ルールを表示します。

```
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`

出力フォーマットでカラム ヘッダーを使用するよう指定します。詳細については、-ch|-column_headers を参照してください。

`-f|-format format`

コマンドの出力フォーマットを指定します。詳細については、-f|-format を参照してください。

キーワードには、組み込み済みのもの（`%fullname`、`%displayname`、`%objectname`）、あるいは `%modify_time`、`%status` などの既存の属性の名前を使用できます。

キーワードのリストについては、組み込み済みキーワードを参照してください。

`-gby|-groupby groupformat`

コマンドの出力をグループ化する方法を指定します。詳細については、-gby|-groupby を参照してください。

`-nch|-nocolumn_header`

出力フォーマットでカラム ヘッダーを使用しないよう指定します。詳細については、-ch|-column_headers を参照してください。

`-nf|-noformat`

カラム配置を使用しないよう指定します。詳細については、-nf|-noformat を参照してください。

`-ns|-nosort|-no_sort`

コマンドの出力をソートしないよう指定します。詳細については、-ns|-nosort を参照してください。

`-pr|-prs|-process_rule|-process_rules`

表示するプロセス ルールを指定します。

`process_spec`

プロセス ルールを表示させるプロセスを指定します。詳細については、プロセスの指定を参照してください。

`-sby|-sortby` *`sortspec`*

コマンドの出力をソートする方法を指定します。詳細については、-ns|-nosort を参照してください。

`-sep|-separator` *`separator`*

`-f|-format` オプションと一緒にのみ使用します。異なる区切り文字を指定します。詳細については、-sep|-separator を参照してください。

`-u|-unnumbered`

コマンドの出力の自動番号付けを抑止します（すなわち、出力に番号付けがされません）。詳細については、-u|-unnumbered を参照してください。

関連トピック

- プロセスの一覧表示
- プロセス情報の表示
- プロセスのプロパティの表示

説明と用途

プロセスは、いくつかのプロセス ルールを 1 つにまとめて名前を付け、協調動作するようにしたものです。

プロセスは、あるリリース用のプロセスルールを特定するために使用します。組み込み済みのプロセス、プロセス ルール、および目的を下表に示します。

プロセス	プロセス ルール	目的
標準	Collaborative Development（共同開発） Custom Development（カスタム開発） Insulated Development（個別開発） Integration Testing（統合テスト） Shared Development（共有開発） System Testing（システム テスト） Visible Development（可視開発）	Collaborative Development（共同開発） Custom Development（カスタム開発） Insulated Development（個別開発） Integration Testing（統合テスト） Shared Development（共有開発） System Testing（システム テスト） Visible Development（可視開発）
分散型	Custom Development（カスタム開発） Insulated Development（個別開発） Local Collaborative Development （ローカル共同開発） Local Integration Testing（ローカル統合テスト） Master Integration Testing（マスタ統合テスト） Shared Development（共有開発） System Testing（システム テスト） Visible Development（可視開発）	Custom Development（カスタム開発） Insulated Development（個別開発） Collaborative Development（共同開発） Integration Testing（統合テスト） Master Integration Testing（マスタ統合テスト） Shared Development（共有開発） System Testing（システム テスト） Visible Development（可視開発）

独自のプロセスを作成することによって、チーム固有の協業方法を定義できます。たとえば、GUI 開発プロジェクトに携わるチームのビルド マネージャは、GUI Process という名前のプロセスを作成します。このプロセスには、Beta Test という名前の専用目的とそれに対応するプロセスルール Beta Test が含まれます。このプロセスルールは、ベータリリースに対する新しいテストを定義します。つまり、ビルドマネージャは新しい Beta Test プロセスルールを使用してベータリリースのテスト向けのビルドを行います。このプロセスルールは、新しいプロセスである GUI Process にのみ存在します。

process_rule コマンド

詳細については、説明と用途を参照してください。process_rule コマンドは以下のサブコマンドをサポートします。

- プロセス ルールへのフォルダとフォルダ テンプレートの追加
- プロセス ルールの比較
- プロセス ルールのコピー
- プロセス ルールの削除
- プロセス ルールの一覧表示
- プロセス ルールの修正
- プロセス ルールからのフォルダとフォルダ テンプレートの削除
- プロセス ルールの管理データベースの設定
- プロセス ルールのベースライン プロジェクト／フォルダ／フォルダ テンプレート／メンバーの表示
- プロセス ルールのプロパティの表示
- プロセス ルール情報の表示

プロセス ルールへのフォルダとフォルダ テンプレートの追加

このサブコマンドにより、フォルダまたはフォルダ テンプレート、あるいはフォルダとフォルダ テンプレートを、指定したプロセス ルールに追加します。フォルダ テンプレートは汎用プロセス ルールにのみ追加でます。フォルダまたはフォルダ テンプレートを追加して、特定のプロセス ルールをリリースできます。このコマンドを使用するには、プロセス ルール マネージャとして作業している必要があります。

```
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`*

各プロセス ルールに追加されるフォルダを指定します。汎用プロセス ルールはフォルダ テンプレートのみを持つことができます。詳細については、フォルダの指定を参照してください。

`-ft|-folder_temp|-folder_temps|-folder_template|-folder_templates` *`folder_template_spec`*

各プロセス ルールに追加されるフォルダ テンプレートを指定します。詳細については、フォルダの指定を参照してください。

例

- フォルダ 3 を `2.1:Insulated Development` プロセス ルールに追加する。

  ```
  ccm pr -add -folders 3 "2.1:Insulated Development"
  ```

- フォルダ テンプレート `integration tested tasks for release %release` を `2.1:Insulated Development` プロセス ルールに追加する。

  ```
  ccm pr -add -folder_temp "integration tested tasks for release
  %release" "2.1:Insulated Development"
  ```

関連トピック

- プロセス ルールの比較
- プロセス ルールのベースライン プロジェクト／フォルダ／フォルダ テンプレート／メンバーの表示
- プロセス ルールのプロパティの表示
- プロセス ルール情報の表示

プロセス ルールの比較

このサブコマンドにより、2 つのプロセス ルールを比較します。

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

`-comp|-compare` *process_rule_spec1 process_rule_spec2*

比較するプロセス ルールを指定します。詳細については、プロセス ルールの指定を参照してください。

例

- `toolkit/2.0:Collaborative Development` プロジェクトと `toolkit/2.0:Insulated Development` プロジェクトのプロセス ルールを比較する。

  ```
  ccm process_rule -compare "toolkit/2.0:Collaborative Development"
  "toolkit/2.0:Insulated Development"
  ```

関連トピック

- プロセス ルールへのフォルダとフォルダ テンプレートの追加
- プロセス ルールのコピー
- プロセス ルールの削除
- プロセス ルールのベースライン プロジェクト／フォルダ／フォルダ テンプレート／メンバーの表示
- プロセス ルールのプロパティの表示
- プロセス ルール情報の表示

プロセス ルールのコピー

このサブコマンドにより、プロセス ルールを別のプロセス ルールにコピーします。このコマンドを使用するには、プロセス ルール マネージャとして作業している必要があります。以下のルールが適用されます。

- 汎用から汎用にコピー

  汎用プロセス ルールを既存の汎用プロセス ルールにコピーすると、コピー先のプロセス ルールは元の名前（4 部名称と case_preserved_name 属性）を保持しますが、他のプロパティはすべてコピー元のプロセス ルールからコピーされます。汎用プロセス ルールを新規汎用プロセス ルールにコピーできます。

- 汎用からリリース固有にコピー

  汎用プロセス ルールを既存のリリース固有プロセス ルールにコピーすると、コピー先のプロセス ルールは元の名前（4 部名称、case_preserved_name 属性、および release 属性）、および汎用プロセス ルールとの元の関係を保持します。他のプロパティはすべて、コピー元のプロセス ルールからコピーされます。

- リリース固有からリリース固有にコピー

  リリース固有プロセス ルールを既存のリリース固有プロセス ルールにコピーすると、コピー先のプロセス ルールは元の名前（4 部名称、case_preserved_name 属性、および release 属性）を保持しますが、他のプロパティはすべてコピー元のプロセス ルールからコピーされます。コピー先のリリース固有プロセス ルールは、汎用プロセス ルールとの既存の関係も保持します。

- リリース固有から汎用にコピー

  リリース固有プロセス ルールは、汎用プロセス ルールにコピーできません。

```
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`

1 番目のプロセス ルールのベースライン ルールを 2 番目のプロセス ルールにコピーします。ベースライン ルールには、ベースライン選択モード、release:purpose ペアのベースライン検索リスト、プロセス ルールで定義された固有のベースライン、プロジェクト バージョン一致文字列が含まれます。

指定するプロセス ルールはリリース固有のプロセス ルールである必要があります。

from_process_rule_spec

コピーするプロセス ルールを指定します。詳細については、プロセス ルールの指定を参照してください。

to_process_rule_spec

更新するプロセス ルールを指定します。詳細については、プロセス ルールの指定を参照してください。

from_rspr_spec

コピーするリリース固有のプロセス ルールを指定します。詳細については、プロセス ルールの指定を参照してください。

to_rspr_spec

更新するリリース固有のプロセス ルールを指定します。詳細については、プロセス ルールの指定を参照してください。

例

- **2.0:Insulated Development** プロセス ルールを既存の **2.1:Insulated Development** プロセス ルールに上書きコピーする。

  ```
  ccm process_rule -copy "2.0:Insulated Development" "2.1:Insulated
  Development"
  ```

関連トピック

- プロセス ルールの比較
- プロセス ルールのベースライン プロジェクト／フォルダ／フォルダ テンプレート／メンバーの表示
- プロセス ルールのプロパティの表示
- プロセス ルール情報の表示

プロセス ルールの削除

このサブコマンドにより、プロセス ルールを削除します。デフォルトでは、プロセス ルールがどのプロジェクト グルーピングにも使用されていない場合にのみ、そのプロセス ルールを削除できます。使用中のプロセス ルールを削除するには、`-force` オプションを使用します。このコマンドを使用するには、プロセス ルール マネージャ として作業している必要があります。

```
ccm pr|process_rule|ut|update_temp|update_template|rt|recon_temp|
      reconfigure_template -d|-delete [-force] process_rule_spec...
```

`-d|-delete`

このコマンドは 1 つまたは複数の *process_rule_spec* 引数を受け入れます。各引数に複数のオブジェクトを指定できます。

`-force`

プロセス ルールがプロジェクト グルーピングによって使用されている場合でも、プロセス ルールを削除します。`-force` コマンドを省略すると、プロセス ルールが削除されるのは、そのプロセス ルールが使用されていない場合だけです。

process_rule_spec

削除するプロセス ルールを指定します。詳細については、プロセス ルールの指定を参照してください。

例

- **2.1:Shared** プロセス ルールを削除する。

  ```
  ccm pr -delete "2.1:Shared"
  ```

関連トピック

- プロセス ルールの比較
- プロセス ルールのベースライン プロジェクト／フォルダ／フォルダ テンプレート／メンバーの表示
- プロセス ルールのプロパティの表示
- プロセス ルール情報の表示

プロセス ルールの一覧表示

このサブコマンドにより、指定した条件と一致するプロセス ルールを一覧表示します。オプションを指定しない場合、すべてのプロセス ルールが一覧表示されます。

```
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`

出力フォーマットでカラム ヘッダーを使用するよう指定します。詳細については、-ch|-column_headers を参照してください。

`-f|-format format`

コマンドの出力フォーマットを指定します。詳細については、-f|-format を参照してください。

キーワードには、組み込み済みのもの（`%fullname`、`%displayname`、`%objectname`）、あるいは `%modify_time`、`%status` などの既存の属性の名前を使用できます。

キーワードのリストについては、組み込み済みキーワードを参照してください。

`-gby|-groupby groupformat`

コマンドの出力をグループ化する方法を指定します。詳細については、-gby|-groupby を参照してください。

`-nch|-nocolumn_header`

出力フォーマットでカラム ヘッダーを使用しないよう指定します。詳細については、-ch|-column_headers を参照してください。

`-nf|-noformat`

カラム配置を使用しないよう指定します。詳細については、-nf|-noformat を参照してください。

`-ns|-nosort|-no_sort`

コマンドの出力をソートしないよう指定します。詳細については、-ns|-nosort を参照してください。

`-sby|-sortby` *`sortspec`*

コマンドの出力をソートする方法を指定します。詳細については、-ns|-nosort を参照してください。

`-sep|-separator` *`separator`*

`-f|-format` オプションと一緒にのみ使用します。異なる区切り文字を指定します。詳細については、-sep|-separator を参照してください。

`-u|-unnumbered`

コマンドの出力の自動番号付けを抑止します（すなわち、出力に番号付けがされません）。詳細については、-u|-unnumbered を参照してください。

例

- すべての定義済みプロセス ルールを一覧表示する。

```
ccm process_rule -list
```

```
Collaborative Development
Insulated Development
Custom Development
Shared Development
Visible Development
Integration Testing
System Testing
1.0:Integration Testing
1.0:System Testing
1.0:Insulated Development
2.0:Integration Testing
2.0:System Testing
2.0:Collaborative Development
2.0:Insulated Development
Local Collaborative Development
Local Integration Testing
Master Integration Testing
```

- リリース **1.0** のプロセス ルールを一覧表示する。

```
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
```

- **Integration Testing**（統合テスト）のプロセス ルールを一覧表示する。

  ```
  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
  ```

- **System Testing**（システム テスト）と **Integration Testing**（統合テスト）のプロセス ルールを一覧表示する。

  ```
  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
  ```

関連トピック

- プロセス ルールの比較
- プロセス ルールの削除
- プロセス ルールの修正
- プロセス ルールのベースライン プロジェクト／フォルダ／フォルダ テンプレート／メンバーの表示
- プロセス ルールのプロパティの表示
- プロセス ルール情報の表示

プロセス ルールの修正

このサブコマンドにより、指定したプロセス ルールを修正します。このコマンドを使用するには、プロセス ルール マネージャとして作業している必要があります。

```
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`

`-brp|-baseline_release_purpose|-baseline_release_purposes` オプションと一緒に使用することで、現在のリリース／目的ペア リストにリリースを追加するよう指定します。

`-brp|-baseline_release_purpose|-baseline_release_purposes` *`release_purposes`*

プロセス ルールのベースライン リリース／目的ペアを指定します。ベースライン リリース／目的リストは、プロセス ルールが最新のベースライン選択モードを使用するときに使用されます。リストの順序は重要です。最新のベースライン検索モードでは、更新時に最初のリリース／目的と一致するベースラインが検索されます。一致するものが見つからなければ、2 番目のリリース／目的と一致するベースラインを検索します。

`release_purposes` 値は、release_spec、コロン（:）、および目的名で構成される 1 つまたは複数の項目のリストです。*`release_spec`* には 1 つのリリース、あるいはキーワード `%release` または `%baseline_release` を指定できます。`%release` キーワードはプロセス ルールの現在のリリースを意味します。`%baseline_release` キーワードは、プロセス ルールのリリースのベースライン リリースを意味します。目的名は定義済みの目的である必要があります。

`-ap|-append` を指定すると、指定したリリース／目的ペアが現在のリストに追加されます。`-pr|-prepend` を指定すると、指定したリリース／目的ペアが現在のリストの先頭に追加されます。いずれのオプションも指定しなかった場合は、指定リリース／目的は現在のリストになります。

`-bn|-baseline_name` *`baseline_spec`*

プロセス ルールが指定したベースラインの選択モードを使用するよう指定します。*`baseline_spec`* には、1 つのベースラインを指定できます。

`-fol|-folder|-folders` *`folder_spec`*

各プロセス ルールから削除するフォルダを指定します。汎用プロセス ルールはフォルダ テンプレートのみを持つことができます。

`-ft|-folder_temp|-folder_temps|-folder_template|-folder_templates` *`folder_template_spec`*

各プロセス ルールから削除するフォルダ テンプレートを指定します。

`-lb|-latest_baseline`

プロセス ルールが最新のベースラインを使用するよう指定します。このプロセス ルールを使用するプロジェクト グルーピングが更新されると、指定したベースラインのリリース／目的ペア リストに一致する最新のベースラインが検索されます。

`-lbp|-latest_baseline_projects`

プロセス ルールが最新のベースライン プロジェクトを使用するよう指定します。プロジェクト グルーピングがこのプロセス ルールを使用している場合、プロジェクトが更新されると、バージョン一致基準と prep ベースライン基準に一致した最新のプロジェクトがベースライン プロジェクトとして選択されます。

`-lsp|-latest_static_projects`

プロセス ルールが最新の静的プロジェクトを使用するよう指定します。このオプションは `-pb|-prep_baseline` オプションと一緒には使用できません。

`-lsbmp|-latest_static_or_build_management_projects`

プロセス ルールが最新の静的プロジェクトまたはビルド管理プロジェクトを使用するよう指定します。このオプションは `-nopb|-noprep_baseline` オプションと一緒には使用できません。

`-matching` *`version_matching_string`*

プロセス ルールが最新のベースライン プロジェクトの選択モードを使用している場合、候補のベースライン プロジェクトのバージョンと照合するための追加の基準を指定します。

ベースラインを識別するために使用可能なバージョンを入力できます。これは、同じリリース値を持つプロジェクトのリリース バージョンが複数あるので、ベースラインのリリース指定だけでは十分でない場合に指定します。

たとえば、3 つのリリース済みプロジェクト階層があり、すべてリリース 1.0 の場合、プロジェクト バージョンは **1.0_alpha**、**1.0_beta**、および **1.0_gr** となります。この場合、Baseline Release オプションを **1.0** として指定しただけでは、このプロセス ルールを使用するプロジェクトを識別できません。バージョンが **1.0_gr** のプロジェクトがベースラインとして使用されていることを示すには、Baseline Versions Matching オプションを **1.0_gr** に設定する必要があります。

1.0_gr プロジェクト階層内のすべてのベースラインが同じではないが、類似のバージョンを持っている場合は、ワイルドカードを指定できます。たとえば、プロジェクト階層にバージョンが **1.0_gr**、**1.0_gr_unix**、および **1.0_gr_windows** のプロジェクトがある場合は、Baseline Versions Matching オプションを `1.0_gr*` に設定します。ここでは、接頭辞 **1.0_gr** 付きのバージョンを選択します。バージョンの他の部分が異なっていてもかまいません（プロジェクトで、複数のベースラインの選択肢がある場合は、プラットフォームが一致するベースラインを選択します。たとえば、プロジェクト **2.0_int_unix** は可能性のあるベースラインとして **1.0_gr_unix** と **1.0_gr_windows** を識別するかもしれませんが、一致するプラットフォームを探して、**1.0_gr_unix** を使用します。これは、Rational Synergy がデフォルトでパラレル プラットフォームの展開をサポートするように設定されているためです）。

`-modify`

既存のプロセス ルールのプロパティを修正するよう指定します。このサブコマンドは 1 つまたは複数の プロセス ルールの指定 引数をとります。それぞれに複数のオブジェクトを指定できます。このサブコマンドは、1 つのベースライン オブジェクトを指定できる ベースラインの指定形式をとる、`-bn|-baseline_name` オプションをとります。このオプションを使用した結果、クエリ選択セットは更新されません。

`-nopb|-noprep_baseline`

このオプションが有効なのは、プロセス ルールにベースライン選択モードとして `latest_baseline_projects` がある場合だけです。これは、*prep* 状態のプロジェクトが、このプロセス ルールを使用する個々のプロジェクトの潜在的なベースライン プロジェクトとして考慮されないことを示します。代わりに `-lsp|-latest_static_projects` オプションを使用します。

`-pb|-prep_baseline`

このオプションが有効なのは、プロセス ルールにベースライン選択モードとして `latest_baseline_projects` がある場合だけです。これは、prep 状態のプロジェクトが、このプロセス ルールを使用する個々のプロジェクトの潜在的なベースライン プロジェクトとして考慮されることを示します。代わりに `-lsbmp|-latest_static_or_build_management_projects` オプションを使用します。

process_rule_spec

修正するプロセス ルールを指定します。詳細については、プロセス ルールの指定を参照してください。

pr|-prepend

-brp|-baseline_release_purpose|-baseline_release_purposes オプションと一緒に使用することで、指定したリリース／目的ペアを現在のリストの先頭に追加するよう指定します。

-usb|-user_selected_baseline

プロセス ルールが、ベースライン プロジェクトを検出するために使用されるベースラインを指定しないことを示します。ベースラインはユーザーが選択します。

例

- 最新のベースラインを使用するために 2.1:Insulated Development プロセス ルールを設定する。

  ```
  ccm pr -m "2.1:Insulated Development" -latest_baseline
  ```

- 指定したリリースと目的の組み合わせを持つ最新のベースライン プロジェクトを使用するために 2.1:Insulated Development プロセス ルールを設定する。

  ```
  ccm pr -m "2.1:Insulated Development" -latest_baseline_projects -
  baseline_release_purpose "2.1:Integration Testing,2.1:System
  Testing,2.0:Any"
  ```

- 特定のプロセス ルールがベースラインを検索するために使用するリリース／目的ペアのリストを修正する。

  ```
  ccm pr -modify -baseline_release_purposes "2.0:Any,1.0:System Testing"
  -prepend "2.0:Integration Testing"
  ```

- *process_rule_spec* が 2.0:Insulated Development であるプロセス ルールの、Build_1234_int という名前のベースラインを選択する。

  ```
  ccm process_rule -modify -bn Build_1234_int "2.0:Insulated
  Development"
  ```

関連トピック

- プロセス ルールの比較
- プロセス ルールの削除
- プロセス ルールのベースライン プロジェクト／フォルダ／フォルダ テンプレート／メンバーの表示

- プロセス ルールのプロパティの表示
- プロセス ルール情報の表示

プロセス ルールからのフォルダとフォルダ テンプレートの削除

このサブコマンドにより、フォルダまたはフォルダテンプレート、またはその両方を指定したプロセスルールから削除します。汎用プロセス ルールはフォルダ テンプレートのみを持つことができます。このコマンドを使用するには、プロセス ルール マネージャとして作業している必要があります。

```
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`*

各プロセス ルールから削除するフォルダを指定します。汎用プロセス ルールはフォルダ テンプレートのみを持つことができます。

`-ft|-folder_temp|-folder_temps|-folder_template|-folder_templates` *`folder_template_spec`*

各プロセス ルールから削除するフォルダ テンプレートを指定します。

`process_rule_spec`

更新するプロセス ルールを指定します。詳細については、プロセス ルールの指定を参照してください。

関連トピック

- プロセス ルールの比較
- プロセス ルールのベースライン プロジェクト／フォルダ／フォルダ テンプレート／メンバーの表示
- プロセス ルールのプロパティの表示
- プロセス ルール情報の表示

プロセス ルールの管理データベースの設定

このサブコマンドは、DCM に対して、指定データベースに管理をハンドオーバするか、指定データベースからの更新のみを受け取るか、または、ローカル管理を行うように、設定します。

```
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`*

指定したデータベースからの DCM 更新を受け取るよう指定します。*`database_spec`* には、1 つのデータベース定義を指定できます。*`database_spec`* の使用の詳細については、データベースの指定を参照してください。DCM 更新の受け入れの詳細については、『Rational Synergy Distributed』の「汎用プロセスとリリース固有プロセスの複製」を参照してください。

`-handover` *`database_spec`*

オブジェクトの管理を現在のデータベースから指定したデータベースに渡すよう指定します。DCM データベース定義作成時のデフォルト値は空白の文字列です。これは、管理を特定のハブ データベースを通して特定のスポークに渡すとき、*`database_spec`* 値にハブ *`database_spec`* を指定する必要があることを意味します。指定した *`database_spec`* は、生成が許可されている既知の DCM データベース定義か、または空白の文字列のいずれかです。空白の文字列は、そのデータベースへの管理の譲渡を認めないことを意味します。

このオプションの詳細については、『Rational Synergy Distributed』の「データベースの管理と管理のハンドオーバ」を参照してください。

`-local`

指定したプロセス ルールをローカルに管理するよう指定します。

例

- `2.1-patch1:Insulated Development` プロセス ルールを使用するための管理データベースを設定する。

  ```
  ccm pr -controlling_database -accept A "2.1-patch1:Insulated
  Development"
  ```

プロセス ルールのベースライン プロジェクト／フォルダ／フォルダ テンプレート／メンバーの表示

このサブコマンドにより、指定したプロセス ルールのベースライン プロジェクト、フォルダ、フォルダ テンプレート、またはメンバー オブジェクトを表示します。どのユーザーでもこのコマンドを実行できます。

```
ccm pr|process_rule|ut|update_temp|update_template|rt|recon_temp|
        reconfigure_template -s|-sh|-show (baseline_projects |
        (fol|folder|folders) |
        (ft|folder_temp|folder_temps|folder_template|folder_templates) |
        members) [-f|-format format] [-nf|-noformat]
        ([-ch|-column_header] | [-nch|-nocolumn_header])
        [-sep|-separator separator] ([-sby|-sortby sortspec] |
        [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat] [-u|-unnumbered]
        process_rule_spec...
```

[-ch|-column_header](#)

[-f|-format format](#)

[-gby|-groupby groupformat](#)

[-nch|-nocolumn_header](#)

[-ns|-nosort|-no_sort](#)

[-sby|-sortby sortspec](#)

[-sep|-separator separator](#)

s|-sh|-show
(baseline_projects|(fol|folder|folders)|(ft|folder_temp|folder_temps|folder_template|folder_templates)|members)

プロセス ルールの表示したい関連オブジェクトを指定します。

- baseline_projects ：プロセス ルールのベースライン プロジェクトを表示します。
- fol|folder|folders ：プロセス ルールのフォルダ メンバーを表示します。
- ft|folder_temp|folder_temps|folder_template|folder_templates ：プロセス ルールのフォルダ テンプレート メンバーを表示します。

- `members` :プロセス ルールのフォルダとフォルダ テンプレートの両方のメンバーを表示します。

`-u|-unnumbered`

関連トピック

- プロセス ルールの比較
- プロセス ルールの削除
- プロセス ルールのプロパティの表示
- プロセス ルール情報の表示

プロセス ルールのプロパティの表示

このサブコマンドにより、指定したプロセス ルールのプロパティを表示します。どのユーザーでもこのコマンドを実行できます。

```
ccm pr|process_rule|ut|update_temp|update_template|rt|recon_temp|
        reconfigure_template -s|-sh|-show
        ((brp|baseline_release_purpose|baseline_release_purposes) |
        (bsm|baseline_selection_mode) | matching | (pb|prep_baseline))
        process_rule_spec...
```

`-s|-sh|-show (brp|baseline_release_purpose|baseline_release_purposes)`

ベースライン リリース／目的リストを表示します。

`-s|-sh|-show (bsm|baseline_selection_mode)`

ベースライン選択モードを表示します。

`-s|-sh|-show matching`

バージョン一致プロパティを表示します。

`-s|-sh|-show pb|prep_baseline`

prep ベースライン プロジェクトが可能かどうかを表示します。

関連トピック

- プロセス ルールの比較
- プロセス ルールの削除
- プロセス ルールのベースライン プロジェクト／フォルダ／フォルダ テンプレート／メンバーの表示
- プロセス ルール情報の表示

プロセス ルール情報の表示

このサブコマンドにより、指定したプロセス ルールの情報を表示します。どのユーザーでもこのコマンドを実行できます。

```
ccm pr|process_rule|ut|update_temp|update_template|rt|recon_temp|
        reconfigure_template -s|-sh|-show (i|info|information)
        -f|-format format [-nf|-noformat]
        ([-ch|-column_header] | [-nch|-nocolumn_header])
        [-sep|-separator separator] process_rule_spec...
ccm pr|process_rule|ut|update_temp|update_template|rt|recon_temp|
        reconfigure_template -s|-sh|-show (i|info|information)
        process_rule_spec...
```

-ch|-column header

-f|-format format

-gby|-groupby groupformat

-nch|-nocolumn header

-ns|-nosort|-no sort

-sby|-sortby sortspec

-sep|-separator separator

-u|-unnumbered

例

- 2.1:Insulated Development プロセス ルールのプロパティを表示する。

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

関連トピック

- プロセス ルールの比較
- プロセス ルールの削除
- プロセス ルールのベースライン プロジェクト／フォルダ／フォルダ テンプレート／メンバーの表示
- プロセス ルールのプロパティの表示

説明と用途

process_rule コマンドにより、プロセス ルールを表示、設定します。process_rule コマンドは、旧リリースでは update_template および reconfigure_template コマンドと呼ばれていました。

プロセス ルールは、プロジェクト グルーピングの更新プロパティとプロジェクトの更新プロパティのためのベースライン選択の方法を指定します。プロセス ルールは以下の選択モードをサポートします。

- **最新のベースライン**

  プロセス ルールのベースライン リリース／目的ペアリストに一致する最新のベースラインが、プロジェクト グルーピングのベースラインとして選択されます。プロジェクトの場合、そのベースライン内で対応するプロジェクトがベースライン プロジェクトとして使用されます。

- **固有のベースライン**

  プロセス ルールに指定されたベースラインが、プロジェクト グルーピングに使用されます。プロジェクトの場合、そのベースライン内で対応するプロジェクトがベースライン プロジェクトとして使用されます。

- **ユーザーが選択したベースライン**

  プロセス ルールはベースラインを指定しません。プロジェクト グルーピングの更新プロパティのためのベースラインを手動で選択する必要があります。プロジェクトの場合、そのベースライン内で対応するプロジェクトがベースライン プロジェクトとして使用されます。

- **最新のベースライン プロジェクト**

  ベースライン プロジェクトは、リリースまたはベースライン リリースに一致し、プロジェクト バージョンについても指定したバージョン一致基準に一致する最新のベースライン プロジェクトを取得することで選択されます。

プロセス ルールは、プロジェクトで更新操作を実行したときに、プロジェクトをどのように更新するかを指定します。プロジェクトの目的とリリース値の組み合わせにより、プロジェクトが使用できるプロセス ルールが決定します。リリース／目的の 1 つのペアについて、複数のプロセス ルールを作成できます。そのため、複数のルールを設定しておき、あるリリースと目的に適用するルールを選択したり、リリース中にプロセス ルールを切り替えたりすることが可能です。また、目的ごとにプロセス ルールを修正せずに、プロセス ルールを以降のリリースで再利用することも可能です。

以下のいずれかの状況が発生した場合に、プロセス ルールが自動作成されます。

- ビルド マネージャが特定のプロセスを使って新規リリースを作成すると、リリース固有プロセス ルールが各汎用プロセスルールについて作成される。もしビルド マネージャが汎用プロセス ルールを使ってリリースを作成すると、リリース固有プロセス ルールが各汎用プロセスルールについて作成される。

- ビルド マネージャがある特定のリリース値についての有効な目的のリストに新規目的を追加すると、そのプロジェクトの目的とリリース値の固有の組み合わせごとに 1 つのプロセス ルールが作成される。
- ビルド マネージャが汎用プロセス ルールをリリースに追加すると、リリース固有プロセス ルールがその汎用プロセス ルールから作成される。
- ビルド マネージャが新規目的を作成すると、その目的の汎用プロセス ルールが作成される。その後、ビルド マネージャはこの新規（空白）プロセス ルールを編集する必要があります。
- ビルド マネージャがプロセス ルールをコピーしたとき。
- ビルド マネージャが新しい汎用プロセス ルール名を使用して汎用プロセス ルールをコピーしたとき。

汎用プロセス ルールは、標準および分散プロセス用の製品に組み込まれています。DCM 用に初期化されていないデータベースには標準プロセス、DCM 用に初期化されたデータベースには標準プロセスと分散プロセスの両方が入っています。これらのプロセス ルールは以下の場合を除いて、同等の動作をします。

- 標準プロセスでは、Collaborative Development（共同開発）がすべてのデータベースからすべての完了タスクを収集するのに対し、分散プロセスでは Local Collaborative Development（ローカル共同開発）がローカル データベースからすべての完了タスクを収集する。
- 標準プロセスでは、Integration Testing（統合テスト）がすべてのデータベースからすべての完了タスクを収集するのに対し、分散プロセスでは Local Integration Testing（ローカル統合テスト）がローカル データベースからすべての完了タスク、および外部データベースからマスタ統合テスト済みタスクを収集する。

標準プロセスは、目的の指定時に Rational Synergy Classic および CLI にプロセス ルールを提供するために使用します。

新規リリースの作成時に、使用するプロセス ルールを指定できます。詳細については、release コマンドを参照してください。

project コマンド

詳細については、説明と用途を参照してください。project コマンドは、プロジェクト情報の表示サブコマンドをサポートします。

プロジェクト情報の表示

