Rational. Synergy

IBM.

# Classic CLI ヘルプ

# IBM Rational Synergy
# Classic CLI ヘルプ
# リリース 7.1a

本書をご使用になる前に、557 ページの「付録：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 7.1a 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

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。

- README とドキュメント
- IBM Rational ソフトウェア サポートへの問い合わせ
- コマンド ライン インターフェイスの使用方法：Windows ユーザー
- コマンンド ライン インターフェイスの使用方法：UNIX ユーザー
- Rational Synergy インターフェイス
- Rational Synergy ヘルプ システム
- Rational Synergy 7.1a での用語と名称の変更
- コマンドと引数の構文
- 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
- 日付フォーマット
- 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

## README とドキュメント

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。README ファイルには、旧バージョンでリリースノートに含まれていた内容の多くが入っています。最新の README は、IBM Rational Software サポート ウェブ サイトから入手できます。

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

# 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。lhttp://www.ibm.com/software/lotus/passportadvantage/howtoenroll.htm

- 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。 |
| 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 ライン インターフェイスの使用方法：**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。一定のフォーマットを使用することにより、リモート マシンの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
```

# コマンンド ライン インターフェイスの使用方法：**UNIX** ユーザー

Rational Synergy は、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 は以下のインターフェイスを提供します。

- Synergy GUI

  このインターフェイスは、開発者およびビルド管理者向けの機能を提供します。管理者機能は提供しません。Synergy GUI は、ウェブモードとトラディショナルモードという 2 種類のモードで実行できます。モードの説明は、以下のウェブモードとトラディショナルモードを参照してください。

- Synergy CLI

  このインターフェイスは、開発者とビルド管理者向けの機能をほぼ完全に提供します。管理者機能は提供しません。また、ウェブモードのみがサポートされています。ウェブモードの説明は、ウェブモードとトラディショナルモードを参照してください。

- Synergy Classic GUI

  このインターフェイスの主目的は、管理機能の提供です。Classic GUI を使用しているユーザーは、Synergy GUI への移行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、将来のリリースで廃止される予定です。

- Synergy Classic CLI

  このインターフェイスの主目的は、管理機能の提供であり、既存スクリプトを Synergy CLI に変換するための移行期間を確保する目的で提供されています。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、将来のリリースで廃止される予定です。現在表示されているドキュメントは、Synergy Classic CLI ヘルプです。

### ウェブモードとトラディショナルモード

Synergy 7.1 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 | 6.5a と同じ | ウェブ モードで使用不可 |
| Synergy CLI | トラディショナル モードで使用不可 | 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 モードで使用不可 |
| 管理 | 6.5a と同じ | 6.5a と同じ。および、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 or HTTPS |
| ユーザー認証 | オペレーティング システム（OS） | IBM® Rational® Directory Server™ (TDS), (LDAP) |
| ワークエリア | 旧リリースと同じ | コピーベースのワークエリアのみサポート |

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。

## Rational Synergy ヘルプ システム

現在表示されているものは、Rational Synergy リリース 7.1a のコマンド ライン インターフェイス（CLI）用に更新されたヘルプです。このヘルプ システム内の情報は現在入手可能な最新の内容です。必ず README を調べて、本ヘルプ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 グラフィカル ユーザー インターフェイス（GUI）を使用する場合、そのインターフェイスで開くヘルプ システムにも同様の情報があります。Rational Synergy Classic のヘルプ システムにも CLI のオンライン ヘルプは入っていますが、最新のコマンド ライン情報ではなく、新規コマンドとオプションが含まれていません。リリース 7.1a で使用できる最新のコマンドとオプションを知りたい場合は、コマンド ライン インターフェイスの使用中にヘルプを起動してください。

Rational Synergy Classic インターフェイスの最新ヘルプは、リリース 6.3 のヘルプです。Rational Synergy Classic GUI のリリース 6.3 からの変更は微小です。具体的な変更点については、README を確認してください。

どのヘルプが表示されているのかが分からない場合は、各ヘルプ ページのフッターを見ればリリースが確認できます。

# Rational Synergy 7.1a での用語と名称の変更

Rational Synergy では、製品のいくつかの機能をコマンドライン インターフェイスとグラフィカル インターフェイスから除去する作業を行っています。この作業はいくつかのリリースをかけて行われる予定であり、お客様の仕事のサイクルに合わせてスクリプトを修正できるような機会を提供します。どのインターフェイスを使用して作業しているかによって、用語や名称に違いがあります。

いくつかのリリースで以下の名称が変更されました。

- リリース 6.4 より前のリリースでは、製品名は CM Synergy でした。CM Synergy は中心的なコンポーネントで、そのインターフェイスは、特に開発者用 CM Synergy という developer ロールで作業するユーザーを対象としていました。
- リリース 6.4 では、旧リリースで開発者用 CM Synergy と呼ばれていたインターフェイスは、SYNERGY/CM という名称になりました。その他のグラフィカル インターフェイスは SYNERGY/CM Classic と呼ばれています。
- リリース 6.5 では、旧リリースで開発者用 SYNERGY/CM と呼ばれていたインターフェイスは、Synergy という名称になりました。その他のグラフィカル インターフェイスは Synergy Classic と呼ばれています。
- リリース 6.6a では、旧リリースで SYNERGY/CM と呼ばれていたインターフェイスは、Rational Synergy という名称になりました。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。
- リリース 7.0 では、Rational Synergy インターフェイスは、ウェブモードとトラディショナルモードの 2 つのモードで実行できるようになりました。他のグラフィカル インターフェイスは、従来通り Synergy Classic と呼ばれます。このインターフェイスに対応するコマンドインターフェイスは、Synergy Classic CLI と呼ばれます。

いくつかの用語は、インターフェイス間で一貫性を保つために変更されました。このヘルプは、Synergy Classic コマンド ライン インターフェイスでの使用を目的としているため、ここに示す用語の変更は該当インターフェイスにも適用されています。Rational Synergy インターフェイスのヘルプでも新しい用語が使用されています。Rational Synergy Classic GUI インターフェイスで使用される用語は変わりません。下表は、旧リリースで使用された用語と、本 CLI ヘルプで使用される新しい用語を示します。

| **Classic CLI** と **Classic GUI** | リリース **6.4** で使用された用語 | **Synergy CLI** と **Synergy GUI** |
|---|---|---|
| リコンフィギュア | 更新 | 更新 |
| メンバーの更新 | 更新 | 更新 |

| **Classic CLI** と **Classic GUI** | リリース **6.4** で使用された用語 | **Synergy CLI** と **Synergy GUI** |
|---|---|---|
| リコンフィギュア テンプレート | 更新テンプレート | 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 | 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 | 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 | 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 | 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 | カレント タスク |

インターフェイス間で一貫性を保つため、コマンドも変更されました。たとえば、`ccm update` コマンドには `ccm reconfigure` という別名（エイリアス）があります。新しい用語に従って名前が変更されたコマンドについては、ヘルプにあるリンクから確認してください。スクリプトでリリース 6.3 と 6.4 のコマンドも使用できるように、別名が用意されています。利便性を考慮して、旧リリースで使用されていた用語は本リリースでのヘルプでも参照できるようになっています。

# 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。

  ```
  Windows:   ccm dir
  UNIX:      ccm ls
  ```

- Windows の管理コマンドについては、『IBM Rational Synergy 管理者ガイド Windows 版』を参照してください。すべてのドキュメントは、IBM Rational Information Center サイトで参照できます。
- UNIX では、いくつかの管理コマンドにもコマンド接頭辞 `ccm` を使用しますが、アンダスコアも必要です。管理コマンドは以下のように入力します。

  ```
  $ ccm_install
  ```

- UNIX の管理コマンンドについては、『IBM 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『IBM Rational Synergy Administration Guide for UNIX (Oracle)』を参照してください。

コマンド セット内の多くのコマンドは、引数として *`file_spec`* または *`project_spec`* を必要とします。これらの引数を使用することで、Rational Synergy データベース内の管理オブジェクトを指定できます。 その他の共通引数には、*`task_spec`*、*`folder_spec`*、*`change_request_spec`* があります。

以下のトピックについて説明します。



## ベースラインの指定

*`baseline_spec`* はベースライン名の参照です。*`baseline_spec`* は、以下のいずれかの形式をとることができます。

- *`baseline_name`*

- 選択セット参照（*@number*）
- 選択セット参照の全体（@）

データベースのリリース時に、前に DCM データベース ID （dbid）と DCM 区切り文字を含む *baseline_spec* を指定できます（例、`J#;`、ただし `J` は dbid、`#` は DCM 区切り文字）。

### 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。属性を指定すると、コピーする `prep`（準備）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。

### 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され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、プラットフォームが存在するかどうかに従って `"solaris_7.1a"` または `"7.1a"` を得るには、以下のように指定します。

```
%{platform:-}%{platform:+_}7.1a
```

- `%{platform:-}` は、プラットフォームが `solaris` の場合 `"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。

## 変更依頼の指定

*change_request_spec* は、1 つ以上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。

- *change_request_number*（1 つの変更依頼番号）
- *change_request_number*,*change_request_number*（カンマで区切られた変更依頼番号のリスト）
- *change_request_number-change_request_number*（値の範囲）
- 選択セット参照（*@number*）

## ファイルの指定

*file_spec* は、Rational Synergy データベース内の非プロジェクト オブジェクト バージョンの参照です。

*file_spec* は、以下の 4 つのいずれかの形式をとることができます。

- ワーク エリアの参照形式
- 選択セットの参照形式
- プロジェクトの 参照形式
- オブジェクトの参照形式

*object_name* には最大151文字、オブジェクトの *version* には最大32文字を指定できます。

### ワーク エリアの参照形式

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体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¥ccmtest¥jobA-1¥jobA` の場合は、相対パス *file_spec* 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
```

バージョンに *file_spec* 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にできます。

*path*¥*object_name*[*-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-1` がある場合は、`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-1
```

`ccm delimiter` コマンドを使用して、オブジェクトとバージョン間の区切り文字を変更できます。このコマンドはデータベース全体の区切り文字を変更するので、こ

のためには *ccm_admin* ロールが必要です （delimiter コマンドの注記を必ず参照してください）。

### 選択セットの参照形式

コマンド ccm query、ccm candidates および ccm show は、オブジェクト バージョンのリストを表示します。オブジェクト バージョンのリストを、「選択セット (*selection set*)」といいます。コマンド ライン選択セット内のオブジェクト バージョンを、*file_specs* として使用できます。

たとえば、ユーザー *joseph* が所有する foo.c という名前のすべてのオブジェクトをリストするには、以下のコマンドを入力します (コマンドの後に選択セットを指定します)。

```
ccm query -name foo.c -owner joseph
1) foo.c-1 integrate joseph csrc 1
2) foo.c-2 integrate joseph csrc 1
3) foo.c-3 integrate joseph csrc 1
4) foo.c-4 working joseph csrc 1
```

コマンド ライン選択セットへの参照形式を使用して、foo.c-2 に関する情報を受け取るには、ccm properties コマンドを以下のように指定します。

```
ccm properties @2
```

コマンドによっては、記号 @ のみを使用してコマンド ライン選択セットのすべての内容を参照できます。

```
ccm properties @
```

### プロジェクトの 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るが、プロジェクトがファイル システム内のワーク エリア ディレクトリ内にない場合は、プロジェクトの相対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。

*relative_path*¥*object_name*[*-version*]*@project_name-project_version*

たとえば、dir1 ディレクトリ内の jobA-1 プロジェクトの foo.c-4 オブジェクト バージョンのプロジェクト参照形式を使用するには、以下のように *file_spec* を使用します。

```
jobA¥dir1¥foo.c@jobA-1
```

参照対象である foo.c のバージョンを知らなくてもかまいません。メンバーのプロジェクト名だけが必要です。

> 注記：DCM データベースでは、*project_spec* は 4 部名称とします（*object_name-version*:*type*:*instance*）。*project_spec* の詳細については、プロジェクトの指定 を参照してください。

### オブジェクトの参照形式

オブジェクトの参照形式（「4 部名称」）には、オブジェクト バージョンの名前、バージョン、タイプ、インスタンスが必要です。オブジェクトの参照形式は以下のとおりです。

*object_name-version*:*type*:*instance*　または
*object_name:version*:*type*:*instance*

たとえば、`foo.c-4` オブジェクト バージョンの 2 番目のインスタンスである `csrc` オブジェクトの場合、以下のオブジェクト参照形式を使用します。

```
foo.c-4:csrc:2
```

> 注記：異なるユーザーが同じ名前とタイプの複数のオブジェクトを持つ場合があります（たとえば、複数の `csrc` ファイルが同じ名前 `main.c` を持つ場合）。インスタンスは、同じ名前とタイプのオブジェクトを区別します。

## フォルダの指定

*folder_spec* は、1 つのフォルダ番号の参照です。*folder_spec* は、以下のいずれかの形式をとることができます。

- *folder_number*（整数値）
- DCM 形式 *folder_number* (*database_id*#*folder_number*)
- 4 部名称（*object_name-version*:*type*:*instance*）
- 選択セットの参照（@*number*）
- **1** つのフォルダ番号を持つファイルの名前（*folder_id_filename*）

*folder_specs* が複数のフォルダ番号を参照する場合、*folder_specs* は以下のいずれかの形式をとることができます。

- *folder_specs*,*folder_specs*（値のリスト）
- *folder_specs*-*folder_specs*（値の範囲）
- **1** つ以上のフォルダ番号を持つファイルの名前（*folder_id_filename*）

*folder_specs*（複数引数）構文で使用される *folder_spec* は、*folder_number*、DCM の *folder_number*、あるいは選択セット形式である必要があります。

たとえば、`irv` データベースの 14 番目のフォルダと現在のデータベース（`irv` 以外）の 6、7、8 番目のフォルダを `jobA-1` プロジェクトの更新プロパティ に追加するには、以下のように *folder_specs* を使用します。

```
ccm up -a -folders irv#14,6-8 jobA-1
```

### フォルダ テンプレートの指定

*folder_template_spec* は、1 つのフォルダ テンプレートの参照です。*folder_template_spec* は以下のいずれかの形式をとることができます。

- *folder_template*（"All completed tasks for release %release" などの名前）
- DCM 形式 *folder_template* (*database_id*#*folder_template*)
- 4 部名称（*object_name-version*:*type*:*instance*）
- 選択セットの参照（*@number*）
- **1** つのフォルダ名を持つファイルの名前（*folder_template_filename*）

*folder_template_specs* が複数のフォルダ テンプレートを参照する場合、*folder_template_specs* は以下のいずれかの形式をとることができます。

- *folder_template_specs,folder_template_specs（値のリスト）*
- **1** つ以上のフォルダの説明を持つファイルの名前（*folder_template_description_filename*）

*folder_template_specs*（複数引数）構文で使用される *folder_template_spec* は、*folder_template*、DCM の *folder_template*、あるいは選択セット形式である必要があります。

たとえば、irv データベースの 9 番目のフォルダ テンプレートと現在のデータベース（irv 以外）の 4 番目のフォルダ テンプレートをリリース 3.2 の "Integration Testing" 目的でプロセス ルールに追加するには、以下の *folder_template_specs* を使用します。

```
ccm process_rules -add -folder_temps irv#T9,T4 3.2:"Integration Testing"
```

### プロセス ルールの指定

*process_rule_spec* はプロセス ルールの参照です。*process_rule_spec* は以下のいずれかの形式をとることができます。

- 選択セット参照（*@number*）
- 4 部名称
- リリース固有ルールでは、*release*:*name of generic process rule* 汎用プロセス ルールでは *defined_name*

> 注記：古い形式の *update_template_spec* は使用でなくなりました。ただし、指定された目的名が汎用プロセス ルール名と同である場合のみ実行できます。

たとえば、フォルダ テンプレート T7 をリリース固有プロセス ルール 1.0: Collaborative Development に追加するには、以下の *process_rule_spec* を使用します。:

```
ccm process_rule -add -folder_temp T7 1.0:"Colloborative Development"
```

## 問題の指定

*problem_spec* は、1 つ以上の問題の参照です。現行バージョンで「問題」は「変更依頼」と呼び方が変更されているため、将来的には *problem_spec* は *change_request_spec* に置き換わります。*problem_spec* は以下のいずれかの形式をとることができます。

- *problem_number*（1 つの問題番号）
- *problem_number*,*problem_number*（値のリスト）
- *problem_number*-*problem_number*（値の範囲）
- 選択セット参照（@*number*）

## プロジェクトの指定

*project_spec* は、プロジェクト オブジェクト バージョンの参照です。

コマンドが引数としてプロジェクトを必要とする場合は、*project_spec* を指定します。DCM データベースでは、プロジェクトが dbid#1 のデフォルト インスタンスを持たない場合、*project_spec* は 4 部名称（*object_name*-*version*:*type*:*instance*）である必要があります。ただし、dbid はデータベースのデータベース識別子です。非ローカル プロジェクト インスタンスが無効な状態で、かつ、現在のデータベースに作成されたプロジェクトの場合（デフォルト設定）、プロジェクトは 2 部名称で指定できます。それ以外の場合、*project_spec* は以下のようにプロジェクトの名前とバージョンで構成されます。

*project_name*-*project_version*

たとえば、jobA-1　プロジェクトについての情報を表示するには、以下のように *project_spec* を使用します。

```
ccm properties -p jobA-1
```

*project_name* には最大 155 文字、プロジェクトの *version* には最大 32 文字を指定できます。

デフォルトの区切り文字は、-（ハイフン）です。区切り文字の変更方法については、delimiter コマンドを参照してください。

以下の形式を使用して、*project_spec* を構文解析することもできます。

*project_name version_delimiter* : project_version : *project_instance*

2 部形式で指定すると（つまり、プロジェクト インスタンスを省略）、以下のようにデフォルト インスタンスが使用されます。

- DCM 用に初期化されていないデータベースの場合：1
- DCM 用に初期化されたデータベースの場合：*dbid dcm_delimiter* 1（例、A#1）

## プロジェクト グルーピングの指定

*project_grouping_spec* はプロジェクト グルーピングの参照です。プロジェクトをグループ化することにより、プロジェクトのグループを同じベースラインとタスクで更新できます。

*project_grouping_spec* は、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 グルーピングは 4 部名称であり、以下で構成される。
  - *name* 属性は、プロセス ルール名で使用される `release` および `member_status` と同じ ASCII コード。
  - *version* 属性は常に `1`。
  - *type* は常に `project_grouping`。
  - *instance* は以下のいずれかとなる。
    DCM 用に初期化されていないデータベースの場合、プライベート プロジェクト グルーピングのインスタンスは、その所有者と同じ。DCM 用に初期化されたデータベースの場合、インスタンスはローカル データベース ID、DCM 区切り文字、所有者で構成される。
    DCM 用に初期化されていないデータベースの場合、非プライベート プロジェクト グルーピングのインスタンスは、`1`。DCM 用に初期化されたデータベースの場合、インスタンスは作成元のデータベースのデータベース ID、DCM 区切り文字、および `1` で構成される。

  以下に、4 部名称の例を示します。
  `CM%002f7.1a%3Acollaborative-1:project_grouping:linda`
- プロジェクト グルーピングのデフォルト表示名。ユーザーがプロジェクト グルーピングの所有者である場合、`My` の代替として *`user_name's`* が使用できます。project_grouping の表示名がカスタマイズされていても、*project_grouping_spec* の構文には影響ありません。

  表示名の例として、`"Linda's Synergy/7.1a Collaborative Development Projects"`、`"All Synergy/7.1a Integration Testing Projects for Database P"` などがあります。
- クエリから選択セット内のプロジェクト グルーピングを参照するクエリ参照 (*`@number`*)、`@1` など。
- 複数のプロジェクト グルーピングが可能なコマンドの場合、選択セット全体を参照するクエリ参照、`:@` など。

## タスクの指定

*task_spec* は、1 つのタスク番号の参照です。*task_spec* は以下のいずれかの形式をとることができます。

- *task_number*（整数値）
- DCM 形式 *task_number* (*database_id#task_number*)
- 4 部名称（*object_name-version*:*type*:*instance*）
- 選択セット参照（*@number*）
- 1 つのタスク番号を持つファイルの名前（*task_id_filename*）

*task_specs* が複数のタスク番号を参照する場合、*task_specs* は以下のいずれかの形式をとることができます。

- *task_specs*,*task_specs*（値のリスト）
- *task_specs*-*task_specs*（値の範囲）
- **1** つ以上のタスク番号を持つファイルの名前（*task_id_filename*）

> 注記：複数指定を参照する task_specs は、カンマと空白文字で区切ることができます。

*task_specs*（複数引数）構文で使用される *task_spec* は、*task_number*、DCM *task_number*、あるいは選択セット形式である必要があります。

たとえば、クエリ出力内の現在のデータベースの 5、8、9、10 番目のタスクを jobA-1 プロジェクトの更新プロパティ に追加するには、以下のように *task_specs* を使用します。

```
ccm up -a -tasks @5,@8-@10 jobA-1
```

# 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、delimiter コマン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。
- 空白を含むファイル名を持つソース ファイルでのキーワード展開は、コンパイル時にエラーが発生することがある。ソース コード ファイル名での空白の使用を避けるか、キーワードをコメントアウト（または削除）します。

  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）

## リリース名の制限

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

- リリース名には、上記の表に示した制限文字は使用できない。
- 値 `Any`、`None`、`none`、`as_is` および `Default Release` は予約値であり、使用できない。
- 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。

- 名前には # 文字を使用できない。

## DCM の制限

以下に、DCM データベースの命名制限を示します。

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。デフォルトの DCM 区切り文字は "#" です。可能な限りこの文字の使用を使用してください。

# 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。

## 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『IBM 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 256 文字（Windows）または 155 文字（UNIX）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『IBM Rational Synergy 管理者ガイド』の `ccmdb_info` コマンンドの説明を参照してください。

## 日付フォーマット

Rational Synergy で使用可能な日付フォーマットの詳細については、日付形式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リストするには、`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。 |
| `%dcm_delimiter` | DCM 用に初期化されていないデータベースの場合はシャープ記号（#）を返します。DCM 用に初期化されているデータベースの場合は実際の DCM 区切り文字を返します。 |
| `%displayname` | 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。 |

| キーワード | 説明 |
|---|---|
| %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。 |
| %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。 |
| %optional_project_instance | デフォルトのインスタンス値（"1"）を持つ場合は空白文字列を返します。デフォルト値以外の場合は、DCM 区切り文字とプロジェクト インスタンスを返します（"*dbid*#1"）。 |
| %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。 |

| キーワード | 説明 |
|---|---|
| %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。 |
| %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

一部のコマンドでは以下の正規表現を使用できます。正規表現を使うと、指定した文字列値との照合や結果の文字列での置換（オプション）を指定できます。

- 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：<br>`str ? * "^abc"` が一致するのは、`str` が `abc` で始まる場合のみです。 |
| $ | 文字列の末尾と一致します。例：<br>`str ? * "abc$"` が一致するのは、`str` が `abc` で終わる場合のみです。 |
| . | 任意の 1 文字と一致します。例：<br>`str ?* "a.c$"` 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：<br>`str ? * "ab*c$"` は、ac、abc、または abbc などを含む `str` の値と一致します。 |
| + | 直前の長さ 1 以上の表現と一致します。例：<br>`str ?* "a+c$"` 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 つの表現と一致します。例：<br>`str ? * "ab?c"` は、`str` が `ac` または `abc` を含む場合のみ一致します。 |
| \| | 直前または直後の表現と一致します。<br>例：<br>`str ?* "a\|b\|c"` は、a、b または c を含む `str` と一致します。 |

| 文字 | 用途 |
|---|---|
| [ ] | [ ] 内にリストされた任意の 1 文字と一致します。<br>例：<br>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：<br>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。 |
| ¥ | 直後の文字のエスケープ文字です。例：<br>str ? * "a¥.c$" は、str が a.c を含む場合に一致し、また str ? * "a¥¥c$" は str が a¥c を含む場合に一致します。<br>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：<br>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

キーワード 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 文字としての * 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 マネージャが持たない場合は、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（以前は、リコンフィギュア プロパティおよび更新テンプレートマネージャと呼ばれていました）は、プロセス ルールを使用するデータベースのために存在します。

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

Rational Synergy には、インストールしてすぐ使用できるように、デフォルトと呼ばれる一連の定義済みの値または設定が出荷時に用意されています。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
- 起動ファイル
- GUI 設定
- 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。これらのオプションのいくつかは、Rational Synergy のオプション ダイアログからも利用できます。

- Rational Synergy のオプション ダイアログボックス

  いくつかのオプションは オプション ダイアログボックスで設定されます。これらのオプションのいくつかはコマンドラインからも設定できます。

## 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、セッションを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 Options 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` レベルの設定より優先します。

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` レベルの設定より優先します。

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。

システムの ini ファイル、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。

## 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（デフォルト）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他にはありません。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

このオプションを使用して、プロジェクトの更新時に *prep* サブプロジェクトを含めることができます。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、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）コンテンツで上書きするリスクが伴います。以下に発生する可能性のあることを説明します。

-- 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、*prep* 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。

## 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、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、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、使用され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、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、prep 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 製品が CM/7.1a リリースを持つ場合、このバージョン文字列には文字列 CM_7.1a 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_dir_compare_cmd
### cli_symlink_compare_cmd
### cli_merge_cli
### cli_dir_merge_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、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、ディレクトリまたはシンボリック リンクではないオブジェクトのことです。このオプションはマージのために選択された 1 番目のディレクトリの cli_compare_cmd 属性となり、この値はデフォルトで %ccm_merge です。

cli_dir_merge_cmd は、CLI から 2 つのディレクトリ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。　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_dir 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より優先します。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、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指示します。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。含めなかった場合は、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、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指示します。

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、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` | オブジェクトの `attrname` 属性値が `attrvalue` に一致するコンフリクトを除外します。 |
| `attrname!=attrvalue` | オブジェクトの `attrname` 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リストし、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
```

各コンフリクトの説明も含むコンフリクトの詳細については、コンフリクト検出 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。

## 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、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、技術サポートが使用するためのものです。そのようなデバッグが必要ない場合は、このオプションは無効にしておいてください。

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。

### 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、ドキュメント『IBM Rational Synergy Distributed』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、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、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、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

データベース アドミニストレータが cvtype に属性を追加可能となり、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。

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有効に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、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_default_file

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

この設定は UNIX オペレーティング システムでのみ有効です。

代替デフォルト HTML ヘルプ ファイルを指定します。HTML ヘルプを呼び出すと、このデフォルト HTML ページが表示されます。

デフォルトは `ccm.htm` です。

システムの ini ファイル、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、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、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、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、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。

### migrate_check_required_task

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

このオプションを TRUE に設定すると、マイグレーション操作によって、新しいバージョンの作成とチェックアウト、およびチェックアウトされたバージョンの先行バージョンのチェックインに、タスク要件が実施されます。タスク要件は、各タイプの Require Task At オプションで定義されます。

デフォルトは TRUE です。

システムの ini ファイル、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。

### migrate_default_arch_state

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

アーカイブ ファイルマイグレーション時に使用するデフォルトの初期状態を指定します。

デフォルトは `integrate`（統合）です。

システムの ini ファイル、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。

### migrate_default_state

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

ファイルのマイグレーション時に使用するデフォルトの初期状態を指定します。

デフォルトは `integrate`（統合）です。

システムの ini ファイル、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。

### migrate_default_type

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 へのファイルのマイグレーション時に使用するデフォルトのファイル タイプを設定します。デフォルト タイプが特別な処理を必要とするときは、`project` または `dir` には指定できません。

> 注記：デフォルト タイプは、マイグレーション操作のほかに、作成操作およびリコンサイル操作にも使用されます。

デフォルトは ascii です。

> 注記：**IBM** は ISO-Latin-1（ISO 8859-1）文字セットをサポートしていません。

システムの ini ファイル、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。<br>例：`status=rejected` |

| 構文 | 説明 |
|---|---|
| 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。<br>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。<br>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。<br>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。<br>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。<br>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外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、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、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 2048 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 33 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、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、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、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（リコンフィギュア）プロパティに候補が存在しないとき、もっとも適合したものをディレクトリに実装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。

モデルを Rational Synergy リリース 4.5 またはそれ以前のリリースに一致させたい場合は、このオプションを TRUE に設定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。

### 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もっと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。詳細については、IBM Rational Information Center を参照してください。

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（リコンフィギュア）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、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、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`、`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（リコンフィギュア）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）を設定できます。

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、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（リコンフィギュア）プロパティを "object status" から "tasks" へ、またはその逆に変更するかを指定します。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。このオプションの設定または変更をするには、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、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（更新テンプレートまたはリコンフィギュア テンプレート）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、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、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、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。

### sync_output

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

同期状態メッセージをコマンドラインから表示させる場合は、何もする必要はありません。デフォルトでこのメッセージが表示されます。このメッセージを表示させないようにするには、以下のいずれかの方法で、`sync_output` オプションを設定します。

システムの ini ファイル、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。

### 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 |
| *.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、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、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、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、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部分ではない部分を定義しているデフォルトのワークエリアパス テンプレートを変更できます。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` をユーザー名で置き換えます。
`%owner` は `%owner` をプロジェクトの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,` です。`%home` はホーム ディレクトリです（ホーム ディレクトリを **Startup Info** 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` です。`%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 ファイルはインターフェイスのインストール 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更新します。

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。

# 起動ファイル

起動ファイル コマンドは ccminit という名前の Rational Synergy ファイルに保存されています。起動ファイル コマンドを使用して以下のことを実行できます

- コマンドの定義
- 別名の定義
- コマンドの実行

以下に対応する起動ファイル コマンドを変更できます。

a. インストール エリア
b. 特定のデータベース
c. 個人用セッション

Windows 起動ファイルの読み込みと実行は以下の順序で行われます。

- *CCM_HOME*¥etc¥ccminit
- *database_directory*¥lib¥ccminit
- *user's_home_directory*¥ccminit

UNIX 起動ファイルの読み込みと実行は以下の順序で行われます。

- *CCM_HOME*/etc/ccminit
- *database_directory*/lib/ccminit
- *user's_home_directory*/.ccminit

これら 3 つの起動ファイルがすべて存在する場合、Rational Synergy はすべてにあるコマンドを読み込んで実行します。それぞれの追加の起動ファイルには起動時に実行する追加のコマンドがあります。

Rational Synergy は最初にインストール エリアの設定を、次にデータベース固有の設定を、その次に個人用の設定を読み込みます。後から読み込むファイルのコマンドが、その前のファイルから読み込んだコマンドに優先します。

> 注記：どの起動ファイルを変更した場合も、新しいコマンドを使用可能にするには、セッションの再起動が必要です。

## インストール エリアの設定

このファイルでのコマンドの設定は、インストール エリアの全ユーザーに影響します。これらのコマンドはシステム起動ファイル（ccminit）に設定されます。

システム起動ファイルは以下のディレクトリにあります。

Windows： *CCM_HOME*¥etc
UNIX： $*CCM_HOME*/etc

## データベース固有の設定

このファイルでのコマンドの設定は、特定データベースの全ユーザーに影響します。これらのコマンドはデータベース起動ファイル（ccminit）に設定されます。

データベース起動ファイルは以下のディレクトリにあります。

Windows：*database_directory*¥lib
UNIX： *database_directory*/lib

## 個人設定

このファイルでのコマンドの設定はユーザー独自のセッションにのみ効果があります。これらのデフォルトをユーザーの個人用起動ファイル（Windows では ccminit 、UNIX では .ccminit ）に設定できます。

個人用起動ファイルをまだ持っていない場合は、自分のホーム ディレクトリに作成する必要があります。

## 例

自分のセッションにのみ恒久的に別名を設定し、ホーム ディレクトリには ccminit ファイルが存在していないと仮定します。

1. 自分のホーム ディレクトリに起動ファイルを作成します。
2. 起動ファイルに自分の別名を追加します。

   たとえば、起動ファイルに以下の行を含むことができます。

   ```
   alias my_tasks "task -query -task_scope all_my_assigned"
   alias cidt "task -checkin default"
   alias exit stop
   ```

   注記 : 必要な数だけ起動ファイルに別名を追加できますが、各別名はそれぞれ 1 行に記述する必要があります。

   注記：Rational Synergy コマンドについては先頭に ccm を付けないでください。たとえば、alias を設定するコマンドは ccm alias です。上記の例の最後の行では、コマンドには alias のみが含まれます。

3. 変更を保存し、起動ファイルの編集を終了します。
4. セッションを実行中の場合は、Rational Synergy を終了し、セッションを再起動します。

   別名は自分で使用するためにあります。

# GUI 設定

Rational Synergy グラフィカル ユーザー インターフェイスの設定は、オプション ダイアログボックスを使用して変更できます。

## 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<br>（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。<br>詳細については、『IBM 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<br>（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。設定下場合、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<br>（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<br>（UNIX のみ） | 必須 | UNIX のホーム ディレクトリを指定します。 |

| 環境変数 | 必須／任意 | 使い方 |
|---|---|---|
| LD_LIBRARY_PATH<br>(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、<br>/usr/lib/X11:/usr/openwin/lib |
| PAGER<br>(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 | 更新（リコンフィギュア）コマンドの実行時間の指定と表示を行います。 |
| SHELL<br>(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<br>(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<br>(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<br>(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。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である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。

最初の例では、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_deliminiter_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ことを行います。

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、そのうちの 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、復帰改行の区切り文字の一部と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_values 属性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 `ccm.ini` ファイル内のオプションを、次の例では UNIX `.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

## 表記

```
ccm alias [alias_name ["string"]]
```

## 説明と用途

alias コマンドにより、コマンドの別名を作成できます。このコマンドは、以下のように使用します。

- 引数なしで使用した場合、alias コマンドは定義済みの別名を一覧表示する。
- 1 つの引数とともに使用した場合、alias コマンドは *name* の値を出力する。
- 2 つの引数とともに使用した場合、alias コマンドは新しい別名 *name* を値 *string* に設定するか、既存の別名 *name* の値を *string* に変更する。

alias コマンドを設定すると、別名は現在のセッションにのみ適用されます。別名を恒久的に設定する方法については、起動ファイルを参照してください。

## オプションと引数

*name*

別名の名前を指定します。

*string*

別名で置き換えるコマンドを指定します。

別名はコマンドを直接置き換えるテキストなので、*string* にはコマンド名、完全なコマンド、またはコマンドを一部を指定できます。*string* にコマンドと引数を指定する場合は、コマンド全体を引用符で囲みます。

## 例

- 定義済みの別名をすべて一覧表示する。

  ```
  ccm alias
  ```

- 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。

関連トピック

- unalias コマンド

# **attribute** コマンド

## 表記

### 属性のコピー

```
ccm attr|attribute -cp|-copy [attr_name[:attr_name...]
                   [-append] from_file_spec to_file_spec
                   [to_file_spec...]
ccm attr|attribute -cp|-copy attr_name[:attr_name...]
                   [-append] [-subproj] [-suball]
                   -p|-project from_project_spec
                   to_project_spec [to_project_spec...]
```

### 属性の作成

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

### 属性の削除

```
ccm attr|attribute -d|-delete attr_name file_spec [file_spec...]
ccm attr|attribute -d|-delete attr_name
                   -p|-project project_spec [project_spec...]
```

### テキスト属性の修正

```
ccm attr|attribute -m|-modify attr_name [-v|-value value]
                   file_spec [file_spec...]
ccm attr|attribute -m|-modify attr_name [-v|-value value]
                   -p|-project project_spec [project_spec...]
```

### 非テキスト属性の修正

```
ccm attr|attribute -m|-modify attr_name -v|-value value
                   file_spec [file_spec...]
ccm attr|attribute -m|-modify attr_name -v|-value value
                   -p|-project project_spec [project_spec...]
```

### 属性の表示

```
ccm attr|attribute -s|-show attr_name file_spec [file_spec...]
ccm attr|attribute -s|-show attr_name
                   -p|-project project_spec [project_spec...]
```

属性の一覧表示

```
ccm attr|attribute -l|-la|-li file_spec [file_spec...]
ccm attr|attribute -l|-la|-li
                   -p|-project project_spec [project_spec...]
```

## 説明と用途

attribute コマンドにより、Rational Synergy のオブジェクトに関する属性を操作します。属性が text 属性の場合は、-v の使用は任意です。

## オプションと引数

-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_name*[:*attr_name*...]