```
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`

出力フォーマットでカラム ヘッダーを使用するよう指定します。詳細については、-ch|-column_headers を参照してください。

`file_spec`

親ディレクトリが関連プロジェクトの決定に使用されるオブジェクトを指定します。省略した場合、現在の作業ディレクトリがデフォルトとなります。詳細については、ファイルの指定を参照してください。

`-f|-format` *`format`*

コマンドの出力フォーマットを指定します。詳細については、-f|-format を参照してください。

`-gby|-groupby` *`groupformat`*

コマンドの出力をグループ化する方法を指定します。詳細については、-gby|-groupby を参照してください。

`-nch|-nocolumn_header`

出力フォーマットでカラム ヘッダーを使用しないよう指定します。詳細については、-nch|-nocolumn_headers を参照してください。

`-nf|-noformat`

カラム配置を使用しないよう指定します。詳細については、-nf|-noformat を参照してください。

`-ns|-no_sort`

コマンドの出力をソートしないよう指定します。詳細については、-ns|-nosort を参照してください。

`-sep|-separator` *separator*

異なる区切り文字を指定します。詳細については、-sep|-separator を参照してください。

`-sby|-sortby` *sortspec*

コマンドの出力をソートする方法を指定します。詳細については、-sby|-sortby を参照してください。

`-u|-unnumbered`

コマンドの出力の自動番号付けを抑止します（すなわち、出力に番号付けがされません）。詳細については、-u|-unnumbered を参照してください。

デフォルトのフォーマットは `%displayname` です。このサブコマンドでは、クエリ選択セットは更新されません。

例

- `$HOME/ccm_wa/database/example-1/example/doc/readme.txt` ワークエリア パスに関連付けられているプロジェクトを決定する。

  ```
  ccm project $HOME/ccm_wa/database/example-1/example/doc/readme.txt
  example-1
  ```

関連トピック

- フォーマットの使用例

説明と用途

`project` コマンドは、指定した *`file_spec`* または現在の作業ディレクトリに関連付けられているプロジェクトを決定します。*`file_spec`* は通常は管理ワークエリア内のワークエリア参照形式です。

project_grouping コマンド

詳細については、説明と用途を参照してください。`project_grouping` コマンドは以下のサブコマンドをサポートします。

- プロジェクト グルーピングの更新プロパティへのタスクの追加
- プロジェクト グルーピングのプロジェクトとベースラインのプロジェクトの比較
- 2 つのプロジェクト グルーピングのプロジェクトの比較
- プロジェクト グルーピングのタスクとベースラインのタスクの比較
- 2 つのプロジェクト グルーピングのタスクの比較
- プロジェクト グルーピング間のタスクのコピー
- プロジェクト グルーピングとメンバーの削除
- プロジェクト グルーピングの一覧表示
- プロジェクト グルーピングのベースラインとタスクの更新のプレビュー
- プロジェクト グルーピングのベースラインとタスクの更新
- プロジェクト グルーピングの更新プロパティからのタスクの削除
- プロジェクト グルーピングの自動更新モードの設定
- プロジェクト グルーピングのプロパティの表示
- プロジェクト　グルーピングの関連プロジェクト、ベースライン、タスクまたはオブジェクトの表示
- プロジェクト グルーピング情報の表示

プロジェクト グルーピングの更新プロパティへのタスクの追加

このサブコマンドにより、指定したプロジェクト グルーピングに指定した 1 つまたは複数のタスクを追加するか、またはすでに削除したすべてのタスクを指定する各プロジェクト グルーピングに戻します。指定したプロジェクト グルーピングを修正可能なユーザーが、このコマンドを使用できます。

```
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))...`

指定したプロジェクト グルーピングに追加するタスクを指定します。追加するタスクが removed tasks リストにある場合、そのリストから削除されます。removed tasks リストにない場合は、プロジェクト グルーピングの手動で追加されたタスクのリストに追加されます。キーワード `all_removed` は、removed tasks リストにあるすべてのタスクを戻すことを意味します。詳細については、タスクの指定を参照してください。

`project_grouping_spec1`

タスクを追加するプロジェクト グルーピングを指定します。詳細については、プロジェクト グルーピングの指定を参照してください。

`project_grouping_spec2`

タスクを追加するプロジェクト グルーピングを指定します。詳細については、プロジェクト グルーピングの指定を参照してください。

例

- プロジェクト グルーピングにタスクを追加する。

  ```
  ccm pg -at G#123 "All A/1.0 Integration Testing Projects from Database G"
  ```

- プロジェクト グルーピングからタスクを削除する。

  ```
  ccm pg -remove_task all "All A/1.0 Integration Testing Projects from Database G"
  ```

プロジェクト グルーピングのプロジェクトとベースラインのプロジェクトの比較

このサブコマンドにより、プロジェクト グルーピングのプロジェクトと指定したベースラインのプロジェクトとを比較します。

`-union`、`-intersection`、または `-not_in` を指定する必要があります。

```
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

比較するベースラインを指定します。詳細については、ベースラインの指定を参照してください。

`-ch|-column_header`

出力フォーマットでカラム ヘッダーを使用するよう指定します。詳細については、-ch|-column_headers を参照してください。

`-f|-format` *format*

コマンドの出力フォーマットを指定します。詳細については、-f|-format を参照してください。

`-gby|-groupby` *groupformat*

コマンドの出力をグループ化する方法を指定します。詳細については、-gby|-groupby を参照してください。

`-int|-intersection`

比較により、プロジェクト グルーピングとベースラインに共通するプロジェクトを表示するよう指定します。

`-nch|-nocolumn_header`

出力フォーマットでカラム ヘッダーを使用しないよう指定します。詳細については、-ch|-column_headers を参照してください。

`-nf|-noformat`

カラム配置を使用しないよう指定します。詳細については、-nf|-noformat を参照してください。

`-ns|-no_sort`

コマンドの出力をソートしないよう指定します。詳細については、-ns|-nosort を参照してください。

`-not|-not_in`

比較により、ベースライン内にないがプロジェクト グルーピング内にあるプロジェクトを表示するよう指定します。

`project_grouping_spec`

比較するプロジェクト グルーピングを指定します。詳細については、プロジェクト グルーピングの指定を参照してください。

`-sep|-separator` *`separator`*

異なる区切り文字を指定します。詳細については、-sep|-separator を参照してください。

`-sby|-sortby` *`sortspec`*

コマンドの出力をソートする方法を指定します。詳細については、-sby|-sortby を参照してください。

`-un|-union`

比較により、プロジェクト グルーピング内のプロジェクトとベースライン内の両方のプロジェクトを表示するよう指定します。

`-u|-unnumbered`

コマンドの出力の自動番号付けを抑止します（すなわち、出力に番号付けがされません）。詳細については、-u|-unnumbered を参照してください。

例

- プロジェクト グルーピング **My 2.0 Collaborative Development projects** とベースライン **1.0 build 123** に共通するプロジェクトを比較する。

  ```
  ccm pg -compare -baseline -project -intersection "My 2.0 Collaborative
  Development projects" "1.0 build 123"
  ```

関連トピック

- 2 つのプロジェクト グルーピングのプロジェクトの比較
- プロジェクト グルーピングのタスクとベースラインのタスクの比較
- 2 つのプロジェクト グルーピングのタスクの比較
- プロジェクト グルーピングの一覧表示
- プロジェクト グルーピングのプロパティの表示
- プロジェクト グルーピング情報の表示

2 つのプロジェクト グルーピングのプロジェクトの比較

このサブコマンドにより、2 つのプロジェクト グルーピング内のプロジェクトを比較します。

`-union`、`-intersection`、または `-not_in` を指定する必要があります。

```
ccm pg|project_grouping -compare -projects
      ([-int|-intersection] | [-not|-not_in] | [-un|-union])
      [-f|-format format] [-nf|-noformat]
      ([-ch|-column_header] | [-nch|-nocolumn_header])
      [-sep|-separator separator] ([-sby|-sortby sortspec] |
      [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat]
      [-u|-unnumbered] project_grouping_spec1 project_grouping_spec2
```

`-ch|-column_header`

出力フォーマットでカラム ヘッダーを使用するよう指定します。詳細については、-ch|-column_headers を参照してください。

`-f|-format format`

コマンドの出力フォーマットを指定します。詳細については、-f|-format を参照してください。

`-gby|-groupby groupformat`

コマンドの出力をグループ化する方法を指定します。詳細については、-gby|-groupby を参照してください。

`-int|-intersection`

比較により、両方のプロジェクト グルーピングに共通するプロジェクトを表示するよう指定します。

`-nch|-nocolumn_header`

出力フォーマットでカラム ヘッダーを使用しないよう指定します。詳細については、-ch|-column_headers を参照してください。

`-nf|-noformat`

カラム配置を使用しないよう指定します。詳細については、-nf|-noformat を参照してください。

`-ns|-no_sort`

コマンドの出力をソートしないよう指定します。詳細については、-ns|-nosort を参照してください。

`-not|-not_in`

比較により、2 番目のプロジェクト グルーピング内にないが 1 番目のプロジェクト グルーピング内にあるプロジェクトを表示するよう指定します。

`project_grouping_spec1`

比較する 1 番目のプロジェクト グルーピングを指定します。詳細については、プロジェクト グルーピングの指定を参照してください。

`project_grouping_spec2`

`project_grouping_spec1` と比較するプロジェクト グルーピングを指定します。詳細については、プロジェクト グルーピングの指定を参照してください。

`-sep|-separator` *`separator`*

異なる区切り文字を指定します。詳細については、-sep|-separator を参照してください。

`-sby|-sortby` *`sortspec`*

コマンドの出力をソートする方法を指定します。詳細については、-sby|-sortby を参照してください。

`-un|-union`

比較によって両方のプロジェクト グルーピングにある両方のプロジェクトを表示するよう指定します。

`-u|-unnumbered`

コマンドの出力の自動番号付けを抑止します（すなわち、出力に番号付けがされません）。詳細については、-u|-unnumbered を参照してください。

例

- プロジェクト グルーピング **My 2.0 Collaborative Development projects** と **My 2.0 Insulated Development projects** にあるプロジェクトの和を表示します。

  ```
  ccm pg -compare -project -union "My 2.0 Collaborative Development
  projects" "My 2.0 Insulated Development projects"
  ```

関連トピック

- プロジェクト グルーピングのプロジェクトとベースラインのプロジェクトの比較
- プロジェクト グルーピングのタスクとベースラインのタスクの比較
- 2 つのプロジェクト グルーピングのタスクの比較
- プロジェクト グルーピングのプロパティの表示
- プロジェクト グルーピング情報の表示

プロジェクト グルーピングのタスクとベースラインのタスクの比較

このサブコマンドにより、プロジェクト グルーピング内にあるすべてのタスクとベースライン内にあるすべてのタスクを比較します。

`-union`、`-intersection`、または `-not_in` を指定する必要があります。

```
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`

比較するベースラインを指定します。詳細については、ベースラインの指定を参照してください。

`-ch|-column_header`

出力フォーマットでカラム ヘッダーを使用するよう指定します。詳細については、-ch|-column_headers を参照してください。

`-f|-format` *`format`*

コマンドの出力フォーマットを指定します。詳細については、-f|-format を参照してください。

`-gby|-groupby` *`groupformat`*

コマンドの出力をグループ化する方法を指定します。詳細については、-gby|-groupby を参照してください。

`-int|-intersection`

比較により、プロジェクト グルーピング内の保存タスクおよび追加タスクとベースライン内のタスクとの間で共通のタスクを表示するよう指定します。

`-nch|-nocolumn_header`

出力フォーマットでカラム ヘッダーを使用しないよう指定します。詳細については、-ch|-column_headers を参照してください。

`-nf|-noformat`

カラム配置を使用しないよう指定します。詳細については、-nf|-noformat を参照してください。

`-ns|-no_sort`

コマンドの出力をソートしないよう指定します。詳細については、-ns|-nosort を参照してください。

`-not|-not_in`

比較により、ベースライン内にないがプロジェクト グルーピング内にある保存タスクおよび追加タスクを表示するよう指定します。

`project_grouping_spec`

比較するプロジェクト グルーピングを指定します。詳細については、プロジェクト グルーピングの指定を参照してください。

`-sep|-separator` *`separator`*

異なる区切り文字を指定します。詳細については、-sep|-separator を参照してください。

`-sby|-sortby` *`sortspec`*

コマンドの出力をソートする方法を指定します。詳細については、-sby|-sortby を参照してください。

`-un|-union`

比較により、プロジェクト グルーピング内にある保存タスクおよび追加タスクとベースライン内にあるタスクの両方を表示するよう指定します。

`-u|-unnumbered`

コマンドの出力の自動番号付けを抑止します（すなわち、出力に番号付けがされません）。詳細については、-u|-unnumbered を参照してください。

例

- プロジェクト グルーピング **My 2.0 Collaborative Development projects** にあるが、ベースライン **1.0 build 123** にはない保存タスクおよび追加タスクを表示する。

  ```
  ccm pg -compare -tasks -not_in -baseline "My 2.0 Collaborative
  Development projects" "1.0 build 123"
  ```

関連トピック

- プロジェクト グルーピングのプロジェクトとベースラインのプロジェクトの比較
- 2 つのプロジェクト グルーピングのプロジェクトの比較
- 2 つのプロジェクト グルーピングのタスクの比較
- プロジェクト グルーピングのプロパティの表示
- プロジェクト グルーピング情報の表示

2 つのプロジェクト グルーピングのタスクの比較

このサブコマンドにより、指定した 2 つのプロジェクト グルーピングのすべてのタスクを比較します。

`-union`、`-intersection`、または `-not_in` を指定する必要があります。

```
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`

2 つのプロジェクト グルーピングの追加されたタスクを比較します。

`-all_tasks`

2 つのプロジェクト グルーピングの追加タスクおよび保存タスクを比較します。

`-ch|-column_header`

出力フォーマットでカラム ヘッダーを使用するよう指定します。詳細については、-ch|-column_headers を参照してください。

`-f|-format format`

コマンドの出力フォーマットを指定します。詳細については、-f|-format を参照してください。

`-gby|-groupby groupformat`

コマンドの出力をグループ化する方法を指定します。詳細については、-gby|-groupby を参照してください。

`-int|-intersection`

比較により、両方のプロジェクト グルーピング間に共通するタスクを表示するよう指定します。

`-nch|-nocolumn_header`

出力フォーマットでカラム ヘッダーを使用しないよう指定します。詳細については、-ch|-column_headers を参照してください。

`-nf|-noformat`

カラム配置を使用しないよう指定します。詳細については、-nf|-noformat を参照してください。

`-ns|-no_sort`

コマンドの出力をソートしないよう指定します。詳細については、-ns|-nosort を参照してください。

`-not|-not_in`

比較により、2 番目のプロジェクト グルーピング内にないが 1 番目のプロジェクト グルーピング内にあるタスクを表示するよう指定します。

`project_grouping_spec1`

比較する 1 番目のプロジェクト グルーピングを指定します。詳細については、プロジェクト グルーピングの指定を参照してください。

`project_grouping_spec2`

`project_grouping_spec1` と比較するプロジェクト グルーピングを指定します。詳細については、プロジェクト グルーピングの指定を参照してください。

`-rt|-removed_tasks`

2 つのプロジェクト グルーピング内の削除されたタスクを比較します。

`-sep|-separator` *`separator`*

異なる区切り文字を指定します。詳細については、-sep|-separator を参照してください。

`-sby|-sortby` *`sortspec`*

コマンドの出力をソートする方法を指定します。詳細については、-sby|-sortby を参照してください。

`-tob|-tasks_on_top_of_baseline`

2 つのプロジェクト グルーピング内のベースライン上のタスクを比較します。

`-un|-union`

比較により、両方のプロジェクト グルーピングにあるタスクを表示するよう指定します。

`-u|-unnumbered`

コマンドの出力の自動番号付けを抑止します（すなわち、出力に番号付けがされません）。詳細については、-u|-unnumbered を参照してください。

例

- プロジェクト グルーピング **All 2.0 Integration Testing projects** にないが、プロジェクト グルーピング **My 2.0 Collaborative Development projects** にあるベースライン上のタスクを表示する。

  ```
  ccm pg -compare -tasks -not_in "My 2.0 Collaborative Development
  projects" "All 2.0 Integration Testing projects"
  ```

関連トピック

- プロジェクト グルーピングのプロジェクトとベースラインのプロジェクトの比較
- 2 つのプロジェクト グルーピングのプロジェクトの比較
- プロジェクト グルーピングのタスクとベースラインのタスクの比較
- プロジェクト グルーピングの一覧表示
- プロジェクト グルーピングのプロパティの表示
- プロジェクト グルーピング情報の表示

プロジェクト グルーピング間のタスクのコピー

このサブコマンドにより、プロジェクト グルーピング間でタスクをコピーします。コピー先のプロジェクト グルーピングを修正可能なユーザーが、このサブコマンドを使用できます。

```
ccm pg|project_grouping -ct|-copy_tasks project_grouping_spec1
        project_grouping_spec2
```

`-ct|-copy_tasks`

ネット タスク（`Saved Tasks` と `Added Tasks`）をプロジェクト グルーピングから他のプロジェクト グルーピングにコピーします。`-add_tasks` オプションを使用した場合と同じ方法で、タスクが 2 番目のプロジェクト グルーピングに追加されます。

ただし、依存関係の分析は行われず、また必要なタスクの計算も行われません。そのため、正確なタスクのセットを別のプロジェクト グルーピングに追加できます。

`project_grouping_spec1`

タスクのコピー元のプロジェクト グルーピングを指定します。詳細については、プロジェクト グルーピングの指定を参照してください。

`project_grouping_spec2`

タスクのコピー先のプロジェクト グルーピングを指定します。詳細については、プロジェクト グルーピングの指定 を参照してください。

例：

- プロジェクト グルーピング間でタスクをコピーする。

  ```
  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
  ```

- **All 2.0 Integration Testing Projects from Database G** から **All 1.0 Integration Testing Projects from Database G** にタスクをコピーする。

  ```
  ccm pg -ct @2 @1
  ```

関連トピック

- プロジェクト グルーピングの一覧表示
- プロジェクト グルーピングのプロパティの表示

- プロジェクト グルーピングの関連プロジェクト、ベースライン、タスクまたはオブジェクトの表示
- プロジェクト グルーピング情報の表示

プロジェクト グルーピングとメンバーの削除

このサブコマンドにより、メンバープロジェクトとともに、またはメンバー プロジェクトなしで、プロジェクト グルーピングを削除します。プロジェクト グルーピングを削除するには、ユーザーが修正可能である必要があります。また、プロジェクト グルーピングにメンバー プロジェクトが入っていてはなりません。

```
ccm pg|project_grouping -d|-delete ([-m|-members] | [-nm|-no_members])
        project_grouping_spec...
```

`-delete`

指定したプロジェクト グルーピングを削除します。複数のオブジェクトに 1 つまたは複数の *project_grouping_spec* 引数を設定できます。このサブコマンドでは、クエリ選択セットは更新されません。

`-m|-members`

プロジェクト グルーピングと共に、プロジェクト グルーピングに関連するプロジェクトも削除するよう指定します。プロジェクトまたはプロジェクト グルーピングで使用されていないすべての関連するフォルダも削除されます。デフォルトは `-nm|-no_members` です。

`-nm|-no_members`

プロジェクト グルーピングに関連するプロジェクトを削除しないよう指定します。この操作は、指定したプロジェクト グルーピングが関連するプロジェクトを持たない場合にのみ成功します。オプションの指定がない場合、これがデフォルトとなります。

project_grouping_spec

削除するプロジェクト グルーピングを指定します。詳細については、プロジェクト グルーピングの指定を参照してください。

関連トピック

- プロジェクト グルーピングの関連プロジェクト、ベースライン、タスクまたはオブジェクトの表示
- プロジェクト グルーピング情報の表示

プロジェクト グルーピングの一覧表示

このサブコマンドにより、指定した条件と一致するプロジェクト グルーピングを一覧表示します。オプションを指定しない場合、すべてのプロジェクト グルーピングが一覧表示されます。

```
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`

出力フォーマットでカラム ヘッダーを使用するよう指定します。詳細については、-ch|-column_headers を参照してください。

`-f|-format format`

コマンドの出力フォーマットを指定します。詳細については、-f|-format を参照してください。

キーワードには、組み込み済みのもの（`%fullname`、`%displayname`、`%objectname`）、あるいは `%modify_time`、`%status` などの既存の属性の名前を使用できます。

キーワードのリストについては、組み込み済みキーワードを参照してください。

`-gby|-groupby groupformat`

コマンドの出力をグループ化する方法を指定します。詳細については、-gby|-groupby を参照してください。

`-nch|-nocolumn_header`

出力フォーマットでカラム ヘッダーを使用しないよう指定します。詳細については、-ch|-column_headers を参照してください。

`-nf|-noformat`

カラム配置を使用しないよう指定します。詳細については、-nf|-noformat を参照してください。

`-ns|-nosort|-no_sort`

コマンドの出力をソートしないよう指定します。詳細については、-ns|-nosort を参照してください。

`-o|-owner` *owner*

指定した所有者のプロジェクト グルーピングのみを一覧表示するよう指定します。owner はユーザー名を表す任意の文字列です。所有者の指定がない場合、すべての所有者のプロジェクト グルーピングが一覧表示されます。

`-purpose` *purpose*

指定した目的のプロジェクト グルーピングを一覧表示するよう指定します。purpose は定義済みの有効な目的の名前です。目的の指定がない場合、すべての目的のプロジェクト グルーピングが一覧表示されます。

`r|-release` *release_spec*

指定したリリースのプロジェクト グルーピングのみを一覧表示するよう指定します。詳細については、リリースの指定を参照してください。リリースの指定がない場合、すべてのリリースのプロジェクト グルーピングが一覧表示されます。

`-sby|-sortby` *sortspec*

コマンドの出力をソートする方法を指定します。詳細については、-sby|-sortby を参照してください。

`-sep|-separator` *separator*

`-f|-format` オプションと一緒にのみ使用します。異なる区切り文字を指定します。詳細については、-sep|-separator を参照してください。

`-u|-unnumbered`

コマンドの出力の自動番号付けを抑止します（すなわち、出力に番号付けがされません）。詳細については、-u|-unnumbered を参照してください。

例

- プロジェクト グルーピングを一覧表示する。

```
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
```

関連トピック

- プロジェクト グルーピングのプロパティの表示
- プロジェクト グルーピング情報の表示

プロジェクト グルーピングのベースラインとタスクの更新のプレビュー

このコマンドにより、指定したプロジェクト グルーピングのベースラインとタスクの更新をプレビューします。メンバーの更新操作を行った場合、この出力を使用して、プロジェクト グルーピングが使用するベースラインとタスクを表示します。

```
ccm pg|project_grouping -pubt|-preview_update_baseline_and_tasks
      [-iat|-include_automatic_tasks] project_grouping_spec...
```

`-iat|-include_automatic_tasks`

プレビューにすべての自動タスクが含まれるようにします。これを指定しない場合、自動タスクは除外されます。

`project_grouping_spec`

プレビューするプロジェクト グルーピングを指定します。詳細については、プロジェクト グルーピングの指定を参照してください。

複数のプロジェクト グルーピングを指定した場合、それぞれが指定した順序で処理されます。

プロジェクト グルーピングは 1 つのベースラインのみを持つことができます。

例

- プロジェクト グルーピング **My 2.0 Collaborative Development projects** の更新イメージを見るため、自動タスクも含めてその中にあるベースラインとタスクをプレビューする。

  ```
  ccm pg -pubt -iat "My 2.0 Collaborative Development projects"
  ```

関連トピック

- プロジェクト グルーピングの一覧表示
- 2 つのプロジェクト グルーピングのタスクの比較
- プロジェクト グルーピングのプロパティの表示
- プロジェクト グルーピング情報の表示
- プロジェクト グルーピングのベースラインとタスクの更新

プロジェクト グルーピングの更新プロパティからのタスクの削除

このサブコマンドにより、指定したプロジェクト グルーピングからタスクを削除します。このコマンドにより、以下の項目を削除できます。

- 指定したタスク
- プロジェクト グルーピングに現在あるすべてのタスク
- プロジェクト グルーピングに手動で追加したすべてのタスク

プロジェクト グルーピングを修正可能なユーザーが、このサブコマンドを使用できます。

```
ccm pg|project_grouping
       -rt|-remove_task|-remove_tasks (task_spec|(all | all_added))
        project_grouping_spec...
```

project_grouping_spec

更新するプロジェクト グルーピングを指定します。詳細については、プロジェクト グルーピングの指定を参照してください。

`(-rt|-remove_task|-remove_tasks (`*task_spec*`|(all|all_added)))...`

指定したプロジェクト グルーピングから削除するタスクを指定します。削除するタスクが added tasks リストにある場合、そのリストから削除されます。タスクが saved tasks リストにある場合、プロジェクト グルーピングの removed tasks リストに追加されます。キーワード all は、すべての追加タスクを削除し、すべての保存タスクを removed tasks リストに追加することを意味します。キーワード `all_added` はすべての追加タスクを削除することを意味します。詳細については、タスクの指定を参照してください。

関連トピック

- プロジェクト グルーピングの更新プロパティへのタスクの追加
- プロジェクト グルーピング間のタスクのコピー
- プロジェクト グルーピングの一覧表示
- プロジェクト グルーピングの関連プロジェクト、ベースライン、タスクまたはオブジェクトの表示
- プロジェクト グルーピング情報の表示

プロジェクト グルーピングの自動更新モードの設定

このサブコマンドにより、指定したプロジェクト グルーピングの自動更新機能を定義します。デフォルトでは、プロジェクトまたはプロジェクト グルーピングのメンバーが更新されると、そのプロジェクト グルーピングのベースラインとタスクが更新されます。. 自動更新機能は解除または設定できます。

```
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`

プロジェクト グルーピングが、更新操作時に必ずベースラインとタスクを更新するよう指定します。

ただし、依存関係の分析は行われず、また必要なタスクの計算も行われません。そのため、正確なタスクのセットを別のプロジェクト グルーピングに追加できます。

`-no_au|-no_auto_update_baselines_and_tasks|-freeze`

プロジェクト グルーピングが、必ず保存されているベースラインとタスクを使用するよう指定します。

`project_grouping_spec`

更新するプロジェクト グルーピングを指定します。詳細については、プロジェクト グルーピングの指定を参照してください。

関連トピック

- プロジェクト グルーピングの一覧表示
- プロジェクト グルーピングのプロパティの表示
- プロジェクト グルーピング情報の表示
- プロジェクト グルーピングのベースラインとタスクの更新

プロジェクト グルーピングのプロパティの表示

このサブコマンドにより、指定したプロジェクトグルーピングの特定のプロパティを表示します。

```
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`

プロジェクト グルーピングが、更新操作時に必ずベースラインとタスクを更新するよう指定します。

`created_in`

プロジェクト グルーピングが作成されているデータベースの名前を表示するよう指定します。

`o|owner`

プロジェクト グルーピングの所有者の名前を表示するよう指定します。

`project_grouping_spec`

表示するプロジェクト グルーピングを指定します。詳細については、プロジェクト グルーピングの指定を参照してください。

`p|purpose`

プロジェクト グルーピングの目的を表示するよう指定します。

`r|release`

プロジェクト グルーピングのリリース値を表示するよう指定します。

`-s|-sh|-show`

引数に指定した順序でプロジェクト グルーピングのプロパティを表示するよう指定します。

`utime|update_time`

ベースラインとタスクを最後に計算（および保存）した時刻を表示するよう指定します。

例

- プロジェクト グルーピングのプロパティを表示する (ベースラインとタスクを自動更新する)。

  ```
  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
  ```

関連トピック

- プロジェクト グルーピングの一覧表示
- プロジェクト グルーピング情報の表示

プロジェクト グルーピングの関連プロジェクト、ベースライン、タスクまたはオブジェクトの表示

このサブコマンドにより、指定したプロジェクト グルーピングの関連するプロジェクト、ベースライン、タスクまたはオブジェクトを表示します。

```
ccm pg|project_grouping -s|-sh|-show ((proj|projects) | (bl|baseline) |
       (at|added_tasks) | (rt|removed_tasks)
       (tob|tasks_on_top_of_baseline) | | all_tasks |
       (obj|objs|objects)) [-f|-format format] [-nf|-noformat]
       ([-ch|-column_header] | [-nch|-nocolumn_header])
       [-sep|-separator separator] ([-sby|-sortby sortspec] |
       [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat] [-u|-unnumbered]
       project_grouping_spec...
```

-ch|-column_header

-f|-format format

-gby|-groupby groupformat

-nch|-nocolumn_header

-nf|-noformat

-ns|-nosort|-no_sort

project_grouping_spec

表示するプロジェクト グルーピングを指定します。詳細については、プロジェクト グルーピングの指定を参照してください。

-sby|-sortby sortspec

-sep|-separator separator

`-s|-sh|-show`

`-show:` と一緒に以下のキーワードを使用できます。

- `proj|projects`

  プロジェクト グルーピングに含まれるすべてのプロジェクトを表示するよう指定します。デフォルト フォーマットは次のとおりです。

```
%displayname %status %owner %release %create_time
```

`-format` オプションを使用すると、デフォルト フォーマットが無効になります。

- `bl|baseline`

  ベースライン名を表示するよう指定します。デフォルト フォーマットは次のとおりです。

  ```
  %displayname:%description
  ```

- `at|added_tasks`

  プロジェクト グルーピングの Added Tasks 内のすべてのタスクを表示するよう指定します。Added Tasks とは、ユーザーが手動で追加したタスクです。

  デフォルト フォーマットは次のとおりです。

  ```
  %displayname %release %owner %create_time
  ```

  `-format` オプションを使用すると、デフォルト フォーマットが無効になります。

- `obj|objs|objects`

  プロジェクト グルーピング内のすべてのプロジェクトに含まれるすべてのオブジェクトを表示するよう指定します。デフォルト フォーマットは次のとおりです。

  ```
  %displayname %status %owner %release %create_time
  ```

  `-format` オプションを使用すると、デフォルト フォーマットが無効になります。

- `rt|removed_tasks`

  プロジェクト グルーピングの Removed Tasks 内のすべてのタスクを表示するよう指定します。これらのタスクはユーザーが手動で削除したタスクです。

  デフォルト フォーマットは次のとおりです。

  ```
  %displayname %release %owner %create_time
  ```

  `-format` オプションを使用すると、デフォルト フォーマットが無効になります。

- `tob|tasks_on_top_of_baseline`

  プロジェクト グルーピング内にあってベースライン内にないタスクを表示するよう指定します。

- `all_tasks`

  更新の際に使用されるタスクを表示するよう指定します。デフォルト フォーマットは次のとおりです。

  ```
  %displayname %release %owner %create_time
  ```

  `-format` オプションを使用すると、デフォルト フォーマットが無効になります。

-u|-unnumbered

例

- プロジェクト グルーピングにあるすべてのタスクを表示する。

  ```
  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
  ```

関連トピック

- プロジェクト グルーピングの一覧表示
- プロジェクト グルーピングのプロパティの表示
- プロジェクト グルーピング情報の表示

プロジェクト グルーピング情報の表示

このサブコマンドにより、指定したプロジェクト グルーピングの情報を表示します。

```
 ccm pg|project_grouping -s|-sh|-show (i|info|information)
        -f|-format format [-nf|-noformat]
        ([-ch|-column_header] | [-nch|-nocolumn_header])
        [-sep|-separator separator] project_grouping_spec...
ccm pg|project_grouping -s|-sh|-show (i|info|information)
        project_grouping_spec...
```

-ch|-column_header

-f|-format format

`-list`

データベース内のプロジェクト グルーピングを一覧表示します。`project_grouping -list` サブコマンドは、番号付き形式オプションをサポートし、クエリ選択セットを設定します。このコマンドでは、ゼロ個、1 つ、または複数のリリース、所有者および目的オプションを使用できます。各リリース オプションでは、1 つのオブジェクトに対して設定可能な *`ReleaseSpec`* オプション値を使用できます。各所有者オプションでは *`owner`* 文字列を使用できます。各目的オプションでは *`purpose name`* 文字列を使用できます。

-nch|-nocolumn_header

-nf|-noformat

`project_grouping_spec`

表示するプロジェクト グルーピングを指定します。詳細については、プロジェクト グルーピングの指定を参照してください。

-sep|-separator separator

例

- プロジェクト グルーピング情報を表示する。

```
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
```

関連トピック

- プロジェクト グルーピングの一覧表示
- プロジェクト グルーピングのプロパティの表示

プロジェクト グルーピングのベースラインとタスクの更新

このコマンドにより、指定したプロジェクト グルーピングのベースラインとタスクを更新します。プロジェクト グルーピングに関連付けられているプロセス ルールが最新のベースラインの検索モードを使用している場合、プロセス ルールについて指定されている条件に一致する最新のベースラインがプロジェクト グルーピングについて評価および選択されます。関連付けられたプロセス ルールによって指定されるタスクがプロジェクト グルーピングに使用されます。

```
ccm pg|project_grouping -ubt|-update_baseline_and_tasks
        project_grouping_spec...
```

project_grouping_spec

更新するプロジェクト グルーピングを指定します。詳細については、プロジェクト グルーピングの指定を参照してください。

例

- プロジェクト グルーピングのベースラインとタスクを更新する。

  ```
  ccm pg -ubt "All Base/1.0 Integration Testing Projects from Database G"
  ```

- **My CM/7.0 Collaborative Development** という名前のプロジェクト グルーピングのベースラインとタスクを更新する。

  ```
  ccm project_grouping -ubt "My CM/7.0 Collaborative Development"
  ```

関連トピック

- プロジェクト グルーピングの一覧表示
- プロセス ルールのベースライン プロジェクト／フォルダ／フォルダ テンプレート／メンバーの表示
- プロジェクト グルーピングのプロパティの表示
- プロジェクト グルーピング情報の表示
- プロジェクト グルーピングのメンバーの更新

説明と用途

プロジェクト グルーピングを使用して、更新操作のためにプロジェクトをリリースおよび目的別に整理します。プロジェクト グルーピングのタスクとベースラインのプロパティは、グループ内のすべてのプロジェクトで一貫したメンバー選択を行うためにプロジェクトの更新時に使用されます。プロジェクトがメンバーとなれるのは、1 つのプロジェクト グルーピングのみです。プロジェクト グルーピングは、プロジェクトの作成時に自動的に作成されます。

プロジェクト グルーピングには、個人用と非個人用があります。個人用オブジェクト内のすべてのプロジェクトは、プロジェクト グルーピングと所有者、リリース、目的および状態が同じになります。個人用プロジェクト グルーピングは、以下のいずれかによって識別されます。

- `My` *`release purpose`* `Projects`

  プロジェクト グルーピングの所有者は現在のユーザーと同じで、データベースは DCM 用に初期化されていないか、またはプロジェクト グルーピングが `My CM/6.5 Insulated Development Projects` などのローカル データベースに作成されています。

- *`owner's release purpose`* `Projects`

  プロジェクト グルーピングの所有者は別のユーザーで、データベースは DCM 用に初期化されていないか、またはプロジェクト グルーピングが `John's CM/6.5 Insulated Development Projects` などのローカル データベースに作成されています。

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

  プロジェクト グルーピングの所有者は現在のユーザーと同じで、データベースは DCM 用に初期化されており、プロジェクト グルーピングは `My CM/6.5 Insulated Development Projects` などのローカル データベースに作成されていません。

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

  プロジェクト グルーピングの所有者は別のユーザーで、データベースは DCM 用に初期化されており、プロジェクト グルーピングは `John's CM/6.5 Insulated Development Projects from Database D` などのローカル データベースに作成されていません。

非個人用オブジェクト内のすべてのプロジェクトは、プロジェクト グルーピングとリリース、目的および状態が同じになります。非個人用プロジェクト グルーピングは、以下のいずれかによって識別されます。

- `All CM/6.5 Integration Testing Projects from Database D` などの、DCM 用に初期化されたデータベースの `All` *`release purpose`* `Projects from Database` *`dbid`*。*`dbid`* はプロジェクト グルーピングが作成されたデータベースのデータベース ID。
- `All CM/6.5 System Testing Projects` などの DCM 用に初期化されていないデータベースの `All` *`release purpose`* `Projects`。

各ローカル プロジェクト グルーピングは、そのリリースと目的に該当するプロセス ルールと関連付けられています。プロジェクト グルーピングは関連するプロセス ルールを 1 つだけ持つことができます。

ただし、プロジェクト グルーピング内のすべてのプロジェクトが、そのプロジェクト グルーピングによって指定された更新プロパティを持っているとは限りません。プロセス ルールを使用するプロジェクトは、同じ更新プロパティを持っています。プロジェクト グルーピングには、プロセス ルールを使用しないプロジェクトが含まれることもあります。またタスクの代わりに使用中のオブジェクトを更新するプロジェクトもあります。このようなプロジェクトを同じグルーピングに含めることにより、フルセットのプロジェクトからのベースライン作成が可能となります。

適切な更新プロパティを持つために、プロジェクト グルーピングはデータベース内の他のオブジェクトと多数の関係を保持しています。プロセス ルールはフォルダとタスクを使用するので、これらの同じフォルダとタスクがプロセス ルールを使用するプロジェクト グルーピングに関連付けられます。また、1 つのプロジェクト グルーピングには一連の保存されたタスク、追加されたタスク、削除されたタスク、自動タスクが含まれており、それぞれがプロジェクト グルーピングに固有のものです。グルーピング内のタスクは追加、削除が可能です。すべてのローカル プロジェクト グルーピングも、プロセス ルールがベースラインを使用する場合は、ベースラインとの関係を持っています。

ビルド マネージャによるプロジェクト グルーピングの最適な使用方法の詳細については、『Rational Synergy ビルド マネージャ ガイド』を参照してください。

project_purpose コマンド

詳細については、説明と用途を参照してください。project_purpose コマンドは以下のサブコマンドをサポートします。

- プロジェクトの目的の作成
- プロジェクトの目的の削除
- プロジェクトの目的の修正
- プロジェクトの目的の表示

プロジェクトの目的の作成

このサブコマンドにより、プロジェクトの目的を作成します。このサブコマンドを使用するには、*build_mgr* または *ccm_admin* ロールを持っている必要があります。

```
ccm project_purpose -cr|-create -n|-name purpose_name
                    -stat|-status status [-ms|-member_status member_status]
```

`-ms|-member_status` *`member_status`*

プロジェクト目的のメンバー状態を指定します。メンバー状態により、更新時に異なる目的で使用される同じ状態の複数プロジェクトを識別できます。値は、データベース内で一意である必要があります。

目的とメンバー状態は類似の表現にすべきです。たとえば、目的 **Test Integration** を作成する場合は、メンバー状態の値を **test_int** のように目的と類似した名前に設定します。

目的の作成時に、メンバー状態を指定しない場合は、一意の値が自動的に作成されて使用されます。

`-n|-name` *`purpose_name`*

新しいプロジェクトの目的の名前を指定します。名前は、データベース内で一意である必要があります。

DCM 用に初期化されたデータベースを使用する場合、DCM クラスタ内のデータベースに同じ目的を作成する必要があります。

`-stat|-status` *`status`*

新しい目的の状態を指定します。これは、*working*、*visible*、*shared* または *prep* などの修正可能な状態です。

例

- 名前が `Test Purpose`、状態が *prep*、メンバー状態が `test` であるプロジェクト目的を作成し、新規に作成された目的を表示する。

  ```
  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
  ```

プロジェクトの目的の削除

このサブコマンドにより、*purpose_name* によって指定したプロジェクトの目的を削除します。プロジェクトの目的を削除すると、Synergy は以下のように動作します。

- 削除した目的でプロジェクトのコピーおよび作成ができない。
- 既存のプロジェクトと製品はメンバー状態の設定を保持する。
- 既存のプロジェクトと製品は、その目的を削除した目的に変更できない。
- その目的のプロセス ルールは削除される。

このサブコマンドを使用するには、*build_mgr* または *ccm_admin* ロールを持っている必要があります。

```
ccm project_purpose -d|-delete purpose_name...
```

`-d|-delete` *purpose_name...*

削除するプロジェクトの目的の名前を指定します。この名前は有効な目的の名前である必要があります。

例

- `Test2 Purpose` という名前のプロジェクト目的を削除する。

  ```
  ccm project_purpose -d "Test2 Purpose"
  ```

プロジェクトの目的の修正

このサブコマンドにより、プロジェクトの目的の名前またはメンバー状態を修正します。このサブコマンドを使用するには、*build_mgr* または *ccm_admin* ロールを持っている必要があります。

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

`-m|-modify [-n|-name` *`purpose_name`*`]`

プロジェクトの目的を修正します。`-name` *`purpose_spec`* オプションと一緒に使用した場合は、修正するするプロジェクト目的名を指定します。`-member_status` オプションと一緒に使用した場合は、修正するメンバー状態の値を指定します。

`-ms|-member_status`

プロジェクトの目的のメンバー状態を指定します。メンバー状態により、更新時に異なる目的で使用される同じ状態の複数プロジェクトを識別できます。値は、データベース内で一意である必要があります。

このオプションを `-modify` オプションと一緒に使用することで、変更する目的のメンバー状態を指定します。プロジェクト目的テーブルからの値は各目的オプションで使用されます。

目的とメンバー状態は類似の表現にすべきです。たとえば、目的 **Test Integration** を作成する場合は、メンバー状態の値を **test_int** のように目的と類似した名前に設定します。

例

- プロジェクト目的の名前とメンバー状態を変更する。

  ```
  ccm project_purpose -m -n "Test2 Purpose" -ms test2 "Test Purpose"
  ```

プロジェクトの目的の表示

```
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`

出力フォーマットでカラム ヘッダーを使用するよう指定します。詳細については、-ch|-column_headers を参照してください。

`-f|-format` *`format`*

コマンドの出力フォーマットを指定します。詳細については、-f|-format を参照してください。

`-gby|-groupby` *`groupformat`*

コマンドの出力をグループ化する方法を指定します。詳細については、-gby|-groupby を参照してください。

`-nch|-nocolumn_header`

出力フォーマットでカラム ヘッダーを使用しないよう指定します。詳細については、-ch|-column_headers を参照してください。

`-nf|-noformat`

カラム配置を使用しないよう指定します。詳細については、-nf|-noformat を参照してください。

`-no_personal|-nopersonal`

shared または *prep* などの個人用でない状態に関連付けられたプロジェクトの目的を表示します。

`-ns|-no_sort`

コマンドの出力をソートしないよう指定します。詳細については、-ns|-nosort を参照してください。

`-personal`

working または *visible* などの個人用の状態に関連付けられたプロジェクトの目的を表示します。

`purpose_name`

表示するプロジェクトの目的の名前を指定します。これは現在のデータベースで定義されている目的である必要があります。

`-rel|-release` *`release_spec`*

指定したリリースに有効なプロジェクトの目的を表示します。*`release_spec`* には、1 つのリリース値を指定できます。詳細については、リリースの指定を参照してください。

`-sep|-separator` *`separator`*

異なる区切り文字を指定します。詳細については、-sep|-separator を参照してください。

`-sby|-sortby` *`sortspec`*

コマンドの出力をソートする方法を指定します。詳細については、-sby|-sortby を参照してください。

`-stat|-status` *`status`*

指定した状態のプロジェクトの目的のみを表示するよう指定します。

例

- *developer* ロールのユーザーのプロジェクト目的を表示する。

```
ccm project_purpose -show -role -personal
 Purpose Name               Member Status     Status
 Collaborative Development  collaborative     working
 Insulated Development      working           working
 Visible Development        visible           visible
```

説明と用途

`project_purpose` コマンドにより、Rational Synergy データベースのプロジェクト目的を、ユーザー ロールに応じて作成または表示します。どのユーザーもプロジェクト目的を表示できます。プロジェクト目的マネージャは、プロジェクト目的を作成できます。プロジェクト目的は、同じプロジェクトの複数の *prep*（準備）バージョン、*shared*（共有）バージョン、*working*（作業）バージョン、*visible*（可視）バージョンを、複数のテスト レベルなど、用途別に設定するために使用します。

プロジェクト目的は以下の内容を含みます。

- 目的名

  この名前は、たとえばパフォーマンス テスト、個人用の使用などの目的を反映します。

- 目的のメンバー状態

  メンバー状態により、更新操作実行時に、異なる目的で使用される同じ状態の複数プロジェクトを識別できます。たとえば、`sqa1`、`sqa2` および `sqa3` というそれぞれ一意の 3 レベルのシステム テストを定義できます。

- プロジェクトの状態

  状態は、この目的のプロジェクト（*working*、*prep* など）が使用可能な状態を示します。

プロジェクト目的テーブルは以下に影響を与えます。

- `ccm copy_project` と `ccm create -type project` のコマンドで指定可能なオプション
- 各目的オプションを使用してコピーされたプロジェクトで使用される `status` および `member_status` 値の指定
- 関連する自動タスク プロジェクトの決定
- 該当する自動タスクの概要

各 Rational Synergy データベースには、1 つのプロジェクト目的リストが入っています。各リリースのプロジェクト目的リストを定義できます。

プロジェクト目的テーブルは、以下の目的を定義します。

```
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
```

properties コマンド

詳細については、説明と用途を参照してください。`properties` コマンドは、以下のサブコマンドをサポートします。

- プロパティの表示
- 指定したフォーマットでのプロパティの表示

プロパティの表示

このサブコマンドにより、オブジェクトの情報を表示します。表示される属性はオブジェクトのタイプによって異なります。最も関連性の高い情報が表示されます。

このコマンドでは、クエリ選択セットは設定されません。クエリにより、指定した順序でオブジェクトが表示されます。

```
ccm info|prop|properties -p|-project [-v|-verbose] project_spec...
ccm info|prop|properties [-v|-verbose] object_spec...
```

object_spec

プロパティを表示させるオブジェクトを指定します。

`-p|-project`

プロジェクトの履歴を表示します。

project_spec

表示するプロジェクトを指定します。詳細については、プロジェクトの指定を参照してください。

`-v|-verbose`

より詳細な情報を表示します。これは、フォルダなどの特定のタイプのオブジェクトに適用されます。詳細情報形式がないオブジェクト タイプについては無視されます。

例

- オブジェクト状態を使用して更新を行う `os_ico-1` プロジェクトの情報を表示する。

  ```
  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
  ```

- 現在のディレクトリ内の全オブジェクトのリリース値を表示する。

  ```
  ccm prop -f "%objectname %release" *
  ```

指定したフォーマットでのプロパティの表示

このサブコマンドにより、指定したフォーマットでオブジェクトの情報を表示します。このコマンドでは、クエリ選択セットは設定されません。クエリにより、指定した順序でオブジェクトが表示されます。

```
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`

出力フォーマットでカラム ヘッダーを使用するよう指定します。詳細については、-ch|-column_headers を参照してください。

`-f|-format format`

コマンドの出力フォーマットを指定します。詳細については、-f|-format を参照してください。

キーワードには、組み込み済みのもの（`%fullname`、`%displayname`、`%objectname`）、あるいは `%modify_time`、`%status` などの既存の属性の名前を使用できます。

キーワードのリストについては、組み込み済みキーワードを参照してください。

`-nch|-nocolumn_header`

出力フォーマットでカラム ヘッダーを使用しないよう指定します。詳細については、-ch|-column_headers を参照してください。

`-nf|-noformat`

カラム配置を使用しないよう指定します。詳細については、-nf|-noformat を参照してください。

`-ns|-nosort|-no_sort`

コマンドの出力をソートしないよう指定します。詳細については、-ns|-nosort を参照してください。

`-p|-project`

プロジェクトの履歴を表示します。

project_spec

一覧表示するプロジェクトを指定します。詳細については、プロジェクトの指定を参照してください。

`-sep|-separator` *separator*

-fl-format オプションと一緒にのみ使用します。異なる区切り文字を指定します。詳細については、-sep|-separator を参照してください。

説明と用途

`properties` コマンドにより、1 つまたは複数のオブジェクトに関する情報を表示します。このコマンドは、指定されたオブジェクトのモデル定義属性のグループの属性値を、標準出力に表示します。

query コマンド

詳細については、説明と用途を参照してください。query コマンドは、オブジェクトのクエリとクエリ選択セットの表示サブコマンドをサポートします。

オブジェクトのクエリとクエリ選択セットの表示

```
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`

出力フォーマットでカラム ヘッダーを使用するよう指定します。詳細については、-ch|-column_headers を参照してください。

`-f|-format format`

コマンドの出力フォーマットを指定します。詳細については、-f|-format を参照してください。

`-gby|-groupby groupformat`

コマンドの出力をグループ化する方法を指定します。詳細については、-gby|-groupby を参照してください。

`-i|-instance instance...`

`instance='instance'` 形式のクエリ文節を含めて、指定したインスタンスを持つオブジェクトを検索します。

`-n|-name name...`

`name='name'` 形式のクエリ文節を含めて、指定した名前を持つオブジェクトを検索します。

`-nch|-nocolumn_header`

出力フォーマットでカラム ヘッダーを使用しないよう指定します。詳細については、-ch|-column_headers を参照してください。

`-nf|-noformat`

カラム配置を使用しないよう指定します。詳細については、-nf|-noformat を参照してください。

`-ns|-no_sort`

コマンドの出力をソートしないよう指定します。詳細については、-ns|-nosort を参照してください。

`-o|-owner` *`owner`*`...`

`owner='`*`owner`*`'` 形式のクエリ節を含めて、指定した所有者を持つオブジェクトを検索します。

`query_string`

クエリ節に結合されるクエリ文字列を指定します。クエリ節はクエリ関連オプションから生成され、評価されるクエリ式を形成します。Rational Synergy CLI ヘルプ、トラディショナル モードの「クエリ式」を参照してください。

`-release` *`release_spec`*`...`

`release='`*`release`*`'` 形式のクエリ節を含めて、指定したリリース値を持つオブジェクトを検索します。*`release_spec`* には、複数のリリース値を指定できます。詳細については、リリースの指定を参照してください。

`-sep|-separator` *`separator`*

異なる区切り文字を指定します。詳細については、-sep|-separator を参照してください。

`-sby|-sortby` *`sortspec`*

コマンドの出力をソートする方法を指定します。詳細については、-sby|-sortby を参照してください。

`-s|-state` *`state`*`...`

`state='`*`state`*`'` 形式のクエリ節を含めて、指定した状態を持つオブジェクトを検索します。

`-task` *`task_spec`*`...`

`is_associated_cv_of(task('task_spec'))` 形式のクエリ節を含めて、指定したタスクの関連オブジェクトを検索します。*`task_spec`* には、複数のタスクを指定できます。詳細については、タスクの指定 を参照してください。

`-t|-type` *`type`*`...`

`cvtype='`*`type`*`'` 形式のクエリ節を含めて、指定したタイプを持つオブジェクトを検索します。

`-v|-version` *`version`*`...`

`version='`*`version`*`'` 形式のクエリ節を含めて、指定したバージョンのオブジェクトを検索します。

`-u|-unnumbered`

コマンドの出力の自動番号付けを抑止します（すなわち、出力に番号付けがされません）。詳細については、-u|-unnumbered を参照してください。

例

- 名前が `foo.c`、所有者が *valerie* であるすべてのオブジェクトを一覧表示する。

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

- 選択セット内の項目 3 の内容を表示するため、以下を入力する。

```
ccm cat @3
```

- タスク 4 の名前が `foo.c`、所有者が *ann* のすべてのオブジェクトを一覧表示する。

```
ccm query -n foo.c -o ann -task 4
1) foo.c-1.2  working   ann csrc 1 4
```

- 名前が `brochure.doc`、所有者が *ann* であるすべてのオブジェクトの最後に修正された名前と時刻を一覧表示する。

```
ccm query -n brochure.doc -o ann -f "%name %modify_time"
1) brochure.doc Tue Aug  6 12:17:55 1996
```

- `santa_fe` データベースからのタスク 3 に関連付けられているすべてのオブジェクトを一覧表示する。

```
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>
```

- 特定の転送セットに関連付けられている変更依頼を一覧表示する。

```
ccm query query_expression
```

ここで、*`query_expression`* は、転送セットで使用されている変更依頼クエリで、"`cvtype=problem`" を含みます。

例：

```
ccm query "cvtype='problem' and product_name='myproduct'"
```

- **Collaborative Development** 汎用プロセス ルールをインスタンス化したリリース固有プロセス ルールを表示する。

  ```
  ccm query ''cvtype='process_rule' and name='Collaborative
  Development'' -f "%none %is_generic_pr_of"
  ```

関連トピック

- finduse コマンド

説明と用途

データベース内のオブジェクトを検索するには、query コマンドを使用します。Rational Synergy では、クエリ式の評価は検索動作中に行われます。クエリ式は、任意のクエリ節から構成されます。クエリ節は、任意の *query_string* 引数を結合したクエリ関連オプションで構成されます。クエリの結果は、選択セットとして表示されます。

デフォルトでは、クエリは、ソートとグループ化で説明しているソート基準を使用してオブジェクトをソートします。

クエリ関連オプションや *query_string* 引数から構成されるクエリ式を指定しないと、このコマンドは、現在の選択セットについてソートを適用して表示します。

クエリ機能とソート

ソート機能のあるクエリ関数（たとえば、recursive_is_member_of）を使用する場合、そのクエリ関数のソート順は、-no_sort が指定されている場合やクエリ関数に複合クエリのために他のクエリ演算子が結合されていない場合には、最終的な表示出力に適用されます。

選択セットの順序付けと使用方法

デフォルトでは、出力には選択セット参照番号が与えられます。したがって、選択セット参照構文（たとえば、@1）を使用して、選択セット内の特定のオブジェクトを参照できます。詳細については、クエリ選択セット参照形式を参照してください。

クエリ式の組み立て

query コマンドは、クエリ式を組み立てるためのさまざまなオプションをサポートします。たとえば、-name オプションは、name='name' 型のクエリ節を組み立てるための別法です。

こういったオプションを繰り返して使用すると、対応するクエリ節が or で結合されたクエリ節が作成されます。たとえば、-n joe -n ann という記述はクエリ節 (name='joe' or name='ann') になります

異なるオプションを指定した場合は、クエリ節は and で結合されます。たとえば、-n joe -s working という記述は、クエリ節 (name='joe') and (status='working') になります。

これらの組み立てられたクエリ節は、指定された任意の *query_string* 引数と and で結合されます。たとえば、-n joe "is_hist_leaf()" という記述はクエリ式 (name='joe') and (is_hist_leaf()) になります。

reconcile コマンド

詳細については、説明と用途を参照してください。`reconcile` コマンドは、以下のサブコマンドをサポートします。

- ワークエリア コンフリクトの表示
- データベースの変更を反映したワークエリアの同期
- ワークエリアの変更を反映したデータベースの同期

ワークエリア コンフリクトの表示

このサブコマンドにより、ワークエリア コンフリクトを特定して表示します。ただし、コンフリクトの解決は行いません。

```
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`

出力フォーマットでカラム ヘッダーを使用するよう指定します。詳細については、-ch|-column_headers を参照してください。

`-cu|-consider_uncontrolled`

リコンサイル中に管理対象外のファイルを考慮するよう指定します。ソース管理下にないファイルは、ワークエリア コンフリクトとして報告されます。`-cu|-consider_uncontrolled` と `-if|-ignore_files|-ignore_uncontrolled` のどちらも指定していない場合は、デフォルトの設定により、管理対象外のファイルは無視されます。

`-f|-format` *format*

コマンドの出力フォーマットを指定します。詳細については、-f|-format を参照してください。

キーワードには、組み込み済みのもの（`%fullname`、`%displayname`、`%objectname`）、あるいは `%modify_time`、`%status` などの既存の属性の名前を使用できます。

キーワードのリストについては、組み込み済みキーワードを参照してください。

file_spec

リコンサイルの対象とするファイルまたはディレクトリを指定します。

`-gby|-groupby` *groupformat*

コマンドの出力をグループ化する方法を指定します。詳細については、-gby|-groupby を参照してください。

`-if|-ignore_files|-ignore_types` *file_type,...*

ファイル名に指定された拡張子が含まれるファイルを、リコンサイルの対象外とするよう指定します。このオプションは管理対象外のファイルにのみ適用されるもので、`-cu|-consider_uncontrolled` オプションと一緒に使用する必要があります。このオプション値は、カンマで区切られた 1 つまたは複数の拡張子のリストです。

`-imwf|-ignore_missing_wa_file`

ワークエリアに不足しているファイルを無視して、ワークエリア コンフリクトとして報告しないよう指定します。`-mwaf|-missing_wa_file` を指定しない場合、これがデフォルトとなります。

`-iu|-ignore_uncontrolled`

管理対象外のファイルをリコンサイルしないよう指定します。`-cu|-consider_uncontrolled` と `-if|-ignore_files|-ignore_uncontrolled` のどちらも指定されていない場合は、デフォルトの設定により、管理対象外のファイルは無視されます。

`-mwaf|-missing_wa_file`

ワークエリアに不足しているファイルをワークエリア コンフリクトとして報告するよう指定します。デフォルトの設定では、ワークエリアに不足しているファイルは無視されます。

`-nch|-nocolumn_header`

出力フォーマットでカラム ヘッダーを使用しないよう指定します。詳細については、-nch|-nocolumn_headers を参照してください。

`-nf|-noformat`

カラム配置を使用しないよう指定します。詳細については、-nf|-noformat を参照してください。

`-nr|-no_recurse`

プロジェクトまたはディレクトリのリコンサイル時に、プロジェクトのサブプロジェクト、またはディレクトリのファイルおよびサブディレクトリまで再帰的にリコンサイル処理を行わないよう指定します。`-nr|-norecurse|-no_recurse` を指定しない場合、これがデフォルトとなります。

`-ns|-nosort|-no_sort`

コマンドの出力をソートしないよう指定します。詳細については、-sby|-sortby を参照してください。

`project_spec`

リコンサイルするプロジェクトを指定します。

`-r|-recurse`

リコンサイル対象に指定したプロジェクトのサブプロジェクト、またプロジェクトのディレクトリ下のファイルおよびサブディレクトリまで再帰的にリコンサイルするよう指定します。デフォルトでは、再帰的なリコンサイルは行いません。

プロジェクトの同期を取るときに、このオプションによりリコンサイル操作の深さを制御します。この制御は重要です。多数のネストしたサブプロジェクトを持つ最上位プロジェクトの同期を取る場合、このオプションでリコンサイル処理を行うと、相当の時間と資源がかかる可能性があるからです。指定した最上位プロジェクト下にあるすべてのサブプロジェクトについてリコンサイルが行われるため、このオプションの指定には検討が必要です。階層全体にわたって同期を取らなければ、時間と資源を節約できます。一方、階層全体をリコンサイルする必要がある場合には、このオプションを使用して 1 回の操作で行うことができます。

ディレクトリを指定して `-recurse` を指定した場合、`reconcile` はそのディレクトリにあるサブプロジェクトは再帰処理しません。

`-sby|-sortby` *`sortspec`*

コマンドの出力をソートする方法を指定します。詳細については、-sby|-sortby を参照してください。

`-sep|-separator` *`separator`*

`-f|-format` オプションと一緒にのみ使用します。異なる区切り文字を指定します。詳細については、-sep|-separator を参照してください。

-s|-show

コンフリクトを解決せずに、そのまま表示するよう指定します。これはデフォルト設定です。

-udb|-update_db

ワークエリア内のバージョンでデータベースを更新します。このオプションには、下記の使い方があります。

- チェックアウトしていないファイルを修正した場合、デフォルトの設定ではリコンサイルによって新しいバージョンが作成され、行った変更に合わせてデータベースが更新される。
- 別のワークエリアからのファイルを変更してデータベースを更新し、このワークエリアからの同じファイルも変更した場合、リコンサイルによってこのワークエリアからデータベースが更新される。

したがって、このオプションを使用するのは、現在のワークエリアに一連の変更が正しく反映されていることが確実な場合だけです。

-uwa|-update_wa

データベースからのバージョンでワークエリアを更新します。このオプションを使用するのは、データベースに一連の変更が正しく反映されていることが確実な場合だけです。

例

- ico_june16-1 プロジェクトをリコンサイルするが、ファイル名に拡張子 .doc、.gif、または .exe が含まれているファイルは対象外とする。

  ```
  ccm reconcile -p ico_june16-1 -ignore_types "*.doc;*.gif;*.exe"
  ```

- UNIX： ico_june16-1 プロジェクトをリコンサイルするが、ワークエリア内で行われた更新は破棄し、そのプロジェクトに属するサブプロジェクトはリコンサイルしない。

  この例では、*working* 状態にある move.c オブジェクトと *integrate* 状態にある colname.c オブジェクトを更新するタスクを担当していると想定しています。ワークエリア内でそれらのオブジェクトをコピーして修正した後で、プロジェクトの方向が変更されて、結局それらの変更は必要なくなりました。

  ```
  % cd ~john/ccm_wa/ccmint15
  % ls
  ico_june16-1
  $ ccm reconcile -p ico_june16-1 -no_recurse
  Examining work area for conflicts...
  not recursing hierarchy, conflicts will be automatically discarded
  Updating '/users/john/ccm_wa/ccmint15/ico_june16-1'...
  Discarding changes to '/users/john/ccm_wa/ccmint15/ico_june16-1/
  ```

```
ico_june16/src/colname.c'..
Discarding changes to '/users/john/ccm_wa/ccmint15/ico_june16-1/
ico_june16/src/move.c'...
Reconciliation complete.
```

ワークエリアはデータベースからの元のファイルで更新されますが、`colname.c` と `move.c` に加えられた変更は破棄されます。

関連トピック

- work_area コマンド

データベースの変更を反映したワークエリアの同期

このサブコマンドにより、データベースの変更によってワークエリアを更新します。*working* または *visible* 状態のプロジェクトの場合、プロジェクトの所有者のみがこの操作を実行できます。ビルド管理プロジェクトの場合、この操作を実行するにはビルド マネージャである必要があります。また、ワークエリアはユーザーから見え、修正可能である必要があります。

```
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`

非管理ファイルをワークエリアから削除するよう指定します。`reconcile_save_uncontrolled` オプションを設定すると、ファイルはゴミ箱に移動します。`-cu|-consider_uncontrolled` と `-if|-ignore_files|-ignore_uncontrolled` のどちらも指定していない場合は、デフォルトの設定により、管理対象外のファイルは無視されます。

`file_spec`

リコンサイルの対象とするファイルまたはディレクトリを指定します。

`-if|-ignore_files|-ignore_types` *`file_type,...`*

-if|-ignore_files|-ignore_types file_type,... を参照してください。

`-imwaf|-ignore_missing_wa_file`

ワークエリアに不足しているファイルを無視して、データベースから再作成されないよう指定します。`-mwaf|-missing_wa_file` を指定しない場合、これがデフォルトとなります。

`-iu|-ignore_uncontrolled`

管理対象外のファイルをリコンサイルしないよう指定します。`-cu|-consider_uncontrolled` と `-if|-ignore_files|-ignore_uncontrolled` のどちらも指定していない場合は、デフォルトの設定により、管理対象外のファイルは無視されます。

`-mwaf|-missing_wa_file`

ワークエリアに不足しているファイルについて、対応するメンバーをデータベース内のプロジェクトから使用解除することで処理するよう指定します。オブジェクトはデータベースから削除されません。デフォルトの設定では、ワークエリアに不足しているファイルは無視されます。

`-nr|-no_recurse`

-nr|-no_recurse を参照してください。

`project_spec`

リコンサイルするプロジェクトを指定します。

`-r|-recurse`

-r|-recurse を参照してください。

`-uwa|-update_wa`

-uwa|-update_wa を参照してください。

例

- `proj1` 内のディレクトリ `src` をリコンサイルし、データベースからワークエリアを更新し、不足しているファイルをチェックする。

  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
  ```

- プロジェクト `proj1` およびそのサブプロジェクトをリコンサイルし、ワークエリアからデータベースを更新し、管理対象外のファイルをチェックする。

  ```
  ccm reconcile -recurse -consider_uncontrolled -update_db -project proj1-1
  ```

関連トピック

- work_area コマンド

ワークエリアの変更を反映したデータベースの同期

このサブコマンドにより、ワークエリアで行った変更によってデータベースを更新します。*working* または *visible* 状態のプロジェクトの場合、プロジェクトの所有者のみがこの操作を実行できます。ビルド管理プロジェクトの場合、この操作を実行するにはビルドマネージャである必要があります。また、ワークエリアはユーザーから見え、修正可能である必要があります。

```
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`

管理対象外のファイルをソース管理下に置き、データベース内のオブジェクトとして作成し、ファイル内容をワークエリアからコピーするよう指定します。`-cu|-consider_uncontrolled` と `-if|-ignore_files|-ignore_uncontrolled` のどちらも指定していない場合は、デフォルトの設定により、管理対象外のファイルは無視されます。

`file_spec`

リコンサイルの対象とするファイルまたはディレクトリを指定します。

`-if|-ignore_files|-ignore_types` *`file_type,...`*

-if|-ignore_files|-ignore_types file_type,... を参照してください。

`-imwaf|-ignore_missing_wa_file`

ワークエリアに不足しているファイルを無視し、プロジェクトの対応メンバーを削除しないよう指定します。`-mwaf|-missing_wa_file` を指定しない場合、これがデフォルトとなります。