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を使用できます。

-c|-create *attr_name*

属性を作成します。

-d|-delete *attr_name*

属性を削除します。

*file_spec*

属性を変更（修正、削除など）するファイルまたはディレクトリを指定します。

-f|-force

-type オプションは -force オプションを一緒に使用する必要があります。

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ます（-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ます。
- 同じ名前を持つ属性が存在するがタイプが異なる場合、操作は失敗します。

`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。

*from_file_spec to_file_spec*

`-cp` とともに使用した場合、このオプションは属性のコピー元のファイルが *from_file_spec* であり、属性のコピー先のファイルが *to_file_spec* であることを指定します。

`-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。

`-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、その属性でエディタが起動します。

`-p|-project` *from_project_spec to_project_spec*

`-cp` とともに使用した場合、このオプションは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。`-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。

`-p|-project` *project_spec*

属性を変更（修正、コピーなど）する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。

`-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`

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。

`-t|-type` *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` *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
  ```

- `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
  ```

- `foo.c` の `release` 属性を `4.2_int` に変更する。

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

- 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
  Copying attrs: version
  from: attr_test-1.
      to: attr_test2-1:project:1.
      to: attr_test3-1:project:1.
      to: attr_test4-1:project:1.
  Attribute Copy completed with 0 errors
  ```

## **baseline** コマンド

### 表記

#### **2** つのベースラインの比較

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

#### ベースラインの作成またはプレビュー

```
ccm baseline -c|-create
          [-rehearse]
          -p|-project project_spec -p|-project [project_spec. . .] |
          -bl|-baseline baseline_spec [-baseline baseline_spec…] |
          -pg|-project_grouping project_grouping_spec [project_grouping_spec. . . ]
          [-r|-release release]
          [-purpose purpose_spec]
          [-d|-description "baseline_description"]
          [-subprojects|-no_subprojects|-all_subprojects]
          [-vt|-version_template version_template]
          [s|-state state_name]
          [-b|-build build_string]
          [baseline_name]
```

#### ベースラインの削除

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

#### ベースラインの一覧表示

```
ccm baseline -l|-list [-release release] [-purpose purpose_spec]
          [-f|-format "format_string"] [-ns|-no_sort] [-u|-un_numbered]
```

#### ベースラインの削除のマーク付け

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

### ベースラインの修正

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

### ベースラインの公開

```
ccm baseline -publish baseline_spec
```

### ベースラインのリリース

```
ccm baseline -rb|-release_baseline
           [-comment comment_string]
           baseline_spec
```

### 削除したベースラインのリストア

```
ccm baseline -undelete
           baseline_spec
```

### ベースラインの表示

```
ccm baseline -sh|-show
           i|info|information|
           proj|project|projects|
           obj|objs|objects|
           t|task|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_string"]
           [-ns|-no_sort]
           [-u|-un_numbered]
           baseline_spec
ccm baseline -sh|-show
           ((r|release)|(p|purpose)|(o|owner)|(desc|description)|(b|build))
           baseline_spec
```

## 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、各 `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、*prep* 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。不要なリビルドを発生させないようにするために、Prep 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、*prep* 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 *prep* 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。

prep 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 prep 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、*prep* 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。既存の文字列にアンダースコア、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 *prep* 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。可視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。

不可視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、不可視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不可視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。どのユーザーも、ベースラインを表示、比較または一覧表示できます。

## オプションと引数

`-all_subprojects`

このオプションを `-create` と一緒に使用して、全プロジェクト階層のベースラインへの追加を指定します。デフォルトの動作は `-subprojects` です。

`-baseline` *`baseline_spec`*

このオプションを `-create` と一緒に使用し、1 つまたは複数の *`baseline_specs`* を指定すると、指定した既存のベースラインに含まれる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` オプションは、どのサブプロジェクトを追加するかに影響します。

デフォルトでは、`-baseline` オプションを使用して、`-project` オプションを使用しない場合、サブプロジェクトは含まれません。ただし、`-project` と `-baseline` オプションを一緒に使用すると、`-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`*

*`baseline_name`* はベースラインに割り当てられる名前です。ベースラインを作成するとき、正式なオブジェクト バージョン名をベースラインに割り当てることができます。

`ccm baseline -create` または `-modify` コマンドを実行するとき、*`baseline_name`* を指定しない場合、ベースラインに一意の名前が自動的に割り当てられます。この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アンダースコア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。

*`baseline_spec`*

*`baseline_spec`* は、ベースライン名が使えるところで、*`baseline_name`* または選択セット参照フォームの使用を可能にします。全選択セット参照 `@` を使用できます。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。

ベースラインのリリース時に、先頭に DCM データベース ID（dbid）と DCM 区切り文字（たとえば、`J` が dbid、`#` が DCM 区切り文字ならば `J#`）を含む *`baseline_spec`* を指定できます。

-build *build_string*

このオプションを -create と一緒に使用した場合、作成時に新しいベースラインに対して *build_string* が使用されます。

-modify と一緒に使用すると、ベースライン上のビルド文字列を変更できます。ベースラインが *released*（リリース済み）状態になければ、ビルド マネージャであるユーザーはビルドを変更できます。リリース済みのベースラインは、*ccm_admin* ロールを持つユーザーのみが変更できます。

-cr|-change_request|-change_requests

このオプションを -show と一緒に使用すると、ベースラインに部分的または完全に含まれる変更依頼が表示されます。デフォルトのフォーマットは %displayname: %problem_synopsis です。

このオプションを -compare と一緒に使用すると、ベースライン比較操作により、2 つのベースライン間の変更依頼の差分が表示されます。たとえば、2 つのベースライン B1 と B2 を比較した場合、比較出力には、B1 と B2 の両方に完全に含まれる変更依頼、B1 と B2 の両方に部分的に含まれる変更依頼、一方のプロジェクトまたは他方のプロジェクトに完全にまたは部分的に含まれる変更依頼が含まれます。

-comment *string*

コメントを指定すると、そのコメントはすべてのベースライン プロジェクトとそれらのメンバーのコメントに追加されます。コメントの追加は、ベースラインのリリース時、およびこれらのプロジェクトとメンバーのチェックイン時に行われます。

-compare

このオプションを -baseline と一緒に使用すると、指定した *baseline_specs* を持つ 2 つのベースラインが比較されます。2 つのベースラインのプロパティ間の差分を表示し、同じプロジェクトの異なるバージョンを比較し、ベースラインに追加または削除されたプロジェクトを表示し、プロジェクト間の変更依頼の相違点を表示します。また、2 つのベースラインのタスクの差分も表示されます。

-create

新しいベースラインを作成します。指定したベースライン名がデータベースがローカルであるベースラインによってすでに使用されていた場合、このコマンドは失敗します。また、無効なリリースまたは非アクティブ リリースを指定した場合も、このコマンドは失敗します。

作成するベースラインには、*project_spec* によって指定される階層内のすべての静的プロジェクトが含まれます。これにはすべての静的プロジェクトと *prep* プロジェクトのコピーが含まれます。階層内のプロジェクトが指定したリリースまたは目的と一致しない場合でも、このコマンドは成功しますが、警告メッセージが表示されます。選択セットの参照形式 または完全な選択セット（@）を、*project_spec* として使用できます。*project_spec* の詳細については、プロジェクトの指定を参照してください。

作成するベースラインに *prep* 状態のプロジェクトを追加できます。

指定するプロジェクト階層には修正可能な製品を含むことができます。他の修正可能なメンバーが製品でもプロジェクトでもない場合、ベースライン作成操作は失敗します。リリースと目的を指定しなかった場合、最初に指定したプロジェクトのリリースと目的が使用されます。

`-delete`

指定した *baseline_spec* を持つベースラインを削除します。この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。`-wp` を指定した場合、ベースラインと一緒にプロジェクトと製品が削除されます。`-np` を指定した場合、ベースラインのみが削除されます。デフォルトでは、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が、ベースラインとともに削除されます。

非静的プロジェクトがベースラインを使用している場合、またはプロセス ルールがベースラインを使用している場合、そのベースラインを削除できません。また、関連付けられている 1 つまたは複数のプロジェクトまたは製品がベースラインに含まれないプロジェクトのメンバーであった場合、ベースラインおよびそのチェックイン済みプロジェクトと製品を削除しようとすると、ベースラインは削除されますが、プロジェクトまたは製品は削除されません。また、ベースラインに含まれる 1 つまたは複数のプロジェクトが別のベースラインのメンバーでもある場合、ベースラインは削除されますが、プロジェクトは削除されません。

`-delete` オプションは、`@[0-9]+` などの単一項目の選択セット参照または全選択セット参照を含む複数のベースラインに対して機能します。

`-description "`*baseline_description*`"`

ベースラインの詳細な説明を提供します（オプション）。説明の長さと内容には制限はありません。*baseline_description* に 1 つまたは複数の空白が含まれる場合は、二重引用符で囲む必要があります。

-f|-format "*format_string*"

コマンドの出力フォーマットを指定します。デフォルトのフォーマットは、-format と一緒に使用するオプション（たとえば、-list または -show）とこれらのオプションのキーワード引数によって異なります。デフォルトの出力フォーマットの詳細については、-format と一緒に使用できるオプションの説明を参照してください。

このフォーマットには、テキストとキーワードの組み合わせを含むことができます。キーワードは、各オブジェクトについての特定のデータに置き換わります。たとえば、ユーザー sue が所有するオブジェクトに関する情報が表示される場合、キーワード %owner は sue に置き換わります。

-list

ベースラインを一覧表示します。-release または -purpose を指定した場合、リリースまたは目的に一致するベースラインのみが表示されます。

-mfd|mark_for_deletion

削除するベースラインにマークを付けます。削除のマークが付けられたベースラインは、*ccm_admin* ロールを持つユーザーが、後でオフライン保存と削除（SOAD）操作または手動で削除できます。*ccm_admin* ロールのユーザーが削除する場合、ベースラインは必ずしも *deleted_baseline* の状態になくてもかまいません。

-modify

ベースラインの各種の属性を修正します。更新テンプレートを更新する場合、実際の更新は名前とビルドの属性が修正された後に行われます。名前とビルドの修正後はテンプレートが異なった値に展開される可能性があるからです。ベースラインを変更するには、ビルド マネージャである必要があります。

name キーワードと build キーワードも指定した場合、テンプレート内のこれらのキーワードは新しく指定された名前とビルドの値に展開されます。

ワークエリアが可視状態であっても、適切なファイルアクセス権限の欠如やディスク領域不足などの他の理由で更新できない場合、-skip_nonvisible_projects オプションの設定に関係なく操作は失敗します。ただし、失敗した場合でも、操作は継続され、最終的にワークエリアを更新するためにすべての失敗を報告します。

-name

-modify と一緒に使用すると、ベースラインの名前を変更できます。ベースラインが *released*（リリース済み）状態になければ、ビルド マネージャであるユーザーはベー

スラインの名前を変更できます。リリース済みのベースラインは、*ccm_admin* ロールを持つユーザーのみが変更できます。

`-np|-no_projects_and_products`

このオプションを `-delete` と一緒に使用すると、ベースラインは削除されますが、ベースラインに関連付けられているプロジェクトとプロジェクト内の製品は削除されません。

`-no_subprojects`

`-create` と一緒に使用して、サブプロジェクトをベースラインに含まないように指定します。

`-ns|-no_sort`

コマンドの出力をソートしないように指定します。

`-objects`

このオプションを `-compare` と一緒に使用すると、ベースライン比較操作によって、共通のオブジェクト、または 2 つのベースライン間で追加または削除されたオブジェクトが表示されます。

`-project`

このオプションを `-create` と一緒に使用し、また *`project_spec`* を指定して、プロジェクトをベースラインに追加するように指示します。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。これを無効にするためには、`-no_subprojects` オプションを使用します。

`-projects`

このオプションを `-compare` と一緒に使用すると、共通のプロジェクトと各ベースライン固有のプロジェクトが一覧表示されます。

`-project_grouping` *`project_grouping_spec`*

このオプションを `-create` と一緒に使用し、1 つまたは複数の *`project_grouping_specs`* を指定すると、指定したプロジェクト グルーピング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。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。`-all_subprojects` オプションを使用すれば、これを無効にすることができます。

`-subprojects`、`-no_subprojects` および `-all subprojects` は、どのサブプロジェクトを追加するかに影響します。

デフォルトでは、`-project_grouping` オプションを使用して、`-project` オプションを使用しない場合、サブプロジェクトは含まれません。ただし、`-project` と `-project_grouping` オプションを一緒に使用すると、`-project` が暗黙的に指示する `-subprojects` デフォルトが、`-project_grouping` が暗黙的に指示する `-no_subprojects` デフォルトに優先します。

*`project_spec`*

指定する各 *`project_spec`* は、ベースラインに含めるプロジェクトを示します。*`project_specs`* の詳細については、プロジェクトの指定を参照してください。

`-publish` *`baseline_spec`*

`test_baseline` 状態のベースラインを `published_baseline` 状態に遷移させます。この遷移を完了するには、ビルド マネージャである必要があります。

`-purpose` *`purpose_spec`*

ベースラインの目的を指定します。これを指定しない場合、最初に指定したプロジェクトに指定された目的となります。

`-rehearse`

ベースラインを構成するプロジェクトと製品を一覧表示します。

ベースライン作成時または `-rehearse` オプション使用時に、バージョンのコンフリクトが見つかった場合、既存のバージョンがすでに存在する理由、または生成されるバージョンが正当なバージョン文字列でない理由に従って警告が表示され、コンフリクトのあるすべての製品とプロジェクト バージョンが一覧表示されます。

`-rb|-release_baseline`

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

`-release` *release*

ベースラインのリリース値を指定します。ベースラインを作成するとき、任意のアクティブ リリース値を使用できます。これを指定しない場合、最初に指定したプロジェクトに指定されたリリースとなります。

`-show`

指定した *baseline_name* に関連付けられているリリース、目的およびプロジェクト情報を表示します。また、ベースライン内の各プロジェクトのリリースと目的も表示します。

`i|info|information`

これを指定した場合、以下の情報が表示されます。

名前
詳細
リリース
目的
リリース
プロジェクト
ビルド

`r|release`

ベースラインのリリース値を表示します。

`p|purpose`

ベースラインの目的を表示します。

`o|owner`

ベースラインの所有者の名前を表示します。

`desc|description`

ベースラインの詳細を表示します。

`projects`

ベースラインに含まれているすべてのプロジェクトを表示します。デフォルトのフォーマットは以下のとおりです。

```
%displayname %status %owner %release %create_time
```

`-format` オプションを使用すると、デフォルトのフォーマットが置き換わります。

`objects`

ベースラインに含まれているすべてのオブジェクトを表示します。デフォルトのフォーマットは以下のとおりです。

```
    %displayname %status %owner %release %create_time
```

`-format` オプションを使用すると、デフォルトのフォーマットが置き換わります。

`tasks`

ベースラインに含まれているすべてのタスクを表示します。デフォルトのフォーマットは以下のとおりです。

```
    %displayname %release %owner %create_time
```

`-format` オプションを使用すると、デフォルトのフォーマットが置き換わります。

`cr|change_requests`

ベースラインに含まれているすべての変更依頼を表示します。

`fcr|fully_included_change_request|fully_included_change_requests`

完全に含まれる変更依頼のみを表示します。

`pcr|partially_included_change_request|partially_included_change_requests`

部分的に含まれる変更依頼のみを表示します。

`-skip_nonvisible_projects`

このオプションを `-modify` と一緒に使用して、可視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に指定します。

不可視の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` を使用している場合、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、ワークエリアが不可視だった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。

`-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。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。SQA 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`

このオプションを `-create` と一緒に使用して、ベースラインにプロジェクト階層を追加するように指定します。これには、すべての *prep* サブプロジェクトが含まれます。また、修正不可サブプロジェクトのリリースのコンポーネント部分がベースラインのリリースに一致する場合、修正不可サブプロジェクトが含まれます。このためには、コンポーネント名が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できます。オプションの指定がない場合、これはデフォルトの動作となります。

`-tasks`

このオプションを `-compare` と一緒に使用すると、各ベースラインに共通のタスクと特有のタスクが一覧表示されます。

`-u|-un_numbered`

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）。

`-undelete`

*deleted_baseline* 状態のベースラインを、削除前の状態に復帰させます。ベースラインが *deleted_baseline* 状態にない場合、何も変化しません。

`-ver|-versions`

プロジェクトと製品のバージョンを変更するように指定します。

*`version_template`*

*`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 属性または組み込まれた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まれた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。

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

`-create` と一緒に使用すると、コマンド実行時にチェックインされるすべての新しいプロジェクトと製品のバージョンについて、そのバージョンの *`version_template`* が使用されます。

`-modify` と一緒に使用すると、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/6.3 SP3 ベースラインが CM/6.3 SP2 ベースラインの 20 の既存の静的プロジェクトと、CM/6.3 SP3 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、prep 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、アンダースコア（_）と 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。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。ワークエリアが不可視状態の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。ワークエリアが可視状態であっても、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。

-wp|-with_projects_and_products

このオプションを -delete と一緒に使用すると、ベースラインに関連付けられているプロジェクトとプロジェクト内の製品が削除されます。

## 例

- リリース 2.2 、目的 Integration Testing（統合テスト）の場合のベースラインのリストを表示する。

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

- 20020401_1 というベースラインにあるプロジェクトと、20020401_2 というベースラインにあるプロジェクトを比較する。

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

- Release 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。このベースラインには、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

- update_properties コマンド
- process_rule コマンド

# **bom** コマンド

## 表記

```
ccm bom file_spec [file_spec...]
```

## 説明と用途

`bom` コマンドにより、1 つまたは複数の指定したオブジェクトの部品構成表（BOM）を標準出力に表示できます。

また、製品オブジェクトのプロパティ ダイアログからも BOM を表示できます。

どのユーザーでもこのコマンドを実行できます。

## オプションと引数

*`file_spec`*

BOM の表示対象となるファイルの名前を指定します。*`file_spec`* は管理製品である必要があります。

## 例

- BOM を表示する。

  ```
  ccm bom file_spec
  ```

# candidates コマンド

## 表記

```
ccm cand|candidates [-recommend] file_spec
```

## 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、そのオブジェクトは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、所有者、それが作成されたプロジェクト、関連付けられているタスク番号が表示されます。

## オプションと引数

*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。

-recommend

一覧表示されているバージョンのうち、Rational Synergy が選択ルールに基づいて選択するバージョンにアスタリスク（*）を付加します。

## 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 chrisb incl projX 1 5
2) Xincls.h-2 integrate chrisb incl projX 1 12
3) Xincls.h-3 integrate terri  incl projX 1 13
4) Xincls.h-4 integrate terri  incl projX 1 15 *
```

関連トピック

- update コマンド
- use コマンド

# **cat** コマンド

## 表記

```
ccm cat file_spec [file_spec...]
```

## 説明と用途

cat コマンドにより、オブジェクトのソースを表示します。このコマンドは、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。

## オプションと引数

*file_spec*

表示するファイルの名前を指定します。

## 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します。

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

関連トピック

- view コマンド
- type コマンド

# **change_type** コマンド

## 表記

```
ccm change_type file_spec -t|-type new_type -task task_number
```

## 説明と用途

特定のオブジェクトのタイプを変更します。指定したタイプのオブジェクトの新しいバージョンが作成されます。指定したオブジェクトが *working*（作業中）状態にある場合、そのオブジェクトは新しいオブジェクトに置き換わり、指定したオブジェクトはデータベースから削除されます。　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、プロジェクト内で旧オブジェクトが新しいオブジェクトに置き換わります。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が実行されない場合、コマンドは失敗します。

どのユーザーでもこの操作を実行できます。

## オプションと引数

`-type new_type`

オブジェクトの新しいタイプを指定します。

*`file_spec`*

タイプを変更するファイルの名前を指定します。

`-task task_number`

新しく作成したディレクトリとオブジェクトを、指定したタスクに関連付けます。

カレント（デフォルト）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。

## 例

- ファイルのタイプを変更する。

  ```
  ccm change_type file_spec -t|-type new_type
  ```

# **checkin** コマンド

## 表記

```
ccm ci|checkin [-s|-state state]
               [-nc|-nocomment] [-cr|-commentreplace]
               [-c|-comment "string"]
               [-ce|-commentedit]
               [-cf|-commentfile file_path]
               [file_spec [file_spec...]
ccm ci|checkin [-source] [-products] [-projects]
               [-h|-hierarchy] [-task task_number]
               [-s|-state state]
               [-ps|-product_state product_state]
               [-ss|-source_state source_state]
               [-nc|-nocomment] [-cr|-commentreplace]
               [-c|-comment "string"]
               [-ce|-commentedit]
               [-cf|-commentfile file_path]
               -p|-project [project_spec project_spec...]
```

## 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可視にして行う必要があります。

## オプションと引数

**-c|-comment "*string*"**

オブジェクトのコメントとして *string* を追加します。タスクをチェックインする場合は、タスクにコメントが追加されます。

**-ce|-commentedit**

コメントを入力するためにエディタを起動します。

`-cf|-commentfile` *`file_path`*

指定したファイルからのコメントを使用します。コメント文字列とコメント ファイルの両方を指定した場合はコメントがマージされ、ファイルのコメントの後ろにコメント文字列が付加されます。

`-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。

*`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。

`-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ません （通常、`-c` または `-cf` オプションを使用してコメントを指定しなかった場合、オブジェクトに何もコメントがなければ、コメントを入れるかどうか尋ねられます)。

`-products`

現在のプロジェクトのすべての製品をチェックインします。

`-p|-project` *`project_spec`* [*`project_spec`*`...`]

指定したオブジェクトがプロジェクトであることを示します。

`-projects`

最上位プロジェクト下のすべてのプロジェクトをチェックインし、次に最上位プロジェクトをチェックインします。

`-ps|-product_state` *`product_state`*

このオプションは、`-products` オプションと一緒に使用します。

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`*

チェックインするファイルまたはプロジェクトの状態を明示的に設定します。状態を指定しないと、デフォルトの次の状態が自動的に算出されます。

`-products` オプションと `-source` オプションを指定し、`-p` オプションと `-ps` オプションを指定しなかった場合、製品オブジェクトとソース オブジェクトの次の状態としてこのオプション（`-state` *`state`*）が使用されます。

`-source`

現在のプロジェクトにある、製品またはプロジェクトではないすべてのオブジェクトをチェックインします。

`-ss|-source_state` *`source_state`*

プロジェクト階層のチェックイン時の、非製品オブジェクトの状態を指定します。

## 例

- 現在のバージョンの `foo.c` を、*visible*（可視）状態でチェックインする。

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

- 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
  ```

- 3 つのファイル（`clear.c`、`concat.c`、`display.c`）をチェックインする。

  ```
  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
  ```

- `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
```

## 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。

関連トピック

- checkout コマンド
- task コマンド

# **checkout** コマンド

## 表記

```
ccm co|checkout [-task task_number]
                [-t|-to file_spec|version]
                [-c|-comment "string"]
                [-ce|-commentedit]
                [-cf|-commentfile file_path]
                file_spec [file_spec...]
ccm co|checkout [-purpose purpose_spec]
                [-platform platform]
                [-release release|as_is|none]
                [ÅwIBM Rational Synergyks|os]
                [-subprojects]
                [-versions "old_ver:new_ver,old_ver:new_ver,..."]
                [-t|-to version]
                [-c|-comment "string"]
                [-ce|-commentedit]
                [-cf|-commentfile file_path]
                -p|-project project_spec
```

プロジェクトのチェックアウトとワークエリア プロパティの設定

```
ccm co|checkout check_out_options
                [-cb|-copy_based|-not_copy_based|-ncb  (UNIX only)]
                [-rel|-relative|-nrel|-not_relative]
                [-path|-set|-setpath] absolute_path
                [-mod|modifiable_wa] [-nmod|not_modifiable_wa]
                [-tl|-translate|-ntl|-no_translate]
                [-wa|-maintain_wa|-nwa|-no_wa]
                [-wa_t|-wa_time|-nwat|-no_wa_time]
                [-u|update|-no_u|-no_update]
                -p|-project project_spec
```

## 説明と用途

checkout コマンドにより、オブジェクトをチェックアウトし、copy_project コマンド を使用してプロジェクトをチェックアウトします。checkout -project 操作は、現バージョンの Rational Synergy では copy_project 操作と呼ばれています。

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。

-p オプションを指定すると、指定したプロジェクト（またはプロジェクト階層全体）がチェックアウトされます。copy_project コマンドの機能は、-p オプションを持つ checkout コマンドと同じです。

静的（修正不可状態）プロジェクトからチェックアウトして、サブプロジェクトをチェックアウトしなかった場合、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、チェックアウトす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チェックアウトすると、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。

checkout コマンドは、以下の 2 つの方法で使用します。

## ファイルまたはディレクトリの修正可能バージョンの作成

オブジェクトをチェックアウトするとき、その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 （Windows）または ccm ls （UNIX）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。

ワークエリア内のオブジェクト バージョンをチェックアウトする場合は、以下の形式を使用します。

- プロジェクト参照形式

  ```
  sub_proj¥foo.c@my_proj-1（Windows）
  sub_proj/foo.c@my_proj-1 （UNIX）
  ```

- 選択参照形式

  ```
  @1
  ```

- オブジェクト参照形式

  ```
  foo.c-1:csrc:1
  ```

- ワークエリア参照形式

  ```
  c:¥users¥tom¥ccm_wa¥main-1¥main¥src¥foo.c（Windows）
  /users/tom/ccm_wa/main-1/main/src/foo.c（UNIX）
  ```

ワークエリア外から、使用するオブジェクトに対してのみプロジェクト参照形式を使用できます。

```
sub_proj¥foo.c@main-1（Windows）
sub_proj/foo.c@main-1（UNIX）
```

オブジェクトがどこからも使用されていない場合（すなわち、フローティング オブジェクトの場合）、ワークエリア外からは、次のいずれかの形式を使用する必要があります。

- 選択参照形式

  ```
  @1
  ```

- オブジェクト参照形式

  ```
  foo.c-1:csrc:1
  ```

### 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の作成

デフォルトでは、プロジェクトをチェックアウトすると、そのプロジェクトは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。プロジェクトをチェックアウトするとき、ワークエリア プロパティを設定できます。

## オプションと引数

`-c|-comment "`*`string`*`"`

コメント文字列を指定します。

`-cb|-copy_based (UNIX only)`

ワークエリアをコピーベースにします。このオプションは `-p` オプションと一緒にのみ使用できます。

詳細については、work_area コマンドを参照してください。

`-ce|-commentedit`

コメントを入力するために、デフォルトのエディタを起動します。

`-cf|-commentfile` *`file_path`*

指定したファイルからのコメントを使用します。コメント文字列とコメント ファイルの両方を指定した場合はコメントがマージされ、ファイルのコメントの後ろにコメント文字列が付加されます。

*file_spec*

チェックアウトするファイルまたはディレクトリの名前を指定します。

-mod|-modifiable_wa

ワークエリアを修正可能に指定します。

-nmod|-not_modifiable_wa

ワークエリアを修正不可に指定します。

-no_u|-no_update

プロジェクトのコピー時にプロジェクトを更新しないように指定します。

-not_copy_based|-ncb (UNIX only)

ワークエリアをリンクベースにします。このオプションは -p オプションと一緒にのみ使用できます。

詳細については、work_area コマンドを参照してください。

-ntl|-no_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 ファイルを変換しないように指示します。このオプションは -p オプションと一緒にのみ使用できます。

詳細については、work_area コマンドを参照してください。

-nrel|-not_relative

Windows 上で、サブプロジェクトのワークエリアを、親プロジェクトのワークエリアの相対ワークエリアではなく、絶対ワークエリアにします。プロジェクトを初めて作成するとき、これがデフォルトとなります。このオプションは -p オプションと一緒にのみ使用できます。

UNIX 上では、サブプロジェクトに対してリンクを使用できるようにして、サブプロジェクトのワークエリアを、親プロジェクトのワークエリアの相対ワークエリアではなく、絶対ワークエリアにします。プロジェクトを初めて作成するとき、これがデフォルトとなります。このオプションは -p オプションと一緒にのみ使用できます。

詳細については、work_area コマンドを参照してください。

`-nwa|-no_wa`

ワークエリアを管理しないようにします（すなわち、ワークエリアをデータベースから切り離します)。このオプションは `-p` オプションと一緒にのみ使用できます。

詳細については、work_area コマンドを参照してください。

`-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。このオプションは `-p` オプションと一緒にのみ使用できます。

詳細については、work_area コマンドを参照してください。

`-platform` *`platform`*

チェックアウトするプロジェクトまたはプロジェクト階層に対して設定するプラットフォーム値を指定します。プラットフォームの選択肢は、Rational Synergy 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）に記載されます。

デフォルトでは、階層内の各プロジェクトの現在のプラットフォーム値が新しいバージョンにコピーされます。このオプションは `-p` オプションと一緒にのみ使用できます。