`-iu|-ignore_uncontrolled`

管理対象外のファイルをリコンサイルしないよう指定します。`-cu|-consider_uncontrolled` と `-if|-ignore_files|-ignore_uncontrolled` のどちらも指定していない場合は、デフォルトの設定により、管理対象外のファイルは無視されます。

`-mwaf|-missing_wa_file`

ワークエリアに不足しているファイルをデータベース内の対応オブジェクトから再作成するよう指定します。デフォルトの設定では、ワークエリアに不足しているファイルは無視されます。

`-nr|-no_recurse`

-nr|-no_recurse を参照してください。

`-p|-project` *`project_spec`*

リコンサイルするプロジェクトを指定します。

`-r|-recurse`

-r|-recurse を参照してください。

`-t|-task` *`-task_spec`*

リコンサイルによって作成またはチェックアウトされた新しいファイルまたはディレクトリを関連付けるタスクを指定します。タスクを指定しない場合、デフォルトでカレント タスクが使用されます。*`task_spec`* には、1 つのタスクを指定できます。

`-udb|-update_db`

-udb|-update_db を参照してください。

例

- ワークエリアからデータベースを更新することにより、`foo.c` ファイルをリコンサイルする。

  ```
  ccm reconcile -update_db foo.c-1:csrc:1
  ```

関連トピック

- work_area コマンド

説明と用途

`reconcile` コマンドにより、ワークエリア内のファイルをデータベース ファイルと比較します。ワークエリアの内容とデータベースの間の不一致を、ワークエリア コンフリクトといいます。`reconcile` コマンドはワークエリア コンフリクトを特定し、ワークエリアとデータベースが一致するようにコンフリクトを解決します。

ワークエリア コンフリクトは下記の場合に発生します。

- チェックアウトしたか否かにかかわらず、ワークエリア内でファイルを修正した。
- 別のワークエリアからファイルを変更してデータベースを更新し、同じファイルをこのワークエリア内でも変更した。
- データベース内のファイルを変更したが、更新すべきワークエリアが更新可能ではなかった。
- ワークエリア内でファイルを作成したが、そのファイルをソース管理の下に置かなかった。
- 別のワークエリアからファイルをチェックインしたが、そのワークエリアは変更を反映するために更新可能ではなかった。
- ワークエリアからファイルを削除したが、プロジェクトからそのファイルを削除しなかった。

管理対象リンク、シンボル リンク、およびワークエリア パスに関して、その他のエラーが発生する可能性があります。この種のコンフリクトは手作業で解決する必要があります。

以下に、チェックアウトしたファイルにこのコマンドを適用するその他の方法を示します。

- ワークエリアがノート パソコン上にあり、Rational Synergy から切り離して作業できる場合には、`reconcile` コマンドを使用して、ワークエリアとデータベースの同期を取ることができる。
- UNIX 上で、修正中のオブジェクトと Rational Synergy データベース間のリンクが使用中のツールによって壊された場合には、`reconcile` コマンドにより変更をリコンサイルしてからリンクを再確立できる。

たとえば、Rational Synergy のセッションが確立されていないときに、チェックアウトしていないオブジェクトを修正する必要がある場合、ワークエリア内でその変更を行ってから、後で Rational Synergy データベースを更新できます。そのためには、該当ファイルの読み取り専用属性を解除して、ファイルを修正します。後で Rational Synergy セッションを開始したときに、`reconcile` コマンドを使用して、ワークエリア内で行った変更に合わせてデータベースを更新できます。

> **注記：**CLI からリコンサイルを中止するには、任意の時点で Ctrl + C キー を押します。

CLI からリコンサイルを中止するときに、ワークエリア内でエラーが発生する可能性があることを示すメッセージが表示されます。ただし、エラーはワークエリアを使用し始めてから発生します。したがって、問題発生を回避するために、使用前に完全にワークエリアをリコンサイルしてください。

操作によっては自動的にリコンサイルが行われる場合があります。

- ファイルをチェックインしておりコンテキスト プロジェクトが使用可能な場合は、対応するワークエリアのコンフリクトの検査が行われ、可能な場合にはワークエリアの変更を反映してデータベースが自動的に更新されます。
- 修正可能なファイルをワークエリア内で変更し、`ccm use` コマンドまたは `ccm update_members` コマンドの結果として別バージョンが使用された場合、ワークエリアで変更されたファイルの内容を反映してデータベースが更新されます。
- ワークエリアで静的ファイルを変更してからチェックアウトすると、チェックアウトしたファイルは更新されたワークエリアの内容を反映して更新されます。
- ワークエリア ファイルの新しい内容を反映してデータベースが更新された場合、ユーザーが修正可能なファイルを使用しており、ユーザーから見える修正可能なワークエリアを持つプロジェクトは、新しい内容を反映して更新されます。この状況は、データベースが `ccm reconcile` コマンドによって明示的に更新された場合、または他の操作の一部として自動的に更新された場合に発生します。

relate コマンド

詳細については、説明と用途を参照してください。`relate` コマンドは、以下のサブコマンドをサポートします。

- オブジェクト間の関係の作成
- オブジェクト間の関係の表示

オブジェクト間の関係の作成

```
ccm relate -n|-name relationship_name -f|-from from_object_spec
           -t|-to to_object_spec
```

`-f|-from from_object_spec`

新しい関係を作成する元のオブジェクトを指定します。*from_object_spec* には、1 つのオブジェクトを指定できます。

`-n|-name relationship_name`

作成する新しい関係の名前を指定します。

`t|-to to_object_spec`

新しい関係を作成する先のオブジェクトを指定します。*to_object_spec* には、1 つのオブジェクトを指定できます。

例

- `clear-2` を `clear-1` の後継にする。

  ```
  ccm relate -n successor -f clear-1 -t clear-2
  ```

- `print.c` のバージョン 5.1.1 をバージョン 6 にリンクする。

  ```
  ccm relate -name successor -from print.c-5.1.1:csrc:1 -to print.c-
  6:csrc:1
  ```

関連トピック

- オブジェクト間の関係の表示

オブジェクト間の関係の表示

```
ccm relate -s|-sh|-show [-l] [-fmt|-format format] [-nf|-noformat]
           [-sep|-separator separator] ( [-sby|-sortby sortspec] |
           [-ns|-nosort|-no_sort] ) [-gby|-groupby groupformat]
           [-n|-name relationship_name] [-f|-from from_object_spec]
           [-t|-to to_object_spec]
```

`-ch|-column_header`

出力フォーマットでカラム ヘッダーを使用するよう指定します。詳細については、-ch|-column_headers を参照してください。

`-fmt|-format format`

コマンドの出力フォーマットを指定します。詳細については、-f|-format を参照してください。

キーワードには、組み込み済みのもの（`%fullname`、`%displayname`、`%objectname`）、あるいは `%modify_time`、`%status` などの既存の属性の名前を使用できます。

キーワードのリストについては、組み込み済みキーワードを参照してください。

`-f|-from from_object_spec`

関係を一覧表示する元のオブジェクトを指定します。*`from_object_spec`* には、1 つのオブジェクトを指定できます。

`-gby|-groupby groupformat`

コマンドの出力をグループ化する方法を指定します。詳細については、-gby|-groupby を参照してください。

`-l`

デフォルトの長いフォーマットを使用するよう指定します。

`-n|-name`

表示する関係の名前を指定します。

`-nch|-nocolumn_header`

出力フォーマットでカラム ヘッダーを使用しないよう指定します。詳細については、-ch|-column_headers を参照してください。

`-nf|-noformat`

カラム配置を使用しないよう指定します。詳細については、-nf|-noformat を参照してください。

`-ns|-nosort|-no_sort`

コマンドの出力をソートしないよう指定します。詳細については、-ns|-nosort を参照してください。

`-sby|-sortby` *`sortspec`*

コマンドの出力をソートする方法を指定します。詳細については、-sby|-sortby を参照してください。

`-sep|-separator` *`separator`*

`-f|-format` オプションと一緒にのみ使用します。異なる区切り文字を指定します。詳細については、-sep|-separator を参照してください。

`-s|-show`

指定されたオブジェクト間の関係を表示します。

`-t|-to` *`to_object_spec`*

関係を一覧表示する先のオブジェクトを指定します。*`to_object_spec`* は *`object_spec`* である必要があります。これには 1 つのオブジェクトを指定できます。

関連トピック

- オブジェクト間の関係の作成

説明と用途

`relate` コマンドにより、*`file_spec1`* と *`file_spec2`* との間に関係（*`relation_name`*）を追加したり、指定したデータとの関係を表示します。

Rational Synergy には、さらに多くの関係があらかじめ定義されています。定義済みの関係については、Rational Synergy CLI ヘルプ、トラディショナル モードの「関係」の表を参照してください。ただし、`relate` コマンドを使用して、新しい関係を定義できます。

release コマンド

詳細については、説明と用途を参照してください。`release` コマンドは、以下のサブコマンドをサポートします。

- リリースの作成
- リリースの削除
- リリースの一覧表示
- リリースの修正
- リリースの管理データベースの設定
- リリースのプロセス ルールの表示
- リリース情報の表示

リリースの作成

このサブコマンドにより、新しいリリース定義を作成します。このサブコマンドを使用するには、*build_mgr* または *ccm_admin* ロールを持っている必要があります。

新しいアプリケーションまたはコンポーネント用のリリースを作成するには、一意のコンポーネント名を持つリリースを作成するか、コンポーネント名を使用せずに作成することができます。

旧リリースをベースとしてリリースを作成するには、`-from` オプションを使用します。デフォルトでは、新しいリリースは、旧リリースに対応するプロセス ルールとプロパティを使用して作成されます。また、新しいリリースは、ベースとするリリースの後継でもあります。旧リリースはベースライン リリースとして使用されます。

```
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`

リリースがアクティブであることを指定します。これはデフォルト設定です。

`-allow_dcm_transfer`

転送セットのリリースの範囲とクエリによってリリースが組み込まれた場合に、リリースが DCM 複製の対象となるよう指定します。新しいコンポーネントのリリースを作成する場合、このオプションのデフォルトは `true` となります。旧リリースをベースとしてリリースを作成する場合、ベースとするリリースの設定がデフォルトとなります。

`-allow_parallel_check_in`

このリリースでオブジェクトのパラレル チェックインを許可するよう指定します。これは、新しいリリースを作成するときのデフォルトです。旧リリースをベースとしてリリースを作成する場合、ベースとするリリースの設定がデフォルトとなります。パラレル チェックインを許可し、パラレル チェックアウトを禁止する組み合わせは無効です。

`-allow_parallel_check_out`

このリリースでオブジェクトのパラレル チェックアウトを許可するよう指定します。これは、新しいリリースを作成するときのデフォルトです。旧リリースをベースとしてリリースを作成する場合、ベースとするリリースの設定がデフォルトとなります。パラレル チェックインを許可し、パラレル チェックアウトを禁止する組み合わせは無効です。

`-baseline` *`release_spec`*

新しいリリースのベースラインとして使用するリリースを指定します。旧リリースをベースとしてリリースを作成する場合、デフォルトで旧リリースがベースラインとして使用されます。新しいコンポーネントのリリースを作成する場合、デフォルトのベースライン リリースは空白です。

`-cct|-create_component_tasks`

作成するリリース定義のベースラインを作成するときに、対応するコンポーネント タスクを作成するよう指定します。このオプションは、リリース定義の作成時に、チームがコンポーネント タスクを必要としているのが分かっている場合などに使用します。ベースラインの作成時に、コンポーネント タスクが自動的に作成されます。これはデフォルト設定です。

`-desc|-description` *`description`*

リリースの説明を指定します。復帰改行文字などの文字は、エスケープ シーケンスを使用して含めることができます。あるいは、複数行の説明を指定するには、`-description_file` または `-description_edit` を使用します。`-description`、`-description_file` および `-description_edit` をすべて一緒に使用すると、`-description` オプション値を使用して説明が作成され、`-description_file` によって指定されたファイルから読み込まれた説明が追加され、さらに現在のデフォルトのテキスト エディタが起動してコメントが表示されます。その後、エディタで保存したテキストが説明の設定に使用されます。

`-desc_edit|-description_edit`

現在のテキスト エディタを起動して、対話形式によってリリースの説明を編集または作成できるようにします。テキスト エディタで保存した結果が説明の設定に使用されます。-desc|-description description を参照してください。

`-desc_file|-description_file` *`file_path`*

説明を含むファイルへのパスを指定します。

`-from` *`release_spec`*

新しいリリースのベースとするリリースを指定します。旧リリースをベースとしてリリースを作成する場合、新しいリリースの設定の多くは旧リリースからコピーされます。デフォルトで、旧リリースがベースライン リリースとして使用されます。

`-groups` *`groups`*

新しいリリースを修正するか、以降のリリースを作成するグループを指定します。旧リリースをベースとしてリリースを作成する場合、デフォルトでは、新規リリースはベースとするリリースと同じグループを使用します。groups 値は、空白またはカンマで区切った 1 つまたは複数のグループ名のリストです。

`-inactive`

リリースが非アクティブであることを指定します。開発者は非アクティブ リリースを使用して開発作業を行うことはできません。デフォルトでは、新しいリリースはアクティブ リリースとして作成されます。

`-included_releases` *`included_releases`*

リリースに含めるべきリリースを 1 つ以上指定します。この文字列には、複数のリリース値をカンマあるいは空白で区切って指定できます。カンマは必須です。ただし、前または後ろに空白が付いたリリース値は使用できません。あるいは、`included_releases_file` オプションを使用して、ファイルからデータを入力することもできます。

デフォルトでは、含まれるリリース値はオブジェクト ステータスベースの更新にのみ使用されます。メンバーの更新操作時に、選択スコアの重み付けに使用されます。

`-included_releases_file` *`file_path`*

含めるべきリリースが格納されているファイルへのパスを指定します。

`-manager` *`manager`*

リリースに関するプロダクト マネージャまたはコンポーネント マネージャを指定します。作成時のデフォルトは、リリース定義を作成するユーザーです。指定できる文字列は 1 行だけです。

`-noallow_dcm_transfer`

リリースが DCM 複製の対象とならないよう指定します。新しいコンポーネントのリリースを作成する場合、デフォルトでリリースの複製は可能です。旧リリースをベースとしてリリースを作成する場合、ベースとするリリースの設定がデフォルトとなります。

`-noallow_parallel_check_in`

このリリースでのオブジェクトのパラレル チェックインを禁止するよう指定します。新しいリリースを作成する場合、デフォルトでパラレル チェックインは許可されます。旧リリースをベースとしてリリースを作成する場合、ベースとするリリースの設定がデフォルトとなります。パラレル チェックインを許可し、パラレル チェックアウトを禁止する組み合わせは無効です。

`-noallow_parallel_check_out`

このリリースでのオブジェクトのパラレル チェックアウトを禁止するよう指定します。新しいリリースを作成する場合、デフォルトでパラレル チェックアウトは許可されます。旧リリースをベースとしてリリースを作成する場合、ベースとするリリースの設定がデフォルトとなります。パラレル チェックインを許可し、パラレル チェックアウトを禁止する組み合わせは無効です。

`-nocct|-nocreate_component_tasks`

作成中のリリース定義用のベースラインを公開するときに、コンポーネント タスクが自動的に作成されるよう指定します。リリースのベースラインを公開した後、ベースラインのコンポーネント タスクの作成 サブコマンドを使用してコンポーネント タスクを手動で作成できます。

`-phase` *`phasename`*

新しいリリースのリリース フェーズを指定します。デフォルトでは、作成する新しいリリースのリリース フェーズ `New` です。有効なリリース フェーズはモデル属性 release_phase_list で定義されます。出荷時のデフォルト値は、`New`、`Requirements Definition`、`Function Definition`、`Implementation`, `Validation`、および `Released` です。指定した値は有効なリリース フェーズ値のいずれかに合致する必要があります。また、大文字と小文字が区別されます。

`-process` *`process_spec`*

作成中のリリースのプロセスを指定します。指定したプロセスの汎用プロセス ルールに関連付けられたリリース固有のプロセス ルールが、新しいリリースに関連付けられます。リリース固有のプロセス ルールが存在しない場合には、作成されます。

`release_spec`

作成する新しいリリースの名前を指定します。

例

- alphabets/1.0 のプロパティを使用して、新しいリリース alphabets/2.0 を作成する。

  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
  ```

- harmony/1.0 という名前の、（既存のリリースをベースとしない）新しいコンポーネント用のリリースを作成する。

  ```
  ccm release -create "harmony/1.0" -desc "new product line to integrate
  X and Y" -manager "sue" -active -noallow_dcm_transfer
  ```

リリースの削除

このサブコマンドにより、1 つまたは複数のリリース定義を削除します。プロジェクト、ファイル、フォルダまたはベースラインに使用されているリリースを削除するには、-force オプションを使用します。-force オプションを省略した場合、このコマンドは、リリースが他のリリースまたはプロセス ルールから参照されていない場合にのみ成功します。削除されるリリースに後継リリースがある場合、削除されるリリースの履歴のみが取り除かれます。このサブコマンドを使用するには、*build_mgr* または *ccm_admin* ロールを持っている必要があります。

```
ccm release -d|-delete [-force] release_spec...
```

`-force`

リリースまたはプロセス ルール以外のオブジェクトから参照されている場合でも、リリースを削除するよう指定します。-force を指定しなかった場合、プロジェクトまたはファイルによって使用されているリリースは削除されません。

`release_spec`

削除するリリースを指定します。詳細については、リリースの指定を参照してください。

例

- 指定したリリースを使用するオブジェクトがある場合でも、`sue/6.5` のリリース定義を削除する。

  ```
  ccm release -delete -force sue/6.5
  ```

リリースの一覧表示

このサブコマンドにより、指定した条件と一致するリリースを一覧表示します。条件を指定しない場合、すべてのリリースが一覧表示されます。

```
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`

アクティブ リリースのみを一覧表示するよう指定します。`-active` と `-inactive` のどちらも指定しない場合、アクティブ リリースと非アクティブ リリースの両方が表示されます。

`-ch|-column_header`

出力フォーマットでカラム ヘッダーを使用するよう指定します。詳細については、-ch|-column_headers を参照してください。

`-component component_name`

特定のコンポーネント名のリリースのみを一覧表示するよう指定します。コンポーネント名を指定しないと、リリースは一覧表示されません。

`-f|-format format`

コマンドの出力フォーマットを指定します。詳細については、-f|-format を参照してください。

キーワードには、組み込み済みのもの（`%fullname`、`%displayname`、`%objectname`）、あるいは `%modify_time`、`%status` などの既存の属性の名前を使用できます。

キーワードのリストについては、組み込み済みキーワードを参照してください。

`-gby|-groupby groupformat`

コマンドの出力をグループ化する方法を指定します。詳細については、-gby|-groupby を参照してください。

`-inactive`

非アクティブ リリースのみを一覧表示するよう指定します。`-active` と `-inactive` のどちらも指定しない場合、アクティブ リリースと非アクティブ リリースの両方が表示されます。

`-nch|-nocolumn_header`

出力フォーマットでカラム ヘッダーを使用しないよう指定します。詳細については、-nch|-nocolumn_headers を参照してください。

`-nf|-noformat`

カラム配置を使用しないよう指定します。詳細については、-nf|-noformat を参照してください。

`-ns|-nosort|-no_sort`

コマンドの出力をソートしないよう指定します。詳細については、-ns|-nosort を参照してください。

`-sby|-sortby` *`sortspec`*

コマンドの出力をソートする方法を指定します。詳細については、-sby|-sortby を参照してください。

`-sep|-separator` *`separator`*

`-f|-format` オプションと一緒にのみ使用します。異なる区切り文字を指定します。詳細については、-sep|-separator を参照してください。

`-u|-unnumbered`

コマンドの出力の自動番号付けを抑止します（すなわち、出力に番号付けがされません）。詳細については、-u|-unnumbered を参照してください。

例

- リリースを一覧表示する。

```
ccm release -list -active -component a
1) a/1.0
ccm release -list -inactive -component b
1) b/1.0
```

リリースの修正

このサブコマンドにより、1 つまたは複数のリリースを修正します。このサブコマンドを使用するには、*build_mgr* または *ccm_admin* ロールを持っている必要があります。

```
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`

リリースをアクティブに設定します。

`-allow_dcm_transfer`

転送セットのリリースの範囲とクエリによってリリースが組み込まれた場合に、リリースが DCM 複製の対象となるよう指定します。

`-allow_parallel_check_in`

このリリースでのオブジェクトのパラレル チェックインを許可するよう指定します。パラレル チェックインを許可し、パラレル チェックアウトを禁止する組み合わせは無効です。

`-allow_parallel_check_out`

このリリースでオブジェクトのパラレル チェックアウトを許可するよう指定します。パラレル チェックインを許可し、パラレル チェックアウトを禁止する組み合わせは無効です。

`-apr|-add_process_rule|-add_process_rules process_rule_spec`

指定したプロセスを、引数によって指定した各リリースに追加します。詳細については、プロセス ルールの指定を参照してください。

`-baseline` *`release_spec`*

修正するリリースのベースライン リリースを設定します。詳細については、リリースの指定を参照してください。

`-cct|-create_component_tasks`

修正するリリース定義のベースラインを作成するときに、対応するコンポーネント タスクを作成するよう指定します。このオプションは、リリース定義の修正時に、チームがコンポーネント タスクを必要としているのが分かっている場合などに使用します。ベースラインの修正時に、コンポーネント タスクが自動的に作成されます。

`-cpr|-clear_process_rules`

プロセス ルールを追加する前に既存のプロセス ルールをクリアし（-apr|-add_process_rule|-add_process_rules process_rule_spec を参照）、完全なプロセス ルール セットを設定します。

`-desc|-description` *`description`*

-desc|-description description を参照してください。

`-desc_edit|-description_edit`

-desc_edit|-description_edit を参照してください。

`-desc_file|-description_file` *`file_path`*

-desc_file|-description_file file_path を参照してください。

`-groups` *`groups`*

新しいリリースを修正するか、以降のリリースを作成するグループを指定します。*groups* 値は、空白またはカンマで区切った 1 つまたは複数のグループ名のリストです。

`-inactive`

修正するリリースが非アクティブに設定されるよう指定します。開発者は非アクティブ リリースを使用して開発作業を行うことはできません。

`-included_releases` *`included_releases`*

-included_releases included_releases を参照してください。

`-included_release_file` *`file_path`*

-included_releases_file file_path を参照してください。

`-manager` *`manager`*

-manager manager を参照してください。

`-m|-modify`

指定したリリースを修正します。

`-noallow_dcm_transfer`

リリースが DCM 複製の対象外となるよう指定します。

`-noallow_parallel_check_in`

このリリースでのオブジェクトのパラレル チェックインを禁止するよう指定します。パラレル チェックインを許可し、パラレル チェックアウトを禁止する組み合わせは無効です。

`-noallow_parallel_check_out`

このリリースでのオブジェクトのパラレル チェックアウトを禁止するよう指定します。パラレル チェックインを許可し、パラレル チェックアウトを禁止する組み合わせは無効です。

`-nocct|-nocreate_component_tasks`

修正中のリリース定義用のベースラインを公開するときに、コンポーネント タスクが自動的に作成されるよう指定します。リリースのベースラインを公開した後、ベースラインのコンポーネント タスクの作成 サブコマンドを使用してコンポーネント タスクを手動で作成できます。

`-phase` *`phasename`*

指定したリリースのリリース フェーズを設定するよう指定します。有効なリリース フェーズはモデル属性 release_phase_list で定義されます。デフォルト値は、**New**、**Requirements Definition**、**Function Definition**、**Implementation**、**Validation**、および **Released** です。

`release_spec`

修正するリリースを指定します。詳細については、リリースの指定を参照してください。

`-rpr|-remove_process_rule|-remove_process_rules` *process_rule_spec*

指定したプロセスを、引数によって指定した各リリースから削除します。詳細については、プロセス ルールの指定を参照してください。

例

- リリース情報を修正して、新しい説明、新しいマネージャ、およびリリースが**導入**フェーズにあることを示す。

  ```
  ccm release -modify -description "version a of release 1.0 without
  graphics capability" -manager jane -phase Implementation client/1.0a
  ```

関連トピック

- リリースのプロセス ルールの表示
- リリース情報の表示

リリースの管理データベースの設定

このサブコマンドにより、1 つまたは複数のリリース用の管理データベースを設定します。このサブコマンドを使用するには、*build_mgr*、*dcm_mgr*、または *ccm_admin* のロールを持っている必要があります。

```
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*

指定したコンポーネント名を持つリリースの管理データベースを設定するよう指定します。文字列が空白の場合、ヌル コンポーネント名を持つリリースの変更に適用されます。

database_spec

DCM の更新が特定のデータベースで受け入れられるよう指定します。詳細については、データベースの指定を参照してください。『Rational Synergy Distributed』の「DCM データベースの設定」を参照してください。

`-handover` *database_spec*

指定したデータベースにリリースの管理をハンドオーバするよう指定します。*database_spec* には、*1* つのデータベース定義を指定します。詳細については、データベースの指定を参照してください。リリースがローカルに管理されている場合のみ、このオプションを使用できます。『Rational Synergy Distributed』の「DCM データベースの設定」を参照してください。

`-local`

ローカル データベースによってデータベースの制御を行うよう指定します。オブジェクトは、他のデータベースからの DCM 複製では更新されなくなります。『Rational Synergy Distributed』の「DCM データベースの設定」を参照してください。

例

- ローカルに管理されているリリース定義の管理を ID が A1 であるデータベースに引き渡す。

  ```
  ccm release -controlling_database -local -handover A1 -component
  releasename
  ```

リリースのプロセス ルールの表示

このサブコマンドにより、1 つまたは複数のリリースのプロセス ルールを表示します。

```
ccm release -s|-sh|-show ((pr|prs|process_rules) |
       (apr|aprs|available_process_rules) | (upr|uprs|unused_process_rules))
        [-f|-format format] [-nf|-noformat]
        ([-ch|-column_header] | [-nch|-nocolumn_header])
        [-sep|-separator separator] ([-sby|-sortby sortspec] |
        [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat] [-u|-unnumbered]
        release_spec...
```

[-ch|-column_header]

[-f|-format format]

[-gby|-groupby groupformat]

[-nch|-nocolumn_header]

[-nf|-noformat]

[-ns|-nosort|-no_sort]

[-sby|-sortby sortspec]

[-sep|-separator separator]

`-show process_rules|pr|prs`

指定した各引数について、現在の有効なプロセス ルールを表示するよう指定します。引数ごとにデフォルトで番号が振られ、選択セットが設定されます。

`-show available_process_rules|aprs|apr`

各リリースの使用可能なプロセス ルールを表示するよう指定します。使用可能なプロセス ルールとは、すでに使用中のプロセス ルールも含め、リリースで使用可能なすべてのプロセス ルールのことです。引数ごとにデフォルトで番号が振られ、選択セットが設定されます。

`-show upr|uprs|unused_process_rules`

各リリースで使用可能で未使用のプロセス ルールを表示します。使用可能で未使用のプロセス ルールとは、有効なプロセス ルールとして現在使用されているプロセス ルールを除外した、使用可能なプロセス ルールのことです。引数ごとにデフォルトで番号が振られ、選択セットが設定されます。

`-u|-unnumbered`

関連トピック

- リリースの一覧表示
- リリース情報の表示

リリース情報の表示

```
ccm release -s|-sh|-show (i|info|information) -f|-format format
        [-nf|-noformat] ([-ch|-column_header] | [-nch|-nocolumn_header])
        [-sep|-separator separator] release_spec...
ccm release -s|-sh|-show (i|info|information) [-v|-verbose]
        release_spec...
```

-ch|-column_header

-f|-format format

-nch|-nocolumn_header

-nf|-noformat

-sep|-separator separator

-v|-verbose
: 詳細フォーマットで情報を表示するよう指定します。

例

- リリース client/3.5 に関する情報を表示する。

  ```
  ccm release -show information client/3.5
  ```

- リリース a/1.0 に関する情報を表示する。

  ```
  ccm release -show information a/1.0
  ```

説明と用途

`release` コマンドを使用して、リリース情報を作成、修正、削除、表示します。

リリースは、プロジェクトがどのように更新されるかを管理し、またパラレル 開発をサポートします。各リリースは一意の名前を持ち、変更はその名前に関連付けられます。リリース名は、コンポーネント名、リリース区切り文字、コンポーネント リリースで構成されます。たとえば、リリース名 **webapp/3.0** は、**webapp** という名前のコンポーネントのリリース **3.0** を意味します。デフォルトのリリース区切り文字は「/」、コンポーネント名の最大長は 64 文字、コンポーネント リリースの最大長は 32 文字です。リリース名の別の形式として、**2.0** のようにコンポーネントリリース値のみを使用する方法があります。このようなリリースは、「ヌル」コンポーネント名を使用したリリースと呼ばれます。リリース名にコンポーネント名を含める形式の利点は、同じコンポーネント名をもつリリースに論理的な関係があることが明らかになること、および区別のために他に名前をつける手間が省けることです。

リリース定義には多数の重要なプロパティが含まれます。ベースライン リリースはプロジェクトの更新時に使用されます。プロセス ルールのベースライン リリース／目的リスト内で `%baseline_release` キーワードが使用されている場合、この値は更新されるプロジェクトまたはプロジェクト グルーピングのリリースのベースライン リリースになります。特定のリリースを開発者によって使用されないようにするには、そのリリースに非アクティブとマーク付けすることができます。

コンポーネント名およびコンポーネント リリースの先頭には、以下の文字を使用できません。

/¥ ' " : * ? [ ] @ % - + ~ 空白、タブ

2 番目以降の文字列には、以下の文字を使用できません。

/ ¥ " : * ? [ ] @ %

コンポーネント名とコンポーネント リリースには、バージョン区切り文字（デフォルトは「-」）を使用できますが、制限文字は使用できません。

オブジェクトがチェックアウトされると、Rational Synergy はカレント タスクから新規オブジェクトにリリースをコピーします。

リリース操作を行うためには、必要なロールで作業をしている必要があります。

- どのユーザーもリリース情報を表示または一覧表示できる。
- ビルド マネージャまたは *ccm_admin* ロールのユーザーは、リリース定義を作成、修正、削除できる。
- *ccm_admin* ロールのユーザーはリリース区切り文字を変更できる。
- ビルド マネージャまたは *ccm_admin* ロールのユーザーは、リリース定義とそれに関連するプロセス ルールだけが更新される場合には、リリース名を変更できる。ただし、他の関連オブジェクトが更新される場合には、*ccm_admin* ロールが必要である。

save_offline_and_delete コマンド

詳細については、説明と用途を参照してください。`save_offline_and_delete` コマンドは以下のサブコマンドをサポートします。

- 削除用のプレビュー オブジェクト リストの作成
- プレビュー オブジェクト リストを使用した削除
- スコープを使用した削除

前提条件

オブジェクトをオフラインで保存するには、現在のデータベースを DCM 用に初期化する必要があります。また、DCM のライセンスが必要です。

削除用のプレビュー オブジェクト リストの作成

このサブコマンドにより、soad スコープによって削除されるオブジェクトのプレビューを行い、プレビューによって発見されたオブジェクトを削除するプレビュー オブジェクト リストを作成します。クエリ選択セットはプレビュー オブジェクト リストの結果に一致します。このコマンドを実行すると、プレビュー オブジェクト リストがクリアされます。プレビュー オブジェクト リストまたは scope コマンドを使用してオブジェクトを削除した場合にも、プレビュー オブジェクト リストがクリアされます。

このサブコマンドによって、`ccm query` のようなコマンドで引き続いて使用できるオブジェクトリストに、プレビュー結果のエントリーを作成します。コマンドの結果は、現在のセッション内でのみ使用できます。

このコマンドを実行するには、指定したスコープに従ったロールが必要です。ロールは、スコープごとに実行用とプレビュー用が決められています。

> **注記**：プレビューの結果は、プレビュー後、`ccm soad -delete` コマンドの実行前にクエリを実行しても上書きされません。

```
ccm soad|save_offline_and_delete -preview -scope scope_name
    [-so|-save_offline] [-f|-format format] [-nf|-noformat]
    ([-ch|-column_header] | [-nch|-nocolumn_header])
    [-sep|-separator separator] ([-sby|-sortby sortspec] |
    [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat]
    [-u|-unnumbered] [-v|-verbose]
    [argument...]
```

`-ch|-column_header`

出力フォーマットでカラム ヘッダーを使用するよう指定します。詳細については、-ch|-column_headers を参照してください。

`-f|-format format`

コマンドの出力フォーマットを指定します。詳細については、-f|-format を参照してください。

`-gby|-groupby groupformat`

コマンドの出力をグループ化する方法を指定します。詳細については、-gby|-groupby を参照してください。

`-nch|-nocolumn_header`

出力フォーマットでカラム ヘッダーを使用しないよう指定します。詳細については、-nch|-nocolumn_headers を参照してください。

`-nf|-noformat`

カラム配置を使用しないよう指定します。詳細については、-nf|-noformat を参照してください。

`-ns|-no_sort`

コマンドの出力をソートしないよう指定します。詳細については、-ns|-nosort を参照してください。

`-sby|-sortby` *`sortspec`*

コマンドの出力をソートする方法を指定します。詳細については、-sby|-sortby を参照してください。

`-sep|-separator` *`separator`*

-f|-format オプションと一緒にのみ使用します。異なる区切り文字を指定します。詳細については、-sep|-separator を参照してください。

`-scope` *`scope_name`* [*`argument...`*]

オフライン保存またはオブジェクトの削除に使用されるスコープ（修正されたクエリ）を指定します。

指定するスコープに適切である場合のみ、引数を必要とします。たとえば、スコープ「`My working projects and products for a specified release`」には、引数であるリリース値を指定する必要があります。

引数は、スコープ定義で使用されているとおりの順序で指定する必要があります。

スコープの詳細な説明については、Synergy CLI ヘルプ、トラディショナル モードの「SOAD スコープ」を参照してください。

`-so|-save_offline`

データベースに残る最後のバージョンとなる任意のオブジェクトのオブジェクト名とインスタンスを保持するオブジェクトリストを作成します。

`-u|-unnumbered`

コマンド出力の自動番号付けを抑止します。詳細については、-u|-unnumbered を参照してください。

`-v|-verbose`

オブジェクトがリストに含められた理由またはリストから除外された理由を、詳しく説明するメッセージを生成するよう指定します。

例

- スコープ **Baselines marked for deletion and related projects and products** によって見つかるオブジェクトの削除をプレビューする。

  ```
  ccm soad -preview -scope "Baselines marked for deletion and related
  projects and products"
  ```

関連トピック

- プレビュー オブジェクト リストを使用した削除
- スコープを使用した削除

プレビュー オブジェクト リストを使用した削除

このサブコマンドにより、削除のためのプレビュー オブジェクト リストを作成するコマンドによって作成された、プレビュー オブジェクト リスト内のオブジェクトを削除します。プレビュー オブジェクト リストは完了時にクリアされます。

先に `ccm soad -preview` コマンドで `-so` オプションを使用していた場合、オブジェクトは削除の前に保存されます。

削除を行っても、各オブジェクトの最後のバージョンとなるオブジェクトはデータベースに保持されます。

```
ccm soad|save_offline_and_delete -delete [-path soad_path]
            [-pn|-package_name package_name] [-v|-verbose]
```

`-path soad_path`

前に指定されたパスを使用して、DCM パッケージへのパスを指定します。

前にパッケージを保存していない場合には、このパスを指定する必要があります。

> **注記**：パスはエンジンから可視であり、*ccm_root* によって書き込み可能である必要があります。

`-pn|-package_name package_name`

オブジェクトを保存する先の、DCM パッケージの名前を指定します。デフォルトの名前は「`Save Offline and Delete saved on %date`」です。

> **注記**：自分用のパッケージを作成する場合、パッケージ名に `%date` キーワードを含めてください。このキーワードを含めておくと、同じスコープを使用して作成されたパッケージどうしを区別できます。

`-v|-verbose`

オブジェクトがリストに含められた理由またはリストから除外された理由を、詳しく説明するメッセージを生成するよう指定します。

例

- 前の soad プレビューで見つかったオブジェクトを削除する。

  ```
  ccm soad -delete
  ```

関連トピック

- 削除用のプレビュー オブジェクト リストの作成
- スコープを使用した削除

スコープを使用した削除

このサブコマンドにより、プレビューを行わずに、スコープによって指定されるオブジェクトを削除します。以前のプレビュー オブジェクト リストは完了時にクリアされます。

> **注意！**オブジェクトを削除する前にプレビューすることで、削除するオブジェクトを削除の前に確認できます。*ccm_admin* ロールで作業している場合、これは重要です。このコマンドではプレビューは実行されません。コマンドが実行されるまで、削除されるオブジェクトを確認することができません。

```
ccm soad|save_offline_and_delete -delete -scope scope_name
        [-so|-save_offline] [-path soad_path]
        [-pn|-package_name package_name] [-v|-verbose]
        [argument...]
```

`-path soad_path`

前に指定されたパスを使用して、DCM パッケージへのパスを指定します。

前にパッケージを保存していない場合には、このパスを指定する必要があります。

> **注記**：パスはエンジンから可視であり、*ccm_root* によって書き込み可能である必要があります。

`-pn|-package_name package_name`

オブジェクトを保存する先の、DCM パッケージの名前を指定します。デフォルトの名前は「`Save Offline and Delete saved on %date`」です。

> **注記**：自分用のパッケージを作成する場合、パッケージ名に `%date` キーワードを含めてください。このキーワードを含めておくと、同じスコープを使用して作成されたパッケージどうしを区別できます。

`-scope scope_name [argument...]`

オフライン保存またはオブジェクトの削除に使用されるスコープ（修正されたクエリ）を指定します。

指定するスコープに適切である場合のみ、引数を必要とします。たとえば、スコープ「`My working projects and products for a specified release`」には、引数であるリリース値を指定する必要があります。

引数は、スコープ定義で使用されているとおりの順序で指定する必要があります。

スコープの詳細な説明については、Synergy CLI ヘルプ、トラディショナル モードの「SOAD スコープ」を参照してください。

`-v|-verbose`

オブジェクトがリストに含められた理由またはリストから除外された理由を、詳しく説明するメッセージを生成するよう指定します。

例

- プレビューを行わずに、**Baselines marked for deletion and related projects and products** スコープによって見つかったオブジェクトを削除する。

  ```
  ccm soad -scope "Baselines marked for deletion and related projects and products"
  ```

関連トピック

- 削除用のプレビュー オブジェクト リストの作成
- プレビュー オブジェクト リストを使用した削除

説明と用途

オフライン保存と削除（SOAD）ユーティリティは、指定のスコープに基づいてオブジェクトを Synergy データベースから削除します。スコープは、どのオブジェクトが削除対象なのか、どの関連オブジェクトを同時に削除するか、また、どのオブジェクトを削除すべきでないかについての規則を決定します。Synergy の出荷時には、典型的なオブジェクト削除理由を示した定義済みのスコープが付いています。ユーザーが独自のスコープを作成することもできます。詳細については、soad_scope コマンド を参照してください。

オブジェクトを Synergy データベースから削除する理由には以下のようなものがあります。

- ディスク領域管理の一環またはバックアップ時間の短縮化のためにデータベースのサイズを縮小する。
- 使用予定のない *working* または *prep* 状態のプロジェクトを削除する。
- 不要な履歴またはベースラインを削除する。

以下に、削除できるデータ タイプの例を示します。

- 特定の開発者の不要な **Insulated Development**（個別開発）プロジェクト
- 不要な **Integration Testing**（統合テスト）プロジェクト
- 後からリリースされたバージョンによって階層が置き換えられている場合、古い静的プロジェクト階層とそれに関連する古いファイル
- 使用されていない古い製品
- 削除のマークが付いているベースライン
- 旧リリース用の**統合テスト（Integration Testing）**ベースライン

また、SOADではオブジェクトの削除を実行する前にオフラインで保存して、後でSynergyデータベースにリストアできるようにします。この機能を使用できるのは、*ccm_admin* ロールを持つユーザーであり、データベースを DCM 初期化している場合です。オブジェクトは DCM パッケージに類似したオフライン保存パッケージに保存されます。詳細は、パッケージの受取りを参照してください。

あるユーザーがどのスコープを使用できるかは、スコープの定義しているロールに従います。さらに、オブジェクトが削除可能になるタイミングは、通常の Synergy のセキュリティ規則で決まります。ユーザーは、任意の *working* 状態のオブジェクトと自分で所有しているオブジェクトを削除できます。ビルドマネージャの場合は、*prep* プロジェクトと *prep* オブジェクトも削除できます。さらに、*ccm_admin* ロールを持っているユーザーは、任意の非 *working* 状態オブジェクトと他人が所有しているオブジェクトを削除できます。SOAD にはさまざまな安全対策が組み込まれており、タイプ定義やその他の管理データなど、特定の種類のオブジェクトが削除されないようになっています。

`soad` コマンドを使用すると、以下の作業を実施できます。

- スコープを使用して削除をプレビューできる。プレビューでは、データベースを変更せずに、どのオブジェクトが削除用に選択されているかを表示できます。オブジェクトを実際に削除する前に、削除対象が期待どおりかどうかを確認できます。

- 以前実行したプレビューで見つかったオブジェクトを削除できる。
- スコープを使用してプレビューを行わずにオブジェクトを削除できる。プレビュー結果を確認する必要がない場合に、この方法を使います。削除を迅速に行える点で便利ですが、削除対象のプレビューによる確認が不要であることが確実でなければなりません。削除対象のオブジェクトを事前確認できないことがこの方法の欠点です。

関連トピック

- soad_scope コマンド

soad_scope コマンド

詳細については、説明と用途を参照してください。soad_scope コマンドは、以下のサブコマンドをサポートします。

- スコープの作成
- スコープの削除
- スコープの一覧表示
- スコープの修正
- スコープの表示

スコープの作成

このサブコマンドにより、soad スコープを作成します。

> **注意！** 新しいスコープを作成するときは、そのスコープを使用してプレビューを行ってスコープをテストしてください。維持すべきデータの削除を防止するために、必ずスコープに除外ルールを設けてください。

```
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`*

1 つまたは複数の除外ルールを指定します。除外ルールにより、関連するオブジェクトが初期オブジェクト リストから削除されます。

たとえば、リリース名を最初のパラメータ（`release='%1'`）にしたクエリにより、指定したリリースに関するすべてのオブジェクトが検索されたとします。除外ルールを適用して、そのスコープを制限できます。具体的には、他のプロジェクトによって使用されているフォルダとタスク、他のフォルダによって使用されているタスクまたは他のオブジェクトと関連しているタスク、他の非静的なプロジェクトによって使用されているベースライン、他の保存されているベースラインの一部であるオブジェクトを、スコープから削除します

`-expand|-expansion_rules` *`expansion_rules`*

1 つまたは複数の展開ルールを指定します。展開ルールにより、関連するオブジェクトが初期オブジェクト リストに追加されます。

たとえば、リリース名を最初のパラメータ（`release='%1'`）にしたクエリにより、指定したリリースに関するすべてのオブジェクトが検索されたとします。それにプロジェクトのフォルダとタスク、プォルダのタスク、およびタスクのオブジェクトを含める展開ルールを追加することにより、スコープを展開できます。

`-exclude_query|-exclusion_query` *`exclusion_query`*

スコープからオブジェクトを削除するために使用するクエリを指定します。

たとえば、`requirements` という属性名を持つオブジェクトをスコープから除外するには、以下のクエリ式を指定します。

```
has_attr('requirements')
```

SOAD がオブジェクト名、クエリ、またはルールを評価するときには、以下の否定文節要素を追加します。

```
and not has_attr('requirements')
```

`-object` *four_part_name*

初期オブジェクト リストに使用するオブジェクトの名前を指定します。(例、`%1`)。結果として得られる展開された文字列はオブジェクトの有効な 4 部名称でなければなりません。

たとえば、最初のパラメータ（`%1`）として指定したプロジェクト オブジェクト名を使用して、そのプロジェクト オブジェクト名の初期オブジェクト リストを設定できます。

`-parameters` *parameters*

`-object`、`-query`、`-exclude_query`、およびそれらの定義に関する引数のラベルを供給します。

たとえば、"`All objects for specified release`" スコープ内で使用されているクエリに関して、以下に示すようにスコープを定義し、1 つのパラメータ ラベル `Release Value` を付けます。

```
ccm soad_scope -create "All objects for specified release"
-parameters "Release Value" -query "release='%1'" other_options
```

次に、以下の `ccm soad -delete` コマンドでスコープを使用します。ここで、**2.3** はリリース値を示します。

```
ccm soad -delete -scope "All objects for specified release" 2.3
```

`-pn|-package_name` *package_name*

該当スコープに関して、オブジェクトを保存する先の DCM パッケージの名前を指定します。パッケージ名にキーワードを含めることができますます。

`-query` *query*

初期オブジェクト リストを定義する、クエリ式を指定します。

たとえば、指定されたリリースに関して、現在のユーザーのプロジェクトおよび製品を初期オブジェクト リストに含めるためには、以下のクエリ式を指定します。

```
(cvtype='project' or is_product=TRUE) and owner='%user' and
status='working' and release='%1'
```

`-roles` *role*

スコープを使用できるロールを指定します。デフォルトでは、*ccm_admin* ロールを持つユーザーのみがこのスコープを変更できます。

scope_name

オフライン保存と削除に関するスコープを指定します。

OS によって制限されていない文字のみを使用します。

この名前はスコープのファイル名として使用されます。空白文字と句読点文字は下線付きの 16 進コードに変換されます。たとえば、スコープ名を **This is my test scope** とした場合、作成されるファイル名は **This_0020is_0020my_0020_test_0020scope.xml** となります。

関連トピック

- スコープの削除
- スコープの一覧表示
- スコープの修正
- スコープの表示

スコープの削除

このサブコマンドにより、soad スコープを削除します。

```
ccm soad_scope|save_offline_and_delete_scope -d|-delete scope_name
```

scope_name

削除するスコープの名前を指定します。

関連トピック

- スコープの作成
- スコープの一覧表示
- スコープの修正
- スコープの表示

除外ルールの一覧表示

このサブコマンドにより、soad スコープで使用可能な有効な除外ルールの名前を一覧表示します。

```
ccm soad_scope|save_offline_and_delete_scope -l|-list
              -exclude|-exclusion_rules

-exclude|-exclusion_rules
```

除外ルールを一覧表示します。

関連トピック

- スコープの作成
- スコープの表示

展開ルールの一覧表示

このサブコマンドにより、soad スコープで使用可能な有効な展開ルールの名前を一覧表示します。

```
ccm soad_scope|save_offline_and_delete_scope -l|-list
              -expand|-expansion_rules
```

`-expand|-expansion_rules`

展開ルールを一覧表示します。

関連トピック

- スコープの作成
- スコープの表示

スコープの一覧表示

このサブコマンドにより、定義されている soad スコープの名前を一覧表示します。

```
ccm soad_scope|save_offline_and_delete_scope -l|-list -scope
```

`-scope`

一覧表示するスコープを指定します。

関連トピック

- スコープの作成
- スコープの表示

スコープの修正

このサブコマンドにより、soad スコープのプロパティを変更します。

```
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`

1 つまたは複数の除外ルールを指定します。除外ルールにより、関連するオブジェクトが初期オブジェクト リストから削除されます。

たとえば、リリース名を最初のパラメータ（`release='%1'`）にしたクエリにより、指定したリリースに関するすべてのオブジェクトが検索されたとします。除外ルールを適用して、そのスコープを制限できます。具体的には、他のプロジェクトによって使用されているフォルダとタスク、他のフォルダによって使用されているタスクまたは他のオブジェクトと関連しているタスク、他の非静的なプロジェクトによって使用されているベースライン、他の保存されているベースラインの一部であるオブジェクトを、スコープから削除します

`-expand|-expansion_rules expansion_rules`

1 つまたは複数の展開ルールを指定します。展開ルールにより、関連するオブジェクトが初期オブジェクト リストに追加されます。

たとえば、リリース名を最初のパラメータ（`release='%1'`）にしたクエリにより、指定したリリースに関するすべてのオブジェクトが検索されたとします。それにプロジェクトのフォルダとタスク、フォルダのタスク、およびタスクのオブジェクトを含める展開ルールを追加することにより、スコープを展開できます。

`-exclude_query|-exclusion_query "query_expression"`

スコープからオブジェクトを削除するために使用するクエリを指定します。

たとえば、`requirements` という属性名を持つオブジェクトをスコープから除外するには、以下のクエリ式を指定します。

```
has_attr('requirements')
```

SOAD がオブジェクト名、クエリ、またはルールを評価するときには、以下の否定文節要素を追加します。

```
and not has_attr('requirements')
```

`-object` *four_part_name*

初期オブジェクト リストに使用するオブジェクトの名前を指定します。(例、`%1`)。結果として得られる展開された文字列はオブジェクトの有効な 4 部名称でなければなりません。

たとえば、最初のパラメータ（`%1`）として指定したプロジェクト オブジェクト名を使用して、そのプロジェクト オブジェクト名の初期オブジェクト リストを設定できます。

`-parameters` *parameters*

`-object`、`-query`、`-exclude_query`、およびそれらの定義に関する引数のラベルを供給します。

たとえば、`All objects for specified release` スコープ内で使用されているクエリに関して、以下に示すようにスコープを定義し、1 つのパラメータ ラベル `Release Value` を付けます。

```
ccm soad_scope -create "All objects for specified release"
-parameters "Release Value" -query "release='%1'" other_options
```

次に、以下の `ccm soad -delete` コマンドでスコープを使用します。ここで、**2.3** はリリース値を示します。

```
ccm soad -delete -scope "All objects for specified release" 2.3
```

`-pn|-package_name` *package_name*

該当スコープに関して、オブジェクトを保存する先の DCM パッケージの名前を指定します。パッケージ名にキーワードを含めることができますます。

`-query` *query*

初期オブジェクト リストを定義する、クエリ式を指定します。

たとえば、指定されたリリースに関して、現在のユーザーのプロジェクトおよび製品を初期オブジェクト リストに含めるためには、以下のクエリ式を指定します。

```
(cvtype='project' or is_product=TRUE) and owner='%user' and
status='working' and release='%1'
```

`-roles` *role*

スコープを使用できるロールを指定します。デフォルトでは、*ccm_admin* ロールを持つユーザーのみがこのスコープを変更できます。

scope_name

オフライン保存と削除に関するスコープを指定します。

OS によって制限されていない文字のみを使用します。

この名前はスコープのファイル名として使用されます。空白文字と句読点文字は 16 進コードの下線に変換されます。たとえば、スコープ名を **This is my test scope** とした場合、作成されるファイル名は **This_0020is_0020my_0020_test_0020scope.xml** となります。

関連トピック

- スコープの削除
- スコープの一覧表示
- スコープの表示

スコープの表示

このサブコマンドにより、soad スコープの詳細を表示します。

```
ccm soad_scope|save_offline_and_delete_scope -s|-sh|-show scope_name...
```

scope_name...

すべてのスコープを以下の詳細とともに表示します。

- ロール
- パラメータ ラベル
- オブジェクト
- クエリ
- 展開ルール
- 除外ルール
- 除外クエリ
- パッケージ名

複数のスコープ名を表示するよう指定できます。

例

- スコープ **Baselines marked for deletion and related projects and products** の詳細を表示する。

  ```
  ccm soad_scope -show "Baselines marked for deletion and related projects and products"
  ```

関連トピック

- スコープの作成
- スコープの削除
- スコープの一覧表示
- スコープの修正

説明と用途

soad_scope コマンドにより、オブジェクトのオフライン保存と削除の際に使用するスコープ（適用範囲）を編集、作成、削除します。

スコープを編集、作成、または削除できるのは、*ccm_admin* ロールで作業している場合のみです。

set コマンド

詳細については、説明と用途を参照してください。set コマンドは、以下のサブコマンドをサポートします。

- オプションの設定
- オプションの表示
- オプションの表示

オプションの設定

このサブコマンドにより、オプション値を設定します。たとえば、自分のロールを *developer* に設定するには、**ccm set role developer** を使用します。オプション値は、設定するオプションに対して有効である必要があります。詳細については、デフォルト オプションを参照してください。

```
ccm set option value
```

option

設定するオプションの名前を指定します。詳細については、デフォルト設定を参照してください。

value

設定するオプションの値を指定します。詳細については、デフォルト設定を参照してください。

例

- 自分のロールを *developer* に設定する。

  ```
  ccm set role developer
  ```

- 自分の現在のロールを表示する。

  ```
  ccm set role
  developer
  ```

関連トピック

- オプションの表示
- オプションの表示
- オプションの設定解除

オプションの表示

このサブコマンドにより、オプションの値を表示します。

```
ccm set option
```

option

表示するオプションの名前を指定します。詳細については、デフォルト設定を参照してください。

例

- `text_editor` の値を表示する。

  Windows :
  ```
  ccm set text_editor
  notepad %filename
  ```

  UNIX :
  ```
  ccm set text_editor
  vi %filename
  ```

関連トピック

- オプションの設定
- オプションの表示
- オプションの設定解除

オプションの表示

このサブコマンドにより、使用可能なオプションを表示します。

```
ccm set
```

関連トピック

- オプションの設定
- オプションの表示
- オプションの設定解除

説明と用途

オプションにより、特定の Synergy 操作の動作を制御します。オプションによっては、現在の CLI セッションのみに適用され、次のセッションでは維持されない場合があります。また、データベースに保存されたユーザー環境を維持するオプションもあります。オプションの中には、定義済みで読み取り専用のものや、修正できないものもあります。

設定可能なオプションには、`text_editor`、`text_viewer`、`role`、`verbosity` などがあります。オプションの総合的なリストおよびオプションを設定する方法と場所については、デフォルト設定を参照してください。

show コマンド

詳細については、説明と用途を参照してください。show コマンドは、以下のサブコマンドをサポートします。

- プロジェクトの表示
- タイプの表示

プロジェクトの表示

このサブコマンドにより、特定の条件と一致するデータベース内のすべてのプロジェクトを表示します。クエリに関連するオプションを指定しない場合、すべてのプロジェクトが表示されます。

クエリ関連の各オプションはクエリ式を組み立てるために使用されます。たとえば、`-name` *`name`* という記述は `name='name'` のクエリ式になります。こういったクエリ オプションを繰り返して使用すると、`or` で結合されたクエリ節になります。たとえば、`-name example.txt -name another.txt` という記述はクエリ式 "`(name='example.txt' or name='another.txt')`" になります。

異なるオプションを指定した場合は、クエリ節は `and` で結合されます。たとえば、`-n example.txt -s working` という記述は、クエリ式 "`(name='example.txt')` and `(status='working')`" になります。

```
ccm show -p|-projects [(-o|-owner owner)...][(-n|-name name)...]
         [(-v|-version version)...][(-s|-state state)...]
         [(-task task_spec)...][-f|-format format] [-nf|-noformat]
         ([-ch|-column_header] | [-nch|-nocolumn_header])
         [-sep|-separator separator] ([-sby|-sortby sortspec] |
         [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat]
```

`-ch|-column_header`

出力フォーマットでカラム ヘッダーを使用するよう指定します。詳細については、-ch|-column_headers を参照してください。

`-f|-format` *`format`*

コマンドの出力フォーマットを指定します。詳細については、-f|-format を参照してください。

キーワードには、組み込み済みのもの（`%fullname`、`%displayname`、`%objectname`）、あるいは `%modify_time`、`%status` などの既存の属性の名前を使用できます。

キーワードのリストについては、組み込み済みキーワードを参照してください。

`-gby|-groupby` *`groupformat`*

コマンドの出力をグループ化する方法を指定します。詳細については、-gby|-groupby を参照してください。

`-n|-name` *`name`*

"`name='name'`" 形式のクエリ文節を含めて、指定した名前を持つプロジェクトを検索します。

`-nch|-nocolumn_header`

出力フォーマットでカラム ヘッダーを使用しないよう指定します。詳細については、-ch|-column_headers を参照してください。

`-nf|-noformat`

カラム配置を使用しないよう指定します。詳細については、-nf|-noformat を参照してください。

`-ns|-nosort|-no_sort`

コマンドの出力をソートしないよう指定します。詳細については、-ns|-nosort を参照してください。

`-o|-owner` *`owner`*

"`owner='owner'`" 形式のクエリ文節を含めて、指定した所有者のプロジェクトを検索します。

`-sby|-sortby` *`sortspec`*

コマンドの出力をソートする方法を指定します。詳細については、-ns|-nosort を参照してください。

`-sep|-separator` *`separator`*

`-f|-format` オプションと一緒にのみ使用します。異なる区切り文字を指定します。詳細については、-sep|-separator を参照してください。

`-s|-state` *`state`*

"`status='state'`" 形式のクエリ文節を含めて、指定した状態のプロジェクトを検索します。

`-task` *`task_spec`*

"`is_associated_cv_of(task('task_spec'))`" 形式のクエリ文節を含めて、指定したタスクに関連付けられたプロジェクトを検索します。*`task_spec`* には、1 つまたは複数のタスクを指定できます。詳細については、タスクの指定 を参照してください。

`-v|-version` *`version`*

"`version='version'`" 形式のクエリ文節を含めて、指定したバージョンのプロジェクトを検索します。

例

- 状態が *integrate* であり、所有者が *john* である、データベース内のプロジェクトを表示する。

```
ccm show -p -s integrate -o john
1) projY-1   integrate john project projY 1 2
2) projY-2   integrate john project projY 1 7
3) projY-2.1 integrate john project projY 1 8
```

- 状態が *integrate*、所有者が *john* であり、タスク 8 に関連付けられている、データベース内のプロジェクトを表示する。

```
ccm show -p -s integrate -o john -task 8
1) projY-2.1 integrate john project projY 1 8
```

- データベース内に定義されているタイプを表示する。

```
ccm show -t

ascii
binary
c++
csrc
dir
executable
incl
library
lsrc
makefile
project
relocatable_obj
shared_library
shsrc
symlink (UNIX)
ysrc
```

タイプの表示

このサブコマンドにより、データベースに定義されているオブジェクト タイプを表示します。

```
ccm show -t|-types
```

説明と用途

show コマンドにより、プロジェクトの特定属性の設定を表示したり、データベース内のすべてのタイプを表示します。

show_servers コマンド

詳細については、説明と用途を参照してください。show_servers コマンドは、サーバーの表示 サブコマンドをサポートします。

サーバーの表示

このサブコマンドにより、ルーターが認識しているデータベースおよび関連する Rational Synergy サーバーを表示します。このコマンドを使用するために、Rational Synergy CLI セッションを実行する必要はありません。

このコマンドを使用して、Rational Synergy サーバーが提供するデータベース、および各データベースの Synergy CLI セッションを開始するために指定すべきサーバー URL を表示します。

```
ccm show_servers
```

説明と用途

`show_servers` コマンドは、ルーターが認識しているデータベース、および関連する Synergy サーバーを表示するために使用します。この種の情報は、使用すべきデータベースを知るため、またセッション開始時に指定すべきサーバー URL を見つけるために必要となります。

start コマンド

詳細については、説明と用途を参照してください。`start` コマンドは、CLI セッションの開始 サブコマンドをサポートします。

CLI セッションの開始

```
ccm start [-q] -d database_path -s server_url [-m]
          [-pw password] [-n username] [-r initial_role] [-fpw]
```

`-d database_pathname`

データベースの絶対パスを指定します。UNIX サーバー上のデータベース ホストの場合は、UNIX の絶対パスを使用します。WIndows サーバー上のデータベース ホストの場合は、UNC パスを使用します。

`-fpw`

デフォルトの `.ccmrc` ファイルの代わりに使用するパスワード ファイルへのパスを指定します。

`-m`

複数のセッションが使用されるよう指定します。使用するセッションに `CCM_ADDR` 環境変数を設定する必要があります。省略すると、別のセッションの `CCM_ADDR` が定義されず、セッション情報がクライアントの `.ccm.addr` ファイルに書き込まれます。これはデフォルトのアドレスとして使用されます。

`-n user_name`

Windows では、使用する Synergy ユーザー名を指定します。UNIX では、Synergy ユーザー名はオペレーティング システムのユーザー名と同じなので、`-n` を指定する必要はありません。`-n` を指定する場合は、ユーザー名は現在のユーザーと一致する必要があります。

`-pw password`

Synergy ユーザーのパスワードを指定します。パスワードを指定しない場合、デフォルトのパスワードは `.ccmrc` ファイルから取得されます。`-fpw` オプションを指定すると、そのファイルがパスワード ファイルとして使用されます。`-fpw` オプションを指定しない場合、UNIX では `$HOME/.ccmwc`、Windows では `%HOMEPATH%¥.ccmrc` が使用されます (デフォルトのパスワードは `ccm set_password` コマンドを使って定義します。このコマンドの詳細については、『Rational Synergy 管理者ガイド UNX 版』を参照してください)。

`-q`

quiet モードでセッションを開始します。このオプションを使用した場合、Rational Synergy は CLI セッション用の `CCM_ADDR` を表示します。このオプションを使用しない場合、開始時、`CCM_ADDR` の前に著作権情報などの追加情報が表示されます。

`-r initial_role`

使用する初期ロールを指定します。ロールを指定しない場合、デフォルトの初期ロールが使用されます。これは、データベース ユーザー リストで定義されている最初のロールに基づきます。

`-s server_url`

接続するサーバーを指定します。*server_url* は、`http://` または `https://` から始まる有効な URL であることを確認します。

接続する Synergy サーバーの詳細については、適切な『Rational Synergy 管理者ガイド』の「CCM サーバーについて」を参照してください。

例

- サーバー URL を使用して Rational Synergy を起動する。

  ```
  ccm start -d /data/db1 -n bob -pw **** -s http://unixXYZ:8400
  ```

- 別のセッションを実行しながら、サーバー URL を *quiet* モードで使用して、Rational Synergy セッションを開始する。

  ```
  ccm start -d ¥¥ccmdb¥db1 -n bob -pw **** -s http://winXYZ:8400 -q -m
  ```

警告

追加の セッションを開始し、コマンドラインを使用しようとすると、警告メッセージが表示されます。新しいセッションの `CCM_ADDR` 変数を、以下のように、Rational Synergy の start コマンドによって表示された、以下のようなアドレスに設定します。

```
set CCM_ADDR=prefect.cwi.com:1368
```

これにより、Rational Synergy コマンドは、それまで実行していたセッションの代わりに、新しいセッションによって実行されるようになります。

ユーザー *ccm_root* として実行するときは、常に `-m` オプションを使用し、環境内の `CCM_ADDR` を必ず設定します。それにより、自分の *ccm_root* セッションが他のユーザーが実行している *ccm_root* セッションから区別されます。

環境変数

`CCM_ADDR`

ファイル

`ccm.properties`

`ccm.user.properties`

`.ccm.rc`

関連トピック

- stop コマンド

説明と用途

start コマンドにより、Rational Synergy CLI セッションを開始します。セッションの開始後、コマンド ウィンドウ（Windows）またはセッションを開始したシェル（UNIX）に、ＣＬＩ セッションの一意の識別子である Rational Synergy アドレス（CCM_ADDR）が表示されます。

複数の Rational Synergy セッションを実行する場合、CCM_ADDR 環境変数を設定して、どのセッションで Rational Synergy コマンドを使用するかを指定します。CCM_ADDR を設定しない場合、デフォルトのアドレスが使用されます。このデフォルトのアドレスは .ccm_addr ファイルから読み込まれます。このファイルは、-m（複数セッション）オプションなしで CLI セッションを開始した場合に生成されます。

stop コマンド

詳細については、説明と用途を参照してください。`stop` コマンドは、CLI セッションの終了サブコマンドをサポートします。

CLI セッションの終了

このサブコマンドにより、CLI セッションを終了します。

```
ccm stop|quit
```

例

- 現在の Rational Synergy CLI セッションを終了する。
  ```
  ccm stop
  ```

関連トピック

- start コマンド

説明と用途

`stop` コマンドにより、Rational Synergy セッションを終了します。

sync コマンド

詳細については、説明と用途を参照してください。sync コマンドは、ワークエリアの書き換えサブコマンドをサポートします。

ワークエリアの書き換え

sync コマンドにより、プロジェクトのワークエリアを完全に書き換えます。すべてのプロジェクトのワークエリアが作成されるデフォルトのディレクトリは、ユーザーのホームディレクトリ内の ccm_wa ディレクトリの下のデータベース名の付いたディレクトリです。sync コマンドを使用するのは、ワークエリアを手動で同期したい場合です。

> **注記**：ビルドマネージャまたは *ccm_admin* のロールを持つユーザーのみが、書き込み禁止プロジェクトの同期を取ることができます。