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

プロジェクトをチェックアウトするか、あるいは、`-subprojects` オプションと組み合わせて指定した場合に、プロジェクト階層全体をチェックアウトします。

`-purpose` *`purpose_spec`*

指定したプロジェクトに関連付ける目的を指定します。

目的の選択肢はプロジェクト目的テーブルに記載され、データベースで使用されるすべての目的値を含みます。

開発者としてチェックアウトを実行する場合、このオプションのデフォルトは `Insulated Development`（個別開発）ですが、代わりに `Shared`（共有）を指定できます。ビルド マネージャであるか *ccm_admin* ロールを持っている場合は、デフォル

ト設定は Integration Testing（統合テスト）です。このオプションは -p オプションと一緒にのみ使用できます。

ÅwIBM Rational Synergyks|os

チェックアウトするプロジェクトまたはプロジェクト階層を、ベースラインとタスク（tasks）を使用するか、あるいは候補オブジェクトの状態（os）を使用して更新するどうかを指定します （候補オブジェクトの状態を使用する方法は、Synergy 4.2 より以前のリリースで使用される唯一の方法です）。このオプションは -p オプションと一緒にのみ使用できます。

-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。このオプションは、プロジェクトを 1 つの場所で使用する場合のみ設定できます。設定後、このプロジェクトは、他のプロジェクトのサブプロジェクトとしては使用でなくなります。このオプションは -p オプションと一緒にのみ使用できます。

-release *release*

チェックアウトするプロジェクトまたはプロジェクト階層に対して設定するリリース値を指定します。リリース選択肢には、データベースで使用されるリリース値のすべて、as is、または none があります。

デフォルトでは、階層内の各プロジェクトの現在のリリース値が新しいバージョンにコピーされます。このオプションは -p オプションと一緒にのみ使用できます。

-subprojects

指定したプロジェクトの階層内にあるすべてのサブプロジェクトをチェックアウトします。このオプションは -p オプションと一緒にのみ使用できます。

-t|-to *file_spec*|*version*

バージョンの指定、新しい非プロジェクト オブジェクトの名前の変更、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の指定を行います。

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 バージョンは以下のようになります。

```
foo.c-bar
```

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 バージョンは以下のようになります。

```
bar.c-1
```

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_number`*

チェックアウトする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（デフォルト）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。

`-tl|-translate`

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に指示します。このオプションは `-p` オプションと一緒にのみ使用できます。

詳細については、work_area コマンドを参照してください。

`-u|-update`

プロジェクトのコピー時にプロジェクトを更新するように指定します。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を順守し、プロジェクトをワークエリアなしでチェックアウト後に更新します。次に、プロジェクトの同期を取ります。

`-versions "old_ver:new_ver,old_ver:new_ver,..."`

このオプションは `-p` オプションと一緒にのみ使用できます。

Rational Synergy に組み込まれたアルゴリズムを使用して、デフォルトの次のバージョンを計算します （ほとんどの場合、現在のバージョンに「1」が足されます)。次のバージョンを変更する場合は、上記の構文を使用して、旧バージョンを新しいバージョンに対応付けます。

プロジェクト階層をチェックアウトする場合、各対応付けが、現在値を持っている階層内の全プロジェクトに適用されます。*new_ver* が `NoCheckOut` の場合、*old_ver* に関連付けられているプロジェクトはチェックアウトされません。

チェックアウトするすべての新しいプロジェクトに同じバージョンを指定する場合は、`-to` オプションを使用します。`-to` オプションと `-version` オプションは一緒には使用できません。

`-wa|-maintain_wa`

ワークエリアを管理します（ワークエリアの同期を取り、同期の状態を維持します)。このオプションは `-p` オプションと一緒にのみ使用できます。

詳細については、work_area コマンドを参照してください。

`-wat|-wa_time`

Rational Synergy の変更時刻ではなく、ファイルがワークエリアにコピーされた時刻を示すように、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。このオプションは `-p` オプションと一緒にのみ使用できます。

詳細については、work_area コマンドを参照してください。

## 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¥bob¥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 *`object_version(s)`* にタスクを関連付ける。

  ```
  ccm co -c "comment string" -task task_number object_name1 object_name2
  ```

- 既存のプロジェクト階層から新しい *working*（作業中）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 *prep* 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 sqa -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 sqa -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

# **checkpoint** コマンド

## 表記

```
ccm ckpt|checkpoint [-t|-to version] [-cr|-commentreplace]
                    [-c|-comment "string"]
                    [-ce|-commentedit]
                    [-cf|-commentfile file_path]
                    [-task task_id]
                    file_spec [file_spec...]
ccm ckpt|checkpoint [-t|-to version][-cr|-commentreplace]
                    [-c|-comment "string"]
                    [-ce|-commentedit]
                    [-cf|-commentfile file_path
                    [-task task_id]
                    -p|-project [project_spec...]
```

## 前提条件

`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。作業中オブジェクトのみをチェックポイント処理できます。

## 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。

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。

## オプションと引数

`-c|-comment "`*`string`*`"`

コメント文字列を指定します。

`-ce|-commentedit`

コメントを入力するためにエディタを起動します。

`-cf|-commentfile` *`file_path`*

指定したファイルからのコメントを使用します。コメント文字列とコメント ファイルの両方を指定した場合はコメントがマージされ、ファイルからコメントの後ろにコメント文字列が付加されます。

`-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の名前を指定します。

`-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。

`-task` *`task_id`*

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。

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。

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

プロジェクトをチェックポイント処理します。

## 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 Jane’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

- collapse コマンド

# **clean_cache** コマンド

## 表記

```
ccm clean_cache [-t|-type type] [-s|-status status]
                [-c|-cutoff_time time] [-u|-used]
                [-v|-verbose]
```

## 説明と用途

デフォルトでは、clean_cache コマンドは、14 日以上経過してワークエリアで使用されていない、変更不可オブジェクトのアーカイブ済みキャッシュ ファイルを削除します。

アーカイブされていないオブジェクト バージョンのキャッシュ ファイルは削除されません。

*ccm_admin* ロールを持つユーザーのみがこのコマンドを使用できます。

Windows では、キャッシュ ファイルの削除の詳細については、『IBM Rational Synergy 管理者ガイド Windows 版』の「キャッシュ ファイルの削除」を参照してください。

UNIX では、キャッシュ ファイルの削除の詳細については、『IBM Rational Synergy 管理者ガイド UNIX 版』の「キャッシュ ファイルの削除」を参照してください。

## オプションと引数

`-c|-cutoff_time` *`time`*

*`time`* 以上経過したソースを持つオブジェクト バージョンに対してのみキャッシュ ファイルを削除するように指定します。ここで使用される *time* は、変更時刻ではなく、アクセス時刻です。デフォルトでは、14 日以上経過したファイルを削除します (`-c "-14:0:0:0"`)。

`-s|-status` *`status`*

キャッシュ ファイルを削除するオブジェクト バージョンの状態を指定します。デフォルトでは、変更不可状態にあるバージョンが選択されます。

`-t|-type` *`type`*

キャッシュ ファイルを削除するオブジェクト バージョンのタイプを指定します。デフォルトでは、すべてのタイプのキャッシュ ファイルが削除されます。

`-u|-used`

ワークエリアを持つか持たないかに関係なく、プロジェクトで使用されているキャッシュファイルを削除します。デフォルトでは、ワークエリアを持つプロジェク

トで使用されるキャッシュ ファイルは削除されません。UNIX では、プロジェクトのワークエリアには、存在しなくなったファイルへのシンボリック リンクが残ります。

注記：このオプションは、その使用による影響を理解していない場合は使用しないでください。

## 例

- 45 日以上経過した `csrc` キャッシュ ファイルを削除する。

```
ccm clean_cache -type csrc -cutoff_time "-45:0:0:0"
```

# **clean_up** コマンド

## 表記

```
ccm clean_up -all|-task|-rpt [-used]
             [-q|-quiet | -v|-verbose]
             [-rel|-release release]
             [-user user_name]
```

## 説明と用途

clean_up コマンドにより、使用されていない自動タスクを削除するか、または Rational Synergy データベースからプロセス ルールを削除します。

プロセス ルールを変更するには、*ccm_admin* ロールを持っている必要があります。PT アドミニストレータとして作業しているときにプロセス ルールを削除しようとすると、プロセス ルールが削除されないことを通知するメッセージが表示されます。

また、どのユーザーでも自分の自動タスクは削除できますが、他のユーザーが所有する自動タスクを削除する場合は、*ccm_admin* ロールを持っているか、PT アドミニストレータである必要があります。

## オプションと引数

**-all**

このオプションは、-task オプションと -rpt オプションの両方を指定することと同じです。このオプションにより、未使用の自動タスクと未使用のプロセス ルールの両方が削除されます。

**-q|-quiet**

出力メッセージを最小限に抑えます。

**-rel|-release *release***

指定したリリースのみの自動タスクまたはプロセス ルールを削除します。

**-rpt**

未使用のプロセス ルールを削除します。

プロセス ルールは、プロジェクトがそのプロセス ルールを使って更新プロパティを設定しない場合は、未使用の状態にあります。

`-task`

未使用の自動タスクを削除します。

> 注記：*ccm_admin* ロールを持っており、このコマンドを `-user` オプションなしで実行した場合、データベース内のすべての未使用タスクが削除されます。

自動タスクがプロジェクトの更新プロパティに含まれていない場合、かつ、自動タスクに関連付けられているオブジェクトがない場合、自動タスクは未使用状態にあるといえます。

CM アドミニストレータまたは PT アドミニストレータとして作業していない場合は、`-task` は `-task -user` *your_user_name* と同じです。

`-used`

未使用のプロセス ルールだけではなく、使用中のプロセス ルールも削除します。Rational Synergy は、使用中のプロセス ルールのプロジェクトについて、更新プロパティを「手動」に設定します。

このオプションは、`-rpt` オプションと一緒に使用した場合にのみ有効です。

`-user` *`user_name`*

*user_name* が所有する自動タスクを削除します。このオプションは `-rpt` オプションと一緒には使用できません。

`-v|-verbose`

出力メッセージを最大にします。

## 例

- すべての未使用のプロセス ルールを削除する。

  ```
  ccm clean_up -rpt
  ```

- 使用中と未使用の両方のプロセス ルールを削除する。

  ```
  ccm clean_up -rpt -used
  ```

- リリース 5.0 の使用中と未使用の両方のプロセス ルールを削除し、コマンドを詳細モードで実行する。

```
ccm clean_up -rpt -used -rel 5.0 -v
```

- 別のユーザー *user_name* に属するすべての自動タスクを削除する。

  ```
  ccm clean_up -task -user user_name
  ```

# **collapse** コマンド

## 表記

```
ccm collapse [-from file_spec] file_spec [file_spec...]
ccm collapse -all file_spec [file_spec...]
```

## 説明と用途

collapse コマンドにより、データベースからオブジェクト バージョンを削除した上で履歴のリンクを張り直せます。削除するオブジェクトに後継オブジェクトがあるときに、このコマンドを使用します。

コラプスしたオブジェクトはすべてデータベースから削除されます。プロジェクトをコラプスすると、そのメンバーは使用されない状態になり、プロジェクトから削除されます。

> 注記：*working*（作業中）バージョンを修正不可状態にチェックインする前に、データベースに入れたくない前のチェックポイント バージョンを確実にコラプスします。ここでコラプスしないと、チェックポイント バージョンの削除には *ccm_admin* ロールが必要になります。
>
> 先行バージョンをコラプスするためには、チェックポイント バージョンの直後のバージョンが書き込み可能である必要があります。たとえば、bufcolor.c という名前のオブジェクトがあるとします。バージョン履歴は、1 --> 2 --> 3 --> 4、バージョン 1 は *integrate*（統合）状態、バージョン 2 と 3 はチェックポイント バージョン、バージョン 4 は *working*（作業中）状態です。バージョン 4 が *working*（作業中）状態にあるので、バージョン 3 のコラプスは可能です。バージョン 4 が *integrate*（統合）状態の場合は、バージョン 3 をコラプスできません。

オブジェクトが以下のいずれかの特性を持つ場合を除き、指定したオブジェクトに対して collapse コマンドを使用できます。

- オブジェクトが修正不能であり、ユーザーが *ccm_admin* ロールとして作業していない。
- オブジェクトがプロジェクトのメンバーである。

> 注記：複数の後継バージョンと先行バージョンを持つバージョンをコラプスすると、先行バージョンと後継バージョンのそれぞれがリンクされます。先行バージョンと複数の

後継バージョンを持たないバージョンをコラプスすると、
後継バージョンの履歴のリンクが削除されます。

すべてのユーザーは、書き込み権限を持つオブジェクトをコラプスできます。

*ccm_admin* ロールを持つユーザーは、修正不可オブジェクトもコラプスできます。

ccm_admin ロールを持つユーザーが `collapse -all` コマンドを実行すると、簡略化した履歴が作成されます。通常、この履歴には、プロジェクトのメンバーであるオブジェクトバージョンのみが含まれます。中間バージョンは、その状態に関係なくすべて削除されます。

## オプションと引数

`-all`

*`file_spec`* によって指定されたファイルまたはディレクトリのすべてのバージョン（すなわち、全履歴）をコラプスすることを指示します。

注意！ 作業中のオブジェクトでも、プロジェクト メンバーでない場合はコラプスされます。`-all` オプションの使用には十分な注意が必要です。

*`file_spec`*

コラプスする範囲に含まれる最初のオブジェクトを指定します。

`-from`

範囲に含まれる最後のオブジェクトを指定します。コラプスは、範囲内の最初のオブジェクトから最後のオブジェクトまでを対象として実行されます （以下の最初の例を参照してください)。

## 例

`ico_616-1` プロジェクトの `bufcolor.c` のバージョン履歴は、1 --> 2 --> 3 --> 4 --> 5 です。バージョン 1 は *integrate*（統合）状態、バージョン 2、3、4 は *checkpoint*（チェックポイント）状態、バージョン 4 は *working*（作業中）状態です。

- `bufcolor.c-4` をコラプスする。

```
ccm collapse bufcolor.c-4 -from bufcolor.c-5
Starting Collapse Process...
Unable to remove bufcolor.c-5:csrc:1 : it is a member of a project.
bufcolor.c-4:csrc:1 removed.
Collapse complete with 1 success and 1 failure.
```

バージョン 5 が削除されなかったことに注目してください。バージョン 5 はプロジェクトのメンバーなので削除できません。bufcolor.c の新しいバージョン履歴は 1 --> 2--> 3 --> 5 となり、バージョン 5 は working（作業中）バージョンのままです。

- bufcolor.c-3 のみをコラプスする。

  ```
  ccm collapse bufcolor.c-3
  bufcolor.c-3:csrc:1 removed.
  ```

  ico_616-1 プロジェクトの bufcolor.c オブジェクトのバージョン履歴は 1 --> 2 --> 5 となります。

  bufcolor.c に変更を加える必要があり、チェックアウトし、その後このオブジェクトのいくつかのバージョンをチェックインするとします。ico_616-1 プロジェクトの bufcolor.c オブジェクトのバージョン履歴は、現在 1 --> 2 --> 5 --> 6 --> 7 --> 8 です。バージョン 1 は *integrate*（統合）状態、バージョン 2、5、6、7 は *checkpoint*（チェックポイント）状態、バージョン 8 は *working*（作業中）状態です。

- ico_616-1 プロジェクトの bufcolor.c オブジェクトの残りのチェックポイントバージョンをすべてコラプスする。

  ```
  ccm collapse bufcolor.c all
  Starting Collapse Process...
  Unable to remove bufcolor.c-1:csrc:1 : no write access.
  bufcolor.c-2:csrc:1 removed.
  bufcolor.c-5:csrc:1 removed.
  bufcolor.c-6:csrc:1 removed.
  bufcolor.c-7:csrc:1 removed.
  Unable to remove bufcolor.c-8:csrc:1 : it is a member of a project.
  Collapse complete with 4 successes and 2 failures.
  ```

  ico_616-1 プロジェクトの bufcolor.c オブジェクトのバージョン履歴は 1 --> 8 となります。

- どのプロジェクトでも使用されていない、10 日以上経過した製品をすべてコラプスする。修正不可オブジェクトをコラプスするには、*ccm_admin* ロールを持っている必要があります。

  1. 製品をクエリします。

     ```
     ccm query "is_product=TRUE and not is_bound()
     and create_time<time('-10:0:0:0')"
     ```

  2. 製品オブジェクトをコラプスします。

     ```
     ccm collapse @
     ```

関連トピック

- checkpoint コマンド

# **conflicts** コマンド

## 表記

```
ccm conflicts [-r|-recurse] [-t|-tasks] [-v|-verbose] [-noformat] [-nowrap]
project_spec
```

## 説明と用途

`conflicts` コマンドにより、タスクおよびベースラインを使用して更新するように更新プロパティが設定されているプロジェクトに関する、コンフリクトを表示できます。

プロジェクトの更新プロパティ がオブジェクトの状態を使用して更新するように設定されているのにこのコマンドを使用した場合、このコマンドを使用するにはプロジェクトはがタスクを使用して更新するように設定されていなければならないことを知らせる、警告メッセージが表示されます。

コンフリクトのさらに詳細およびコンフリクトを特定する方法については、コンフリクト検出を参照してください。

## オプションと引数

*project_spec*

コンフリクトを表示したいプロジェクトを指定します。

`-noformat`

コマンド出力を自動改行もフォーマットも行わず、各オブジェクト バージョンとそのコンフリクトを 1 行で表示することを指定します。各フィールドは <tab> 文字によって区切られます。それにより、各行内のフィールドを解析できます。

このオプションにより、たとえば「No Task」コンフリクトの検出箇所が含まれる行を削除して、「Parallel」コンフリクトの検出箇所が含まれる行を保持するように、出力にフィルタリングすることもできます。

`-nowrap`

コマンド出力を改行せず、各オブジェクト バージョンとそのコンフリクトを 1 行で表示することを指定します。

このオプションにより、たとえば「No Task」コンフリクトの検出箇所が含まれる行を削除して、「Parallel」コンフリクトの検出箇所が含まれる行を保持するように、出力にフィルタリングすることもできます。

デフォルトでは、conflicts コマンドからの出力は 80 文字で改行されます。

-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およびサブプロジェクトで、コンフリクトを検出します。

-t|-tasks

タスクのコンフリクトを表示するように指定します。デフォルトでは、オブジェクトのコンフリクトが表示されます。

-v|-verbose

追加のコンフリクト検出メッセージを出力することを指定します。

## 例

- toolkit-2 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 toolkit-2

Project: toolkit-2
Object Ver.  Task Conflicts
-----------  ---- ----------------------------------------------
main.c-0.1.1 57   Implicitly required but not included - parallel
draw.c-4     117  Explicitly specified but not included - parallel
init.c-2             No task
```

## **copy_project** コマンド

### 表記

```
ccm copy_project|cp|checkout -p|-project|co -p|-project
                [-purpose purpose_spec]
                [-platform platform]
                [-release release|as_is|none]
                [ÅwIBM Rational Synergyks|os]
                [-subprojects]
                [-versions "old_ver:new_ver,old_ver:new_ver,..."]
                [-t|-to version]
                [-c|-comment "string"]
                [-ce|-commentedit]
                [-cf|-commentfile file_path]
                project_spec
```

#### ワークエリア プロパティのコピーと設定

```
ccm copy_project|cp|checkout|co copy_project_options
                [-cb|-copy_based|-not_copy_based|-ncb  (UNIX only)]
                [-rel|-relative|-nrel|-not_relative]
                [-path|-set|-setpath] absolute_path
                [-mod|modifiable_wa] [-nmod|not_modifiable_wa]
                [-tl|-translate|-ntl|-no_translate]
                [-wa|-maintain_wa|-nwa|-no_wa]
                [-wa_t|-wa_time|-nwat|-no_wa_time]
                [-u|update|-no_u|-no_update]
                project_spec
```

#### 新しいプロジェクトのコピーと更新

```
ccm copy_project|cp|checkout -p|-project|co -p|-project copy_project_options
                [-u|-update|-no_u|-no_update]
                project_spec
```

### 説明と用途

`copy_project` コマンドを使用して、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。`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。

-p オプションを指定すると、指定したプロジェクト（またはプロジェクト階層全体）がコピーされます。

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。

## オプションと引数

**-c|-comment "*string*"**

コメント文字列を指定します。

**-cb|-copy_based (UNIX only)**

ワークエリアをコピーベースにします。

詳細については、work_area コマンドを参照してください。

**-ce|-commentedit**

コメントを入力するために、デフォルトのエディタを起動します。

**-cf|-commentfile *file_path***

指定したファイルからのコメントを使用します。コメント文字列とコメント ファイルの両方を指定した場合はコメントがマージされ、ファイルからのコメントの後ろにコメント文字列が付加されます。

**-mod|-modifiable_wa**

ワークエリアを修正可能に指定します。

`-nmod|-not_modifiable_wa`

ワークエリアを修正不可に指定します。

`-no_u|-no_update`

プロジェクトのコピー時にプロジェクトを更新しないように指定します。

`-not_copy_based|-ncb (UNIX only)`

ワークエリアをリンクベースにします。

詳細については、work_area コマンドを参照してください。

`-ntl|-no_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 ファイルを変換しないように指示します。

詳細については、work_area コマンドを参照してください。

`-nrel|-not_relative`

Windows 上で、サブプロジェクトのワークエリアを、親プロジェクトのワークエリアの相対ワークエリアではなく、絶対ワークエリアにします。プロジェクトを初めて作成するとき、これがデフォルトとなります。

UNIX 上では、サブプロジェクトに対してリンクを使用できるようにして、サブプロジェクトのワークエリアを、親プロジェクトのワークエリアの相対ワークエリアではなく、絶対ワークエリアにします。プロジェクトを初めて作成するとき、これがデフォルトとなります。

詳細については、work_area コマンドを参照してください。

`-nwa|-no_wa`

ワークエリアを管理しないようにします（すなわち、ワークエリアをデータベースから切り離します）。

詳細については、work_area コマンドを参照してください。

`-nwat|-no_wa_time`

Rational Synergy に格納されている最後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。

詳細については、work_area コマンドを参照してください。

`-platform` *`platform`*

コピーするプロジェクトまたはプロジェクト階層に対して設定するプラットフォーム値を指定します。プラットフォームの選択肢は、Rational Synergy 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）に記載されます。

デフォルトでは、階層内の各プロジェクトの現在のプラットフォーム値が新しいバージョンにコピーされます。このオプションは `-p` オプションと一緒にのみ使用できます。

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

プロジェクトをコピーするか、あるいは、`-subprojects` オプションと組み合わせて指定した場合に、プロジェクト階層全体をコピーします。

`-purpose` *`purpose_spec`*

指定したプロジェクトに関連付ける目的を指定します。

目的の選択肢は Synergy Classic の Project Purpose Table に記載され、データベースで使用される目的値のすべてを含みます。

開発者としてプロジェクトをコピーする場合、このオプションのデフォルトは `Insulated Development`（個別開発）ですが、代わりに `Shared`（共有）を指定できます。ビルド マネージャであるか *ccm_admin* ロールを持っている場合は、デフォルト設定は `Integration Testing`（統合テスト）です。

`ÅwIBM Rational Synergyks|os`

コピーするプロジェクトまたはプロジェクト階層を、ベースラインとタスク（`tasks`）を使用するか、あるいは候補オブジェクトの状態（`os`）を使用して更新するどうかを指定します （候補オブジェクトの状態を使用する方法は、Synergy 4.2 より以前のリリースで使用される唯一の方法です）。

`-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。このオプションは、プロジェクトを 1 つの場所で使用する場合のみ設定できます。設定後、このプロジェクトは、他のプロジェクトのサブプロジェクトとしては使用でなくなります。

`-release` *`release`*

コピーするプロジェクトまたはプロジェクト階層に対して設定するリリースム値を指定します。リリース選択肢には、データベースで使用されるリリース値のすべて、`as is`、または none があります。

デフォルトでは、階層内の各プロジェクトの現在のリリース値が新しいバージョンにコピーされます。

`-subprojects`

指定したプロジェクトの階層内にあるすべてのサブプロジェクトをコピーします。

`-t|-to` *`file_spec|version`*

新しいプロジェクトまたはプロジェクト階層を指定します。

プロジェクトをコピーする場合は、バージョンのみを指定できます。プロジェクトの階層をコピーする場合、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。

`-tl|-translate`

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に指示します。

詳細については、work_area コマンドを参照してください。

`-u|-update`

プロジェクトのコピー時にプロジェクトを更新するように指定します。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を順守し、

プロジェクトをワークエリアなしでコピー後に更新します。ワークエリアの管理は可能です。つまり、このプロジェクトに対応するワークエリアが管理されます。

`-versions "old_ver:new_ver,old_ver:new_ver,..."`

Rational Synergy に組み込まれたアルゴリズムを使用して、デフォルトの次のバージョンを計算します （ほとんどの場合、現在のバージョンに「1」が足されます)。次のバージョンを変更する場合は、上記の構文を使用して、旧バージョンを新しいバージョンに対応付けます。

プロジェクト階層をコピーする場合、各対応付けが、現在値を持っている階層内の全プロジェクトに適用されます。*new_ver* が `NoCopy` の場合、*old_ver* 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。

コピーするすべての新しいプロジェクトに対して同じバージョンを指定する場合は、`-to` オプションを使用します。`-to` オプションと `-version` オプションは一緒には使用できません。

`-wa|-maintain_wa`

ワークエリアを管理します（ワークエリアの同期を取り、同期の状態を維持します)。

詳細については、work_area コマンドを参照してください。

`-wat|-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 の修正時刻を示すように、プロジェクトの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。

詳細については、work_area コマンドを参照してください。

## 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 *working*（作業中）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 bill
```

- システム テストのため、新しい *prep* 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 sqa -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

- checkin コマンド

# **copy_to_file_system** コマンド

## 表記

```
ccm cfs|copy_to_file_system|wa_snapshot project_spec [project_spec] [-p|-path
path] [-r|-recurse]
```

## 説明と用途

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できます。

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。

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、つねにコピーベース
- ファイルは読み出し専用
- 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

## オプションと引数

-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 [project_spec]*

コピーされるプロジェクトを指定します。

-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します。

```
ccm copy_to_file_system -path C:¥ccm_wa¥ccm_docs proj1-1 proj2-1
```

# **create** コマンド

## 表記

```
ccm create [-t|-type type] [-task task_number]
           [-c|-comment "string"] [-ce|-commentedit]
           [-cf|-commentfile file_path]
           file_spec [file_spec...] [-v version]
ccm create [-t|-type project]
           [-c|-comment "string"] [-ce|-commentedit]
           [-cf|-commentfile file_path]
           [-release release] [-plat|-platform platform]
           [-purp|-purpose purpose_spec]
           [-task task_number]
           [-reconf|-reconfigure tasks|os]
           [-wa|-maintain_wa|-nwa|-no_wa] [-set]
           project_spec [project_spec...]
ccm create -t|-type project
           [-c|-comment "string"] [-ce|-commentedit]
           [-cf|-commentfile file_path]
           [-release release] [-plat|-platform platform]
           [-purp|-purpose purpose_spec]
           [-task task_number]
           [-reconf|-reconfigure tasks|os]
           -r|-root dir_spec -v version
```

## 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 Rational Synergy 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、それは、Rational Synergy 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名を作成する場合は、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。初期化ファイル内の `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る （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 Rational Synergy プロジェクト内のサブプロジェクトとして作成できる （警告を参照してください）。
- プロジェクトを作成する場合、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（ワークエリアの場所の変更方法については、work_area コマンド を参照してください）。
- `-t project -r` *`dir_spec`* `-v` *`version`* オプションを使用してプロジェクトを作成する場合は、Rational Synergy は既存の *`dir_spec`* を新しいプロジェクトのルート ディレクトリとして使用し、このプロジェクトは *`dir_spec`* オブジェクトの名前を使用します。
- 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。

## オプションと引数

`-c|-comment "`*`string`*`"`

コメント文字列を指定します。

`-ce|-commentedit`

コメントを入力するためにエディタを起動します。

`-cf|-commentfile` *`file_path`*

指定したファイルからのコメントを使用します。コメント文字列とコメント ファイルの両方を指定した場合はコメントがマージされ、ファイルからのコメントの後ろにコメント文字列が付加されます。

*`file_spec`*

作成するファイルの名前を指定します。

`-nwa|-no_wa`

ワークエリアを管理しないようにします（すなわち、ワークエリアをデータベースから切り離します）。

`-plat|-platform` *`platform`*

作成するプロジェクトのプラットフォームを指定します。

*`project_spec`*

作成するプロジェクトの名前を指定します。新しいプロジェクトは、それがサブプロジェクトでなければ、自動的に表示されます。

`purpose` *`purpose_spec`*

作成するプロジェクトの目的の名前とメンバーの状態を指定します。すべてのプロジェクトの目的のリストを表示する場合は、`ccm project_purpose show` を使用します。

`-r|-root` *`dir_spec`*

*`dir_spec`* が新しいプロジェクトのルート ディレクトリとなるように指定します。新しいプロジェクトの名前にはルート ディレクトリの名前が使用されます。

`-reconf|-reconfigure tasks|os`

新しいプロジェクトの更新（リコンフィギュア）の方法を指定します。

`-release` *`release`*

作成するプロジェクトのリリースを指定します。

`-set`

ワークエリア パスを設定します。このオプションは `-wa` オプションと一緒にのみ使用できます。

`-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）。

有効なタイプのリストを表示するには、show コマンド を使用します。

`-task` *`task_number`*

新しく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。また、オブジェクトが読み出し専用ディレクトリで作成された場合、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（デフォルト）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、作成また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。

`-v` *`version`*

新しいプロジェクトのバージョンを指定します。このオプションは、`type` オプションと `root` オプションと一緒に使用する必要があります。

allow_delimiter_in_name を `FALSE` に設定すると、オブジェクト名にバージョンを含めることができます。例：

```
ccm create foo-1
```

`allow_delimiter_in_name` を `TRUE` に設定した場合は、version オプションを使用する必要があります。例：

```
ccm create foo -v 1
```

`-wa|-maintain_wa`

ワークエリアを管理します（ワークエリアの同期を取り、同期の状態を維持します）。

## 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
  ```

- 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
  ```

- 新しいプロジェクトを GUI ディレクトリから最終バージョンで作成する。

  ```
  ccm create -t project -v final -r GUI
  ```

- 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
  ```

- Rational Synergy ディレクトリを現在のディレクトリの下のサブプロジェクトに変更する。

  1. プロジェクトを作成します。

     ```
     ccm create -t project -purpose project_purpose -release
     release_value -r directory_name -v version [project_create_options]
     ```

  2. ディレクトリをサブプロジェクトに置き換えます。最初に、ディレクトリの使用を解除します。

     ```
     ccm unuse directory_name
     ```

  3. 新しいサブプロジェクトを使用します。

     ```
     ccm use -p project-version
     ```

## 警告

-r *dir_spec* オプションを使用して、最上位のプロジェクトのルート ディレクトリに新しいプロジェクトを作成することはできません。このオプションはサブディレクトリに対してのみ使用します。

関連トピック

- delete コマンド
- show コマンド
- use コマンド

# **dcm** コマンド

## 表記

### 初期化

```
ccm dcm -init -dbid|-database_id database_id
                      [-delim "single-character_delimiter"]
                      [-description description]
                      [-location location]
                      [-admin_info admin_info]
```

### 追加

```
ccm dcm -add -ts|-transfer_set "transfer_set_name"
                      [-h|-history] | [-nh|-no_history]
                      file_spec [file_spec...]
```

### データベース **ID** の変更と影響を受けるオブジェクトの更新

```
ccm dcm -change -dbid|-database_id database_id -convert
```

### **DCM** 区切り文字の変更とすべてのオブジェクトの更新

```
ccm dcm -change [-from_delim|-from_delimiter delimiter]
                -delim|-delimiter delimiter
```

### オブジェクトの更新を伴わないデータベース **ID** の変更

```
ccm dcm -change -dbid|-database_id database_id
```

### ディレクトリ プロジェクト インスタンスの変更

```
ccm dcm -change_dir_project_instance |
        -cdpi project_name instance dirobjectspec1 [...dirobjectspecN]
```

### 別のデータベースで作成したオブジェクトのデータベース **ID** の変換

```
ccm dcm -change -from_dbid|-from_database_id fromdbid
        -to_dbid|-to_database_id todbid
```

### データベース定義の作成

```
ccm dcm -create -dbid|-database_id database_id
                        [-desc|-description description_of_database]
                        [-tm|-transfer_mode transfer_mode_name]
                        [-ar|-automatic_receive|-noar|-noautomatic_receive
                        [-rb|-run_in_background|-norb|-norun_in_background]
                        [-host hostname]
                        [-os|-operating_system UNIX|Windows]
                        [-path path]
                        [-ccm_home path]
                        [-zip|-nozip]
                        [-tp|-transfer_path transfer_path]
                        [-ga|-generate_allowed|-noga|-nogenerate_allowed]
                        [-hidden|-nohidden]
                        [-handover_dbid|-handover_database_id dbid]
                        [-location location]
                        [-admin_info admin_info]
```

### 転送セットの作成

```
ccm dcm -create -ts|-transfer_set "transfer_set_name"
                        [-email email_address]
                       [-crsc|-change_request_scope change_request_scope_name]
                        [-crq|-change_request_query query_expression]
                        [-exclude_products|-noexclude_products]
                        [-exclude_imported_objects|
                        -noexclude_imported_objects]
                        [-exclude_types "list_of_types"]
                        [-exclude_typedefs|-noexclude_typedefs]
                        [-local_parallel|nolocal_parallel]
                        [-ferp|-noferp]
                        [-rsc|-release_scope release_scope_name]
                        [-rq|-release_query release_query_string]
                        [-cumrsc|-cumulative_release_scope|-nocumrsc| -
nocumulative_release_scope]
                        [-dir]
                        [-ib|-include_baselines|-noib|-noinclude_baselines]
                        [-ep|-email_policy policy]
                        [-exclude_nct|-exclude_non_completed_tasks|
                        -noexclude_nct|-noexclude_non_completed_tasks]
                        [-exclude_db_info|-noexclude_db_info]
                        [-cumulative|-nocumulative]
```

### データベース定義の削除

```
ccm dcm -delete -dbid|-database_id database_id
```

### 転送セットの削除

```
ccm dcm -delete -ts|-transfer_set "transfer_set_name"
```

### 生成

```
ccm dcm -gen|-generate -dbid|-database_id database_id
                       [-ts|-transfer_set "transfer_set_name"|-notransfer]
                       [-lg|-last_generated last_generated_value]
                       [-wait|-nowait]
                       [-email email_address|-noemail]
```

### 生成と転送

```
ccm dcm -gen|-generate -trn|-transfer
                       -dbid|-database_id database_id
                       -ts|-transfer_set "transfer_set_name"
```

### 生成、転送、および受取り

```
ccm dcm -gen|-generate -trn|-transfer -rec|-receive
                       -dbid|-database_id database_id
                       -ts|-transfer_set "transfer_set_name"
```

### データベース定義の修正

```
ccm dcm -modify -dbid|-database_id database_id
                       [-desc|-description description_of_database]
                       [-tm|-transfer_mode transfer_mode_name]
                       [-ar|-automatic_receive|-noar|-noautomatic_receive]
                       [-rb|-run_in_background|-norb|-norun_in_background]
                       [-host hostname]
                       [-os|-operating_system UNIX|Windows]
                       [-path path]
                       [-ccm_home path]
                       [-zip|-nozip]
                       [-tp|-transfer_path transfer_path]
                       [-ga|-generate_allowed|-noga|-nogenerate_allowed]
                       [-hidden|-nohidden]
                       [-handover_dbid|-handover_database_id dbid]
                       [-location location]
                       [-admin_info admin_info]
```

### 転送セットの修正

```
ccm dcm -modify -ts|-transfer_set "transfer_set_name"
                        [-email email_address|-noemail]
                       [-crsc|-change_request_scope change_request_scope_name]
                        [-crq|-change_request_query query_expression]
                        [-exclude_products|-noexclude_products]
                      [-exclude_imported_objects|-noexclude_imported_objects]
                        [-exclude_types "list_of_types"]
                        [-exclude_typedefs|-noexclude_typedefs]
                        [-local_parallel|nolocal_parallel]
                        [-ferp|-noferp]
                        [-dir]
                        [-rsc|-release_scope release_scope_name]
                        [-rq|-release_query release_query_string]
                        [-cumrsc|-cumulative_release_scope|-nocumrsc| -
nocumulative_release_scope]
                        [-ib|-include_baselines|-noib|-noinclude_baselines]
                        [-ep|-email_policy policy]
                        [-exclude_nct|-exclude_non_completed_tasks|-
noexclude_nct|-noexclude_non_completed_tasks]
                        [-exclude_db_info|-noexclude_db_info]
                        [-cumulative|-nocumulative]
```

### **DCM** 設定の修正

```
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]
```

受取り

```
ccm dcm -rec|-receive [-a|-all]
                      [-dbid|-database_id database_id]
                      [-dir directory]
                      [-ts|-transfer_set "transfer_set_name"]
                      [-im|-ignore_missing]
                      [-wait|-nowait]
                      [-ic|-ignore_checks|-noic|-noignore_checks]
```

転送セットの間接変更依頼メンバーの再計算

```
ccm dcm -recompute -crs|-change_requests|-problems -ts|-transfer_set
"transfer_set_name" [-dbid|-database_id database_id]
```

転送セットのメンバーの再計算

```
ccm dcm -recompute -ts|-transfer_set "transfer_set_name"
```

再初期化

```
ccm dcm -init [-dbid|-database_id database_id]
                       [-delim "single-character_delimiter"]
                       [-description description]
                       [-location location]
                       [-admin_info admin_info]
```

転送セットからのオブジェクトの削除

```
ccm dcm -remove -ts "transfer_set_name" filespec [filespec...]
```

すべての **ID** と説明の表示

```
ccm dcm -show -dbid|-database_id -all
```

現在の **DCM** のデータベース **ID** の表示

```
ccm dcm -show -dbid|-database_id
```

データベース定義の表示

```
ccm dcm -show -dbid|-database_id database_id
```

**DCM** プロパティの表示

```
ccm dcm -show -prop|-properties objectspec1 [ objectspec2 ... objectspecN]
```

**DCM** 設定の表示

```
ccm dcm -show -settings
```

最後の生成時刻の表示

```
ccm dcm -show -ts|-transfer_set "transfer_set_name"
                      -dbid|-database_id "database_id"
```

指定した **1** つのイベントの表示

```
ccm dcm -show -event_log|-el -index number
                      [-f|-format format]
                      [-info]
                      [-messages]
```

受取りロックの表示

```
ccm dcm -show -receive_lock
```

イベントの概要の表示

```
ccm dcm -show -event_log|-el
                      [-f|-format format]
                      [-dbid|-database_id dbid]
                      [-ts|-transfer_set transfer_set_name]
```

転送セットの表示

```
ccm dcm -show -ts|-transfer_set "transfer_set_name"
                      [-members direct|all][-u]
```

> 注記：`ccm dcm -show -ts` コマンドによって変更依頼の範囲と変更依頼のクエリ情報を表示できるのは、Distributed Change（DCS）ライセンスを利用できる場合のみです。

転送

```
ccm dcm -trn|-transfer [-a|-all] |
                       [-dbid|-database_id database_id]
                       [-ts|-transfer_set "transfer_set_name"]
```

## 前提条件

現在のデーベースが、Rational Synergy Distributed とも呼ばれる、分散変更管理（DCM）用に初期化されている必要があります。詳細は、IBM Rational Information Center でドキュメント『IBM Rational Synergy Distributed』を参照してください。

## 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。

## オプションと引数

-a|-all

-trn オプションと一緒に使用した場合はすべての転送セットをデスティネーション データベースに送出し、-rec オプションと一緒に使用した場合はすべての転送セットを受け取って現在のデータベースに保存します。

このオプションは、ccm dcm -show コマンドの場合は -dbid と一緒に使用できますが、ccm dcm -transfer コマンドの場合は -dbid と一緒には使用できません。-all オプションは、-ts オプションと一緒には使用できません。

-add

指定したオブジェクト（1 つのファイル、プロジェクト、タスク、またはフォルダ）を、指定した転送セットに追加します。転送セットは定義済みである必要があります。

-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*

DCM 管理の問題の担当者の連絡先情報を指定します。*admin_info* の値は、1 つまたは複数の名前、電話番号、電子メール アドレスを含むフリー形式のテキストです。

`-ar|-automatic_receive`

生成操作時に受取りが自動的に開始されるように指定します。

`-ccm_home` *`path`*

Rational Synergy インストール エリアへの絶対パスを指定します。Windows サーバーを使用している場合は、UNC パスを入力します。

`-change`

現在のデータベース ID または DCM 区切り文字を変更するように指定します。このオプションでは、データベースが保護され、データベース上でほかのセッションが実行されていないことが要求されます。

`-change_dir_project_instance|-cdpi`

1 つまたは複数の指定サブディレクトリにある指定インスタンスを参照させるため、プロジェクトのディレクトリ エントリを修正します。旧リリースからのアップグレード後、または旧リリースからのパッケージを受け取った後で、このオプションにより、そのようなディレクトリ エントリを修正できます。`-change_dir_project_instance` オプションと一緒に使用するディレクトリ引数は、クエリ参照を含むことができる標準のオブジェクト仕様です。

`-crq|-change_request_query` *`query_expression`*

変更依頼のクエリ式を指定します。`none` 以外の `change_request_scope` について、クエリ式を定義できます。このオプションを使用するには、DCS ライセンスが必要です。DCS ライセンスなしでこのオプションを使用した場合、このコマンドは失敗し、エラー メッセージが表示されます。

`-crsc|-change_request_scope` *`change_request_scope_name`*

転送パッケージに含めることができる変更依頼と関連するタスクおよびオブジェクトを指定します（詳細は『IBM Rational Synergy Distributed』を参照してください）。このオプションを使用するには、DCS ライセンスが必要です。DCS ライセンスなしでこのオプションを使用した場合、このコマンドは失敗し、エラー メッセージが表示されます。

*`change_request_scope_name`* の値には以下のいずれかを設定します。

- `none`
- `crs`
- `crs only`
- `crs and tasks`

- crs_and_tasks
- crs_tasks_and_objects
- crs, tasks and objects
- change_requests
- change_requests_and_tasks
- change_requests_tasks_and_objects
- problems
- problems_and_tasks
- problems_tasks_and_objects

-crs|-change_requests

-recompute と一緒に使用して、転送セットに自動的に追加された変更依頼と現在生成されている変更依頼を、次の DCM 転送パッケージで送出しないようにします。

-convert

変更操作によって現在のデータベースのデータベース ID を変更し、影響されるオブジェクトを更新するように指定します。このオプションは、望ましいデータベース ID を持たないデータベース上で使用します。したがって、現在のデータベースで作成されたすべてのオブジェクトは、正しいデータベース ID を参照するように更新する必要があります。

-cumulative

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、生成操作または生成プレビュー操作の実行時に必ず評価されます。ただし、cumulative を指定すると、前のクエリで見つかった古いメンバーは削除されません。間接クエリベースのメンバーシップは、追加されるのみで、累積されていきます。

-cumrsc|-cumulative_release_scope

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必ず評価され、前のクエリで見つかった古いメンバーは削除されません。

-dbid|-database_id *customer_dbid*

生成されるパッケージの受取りが可能な顧客のデータベース ID を指定します。

-dbid|-database_id *database_id*

-gen オプションまたは -trn オプションと一緒に使用した場合は転送セットのデスティネーション データベースを指定し、-rec オプションと一緒に使用した場合は転送セットのソース デスティネーション データベースを指定します。

`-init` オプションと一緒に使用した場合は、DCM 初期化の対象となるデータベースのデータベース ID を指定します。

`-change` オプションと一緒に使用した場合は、現在のデータベース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。多くのオブジェクトの更新が必要な場合、`ccm dcm -change` コマンドの実行には長い時間がかかります。

`-event_log` オプションと一緒に使用した場合は、指定したデータベースのエントリのみを一覧表示するように指定します。

`-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に指定します。

`-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。

`-delim` *`delimiter`*

現在のデータベースに割り当てる新しい区切り文字を指定します。

a ～ z、A ～ Z の文字、または 0 ～ 9 の数字は、区切り文字として使用できません。「!」、「~」、または「=」を代替区切り文字として使用できます。デフォルトの DCM 区切り文字は「#」であり、可能な限りこの文字を使用してください。

多くのオブジェクトの更新が必要な場合、`ccm dcm -change -delim` *`delimiter`* コマンドの実行には長い時間がかかります。

`-delim` オプションは必須ではありません。これを省略した場合、デフォルトの「#」が使用されます。

`-init` オプションを使用して DCM データベースの再初期化をする場合は、指定する `-delim` オプションは、現在のデータベースのオプションと一致する必要があります。

`-desc|-description` *`description_of_database`*

データベースの説明を指定します。

`-dir|-directory`

`ccm dcm -create -ts` コマンド、または `ccm dcm -modify -ts` コマンドと一緒に使用して、転送セットの生成ディレクトリを設定します。転送セット作成時のデフォルトは空白の文字列です。空白文字列は、現在のデータベースのデフォルトの `dcm -generate` パスが使用されることを意味します。

`-dir` *`directory`*

デフォルトのディレクトリ *`database`*`/dcm/receive` ではなく、指定したディレクトリから DCM パッケージを受け取るように指定します。

`-email` *`email_address`*

生成または受取りの後に、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。

`-ep|-email_policy` *`policy`*

生成操作時または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。*`policy`* に入力する値は、`Transfer`、`Generate`、`Always` のいずれかです。これらの文字列には大文字と小文字の区別はありません。

- `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が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に指定します。DCM 生成操作が実行されたときに含まれるオプションがない場合、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に指定します。新しい転送セットの作成時には、これがデフォルトとなります。ただし、DCM 生成操作の実行時に含め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に指定します。DCM 生成操作の実行時に含め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され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。

`-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`* はゼロ以外の正の整数です。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。

`-exclude_db_info`

現在のデータベースおよび既知の DCM データベースの定義に関する情報を、情報ファイルから除外するように指定します。このオプションは `-noexclude_db_info` オプションと一緒には使用できません。

`-exclude_imported_objects`

インポートしたオブジェクトを、DCM 転送セットから除外するように指定します。

`-exclude_nct|-exclude_non_completed_tasks`

完了していないタスクを、転送リストから除外するように指定します。`-exclude_non_completed_tasks` オプションと `-noexclude_non_completed_tasks` オプションは一緒に使用できません。

`-exclude_products`

製品オブジェクトを、DCM 転送セットから除外するように指定します。

`-exclude_typedefs`

転送パッケージにユーザー定義タイプの定義を含めないようにします。このオプションは、`ccm dcm -create -ts` コマンドまたは `ccm dcm -modify -ts` コマンドと一緒にのみ使用できます。`-exclude_typedefs` オプションと `-noexclude_typedefs` オプションは一緒に使用できません。

`-exclude_types "`*`list_of_types`*`"`

リスト内のタイプを、DCM 転送セットから除外するように指定します。

`-ferp`

プロジェクトの更新（リコンフィギュア）プロパティを完全に展開して以下を含むように指定します。

- すべてのタスク オブジェクト（これらのオブジェクトがプロジェクト階層のメンバーである場合でも）
- プロジェクトの更新プロパティ内のすべてのフォルダとタスク

`-ferp` オプションと `-noferp` オプションは一緒に使用できません。転送セット作成時のデフォルトは `-noferp` です。

*`-file_spec`*

転送セットに追加するファイル、プロジェクト、フォルダ、またはタスクの名前を指定します。

*-format format*

コマンドの出力フォーマットを指定します。出荷時のイベント ログのデフォルトのフォーマットは以下のとおりです。

```
%index %event_time %event_type %status %dbid %ts
```

ここで、

`%index` はインデックス番号です。これは、イベント エントリを識別する一意の番号です。

`%event_time` は、ログ ファイルが作成された日時です。

`%event_type` はログされるイベントのタイプです。`%event_type` の値には `Generate`、`Transfer`、および `Receive` があります。

`%status` は操作の状態です。`%status` の値には、`Started`、`Successful`、`Failed`、`Cancelled` があります。

生成イベントまたは転送イベントの場合、`%dbid` はデスティネーション イベントです。受取りイベントの場合、`%dbid` はソース データベース ID です。

`%ts` は転送セットの名前です。

また、`%user` キーワードを使用して、イベント リスト内のフィールドを定義できます。ここで、`%user` は操作を実行したユーザーです。

`-from_dbid|-from_database_id` *`fromdbid`*

*`fromdbid`* データベースを参照するオブジェクトを変換して、このオブジェクトが *`todbid`* データベースのデータベース ID を使用できるように指定します。これによって、現在の DCM データベース ID は変更されません。多くのオブジェクトの更新が必要な場合、このオプションを使用するコマンドの実行には長い時間がかかります。

`-from_delim|-from_delimiter` *`delimiter`*

変換の対象となる DCM 区切り文字を指定します。`-from_delim` オプションを省略した場合、古い DCM 区切り文字が指定した新しい区切り文字に変換されます。

`-ga|-generate_allowed`

定義を作成または変更するデータベースを、DCM 生成操作のデスティネーション データベースとして使用できるように指定します。新しいデータベース定義を作成するとき、これがデフォルトとなります。このオプションは `-nogenerate_allowed` オプションと一緒には使用できません。

`-gen|-generate`

指定した転送セットとデスティネーション データベースの組み合わせの転送パッケージを生成します。

`dcm -gen` と入力して、使用状況メッセージを表示します。

`-handover_dbid|-handover_database_id` *`dbid`*

管理を指定するデータベースに渡す際に、その管理を中継するデータベースを指定します。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、*`dbid`* オプション値にハブデータベースの dbid を指定する必要があることを意味します。指定した *`dbid`* 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。

`-hidden`

定義を作成または変更するデータベースのデータベース ID（dbid）が、データベースに関するダイアログのリスト ボックスに表示されないように指定します。このオプションは `-nohidden` オプションと一緒には使用できません。

`-h|-history`

指定した各オブジェクトの履歴が転送セットに含まれるようにします。フォルダまたはタスクの履歴ではなく、単一のファイルまたはプロジェクトの履歴のみを含めることができます。このオプションを使用するのは、`-add` オプションを使用する場合のみです。

`-ib|-include_baselines`

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が含まれるように指定します。このオプションは `-noib|-noinclude_baselines` オプションと一緒には使用できません。転送セット作成時のデフォルトの設定は、DCM の `Default Include Baselines` 設定によって決まります。

`-ignore_checks`

転送パッケージの検査の結果潜在的な問題が指摘された場合でも、DCM 受取り操作を続行できるようにします。

> 注意！このオプションが持つ意味を理解しないまま使用すると、不測の結果を招くことがあります（詳細については、『IBM Rational Synergy Distributed』を参照してください）。

このオプションは `ccm dcm -receive` コマンドと一緒にのみ使用できます。`-ignore_checks` オプションと `-noignore_checks` オプションは一緒に使用できません。

`-ignore_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たないように指定します。

`-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。

`-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に指定します。

`-init`

指定したデータベースを指定したデータベース ID と DCM 区切り文字に初期化します。

DCM 用に初期化されていないデータベースを初期化する場合、`-dbid` オプションを使用する必要があります。データベースがすでに DCM 用に初期化されている場合、`-dbid` は省略可能です。このオプションを指定する場合、その値は現在のデータベース ID の値と一致する必要があります。同様に、`-init` オプションを使用して DCM データベースを再初期化する場合、指定する `-delim` オプションは、現在のデータベースの値と一致する必要があります。

`-keep_typedefs`

受取り操作の完了後にタイプ定義を維持するように指定します。

`-lg|-last_generated`

last_generated 時刻を指定します。

*last_generated* は、以下のいずれかの引数をとる必要があります。

— `never`
— `current`
— `integer index` ここで、`1` は最後に生成された転送パッケージを表します。

このオプションは上級ユーザー用です。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。

`current` を指定すると、今生成されたものではない場合でも、今生成されたことを示すタイム スタンプが設定されます。この結果、すべてのメンバーが最新であるように表示され、転送パッケージにはオブジェクトが含まれません。新しい転送パッケージを使用して更新をハブ データベースまたは公開者データベースに送出する場合に、`current` を使用します。

> 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` の値が `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、過去の転送で含まれたすべてのオブジェクトと、それに加えて、最後の転送以降に変更されたすべてのオブジェクトが含まれます。

`-local_parallel`

受け取ったオブジェクトのパラレル バージョンのローカル所有者に、電子メールでパラレル通知を送信します。これは、新しい転送セットを作成するときのデフォルトです。このオプションは、`ccm dcm -create -ts` コマンドまたは `ccm dcm -modify -ts` コマンドと一緒にのみ使用できます。`-local_parallel` オプションと `-nolocal_parallel` オプションは一緒に使用できません。

`-location` *location*

データベースの地理的な位置を指定します（例、`Irvine, California`）。*location* の値は、どのサイトがデータベースを所有するかを指示します。

`-log`

ログ ファイルから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に指定します。

`-members direct|all`

`-members direct` を指定した場合、このコマンドを実行すると直接メンバーのみが表示されます。`all` を指定した場合、このコマンドを実行すると直接メンバーと間接メンバーの両方が表示されます。

`-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に指定します。

`-mpi|-map_project_instances`

定義を作成または変更するデータベースにあるプロジェクト インスタンスを、DCM 生成操作時に値「1」に対応付けるように指定します。このオプションは `-nomap_project_instances` オプションと一緒には使用できません。デフォルトでは、新しい DCM データベース定義を作成するとき、プロジェクト インスタンスは値「1」に対応付けられません。

`-nh|-no_history`

指定した各オブジェクトの履歴が転送セットに自動的に含まれないようにします。

オブジェクトの指定したバージョンのみを転送セットに含めるようにする場合は、このオプションを使用します。このオプションを使用するのは、`-add` オプションを使用する場合のみです。

`-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、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。

`-noar|-noautomatic_receive`

生成操作時に受取りが自動的に開始されないように指定します。

`-nocumulative`

変更依頼範囲が累積されないように指定します。新しい転送セットの作成時には、これがデフォルトとなります。転送セットの変更依頼範囲とクエリは、生成操作または生成プレビュー操作の実行時に評価されます。ただし、`nocumulative` を指定すると、転送セットの間接クエリ メンバーであったが、クエリによって検出されなくなったオブジェクトは、転送セットから削除されます。この結果、変更依頼範囲は累積されません。

`-nocumrsc|-nocumulative_release_scope`

リリース範囲が累積されないように指定します。転送セットのリリース範囲とクエリは必ず評価され、前のクエリで見つかった旧メンバーは削除されます。

`-nocumulative_release_scope`

リリース範囲が累積されないように指定します。転送セットのリリース範囲とクエリは必ず評価され、前のクエリで見つかった旧メンバーは削除されます。

`-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と一緒に追加されないように指定します。

`-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、転送セットに含まれないように指定します。

`-noemail`

電子メール パラメータを、転送パッケージに対して定義しないように指定します。

`-noexclude_db_info`

現在のデータベースおよび既知の DCM データベースの定義に関する情報を、DCM 情報ファイルに含めるように指定します。転送セット作成時のデフォルトは `-noexclude_db_info` です。このオプションは `-exclude_db_info` オプションと一緒には使用できません。

`-noexclude_imported_objects`

インポートしたオブジェクトを DCM 転送セットに含めるように指定します。

-noexclude_nct|-noexclude_non_completed_tasks

完了していないタスクを転送リストに含めるように指定します。このオプションは -exclude_non_completed_tasks オプションと一緒には使用できません。

-noexclude_products

製品オブジェクトを DCM 転送セットに含めるように指定します。

-noexclude_typedefs

転送パッケージにユーザー定義タイプの定義が含めるようにします。このオプションは、ccm dcm -create -ts コマンドまたは ccm dcm -modify -ts コマンドと一緒にのみ使用できます。このオプションは -exclude_typedefs オプションと一緒には使用できません。

-noferp

更新に含むため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に指定します。したがって、プロジェクトの更新プロパティには以下は含まれません。

- すべてのタスク オブジェクト
- プロジェクトの更新プロパティ内のすべてのフォルダとタスク

-ferp オプションと -noferp オプションは一緒に使用できません。転送セット作成時のデフォルトは -noferp です。

-noga|-nogenerate_allowed

定義を作成または変更するデータベースを、DCM 生成操作のデスティネーション データベースとして使用できないように指定します。このオプションは、定義を作成または変更するデータベースを DCM Generate ダイアログに表示せず、ccm dcm -generate コマンドで使用できないように指定します。このオプションは -generate_allowed オプションと一緒には使用できません。

-nohidden

定義を作成または変更するデータベースのデータベース ID（dbid）が、データベースに関するダイアログのリスト ボックスに表示されるように指定します。このオプションは -hidden オプションと一緒には使用できません。

-noib|-noinclude_baselines

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が含まれないように指定します。このオプションは -ib|-include_baselines オプションと一緒には

使用できません。転送セット作成時のデフォルトの設定は、DCM の `Default Include Baselines` 設定によって決まります。

`-noignore_checks`

転送パッケージの検査の結果潜在的な問題が指摘された場合、DCM 受け取り操作を続行できないようにします。安全上の理由から、これがデフォルトです。このオプションは `-receive` オプションと一緒には使用できません。`-ignore_checks` オプションと `-noignore_checks` オプションは一緒に使用できません。

`-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に指定します。

`-nokeep_typedefs`

受取り操作の完了後にタイプ定義を維持しないように指定します。

`-nolocal_parallel`

受け取ったオブジェクトのパラレル バージョンのローカル所有者に、電子メールでパラレル通知を送信しないようにします。このオプションは、`ccm dcm -create -ts` コマンドまたは `ccm dcm -modify -ts` コマンドと一緒にのみ使用できます。`-local_parallel` オプションと `-nolocal_parallel` オプションは一緒に使用できません。

`-norb|-norun_in_background`

自動受取りをバックグラウンドで実行しないように指定します。その結果、セッションを使用を続けるには、転送パッケージを受け取る必要があります。

`-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に指定します。

`-noupdate_rtf`

受取り操作時に、プロセス ルールを更新しないように指定します。

`-nowait`

別のセッションが受取りを実行していても、転送パッケージが DCM 受取りまたは自動受取りを実行できるようにします。デフォルトは `-wait` です。

`-nozip`

転送パッケージを、そ圧縮せずにデータ ファイルの集りのままにしておき、ファイルごとに転送するように指定します。`-zip` オプションと `-nozip` オプションは一緒に使用できません。

`-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、ゼロ以外の正の浮動小数点数とします。

`-os|-operating_system UNIX|Windows`

デスティネーション データベースのオペレーティング システムを指定します。

`-parallel_checking` *`parallel_check_keyword`*

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。*`parallel_check_keyword`* には、`none`、`created`、`updated` などを指定できます。これらの値は大文字と小文字を区別します。

`-path` *`path`*

データベース名を含む、デスティネーション データベースへの絶対パスを指定します。Windows サーバーを使用している場合は、UNC パスを入力します。

`-prop|-properties`

指定したオブジェクトをメンバーに持つ転送セット、メンバーのタイプ、オブジェクトの送り先となったデータベースを表示します。このオプションは、Show DCM Properties ダイアログと同じ情報を表示します。

`-rb|-run_in_background`

自動受取りをバックグラウンド実行します。その結果、転送パッケージの受取りを待つことなく、セッションの使用を続けることができます。

`-rec|-receive`

`-gen` オプションまたは `-trn` オプションと一緒に使用した場合はデスティネーション データベースで転送パッケージの自動受取りが行われ、`-rec` オプションと一緒に使用した場合は現在のデータベースで受取り操作を開始します。

`-ts` オプションを入れ、`-dbid` オプションを除くことによって、「*`transfer_set_name`*」の名前を持つすべての転送パッケージ（すべてのデータベー

スから）を受け取ります。-dbid オプションを入れ、-ts オプションを除くことによって、指定したデータベースからのすべての転送パッケージを受け取ります。

dcm -rec と入力して、使用状況メッセージを表示します。

-receive_lock

受取りロックが存在する場合、その受取りロックの詳細を表示します。受取りロックが存在しなければ、このオプションは出力を表示しません。

-nowait オプションを使用して DCM 受取りを開始した場合、その受取りでは受取りロックされていない可能性があります。その場合、-receive_lock オプションによって、このような DCM 受け取りが現在のデータベース上で実行されるどうかを決定することはできません。

-recompute

転送セットに間接メンバーの変更を含まないように、転送セットを更新します。-change_requests と一緒に使用すると、転送セットに自動追加された変更依頼と現在生成されている変更依頼を、次の DCM 転送パッケージで送出されないようにします。このためには、変更依頼メンバーの再計算を実行し、DCM 同期許容時間 ＋ 1 秒間（デフォルトは 61 秒間）待ってから DCM 生成を実行します。

-remove

指定したオブジェクト（1 つのファイル、プロジェクト、タスク、またはフォルダ）を、指定した転送セットから削除します。転送セットは定義済みである必要があります。

-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。

`-rq|-release_query` *`release_query_string`*

指定した新しいリリース クエリ文字列が、転送セットの作成または変更時に使用されるように指定します。このオプションは、`ccm dcm -create -ts` コマンドまたは `ccm dcm -modify -ts` コマンドと一緒にのみ使用できます。

`-rsc|-release_scope` *`release_scope_name`*

新しいリリース範囲の値が、転送セットの作成または変更時に使用されるように指定します。このオプションは、`ccm dcm -create -ts` コマンドまたは `ccm dcm -modify -ts` コマンドと一緒にのみ使用できます。

*`release_scope_name`* の値には以下のいずれかを設定します。

```
none
releases
releases_templates
releases_and_templates
release and templates.
```

`-settings`

`ccm dcm -modify` コマンドと一緒に使用して、DCM 設定を変更するように指示します。`ccm dcm -show` コマンドと一緒に使用して、DCM 設定を表示するように指示します。

`-show`

`-dbid` オプションまたは `-database_id` オプションと一緒に使用して、現在のデータベース ID を表示します。`ccm dcm -show -dbid` *`dbid`* コマンドの出力には、以下が含まれます。

- データベースの地理的な位置。
- データベースの DCM 管理の責任を負う 1 人または複数の担当者の連絡先情報。
- ハンドオーバ データベースの dbid（空の文字列は、DCM 受け取り操作時に、ハンドオーバ データベースで管理され、現在のデータベースで受け取るオブジェクトは保留されているハンドオーバとして表示されないことを意味します。また、これらのオブジェクトはそのデータベースからの更新がいつでも可能です）。
- 現在のエクスポート形式。
- zip 設定。
- DCM 生成操作時にプロジェクト インスタンスが「1」に対応付けられるかどうか。
- 転送パス。

転送モードを direct または ftp のいずれかに指定した場合、zip 設定は自動的に以下のように変更されます。direct の場合は zip 設定は OFF に、ftp の場合は ON に変更されます。

-settings オプションと一緒に使用した場合、-show オプションは DCM 設定リストを表示します。ccm dcm -show -settings コマンドの出力には、以下が含まれます。

- 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。
- 受け取り操作時にリリース定義がどのように更新されるか。
- タイプ定義を受け取り操作後に維持するかどうか。
- 受け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。
- 受け取り操作時に、DCM データベース情報を更新するかどうか。
- 受け取り操作時に、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）。

-ts オプションまたは -transfer_set オプションと一緒に使用した場合、-show オプションは現在の転送セット名を表示します。ccm dcm -show -ts コマンドの出力には、以下が含まれます。

- -ferp|-noferp オプションの設定。
- -dir オプションの値。このディレクトリを空白文字列として物理的に格納する場合、Generate Directory は現在のデータベースのデフォルトの dcm/generate ディレクトリとして表示されます。

-tm|-transfer_mode *transfer_mode_name*

転送パッケージをデスティネーション データベースに送るために使用するメカニズムを指定します。

*transfer_mode* は、以下のいずれかの引数をとる必要があります。

- manual | manual_copy
- cp | copy | local_copy
- ftp | file_transfer_protocol

- rcp | remote_copy
- user | user_defined

転送モードは以下のとおりです。

Manual Copy：テープなどの手動方式を使用して、転送パッケージをデスティネーション データベースに送ります。manual_copy は、転送パッケージを生成しますが、その送信を行わない転送モードです。

Local Copy：copy コマンドを使用してアクセス可能なデータベースに転送パッケージを送ります。送出側データベースのエンジンがデスティネーション データベースの database_dir/dcm/receive ディレクトリにコピーできる場合、コピー コマンドを使用してデータベースにアクセス可能です。

File Transfer Protocol：ftp を使用して転送パッケージを送信します。転送の前に、ftp ログインとデスティネーション ディレクトリを設定する必要があります。

Remote Copy：rcp コマンドを使用してアクセス可能なデータベースに転送パッケージを送ります。送出側データベースのエンジンがデスティネーション データベースの database_dir/dcm/receive ディレクトリに rcp できる場合、rcp コマンドを使用してデータベースにアクセス可能です。

User Defined：ユーザー独自のメカニズムを使用して転送パッケージを送信します。

`-to_dbid|-to_database_id` *`todbid`*

*`fromdbid`* データベースを参照するオブジェクトを変換して、このオブジェクトが *`todbid`* データベースのデータベース ID を使用できるように指定します。これによって、現在の DCM データベース ID は変更されません。多くのオブジェクトの更新が必要な場合、このオプションを使用するコマンドの実行には長い時間がかかります。

`-tp|-transfer_path` *`transfer_path`*

-gen オプションと一緒に使用した場合は単一の転送パッケージをそのデスティネーション パッケージに送出し、-trn オプションと一緒に使用使用した場合は 1 つまたは複数の転送パッケージをデスティネーション データベースに送出します。

`-trn|-transfer`

指定した値への転送パスを設定します。DCM データベース定義作成時のデフォルトの transfer_path は、空白の文字列です。

-ts オプションを入れ、-dbid オプションを除くことによって、「*transfer_set_name*」の名前を持つすべての転送パッケージを送出します。-dbid オプションを入れ、ts オプションを除くことによって、指定したデータベース宛にすべての転送パッケージを送出します。

dcm -trn と入力して、使用状況メッセージを表示します。

-ts|-transfer_set "*transfer_set_name*"

生成、送出または受取りを行う転送セットの名前、あるいは 1 つまたは複数のオブジェクトの追加先の転送セットを指定します。

-event_log オプションと一緒に使用した場合は、指定した転送セットのエントリのみを一覧表示するように指定します。

転送セット名に 1 つまたは複数の空白が含れる場合は、二重引用符で囲む必要があります。

-u

コマンドの出力の自動番号付けを抑止します（番号付けされません）。

-update_db_info

受取り操作時に、DCM データベース情報を更新するように指定します。この情報は、DCM Information ファイルにあるデータで更新されます。

-update_rtf

受取り操作時に、プロセス ルールを更新するように指定します。

-update_releases *release_action_keyword*

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。*release_action_keyword* には、none、active、inactive などを指定できます。これらの値は、大文字と小文字が区別されます。詳細は以下のとおりです。

- none

  リリース定義が作成も更新もされないように指定します。

- 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現在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（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（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。

`-wait`

受取り側データベースが受取りを完了するまで、転送パッケージは無期限に待機します。ユーザーはコマンドを中断して（Ctrl + C）、操作を中止できます。デフォルトは `-wait` です。

`-zip`

転送パッケージを tar 形式または zip 形式で圧縮するように指定します。転送モード ID が `Direct` の場合を除き、`-zip` のデフォルト値は `TRUE` です。このオプションは `direct` 転送モードには影響しません。`-zip` オプションと `-nozip` オプションは一緒に使用できません。

関連トピック

- dcm コマンドの例

## dcm コマンドの例

以下の操作の例について説明します。

- 追加
- 変更
- 作成
- 削除
- 生成
- 生成と転送
- 生成、転送、および受取り
- 設定の修正
- 受取り
- 表示
- 転送

### 追加

- `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
  Adding object 'infotec-23:project:1' to transfer set 'InfoServer source'.
  ```