```
ccm sync [-r|-recurse] [-nr|-norecurse|-no_recurse] [-s|-static]
         [-p|-project] project_spec...
```

`-nr|-norecurse|-no_recurse`

プロジェクト同期時にプロジェクト階層の再帰処理を行わないよう指定します。指定したプロジェクトのみを同期させます。

これはデフォルト設定です。

`-p|-project` *`project_spec`*`...`

同期させるプロジェクトを指定します。

プロジェクト指定である 1 つまたは複数の引数を指定できます。各引数には、1 つまたは複数のプロジェクトを指定できます。詳細については、プロジェクトの指定を参照してください。

`-r|-recurse`

プロジェクト階層内のすべてのオブジェクトを、指定したプロジェクトと同期させます。

`-s|-static`

既存の静的なワークエリアをデータベースの現在のデータで更新します。静的なワークエリアとは、静的なサブプロジェクトのワークエリアのローカル コピーです。また、ccm sync コマンドの実行対象の階層内のすべての静的ワークエリアを更新します。これによって、1 つのコマンドを使用して、階層内のすべての静的ワークエリアを完全に同期させます。階層内に静的ワークエリアが存在しない場合には、このオプションは無視されます。

例

- toolkit-linda のワークエリアとそのサブプロジェクトを同期させる。

  ```
  ccm sync -recurse -project toolkit-linda
  ```

- 指定したプロジェクトのワークエリアを作成する。

```
ccm sync -p ico_aug1-1
```

デフォルト

`ccm.ini` ファイル（Windows）または `.ccm.ini` ファイル（UNIX）内で、以下の関連オプションを設定できます。

- `wa path template`

関連トピック

- ワークエリア プロパティの修正
- ワークエリア プロパティの表示

説明と用途

プロジェクトを作成したとき、および checkout コマンドを使用してプロジェクトをチェックアウトしたときに、ワークエリアが自動的に作成されます。プロジェクトに新しいメンバーを追加すると、ワークエリアは自動的に更新されます。

以下の場合に、ワークエリアを手動で同期させる（強制的に同期する）必要があります。

- ワークエリア内の一部または全部のオブジェクトを削除した場合

  強制的に同期を取ると、データベース内の必要なオブジェクトのみがワークエリアに書き込まれます。

- ワークエリア パスの変更時に `work_area` コマンドが失敗した場合

  CLI または GUI のいずれかでワークエリア パスを変更すると、Rational Synergy は新しい場所へのワークエリア パスの更新を試みます。もし別のアプリケーションが古いワークエリア パスを使用している場合、移動はエラーとなり、ワークエリアを同期する必要が生じます。

- ワークエリアのタイプを、ローカル コピー使用からシンボリック リンク使用（またはその逆）に変更した場合

  ワークエリアのタイプを変更したい場合には、以下の手順を行います。

  1. 現状に応じて、ローカル コピーまたはシンボリック リンクのいずれかを使用して、ワークエリアをリコンサイルします。
  2. ファイル システムからワークエリア オブジェクトを削除します。
  3. ワークエリアのパスとオプションを設定します。
  4. 好みに応じて、ローカル コピーまたはシンボリック リンクのいずれかのクライアントを使用して、新しいセッションを開始します。
  5. 同期を強制実行（sync コマンドを実行）してワークエリアを作成し直します。

     注記：CLI から 同期を中止するには、任意の時点で Ctrl + C キーを押します。

     同期処理を中止すると、ワークエリア内でエラーが発生した可能性があることを知らせる、エラー メッセージが表示されます。エラーはワークエリアを使用し始めてから発生します。したがって、問題発生を回避するために、使用前にワークエリアを完全に同期してください。

task コマンド

詳細については、説明と用途を参照してください。task コマンドは、以下のサブコマンドをサポートします。

- タスクの割り当て
- オブジェクト、タスク、変更依頼へのタスク関連付け
- タスクの完了
- タスクのコピー
- タスクの作成
- オブジェクト、タスク、変更依頼とタスクとの関連付けの解除
- タスクの修正
- タスクの変更
- タスクのクエリ
- カレント タスクの設定または解除
- タスク プロパティの表示
- タスクに関連付けられているオブジェクト、変更依頼、タスクの表示
- タスク情報の表示
- 別の状態へのタスクの遷移

タスクの割り当て

このサブコマンドにより、指定したタスクを指定した担当者に割り当てます。この操作を実行できるのは、タスク割り当て権限を持つユーザーとタスクを修正できるユーザーです。

```
ccm task -as|-assign -t|-to resolver [-q|-quiet] task_spec...
```

`-t|-to resolver`

タスクに割り当てる担当者を指定します。*resolver* は有効なタスク担当者である必要があります。

`-q|-quiet`

確認メッセージに、割り当てられた各タスクのタスク ID だけを含むよう指定します。

task_spec`...`

割り当てるタスクを指定します。*task_spec* には、複数のタスクを指定できます。詳細については、タスクの指定を参照してください。

例

- ユーザー *joe* にタスク **54**、**60-63**、および **74** を割り当てる。

```
ccm task -as 54,60-63,74 -to joe
```

```
Assigned task 54
Assigned task 60
Assigned task 61
Assigned task 62
Assigned task 63
Assigned task 74
```

関連トピック

- query コマンド

オブジェクト、タスク、変更依頼へのタスク関連付け

このサブコマンドは、指定したタスクを、指定のオブジェクト、変更依頼、または修正対象タスクに関連付けます。オブジェクトとの関連付けの場合、タスクの修正権限を持つユーザーがこの操作を実行できます。変更依頼との関連付けの場合、変更依頼がタスクの関連付けが可能な状態にあれば、変更依頼の修正権限を持つユーザーがこの操作を実行できます。

また、あるタスクを修正対象タスクに関連付けるための要件は以下のとおりです。

- 相互に関連するタスクは異なるデータベースからのものでよい。
- 修正対象タスクの状態は *completed*（完了）または *excluded*（除外）のどちらかである必要がある。
- 修正タスクはこの関係を設定するユーザーによって修正可能である必要がある。
- 1 つのタスクは 1 つのタスクのみ修正できる。

```
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

タスクに関連付ける変更依頼を指定します。*change_request_spec* には、複数の変更依頼を指定できます。詳細については、変更依頼の指定を参照してください。

file_spec

タスクに関連付けるオブジェクトを指定します。オブジェクトはプロジェクト、ディレクトリ、ファイルのいずれでもかまいません。詳細については、ファイルの指定を参照してください。

`-fixes` *task_spec*

修正するタスクを指定します。*task_spec* には、1 つのタスクを指定できます。詳細については、タスクの指定を参照してください。

`-prob|-problem|-change_request`

タスクに関連付ける変更依頼を指定します。

例

- タスク **17** をオブジェクト `MAIN.C-3:csrc:1` と関連付ける。

  ```
  ccm task -a 17 -obj MAIN.C-3:csrc:1
  ```

- タスク **54** を変更依頼 **D#1231** と関連付ける。

```
ccm task -associate 54 -change_request D#1231
```

関連トピック

- query コマンド

タスクの完了

このサブコマンドにより、タスクの関連オブジェクトを修正不可状態にチェックインし、そのタスクを *completed*（完了）状態に遷移させて、タスクを完了します。このサブコマンドは、タスクの担当者またはアドミニストレータが実行できます。引数のキーワード `current` または `default` は、カレント タスクを完了することを意味します。

```
ccm task -ci|-checkin|-complete [-time|-time_actual task_duration]
        [-c|-comment comment_string] [-ce|-commentedit]
        [-cf|-commentfile file_path] [-commentreplace]
        (task_spec|(current|default))...
```

`-c|-comment comment`

すべてのベースライン プロジェクトとそのメンバーを *released* 状態にチェックインするとき、それらに追加するコメントを指定します。*comment* には、複数行を含むことができ、円記号でコード化した値を使用できます。

このオプションは、`-commentedit` および `-commentfile` と一緒に使用できます。`-commentedit` オプションを使用した場合、コメントはデフォルトのテキスト エディタで表示されます。

`-ce|-commentedit`

コメントの作成と編集用にデフォルトのテキスト エディタを起動するよう指定します。テキスト エディタで保存した結果が最終的なコメントとして使用されます。このオプションは、`-comment` および `-commentfile` オプションと一緒に使用できます。

`-cf|-commentfile file_path`

指定したファイルの内容をコメントとして使用するよう指定します。`-comment` を指定した場合、それがコメントに追加されます。このオプションは、`-commentedit` オプションと一緒に使用できます。

`-commentreplace`

コメントを置き換えるよう指定します。

`-time|-time_actual task_duration`

タスクの完了に必要な時間を指定します。*task_duration* は任意の文字列です。ただし、報告とメトリクスを容易にするために、フォーマットと単位には一貫した規則を使用するようにしてください。

task_spec|(current|default)

完了するタスクを指定します。current または default はカレント タスクを完了することを意味します。*task_spec* には、複数のタスクを指定できます。詳細については、タスクの指定を参照してください。

例

- タスク **40** に関連付けられたすべてのオブジェクトをチェックインする。

  ```
  ccm task -complete 40 -comment "The problem is fixed."
  ```

- カレント タスクを完了する。

  ```
  ccm task -complete default
  ```

関連トピック

- query コマンド

タスクのコピー

このサブコマンドにより、指定したタスクをコピーして新しいタスクを作成します。リリース向けに修正したタスクを別のリリースに適用する必要が生じたときに、タスクをコピーします。コピーされたタスクおよび元のタスクは、同じオブジェクトと関連していることも、別のオブジェクトと関連していることも、その両方であることもあります。デフォルトでは、既存のタスクに関連付けられているオブジェクトは対応するコピーされたタスクにも関連付けられます。

```
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`*

作成するタスクの完了予定日を指定します。完了予定日を指定しない場合、コピー元のタスクの完了予定日が適用されます。*`date_estimate`* は有効な日付である必要があります。

`-def|-default|-current`

タスクのコピーによって作成される最初のタスクがこの CLI セッションのカレントタスクとなるよう指定します。

`-desc|-description` *`description`*

1 行の説明を指定します。復帰改行文字を含めることはできません。

`-desc_edit|-description_edit`

複数行の説明の編集または作成用に、デフォルトのテキスト エディタを起動するよう指定します。

`-desc_file|-description_file` *`file_path`*

複数行の説明を含むファイルへのパスを指定します。

`-no_objects`

コピー元のタスクに関連付けられているオブジェクトが、コピーによって作成されるタスクに関連付けられないよう指定します。

`-p|-priority` *`priority`*

新しいタスクの優先度を指定します。優先度を指定しない場合、コピー元のタスクの優先度が適用されます。優先度は有効なタスク優先度である必要があります。デフォルトの有効な優先度は `High`、`Medium`、`Low` です。

`-plat|-platform` *`platform`*

作成するタスクのプラットフォームを指定します。プラットフォームを指定しない場合、コピー元のタスクのプラットフォームが適用されます。プラットフォームは有効なプラットフォームである必要があります。

`-prob|-problem|-change_request` *`change_request_spec`*

新しいタスクを、指定した変更依頼に関連付けるよう指定します。変更依頼は自分が修正可能であり、タスクの関連付けが可能な状態にある必要があります。*`change_request_spec`* には、1 つの変更依頼を指定できます。詳細については、変更依頼の指定を参照してください。変更依頼を指定せず、コピー元のタスクが *assigned* 状態の変更依頼に関連付けられている場合、新しいタスクも同じ変更依頼に関連付けられます。

`-quiet`

確認メッセージに、作成した各タスクのタスク ID だけを含むよう指定します。

`-register`

タスクを *registered*（登録済み）状態で作成するよう指定します（この状態のタスクは Synergy に登録され、誰にも割り当てられていない状態です）。`-register` を指定しないと、コピー元のタスクと同じユーザーまたは `-resolver` オプションに指定されたユーザーに、タスクが割り当てられます。

`-rel|-release` *`release_spec`*

作成するタスクのリリース値を指定します。リリース値を指定しない場合、コピー元のタスクのリリース値が適用されます。*`release_spec`* には、1 つのリリース値を指定できます。詳細については、リリースの指定を参照してください。

`-r|-resolver` *`resolver`*

タスクの解決を担当するユーザーを指定します。担当者を指定しない場合、コピー元のタスクの担当者が適用されます。*`resolver`* は有効なタスク担当者である必要があります。

`-sub|-subsystem` *`subsystem`*

作成するタスクが属するタスク サブシステムを指定します（例、Any、GUI code、CLI code、または documentation）。サブシステムを指定しない場合、コピー元のタスクのサブシステムが適用されます。サブシステムは有効なタスク サブシステムである必要があります。

`-s|-synopsis` *`synopsis`*

コピーするタスクの概要を指定します。概要には、復帰改行文字を含まない任意の文字列を使用できます。

`task_spec...`

コピーするタスクを指定します。*`task_spec`* には、複数のタスクを指定できます。詳細については、タスクの指定を参照してください。

`-time|-time_estimate` *`time_estimate`*

作成するタスクを完了するために要する予定時間を指定します。完了予定時間を指定しない場合、コピー元の完了予定時間が適用されます。*`time_estimate`* には任意の文字列を使用できます。ただし、報告とメトリクスを容易にするために、フォーマットと単位には一貫した規則を使用するようにしてください。

例

- タスク **40** をコピーし、コピー元とは別の概要、リリース、担当者、説明を指定する。タスクに関連付けられているオブジェクトはコピーしません。

```
ccm task -copy 40 -synopsis "Fix GUI color problem" -release 2.0 -
resolver donho -no_objects -description "check RGB module"
Task hawaii#50 created.
```

関連トピック

- query コマンド

タスクの作成

このサブコマンドにより、タスクを作成します。担当者を指定すると、指定した担当者にタスクが割り当てられます。タスク作成時に、担当者を指定しないか `-register` を指定した場合、タスクは Synergy に登録されますが、担当者には割り当てられません。

```
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`*

完了予定日を指定します。*`date_estimate`* は有効な日付である必要があります。

`-def|-default|-current`

作成するタスクがこの CLI セッションのカレント タスクとして設定されるよう指定します。

`-desc|-description description`

`-desc edit|-description edit`

`-desc file|-description file file path`

`-plat|-platform` *`platform`*

プラットフォームを指定します。プラットフォームは有効なプラットフォームである必要があります。

`-p|-priority` *`priority`*

優先度を指定します。優先度は有効なタスク優先度である必要があります。デフォルトの有効な優先度は `High`、`Medium`、`Low` です。

`-prob|-problem|-change_request` *`change_request_spec`*

新しいタスクを、指定した変更依頼に関連付けるよう指定します。変更依頼は自分が修正可能であり、タスクの関連付けが可能な状態にある必要があります。*`change_request_spec`* には、1 つの変更依頼を指定できます。詳細については、変更依頼の指定を参照してください。

`-quiet`

確認メッセージに、作成した各タスクのタスク ID だけを含むよう指定します。

`-register`

タスクが registered（登録済み）状態で作成されるよう指定します。

`-rel|-release` *`release_spec`*

リリース値を指定します。*`release_spec`* には、1 つのリリース値を指定できます。詳細については、リリースの指定を参照してください。

`-r|-resolver` *`resolver`*

タスクの解決を担当するユーザーを指定します。*`resolver`* は有効なタスク担当者である必要があります。

`-sub|-subsystem` *`subsystem`*

タスク サブシステムを指定します。サブシステムは有効なタスク サブシステムである必要があります。

`-s|-synopsis` *`synopsis`*

作成するタスクの概要を指定します。概要には、復帰改行文字を含まない任意の文字列を使用できます。

`-time|-time_estimate` *`time_estimate`*

タスクの完了に要する予定時間を指定します。*`time_estimate`* には任意の文字列を使用できます。ただし、報告とメトリクスを容易にするために、フォーマットと単位には一貫した規則を使用するようにしてください。

例

- `Entanglement methods` という概要名を持つタスクを作成する。

```
ccm task -create -synopsis "Entanglement methods"
Task 44 created.
```

オブジェクト、タスク、変更依頼とタスクとの関連付けの解除

このサブコマンドにより、2 つのオブジェクト間の関連付けを解除します。指定したタスクと指定のオブジェクトや変更依頼との関連付けを解除したり、指定したタスクと修正対象タスクとの関連付けを解除します。ユーザーがそのタスクが変更できる場合のみ、オブジェクトや修正対象タスクとの関連付けを解除できます。タスクと変更依頼との関連付けを解除する場合、変更依頼とタスクの両方を変更できる必要があります。

```
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

タスクとの関連付けを解除する変更依頼を指定します。*change_request_spec* には、複数の変更依頼を指定できます。詳細については、変更依頼の指定を参照してください。

`-fixes` *task_spec*

修正したタスクを指定します。*task_spec* には、1 つのタスクを指定できます。詳細については、タスクの指定を参照してください。

`-obj|-object` *file_spec*`...`

指定したタスクとの関連付けを解除するファイルまたはディレクトリの名前を指定します。

`-prob|-problem|-change_request`

タスクとの関連付けを解除する変更依頼を指定します。

例

- タスク **35** とオブジェクト バージョン `MAIN.C-3:csrc:1` との関連付けを解除する。

  ```
  ccm task -d 34 -obj MAIN.C-3:csrc:1
  Disassociated object version from task 34:MAIN.C-3:csrc:1
  ```

- タスク **10668** と変更依頼 **6569** との関連付けを解除する。

  ```
  ccm task -d 10668 -change_request 6569
  ```

関連トピック

- query コマンド

タスクの修正

このサブコマンドにより、タスクを作成し、修正するタスクとの間に関係を設定します。この関係を設定しておくと、プロジェクトが 1 つのタスクを使用してもう 1 つを使用しない場合に検出することができます（これをコンフリクトといいます）。コンフリクトの詳細については、conflicts コマンドを参照してください。

既存のタスクによってタスクを修正する場合は、オブジェクト、タスク、変更依頼へのタスク関連付けを参照してください。修正タスクをさらに修正または拡張するには、最初の修正タスクを修正する新しい修正タスクを作成できます。

以下に、修正関係を設定するための要件を示します。

- 相互に関連するタスクは異なるデータベースからのものでよい。
- 修正するタスクの状態は *completed*（完了）または *excluded*（除外）のどちらかである必要がある。
- 修正タスクは関係を設定するユーザーによって修正可能である必要がある。
- 1 つのタスクは 1 つのタスクのみ修正できる。

```
ccm task -fix [-exclude] -s|-synopsis synopsis
        [-prob|-problem|-change_request change_request_spec]
        ([-def|-default|-current] | [-register])
        [-desc|-description description]
        [-desc_edit|-descriptionedit|-description_edit]
        [-desc_file|-descriptionfile|-description_file file_path]
        [-p|-priority priority] [-plat|-platform platform]
        [-r|-resolver resolver] [-rel|-release release_spec]
        [-sub|-subsystem subsystem] [-time|-time_estimate time_estimate]
        [-date|-date_estimate date_estimate] [-q|-quiet] task_spec...
```

-date|-date estimate date estimate

-def|-default|-current

作成する修正タスクがこの この CLI セッションのカレント タスクとして設定されるよう指定します。

-desc|-description description

-desc_edit|-description_edit

-desc_file|-description_file file_path

`-exclude`

修正するタスクを *excluded*（除外）状態に遷移させるよう指定します。このオプションを使用して、以降のビルドに自動的に含まれないように修正タスクを除外します。

`-plat|-platform platform`

`-p|-priority priority`

`-prob|-problem|-change_request` *`change_request_spec`*

修正タスクを、指定した変更依頼に関連付けるよう指定します。変更依頼は自分が変更可能であり、タスクの関連付けが可能な状態にある必要があります。変更依頼を指定せず、修正元のタスクが *assigned* 状態の変更依頼に関連付けられている場合、新しいタスクも同じ変更依頼に関連付けられます。

`change_request_spec` には、1 つの変更依頼を指定できます。詳細については、変更依頼の指定を参照してください。

`-quiet`

確認メッセージに修正タスクのタスク ID だけを含むよう指定します。

`-register`

`-rel|-release release_spec`

`-r|-resolver` *`resolver`*

タスクの解決を担当するユーザーを指定します。担当者を指定しない場合は、修正対象タスクの担当者が適用されます。*`resolver`* は有効なタスク担当者である必要があります。

`-sub|-subsystem subsystem`

`-s|-synopsis synopsis`

`task_spec``...`

修正するタスクを指定します。*`task_spec`* には、複数のタスクを指定できます。詳細については、タスクの指定を参照してください。

-time|-time estimate time estimate

例

- 修正タスク（**19**）と修正されるタスク （**4**）との間の関係を作成する。

  ```
  ccm task -relate 19 -fixes 4
  ```

- 修正タスク（**25**）とそれが修正したタスク（**12**）との間の関係を解除する。

  ```
  ccm task -unrelate 25 -fixes 12
  ```

- タスク **4** の修正タスクを作成する。

  ```
  ccm task -fix -s "Create a fix task for task 4" 4
  Task 17 created to fix Task 4.
  ```

- 修正タスクを作成し、修正対象タスクを *excluded* 状態に遷移させる。

  ```
  ccm task -fix -exclude -s "exclude task 1 and create new for release
  1.0" 1
  Task 16 created to fix Task 1.
  ```

関連トピック

- query コマンド

タスクの変更

```
ccm task -mod|-modify [-s|-synopsis synopsis]
         [-desc|-description description]
         [-desc_edit|-descriptionedit|-description_edit]
         [-desc_file|-descriptionfile|-description_file file_path]
         [-desc_replace|-descriptionreplace|-description_replace]
         [-p|-priority priority] [-plat|-platform platform]
         [-r|-resolver resolver] [-rel|-release release_spec]
         [-sub|-subsystem subsystem] [-time|-time_estimate time_estimate]
         [-date|-date_estimate date_estimate] task_spec...
```

-date|-date estimate date estimate

-desc|-description description

-desc_edit|-description_edit

-desc_file|-description file file path

-desc_replace|-descriptionreplace|-description_replace

既存のタスク説明を指定した説明で置き換えるよう指定します。デフォルトでは、説明は既存のタスク説明に追加されます。復帰改行文字を含めることはできません。

-plat|-platform platform

-p|-priority priority

-rel|-release release spec

-r|-resolver *resolver*

タスクの解決を担当するユーザーを指定します。担当者を指定しない場合、変更されるタスクの担当者が適用されます。*resolver* は有効なタスク担当者である必要があります。

-sub|-subsystem subsystem

-s|-synopsis synopsis

task_spec...

変更するタスクを指定します。*task_spec* には、複数のタスクを指定できます。詳細については、タスクの指定を参照してください。

-time|-time_estimate time_estimate

例

- タスク 68 のリリース値を 4.1 に変更する。

```
ccm task -modify -release 4.1 68
```

関連トピック

- query コマンド

タスクのクエリ

このサブコマンドにより、リリース内のタスク、リリースにないタスク、あるいは指定したクエリ条件またはクエリ式を満足するタスクをクエリします。クエリによって見つかったタスクを使用して、クエリ選択セットが設定されます。

```
ccm task -qu|-query -in_rel|-in_release [-f|-format format]
         [-nf|-noformat] ([-ch|-column_header] | [-nch|-nocolumn_header])
         [-sep|-separator separator] ([-sby|-sortby sortspec] |
         [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat]
         [-u|-unnumbered] old_project_spec project_spec
ccm task -qu|-query -in_rel|-in_release [-f|-format format]
         [-nf|-noformat] ([-ch|-column_header] | [-nch|-nocolumn_header])
         [-sep|-separator separator] ([-sby|-sortby sortspec] |
         [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat]
         [-u|-unnumbered] project_spec
ccm task -qu|-query -not_in_rel|-not_in_release [-f|-format format]
         [-nf|-noformat] ([-ch|-column_header] | [-nch|-nocolumn_header])
         [-sep|-separator separator] ([-sby|-sortby sortspec] |
         [-ns|-nosort|-no_sort]) [-gby|-groupby groupformat]
         [-u|-unnumbered] project_spec
ccm task -qu|-query [-cus|-custom custom_query]
         [(-db|-dbid|-database_id database_spec)...]
         [(-plat|-platform platform)...][(-purpose purpose)...]
         [(-rel|-release release_spec)...]
         [(-sub|-subsystem subsystem)...]
         [-ts|-scope|-task_scope (user_defined |
         (all_my_assigned|all_owners_assigned) |
         (all_my_assigned_or_completed|all_owners_assigned_or_completed) |
         (all_my_completed|all_owners_completed) |
         (all_my_tasks|all_owners_tasks) | all_completed | all_tasks)]
         (ct_projs|ct_projects|component_task_projects) |
         (ct_prods|ct_products|component_task_products) |
         (ct_projs_prods|ct_projects_products |
         component_task_projects_products))]
         [-f|-format format] [-nf|-noformat] ([-ch|-column_header] |
         [-nch|-nocolumn_header]) [-sep|-separator separator]
         ([-sby|-sortby sortspec] | [-ns|-nosort|-no_sort])
         [-gby|-groupby groupformat] [-u|-unnumbered]
```

`-ch|-column_header`

出力フォーマットでカラム ヘッダーを使用するよう指定します。詳細については、-ch|-column_headers を参照してください。

`-cus|-custom custom_query`

指定したカスタム クエリ式をクエリに含めるよう指定します。

`-db|-dbid|-database_id` *`database_spec`*

`-task_scope` オプションと一緒に使用することで、タスク範囲から生成されるクエリを修飾するデータベース ID を指定します。詳細については、データベースの指定を参照してください。

`-f|-format` *`format`*

コマンドの出力フォーマットを指定します。詳細については、-f|-format を参照してください。

`-gby|-groupby` *`groupformat`*

コマンドの出力をグループ化する方法を指定します。詳細については、-gby|-groupby を参照してください。

`-in_rel|-in_release` *`old_project_spec project_spec`*

`project_spec` をルートとするプロジェクト階層内にある、すべてのタスクを表示します。これは、そのプロジェクト階層内のすべてのオブジェクトとその祖先に関するすべてのタスクを入手し、次に *`old_project_spec`* をルートとするプロジェクト階層のすべてのオブジェクトとその祖先に関するすべてのタスクを入手して、前者から後者を差し引いて決定されます。

`old_project_spec` を指定しない場合、タスクは差し引かれません。ベースラインリリースが存在しない製品の最初のリリースの場合以外では、*`old_project_spec`* を指定すべきです。

`-nch|-nocolumn_header`

出力フォーマットでカラム ヘッダーを使用しないよう指定します。詳細については、-ch|-column_headers を参照してください。

`-nf|-noformat`

カラム配置を使用しないよう指定します。詳細については、-nf|-noformat を参照してください。

`-ns|-no_sort`

コマンドの出力をソートしないよう指定します。詳細については、-ns|-nosort を参照してください。

`-not_in_rel|-not_in_release` *`project_spec`*

`project_spec` をルートとするプロジェクト階層内にない、すべてのタスクを表示します。

`-plat|-platform` *`platform`*

プラットフォームを指定します。プラットフォームは有効なプラットフォームである必要があります。

`-purpose` *`purpose`*

指定した目的のクエリを含むタスク クエリによってフォルダを作成するよう指定します。目的の詳細については、`project_purpose` コマンドの説明と用途を参照してください。

通常、このオプションは、`component_task_projects`、`component_task_products`、または `component_task_projects_products` のいずれかの範囲で指定されるコンポーネント タスクのクエリに適用されます。

`-rel|-release` *`release_spec`*

リリース値を指定します。*`release_spec`* には、1 つのリリース値を指定できます。詳細については、リリースの指定を参照してください。

`-sep|-separator` *`separator`*

異なる区切り文字を指定します。詳細については、-sep|-separator を参照してください。

`-sby|-sortby` *`sortspec`*

コマンドの出力をソートする方法を指定します。詳細については、-sby|-sortby を参照してください。

`-sub|-subsystem` *`subsystem`*

タスク サブシステムを指定します。サブシステムは有効なタスク サブシステムである必要があります。

`-ts|-scope|-task_scope`

タスク クエリを使用するよう指定します。タスク クエリには指定した範囲によって決まるクエリ式が含まれます。指定した範囲に関連付けられるクエリ式は、`-database_id` オプションによっても異なります。以下の範囲を使用できます。

- *`user_defined`*

この範囲は default_task_query オプションによって定義されます。-database_id を指定すると、クエリには、指定したデータベース内で修正可能または完了したタスクのクエリ式も含まれます。

- all_my_assigned|all_owners_assigned

  この範囲では、自分に割り当てられているすべてのタスクをクエリします。-database_id を指定すると、指定したデータベースで修正可能な、自分に割り当てられているすべてのタスクをクエリします。

- all_my_assigned_or_completed|all_owners_assigned_or_completed

  この範囲では、自分に割り当てられているすべてのタスク、または自分が完了したすべてのタスクをクエリします。-database_id を指定すると、指定したデータベース内で自分に割り当てられている修正可能なタスク、または指定したデータベース内で自分が完了したすべてのタスクをクエリします。

- all_my_completed|all_owners_completed

  この範囲では、自分が完了したすべてのタスクをクエリします。-database_id を指定すると、指定したデータベース内で自分が完了したすべてのタスクをクエリします。

- all_my_tasks|all_owners_tasks

  この範囲では、自分がタスク担当者であるすべてのタスクをクエリします。-database_id を指定すると、自分がタスク担当者であり、指定したデータベース内で修正可能または自分が完了したすべてのタスクをクエリします。

- all_completed

  この範囲では、完了したすべてのタスクをクエリします。-database_id を指定すると、指定したデータベースで完了したすべてのタスクをクエリします。

- all_tasks

  この範囲では、すべてのタスクをクエリします。-database_id を指定すると、指定したデータベース内で修正可能または完了したすべてのタスクをクエリします。

- component_task_projects|component_task_products|component_task_projects_products

  この範囲では、プロジェクト、製品、またはプロジェクトと製品のコンポーネント タスクをクエリします。-database_id を指定すると、指定したデータベースで作成されたすべてのコンポーネント タスクをクエリします。-purpose を指定すると、指定した目的を持つコンポーネント タスクをクエリします。

-u|-unnumbered

コマンドの出力の自動番号付けを抑止します（すなわち、出力に番号付けがされません）。詳細については、-u|-unnumbered を参照してください。

例

- リリース値が 3.0 に設定されているタスクをクエリする。タスク概要だけを表示するよう、出力をフォーマットする。

```
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
```

関連トピック

- query コマンド

カレント タスクの設定または解除

このサブコマンドにより、カレント タスクを設定または解除します。

デフォルトで、変更をタスクに関連付ける操作を実行する際にタスクを指定しないと、Rational Synergy がこのカレントタスクを使用します。

```
ccm task -def|-default|-current [(task_spec|none)]
```

指定したタスクを、この CLI セッションのカレント タスクとして設定します。*task_spec* には、自分に割り当てられている 1 つのタスクを指定できます。詳細については、タスクの指定を参照してください。

none キーワードを指定すると、カレント タスクが解除されます。

引数を指定しない場合、このコマンドによってカレント タスクが表示されますが、設定はされません。

例

- カレント タスクを表示する。

  ```
  ccm task -current
  The current task is not set.
  ```

- カレント タスクを設定する。

  ```
  ccm task -current 26
  The current task is set to:
  26:Close box no longer active
  ```

- カレント タスクを解除する。

  ```
  ccm task -current none
  The current task has been cleared.
  ```

関連トピック

- query コマンド

タスク プロパティの表示

このサブコマンドにより、指定したタスクのプロパティを表示します。

```
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...

プロパティを表示したいタスクを指定します。詳細については、タスクの指定を参照してください。

関連トピック

- query コマンド

タスクに関連付けられているオブジェクト、変更依頼、タスクの表示

このサブコマンドにより、以下の項目を表示します。

- 関連付けられているオブジェクト
- 指定したタスクの変更依頼
- 指定したタスクによって修正されたタスク
- 指定したタスクを修正するタスク

クエリ選択により、関連付けられているオブジェクトが表示されます。

```
ccm task -s|-sh|-show ((obj|objs|objects) |
        (cr|change_request|change_requests|prob|problem|problems) |
        (fix|fixes) | fixed_by)
        [-f|-format format] [-nf|-noformat [-ch|-column_header] |
        [-nch|-nocolumn_header]) [-sep|-separator separator]
        ([-sby|-sortby sortspec] | [-ns|-nosort|-no_sort])
        [-gby|-groupby groupformat] [-u|-unnumbered] task_spec...
```

-ch|-column header

-f|-format format

-gby|-groupby groupformat

-nch|-nocolumn header

-nf|-noformat

-ns|-no_sort

-sep|-separator separator

-sby|-sortby sortspec

task_spec...