クエリ出力を使用してオブジェクト名を指定することもできます。

### 変更

- オブジェクトを更新せずに、データベース ID を `ldn1` に変更する。

  ```
  ccm dcm -change -dbid ldn1
  ```

- すべてのオブジェクトを更新し、データベース ID を `jfil` から `sdg1` へ変更する。

  ```
  ccm dcm -change -from_dbid jfil -to_dbid sdg1
  ...progress messages...
  DCM database conversion is complete with no errors.
  4 objects had attributes updated.
  No objects had directory entries updated.
  Your Rational Synergy session must be restarted.
  Rational Synergy engine exiting.
  ```

- DCM 区切り文字を新しい値に変更し、現在のデータベース内のすべてのオブジェクトを更新して同じ区切り文字を使用させる。

  ```
  ccm dcm -change -from_delimiter delimiter -delim delimiter
  ```

### 作成

- DCM 定義を作成し、それを DCM デスティネーション データベース定義に追加する。

```
ccm dcm -create -dbid database_id -desc description -tm
transfer_mode_name
```

- 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_name" -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_name" -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_name" -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_name" -exclude_db_info
```

### 削除

- 1 つまたは複数の転送セットを削除する。

```
ccm dcm -delete -ts "transfer_set_name"
```

### 生成

- `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.
```

### 生成と転送

- `Visual interface include files` 転送セットと `CA` データベースの転送パッケージを作成して転送する。

```
   ccm dcm -gen -ts "Visual interface include files" -dbid CA -trn
   Computing transfer package...
   Computing transfer package for 'Visual interface include files' going to
database 'CA'...
   123 objects will be included in transfer package for 'Visual interface
   include files' going to database 'CA'...
   Generating transfer package...
   ...
   Transferring package...
   Transfer successful, clean up in progress...
   Sending transfer package status email...

   DCM generate-transfer email notification has been sent to:
   dcmadmin@company.comUpdating database...
   DCM Generate completed successfully.
```

### 生成、転送、および受取り

- `Visual Interface Project` 転送セットと `CH` データベースの転送パッケージの生成、送出および自動受取りを行う。

```
   ccm dcm -gen -ts "Visual Interface Project" -dbid CH -trn -rec

   Computing transfer package...
   Computing transfer package for 'Visual Interface Project' going to
database 'CH'...
   335 objects will be included in transfer package for 'Visual Interface
Project' going to database 'CH'...
   Generating transfer package...
   Creating transfer package information files...
   DCM transfer information saved to file '
   ¥¥ccmsrv¥ccmdbs¥appdevdb¥dcm¥generate¥CA#7#CH#865893092#dcm_info.txt'
   DCM transfer preview information saved to file
'¥¥ccmsrv¥ccmdbs¥appdevdb¥dcm¥generate¥CA#7#CH#865893092#dcm_preview.txt'
   Creating transfer package...
   Creating transfer package for 'Visual Interface Project' going to
database CH...
   Compressing transfer data...
   Compressing transfer data for transfer set 'Visual Interface Project' and
database CH...
   Cleaning up temp files...
   DCM data generated to file
   '¥¥ccmsrv¥ccmdbs¥appdevdb¥dcm¥generate¥CA#7#CH#865893092.tar.gz'
   Transferring package...
   Transfer successful, clean up in progress...
   Doing remote receive...
   Starting receive remotely. Progress will not update until the receive
completes.
```

```
   Executing command '¥¥dbsrv¥ccm¥bin¥util¥ccm_receive -h dbsrv -d
¥¥dbhost¥ccmdbs¥visystem -dbid CA -ts Visual Interface Project
ccm_home¥¥dbsrv¥ccm'
   Receiving all transfer sets 'Visual Interface Project' from database
'CA'...
   Receiving transfer package 'Visual Interface Project' from database CA...
   Receiving 1 of 1 transfer packages...
   Decompressing data generated 06-09-97 14:51:32 PDT for transfer set
'Visual Interface Project' and database CA...
   Extracting data...
   Extracting data generated 06-09-97 14:51:32 PDT for transfer set Visual
Interface Project' and database CA...
   Importing data...
   Receive complete for transfer package 'Visual Interface Project' from
database CA.
   DCM Receive completed successfully.
   Rational Synergy engine exiting.

   Mon Jun  9 14:51:53 2004

   Receive DCM data done. Remote receive complete
   Sending transfer package status email...
   DCM generate-transfer email notification has been sent to:
   dcmadmin@company.com
   Updating database...
   DCM Generate completed successfully.
```

## 初期化

- データベース ID `jfil` とデフォルトの DCM 区切り文字（「`#`」）を使用して現在のデータベースを初期化する。

  初期化コマンドに以下を含めます。

  — データベースの説明。

  — データベースを所有するサイトの地理的な位置。

  — DCM 管理の問題の担当者の連絡先の情報。

  他のデータベースを使って複製するとき、上記のデータを入力することにより、意味のあるデータで実装される `description`、`location`、`admin_info` のフィールドを持つ DCM データベース定義が作成されます。

```
   ccm dcm -init -dbid jfil -description "Development database for
ProductXYZ" -location "Los Angeles, California" -admin_info "Jane Smith,
(206) 555-9090, JSmith@xyz.com"


   ...progress messages...
```

```
DCM database conversion is complete with no errors.
4 objects had attributes updated.
No objects had directory entries updated.
Your Rational Synergy session must be restarted.
Rational Synergy engine exiting.
```

### 設定の修正

- 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 -modify -settings -event_log_size log_size -
no_of_old_generate_times old_generate_times
```

### 受取り

- 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 -rec -dir /vol/dbserver1/dcm/receive/ -ts "transfer_set_name" -
dbid src_database_ID
```

### 表示

- 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

66 Mon Jul 22 15:16:53 2002 Receive      Successful E   eproj
65 Mon Jul 22 15:15:22 2002 Receive      Successful E   eproj
64 Wed Jul 17 15:27:13 2002 Receive      Successful K   All projects and
related objects for Release rename/1.0 saved on Wed Jul 17 15:23:45 2002
63 Wed Jul 17 15:23:45 2002 Save Offline Successful Any All projects and
related objects for Release rename/1.0 saved on Wed Jul 17 15:23:45 2002
62 Fri Jul 12 14:53:00 2002 Receive      Successful K3  M12251
61 Fri Jul 12 14:50:55 2002 Generate     Successful K3  M12251
60 Fri Jul 12 12:30:44 2002 Receive      Successful E   task completed_in
test
59 Fri Jul 12 12:29:48 2002 Receive      Successful E   task completed_in
test
58 Thu Jun 27 17:42:49 2002 Generate     Successful foo foo
57 Thu Jun 27 16:09:10 2002 Generate     Failed     foo foo
56 Thu Jun 27 15:25:13 2002 Generate     Cancelled  foo foo
55 Thu Jun 27 15:23:19 2002 Generate     Successful foo foo
54 Thu Jun 27 15:22:11 2002 Generate     Cancelled  foo smallexport
53 Wed Jun 26 13:13:42 2002 Generate     Cancelled  K3  R17951
52 Wed Jun 26 13:12:07 2002 Generate     Cancelled  K3  R17951
51 Wed Jun 26 13:10:53 2002 Generate     Successful K3  R17951
50 Wed Jun 26 13:10:20 2002 Generate     Cancelled  K3  R17951
49 Wed Jun 26 13:09:57 2002 Generate     Cancelled  K3  R17951
```

```
48 Tue Jun 25 14:53:48 2002 Generate     Successful K3  smallexport
47  Wed Jun 19 16:08:20 2002 Generate     Successful K3  jre
46  Wed Jun 19 16:08:05 2002 Generate     Successful K3  jre
45  Wed Jun 19 16:06:57 2002 Generate     Failed     K3  jre
44  Fri Jun 14 15:18:50 2002 Receive      Successful E   skipback
43  Fri Jun 14 15:17:27 2002 Generate     Successful E   skip
42  Fri Jun 14 15:02:32 2002 Generate     Successful E   skip
41  Fri Jun 14 10:18:14 2002 Generate     Successful E   jre
40  Thu Jun 13 18:37:15 2002 Generate     Successful E   jre
39  Thu Jun 13 16:08:20 2002 Receive      Started    E   eproj
38  Thu Jun 13 15:30:37 2002 Generate     Successful E   jre
37  Thu Jun 13 13:31:42 2002 Generate     Successful E   jre
36  Mon Jun 10 23:27:56 2002 Save Offline Successful Any Projects named ccm
on Mon Jun 10 23:27:56 2002
35  Mon Jun 10 23:23:29 2002 Save Offline Successful Any Projects named ccm
on Mon Jun 10 23:23:29 2002
34  Thu Jun 06 15:31:19 2002 Generate     Successful K3  test subproject
rename
33  Thu Jun 06 15:29:04 2002 Generate     Successful K3  test subproject
rename
32  Thu Jun 06 13:55:33 2002 Generate     Successful K3  test subproject
rename
31  Tue Jun 04 16:15:29 2002 Generate     Successful K3  folder only
30  Tue Jun 04 16:05:47 2002 Generate     Started    K3  folder only
29  Thu May 30 19:34:15 2002 Generate     Successful K3  testwa
28  Thu May 30 19:32:59 2002 Generate     Successful K3  testwa
27  Thu May 30 19:30:20 2002 Generate     Successful K3  testwa
```

## 転送

- すべての転送セットの転送パッケージを BST データベースに送る。

  ```
  ccm dcm -transfer -dbid BST
  Transferring all transfer packages with destination database 'BST'...
  Transferring 'Secure transformer layer' generated on 06-09-97 13:48:32
  PDT to database 'BST'.
  Sending transfer package status email...
  DCM generate-transfer email notification has been sent to:
  dcmadmin@company.com
  Transfer completed successfully.
  ```

- RDBMS Server API 転送セット（CH データベース用に生成された）の保存転送パッケージを転送する。

  ```
  ccm dcm -transfer -ts "RDBMS Server API"
  Transferring all transfer packages 'RDBMS Server API'...
  Transferring 'RDBMS Server API' generated on 06-09-97 14:42:20 PDT to
  database 'CH'.
  Sending transfer package status email...
  ```

```
DCM generate-transfer email notification has been sent to:
dcmadmin@company.com
Transfer completed successfully.
```

# **delete** コマンド

## 表記

```
ccm del|delete [-repl|-replace] [-scope delete_scope]
               [-t|-task task_number]
               file_spec [file_spec...]
ccm del|delete [-repl|-replace] -force file_spec [file_spec...]
ccm del|delete [-r|-recurse] [-repl|-replace]
               [-t|-task task_number]
               file_spec [file_spec...]
ccm del|delete [-repl|-replace] [-scope delete_scope]
               -p|-project project_spec [project_spec...]
ccm del|delete [-r|-recurse] [-repl|-replace]
               -p|-project project_spec [project_spec...]
ccm del|delete [-r|-recurse] [-repl|-replace] [-h|-hierarchy]
               [-t|-task task_number] [file_spec...]
ccm del|delete [-r|-recurse] [-repl|-replace] [-h|-hierarchy]
               -p|-project project_spec [project_spec...]
```

## 説明と用途

`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できます。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。

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。初期化ファイル内の `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）。
>
> 詳細については、共有プロジェクトを参照してください。
>
> 注意！ 削除操作は恒久的です。

## オプションと引数

*file_spec*

削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。

-force

-force オプションは、確認メッセージを抑止し、削除操作を強制的に実行します。

-h|-hierarchy

プロジェクト階層全体を削除します。-recurse オプションと一緒にのみ使用します。

-p|-project *project_spec*

指定したプロジェクトを削除します。

-r|-recurse

コマンドの対象がディレクトリまたはプロジェクトの場合、このオプションは、ディレクトリまたはプロジェクトを含むすべてのオブジェクトを削除し、削除できなかったオブジェクトはフローティング オブジェクトとなります。

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。

- -recurse *file_spec* または -recurse -hierarchy *file_spec* は、現在のプロジェクト内の指定されたファイルを削除する （*file_spec* がディレクトリではない場合、Rational Synergy は -recurse と -hierarchy の両方を無視します)。
- -recurse *file_spec*（*file_spec* はディレクトリ）は、サブプロジェクトを除く、ディレクトリ内のすべてのオブジェクト バージョンを削除する。
- -recurse -hierarchy *file_spec*（*file_spec* はディレクトリ）は、指定したディレクトリの下にある、サブプロジェクトを含むすべてのオブジェクト バージョンを削除する。
- -recurse -project *project_spec* は、サブプロジェクトを除く、プロジェクト内のすべてのオブジェクト バージョンを削除する。
- -recurse -hierarchy -project *project_spec* は、指定したプロジェクトの下にある、サブプロジェクトを含むすべてのオブジェクト バージョンを削除する。

*project_spec* と一緒に使用した場合、-recurse オプションは、-scope project_and_non-project_members オプションで削除の範囲を定義することと同じです。-recurse -hierarchy オプションは、-scope entire_project_hierarchy オプションで削除の範囲を定義することと同じです。

`-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` *`delete_scope`*

削除の範囲を指定します。有効な *`delete_scope`* 値は以下のとおりです。

- `project_only`
- `project_and_non-project_members`
- `project_and_subproject_hierarchy`
- `entire_project_hierarchy`

ディレクトリ オブジェクトの有効な値は以下のとおりです。

- `directory_only`
- `directory_and_non-project_members`
- `entire_directory_hierarchy`

`-t|-task` *`task_number`*

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。

## 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。

```
ccm delete sort.c
Member sort.c-1 deleted from project ico_proj-1
```

関連トピック

- create コマンド
- unuse コマンド
- use コマンド

# **delimiter** コマンド

## 表記

```
ccm delim|delimiter [value]
```

## 説明と用途

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。プロジェクトの初期ワークエリアを作成するとき、区切り文字を使用して、バージョンとプロジェクト名を区切ることもできます。

*ccm_admin* ロールのユーザーは、`delimiter` コマンドを使用して区切り文字の値を変更できます。デフォルトはダッシュ（`-`）です。区切り文字は非制限文字に設定できます（コマンドと引数の構文参照）。また、命名制限も参照してください。区切り文字を設定すると、その区切り文字は Rational Synergy データベースに対して設定されます。

区切り文字を変更する理由は、Rational Synergy データベース内のオブジェクト名に使用されている文字の使用を避けることにあります。*ccm_admin* ロールのユーザーは、デフォルトの区切り文字が名前の一部にダッシュが含まれるオブジェクトと衝突していないことを確認してください。衝突している場合は、ユーザーがデータベースの使用を開始する前にデフォルトの区切り文字を変更します。

> 注記：Rational Synergy の管理下でソフトウェアをマイグレートする前に、データベースの区切り文字を変更してください。区切り文字はいつでも変更できますが、プロジェクトがデータベースに存在するときに変更すると、その区切り文字を含むプロジェクトのワークエリア パスを変更する必要が生じます。
>
> 区切り文字を変更し、インターフェイスを再起動すると、その後で作成する新しいプロジェクトには設定した新しい区切り文字が使用されます。既存のプロジェクトのワークエリア パスには影響しません。
>
> 注意！ 区切り文字の変更はワークエリアに影響することがあります。詳細については、work_area コマンドを参照してください。

このコマンドを使用するには、*ccm_admin* ロールを持っている必要があります。

### 区切り文字の制限

一部の文字は、`ccm delimiter` コマンドによって、区切り文字としての使用することが禁止されます。これらの文字は 命名制限に記載されています。

区切り文字が制限文字かどうかを制御できます。非プロジェクト オブジェクトの名前の制限の変更の詳細については、allow_delimiter_in_name を参照してください。区切り文字の、バージョン、タイプ、インスタンス、プロジェクトでの使用については、制限があります。

## オプションと引数

*`value`*

データベース全体の区切り文字として使用される新しい文字を指定します。

## 例

- 表示名、オブジェクト名、ワークエリア　パスで、区切り文字を使用して名前とバージョンを分離する。たとえば、プロジェクトのワークエリアは次の場所にある場合があります。

  Windows :
  ```
  c:¥users¥linda¥ccm_wa¥ccmint22¥hello-linda
  ```
  UNIX :
  ```
  ~linda/ccm_wa/ccmint22/hello-linda
  ```

  以下のように `delim` コマンドを使用して、ファイル名とバージョンの間の区切り文字をカンマに変更できます。

  ```
  ccm delim ","
  ```

  インターフェイスを再起動して新しいプロジェクトを作成すると（例、`goodbye,linda`）、プロジェクトのワークエリアの場所は以下のようになります。

  Windows :
  ```
  c:¥users¥linda¥ccm_wa¥ccmint22¥goodbye,linda
  ```
  UNIX :
  ```
  ~linda/ccm_wa/ccmint22/goodbye,linda
  ```

- 自分のワークエリアに `csrc` ファイルのバージョン 3（`poly.c,3`）があるにもかかわらず、`poly.c,2` のワークエリア バージョンを参照したいとします。区切り文字がカンマに設定されている場合、以下のようにファイルを指定します。

  Windows :
  ```
  ccm dir poly.c,2
  integrate mary Aug 01 08:07 csrc 1 poly.c,2
  ```
  UNIX :
  ```
  ccm ls -l poly.c,2
  integrate mary Aug 01 08:07 csrc 1 poly.c,2
  ```

区切り文字としてダッシュを使用してファイルを指定すると、以下のエラー メッセージが表示されます。

Windows :

```
ccm dir poly.c-2
Referenced object version could not be identified: 'poly.c-2'
```

UNIX :

```
ccm ls -l poly.c-2
Referenced object version could not be identified: 'poly.c-2'
```

関連トピック

- work_area コマンド

# **depend** コマンド

## 表記

```
ccm depend -a|-append dependency_file [-f makefile]

ccm depend -w|-write dependency_file [-f makefile]

ccm depend -d|-delete [-f makefile]

ccm depend -s|-show [-f makefile]
```

## 説明と用途

depend コマンドにより、サードパーティの *make* ツールを使用して make ファイルの依存関係を更新できます。

最初に、サードパーティのツールを使用して、make ファイル フォーマットの依存関係を生成します。次に、depend コマンドを実行して、生成した依存関係を管理下にある make ファイルの依存属性にコピーします。

> 注記：-f オプションを使用しない場合、Rational Synergy は現在のディレクトリから有効な管理されている make ファイル（大文字、小文字、または大文字／小文字混合の「makefile」または「makefile.mk」）を検索し、make ファイル オブジェクトの依存属性に対して操作を実行します。

どのユーザーでもこのコマンドを実行できます。

## オプションと引数

-a|-append

*dependency_file* 内の依存関係を make ファイルに関連付けられている既存の依存関係に追加します。

-d|-delete

make ファイルの依存属性から依存関係を削除します。

*dependency_file*

依存関係を格納しているファイルの名前を指定します。

`-f makefile`

依存関係のコピー先となる make ファイルの名前を指定します。

`-s|-show`

make ファイルの依存属性を表示します。Rational Synergy はデフォルトの表示方法を使用して属性を表示します。

`-w|-write`

依存関係を make ファイルの *dependency_file* に保存します。このオプションは、make ファイルの既存の依存関係を上書きします。

## 例

- 依存関係を *make* ツールを使用して生成し、次にその属性を *makefile_name* に関する依存属性に保存する。

  Windows :
  ```
  mytool foo.c > depend_file
  ccm depend -w depend_file -f makefile_name
  del depend_file
  ```

  UNIX :
  ```
  mytool foo.c > depend_file
  ccm depend -w depend_file -f makefile_name
  rm depend_file
  ```

- *makefile_name* に関連付けられている依存関係を削除する。

  ```
  ccm depend -d -f makefile_name
  ```

- 新しい一組の依存関係を *depend_new_file* から *makefile_name* に追加する。

  ```
  ccm depend -a depend_new_file -f makefile_name
  ```

関連トピック

-

# **diff** コマンド

## 表記

```
ccm diff [-vc|-versioncompare] file_spec1 file_spec2
ccm diff [-vc|-versioncompare]
         -p|-project project_spec1 project_spec2
```

## 説明と用途

`diff` コマンドにより、ファイル間、ディレクトリ間またはプロジェクト間の差分を表示します。

このコマンドは、ソース比較（デフォルト）と バージョン比較の 2 つのタイプの比較に使用します。

### 比較

ソース ファイル間､ディレクトリ間またはプロジェクト間の差分を表示します｡ディレクトリについて `diff` コマンドを実行すると、バージョン付けられていないメンバーのリストの比較を実行します｡プロジェクトについては、バージョン付きメンバーのリストを比較します。

### バージョン比較

ソースと見なされないファイル、ディレクトリおよびプロジェクトの他の属性、たとえば、`create_time`、`modify_time`、`name`、`version` などを比較します。

## オプションと引数

*`file_spec1`*

比較の対象となる一方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`*

比較の対象となるもう一方のファイルまたはディレクトリの名前を指定します。

`-g`

比較するオブジェクトのタイプに応じて、グラフィカルな比較ツールまたは適切なダイアログが起動されます｡差分の表示対象となるオブジェクトの名前とバージョンを入力します。これを入力しないと、エラー メッセージが表示されます。

`-nogui` オプションを使用してセッションを開始した場合、オブジェクト間の差分はグラフィカルに表示されません。

`-p|-project`

プロジェクト間の差分を表示します。

*`project_spec1`*

比較の対象となる一方のプロジェクトの名前を指定します。

*`project_spec2`*

比較の対象となるもう一方のプロジェクトの名前を指定します。

`-vc|-versioncompare`

ソース比較ではなく、バージョン比較を実行します （ソース比較がデフォルトです)。

## 例

- `draw.c` の現バージョンと元バージョンを比較する。

  ```
  ccm diff draw.c draw.c-1
  ```

- `tools` ディレクトリと `my_tools` ディレクトリを比較する。

  ```
  ccm diff tools my_tools
  ```

- `rel-test3.1` プロジェクトと `rel-joe_3.1` プロジェクトを比較する。

  ```
  ccm diff -p rel-test3.1 rel-joe_3.1
  ```

- `draw.c-4` と `draw.c-5` のバージョン比較を行う。

  ```
  ccm diff -vc draw.c-4 draw.c-5
  ```

- `rel-test3.1` プロジェクトと `rel-joe_3.1` プロジェクトのバージョン比較を行う。

  ```
  ccm diff -vc -p rel-test3.1 rel-joe_3.1
  ```

関連トピック

- merge コマンド

# **dir** コマンド

## 表記

```
ccm dir [-m] [-w] [-s] [-f|-format "format_string"] [file_spec...]
```

## 説明と用途

`dir` コマンドは、Windows オペレーティング システムでのみ機能します。

`dir` コマンドは、ディレクトリの内容をワークエリアに一覧表示します。オプションを付けずにこのコマンドを入力すると、オブジェクトが長いフォーマットで表示されます。長いフォーマットには、状態、所有者、最後の修正時刻、タイプ、インスタンス、名前、バージョン、およびタスクが含まれます。

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。

`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。これらのファイルの表示方法については、`-w` オプションと `-m` オプションを参照してください。

## オプションと引数

`-f|-format "format_string"`

出力のフォーマットを指定します。このフォーマットは管理ファイルにのみ適用されます。必須文字列には、以下のようにキーワードと文字テキストを使用します。

```
%displayname %owner
```

キーワードには、組み込まれたもの（`%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。

フォーマット文字列内に `%path` を指定すると、すべてのオブジェクトは絶対ワークエリア パス付きで表示されます。ワークエリアが見えない場合は、パスが計算されます。

*`file_spec`*

表示するファイルを指定します。

`-m`

管理ファイルと非管理ファイルを表示します。非管理ファイルとは、ワークエリア内に存在するが、プロジェクトに含まれていないファイルのことです。

このオプションを使用すると、適切な場合には以下の記号が出力の前に表示されます。

- LC（ローカル コピー）

プロジェクト内にあって、しかもコピーベースのワークエリア内にもあるファイルを意味します（すべての Windows クライアント ユーザーはコピーベースのワークエリアで作業します）。

- NS（同期外）

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。この状況は、ファイルをプロジェクト内で作成したが、同期してワークエリアに出力しなかった場合、またはワークエリアのコピーが削除された場合に起こります。

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。リコンサイル機能の詳細については、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 -w オプションを付けて使用する必要があります。

`-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。

`-w`

ファイル名とバージョンのみを、フォーマットされてないカラムに一覧表示します。

## 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) working  bill Jan 11 18:09        csrc 1 cli.c-2 543
```

```
(UC) working  bill Jan 11 18:09        csrc 1 input.c-1 432
(UC) working  bill Jan 11 18:09        csrc 1 simple.c-2 543
(UC) working  bill Jan 11 18:09        csrc 1 main.c-1 432
(UC) drwx        0 Feb 02 11:11        images
     released   bob Jan 11 18:11          dir 1 scripts-2 440
```

- 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
(LC)  integrate  paul  Dec 24  16:48  makefile    4  Makefile-1
(LC)  integrate  paul  Dec 24  16:48      csrc    1  callback.c-1
(LC)  integrate  paul  Dec 24  16:46      csrc    1  error.c-1
(LC)  integrate  paul  Dec 24  16:46      csrc    1  gui.c-1
(LC)  integrate  paul  Dec 24  16:46      csrc    1  info.c-1
(LC)  integrate  paul  Dec 24  16:47      csrc    1  init.c-1
(LC)    working  paul  Dec 24  17:17      csrc    2  main.c-3
(LC)  integrate  paul  Dec 24  16:48      csrc    1  output.c-1
(LC) integrate   paul  Dec 24  16:49      csrc    1  release.c-1
```

- 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 ann Jun 19  2003      dir  J#1 include,2 J#5565
(LC) integrate pat Jan 26 15:41 makefile  J15 Makefile.pc,#7 J#6103
  released    ann Jan 16  2001      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 joe Mar 27  2003 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"

VersionedObject.java,10 java C:¥joe¥ccm_wa¥ccm_java¥objectapi-
joe¥objectapi¥src¥com¥telelogic¥cm¥objectapi¥VersionedObject.java
```

# **edit** コマンド

## 表記

```
ccm edit file_spec [file_spec...]
```

## 説明と用途

`edit` コマンドにより、指定したソース ファイルを編集できます。

このコマンドを使用して、現在ディレクトリのメンバーではないファイルのバージョンを表示するか、またはそのオブジェクトに対してタイプベースのエディタを起動します。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。

> 注記：ファイルを編集するとき（Windows）、またはファイルをコピーベースのワークエリアで編集するとき（UNIX）、編集による更新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。したがって、ファイルの編集と保存を行うときは、ワークエリアを定期的にリコンサイルしてください。

## オプションと引数

*`file_spec`*

編集するファイルを指定します。

## 例

`log.c` ファイルのバージョン 8 を編集します。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
- typedef コマンド

- cli.text_editor
- cli.text_viewer

# **expand** コマンド

## 表記

```
ccm expand [-c] [-s suffix] makefile [makefile...]
ccm expand -a|-all [-c] [-s suffix]
```

## 説明と用途

expand コマンドは、Rational Synergy Makefile を独立した make ファイルに変換します。

## オプションと引数

`-a|-all`

現在のプロジェクト内の `makefile` タイプのオブジェクトをすべて展開します。

`-c`

展開した make ファイルを Rational Synergy の下で管理します。

*`makefile`*

展開する 1 つまたは複数の make ファイルを指定します。`-a` オプションを使用している場合は、make ファイルを指定する必要はありません。

`-s` *`suffix`*

展開後の make ファイルの接尾辞を指定します（デフォルトの接尾辞は `.out` です）。`%platform` 文字列が接尾辞に含まれる場合、expand はその文字列をプロジェクトまたは make ファイルからの `platform` 属性に置き換えます。make ファイルの `platform` 属性は、プロジェクトの `platform` 属性に優先します。

## 例

- `Makefile` という名前の make ファイルから独立した make ファイルを生成する。

  ```
  ccm expand Makefile -s
  ```

展開後の make ファイルのデフォルトの名前は *`makefile_name`*`.out` です。

# **export** コマンド

## 表記

```
ccm export [-r|-recurse] [-t|-to pathname] [-q|-quiet]
           [-delimiter delimiter_value]
           file_spec [file_spec...]
ccm export [-r|-recurse] [-t|-to pathname] [-q|-quiet]
           [-delimiter delimiter_value]
           [-leave_wa_paths|-lwp]
           -p|-project project_spec [project_spec...]
```

## 前提条件

*ccm_root* ユーザーはエクスポート ディレクトリへの書き込みが可能である必要があります。これは、Rational Synergy エンジン プロセスがエクスポートを実行し、そのプロセスが *ccm_root* ユーザーとして実行されるからです。

## 説明と用途

`export` コマンドは、Rational Synergy インポート／エクスポート フォーマットで、Rational Synergy データベースからファイル システムへオブジェクトをエクスポートします。

どのユーザーでもこのコマンドを実行できます。

注記: エクスポート ディレクトリはエンジン ホストからは見える必要があります。

`-t` オプションを指定しなかった場合、オブジェクトは現在の作業ディレクトリにエクスポートされます。

`-p` オプションを指定した場合、プロジェクト オブジェクトとそのルート ディレクトリがエクスポートされます。

`-r` オプションを指定し、エクスポートの対象として指定したオブジェクトがプロジェクトまたはディレクトリの場合、プロジェクトまたはディレクトリ階層で使用されるすべてのオブジェクトがエクスポートされます。

## オプションと引数

`-delimiter` *`delimiter_value`*

オブジェクトの 4 部名称の各部を区切る区切り文字を指定します。現在のデータベースと異なる区切り文字を使用するデータベースにエクスポートする場合に、このオプションを使用します。たとえば、4.4 UNIX データベースから 4.2.1 UNIX データベースにエクスポートするには、*`delimiter_value`* にコロンを使用します。`-nid` オプションはこのオプションと一緒に使用する必要があります。

デフォルトは「@」です。

*`file_spec`*

エクスポートするファイルまたはディレクトリの名前を指定します。

`-leave_wa_paths|-lwp`

エクスポートするオブジェクトの現在のデータベースでのワークエリア パスを維持します。

そのオブジェクトが新しいデータベースにインポートされるとき、そのワークエリア パスは前のデータベース パスに設定されます。

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

エクスポートするプロジェクトを指定します。

*`project_spec`*

エクスポートするプロジェクトの名前を指定します。

`-q|-quiet`

コマンドのメッセージ出力を抑止します。

`-r|-recurse`

ディレクトリまたはプロジェクト階層のメンバーであるオブジェクトのすべてを再帰的にエクスポートします。

`-t|-to` *`pathname`*

指定したオブジェクトのエクスポート先のディレクトリ パスを指定します。リモート クライアント（Windows）またはローカル クライアント（UNIX）をエクスポー

トする場合、このオプションを使用する必要があります。指定するパスはエンジンから見える状態にあり、*ccm_root* が書き込み可能である必要があります。

## 例

Windows：
`c:¥users¥doug¥cm_test_db¥job-1¥job` ディレクトリ内のワークエリアから、以下のいずれかを実行できます。

- `src¥foo.c` オブジェクトを `/users/doug/export_dir` ディレクトリにエクスポートする。

  ```
  ccm export -to ¥¥users¥doug¥export_dir src¥foo.c
  ```

- `job-1` プロジェクトを `/users/doug/export_dir` ディレクトリに再帰的にエクスポートする。

  ```
  ccm export -to ¥¥users¥doug¥export_dir /recurse -p job-1
  ```

- `src` ディレクトリ オブジェクトを `/users/doug/export_dir` ディレクトリにエクスポートする。

  ```
  ccm export -to ¥¥users¥doug¥export_dir src
  ```

- `src` ディレクトリ オブジェクトとそのディレクトリの下にあるすべてのオブジェクトを `/users/doug/export_dir` ディレクトリにエクスポートする。

  ```
  ccm export -to ¥¥users¥doug¥export_dir -recurse src
  ```

- `src¥foo.c-1` オブジェクト バージョンを `/users/doug/export_dir` ディレクトリにエクスポートする。

  ```
  ccm export -to ¥¥users¥doug¥export_dir foo.c-1@csrc@1
  ```

UNIX：
`/users/doug/cm_test_db/job-1/job` ディレクトリ内のワークエリアから、以下のいずれかを実行できます。

- `src/foo.c` オブジェクトを `/users/doug/export_dir` ディレクトリにエクスポートする。

  ```
  ccm export -to /users/doug/export_dir src/foo.c
  ```

- `job-1` プロジェクトを `/users/doug/export_dir` ディレクトリに再帰的にエクスポートする。

  ```
  ccm export -to /users/doug/export_dir -recurse -p job-1
  ```

- `src` ディレクトリ オブジェクトを `/users/doug/export_dir` ディレクトリにエクスポートする。

  ```
  ccm export -to /users/doug/export_dir src
  ```

- `src` ディレクトリ  オブジェクトとそのディレクトリの下にあるすべてのオブジェクトを `/users/doug/export_dir` ディレクトリにエクスポートする。

  ```
  ccm export -to /users/doug/export_dir -recurse src
  ```

- `src/foo.c-1` オブジェクト バージョンを `/users/doug/export_dir` ディレクトリにエクスポートする。

  ```
  ccm export -to /users/doug/export_dir foo.c-1:csrc:1
  ```

  注記：デフォルトの区切り文字は `@`（アット マーク）です。

## 警告

このコマンドはリモート クライアント（Windows）またはローカル クライアント（UNIX）から実行できますが、エンジン プロセスから見える状態にある、*ccm_root* ユーザーが書き込み可能なパスにのみオブジェクトをエクスポートできます。

関連トピック

- import コマンド

# **finduse** コマンド

## 表記

```
ccm finduse [finduse_scope_spec] file_spec

ccm finduse -task [finduse_scope_spec] task_id [task_id...]

ccm finduse -fol|/folder [finduse_scope_spec] folder_id [folder_id...]

ccm finduse -baseline [baseline_spec]

ccm finduse [-q|-query "query_expression"]
            [-n|-name object_name] [-o|-owner owner]
             [-s|-state state] [-t|-type type]
             [-v|-version version] [-i|-instance instance]
             [finduse_scope_spec]  [-p project_spec]
```

## 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を検索し、そのオブジェクトが使用されている箇所を識別する参照仕様のリストを返します。

## オプションと引数

`-apg|-all_project_groupings`

すべてのプロジェクト グルーピング内のオブジェクトの使用箇所を検索します。

`-baseline` *`baseline_spec`*

指定したベースラインを使用するプロジェクト グルーピングを表示します。デフォルトの適用範囲は、指定したベースラインを使用するすべてのプロジェクト グルーピングを表示する `-all_project_groupings` です。ベースラインは、それがプロジェクト グルーピングのベースラインの場合、プロジェクト グルーピング内で「使用」されていると見なされます。ベースラインを表示するには、プロジェクト グルーピングの **Properties** ダイアログか、または `ccm project_grouping -show baseline` コマンドを使用します。

プロジェクト グルーピングに関する各種適用範囲により、通常の方法で返されるプロジェクト グルーピングが制限されます。それらの適用範囲は以下のとおりです。

```
-all_project_groupings
-working_project_groupings
```

```
-my_project_groupings
-prep_project_groupings
-shared_project_groupings
```

*file_spec*

使用箇所の検索対象とするファイル、ディレクトリ、またはプロジェクトの名前を指定します。

*finduse_scope_spec*

検索の範囲を指定します。有効な *finduse_scope_spec* 値は以下のとおりです。

```
[-working_proj]
[-shared_proj]
[-prep_proj]
[-released_proj]
[-all_proj]
[-personal_folder]
[-shared_folder]
[-prep_folder]
[-non_write_folder]
[-all_folder]
[-all_baseline|-all_baselines]
[-wpg|-working_project_groupings]
[-mpg|-my_project_groupings]
[-ppg|-prep_project_groupings]
[-spg|-shared_project_groupings]
[-apg|-all_project_groupings]
```

*finduse_scope_spec* を指定しない場合、`-all_proj` が使用されます。

`-f|-folder` *folder_id*

更新プロパティの *folder_id* で、指定するフォルダを持つすべてのプロジェクトを検索します。

`-i|-instance` *instance*

インスタンス番号 *instance* を持つオブジェクトを含むすべてのプロジェクトを検索します。

`-mpg|-my_project_groupings`

現在のユーザーが所有するすべての個人用プロジェクト グルーピングで、オブジェクトの使用箇所を検索します。

`-n|-name` *`object_name`*

*`object_name`* という名前を持つオブジェクトが含まれる、すべてのプロジェクトを検索します。

`-o|-owner` *`owner`*

*`owner`*`.` が所有するオブジェクトが含まれる、すべてのプロジェクトを検索します。

`-ppg|-prep_project_groupings`

すべての prep プロジェクト グルーピングで、オブジェクトの使用箇所を検索します。

`-p` *`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。

`-q|-query "`*`query_expression`*`"`

一連のオブジェクトを生成するクエリ式を指定し、次にその中の各オブジェクトの使用箇所を表示します。`-q` オプションは、`-p` オプションと一緒には使用できません。

`-spg|-shared_project_groupings`

すべての共有プロジェクト グルーピングで、オブジェクトの使用箇所を検索します。

`-s|-state` *`state`*

*`state`* という状態にあるオブジェクトが含まれる、すべてのプロジェクトを検索します。

`-t|-task` *`task_id`*

更新プロパティに指定した *`task_id`* を持つタスクを直接またはフォルダを通して使用している、すべてのプロジェクトを検索します。フォルダの適用範囲が `finduse_scope_spec` 使用して指定されている場合は、*`task_id`* というタスクを含むフォルダも検索します。

`-t|-type` *`type`*

*`type`* というタイプのオブジェクトが含まれる、すべてのプロジェクトを検索します。

`-v|-version` *`version`*

バージョンが *`version`* であるオブジェクトが含まれる、すべてのプロジェクトを検索します。

`-wpg|-working_project_groupings`

すべての作業中プロジェクト グルーピングで、オブジェクトの使用箇所を検索します。

## 例

- プロジェクト内で、display.c という名前のオブジェクト バージョンのすべての使用箇所を検索する。

```
ccm finduse -name display.c
display.c-1 integrate epresley csrc ico_proj 1
ico_proj/src/display.c-1@ico_proj-1
display.c-2 integrate epresley csrc ico_proj 1
ico_proj/src/display.c-2@ico_proj-int2
display.c-3 integrate epresley csrc ico_proj 1
Object not used in scope
display.c-4 integrate epresley csrc ico_proj 1
ico_proj/src/display.c-4@ico_proj-epresley
```

- 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、すべての使用を検索する。

```
ccm finduse draw.c
draw.c-1 integrate epresley csrc gditest EAP#3 EAP#274
    gditest/draw.c-1@gditest-org
    gditest/draw.c-1@gditest-shared2.1
    gditest/draw.c-1@gditest-visible2.1
    gditest/draw.c-1@gditest-int2.1
    gditest/draw.c-1@gditest-org1
```

- タスク 128 が含まれるすべてのフォルダを検索する。

```
ccm finduse -all_folders -task 128
Task EAP#128:  Correct color of icons
Folder EAP#3:  All Completed Tasks for Release 1.2
     Folder EAP#7:  bill's Completed Tasks for Release 1.2
     Folder EAP#8:  Integration Testing Tasks for Release 1.2
```

- 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
draw.c-2 integrate bill csrc draw_proj EAP#1 EAP#128
 Folder EAP#7:  bill's Completed Tasks for Release 1.2
     Folder EAP#9:  bill's Assigned or Completed Tasks for Release 1.2
```

- タスク 128 が含まれる *prep* プロジェクトを検索する。

```
ccm finduse -prep_proj -task 128
Task EAP#128:  Correct color of icons
draw_proj-int1.2
util_proj-int1.2
```

- 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
util_proj-bill
```

# **folder** コマンド

## 表記

### フォルダの比較

```
ccm folder -comp|-compare folder_spec1
           -un|-union |
           -int|-intersection |
           -not|-not_in
           [-f|-format "format_string"] [-ns|-no_sort] [-u]
           folder_spec2
```

### フォルダのコピー

```
ccm folder -cp|-copy folder_spec
           [-e|-existing existing_folder_spec [-append]] |
           [-new "new_folder_name"]
           [-y] [-q|-quiet]
```

### フォルダの作成

```
ccm folder -cr|-create -n|-name "folder_name"
           [-us|-usable usable_by]
           [-w|-writable writable_by]
           [-qu|-query] [-q|-quiet]
           query_spec
```

### フォルダの削除

```
ccm folder -delete [-y] folder_specs
           [-q|-quiet]
```

### フォルダの使用箇所の検索

```
ccm folder -fu|-find_use
           [-f|-format "format_string"] [-ns|-no_sort] [-u]
           folder_spec
```

### フォルダの一覧表示

```
ccm folder -l|-list [scope]
           [-f|-format "format_string"] [-ns|-no_sort] [-u]
```

### フォルダの修正

```
ccm folder -m|-modify
           [-at|-add_task|-add_tasks task_specs] [-related]|
```

```
          [-rt|-remove_task|-remove_tasks task_specs] [-related]|
          [-up|-update] |
          [-mode {man|manual | uq|use_query}] |
          ["query_expression"] |
          [-n|-name "name_string"] |
          [-us|-usable usable_by] |
          [-w|-writable writable_by]
          [-y] [-q|-quiet]
          folder_specs
```

フォルダ情報の表示

```
ccm folder -sh|-show
          i|info|information [-v|-verbose]|
          obj|objs|objects |
          t|task|tasks [-v|-verbose]
          [-f|-format "format_string"] [-ns|-no_sort] [-u]
          folder_specs
ccm folder -sh|-show
          mode |
          n|na|name |
          q|qu|query |
          u|us|usable |
          w|wr|writable
          folder_specs
```

## 説明と用途

`folder` コマンドにより、以下のタスク ベースの Rational Synergy 操作を実行します。

- 2 つのフォルダを比較する。
- フォルダをコピーする。
- フォルダを作成する。
- 1 つ以上のフォルダを削除する。
- フォルダを表示する。
- フォルダを修正する。
- フォルダの使用箇所を検索する。
- フォルダを一覧表示する。

## オプションと引数

`-at|-add_task|-add_tasks task_specs`

指定されたフォルダに 1 つまたは複数のタスクを追加します。

このオプションは、-modify オプションと一緒にのみ使用できます。また、タスクを追加するフォルダが書き込み可能である必要があります。

-append

ソース フォルダ（*folder_id1*）の内容をデスティネーション フォルダ（*folder_id2*）の内容に追加します。

-append オプションは、-copy および -existing オプションと一緒にのみ使用できます。

-comp|-compare *folder_spec1 folder_spec2*

指定したフォルダの内容を比較します。選択セットには、出力に一覧表示されるタスクが示されます。

-f オプションを使用して、コマンドの出力フォーマットを変更します。-u によって出力の自動番号付けを抑止し、-ns によってソートを抑止します。

folder -compare のデフォルト出力フォーマットは以下のとおりです。

```
Task %displayname: %task_synopsis
```

ここで、

DCM が無効な場合、%displayname は %name です。
DCM が有効な場合、%displayname は
<*database_ID*><*DCM_delimiter*><*task_number*> です。
%task_synopsis はタスクの説明です。

*folder_spec1* との相対的な差分が表示されます。実行できる比較のタイプは以下のとおりです。

- union（いずれかのフォルダにあるタスクを表示）
- intersection（両フォルダに共通タスクを表示）
- not_in（*folder_spec1* に存在するが、*folder_spec2* には存在しないタスクを表示）

コマンド実行が成功した場合は値 0 が返され、成功しなかった場合はゼロ以外の値が返されます。

-cp|-copy

*folder_spec* のすべてのフォルダ定義を新規フォルダ（-new "*new_folder_name*"）または既存のフォルダ（-e|-existing *existing_folder_spec*）にコピーします。また、append オプションを使用して、ソース フォルダ（*folder_id1*）の内容を既存のデスティネーション フォルダ（*folder_id2*）に追加できます。

このオプションを -q（消音モード）と一緒に使用して、作成済みのフォルダのフォルダ ID 以外の、コマンド出力を抑止します。

コマンド実行が成功した場合は値 0 が返され、成功しなかった場合はゼロ以外の値が返されます。

-cr|-create

指定したプロパティを持つフォルダを作成します。

このオプションを -q（消音モード）と一緒に使用して、フォルダ ID 以外のコマンド出力を抑止します。

このオプションは、name オプションと一緒に使用する必要があります。

コマンド実行が成功した場合は値 0 が返され、成功しなかった場合はゼロ以外の値が返されます。

-delete

指定したフォルダを削除します。-y オプションも指定した場合、確認メッセージを表示せずにフォルダを削除します。（-y オプションの「注意」を参照してください。）

削除するフォルダは、書き込み可能である必要があります。

このオプションを -q（消音モード）と一緒に使用して、削除されたフォルダ数以外のコマンド出力を抑止します。

コマンド実行が成功した場合は値 0 が返され、成功しなかった場合はゼロ以外の値が返されます。

`-e|-existing` *`existing_folder_spec`*

*`folder_spec`* からタスクをコピーする先の既存のフォルダを指定します。

コピー操作では、手動でタスクを追加するデスティネーション フォルダを設定します。

このオプションは `-copy` オプションと一緒にのみ使用できます。またデスティネーション フォルダが書き込み可能である必要があります。また、`-append` オプションを `-copy` および `-existing` オプションと一緒に使用して、ソース フォルダ（*`folder_id1`*）の内容をデスティネーション フォルダ（*`folder_id2`*）の内容に追加できます。

`-f|-format "`*`format_string`*`"`

コマンドの出力フォーマットを指定します。デフォルトのフォーマットは、`-format` と一緒に使用するオプション（`-finduse`、`-list`、または `-show`）、とこれらのオプションのキーワード引数によって異なります。デフォルト出力フォーマットについては各オプションの説明を参照してください。

このフォーマットには、テキストとキーワードの組み合わせを含むことができます。キーワードは、表示されるときに各オブジェクトについての特定の情報に置き換わります。たとえば、ユーザー *sue* が所有するオブジェクトが表示される場合は、キーワード `%owner` は `sue` に置き換わります。

任意の既存の属性名をキーワードとして使用できます。また、`%displayname`、`%task_number` など、多数のキーワードがあらかじめ定義されて組み込まれれています。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。

`-fu|-find_use`

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。

`-format` オプションを使用して、コマンドの出力フォーマットを変更します。`-u` によって出力の自動番号付けを抑止し、`-no_sort` によってソートを抑止します。

`folder -finduse` のデフォルト出力フォーマットは以下のとおりです。

```
%name %status %owner %version
```

ここで、

- %name はプロジェクト名です。
- %status はプロジェクトの状態です。
- %owner はプロジェクトの所有者です。
- %version はプロジェクトのバージョンです。

*folder_spec*

一覧表示、追加、削除または変更するフォルダの ID を指定します。複数のフォルダは、カンマまたは空白文字で区切ります。

この引数の構文については、フォルダの指定 を参照してください。

このオプションは、-folder_spec オプションと一緒に使用する必要があります。

-int|-intersection

指定した両フォルダに共通するすべてのタスクを、比較結果に表示するよう指定します。

このオプションは、-compare オプションと一緒にのみ使用できます。

-l|-list [*scope*]

すべてのフォルダ、または *scope* で指定されたフォルダを一覧表示します。

-format オプションを使用して、コマンドの出力フォーマットを変更します。-u によって出力の自動番号付けを抑止し、-no_sort によってソートを抑止します。

folder -list のデフォルト出力フォーマットは以下のとおりです。

*Folder %displayname: %description*

ここで、

DCM が無効な場合、%displayname は %name です。
DCM が有効な場合は、*database_ID* は、
*<database_ID><DCM_delimiter><task_number>* です。
%description はフォルダ名です。

*scope* 引数は、以下のいずれか 1 つの値を持つ必要があります。

- `all_personal`
- `all_build_mgrs`
- `all_shared`
- `all_non_writable`
- `all`

デフォルトでは、自分のすべての個人用フォルダが一覧表示されます。

コマンド実行が成功した場合は値 `0` が返され、成功しなかった場合はゼロ以外の値が返されます。

`-mode {man|manual | uq|use_query}`

フォルダの内容を手動（`manual`）、またはクエリを使用して（`use_query`）定義します。

このオプションは、`-modify` オプションと一緒にのみ使用できます。

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。

- 手動フォルダをクエリ　ベース　フォルダに変更し、そのフォルダが前はクエリ ベースであった場合は、最後のクエリを使用する。
- 手動フォルダをクエリ　ベース　フォルダに変更し、そのフォルダが前にクエリ ベースであったことがなく、さらにユーザー定義クエリ（default_task_query）が存在する場合、ユーザー定義クエリがクエリとなる。
- 手動フォルダをクエリ　ベース　フォルダに変更し、そのフォルダが前にクエリ ベースであったことがなく、ユーザー定義クエリ（default_task_query）が存在せず、さらにビルド　マネージャとして作業している場合、クエリは　`All Completed Tasks` となる。
- 手動フォルダをクエリ　ベース　フォルダに変更し、そのフォルダが前にクエリ ベースであったことがなく、ユーザー定義クエリ（default_task_query）が存在せず、さらにビルド マネージャとして作業していない場合、クエリは `All Tasks Assigned to` *`your_user_name`* となる。

`-m|-modify`

以下のサブオプションを任意の組み合わせで使用して、フォルダ プロパティを変更できます。

```
-at|-add_task task_specs [-related]
-rt|-remove_task|-remove_tasks task_specs [-related]
-up|-update
```

```
-mode {man|manual | uq|use_query}
query_spec
-n|-name "name_string"
-us|-usable usable_by
-w|-writable writable_by
-q|-quiet
folder_specs
```

`-modify` オプションには、複数のサブオプションを使用できます。

このオプションを使用して、フォルダのデフォルト スタイルまたはカスタム スタイル クエリの一部を変更できます。

フォルダ A が `All completed tasks for release 2.1` のデフォルト スタイル クエリを使用しているときに、そのフォルダに対して `ccm folder -modify -release 2.2` コマンドを実行すると、クエリは `All completed tasks for release 2.2` に変更されます。

フォルダ A がカスタム スタイル クエリを使用しているときに `ccm folder -modify` *`query_spec`* コマンドを実行すると、クエリが置き換えられます。また、`-custom` オプションを使用せずに、`-task_scope` オプションを使用すると、フォルダのクエリはデフォルト スタイルに変わります。

オプションを変更するフォルダは、書き込み可能である必要があります。

このオプションを `-quiet` と一緒に使用して、コマンドの出力を抑止します。`-y` オプションは、確認メッセージを抑止します。

`-add_task` オプションを使用して除外されたタスクをフォルダに追加すると、確認メッセージが表示されます。適切に応答して続行します。

コマンド実行が成功した場合は値 `0` が返され、成功しなかった場合はゼロ以外の値が返されます。

`-n|-name "`*`folder_name`*`"`

指定フォルダの名前を変更します。

このオプションは、-create または -modify オプションと一緒にのみ使用できます。また、名前変更するフォルダが書き込み可能である必要があります。

-new "*new_folder_name*"

*folder_spec* からフォルダ プロパティをコピーする先の新規フォルダ名を指定します。

コピー操作では、デスティネーション フォルダ *new_folder_name* が作成されるので、タスクは手動で追加します。

このオプションは、-cp オプションと一緒にのみ使用できます。

-not|-not_in

*folder_spec1* に含まれていて、*folder_spec2* に含まれていないすべてのタスクを、比較結果に表示するよう指定します。

このオプションは、-compare オプションと一緒にのみ使用できます。

-ns|-no_sort

コマンドの出力をソートしいなよう指定します。

-q|-quiet

コマンド実行時の出力メッセージ数を減らします。このオプションを指定すると、フォルダの作成時にフォルダ ID（例、3）のみが出力されます。

-qu|-query

フォルダをクエリ ベースにします。

このオプションは、-create オプションと一緒にのみ使用できます。

*query_spec* を使用しない場合、クエリは以下の条件に従って定義されます。

- ユーザー定義クエリ（default_task_query）がある場合、ユーザー定義クエリがクエリとなる。
- ユーザー定義クエリ（default_task_query）が存在せず、ビルド マネージャとして作業している場合は、クエリは All Completed Tasks となる。

- ユーザー定義クエリ（default_task_query）が存在せず、ビルド マネージャ以外として作業している場合は、クエリは `All Tasks Assigned to` *`your_user_name`* となる。

*`query_spec`*

指定フォルダにタスクを追加するために使用するクエリを変更します。

*`query_spec`* の構文は以下のとおりです。

```
[-cus|-custom "query_expression"]
[-db|-dbid|-database_id database_id]
[-plat|-platform platform]
[-rel|-release release]
[-sub|-subsystem subsystem]
[-ts|-scope|-task_scope task_scope]
```

ただし、*`task_scope`* は以下のいずれかのフォルダ名です。

`user_defined`
`all_my_assigned_or_completed` （デフォルト）
`all_my_assigned`
`all_my_completed`
`all_my_tasks`
`all_completed`
`all_tasks`

CLI 内のすべてのユーザー作成クエリは、カスタム クエリ（クエリ文節なし）として保存されます。指定したオプションに関連付けられた適用範囲（scope）は保存されません。

タスク スコープを指定しなかった場合、値 `all_my_assigned_or_completed` が使用されます。

*`query_expression`* は、query コマンド の *`query_expression`* と同じです。

このオプションは、`-create` または `-modify` オプションと一緒にのみ使用できます。

`related`

`-related` は、`-add_tasks` または `-remove_tasks` と一緒に使用して、*completed* の状態を持つすべてのタスクを 1 つまたは複数の指定フォルダに関連付けます。

`-rt|-remove_task` *`task_specs`*

指定されたフォルダから 1 つまたは複数のタスクを削除します。*`task_specs`* 引数の構文については、タスクの指定 を参照してください。

このオプションは、`-modify` オプションと一緒にのみ使用できます。また、修正するフォルダが書き込み可能である必要があります。

*`scope_spec`*

フォルダの一覧表示に使用するクエリを変更します。

*`scope_spec`* は以下のいずれか 1 つです。

```
all_personal | all_build_mgrs | all_shared | all_non_writable | all
```

`-sh|-show`

指定したフォルダのプロパティを表示します。このオプションを `info`、`objs` または `tasks` キーワードまたはそのバリアントと一緒に使用すると、選択セットには出力に一覧表示されるオブジェクトまたはタスクが含まれます。

`-format` オプションを使用して、コマンドの出力フォーマットを変更します。`-u` によって出力の自動番号付けを抑止し、`-no_sort` によってソートを抑止します。

以下のいずれか 1 つのキーワードを使用する場合は、`-format` オプションも使用できます。

```
i|info|information
obj|objs|objects
t|task|tasks
```

`-show info` を使用する場合は、`-verbose` オプションを使用して、すべてのフォルダとすべての関連タスク、それにタスクの影響を受けるすべてのオブジェクトを表示できます。`-show tasks` を使用する場合は、`-verbose` オプションを使用して、各フォルダのすべてのタスクおよびタスクの影響を受けるすべてのオブジェクトを表示できます。

`folder -show information` のデフォルト出力フォーマットは以下のとおりです。

```
Folder %displayname: %description
```

ここで、

DCM が無効な場合、*%displayname* は %name です。
DCM が有効な場合、*%displayname* は
*<database_ID><DCM_delimiter><task_number>* です。
*%description* はフォルダ名です。

これらの行の後に、いくつかのフォルダ プロパティに関する追加情報が表示されます。

`folder -show objects` のデフォルト出力フォーマットは以下のとおりです。

```
%objectname %status %owner %task
```

ここで、

`%objectname` はオブジェクトの *name-version:type:instance* です。
`%status` はオブジェクトの状態です。
`%owner` はオブジェクトの所有者です。
`%task` はオブジェクトに関連付けられたタスクです。

`folder -show tasks` のデフォルト出力フォーマットは以下のとおりです。

```
Task %displayname: %task_synopsis
```

ここで、

DCM が無効な場合、`%displayname` は `%name` です。
DCM が有効な場合、`%displayname` は
*<database_ID><DCM_delimiter><task_number>* です。
`%task_synopsis` はタスクの説明です。

`-format` オプションを使用しない場合は、以下のいずれか 1 つのキーワードを使用してフォルダ プロパティを表示できます。

```
mode
n|na|name
q|qu|query
u|us|usable
w|wr|writable
```

コマンド実行が成功した場合は値 `0` が返され、成功しなかった場合はゼロ以外の値が返されます。

*task_specs*

タスクの ID を指定します。この引数の構文については、タスクの指定 を参照してください。

-u

このコマンド出力の自動番号付与を抑止します（"un-numbered"）。

-un|-union

指定したフォルダのいずれかに存在するすべてのタスクを比較結果に表示するよう指定します。

このオプションは、-compare オプションと一緒にのみ使用できます。

-up|-update

指定したフォルダのタスクに関連付けられたオブジェクトを更新します。

このオプションは、-modify オプションと一緒にのみ使用できます。

-us|-usable *usable_by*

フォルダを Owner、Build_Manager、All または None で使用可能とします。

このオプションは、-create または -modify オプションと一緒にのみ使用できます。また、修正するフォルダが書き込み可能である必要があります。

folder usable のデフォルト出力フォーマットは以下のとおりです。

```
%owner
```

ここで、

%owner はプロジェクトの所有者です。

-v|-verbose

-show info を使用する場合は、-verbose オプションを使用して、すべてのフォルダとすべての関連タスク、それにタスクの影響を受けるすべてのオブジェクトを表示できます。-show tasks を使用する場合は、-verbose オプションを使用して、各フォルダのすべてのタスクおよびタスクの影響を受けるすべてのオブジェクトを表示できます。

`-w|-writable` *`writable_by`*

フォルダを `Owner`、`Build_Manager`、`All` または `None` で書き込み可能とします。このオプションを `None` に設定しようとすると、これによってフォルダがすべてのユーザーに対して読み出し専用となるため、確認メッセージが表示されます。`-y` オプションを使用して、確認メッセージを抑止できます。

このオプションは、`-create` または `-modify` オプションと一緒にのみ使用できます。また、修正するフォルダが書き込み可能である必要があります。

`folder -writable` のデフォルト出力フォーマットは以下のとおりです。

```
%owner
```

ここで、

`%owner` はプロジェクトの所有者です。

`-y`

フォルダの削除または変更時に表示される確認メッセージを抑止します。このオプションは、`-delete` または `-modify` オプションと一緒にのみ使用できます。

確認メッセージは、ユーザー エラーに対する重要な保護手段となります。`-y` オプションの使用には注意が必要です。

関連トピック

- folder コマンドの例

## **folder** コマンドの例

以下の操作の例について説明します。

- 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
- フォルダの名前変更
- フォルダ情報の表示

### フォルダの比較

- フォルダ 154 または 155 にあるタスクを表示する。

```
ccm folder -comp 154 -un 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 154 -int 155
1) Task 15: Correct spelling 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 に存在して、155 に存在しないタスクを表示する。

```
ccm folder -comp 154 -not 155
1) Task 12: System error when time zone changes
2) Task 31: Wrong window receives message
```

- データベース内の 2 つのフォルダを比較する。まず、すべてのフォルダをクエリします。

```
ccm folder -list all
```

以下のような出力が表示されます。

```
1) Folder 27: All completed tasks for release 2.0
2) Folder 32: All tested tasks for release 2.0
3) Folder 13: Bills tasks for release 2.0
```

次に、All completed tasks for release 2.0 フォルダに存在して、All tested tasks for release 2.0 フォルダに存在していないタスクを表示します。

```
$ ccm folder -compare @1 -not_in @2
Task 304: Change splash screen for release 2.0
Task 306: Change copyright for release 2.0
```

### フォルダのコピー

- フォルダ 95 を Tasks Completed for Release 3.4 on September 15, 1997 という名前の新規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 All completed tasks for 2.1 を既存のフォルダ All completed tasks for 2.0 にコピーし、これら 2 つのフォルダの内容をマージする。

  ```
  ccm folder -copy All completed tasks for 2.1 -existing All completed
  tasks for 2.0 -append
  ```

### フォルダの作成

- 所有者が書き込み可能で、すべてのユーザーが使用可能な 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 以外のコマンド出力をすべて抑止する。

  ```
  ccm folder -cr -n "Tested Tasks for Release 3.5" -w Owner -us All -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.
  ```

### フォルダの削除

- フォルダ 109、110 および 158 を削除する。

```
ccm folder -delete 109-110,158
Are you sure that you want to delete folder '109: Tasks Completed for Release
2.1 on May 1, 1996'? (Yes/All/No) [No] y
Deleted folder '109: Tasks Completed for Release 2.1 on May 1, 1996'.
Are you sure that you want to delete folder '110: Tasks Completed for Release
2.2 on July 1, 1996'? (Yes/All/No) [No] y
Deleted folder '110: Tasks Completed for Release 2.2 on July 1, 1996'.
Are you sure that you want to delete folder '158: Tasks Completed for Release
3.4 on September 15, 1997'? (Yes/All/No) [No] y
Deleted folder '158: Tasks Completed for Release 3.4 on September 15, 1997'.
```

### フォルダの使用箇所の検索

- 現在のデータベース内で All Completed Tasks for Release 2.1 という名前のフォルダの使用箇所を検索する。

```
ccm folder -list all_non_writable -format "%displayname %description"
All Non-Writable Folders
1) 42 All Completed Tasks for Release 2.1
2) 89 All Completed Tasks for Release 2.2
```

### フォルダの一覧表示

- 現在のデータベース内にあるビルド マネージャのすべての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
  1) Folder 111: mary's Insulated Development Folder
  2) Folder 145: mary's Completed Tasks for Release 4.2
  3) Folder 146: mary's Assigned Tasks
  ```

- データベース内のすべてのフォルダ テンプレートを一覧表示する。

  ```
  ccm folder -list -template all
  ```

## フォルダの修正

- 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
Folder '111: mary's Insulated Development Folder' has been changed to add
tasks using a query.
```

- 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: mary's Insulated Development Folder' has been
changed to: owner='mary' and release='3.5'
```

- *`folder_number`* について、使用権限と書き込み権限を *`use_permission`* と *`write_permission`* にそれぞれ変更する。

```
ccm folder -modify -usable use_permission -writable write_permission
folder_number
```

*read_permission* と *write_permission* は以下のいずれかの値を持つことができます。

- Owner
- Build_Manager
- All
- None

> 注意！ 使用権限または書き込み権限を None に変更すると、すべてのユーザーがそのフォルダを使用不可または読み出し専用になります。

## フォルダの名前変更

- *folder_number* の名前を変更する。

```
ccm folder -modify -name "new_folder_name_string" folder_number
```

> 注記：フォルダがテンプレートで管理されている場合は、フォルダ名を変更しようとすると確認メッセージが表示されるので、応答する必要があります。コマンドに -y オプションを追加することにより、そのメッセージを抑止できます。