プロパティを表示したいタスクを指定します。詳細については、タスクの指定を参照してください。

`-u|-unnumbered`

コマンドの出力の自動番号付けを抑止します（すなわち、出力に番号付けがされません）。詳細については、-u|-unnumbered を参照してください。

例

- タスク **68** に関連付けられている変更依頼を表示する。

```
ccm task -show change_request 68
```

```
1) Change request 5
2) Change request 6
```

- タスク **4** と **5** に関連付けられているオブジェクトを表示する。

```
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
```

関連トピック

- query コマンド

タスク情報の表示

このサブコマンドにより、指定したタスクの概要、説明、状態、担当者などの情報を表示します。

```
ccm task -s|-sh|-show (i|info|information) -f|-format format
         [-nf|-noformat] ([-ch|-column_header] | [-nch|-nocolumn_header])
         [-sep|-separator separator] task_spec...
ccm task -s|-sh|-show (i|info|information) [-v|-verbose] task_spec...
```

-ch|-column_header

-f|-format format

-nch|-nocolumn_header

-nf|-noformat

-sep|-separator separator

task_spec...

プロパティを表示したいタスクを指定します。詳細については、タスクの指定を参照してください。

-v|-verbose

デフォルトの詳細フォーマットを使用してタスクの情報を表示するよう指定します。

例

- リリース **1** のタスク情報を表示する。

  ```
  ccm task -s release 1
  ```

  ```
  Task 1:a/1.0
  ```

- タスク 30 - 33 の情報をフォーマットして表示する。

  ```
  ccm task -show info 30-34 -format "%priority %30-33 %task_synopsis"
  -ns
  ```

  ```
  1) high 33  Date field not validated on Inventory Form
  2) high 41  Wrong window receives message
  3) high 22 Saving a file takes forever
  4) low 39  Button icons are rather obscure
  5) low 4  OK button not default
  ```

- タスク **14** に関連付けられているオブジェクトの情報を表示する。

  ```
  ccm task -sh obj 14
  ```

  ```
  Task 14:
  1) a.txt-1.1:ascii:1 integrate jane
  2) b.txt-1.1:ascii:1 integrate  jane
  ```

関連トピック

- query コマンド

別の状態へのタスクの遷移

```
ccm task -st|-state task_state [-r|-resolver resolver]
        [-desc|-description description]
        [-desc_edit|-descriptionedit|-description_edit]
        [-desc_file|-descriptionfile|-description_file file_path]
        task_spec...
```

-desc|-description description

-desc edit|-description edit

-desc file|-description file file path

task_state

タスクの遷移後の状態を指定します。

-r|-resolver *resolver*

指定したタスクの担当者を設定するよう指定します。このオプションは、タスクを *task_assigned* 状態に遷移させるときのみ使用できます。*resolver* は有効なタスク担当者である必要があります。

task_spec...

遷移させるタスクを指定します。詳細については、タスクの指定を参照してください。

例

- タスクの状態を *completed*（完了）から *excluded*（除外）へ遷移させる。

  ```
  ccm task -state excluded 94
  ```

- タスクの状態を *excluded*（除外）から *completed*（完了）へ遷移させる。

  ```
  ccm task -state completed 94
  ```

関連トピック

- query コマンド

説明と用途

task コマンドにより、以下のタスクベースの操作を行うことができます。

- タスクを割り当てる。
- オブジェクト、他のタスク、変更依頼にタスクを関連付ける。
- タスクを完了（チェックイン）する。
- タスクをコピーする。
- タスクを作成する。
- オブジェクト、他のタスク、変更依頼とのタスクの関係を解除する。
- タスクを修正する。
- タスクを変更する。
- タスクのクエリを行う。
- タスクとタスクまたはタスクとオブジェクトの間の関係を作成する、または解除する。
- カレント（デフォルト）タスクを設定または解除する。
- タスク情報を表示する。
- タスクを別の状態に遷移させる。

unalias コマンド

詳細については、説明と用途を参照してください。unalias コマンドは、別名の削除サブコマンドをサポートします。

別名の削除

`ccm unalias` *`alias_name`*

`alias_name`

削除したい別名を指定します。

例

- `getf` コマンドの別名を削除する。

  ```
  ccm unalias getf
  ```

関連トピック

- alias コマンド

説明と用途

unalias コマンドにより、定義済みの別名を削除します。

unalias コマンドは、現在のセッションで定義された別名のみを削除します。

undo_update コマンド

詳細については、説明と用途を参照してください。`undo_update` コマンドは、以下のサブコマンドをサポートします。

- ディレクトリの更新の取り消し
- プロジェクトの更新の取り消し
- プロジェクト グルーピングの更新の取り消し

ディレクトリの更新の取り消し

```
ccm unupd|undo_update|unreconf|undo_reconfigure [-r|-recurse]
          [-v|-verbose] dir_spec...
```

dir_spec...

更新を元に戻すディレクトリを指定します。*dir_spec* には、複数のディレクトリ オブジェクトを指定できます。*dir_spec* はファイルの指定で説明している形式をとります。

-r|-recurse

サブプロジェクトを含めるよう指定します。

-v|-verbose

詳細な更新取り消しメッセージを表示します。

関連トピック

- プロジェクトの更新の取り消し
- プロジェクト グルーピングの更新の取り消し

プロジェクトの更新の取り消し

```
ccm unupd|undo_update|unreconf|undo_reconfigure -p|-project
          [-r|-recurse] [-v|-verbose] project_spec...
```

project_spec

更新を元に戻すプロジェクトを指定します。*project_spec* には、複数のプロジェクトを指定できます。*project_spec* はプロジェクトの指定で説明している形式をとります。

`-r|-recurse`

サブプロジェクトも含めるよう指定します。

`-v|-verbose`

詳細な更新取り消しメッセージを表示します。

例

- `proj1-1` プロジェクトの更新を元に戻す。

  ```
  ccm unupd -p proj1-1
  ```

- サブプロジェクトも含まれている `toolkit-jane` という名前のプロジェクトの更新を元に戻す。

  ```
  ccm undo_update -recurse -project toolkit-jane
  ```

関連トピック

- ディレクトリの更新の取り消し
- プロジェクト グルーピングの更新の取り消し

プロジェクト グルーピングの更新の取り消し

```
ccm unupd|undo_update|unreconf|undo_reconfigure
          -pg|-project_grouping [-r|-recurse] [-v|-verbose]
          project_grouping_spec...
```

project_grouping_spec

更新を元に戻すプロジェクト グルーピングを指定します。*project_grouping_spec* には、複数のプロジェクト グルーピングを指定できます。

プロジェクト グルーピングのベースラインとタスクは変更されません。詳細については、プロジェクト グルーピングの指定を参照してください。

`-r|-recurse`

サブプロジェクトも含めるよう指定します。

`-v|-verbose`

詳細な更新取り消しメッセージを表示します。

関連トピック

- ディレクトリの更新の取り消し
- プロジェクトの更新の取り消し

説明と用途

undo_update コマンドにより、指定されたディレクトリまたはプロジェクト オブジェクトについての更新処理を元に戻します（取り消します）。

処理パフォーマンス向上のため、undo_update コマンドはデフォルトではパラレル オブジェクト バージョンを検出してもパラレル バージョン通知を行いません。パラレル バージョン通知を有効にするには、初期設定ファイル内の reconfigure_parallel_check ユーザー オプションを TRUE に設定します。

取り消し処理の中の 1 つの操作でも失敗すると、更新の取り消し処理は停止します。たとえば、オブジェクトの現在のバージョンにワークエリア コンフリクトがあると、処理が停止し、新しいバージョンの自動作成は行われません。これは、ユーザーのワークエリア内のデータを保護するためです。

undo_update コマンドの処理を停止するデフォルトの設定は、初期設定ファイルの修正によって変更できます。この方法は、取り消し処理中に個別の処理でエラーが発生した場合でも、undo_update 操作を続けたい場合に有用です。更新処理を続行するように設定するには、初期設定ファイルで reconf_stop_on_fail オプションを False に設定します。

直前の更新取り消しを取り消すために、undo_update コマンドを使用できます。ただし、更新取り消しの取り消しを複数回実行しても、取り消されるのは直前の取消し操作のみです。

unrelate コマンド

詳細については、説明と用途を参照してください。unrelate コマンドは、2 つのオブジェクトの間の関係の削除サブコマンドをサポートします。

2 つのオブジェクトの間の関係の削除

```
ccm unrelate -n|-name relationship_name -f|-from from_object_spec
             -t|-to to_object_spec
```

`-f|-from object_spec`

関係元のオブジェクトを指定します。*object_spec* には、任意の種類の 1 つのオブジェクトを指定できます。

`-n|-name relationship_name`

削除する関係の名前を指定します。

`-t|-to object_spec`

関係先のオブジェクトを指定します。*object_spec* には、任意の種類の 1 つのオブジェクトを指定できます。

例

- `clear.c-2` から `clear.c-1` への **successor** 関係を削除する。

  ```
  ccm unrelate -n successor -f clear.c-1:csrc:1 -t clear.c-2:csrc:1
  ```

関連トピック

- オブジェクト間の関係の作成
- オブジェクト間の関係の表示

説明と用途

`unrelate` コマンドにより、*`file_spec1`* と *`file_spec2`* との間の関係 *`rel_name`* を削除します。

Rational Synergy には、多くの関係があらかじめ定義されています。定義済みの関係については、Synergy CLI ヘルプ、トラディショナル モードの「関係」の表を参照してください。さらに、`relate` コマンドを使用して、新しい関係を定義することもできます。

2 つのオブジェクトの間の関係を削除するには、上で説明した 3 つのオブジェクト仕様をすべて含める必要があります。

このコマンドでは、クエリ選択セットは更新されません。

unset コマンド

詳細については、説明と用途を参照してください。unset コマンドは、オプションの設定解除サブコマンドをサポートします。

オプションの設定解除

```
ccm unset option
```

option

値の設定を解除するオプションの名前を指定します。

例

- `proj_log` オプションの設定を解除する。

  ```
  ccm unset proj_log
  ```

関連トピック

- オプションの設定
- オプションの表示
- オプションの表示

説明と用途

unset コマンドにより、オプションの値の設定を解除します。これにより、オプションの設定は出荷時のデフォルト値に戻ります。オプションの中には、あらかじめ定義されている読み出し専用のものや、設定解除できないものがあります。オプションの詳細については、デフォルト オプションを参照してください。

unuse コマンド

詳細については、説明と用途を参照してください。unuse コマンドは、以下のサブコマンドをサポートします。

- 指定したディレクトリからのプロジェクトの削除
- 現在のディレクトリからのプロジェクトの削除
- プロジェクトからのオブジェクトの削除

指定したディレクトリからのプロジェクトの削除

このサブコマンドにより、指定したプロジェクトを、指定のディレクトリおよびそのプロジェクトに関連するコンテキストプロジェクトから削除します。コンテキストプロジェクトを示すには、プロジェクト参照形式またはワークエリア参照形式を使用してディレクトリを指定します。コマンドを実行できるのは、コンテキスト プロジェクトが *working*（作業中）状態の場合は、そのプロジェクトの所有者です。コンテキスト プロジェクトが *prep*（準備）状態の場合は、ビルド マネージャです。

```
ccm unuse -p|-project -dir dir_spec [-t|-task task_spec]
          [-d|-delete [-f|-force]] [-r|-replace] project_spec...
```

`-d|-delete`

ディレクトリとそのコンテキスト プロジェクトからオブジェクトを削除し、それをデータベースから削除します。

`-dir` *`dir_spec`*

オブジェクトの使用を解除するディレクトリを指定します。*`dir_spec`*（ファイルの指定を参照）は、1 つのディレクトリ オブジェクトを指定してコンテキスト プロジェクトを与える `file_spec` です。プロジェクト参照形式またはワークエリア参照形式によってこのようなコンテキスト プロジェクトを与えます。

`-force`

このオプションは、`-d|-delete` オプションと一緒にのみ使用できます。このオプションを指定すると、オブジェクトを削除したときに、そのオブジェクトがユーザーが修正可能なすべてのプロジェクトから削除され、さらにデータベースからも削除されます。このオプションを指定しない場合、オブジェクトはそのコンテキスト プロジェクトから使用解除されるだけです。オブジェクトがデータベースから削除されるのは、オブジェクトがどのプロジェクトのメンバーでもない場合です。

`project_spec`

使用解除するプロジェクトを指定します。詳細については、プロジェクトの指定を参照してください。

`-r|-replace`

ディレクトリ内のオブジェクトを直前バージョンと置き換えます。このオプションを指定しても、ディレクトリ内のファイルのリストは変わりません。指定したオブジェクトのバージョンが変わるだけです。

`-t|-task` *`task_spec`*

自動的にチェックアウトされるディレクトリに関連付けられるタスクを指定します。`-r|-replace` を指定していない場合、オブジェクトを含むディレクトリはそのオブジェクトのエントリを削除するように更新されます。ディレクトリが静的状態でない場合、ディレクトリは自動的にチェックアウトされます。`-t|-task` オプションを指定しない場合、デフォルトで現在のタスクが使用されます。詳細については、タスクの指定を参照してください。

例

- 最上位のプロジェクト `ico_jan4-1` から、サブプロジェクト `ico_jan5` と `ico_jan6` を削除する。

  ```
  ccm unuse ico_jan5 ico_jan6
  ```

  ```
  Member ico_jan5-1 removed from project ico_jan4-1
  Member ico_jan6-1 removed from project ico_jan4-1
  ```

- プロジェクト `Project_One-1` の `Dir` 下にあるサブプロジェクト `SubProject_One-1` を使用解除する。

  ```
  ccm unuse -p -dir Project_One¥Dir@Project_One-1 SubProject_One-
  1:project:1
  ```

関連トピック

- use コマンド
- delete コマンド

現在のディレクトリからのプロジェクトの削除

このサブコマンドにより、現在の作業ディレクトリからプロジェクトを削除します。現在の作業ディレクトリは、ユーザーが修正可能なプロジェクトの管理ワークエリア内になければなりません。コマンドを実行できるのは、コンテキスト プロジェクトが *working*（作業中）状態の場合は、そのプロジェクトの所有者です。コンテキスト プロジェクトが *prep*（準備）状態の場合は、ビルド マネージャです。

```
ccm unuse -p|-project [-t|-task task_spec] [-d|-delete [-f|-force]]
          [-r|-replace] project_spec...
```

`-d|-delete`

-d|-delete を参照してください。

`-force`

-force を参照してください。

`project_spec`

project spec を参照してください。

`-r|-replace`

-r|-replace を参照してください。

`-t|-task` *`task_spec`*

-t|-task task_spec を参照してください。

例

- 現在のディレクトリから `sort.c` を切り取る。

  ```
  ccm unuse sort.c-1:csrc:1
  ```

関連トピック

- use コマンド
- delete コマンド

プロジェクトからのオブジェクトの削除

このサブコマンドにより、プロジェクトからオブジェクトを削除します。`-dir` オプションを指定している場合、オブジェクトはディレクトリおよび関連付けられているコンテキスト プロジェクトから削除されます。`-dir` オプションを指定していない場合、オブジェクトは現在の作業ディレクトリから削除されます。現在の作業ディレクトリは、ユーザーが修正可能なプロジェクトの管理ワークエリア内になければなりません。コマンドを実行できるのは、コンテキスト プロジェクトが *working*（作業中）状態の場合は、そのプロジェクトの所有者です。コンテキスト プロジェクトが *prep*（準備）状態の場合は、ビルド マネージャです。

ccm unuse [-dir *dir_spec*] [-t|-task *task_spec*] [-d|-delete [-f|-force]]
[-r|-replace] *file_spec*...

`-d|-delete`

-d|-delete を参照してください。

`-dir` *dir_spec*

-dir dir_spec を参照してください。

file_spec

使用解除するオブジェクトを指定します。

`-force`

-force を参照してください。

`-r|-replace`

ディレクトリ内のオブジェクトを直前バージョンと置き換えます。このオプションを指定しても、ディレクトリ内のファイルのリストは変わりません。指定したオブジェクトのバージョンが変わるだけです。

`-t|-task` *task_spec*

-t|-task task_spec を参照してください。

例

- 現在のプロジェクトから `sort.c` オブジェクトを切り取る。

```
ccm unuse sort.c
Member sort.c-1 removed from project ico-1
```

- プロジェクト `Project_One-1:` の `Dir` フォルダの下にある `ico_jan5-1` を切り取る。

  ```
  ccm unuse -dir Project_One¥Dir@Project_One-1 ico_jan5-1:ascii:1 -task
  10
  ```

関連トピック

- use コマンド
- delete コマンド

説明と用途

現在のプロジェクトまたは現在のディレクトリから、既存のファイル、ディレクトリ、ルート ディレクトリ、またはプロジェクトを取り除きます。メンバーを削除するためには、ディレクトリをチェックアウトする必要があります。ただし、修正不可ディレクトリからオブジェクトを削除しようとすると、Rational Synergy がディレクトリを自動的にチェックアウトします（`-r` オプションを指定しない場合）。ディレクトリ内の変更を他のユーザーも利用できるようにするためには、ディレクトリを**チェックイン**する必要があります。unuse は 切り取り（cut）と呼ばれるようになりました。

`-d` オプションと `-r` オプションも指定した場合にのみ、ルート ディレクトリもこのコマンドの対象とできます。別バージョンのルート ディレクトリを使用したい場合は、use コマンドを使用します。ルート ディレクトリは置き換えずに切り取ることはできません。なぜなら、プロジェクトには常にルート ディレクトリが必要だからです。

> **注記**：修正不可ディレクトリからオブジェクトを切り取ると、新しいディレクトリ バージョンが自動的にチェックアウトされます。ただし、オブジェクトを別バージョンで置き換える場合は別です。
>
> 共有プロジェクト内にいて、現在のディレクトリが修正不可の場合、そのディレクトリはチェックアウトされて、カレント（または指定した）タスクと関連付けられ、*integrate*（統合）状態にチェックインされます。初期設定ファイル内の `shared_project_directory_checkin` を `FALSE` に設定して、自動チェックイン機能を無効にできます（shared_project_directory_checkin を参照してください）。

プロジェクトを削除したい場合は、delete コマンド（`ccm delete -p` *`project_name-version`*）を参照してください。

プロジェクト参照形式を使用している場合は、このコマンドを使用するために、ワークエリア内にいる必要はありません。

Windows：*`relative_path¥object_name@project_name-project_version`*

UNIX：*`relative_path/object_name@project_name-project_version`*

以下に、プロジェクト参照形式の例と、ルート ディレクトリ `ico/hi_world.c@final-1:` を削除するためにそれを使用する方法を示します。

```
ccm unuse -d -r final@final-1
```

削除された *`object_version`* と置き換え後のバージョンを示すメッセージが表示されます。

update コマンド

詳細については、説明と用途を参照してください。`update_members` コマンドは、以下のサブコマンドをサポートします。

- ディレクトリのメンバーの更新
- プロジェクト グルーピングのメンバーの更新
- プロジェクト メンバーの更新

ディレクトリのメンバーの更新

```
ccm u|update|update_members|reconf|reconfigure
            ([-ks|-keep_subprojects] | [-rs|-replace_subprojects])
            [-r|-recurse] [-v|-verbose] dir_spec...
```

dir_spec

更新するディレクトリを指定します。*dir_spec* には、複数のディレクトリ オブジェクトを指定できます。*dir_spec* はファイルの指定で説明している形式をとります。

-ks|-keep_subprojects

現在の場所にサブプロジェクトを維持するよう指定します。
-keep_subprojects または -replace_subprojects を指定しない場合、デフォルトは replace_subproj の設定によって異なります。

-rs|-replace_subprojects

サブプロジェクトを新しいサブプロジェクトに置き換えるよう指定します。サブプロジェクトが置き換えられるのは、選択ルールによってサブプロジェクトの他のバージョンが選択された場合だけです。
-keep_subprojects または -replace_subprojects を指定しない場合、デフォルトは replace_subproj の設定によって異なります。

-r|-recurse

サブプロジェクトも含めるよう指定します。

-v|-verbose

詳細な更新取り消しメッセージを表示します。

関連トピック

- undo_update コマンド
- project_grouping コマンド

プロジェクト グルーピングのメンバーの更新

```
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...

更新するプロジェクト グルーピングを指定します。*project_grouping_spec* には、複数のプロジェクト グルーピングを指定できます。*project_grouping_spec* はプロジェクト グルーピングの指定で説明している形式をとります。

`-ks|-keep_subprojects`

現在の場所にサブプロジェクトを維持するよう指定します。
`-keep_subprojects` または `-replace_subprojects` を指定しない場合、デフォルトは replace_subproj の設定によって異なります。

`-rs|-replace_subprojects`

サブプロジェクトを新しいサブプロジェクトに置き換えるよう指定します。サブプロジェクトが置き換えられるのは、選択ルールによってサブプロジェクトの他のバージョンが選択された場合だけです。
`-keep_subprojects` または `-replace_subprojects` を指定しない場合、デフォルトは replace_subproj の設定によって異なります。

`-r|-recurse`

サブプロジェクトも含めるよう指定します。

`-v|-verbose`

詳細な更新取り消しメッセージを表示します。

関連トピック

- undo_update コマンド
- project_grouping コマンド

プロジェクト メンバーの更新

```
ccm u|update|update_members|reconf|reconfigure -p|-project
            ([-ks|-keep_subprojects] | [-rs|-replace_subprojects])
            [-r|-recurse] [-v|-verbose] project_spec...
```

project_spec...

更新するプロジェクトを指定します。*project_spec* には、複数のプロジェクトを指定できます。*project_spec* はプロジェクトの指定で説明している形式をとります。

`-ks|-keep_subprojects`

現在の場所にサブプロジェクトを維持するよう指定します。
`-keep_subprojects` または `-replace_subprojects` を指定しない場合、デフォルトは replace_subproj の設定によって異なります。

`-rs|-replace_subprojects`

サブプロジェクトを新しいサブプロジェクトに置き換えるよう指定します。サブプロジェクトが置き換えられるのは、選択ルールによってサブプロジェクトの他のバージョンが選択された場合だけです。
`-keep_subprojects` または `-replace_subprojects` を指定しない場合、デフォルトは replace_subproj の設定によって異なります。

`-r|-recurse`

サブプロジェクトも含めるよう指定します。

`-v|-verbose`

詳細な更新取り消しメッセージを表示します。

例

- `proj1-1` という名前のプロジェクトを更新して、そのサブプロジェクトを置き換える。

  ```
  ccm update -rs -p proj1-1
  ```

- `All Fox/2.01 Integration Testing Projects` という名前のグルーピング内のすべてのプロジェクトを更新する。

  ```
  ccm update -pg "All Fox/2.01 Integration Testing Projects"
  ```

関連トピック

- undo_update コマンド

- project_grouping コマンド

説明と用途

`update` コマンドにより、指定されたディレクトリ、プロジェクト オブジェクト、またはプロジェクト グルーピングを更新します。このコマンドは、プロジェクト グルーピングのベースラインおよびタスクを使用して適切な候補と選択ルールを検索し、適切な場合には、メンバーの新しいバージョンを選択します。また、更新する対象として、プロジェクト グルーピングを指定することもできます。なお、Rational Synergy では、メンバーの更新は、「更新」と呼ばれるようになりました。

更新処理の中の 1 つの操作でも失敗すると、更新処理は停止します。たとえば、オブジェクトの現在のバージョンにワークエリア コンフリクトがあると、処理が停止し、新しいバージョンの自動作成は行われません。これは、ユーザーのワークエリア内のデータを保護するためです。

更新処理を停止するデフォルトの設定は、初期設定ファイルの修正によって変更できます。この方法は、更新処理中に個別の処理でエラーが発生した場合でも、更新操作を続けたい場合に有用です。更新処理を続行するように設定するには、初期設定ファイルで `reconf_stop_on_fail` オプションを `False` に設定します。

プロジェクト グルーピングを使用して、プロジェクト グルーピング内の一連のプロジェクトが同時には更新されない、マルチフェーズ ビルドを行うことができます。プロジェクト グルーピングを使用した場合、すべてのプロジェクトが同じベースラインとタスクを使用して更新されます。開発者またはビルド マネージャは、同じプロジェクト グルーピング内の追加のプロジェクトを、その同じベースラインとタスクを使って更新できます。ベースラインとタスクをリフレッシュする必要はありません。この仕組みの実現のためには、ベースラインとタスクを算出して保存し、以降のプロジェクトの更新でこの保存されたベースラインとタスクが使われるように指定する必要があります。

プロジェクトによっては、その更新プロパティを変更する必要が生じる可能性があります。更新プロパティは、タスクと 1 つのベースラインを使用してプロジェクトを更新するか、またはオブジェクト状態を使用してプロジェクトを更新するかを決定します。さらに、更新プロパティによって、オブジェクトの選択を制御するパラメータの設定を行います。以下に、変更するプロパティとその変更のために使用するコマンドを例示します。

- `ccm attr` コマンドを使用して、プロジェクトのリリース値または目的を変更する。
- `ccm project_grouping` コマンドを使用して、プロセス ルールベースのプロジェクトにタスクを追加または削除する。
- `ccm project_grouping` コマンドを使用して、カスタム開発目的のプロセス ルールベースのプロジェクトのベースラインを設定する。
- `ccm update_properties` コマンドを使用して、手動プロジェクト用タスクを追加または削除してベースラインを設定する。

Rational Synergy GUI を使用して、更新プロパティを変更することもできます。

パフォーマンス向上のため、update コマンドは、デフォルトではパラレル オブジェクト バージョンを検出してもパラレル バージョン通知を行いません。パラレル バージョン通知機能を有効にするには、reconfigure_parallel_check ユーザー オプションを設定します。

use コマンド

詳細については、説明と用途を参照してください。use コマンドは、以下のサブコマンドをサポートします。

- 現在のディレクトリへのプロジェクトの追加
- 別バージョンの使用または指定したディレクトリへのオブジェクトの追加
- 別バージョンの使用または現在のディレクトリへのオブジェクトの追加

現在のディレクトリへのプロジェクトの追加

このサブコマンドによって、1 つまたは複数の既存のプロジェクトを、現在の作業ディレクトリに追加します。現在の作業ディレクトリは、ユーザーが修正可能なプロジェクトの管理ワークエリア内になければなりません。このサブコマンドを使用して、プロジェクトをサブプロジェクトとして追加します。共有プロジェクトでは、特別な処理が行われる場合もあります（shared_project_directory_checkin を参照してください）。コマンドを実行できるのは、コンテキスト プロジェクトが *working*（作業中）状態の場合は、そのプロジェクトの所有者です。コンテキスト プロジェクトが *prep*（準備）状態の場合は、ビルド マネージャです。

サブコンポーネントの別バージョンを使用する場合は、以下の点に注意が必要です。サブコンポーネントが親プロジェクトとは異なるコンポーネントを含む静的サブプロジェクトである場合、そのサブプロジェクトの別バージョンはカレント タスクに関連付けられます。

```
ccm use -p|-project [-t|-task task_spec] project_spec...
```

`project_spec`

使用中のプロジェクトを指定します。詳細については、プロジェクトの指定を参照してください。

`-t|-task -task_spec`

新しいメンバーを追加するためにチェックアウトされたディレクトリと関連付けられるタスクを指定します。これを省略した場合、カレント タスクが使用されます。オブジェクトをディレクトリに追加するとき、ディレクトリが *integrate*（統合）などの静的状態にある場合、自動的にチェックアウトされます。ディレクトリが書き込み可能な状態である場合、既存のディレクトリ バージョンは新しいメンバーで更新されます。詳細については、タスクの指定を参照してください。

例

- `SubPrj-one:project:1` プロジェクトを現在のディレクトリに追加する。

  ```
  ccm use -p -task 31 SubPrj-one:project:1
  ```

関連トピック

- delete コマンド
- unuse コマンド

別バージョンの使用または指定したディレクトリへのオブジェクトの追加

このサブコマンドにより、オブジェクトの別バージョンを使用するか、あるいは指定したディレクトリの下で新しいプロジェクト メンバーとして既存のオブジェクトを追加します。オブジェクトが追加されたディレクトリは、プロジェクト参照形式またはワークエリア参照形式のようなコンテキスト プロジェクトを提供する形式で指定する必要があります。

オブジェクトのディレクトリ エントリが既存の場合、このコマンドによって対応するディレクトリ エントリの下にある指定されたオブジェクトが使用されます。ディレクトリ エントリがオブジェクトに存在しない場合、ディレクトリは指定したタスクに対して自動的にチェックアウトされ、新しいディレクトリ エントリがオブジェクトに作成され、指定したオブジェクトが、指定したディレクトリと関連付けられたコンテキスト プロジェクトで使用されます。共有プロジェクトでは、特別な処理が行われる場合もあります (shared_project_directory_checkin を参照してください)。コマンドを実行できるのは、コンテキスト プロジェクトが *working*（作業中）状態の場合は、そのプロジェクトの所有者です。コンテキスト プロジェクトが *prep*（準備）状態の場合は、ビルド マネージャです。

サブコンポーネントの別バージョンを使用している場合、以下が適用されます。サブコンポーネントが親プロジェクトとは異なるコンポーネントを含む静的製品である場合、その製品の別バージョンが現在のタスクに関連付けられます。

```
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`

オブジェクトの別バージョンまたは既存のオブジェクトを追加するディレクトリを指定します。*dir_spec* には 1 つのディレクトリ オブジェクトを設定でき、コンテキスト プロジェクトを提供する、*file_spec*（ファイルの指定を参照）のようなものです。プロジェクト参照形式またはワークエリア参照形式によって、コンテキスト プロジェクトなどが提供されます。

file_spec

使用中のオブジェクト バージョンを指定します。詳細については、ファイルの指定を参照してください。

project_spec

project_spec を参照してください。

`-t|-task -task_spec`

-t|-task -task_spec を参照してください。

例

- SubPrj-2 プロジェクトの別バージョンを使用する。

  ```
  ccm use -p SubPrj-2:project:1
  ```

- 選択ルールによって選択された clear.c のバージョンを使用する。

  ```
  ccm use -rules clear.c
  ```

- SubPrj-one:project:1 プロジェクトを TopPrj-top:project:1 プロジェクトのルート ディレクトリに追加する (現在のディレクトリが任意のディレクトリである場合、またはプロジェクトが管理ワークエリアを含むかどうか分からない場合がある)。

  ```
  ccm use -p -dir TopPrj@TopPrj-top -task 31 SubPrj-one:project:1
  ```

- TopPrj-top:project:1 プロジェクトのルート ディレクトリの下の dir1 ディレクトリに、a.txt-1.2:ascii:1 オブジェクトを使用する。

  ```
  ccm use -dir TopPrj¥dir1@TopPrj-top -task 31 a.txt-1.2:ascii:1
  ```

- a.txt-1.1:ascii:1 オブジェクトの別バージョンを使用する。

  ```
  ccm use -dir TopPrj¥dir1@TopPrj-top a.txt-1.1:ascii:1
  ```

関連トピック

- delete コマンド
- unuse コマンド

別バージョンの使用または現在のディレクトリへのオブジェクトの追加

このサブコマンドにより、現在の作業ディレクトリにオブジェクトの別バージョンを使用するか、あるいは既存のオブジェクトを現在の作業ディレクトリの下に新しいプロジェクト メンバーとして追加します。現在の作業ディレクトリは、ユーザーが修正可能なプロジェクトの管理ワークエリア内になければなりません。共有プロジェクトでは、特別な処理が行われる場合もあります（shared_project_directory_checkin を参照してください）。コマンドを実行できるのは、コンテキスト プロジェクトが *working*（作業中）状態の場合は、そのプロジェクトの所有者です。コンテキスト プロジェクトが *prep*（準備）状態の場合は、ビルド マネージャです。

サブコンポーネントの別バージョンを使用する場合は、以下の点に注意が必要です。サブコンポーネントが親プロジェクトとは異なるコンポーネントを含む静的製品である場合、その製品の別バージョンはカレント タスクに関連付けられます。

```
ccm use [-r|-rules|-recommend] [-t|-task task_spec] file_spec...
```

file_spec

file_spec を参照してください。

`-r|-rules|-recommend`

選択ルールによって選択されたバージョンを使用します。

`-t|-task -task_spec`

-t|-task -task_spec を参照してください。

例

- `util-b2` プロジェクトと `tools-b2` プロジェクトを現在のディレクトリに追加する。

  ```
  ccm use -p util-b2 tools-b2
  ```

- 現在のディレクトリの下にある `file_top_1.txt` の推奨バージョンを使用する。

  ```
  ccm use -rules file_top_1.txt
  ```

- 現在のディレクトリに既存のメンバー `file_sub_1.txt-1` を追加する。

  ```
  ccm use -task 29 file_sub_1.txt-1:ascii:1
  ```