## フォルダ情報の表示

- フォルダ 160 の情報を表示する。

```
ccm folder -sh info 160
Folder '160: My Tasks for Release 3.5'
    Owner:        mary
    Writable By:  Owner
    Usable By:    Owner
    Query Type:   All My Tasks
    Query:        owner='john' and release='3.5'
```

- 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  mary 19
```

```
2) MSGS.C-3:csrc:1    integrate  mary 19
3) MSGS.H-2:incl:1    integrate  mary 19
4) DIALOG.C-8:csrc:1  integrate  mary 57
5) DIALOG.H-13:incl:1 integrate  mary 57
```

## **folder_template** コマンド

### 表記

#### フォルダ テンプレートの作成

```
ccm ft|folder_temp|folder_template -c|-create
          [-desc|-description description]
          [-us|-usable usable_by] [-w|-writable writable_by]
          [-mode {man|manual | uq|use_query]
          [-task_scope|-ts task_scope]
          [-database_id|-dbid|-db dbid]
          [-must_be_local|-nomust_be_local]
          [-release|-rel release]
          [-platform|-plat platform]
          [-subsystem|-sub task_subsystem]
          [-custom "query_expression"]
          "folder_template_name"
```

#### フォルダ テンプレートの修正

```
ccm ft|folder_temp|folder_template -m|-modify
          [-desc|-description description]
          [-us|-usable usable_by] [-w|-writable writable_by]
          [-mode {man|manual | uq|use_query]
          [-task_scope|-ts task_scope]
          [-database_id|-dbid|-db dbid]
          [-must_be_local|-nomust_be_local]
          [-release|-rel release]
          [-platform|-plat platform]
          [-subsystem|-sub task_subsystem]
          [-custom "query_expression"]
          folder_template_specs [folder_template_specs]*
```

#### フォルダ テンプレートの削除

```
ccm ft|folder_temp|folder_template -d|-delete
          folder_template_specs [folder_template_specs]*
```

#### フォルダ テンプレートの一覧表示

```
ccm ft|folder_temp|folder_template -l|-list
```

#### フォルダ テンプレートの管理データベースの設定

```
ccm ft|folder_temp|folder_template -cdb|-controlling_database
          (-local | -handover database_id | -accept database_id)
```

フォルダ テンプレートの詳細プロパティの表示

```
ccm ft|folder_temp|folder_template -sh|-show (i|info|information)
          folder_template_specs [folder_template_specs]*
```

フォルダ テンプレートの特定プロパティの表示

```
ccm ft|folder_temp|folder_template -sh|-show
          (description|mode|description|query|(u|us|usable)|(w|wr|writable))
          folder_template_specs [folder_template_specs]*
```

## 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。プロセス ルールは、プロジェクトをどのように更新（リコンフィギュア）するかの基準の 1 つとして、フォルダ テンプレートを使用できます。

`folder_temp` コマンドにより、以下のタスク ベースの Rational Synergy 操作を実行します。

- フォルダテンプレートを作成する。
- フォルダ テンプレートを修正する。
- フォルダ テンプレートを削除する。
- フォルダ テンプレートを一覧表示する。
- フォルダ テンプレートの管理データベースを設定する。
- フォルダ テンプレートの詳細プロパティを表示する。
- フォルダ テンプレートの特定プロパティを表示する。

## オプションと引数

`-accept`

オブジェクトが指定データベースから管理を受け付けるように設定します。

`-cdb|-controlling_database`

指定したフォルダ テンプレートの管理データベースを指定します。

`-cr|-create`

フォルダ テンプレートを作成し、その名前を指定された *`folder_template_name`* とします。同じ名前を持つフォルダ テンプレートが既に存在している場合、エラー メッセージが表示され、新しいフォルダ テンプレートは作成されません。

コマンド実行が成功した場合は値 `0` が返され、成功しなかった場合はゼロ以外の値が返されます。

`-delete`

指定した 1 つまたは複数のフォルダ テンプレートを削除します。指定したフォルダ テンプレートがプロセス ルールで使用されている場合、またはシステム定義のフォルダ テンプレートである場合、エラー メッセージが表示され、フォルダ テンプレートは削除されません。

削除するテンプレートは、書き込み可能である必要があります。

コマンド実行が成功した場合は値 `0` が返され、成功しなかった場合はゼロ以外の値が返されます。

`-dbid|-database_id` *`dbid`*

作成または修正するフォルダ 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。*`dbid`* を指定しないと、現在のデータベースのデータベース ID が使用されます。

`-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。説明を指定しないと、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。

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、3 つのキーワード `%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。

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 プロジェクトの更新プロパティ 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。

`%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、`joe` 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 `joe'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ます。

`%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、`joe` 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 `joe'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ます。

*`folder_template_name`*

作成または修正するフォルダ テンプレートの名前を指定します。フォルダ テンプレート名は必須で、固有の名前でなければなりません。二重引用符と一重引用符はフォルダ テンプレート名として使用できません。たとえば、フォルダ テンプレート名 "%owner's tasks" は無効です。

*`folder_template_specs`*

作成または修正するフォルダ テンプレートを指定します。

*folder_template_specs* タイプの引数は以下のいずれかです。

- フォルダ テンプレート名
- カンマで区切られたフォルダ テンプレート名のリスト
- オブジェクトの 4 部名称
- 選択セット参照（@n または @）、またはカンマで区切られた複数の選択セット参照
- cvid 形式（@=*n*）またはカンマで区切られた複数形式
- 1 つまたは複数のフォルダ テンプレート名（それぞれの名前を 1 行に指定）を含むファイル名

`-handover`

オブジェクトの管理を現在のデータベースから指定データベースに渡します。

`-l|-list`

現在定義されているすべてのフォルダ テンプレートを一覧表示します。

`-local`

ローカル管理を設定し、他ののデータベースから作成された DCM 複製があれば破棄するよう指定します。

`-mode {man|manual | uq|use_query}`

フォルダ テンプレートの内容を手動（`manual`）、またはクエリを使用して（`use_query`）定義します。

このオプションは、`-create` または `-modify` オプションと一緒にのみ使用できます。

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、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。
- 手動フォルダ 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、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、さらにユーザー定義クエリ（default_task_query）が存在する場合には、ユーザー定義クエリを使用する。
- 手動フォルダ 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、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）が存在せず、さらにビルド マネージャとして作業している場合には、`All Completed Tasks` を使用する。

- 手動フォルダ 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、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）が存在せず、さらにビルド マネージャとして作業していない場合には、`All Tasks Assigned to` *`your_user_name`* を使用する。

`-m|-modify`

フォルダ テンプレート プロパティを変更できます。`-modify` オプションには、複数のサブオプションを使用できます。このオプションを使用して、フォルダ テンプレートのデフォルト スタイルまたはカスタム スタイル クエリの一部を変更できます。

オプションを変更するフォルダ テンプレートは、書き込み可能である必要があります。

コマンド実行が成功した場合は値 `0` が返され、成功しなかった場合はゼロ以外の値が返されます。

`-must_be_local`

ローカルに作成されたプロジェクトの更新プロパティの場合に、フォルダ テンプレートによってローカル フォルダが使用される必要があることを示します。

`-must_be_local` と `-nomust_be_local` オプションは一緒に使用できません。フォルダ テンプレートの作成時にどちらも指定しないと、デフォルトは以下のようになります。
`-nomust_be_local`

`-nomust_be_local`

ローカルに作成されたプロジェクトの更新プロパティの場合に、フォルダ テンプレートによって非ローカル フォルダが使用される必要があることを示します。

`-must_be_local` と `-nomust_be_local` オプションは一緒に使用できません。フォルダ テンプレートの作成時にどちらも指定しないと、デフォルトは以下のようになります。
`-nomust_be_local`

`-plat|-platform` *`platform`*

フォルダ テンプレートに関連する変更を適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を設定しないでください。

このオプションは、`-create` または `-modify` オプションと一緒にのみ使用できます。

`-qu|-query`

フォルダ テンプレートをクエリ ベースにします。

このオプションは、`-create` オプションと一緒にのみ使用できます。

CLI 内のすべてのユーザー作成クエリは、カスタム クエリ（クエリ文節なし）として保存され、指定オプションに関連付けられた適用範囲は保存されません。

`-r|-release` *`release_value`*

フォルダ テンプレートのリリース値を指定します。可能なリリース値には、このデータベース内の定義済みの有効なリリース値がありますが、それに限定されません。

`-rt|-remove_task` *`task_specs`*

指定されたフォルダから 1 つまたは複数のタスクを削除します。*`task_specs`* 引数の構文については、タスクの指定を参照してください。

このオプションは、`-modify` オプションと一緒にのみ使用できます。また、修正するフォルダが書き込み可能である必要があります。

*`scope_spec`*

フォルダの一覧表示に使用するクエリを変更します。

*`scope_spec`* は以下のいずれか 1 つです。

```
all_personal | all_build_mgrs | all_shared | all_non_writable | all
```

`-sh|-show`

指定したフォルダ テンプレートのプロパティを表示します。このオプションを `info` キーワードと一緒に使用すると、選択セットには出力に一覧表示されるオブジェクトまたはタスクが含まれます。

`folder_template -show information` のデフォルト出力フォーマットは以下のとおりです。

```
Folder_Template %displayname: %description
```

ここで、

`%DCM` が無効な場合、*`%displayname`* は `%name` です。
`%DCM` が有効な場合、*`%displayname`* は *`<database_ID><DCM_delimiter><task_number>`* です。
`%description` はフォルダ名です。

これらの行の後に、いくつかのフォルダ プロパティに関する追加情報が表示されます。

`folder_template -show objects` のデフォルト出力フォーマットは以下のとおりです。

```
%objectname %status %owner %task
```

ここで、

`%objectname` はオブジェクトの *`name-version:type:instance`* です。
`%owner` はオブジェクトの状態です。
`%owner` はオブジェクトの所有者です。
`%task` はオブジェクトに関連付けられたタスクです。

コマンド実行が成功した場合は値 `0` が返され、成功しなかった場合はゼロ以外の値が返されます。

`-sub|-subsystem` *`task_subsystem`*

タスクが属するサブシステムを指定します（例、Any、GUI code、CLI code、または documentation）。サブシステムの指定に空白が含まれる場合は、引用符で囲んでください。

このオプションは、`-create` または `-modify` オプションと一緒にのみ使用できます。

`-task_scope|-ts` *`task_scope`*

タスク範囲を指定します。ただし、*`task_scope`* は以下のいずれかのフォルダです。

`all_owners_assigned_or_completed` (default)
`all_owners_assigned`
`all_owners_completed`
`all_owners_tasks`
`all_completed`
`all_tasks`

`-us|-usable` *`usable_by`*

フォルダ テンプレートを `Owner`、`Build_Manager`、`All` または `None` で使用可能とします。

このオプションは、`-create`、`-modify` または `-controlling_database` オプションと一緒にのみ使用できます。また、修正するフォルダ テンプレートが書き込み可能である必要があります。

`folder_template -usable` のデフォルト出力フォーマットは以下のとおりです。

```
%owner
```

ただし、`%owner` はプロジェクトの所有者です。

`-w|-writable` *`writable_by`*

フォルダを `Owner`、`Build_Manager`、`All` または `None` で書き込み可能とします。このオプションを `None` に設定しようとすると、これによってフォルダがすべてのユーザーに対して読み出し専用となるため、確認メッセージが表示されます。

このオプションは、`-create` または `-modify` オプションと一緒にのみ使用できます。また、修正するフォルダが書き込み可能である必要があります。

`folder -writable` のデフォルト出力フォーマットは以下のとおりです。

```
%owner
```

ただし、`%owner` はプロジェクトの所有者です。

## 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
```

- 説明が "%ownerís 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"
  ```

- ローカル管理されているフォルダ テンプレートの管理を、ID が A1 であるデータベースに引き渡す。

  ```
  ccm folder_template -controlling_database -handover A1
  folder_template_spec
  ```

- DCM データベース内のフォルダ テンプレート 99 をデータベース非固有に変更する。

  ```
  ccm folder_template -modify -desc "Completed Tasks for Release %release"
  -database_id Any T99
  ```

- カスタム クエリを使用して、2006 年 4 月 29 日より前に完了したタスクを収集するよう T99 のフォルダ テンプレートを修正する。

  ```
  ccm folder_template -modify -custom "(status=団ompletedí) and
  (completion_date<time(í4/29/06í)) T99
  ```

- 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 クエリを定義します。

  1. 適用範囲を設定します。
  2. リリースを設定します。

  パラレル開発とフォルダ テンプレート管理のためには、この属性を設定してください。

  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。

     ```
     ccm folder_template -create -desc name -task_scope scope -release
     %release -usable usable_by -writable writable_by
     ```

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と使用が可能となります。

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

関連トピック

- folder コマンド

# **fs_check** コマンド

## 表記

```
ccm fs_check [-d|-dir directory_path] [-f|-fix] [object_spec...]
             [-v|-verbose][-t|-type type] [-e|-empty_skip] [-u|-unused_skip]
             [-nd|-noduplicates] [-w|-windows]
             [-n|-null_byte][-z|-zero_counts]
```

## 説明と用途

`ccm fs_check` コマンドにより、Rational Synergy データベースのファイル システムの整合性チェックを行います。デフォルトで、`ccm fs_check` コマンドは以下のことをチェックします。

- キャッシュ エリア内の各ファイルは、既存のオブジェクト バージョンと対応する。
- アーカイブ エリア内の各ファイルは、1 つ以上の静的オブジェクト バージョンと対応する。
- アーカイブ ファイル内の各エントリは、1 つの静的オブジェクト バージョンと対応する。
- プロジェクトまたはディレクトリのソースは空である。

キャッシュおよびアーカイブ エリア内のすべてのファイルのチェックには時間とメモリが必要ですが、`-u|-unused` オプションを使用してチェック量を減らすことができます。

整合性を保証するために、`ccm fs_check` を実行してデータベース全体をチェックしてください。このコマンドを定期的に使用すると、キャッシュ ファイルが使用しているディスク領域を削減できます。しかし、大きなデータベースではチェックに時間がかかる可能性があります。したがって、特定の種類のオブジェクトだけをチェックすることにより、チェックを簡単に済ませることができます。`-t` オプションを使用して特定のオブジェクトだけをチェックするか、あるいは *`object_specs`* を使用して一連のオブジェクト（たとえばクエリ結果を使用して）をチェックできます。`-t` オプションとオブジェクトの一覧を両方指定することはできません。結果を調べられるように、ファイルに出力してください。

予期しないファイルや余分なファイル、またはアーカイブ エントリが見つかった場合は、個別に報告された上、最終的にまとめて表示されます。ただし、このようなケースはエラーとはみなされず、`ccm fs_check` は終了コード 0 で終了します。`ccm fs_check` の `-fix` オプションは、これら余分なエントリを取り除きません。こういったエントリを削除してしまうことは、特別の理由でファイルを手動作成した場合や、ファイル システムとメタデータをバックアップからリストアした場合などに、データが失われることになるからです。不要なキャッシュやアーカイブ エントリの削除については技術サポートにご連絡ください。

この操作はすべてのユーザが行うことができますが、`-fix` オプションを使用するには、*ccm_admin* ロールが必要です。

## オプションと引数

`-d|-dir` *`directory_path`*

整合性のないアーカイブ エントリを書き出すディレクトリを指定します。デフォルトで、これらのファイルは *`database_path`*`/st_root/tmp/check` に書き出されます。

`-e|-empty_skip`

プロジェクトとディレクトリの空でないソースについての警告を抑止します。

`-f|-fix`

以下のような単純なエラーを修復します。

- UNIX で作成したパック ファイルからデータベースをアンパックすると、キャッシュ ファイルは UNIX 形式になっている可能性がある。キャッシュ ファイルとアーカイブの唯一の違いが復帰改行の形式である場合、`-f|-fix` フラグはキャッシュ ファイルを削除する。
- キャッシュ ファイルの長さが 0 であるのに、アーカイブされている内容がそうでない場合、`-f|-fix` フラグはキャッシュ ファイルを削除する。
- キャッシュ ファイルの変更時間が間違っているが、内容がアーカイブと同じ場合、`-f|-fix` オプションを使用して変更時間を `source_modify_time` 属性と同じにする。

`-nd|-noduplicates`

重複アーカイブ エントリのチェックをスキップします。このオプションを使用すると、メモリの不足で失敗する可能性のある非常に大きなデータベースのチェックに使用するメモリを減らすことができます。これによって、アーカイブ チェックの有効性が低くなるので、本当に必要なときのみ使用してください。

*`-n|-null_byte`*

ヌル（0x00）バイトのソース属性をチェックします。タイプ ascii およびサブタイプ ascii のオブジェクトがヌル バイトを含むとき、警告メッセージが表示されます。

*`object_spec`*

チェックするオブジェクトのリストを与えます。この引数にはクエリ結果を使用できます。

この引数は、-t オプションと一緒には使用できません。

-t|-type *type*

チェックするオブジェクトのタイプを指定します。

このオプションは、*object_spec* と一緒には使用できません。

-u|-unused_skip

すべてのキャッシュおよびアーカイブのファイルとエントリの使用チェックを抑止します。

-v|-verbose

各エラーについてより詳細な情報を出力します。以下のことを報告します。

- 問題とタスクを除き source 属性がないオブジェクト。これらのオブジェクトは、キャッシュあるいはアーカイブ エントリもないので、スキップされます。
- 古い 4.1 以前のアーカイブ（SCCS、圧縮、および、RCS。 ccm_rcs は除く）でアーカイブされているファイル。これは、アーカイブの変換が必要なことを意味します。
- キャッシュ ファイルを持たないオブジェクト。このようなオブジェクトは、おそらく以前実行した ccm clean_cache の影響を受けています。
- source_modify_time 属性を持たないオブジェクト。 これは小さなエラーです。このようなオブジェクトは、現在のデータベース標準に正しくアップグレードされていません。タイプが time の source_modify_time 属性を作成し、正しい時刻（ソース ファイルをチェックインする前に最後に編集した時刻）に設定できます。これをキャッシュ ファイルの修正時刻にします。
- source_modify_time 以前の時刻を持つオブジェクト キャッシュファイル。このエラーは重大ではなく、キャッシュ ファイル時刻を設定する呼び出しの失敗によって起きた可能性があります。これはアーカイブ エントリが正しいことを確認した後で、キャッシュ ファイルを削除して修正します。

-w|-windows

ファイル間の違いが CR 文字のみである場合に、警告を抑止します。

`-z|-zero_counts`

最後のサマリでは、レポートの条件に合致するものがない（ゼロ）場合には通常は表示されません。`-z` オプションを使用すると、値がゼロのものを含めてすべてのカウントが出力されます。これは、別プログラムで `fs_check` の結果を分析するときに便利です。また、新しいチェックがいつコマンドに追加されたかも検出できます。

> 注記：このオプションは、空ファイルのチェックをスキップするように指定した `-z` オプションを再使用します。このオプションを前にスクリプト内で使用している場合は、そのスクリプトを変更する必要があります。

## 例

`project1` データベースのファイル システムの整合性をチェックし、詳細な出力情報を出します。

Windows：
```
ccm fs_check -d c:¥data¥ccmdb¥project1 -v
```
UNIX：
```
ccm fs_check -d /vol/hydra1/ccmdb/project1 -v
```

関連トピック

- lmgr_status コマンド

# **groups** コマンド

## 表記

```
ccm groups    [-a|-assign] object_spec
              [-a|-assign] [-v|-value groupname] object_spec
              [-l|-list]
              [-c|-create] group_name
              [-e|-edit] group_name
```

## 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にすべてのオブジェクトの表示、チェックアウトおよび修正を許可するのは、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、オブジェクトの可視性を指定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 セキュリティにより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。

*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ますが、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『IBM Rational Synergy 管理者ガイド』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ため、使用しないでください。

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、『IBM Rational Synergy Distributed』を参照してください。こ

## オプションと引数

`-a|-assign` *`object_spec`*`:readsource`

指定したオブジェクトのセキュリティを割り当てます。デフォルトのテキスト エディタを使用してグループを追加します。`group_name` に空白が含まれる場合は、引用符で囲む必要があります。`:readsource` を使用した場合、そのオブジェクトには読み出しセキュリティが設定されます（ソース ファイルの表示ができません。）

`-c|-create` *`group_name`*

グループ名を作成します。デフォルトのテキスト エディタを使用してグループのメンバーシップを定義します。

`-e|-edit` *`group_name`*

既存の指定グループのユーザー メンバーシップを編集します。

`groupname`

1 つまたは複数の項目を、スペース、タブ、および／またはコンマで区切って、グループ名を指定します。`groupname:readsource` と指定した場合、指定された名前のグループのユーザーは、ソース コンテンツへの読み出しアクセスが許可されます。

`-l|-list`

すべての定義済みグループを一覧表示します。

`-v|-value` *`groupname`*

指定オブジェクトを使用可能なグループを表示します。

## 例

- すべての定義済みグループを一覧表示する。

  ```
  ccm groups -l
  ```

- *`docs`* という名前のグループを作成する。

  ```
  ccm groups -c docs
  ```

- `makefile.pc-1:makefile:tut63#4` という名前のオブジェクトを使用可能なグループを表示する。

  ```
  ccm groups -value makefile.pc-1:makefile:tut63#4
  qa_team
  design_team
  ```

# **help** コマンド

## 表記

`ccm help [`*`argument`*`]`

## 説明と用途

`help` コマンドにより、オンライン ヘルプを使用します。ヘルプについての一般情報は以下の方法で取得できます。

- 引数を指定せずに `ccm help` コマンドを使用して、Rational Synergy コマンド リストを表示する。
- `ccm alias` を入力して、Rational Synergy コマンドの別名を表示する。

オンライン ヘルプは以下の方法で起動します。

- Windows ユーザー

  コマンド ラインから `ccm help` 形式のコマンドでヘルプを要求すると、デフォルト ブラウザが起動し、要求されたコマンドのヘルプが表示されます。

- UNIX ユーザー（GUI プロセスを実行、つまりコマンドの入力で使用したターミナル ウィンドウに `DISPLAY` 環境変数が正しく設定されている場合）。

  コマンド ラインから `ccm help` 形式のコマンドでヘルプを要求すると、デフォルト ブラウザが起動し、要求されたコマンドのヘルプが表示されます。

- UNIX ユーザー（GUI プロセスを実行していない、または GUI プロセスを実行しているがコマンドを入力したターミナル ウィンドウに `DISPLAY` 環境変数が設定されていない場合）

  コマンド ラインから `ccm help` 形式のコマンド形式でヘルプを要求すると、HTML ヘルプ ファイルの ASCII 変換テキストが表示されます。

## オプションと引数

*`argument`*

ヘルプを表示したいコマンドの名前を指定します。

## 例

- `type` コマンドのヘルプを要求する。

  ```
  ccm help type
  ```

# **history** コマンド

## 表記

```
ccm hist|history [-f|-format "format_string"]
                 file_spec [file_spec...]
ccm hist|history [-f|-format "format_string"]
                 -p|-project project_spec [project_spec...]
```

## 説明と用途

history コマンドにより、オブジェクトのバージョン履歴を表示します。-g オプションを指定するとバージョン履歴がグラフィック表示され、このオプションを指定しないとログレポート形式で表示されます。

## オプションと引数

**-f|-format "*format_string*"**

出力のフォーマットを指定します。必須文字列には、以下のようにキーワードと文字テキストを使用します。

```
%displayname %owner
```

キーワードには、組み込まれたもの（%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***

履歴を表示するファイルまたはディレクトリの名前を指定します。

**-g**

適切なダイアログを呼び出します。履歴を表示したいオブジェクトの ***file_spec*** または ***project_spec*** を指定します。指定しなかった場合は、エラー メッセージが表示されます。

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

プロジェクトの履歴を表示します。

## 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:   bob
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:   john
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:   bob
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

- attribute コマンド
- properties コマンド
- relate コマンド
- unrelate コマンド

# **import** コマンド

## 表記

```
ccm import -f|-from import_dir_path [-q|-quiet] [-a|-all]
           [-image] [-delimiter delimiter_value]
ccm import  {[-f|-from import_dir_path]
           [-n|-name name]
           [-t|-type type]
           [-v|-version version]
           [-i|-instance instance]}
           [-image] [-delimiter delimiter_value]
           [-q|-quiet]
```

## 前提条件

ユーザー *ccm_root* はインポート ディレクトリからの読み出し可能権限を持つ必要があります。これは、インポートを実行する Rational Synergy エンジン プロセスがユーザー *ccm_root* として実行されるからです。

## 説明と用途

`import` コマンドは、Rational Synergy インポート／エクスポート フォーマットで、ファイル システムから Rational Synergy データベースにオブジェクト バージョンをインポートします。Windows では、インポート ディレクトリはエンジン ホストから見える必要があります。

また、このコマンドはリモート クライアントから実行できます。その場合、`-from` オプションを使用し、指定するパスはエンジン ホストから見える必要があります。

引数なしでこのコマンドを実行するか、または `-all` オプションを付けた場合、ディレクトリ内のエクスポートされたすべてのオブジェクトがインポートされます。

> 注記：このコマンドは、データベース内の既存のオブジェクトとタスクを警告なしに上書きします。

このコマンドを使用するには、*ccm_admin* ロールを持っている必要があります。

## オプションと引数

`-a|-all`

ディレクトリ内で見つかったすべてのオブジェクトをインポートします。

`-delimiter` *`delimiter_value`*

オブジェクトの 4 部名称の各部を区切る区切り文字を指定します。現在のデータベースと異なる区切り文字を使用するデータベースからインポートする場合に、このオプションを使用します。たとえば、4.2.1 UNIX から 4.4 データベースにインポートするときは、*`delimiter_value`* にコロンを使用します。

デフォルトは "@" です。

`-f|-from` *`import_dir_path`*

インポート元のディレクトリ パスを指定します。指定するパスはエンジンから見える必要があります。

`-image`

データベース内の各オブジェクトのプロパティをインポート ディレクトリ内のプロパティで置き換えます。新規プロパティは追加され、変更されたプロパティは置き換えられ、削除されたプロパティはデータベースから削除されます。

`-i|-instance` *`instance`*

ディレクトリ内のインスタンス *`instance`* を持つすべてのオブジェクトをインポートします。

`-n|-name` *`name`*

ディレクトリ内の名前 *`name`* を持つすべてのオブジェクトをインポートします。

`-q|-quiet`

関係の欠落したオブジェクトの警告など、コマンドから出力される一部のメッセージを抑止します。

`-t|-type` *`type`*]

ディレクトリ内のタイプが *`type`* であるすべてのオブジェクトをインポートします。

`-skip_model`

インポートされたオブジェクトのモデル処理を行いません。オブジェクトのデータフォーマットからこのデータベースで有効なフォーマットへの処理後変換も含行いません。

このオプションの使用には注意が必要です。

`-v|-version` *`version`*`]`

ディレクトリ内のバージョンが *`version`* であるすべてのオブジェクトをインポートします。

## 例

Windows:

- `/users/patty/export_dir` ディレクトリから `foo.c` をインポートする。

  ```
  ccm import -from /users/patty/export_dir -n foo.c
  ```

- `fileserver1` マシンの `¥¥fileserver1¥export_dir¥demo` ディレクトリから `foo.c` をインポートする。マシンはエンジン プロセスから見える必要があります。

  ```
  ccm import -from ¥¥fileserver1¥export_dir¥demo -n foo.c
  ```

UNIX :

- `/users/patty/export_dir` ディレクトリから `foo.c` をインポートする。

  ```
  ccm import -from /users/patty/export_dir -n foo.c
  ```

関連トピック

- export コマンド

# **license** コマンド

## 表記

```
ccm license [-i|-info] [-t|-timeout minutes]
```

## 説明と用途

`license` コマンドにより、ライセンス情報を表示し、またタイムアウト値を設定します。

## オプションと引数

`-i|-info`

現在のライセンスとライセンス タイムアウトを表示します。これはデフォルト設定です。

`-t|-timeout` *`minutes`*

ライセンス タイムアウトを *`minutes`* に設定します。

このオプションは、ユーザーごとのライセンスには適用されません。

## 例

- ライセンス タイムアウトを 60 分に設定する。

  ```
  ccm license -t 60
  ```

- 現在のユーザーがライセンスを持っているか、またはライセンス タイムアウト期間を表示する。

  ```
  ccm license -i
  ```

# **lmgr_status** コマンド

## 表記

```
ccm lmgr_status
```

## 説明と用途

`ccm lmgr_status` コマンドにより、Synergy のユーザーごとのライセンスのみを表示します。

どのユーザーでもこのコマンドを実行できます。

> 注記：このコマンドは、ライセンスがない場合は "--" を表示します。

## オプションと引数

なし

## 例

```
ccm lmgr_status
```

関連トピック

- message コマンド
- monitor コマンド
- ps コマンド

# **ln** コマンド

## 表記

```
ccm ln [-s] [-c "comment_string"] [-ce|-commentedit] -cf|-commentfile
       [-t|-task <task_number>]file_path] path_name file_spec
```

## 説明と用途

ln 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。

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。

> 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にいて、現在のディレクトリが書き込み禁止の場合は、そのディレクトリはチェックアウトされ、デフォルト（または指定された）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）。

## オプションと引数

**-c "*comment_string*"**

"*comment_string*" に入力されたコメントを指定します。

**-ce|-commentedit**

コメントを入力するためにエディタを起動します。

**-cf|-commentfile *file_path***

オブジェクトのコメントをどこで（*file_path*）読み込むかを指定します。

***file_spec***

シンボリック リンクの作成元となるオブジェクトの名前を指定します。

*path_name*

シンボリック リンクが示す先のパスを指定します。

-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_number*>

新規作成されたシンボリック リンクを指定タスク番号と関連付けます。

## 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 を作成します。

```
ccm ln -s ¥

/jane/ccm_jane_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/gdidemo-unix/gdidemo/init.c /users/gke
Member init.c added to project gdidemo-unix
Associated object version with task 44: init.c-1:symlink:1
Associated object version with task 44: subdirectory-2:dir:1
```

関連トピック

- work_area コマンド

# ls コマンド

## 表記

```
ccm ls [-l] [-m] [-R] [-f|-format "format_string"] [file_spec...]
```

## 説明と用途

ln 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。

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。

## オプションと引数

-f|-format "*format_string*"

出力のフォーマットを指定します。このフォーマットは管理ファイルにのみ適用されます。必須文字列には、以下のようにキーワードと文字テキストを使用します。

```
%displayname %owner
```

キーワードには、組み込まれたもの（%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。

フォーマットの文字列内に %path を指定すると、すべてのオブジェクトは絶対ワーク エリア パス付きで表示されます。ワークエリアが見えない場合は、パスが計算されます。

*file_spec*

表示するファイルを指定します。

-l

情報を、状態、所有者、最終修正時刻、タイプ、インスタンス、名前、バージョンを含む長いフォーマットで一覧表示します。

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。

このオプションを使用すると、適切な場合には以下の記号が出力の前に表示されます。

- LC（ローカル コピー）

  プロジェクト内には存在するが、ワーク エリアにシンボリック リンクではなくローカル コピーを持つファイルを意味します。

  表示されたファイルにこのマークが付いていて、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。リコンサイル機能の詳細については、reconcile コマンドを参照してください。

- NS（同期外れ）

  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。リコンサイル機能の詳細については、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。

ワーク エリアの詳細については、ワークエリア を参照してください。

`-m`

管理ファイルと非管理ファイルを表示します。非管理ファイルとは、ワークエリア内に存在するが、プロジェクトに含まれていないファイルのことです。非管理マークの付いたファイルを表示するには、-m オプションに -l オプションを付けて使用する必要があります。

`-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。

## 例

現在のディレクトリを長いフォーマットで一覧表示します。

```
ccm ls -l
working linda Nov 18 11:56 csrc 1 alias.c-4.5
working linda Nov 18 11:56 csrc 1 diff.c-4.5
working linda Nov 18 11:56 csrc 1 move.c-4.5
working linda Nov 18 11:57 csrc  1 start.c-4.5
```

# **merge** コマンンド

## 表記

```
ccm merge [[-create_task] | [-t|-task task_number]]
          [-c|-comment "string"]
          [-ce|-commentedit] [-cf|-commentfile file_path]
          file_spec1 file_spec2
```

## 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、次に各バージョンの最も近い共通祖先と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 ツールも起動できませんでした。

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。

## オプションと引数

`-c|-comment "`*`string`*`"`

コメント文字列を指定します。

`-ce|-commentedit`

デフォルト エディタを起動します。

`-cf|-commentfile` *`file_path`*

指定したファイルからのコメントを使用します。コメント文字列とコメント ファイルの両方を指定した場合はコメントがマージされ、ファイルのコメントの後ろにコメント文字列が付加されます。

`-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`*

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。

*`file_spec2`*

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。

Merge ダイアログを起動するときは、マージされる 2 つのファイルとバージョンを指定する必要があります。指定しないとエラー メッセージが表示されます。マージ

されるオブジェクトがディレクトリの場合、Directory Merge ダイアログが表示されます。

`-task` *`task_number`*

マージされた新規作成オブジェクトを指定タスクに関連付けます。

カレント（デフォルト）タスクが設定されており、このフィールドに別のタスクを指定しない場合は、マージされたオブジェクトは自動的にカレント タスクに関連付けられます。

## 例

- 2 つのファイル `bufcolor.c-4` と `bufcolor.c-2.1.2` をマージする。