関連トピック

- delete コマンド
- unuse コマンド

説明と用途

use コマンドにより、以下のいずれかの操作を行います。

- 既存のファイル、ディレクトリ、プロジェクトを他のバージョンで置き換える。
- 現在のディレクトリ内にまだ存在しないファイル、ディレクトリ、プロジェクトを貼り付ける。

オブジェクトを書き込み禁止ディレクトリへ貼り付けると、新しいディレクトリ バージョンが自動的にチェックアウトされます。

共有プロジェクト内にいて、現在のディレクトリが書き込み禁止の場合、そのディレクトリはチェックアウトされ、デフォルト（または指定した）タスクと自動的に関連付けられ、*integrate*（統合）状態にチェックインされます。初期設定ファイル内の `shared_project_directory_checkin` を `FALSE` に設定して、自動チェックイン機能を無効にできます（shared_project_directory_checkin を参照してください）。

サブコンポーネントの別バージョンを使用する場合は、以下の点に注意が必要です。サブコンポーネントが親プロジェクトと異なるコンポーネントを持つ静的サブプロジェクトまたは製品である場合、サブプロジェクトまたは製品の別バージョンはカレント タスクと関連付けられます。この機能を無効にする場合は、初期設定ファイル内の `add_used_subcomponents_to_task` を `FALSE` に設定します（add_used_subcomponents_to_task を参照してください）。

ディレクトリを「使用」すると、そのディレクトリは自動的に更新されます。内容の変更を他のユーザーも利用できるようにするには、ディレクトリをチェックインする必要があります。

view コマンド

詳細については、説明と用途を参照してください。`view` コマンドは、ファイルの表示サブコマンドをサポートします。

ファイルの表示

```
ccm view file_spec...
```

file_spec

表示するオブジェクトを指定します。詳細については、ファイルの指定を参照してください。

例

- `log.c` オブジェクトのバージョン 8 を表示する。

  ```
  ccm view log.c-8
  ```

関連トピック

- cat コマンド（UNIX のみ）
- edit コマンド

説明と用途

`view` コマンドにより、指定したファイルを表示します。ファイルの表示にはデフォルトのビューアが使用されます。

ワークエリア参照形式またはプロジェクト参照形式のようにコンテキスト プロジェクトを与える形式でファイルを指定し、対応するワークエリアの場所がクライアントから見える場合は、ワークエリアの場所でビューアが起動します。プロジェクト コンテキストが使用できないか、対応するワークエリアが見えない場合、起動したビューアでは、データベースのファイルから一時的な読み取り専用コピーが開きます。

work_area コマンド

詳細については、説明と用途を参照してください。work_area コマンドは、以下のサブコマンドをサポートします。

- ワークエリア プロパティの修正
- ワークエリア プロパティの表示

ワークエリア プロパティの修正

このサブコマンドにより、ワークエリアの管理の有無やワークエリア パスなど、プロジェクトのワークエリア プロパティを変更します。プロジェクトを指定しない場合、現在の作業ディレクトリに関連付けられているワークエリアを持つプロジェクトのワークエリア プロパティが更新されます。

```
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`

ワークエリアをコピーベースに指定します。

`-lb|-link_based|-ncb|-not_copy_based`

ワークエリアをリンクベースに指定します。このオプションは、UNIX ユーザーのみが使用できます。

`-mod|-modifiable_wa`

ワークエリア内のファイルに、チェックアウトされていない場合でも修正可能となるようにアクセス許可を設定します。デフォルトは `-nmod|-not_modifiable_wa` です。

`-nmod|-not_modifiable_wa`

working などの書き込み可能状態の場合にのみ、デフォルトでワークエリア内のファイルが修正可能となるようにアクセス許可を設定します。これはデフォルト設定です。

`-nr|-no_recurse`

これらのオプションを使用すると、プロジェクト階層は再帰処理されません。指定したプロジェクトだけが変更されます。これはデフォルト設定です。

`-nrel|-not_relative`

ワークエリアを絶対パス上に置くよう指定します。

`-ntl|-no_translate|-no_translation`

ワークエリア内の ASCII ファイルが、復帰改行の変換を行わずに Windows と UNIX 間でコピーされるよう指定します。デフォルトは `-tl|-translate` です。

`-nwa|-no_wa`

プロジェクトが管理ワークエリアを持たないよう指定します。このデフォルトは -wa|-maintain_wa です。

`-nwat|-no_wa_time`

プロジェクトのワークエリア内のファイルで、ファイルがワークエリアにコピーされた時刻ではなく、Rational Synergy での最新の修正時刻を反映したタイムスタンプを使用するよう指定します。これはデフォルト設定です。

`-p|-project`

このオプションを指定する必要はありません。

`project_spec`

修正するプロジェクトを指定します。詳細については、プロジェクトの指定を参照してください。

`-pst|-project_subdir_template]` *`template_value`*

指定したプロジェクトのワークエリア パス（プロジェクトがファイル システムと同期される場所）を新しい場所に変更します。このパラメータは、ワークエリア パスのプロジェクト固有部分のみを変更します。ワークエリアのファイル システムの別の部分を変更する場合、あるいはワークエリアを別のプラットフォームと同期させる場合は、-set|-path|-setpath absolute_path を参照してください。

すべてのプロジェクトのワークエリアが作成されるデフォルトのディレクトリは、ホーム ディレクトリの下の `ccm_wa` の後ろに *`database_name`* を付けたものです。デフォルトでは、database_name の後ろにプロジェクト名とバージョンが付けられます。ワークエリア テンプレートを修正することにより、名前のプロジェクト固有部

分に *project_name*、*project_version*、*release*、*platform*、*delimiter* を含むように変更できます。

以前のパスがインターフェイス ホストから見える場合は、新しい場所に移されます。見えない場合は、このオプションを使用して work_area コマンドを実行すると、ワークエリアが作成されます。

`-r|-recurse`

指定したプロジェクトに応じて、プロジェクト階層内のすべてのプロジェクトが更新されるようにします。デフォルトは `-nr|-norecurse` です。

`-rel|-relative`

ワークエリア パスを親プロジェクトのパスに相対的とします。

`-set|-path|-setpath` *`absolute_path`*

プロジェクト用に指定したワークエリア パスを新しい場所に変更します。このオプションは、ワークエリア パスのプロジェクト非固有部分のみを変更します。ワークエリア テンプレートを修正することにより、`project_name`、`project_version`、`release`、`platform`、`delimiter` など、名前のプロジェクト固有部分を変更する場合は、-pst|-project_subdir_template] template_value を参照してください。

以前のパスがインターフェイス ホストから見える場合は、新しい場所に移されます。見えない場合は、このオプションを使用して `work_area` コマンドを実行すると、ワークエリアが作成されます。

読み出し専用プロジェクトのワークエリア パスを変更するには、ビルド マネージャまたは *ccm_admin* ロールを持つユーザーである必要があります。

`tl|-translate|-translation`

ワークエリア内の ASCII ファイルが、復帰改行の変換を行わずに Windows と UNIX 間でコピーされるよう指定します。

`-wa|-maintain_wa`

ワークエリアを管理します。このオプションを設定すると、ワークエリアの同期が保たれます。

CLI から同期を中止するには、任意の時点で Ctrl + C キー を押します。

ただし、同期処理を中止すると、ワークエリア内でエラーが発生した可能性があることを知らせる、エラー メッセージが表示されます。このエラーはワークエリアを使用し始めてから発生します。したがって、問題を回避するために、使用前にワークエリアを完全に同期してください。

読み出し専用プロジェクトにこのオプションを使用するには、*ccm_admin* ロール持っている必要があります。

`-wat|-wa_time`

プロジェクトのワークエリア内のファイルで、Rational Synergy での最新の修正時刻ではなく、ファイルがワークエリアにコピーされた時刻を反映したタイムスタンプを使用するよう指定します。デフォルトは `-no_wa_time` です。

関連トピック

- delimiter コマンド
- reconcile コマンド

ワークエリア プロパティの表示

このサブコマンドにより、プロジェクトのワークエリア プロパティを表示します。プロジェクトを指定しない場合、現在の作業ディレクトリに関連付けられているワークエリアを持つプロジェクトのワークエリア プロパティが表示されます。

```
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`

出力フォーマットでカラム ヘッダーを使用するよう指定します。詳細については、-ch|-column_headers を参照してください。

`-f|-format format`

コマンドの出力フォーマットを指定します。詳細については、-f|-format を参照してください。

キーワードには、組み込み済みのもの（`%fullname`、`%displayname`、`%objectname`）、あるいは `%modify_time`、`%status` などの既存の属性の名前を使用できます。

キーワードのリストについては、組み込み済みキーワードを参照してください。

`-gby|-groupby groupformat`

コマンドの出力をグループ化する方法を指定します。詳細については、-gby|-groupby を参照してください。

`-nch|-nocolumn_header`

出力フォーマットでカラム ヘッダーを使用しないよう指定します。詳細については、-ch|-column_headers を参照してください。

`-nf|-noformat`

カラム配置を使用しないよう指定します。詳細については、-nf|-noformat を参照してください。

`-ns|-nosort|-no_sort`

コマンドの出力をソートしないよう指定します。詳細については、-ns|-nosort を参照してください。

`p|-project`

-p|-project を参照してください。

`project_spec`

表示するプロジェクトを指定します。詳細については、プロジェクトの指定を参照してください。

`-sby|-sortby` *`sortspec`*

コマンドの出力をソートする方法を指定します。詳細については、-ns|-nosort を参照してください。

`-sep|-separator` *`separator`*

`-f|-format` オプションと一緒にのみ使用します。異なる区切り文字を指定します。詳細については、-sep|-separator を参照してください。

`-r|-recurse`

指定したプロジェクトに応じて、プロジェクト階層内のすべてのプロジェクトが表示されるようにします。デフォルトでは、指定したプロジェクトだけが表示されます。

例

- ワークエリア プロパティを表示する。

  ```
  ccm wa -show -recurse project-2
  ```

関連トピック

- reconcile コマンド

説明と用途

`work_area` コマンドにより、ワークエリア オプションの表示と修正を行います。

Rational Synergy ヘルプへのリンク

以下のリンクにより HTML または PDF 形式の Rational Synergy ヘルプを開きます。

- Rational Synergy Classic CLI ヘルプ HTML | PDF
- 開発者用 Rational Synergy ヘルプ HTML | PDF
- ビルド マネージャ用 Rational Synergy ヘルプ HTML | PDF
- エクスプローラ インターフェイス ヘルプ HTML | PDF
- タスクバー インターフェイス ヘルプ HTML | PDF

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものであり、本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、または サービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の 製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの 評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について 実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒 106-8711
東京都港区六本木 3-2-12
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

以下の保証は、国または地域の法律に沿わない場合は、適用されません。IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示 もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムと その他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を

可能にすることを目的として、本プログラムに関する情報を必要とする方は、製造元に連絡してください。

Intellectual Property Dept. for Rational Software
IBM Corporation
1 Rogers Street
Cambridge, Massachusetts 02142
U.S.A.

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

この情報をソフトコピーでご覧になっている場合は、写真やカラーの図表は表示されない場合があります。

商標

IBM および関連の商標については、www.ibm.com/legal/copytrade.shtml をご覧ください。Microsoft、Windows、Windows 2003、Windows XP、Windows Vista および / またはその他の Microsoft 製品は、Microsoft Corporation の米国およびその他の国における商標または登録商標です。

UNIX は、The Open Group の米国およびその他の国における登録商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

索引

記号

%baseline 59
%change_request 59
%change_request_duplicates 59
%change_request_original 59
%change_request_release 59
%change_request_status 59
%change_request_synopsis 59
%displayname 59
%fullname 59
%in_baseline 59
%in_build 59
%instance 60
%model 60
%objectname 60
%problem_duplicates 60
%problem_original 60
%purpose 60
%requirement_id 60
%root 60
%sourcename 60
%states 60
%task 60
%task_platform 60
%task_release 60
%task_status 60
%task_subsystem 61
%task_synopsis 61
%type 61
.ccm_addr ファイル 572
@cvid 22

A

add_object_task_assoc 69
alias、削除 612
allow_delimeter_in_name 69
allow_prep 72
AUTOMOUNT_FIX 100

B

baseline_template 72
baseline_template_date_format 73
baseline_template_repl_char 73

C

ccm.ini ファイル 66
    個人用ファイルの場所、UNIX 99
    個人用ファイルの場所、Windows 99
    システム フィルの場所 99
CCM_ADDR
    ccm_root として設定 570
    格納場所 572
    使い方の説明 100
ccm_eng.log
    出力のリダイレクトに使用 100
    場所 100
CCM_ENGLOG 100
CCM_HOME
    UNIX の場所 10
    Windows の場所 8
    変数の設定 100
CCM_PAGER 100
ccm_ui.log
    出力のリダイレクトに使用 100
    場所 100
CCM_UILOG 100
ccm クエリ
    オブジェクト バージョンの表示 22
    例 488
check_release 75
CLI、セッションの起動 569
column_headers オプション 51
compare_cmd 75
conflict_parameters 78
copy_db_always 79
copy_project コマンド 199
copy_to_file_system コマンド 206

D

date_modified 79
dcm_broadcast_dbid 79

dcm_time_sync_tolerance 80
DCM（分散型構成管理）
    イベント概要情報の表示 274
    イベントログ情報の表示 22, 276
    制限文字 57
    生成時刻の表示 278
    生成パッケージの転送 286
    データ送出 287
    データベース ID の表示 279
    データベース定義情報の表示 267
    プロパティの表示 270
default_task_query 81
default_version 81
delimiter コマンド 294
DISPLAY 100

E

engine_host 81, 82
expand_on_checkin 82

G

groupby
    オプション 50
    例 53

H

HOME 100
HTML
    デフォルトのブラウザ 82
    ヘルプ ファイル ロケーションのデフォルト 82
html_browser 82
html_location 82

I

IBM Rational ソフトウェア サポート 16
include_required_tasks 83
initial_role 83
initials、設定オプション 83

L

LD_LIBRARY_PATH 100

M

mail_cmd 85
multiple_local_proj_instances 85

N

nocolumn_headers オプション 51
noformat オプション 43
nosort、フォーマット例 52
NS、同期外れ記号 384

P

PAGER 101
PATH 101
PRINT_EDIT_CMD 101
PRINT_TOOL_CMD 101
proj_idx_wa_cache 87
project_subdir_template_unix 87

R

range_for_keyword_expand 88
README の内容 7
reconcile.control_files_below_new_project 90
reconcile.save_uncontrolled 90
reconf_consider_all_cands 90
reconf_stop_on_fail 90, 91
RECONF_TIME 101
reconfigure_parallel_check 91
reconfigure_using_tasks 91
release_phase_list 92
required_attributes 92
restrict_reconf_setting 93
デフォルト
    role 93

S

save_to_wastebasket 94
shared_project_directory_checkin 94
SHELL 変数 101
soad_scope コマンド 538
soad コマンド 528
start_day_of_week 95

sync
コマンド 578
system_filename_filters 95

T

TERM 101
text_viewer 76

U

UC（非管理）、説明 299
UIDPATH 101
UNC（汎用名付け規則）、説明 8
unnumbered オプション 51
unuse コマンド、削除するために使用 627
update_on_checkin_if_equal 96
USER 101

V

verbosity オプション 96

W

wa_path_cache_size 97
wa_path_template 97
wastebasket 96
work_area コマンド 649

い

一覧表示
フォルダ 333

え

英数字 55
エンジン
ログ ファイル 100

お

オフライン保存と削除
コマンド 528
作成 529
プレビュー 529
オブジェクト
アクセス管理 371
移動 393
インスタンス 21
関係の作成 504
関係の表示 505
クエリ 486, 599
グループ割り当て 368
グループ割り当ての解除 370
検索 485
現在のディレクトリへの追加 642, 644
個人使用のために保存 182
別バージョンの使用 644
最新バージョンの入手 639
作成 214, 217
指定 33
指定構文 25, 28
使用箇所の検索 317
新規 214
ソースの表示 646
チェックイン 169
チェックポイント 182
追加 640
データベース内で検索 485
転送セットからの削除 266
転送セットへの追加 220
名前の長さ制限 28
名前の変更 393
バージョンの変更 642, 644
フローティング、プロジェクトに追加 217
プロジェクトから削除 631
プロジェクト内で検索 29
別バージョンの使用 642
オブジェクトのクエリ 486
オブジェクト名
選択セット参照形式 22
ファイル 28
プロジェクト参照形式 29
ベースライン 25
ワークエリア参照形式 28
オブジェクト名形式 21
オプション
ccm.ini ファイル内の設定 68

groupby 50
unnumbered 51
verbosity 96
暗黙的な設定 556
区切り文字 42
初期設定ファイルの Options セクション 66
初期値が設定されている場所 556
オプション区切り文字
UNIX 10
Windows 8

か

階層、削除 289
カラム
配置 42
ヘッダー、フォーマット 51
要素、フォーマット 48
カレント タスク
解除 604
設定 604
説明 604
環境変数
CCM_ADDR、ccm_root として設定 570
関係
relate コマンドの使い方 507
削除 593, 620, 621
説明 503
表示 505
関係の解除
オブジェクト 622
関数、クエリとソート 490
管理データベース、プロセス ルールの設定 429
関連付け
タスクとオブジェクト 584
プロジェクトと目的 178

き

記号
NS（同期されていない）384
UN（非管理）384
共有プロジェクト、作成されるファイルの状態 217
キーワード
組み込み済み 59
属性名を使用 59
動作変更 150
フォーマット オプション 44

く

クエリ
関数とソート 490
式の構築 490
選択セット参照形式 22
タスク 599
フォーマットの制御 59
クエリ選択セットの表示 486
区切り文字
UNIX 10
Windows 8
定義 295
区切り文字オプション 42
グルーピング プロジェクトの更新の取り消し 618
グループ
オブジェクトへのグループの割り当て 368
オブジェクトへの割り当ての解除 370

け

形式
cvid 参照 23
選択セット、参照 22
ファイル内容 24
プロジェクト参照 29
現在のディレクトリ
プロジェクトの削除 630
プロジェクトの追加 641

こ

更新
一貫性のためのプロジェクト グルーピングの設定 471
候補の表示 157

コマンドの取り消し 618
ディレクトリ メンバー 635
取り消し 615, 617, 618
表示と時刻 101
元に戻す 617, 618
更新プロパティ
削除 461
タスクの追加 441
構文
cvid 参照形式 23
オブジェクトの指定 33
オブジェクト名形式 21
コマンド 19
タスクの指定 39
データベースの指定 27
転送セットの指定 40
ファイル内容形式 24
ファイルの指定 28
フォルダの指定 31
フォルダ テンプレートの指定 32
プロジェクト 36
プロジェクト グルーピングの指定 37
プロジェクト参照形式 29
プロジェクトの指定 36
プロセスの指定 34
プロセス ルールの指定 35
ベースラインの指定 25
変更依頼の指定 26
リリースの指定 38
ワークエリア参照形式 28
個人
ccm.ini ファイルの場所、Windows 99
ccm.ini ファイル、場所、UNIX 99
デフォルト設定 66
コマンド
alias 109
baseline 124
bom 151
candidates 154
cat 158
change_type 161
checkin 164
checkout 172
cmdhistory 188
conflicts 193
copy_project 199
copy_to_file_system 206
dcm 218
delete 288
delimiter 294
finduse 306
foder 320
folder_template 346
groups 367
history 373
ln 378
move 391
process 398
process_rule 414
project 436
project_grouping 440
project_purpose 473
properties 480
query 485
reconcile 491
release 508
set 552
show 557
show_servers 564
soad 528
soad_scope 538
start 568
stop 574
sync 578
task 582
undo_update 615
unset 624
unuse 627
update_members 634
use 640
view 646
work_area 649
構文 19
コマンドラインのデフォルト設定 67
コマンド履歴、表示 188
コンフリクト
オブジェクト コンフリクトの表示 194

更新処理 639
タスク コンフリクトの表示 196
マージ操作 390
リコンサイル操作 501

さ

再帰的に削除 289
最上位プロジェクト、作成 210
削除
シンボリック リンク 633
ディレクトリのプロジェクト 628, 630
サーバー、表示 564

し

式、クエリの構築 490
時刻更新操作 101
システム ccm.ini ファイル、場所、Windows 99
システム デフォルトの設定 66
指定
オブジェクト 33
タスク 39
データベース 27
転送セット 40
ファイル 28
フォルダ 31
フォルダ テンプレート 32
プロジェクト 36
プロジェクト グルーピング 37
プロセス 34
プロセス ルール 35
ベースライン 25
変更依頼 26
リリース 38
自動更新モード、プロジェクト グルーピングの設定 462
詳細、更新メッセージ 96
初期設定ファイル 99
個人 99
個人用エントリ作成場所 66
システム 99
場所 66
シンボリック リンク
削除 633
作成 379

せ

制限
DCM 文字 57
名前 55
文字 55
生成パッケージの転送 286
セキュリティ
設定の適用 371
読み出しを設定 371
レベルの割り当て 371
セッション
CCM_HOME 変数 100
インターフェイス アドレス 572
起動 572
終了 574
設定
タスク完了時の必須フィールド 92
ファイル パターン、同期時に無視 95
設定、属性の表示 557
選択セット
参照形式 22
順序付けと使用方法 490
選択セット、表示 486
全般的な使用方法 6

そ

属性
一覧表示 120
コピー 115
修正 121
新規 117
設定の表示 557
必須フィールドの設定 92
表示 122
変更 121
編集 121

た

タスク

オブジェクトとの関連付け 584
完了時の必須フィールドの設定 92
関連付けの解除 593
新規 591
オブジェクト 593
オブジェクト、タスク、変更依頼 584
オブジェクトの表示、変更依頼、タスク 606
解除 604
カレント 604
カレントの設定 604
関係 584
関係の解除 593
関係の変更 594
完了 586
関連付けの解除 593
関連付けられているオブジェクト 606
関連付けられている変更依頼 606
クエリ 599
検索 599
更新プロパティから削除 461
コピー 588
作成 591
作成と割り当て 604
指定 39
修正 594
状態の変更 610
情報の表示 608
設定 604
チェックイン 168
プロジェクト間のコピー 454
プロパティの表示 605
別の状態に遷移 610
変更 597
変更依頼 593
編集 597
割り当て 583
タスクの変更 597
タスクの編集 597

ち

チェックイン
オブジェクト 169
タスク 168
プロジェクト 165
著作権 658

て

ディレクトリ
置き換え 640
更新 634
自動的にチェックアウト 174
新規時に追加される場所 217
ファイルの削除 627
プロジェクトの削除 628
プロジェクトの追加 641
プロジェクト階層の修正可能バージョンの作成 200
マージ 390
メンバーの更新 635
ディレクトリの更新
取り消し 616
元に戻す 616
ディレクトリの更新の取り消し 616
ディレクトリの更新を元に戻す 616
ディレクトリの内容
一覧表示、UNIX 382
一覧表示、Windows 297
オブジェクトを長いフォーマットで一覧表示 383
データ、DCM を使用して送出 287
データベース 361
ID の表示 272
管理 521
最新のマークを付ける 243
命名制限 56
データベース指定、構文 27
データベース定義
削除 235
作成 222
修正 244
デフォルト
activecm.disable_sync_at_startup 69
add_object_task_assoc 69
allow_delimiter_in_name 69

allow_prep 72
baseline_template 72
baseline_template_date_format 73
baseline_template_repl_char 73
check_release 75
compare_cmd 75
conflict_parameters 78
copy_db_always 79
date_modified 79
dcm_broadcast_dbid 79
dcm_time_sync_tolerance 80
default_task_query 81
default_version 81
engine_host 81
expand_on_checkin 82
html_browser 82
html_location 82
include_required_tasks 83
initial_role 83
initials 83
mail_cmd 85
multiple_local_proj_instances 85
proj_idx_wa_cache 87
project_subdir_template_unix 87
range_for_keyword_expand 88
reconcile.control_files_below_new_project 90
reconcile.save_uncontrolled 90
reconf_consider_all_cands 90
reconf_stop_on_fail 90, 91
reconfigure_parallel_check 91
reconfigure_using_tasks 91
release_phase_list 92
required_attributes 92
restrict_reconf_setting 93
save_to_wastebasket 94
shared_project_directory_checkin 94
start_day_of_week 95
system_filename_filters 95
text_viewer 76
update_on_checkin_if_equal 96
verbosity 96
wa_path_cache_size 97
wa_path_template 97
wastebasket 96
格納場所 66
個人 66
コマンドライン 67
システム全体 66
設定する場所 66
ワークエリア ディレクトリ 579
デフォルトのブラウザ 82
転送セット
オブジェクトの削除 266
オブジェクトの追加 220
間接 CR メンバーの再計算 263
間接リリース メンバーの再計算 264
削除 236
作成 227
指定 40
修正 248
情報の表示 281
表示 284
編集 248
メンバーの再計算 265
メンバーの表示 282
転送パッケージ
受取り 259
生成 239

と

同期
中止 581
特記事項、法務知的財産 658

は

バージョン
長さ制限 28
履歴表示 373
パス
CCM_HOME - UNIX 10
CCM_HOME - Windows 8
プロジェクト固有ディレクトリの定義 87
プロジェクト非固有ディレクトリの定義 97
パフォーマンス、向上 97
番号付きフォーマット 48

汎用名付け規則（UNC）、説明 8

ひ

比較
    フォルダ 321
    マージするファイル 390
非管理（UC）
    記号 299
    定義 299
非管理記号 384
必須フィールド、定義 92
日付形式
    ベースライン テンプレート 73
標準プロセス 413

ふ

ファイル
    .ccm_addr 572
    ccm.ini 99
    ccm_eng.log 100
    ccm_ui.log 100
    移動 391
    大文字と小文字を区別した名前 58
    置き換え 640
    関係の追加 503
    削除 627
    指定 28
    使用箇所の検索 306
    新規、プロジェクト内の追加される場所 217
    追加 640
    同期外れの表示 299
    比較／マージ 387
    ファイル内容形式 24
    編集 303
    マージ 390
    マージ済み、マーク付け 390
    ローカル コピー記号 383
ファイル指定構文 28
ファイル名 28
フィールド、必須定義 92
フォーマット オプション 48
    column_headers オプション 51
    groupby 50
    groupby の例 53
    nocolumn_headers オプション 51
    noformat オプション 43
    nosort オプション 49
    nosort の例 52
    sortby オプション 49
    truncate の例 54
    unnumbered オプション 51
    wrap の例 54
    カラム配置 42
    カラム フォーマット 48
    カラム ヘッダー、説明 51
    キーワード 44
    区切り文字オプション 42
    説明 48
    ソートとグループ化 49
    フォーマット文字列 42
    フォーマット文字列、高度な使用 43
    プロパティのフォーマット 46
    例、使用 52
フォーマットつきでプロパティを表示 482
フォーマット文字列
    高度な使用方法 43
    説明 48
フォルダ
    一覧表示 333
    オブジェクトの表示 341
    コピー 324
    削除 330
    指定構文 31
    修正 335
    情報を表示 343
    新規作成 326
    タスクの表示 341
    比較 321
    プロセス ルールに追加 415
    プロパティの表示 340
フォルダ テンプレート
    一覧表示 355
    管理データベースの設定 361
    削除 352, 428
    作成 347
    指定 32

修正 357
情報の表示 364
表示 355
プロセス ルールに追加 415
プロパティの表示 363
変更 357
フローティング オブジェクト
プロジェクトに追加 217
プロジェクト
置き換え 640
オブジェクト コンフリクトの表示 194
オブジェクトの削除 631
書き込み可能バージョンの作成 200
現在のディレクトリへの追加 641
コピー 200
コピーの作成 206
最上位プロジェクトの作成 210
指定 36
新規 210
新規オブジェクトの追加 217
タスク コンフリクトの表示 196
チェックアウト 200
チェックイン 165
名前 36
名前の変更 392
表示 558
フローティング オブジェクトとして作成 217
プロジェクトのコピー 200
プロジェクト グルーピングとベースラインの比較 442
プロジェクト グルーピング
一覧表示 457
オブジェクトの表示 465
オブジェクト、ベースライン、タスクの表示 465
グルーピングとメンバーの削除 456
更新 618, 634
更新の取り消し 618
更新プロパティからタスクを削除 461
更新プロパティにタスクを追加 441
指定 37
自動更新モードの設定 462
情報の表示 468
説明 471
タスクの表示 465
表示 457
プロジェクト間のタスクのコピー 454
プロジェクトの比較 442
プロジェクトの表示 465
プロパティの表示 463
ベースラインとタスクの更新 470
ベースラインの表示 465
メンバーの更新 636
プロジェクト グルーピングの更新
取り消し 618
元に戻す 618
プロジェクト グルーピングの更新の取り消し 618
プロジェクト参照形式 29
プロジェクトの更新
取り消し 617
元に戻す 617
プロジェクトの目的
作成 474
新規 474
表示 477
マネージャ、説明 64
プロジェクト メンバー、更新 637
プロセス
一覧表示 404
既存のプロセスへのコピー 399
削除 403
作成 402
指定 34
修正 406
情報の表示 408
新規プロセスにコピー 400
表示 404
標準プロセス 413
プロパティの表示 410
変更 406
編集 406
プロセスの修正 406
プロセスの編集 406
プロセス ルール
管理データベースの設定 429

コピー 417
削除 419
作成 434
指定 35
修正 423
情報の表示 433
説明 434
表示 420, 523
標準の振る舞い 435
フォルダ テンプレートの追加 415
フォルダの削除 428
フォルダの追加 415
フォルダの表示 430
プロジェクト グルーピングで使用 472
プロパティの表示 432
ベースライン プロジェクトの表示 430
変更 423
編集 423
メンバーの表示 430
プロセス ルールの修正 423
プロセス ルールの編集 423
プロパティ
オブジェクト、フォーマット 46
表示 481
フォーマットつきで表示 482
分散型構成管理（DCM）
イベント概要情報の表示 274
イベントログ情報の表示 22, 276
データベース定義情報の表示 267
プロジェクト階層の修正可能バージョン、作成 200

へ

ベースライン
一覧表示 133
オブジェクト、タスク、CR の表示 143
公開 138
削除 132, 135
削除したもののリストア 141
削除の対象とする 135
作成 126
指定構文 25
修正 136
状態の定義 129
情報の表示 145
名前 25
比較 125
プレビュー 126
プロジェクトの追加 127
プロジェクトの比較 442
プロパティの表示 142
変更 136
編集 136
命名制限 56
リカバリ 141
リストア 141
リリース 139
ベースライン プロパティ、表示 142
ヘルプ、代替ロケーションの指定 82
変更
タスク 597
プロセス 406
プロセス ルール 423
ワークエリア プロパティ 650
変更依頼
構文 26
再計算 263
指定 26
変数
AUTOMOUNT_FIX 100
CCM_ADDR 100
CCM_ENGLOG 100
CCM_HOME 100
CCM_PAGER 100
CCM_UILOG 100
DISPLAY 100
HOME 100
LD_LIBRARY_PATH 100
PAGER 101
PATH 101
PRINT_EDIT_CMD 101
RECONF_TIME 101
SHELL 101
TERM 101
UIDPATH 101
USER 101

暗黙的に設定 626
変数の設定解除 626

ま

マージ
コンフリクト 390
ディレクトリ 390
ファイル 390
マージ済みファイルのマーク付け 390
マネージャ
説明 64
プロジェクト目的マネージャ、説明 64
プロセス ルール 64
リリース、説明 64

め

命名制限
オブジェクト 55
データベース 56
リリース 56
メンバー、削除 456

も

文字列、フォーマット、説明 48
問題、変更依頼を参照。

ゆ

ユーザー
オブジェクトのアクセスを制限 371
グループ化の定義 371

り

リスト
フォルダ テンプレート 355
リリース
一覧表示 515
管理データベースの設定 521
削除 514
作成 509
指定 38
修正 517
情報の表示 525
新規 509
プロセス ルールの表示 523
命名制限 56
履歴
記録する最大コマンド数の設定 191
現在のコマンドの表示 190
項目のクリア 189
コマンド、クリア 188
削除 189
履歴、表示 373
リンク
置き換え 640
削除 627

ろ

ローカル コピー
記号 298
説明 383
ロール、デフォルトの設定 93

わ

ワークエリア
オプションの変更 649
更新 578
参照形式 28
デフォルトの作成場所 579
パス定義 87
プロジェクトの作成 217
プロパティ、修正、変更、編集 650
プロパティの表示 654
リコンサイル 491
ワークエリアの更新 578
ワークエリア プロパティの修正 650
ワークエリア プロパティの表示 654
ワークエリア プロパティの編集 650