```
ccm merge bufcolor.c-4 bufcolor.c-2.1.2
The current task is set to:
ccmint24#22: Merge 'bufcolor.c-4' (associated with task '21') ...
Task 'ccmint24#22: Merge 'bufcolor.c-4' (associated with task '21') ...'
was created with a release value of 1.1.
Merging attribute source between objects:
        bufcolor.c-4 and bufcolor.c-2.1.2
        with ancestor bufcolor.c-2.1.1.
        Merge Source in progress...
Parallel versions exist for bufcolor.c-4:csrc:ccmint24#1
Adding 'release' attribute with value '' to object bufcolor.c
5:csrc:ccmint24#1
Warning: Merge Source warning. (overlaps during merge).
        Merge conflicts have been noted in bufcolor.c-5.
        Search for '<<<<<<<' to find conflicts.
Merge Source completed successfully.
        Merged object is bufcolor.c-5
Associated object version with task 22: bufcolor.c-5:csrc:ccmint24#1
Using file 'bufcolor.c-5' in project 'test-laura'...
Replaced 'bufcolor.c-2.1.2' with 'bufcolor.c-5' in project 'test-laura'...
```

- 2 つのファイルをマージして新規タスクを作成する。

  2 つのオブジェクト バージョンをマージする場合、Rational Synergy は各ファイルのデータを用いて 3 番目のオブジェクト バージョンを作成します。3 番目のオブジェクトのバージョンは、コマンド ラインから指定された最初のバージョンから取得されます。

  `ccm merge` *`file_spec1 file_spec2`* `-create_task -comment "`*`Your comment.`*`"`

  どのバージョンとマージするかわからない場合は、以下に示す `query` コマンドを実行して正しいオブジェクト バージョンを見つけてください。

```
ccm query -o username -v version -s state -t type
```

- 2 つのディレクトリをマージする。

  ```
  ccm merge directory1-version directory2-version
  ```

関連トピック

- diff コマンド

# **message** コマンド

## 表記

```
ccm message   -u|-user username "message_text"
              -d database_path "message_text"
              -rfc_address address "message_text"
              -attr value "message_text"
```

## 説明と用途

`ccm message` コマンドにより、Rational Synergy セッションを実行している複数ユーザーに対して、メッセージを直接通知するかブロードキャストします。

メッセージの先頭には、送信元の名前が付きます。メッセージを特定のセッションへ送るには、`/rfc_address` スイッチを使用します。

## オプションと引数

*`-attr value`*

メッセージ宛先の追加条件を指定します。*attr* オプションに使用できる値は、`process`、`callback`、`display`、`pid`、`user`、`host`、`database`、`engine_address`、`pwa_path` のいずれかです。

一度に 1 つのスイッチのみ使用できます。

`-d` *`database_path`*

メッセージの送り先のデータベースを指定します。

ACCENT 正規表現を使用して複数のデータベースを指定できます。ACCENT 正規表現は、疑問符（?）で始まる必要があります。

`-rfc_address` *`address`*

メッセージの送り先の RFC（Remote Function Call）を指定します。このフォーマットは、`host:port[:ip]` または `ip:port[:ip]` などのホスト名または IP アドレスで開始します。`[:ip]` は、ゼロ以上の IP アドレスを示します。

> 注記：指定した `rfc_address` がエンジン用の場合、`ccm message` コマンドは失敗します。

`-user` *`user`*

メッセージの送信先のユーザを指定します。

## 例

- `host` 属性を使用して、`ccm message` コマンドを実行する。

  ```
  ccm message -host comp1 "New compile server is up"
  ```

- `database` 属性を使用して、すべてのデータベースに対するメッセージを指定する。

  ```
  ccm message -d "?" "Server going down in 2 minutes..."
  ```

## 警告

指定した `-rfc_address` がエンジン用の場合、command は失敗します。

関連トピック

- fs_check コマンド
- monitor コマンド
- ps コマンド
- diff コマンド

# **migrate** コマンド

## 表記

**Migrate**

```
ccm migrate -d|-dir|-directory dirname -p|-project project_spec
            [-a|-arch_state arch_state]
            [-cb|-copy_based|-not_copy_based|-ncb] - UNIX only
            [-dt|-default_type type]
            [-i|-import_identical]
            [-mc|-meta_create_time]
            [-mo|-meta_owner]
            [-mr|-meta_release]
            [-ms|-meta_status]
            [-n|-nomerge]
            [-nt|-notask]
            [-path|-set|-setpath]absolute_path
            [-r|-release release_value]
            [-rel|-relative|-nrel|-not_relative]
            [-rl|-rules filename]
            [-st|-state state]
            [-t|-task task_number]
            [-tl|-translate|-ntl|-no_translate]
            [-wa|-maintain_wa|-nwa|-no_wa]
            [-wat|-wa_time |-nwat|-no_wa_time]
```

## 説明と用途

`migrate` コマンドにより、ファイル システムから Rational Synergy にファイル構造をロードします。

どのユーザーでもこのコマンドを実行できます。

マイグレーション プロセスの詳細については、マイグレーション ルールを参照してください。

## オプションと引数

`-a|-arch_state` *`arch_state`*

PVCS®（Windows）または RCS および SCCS（UNIX）アーカイブ オブジェクト バージョンの Rational Synergy へのロード後の状態を指定します。

この値をすべてのマイグレーションの初期設定ファイルに設定できます。詳細については、migrate_default_arch_state を参照してください。

`-cb|-copy_based`

ワークエリアをコピー ベースにします（UNIX のみ）。

これはワークエリアのオプションです。詳細については、work_area コマンドを参照してください。

`-d|-dir|-directory` *`dirname`*

マイグレートするルート ディレクトリを指定します。

`-dt|-default_type` *`type`*

マイグレーション ルールを持たないファイルのオブジェクト タイプを指定します。たとえば、`default_type` を `binary` に設定し、`*.pdf` ファイルのマイグレーション ルールがない場合、`*.pdf` ファイルが `binary` ファイルとしてマイグレートします。

既存オブジェクトの新規バージョンの増分マイグレーションでは、同じタイプが保持されます。

この値をすべてのマイグレーションの初期設定ファイルに設定できます。詳細については、migrate_default_type を参照してください。

`-g`

適切なダイアログを呼び出します。

`-i|-import_identical`

データベース内の該当するオブジェクトと同じファイル システム内のファイルについては、新規バージョンを作成します （ファイルとその該当オブジェクトの最新バージョンは、ソース コードが同じであれば「同じ」です)。通常は、同じバージョンはスキップしますが（デフォルト)、他のベンダーのソフトウェアの新規リリースをマイグレートする場合は、同じバージョンのロードを選択する場合があります。

デフォルトでは、同じファイルはスキップされます。

`-mc|-meta_create_time`

マイグレートするファイルの Rational Synergy での作成時刻属性が、サードパーティ アーカイブ作成時刻に設定されます。アーカイブは、PVCS（Windows)、SCCS または RCS9（UNIX）です。

注記：この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。

`-mo|-meta_owner`

マイグレートするファイルの Rational Synergy での所有者属性が、サードパーティアーカイブの所有者に設定されます。アーカイブは、PVCS（Windows）、SCCS または RCS9（UNIX）です。

注記：この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。

`-mr|-meta_release`

マイグレートするファイルの Rational Synergy でのリリース属性が、サードパーティ アーカイブのリリース値に設定されます。アーカイブは、PVCS（Windows）、SCCS または RCS9（UNIX）です。

Rational Synergy は、マイグレートしたファイルのリリース値を設定する前は、サードパーティ アーカイブのリリース値が有効なリリースであることを保証しません。値が存在しない場合は、リリースを手動で追加する必要があります。

**Windows** の注意事項：PVCS ファイルに複数のバージョン（リリース値）ラベルがある場合、複数ラベルは空白で区切った 1 ラインに連結されるので、ファイルのマイグレーション後に編集が必要です。

`-ms|-meta_status`

マイグレートするファイルの Rational Synergy での状態属性が、サードパーティアーカイブの状態値に設定されます。アーカイブは、PVCS（Windows）、SCCS または RCS9（UNIX）です。

注記：この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。

注意！ 状態は Synergy の状態として定義する必要があります。

`-n|-nomerge`

ディレクトリ オブジェクトの旧バージョンが新規バージョンに置き換えられます。その結果、新規ディレクトリ オブジェクトのメンバー リストには、新旧メンバーのマージされたリストではなく、マイグレーション直後のメンバーのみが含まれます。

`-ncb|-not_copy_based - UNIX only`

ワークエリアをリンク ベースにします。

これはワークエリアのオプションです。詳細については、work_area コマンドを参照してください。

`-nrel|-not_relative`

Windows：サブプロジェクトのワークエリアを、親プロジェクトのワークエリアに相対的とはせずに、絶対的にします。これは、最初のプロジェクト作成時のデフォルトです。

UNIX：リンクをサブプロジェクト用に使用し、サブプロジェクトのワークエリアを親プロジェクトのワークエリアに相対的とはせずに、絶対的にします。これは、最初のプロジェクト作成時のデフォルトです。

これはワークエリアのオプションです。詳細については、work_area コマンドを参照してください。

`-nt|-notask`

マイグレーション操作でタスク要件を無視します。

デフォルトでは、マイグレーション操作で新規バージョンの作成とチェックアウト、およびチェックアウトされたバージョンの祖先のチェックイン時にタスク要件に従います。タスク要件は、各タイプの Require Task At オプションに定義されます。

`-ntl|-no_translate`

UNIX サーバーと Windows クライアントを使用している場合は、`ascii`-type オブジェクト ソースの Windows フォーマットを保持します。たとえば、`csrc` オブジェクト内の Windows 復帰改行は UNIX 復帰改行に変換されずに保持されます。

これはワークエリアのオプションです。詳細については、work_area コマンドを参照してください。

`-nwa|-no_wa`

ワークエリアを管理しません（つまり、マイグレートしたプロジェクトのワークエリアをデータベースから切り離します）。

これはワークエリアのオプションです。詳細については、work_area コマンドを参照してください。

`-nwat|-no_wa_time`

このオプションを使用するのは、このプロジェクトでサードパーティ ビルドの実行を停止する場合、前に GUI で Use New Timestamps を設定した場合、またはコマンド ラインで `-wa_time` オプションを使用した場合に限定してください。ワークエリアの詳細については、work_area コマンド を参照してください。

オブジェクトのタイムスタンプが、ファイル システムの時刻ではなく、関連する Rational Synergy 修正（データベースの作成または使用）時刻に設定されていることを確認します。

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

オブジェクトをマイグレートする先のプロジェクトとバージョンを指定します。新規オブジェクトを作成する場合は、一意の名前を使用してください。増分マイグレーションを実行する場合は、必ず既存のプロジェクト名と階層を使用してください。

> 注記：使用する名前が既存の名前と一致する場合、プロジェクトはマイグレートしたオブジェクトで上書きされます。オブジェクト変更前の警告はありません。

`-path|-set|-setpath` *`absolute_path`*

指定したプロジェクトのワークエリア パスを新しい場所に変更します。

これはワークエリアのオプションです。詳細については、work_area コマンドを参照してください。

`-r|-release` *`release_value`*

マイグレートしたファイルとディレクトリのリリース値を指定します。データベースで使用されているコンポーネントとコンポーネント リリース値、それにデータベースで使用されているほかのリリース値などを設定でいます。

`-rl|-rules` *`filename`*

使用するオプションのルール ファイル名を指定します。無限数のルール ファイルを設定できますが、同時に使用できるファイルは 1 つだけです。

ルールの詳細については、マイグレーション ルールを参照してください。

`-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。このオプションは、プロジェクトが 1 つの場所でのみ使用される場合に使用できます。プロジェクトのワークエリア パスを相対的に設定した場合、プロジェクトを複数のプロジェクトで使用できなくなります。

サブプロジェクトとしてマイグレートしたディレクトリは、親プロジェクトを持つため、このオプションの影響を受けます。親プロジェクトは影響を受けません。

これはワークエリアのオプションです。詳細については、work_area コマンドを参照してください。

`-st|-state` *`state`*

マイグレートしたファイルに与えられる状態を指定します。

アーカイブ状態を書き込み可能として設定した場合、アーカイブ状態のオブジェクトがマイグレートされると、その履歴は保持されません。

書き込み禁止の初期状態で SCCS または RCS ファイルをマイグレートすると、すべてのファイルのバージョンがデータベースにマイグレートします。結果として得られるファイル オブジェクトの履歴は、保持されたバージョン履歴を示します。つまり、後継のバージョンが前のバージョンからチェックアウトされ、その結果、アーカイブの履歴が保存されます。アーカイブ タイプのインポート状態として書き込み可能状態を選択すると、書き込み可能バージョンからのチェックアウトはできないので、前のバージョンからのチェックアウトは行わずに指定ファイルのオブジェクト バージョンが作成されます。その結果、アーカイブのインポートが書き込み禁止である場合を除き、アーカイブの履歴を保持することはできなくなります。

この値をすべてのマイグレーションの初期設定ファイルに設定できます。詳細については、migrate_default_state を参照してください。

`-t|-task` *`task_number`*

新規にチェックアウトまたは作成したバージョンに関連付けるタスクを指定します。

`-tl|-translate`

UNIX サーバーと Windows クライアントを使用している場合は、`ascii`-type オブジェクト ソースを Windows フォーマットから UNIX フォーマットに変換します。たとえば、`csrc` オブジェクト内の Windows 復帰改行は UNIX 復帰改行に変換されます。

これはワークエリアのオプションです。詳細については、work_area コマンドを参照してください。

`-wa|-maintain_wa`

ワークエリアを管理します（ワークエリアの同期を取り、同期の状態を維持します）。

これはワークエリアのオプションです。詳細については、work_area コマンドを参照してください。

`-wat|-wa_time`

オブジェクトのタイムスタンプが、Rational Synergy 修正（データベース作成または使用）時刻ではなく、ファイル システム時刻に設定されていることを確認します。

このオプションは、このプロジェクトでサードパーティ ビルドを実行する場合にのみ使用してください。ワークエリアの詳細については、work_area コマンド を参照してください。

## 例

Windows：
`C:¥rules¥migrate.rul` にあるマイグレーション ルールを使用して、プロジェクト `ico-1` をディレクトリ `C:¥examples¥ico¥man` からマイグレートします。

```
ccm migrate -p ico-1 -d C:¥examples¥ico¥man -rl C:¥rules¥migrate.rul
```

UNIX：
~/migrate.rul にあるマイグレーション ルールを使用して、プロジェクト `ico-1` をディレクトリ `/usr/local/ccm/examples/ico/man` からマイグレートします。

```
ccm migrate -p ico-1 -d /usr/local/ccm/examples/ico/man -rl ~/migrate.rul
```

- *directory_path* ディレクトリ階層を *project_name-version* プロジェクトにマイグレートし、オブジェクトとアーカイブ状態をそれぞれ *state* と *archive_state* に設定する。

  ccm migrate -p *project_name-version* -d *directory_path* -st *state* -a *arch_state*

- *directory_path* ディレクトリ階層を *project_name-version* プロジェクトにマイグレートし、デフォルトのオブジェクト タイプを html に設定する。

  ccm migrate -p *project_name-version* -d *directory_path* -dt html

- *directory_path* ディレクトリ階層を *project_name-version* プロジェクトにマイグレートし、同じバージョンをロードする。

  ccm migrate -p *project_name-version* -d *directory_path* -i

# **monitor** コマンド

## 表記

```
ccm monitor -u|-user username
            -d|-database database
            -rfc_address address
            -attr value
```

## 説明と用途

`ccm monitor` コマンドは、以下の情報を含む Rational Synergy のユーザーとプロセス情報を表示します。

- ユーザー
- プロセス タイプ（エンジン、ユーザー インターフェイス、ルーター、ライセンス マネージャ、またはオブジェクト レジストラ）
- ホスト
- ポート
- プロセス ID
- データベース パス

`ccm monitor` コマンドは、プロセスが一定の時間ルーターに応答していない場合、プロセスの状態フィールドに感嘆符（`!`）を付加します。このように応答がないと、プロセスを実行しているマシンがダウンしている、あるいはプロセスが停止しているなどの問題が発生していると判断します。

マシンが使用中のために応答が遅れている場合は、マシンを使用している処理が済むと感嘆符は消えます。

監視表示は、変更内容に応じて更新されます。

UNIX での監視を中止するには、`Ctrl + C` キーを押します。

## オプションと引数

*`-attr value`*

監視するフィールドの名前を指定します。*`attr`* オプションに使用できる値は、`process`、`display`、`pid`、`user`、`host`、`database`、`engine_address`、`pwa_path` のいずれかです。

一度に 1 つのスイッチのみ使用できます。

`-database` *`database`*

*`database`* のすべてのユーザーを監視することを指定します。

正規表現を使用して複数のデータベースを監視できます。正規表現は、疑問符（?）で始まる必要があります。

`-rfc_address` *`address`*

監視する Rational Synergy インターフェイス（GUI）プロセスの RFC（Remote Function Call）アドレスを指定します。このフォーマットは、`host:port[:ip]` または `ip:port[:ip]` などのホスト名または IP アドレスで開始します。`[:ip]` は、ゼロ以上の IP アドレスを示します。

> 注記：指定した `rfc_address` がエンジン用の場合、`ccm monitor` コマンドは失敗します。

`-user` *`user`*

監視するユーザーを指定します。

## 例

ユーザー *kim* のエンジン プロセスを監視します。

```
ccm monitor -user kim -process engine

IBM Rational process monitor...7 process(es) located:
user     process  host     port   pid    database path
----     -------  ----     ----   ---    -------------
ccm_root router   galaxy   1514   28496  -
ccm_root objreg   orbit    34525  18273  -
ccm_root objreg   galaxy   41587  28507  -
ccm_root objreg   dbserver 62240  19592  -
linda  engine   lego    34728  21182 /vol/dbserver/ccmdb/ccm51new
linda  gui      lego    34725  21181 /vol/dbserver/ccmdb/ccm51new
linda  monitor  lego    34737  21205  -
[Press ^C to quit IBM Rational monitor.]
```

関連トピック

- fs_check コマンド
- lmgr_status コマンド
- message コマンド
- ps コマンド

# **move** コマンド

## 表記

```
ccm move file_spec [file_spec...] dest_directory
         [-task task_number]
ccm move directory [directory...] dest_directory
         [-task task_number]
ccm move file_spec new_file_spec [-task task_number]
ccm move -p|-project project_spec new_project_spec
```

## 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。

  注記：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に、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。

- 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。

  注記：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、以下のいずれかが起こります。

  Rational Synergy が自動的に新規ディレクトリ バージョンをチェックアウトします。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。タスクベースの手法を使用している場合、これはデフォルト（または指定された）タスクをチェックインするときに自動的に行われます。オブジェクト ステータス ベースの手法を使用している場合は、ディレクトリのチェックインを忘れずに行う必要があります。

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（または指定された）タスクと自動的に関連付けし、その後 *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）。

プロジェクトの 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：現在のディレクトリまたは指定されたディレクトリ以外のディレクトリまたはディレクトリ バージョンで使用されているプロジェクトまたはオブジェクトの名前を変更しようとすると、オブジェクトの新規バージョンをチェックアウトするよう要求されます。オブジェクトの名前変更では、指定オブジェクトを結合しているすべてのディレクトリの修正が必要となりますが、現在のディレクトリまたは指定されたディレクトリのみが安全に更新できます。

## オプションと引数

***dest_directory***

ファイルまたはディレクトリの移動先ディレクトリの名前を指定します。

***directory***

移動するディレクトリの名前を指定します。

***file_spec***

移動するファイルまたはディレクトリの名前を指定します。

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

名前変更するプロジェクトの名前と新規名を指定します。

最上位プロジェクトの名前を変更する場合は、このオプションを使用する必要があります。

このオプションを使用して、サブプロジェクトとして使用されているプロジェクトの名前は変更できません。

`-task` *`task_number`*

新規に作成されたディレクトリを指定タスクと関連付けます。

カレント（デフォルト）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。

## 例

- ファイル `oops.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¥oops.h incl/
  ```
  UNIX:
  ```
  ccm move src/oops.h incl/

  Member oops.h-3 removed from the project sandbox-lb
  Adding 'release' attribute with value '2.0' to object incl-2:dir:5
  Associated object incl-2:dir:5 with task 36.
  Member oops.h-3 added to project sandbox-lb
  ```

- ファイル `turquoise.c` を現在のプロジェクト内の `magenta.c` に移動する。

  ```
  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¥linda¥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'/linda/ccm_wa/ccmint07/
  test_a-1'...
  ```

- `hello.c` ファイルを `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

  Member hi_world.c-1 removed from project proj-1
  Member hi_world.c-1 added to project 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

Member hello.c-1 removed from project beta-1
Member hello.c-1 added to project final-1
```

# **ps** コマンド

## 表記

```
ccm ps -user username "message_text"
        -d|-database database "message_text"
        -rfc_address address "message_text"
        -attr value "message_text"
```

## 説明と用途

`ccm ps` コマンドは、Rational Synergy ユーザーおよびプロセスに関するネットワーク全体のプロセス状態情報を提供します。これは、`ccm monitor` の出力を最も詳細に表示したものです。

## オプションと引数

`-attr` *`value`*

監視するフィールドの名前を指定します。*`attr`* オプションに使用できる値は、`process`、`display`、`pid`、`user`、`host`、`database`、`engine_address`、`and pwa_path` のいずれかです。

一度に 1 つのスイッチのみ使用できます。

`-d|database` *`database`*

*`database`* のすべての使用を監視することを指定します。

正規表現を使用して複数のデータベースを監視できます。正規表現は、疑問符（?）で始まる必要があります。

`-rfc_address` *`address`*

表示する Rational Synergy インターフェイス（GUI）プロセスの RFC（Remote Function Call）アドレスを指定します。このフォーマットは、`host:port[:ip]` または `ip:port[:ip]` などのホスト名または IP アドレスで開始します。`[:ip]` は、ゼロ以上の IP アドレスを示します。

> 注記：指定した `rfc_address` がエンジン用の場合、`ccm process` コマンドは失敗します。

`-user` *`username`*

監視するユーザーを指定します。

## 例

- ホストアドレスが「horse.abbd0.com」のインターフェイスのプロセス情報を表示する。

  ```
  ccm ps -host horse.abcc0.com
  ```

- 名前に「training」を含むデータベースのプロセス情報を表示します。

  ```
  ccm ps -d "?training"
  ```

関連トピック

- fs_check コマンド
- message コマンド
- monitor コマンド

## **process_rule** コマンド

### 表記

フォルダおよび／またはフォルダ テンプレートのプロセス ルールへの追加

```
ccm pr|process_rule|ut|update_temp|update_template|
           rt|recon_temp|reconfigure_template -a|-ad|-add
      [-fol|-folder|-folders folder_specs]
      [-ft|-folder_temp|-folder_temps|
      -folder_template|-folder_templates folder_template_specs]
      [-q|-quiet] process_rule_specs
```

プロセス ルールのコピー

```
ccm pr|process_rule|ut|update_temp|update_template|
           rt|recon_temp|reconfigure_template -cp|-copy
     process_rule_spec1 process_rule_spec2
```

**2** つのプロセス ルールの比較

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

プロセス ルールの削除

```
ccm pr|process_rule|ut|update_temp|update_template|
             rt|recon_temp|reconfigure_template -d|-delete [-force]
     process_rule_specs
```

プロセス ルールの一覧表示

```
ccm ut|update_temp|update_template|
            rt|recon_temp|reconfigure_template -l|-list
```

プロセス ルールの修正

```
ccm pr|process_rule|ut|update_temp|update_template|
              rt|recon_temp|reconfigure_template -m|-modify
      [-fol|-folder|-folders folder_specs]
      [-ft|-folder_temp|-folder_temps|-folder_template|
      -folder_templates folder_template_specs]
      [(-bn|-baseline_name baseline_name)|
      (-lb|-latest_baseline)|(-usb|-user_selected_baseline)|
      (-lbp|-latest_baseline_projects)]
      [-brp|-baseline_release_purpose|
      -baseline_release_purposes release_purposes [(-pr|-prepend)|
```

```
      (-ap|-append)]
      [(-pb|-prep_baseline)| (-nopb|-noprep_baseline)]
      [-matching "version_matching_string"]
      [-default|-nodefault]
            process_rule_specs
```

フォルダおよび／またはフォルダ テンプレートのプロセス ルールからの削除

```
ccm pr|process_rule|ut|update_temp|update_template|
             rt|recon_temp|reconfigure_template -rem|-remove
      [-fol|-folder|-folders folder_specs]
      [-ft|-folder_temp|-folder_temps|-folder_template|
      -folder_templates folder_template_specs]
      [-q|-quiet] process_rule_specs
```

プロセス ルールの管理データベースの設定

```
ccm pr|process_rule|ut|update_temp|update_template|
            rt|recon_temp|reconfigure_template -cdb|-controlling_database
      [-local|-handover dbid|-accept dbid]
      process_rule_specs
```

プロセス ルールについての情報の表示

```
ccm pr|process_rule|ut|update_temp|update_template|
            rt|recon_temp|reconfigure_template -sh|-show keyword
     process_rule_specs
```

## 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、旧リリースの Rational Synergy 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、プロジェクトで更新操作を実行したときに、プロジェクトがどのように更新されるかを指定します。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するプロセス 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、多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 マネージャが新規リリース値を作成すると、そのリリース値に関連付けられた有効な目的ごとに 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。

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、同等な動作をします。

- 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 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。

このコマンドは、以下の操作を行うために使用します。

- プロセス ルールを編集する。
- フォルダおよび／またはフォルダ テンプレートをプロセス ルールに追加する。
- フォルダおよび／またはフォルダ テンプレートをプロセス ルールから削除する。
- プロセス ルールをコピーする。
- 2 つのプロセス ルールを比較する。
- 1 つまたは複数のプロセス ルールを削除する。
- 定義済みプロセス ルールを一覧表示する。
- プロセス ルールの管理データベースを設定する。
- 1 つまたは複数のプロセス ルールについての情報を表示する。

## オプションと引数

`-accept` *`dbid`*

オブジェクトが指定データベースからの管理を受け付けるように設定します。これにより、DCM を使用したプロセス ルールの集中管理が可能となります。

`-bn|-baseline_name` *`baseline_name`*

プロセス ルールに使用するベースライン（*`baseline_name`*）の名前を指定します。

`-brp|-baseline_release_purpose|-baseline_release_purposes` *`release_purposes`* `[(-pr|-prepend) | (-a|-append)]`

プロセス ルールのリリース／目的ペアのリストを指定します。*`release_purposes`* 値は、1 つ以上のリリース／目的項目をカンマで区切ったリストです。リリース／目的項目は、リリース、値、コロン（:）、および目的値で構成されます。リリース値としては、有効なリリース名またはキーワード `%release` または `%baseline_release` が使用できます。目的値としては、プロジェクト目的テーブルに定義されている有効目的、または値 `Any` が使用できます。リリース／目的項目は、検索優先度の降順に指定します。`-pr|-prepend` オプションを指定した場合、指定した値が既存のリストの先頭に挿入されます。`-a|-prepend` オプションを指定した場合、指定した値が既存のリストの最後に追加されます。いずれのオプションも指定しなかった場合は、指定値は現在のリストになります。

`-cdb|-controlling_database`

指定プロセス ルールに管理データベースがあることを指定します。`-local`、`-handover` または `-accept` オプションと一緒に使用して、テンプレートが定義されている場所を指定します。これらのオプションはプロセス ルールの集中管理を設定します。

`-comp|-compare` *`process_rule_spec1 process_rule_spec2`*

2 つのプロセス ルールを比較します。

`-cp|-copy` *`process_rule_spec1 process_rule_spec2`*

指定したプロセス ルールを新規プロセス ルールにコピー、または既存のプロセス ルールに上書きします。プロセス ルールのコピー時は、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、ターゲットのプ

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、ほかのプロパティはすべてソース 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、ターゲットのプロセス 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、ソース 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、ターゲットのプロセス 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ソース プロセス ルールからコピーされます。

ターゲットのリリース固有プロセス ルールも汎用プロセス ルールとの既存の関係を保持します。

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。

このオプションを使用するには、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。

`-default|-nodefault`

`-default` を指定した場合、編集しているプロセス ルールと同じリリース値と目的の組み合わせを持つすべての新規ルールが自動的にそのプロセスルールを使用します。`-nodefault` を指定した場合、プロセス ルールと同じリリース値と目的を持つすべての新規ルールがそのプロセス ルールを使用せず、手動で更新します。

このオプションを指定しない場合のデフォルトは `-default` です。

`-d|-delete` *`process_rule_specs`*

指定した *process_rule_spec* を削除します。指定したプロセス ルールが汎用であるか、または 1 つ以上のプロジェクトで使用されている場合、そのプロセス ルールは `-force` オプションを指定した場合にのみ削除されます。

`-fol|-folder|-folders` *`folder_spec`*

追加または削除するフォルダの ID を指定します。この引数の構文については、フォルダの指定 を参照してください。

注記：*`folder_spec`* が指定可能な場合は、*`folder_spec`* を含むファイル名を指定できます。

`-force`

`-force` オプションは、確認メッセージを抑止し、削除操作を強制的に実行します。

`-ft|-folder_temp|-folder_temps|-folder_template|-folder_templates` *`folder_template_specs`*

追加または削除するフォルダ テンプレートを指定します。

`-handover` *`dbid`*

オブジェクトの管理を現在のデータベースから指定データベースに渡します。このオプションは DCM を使用したプロセス ルールの集中管理を設定します。

`-lb|-latest_baseline`

作成時刻が最新で、一致する *`baseline_release`* と *`baseline_purpose`* を持つベースラインは、プロジェクトのインスタンス化のためのベースラインとして使用されます。このオプションを使用すると、オプション `-baseline_name`、`-latest_baseline_projects`、`-prep_baseline`、`-matching` は使用できなくなります。

このオプションを使用すると、現在選択されているプロセス ルールが、いずれか 1 つのリリース／目的ペアと一致するベースラインの、空白以外のプールからベースラインを選択します。`latest_baseline` を使用してベースライン プロジェクトを選択できない場合、プロジェクトはベースラインを持ちません。

`-lbp|-latest_baseline_projects`

このテンプレートを使用する各プロジェクトは、更新動作の実行時に自身のプロジェクト ベースラインを選択します。ベースライン プロジェクトとは、一致する *`baseline_release`* と *`baseline_purpose`*ÅA および *`version_matching_string`* と一致するバージョンを持つプロジェクトのことです。`-prep_baseline` オプションと `-matching` オプションは、このオプションと一緒にのみ使用できます。

`-local`

ローカル管理を設定し、他のデータベースから作成された DCM 複製があれば破棄するよう指定します。

`-l|-list [`*`scope`*`]`

定義済みプロセス ルールを一覧表示します。

`-matching "`*`version_matching_string`*`"`

関連するベースラインを、プロセス ルールを使用するプロジェクトと比較します。

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します。このフィールドは、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` のプロジェクトを使用することを示すには、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します。

`-nopb|-noprep_baseline`

このオプションが関係するのは、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` がある場合のみです。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。

`-pb|-prep_baseline`

このオプションが関係するのは、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` がある場合のみです。これは、静的プロジェクトと *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。

`-quiet`

表示される出力メッセージ数を減らします。このオプションはスクリプト作成時に便利です。

`-rem|-remove`

指定したタスクまたはフォルダをプロセス ルールから削除します。

`-sh|-show` *`keyword`*

指定したプロセス ルールについての情報を表示します。

このオプションでは以下のキーワードを使用できます。

- `baseline_projects`

  プロセス ルールのベースライン プロパティと一致するベースライン プロジェクトを表示します。

- `brp|baseline_release_purpose|baseline_release_purposes`

  プロセス ルールのベースライン リリースと目的を表示します。

- `bsm|baseline_selection_mode`

  以下のいずれか 1 つのベースライン選択モードを表示します。

  *`Baseline_name`*

  `Latest Baseline`

  `User-Selected Baseline`

  `Latest Baseline Projects`

- `default`

プロセス ルールがデフォルトで新規プロジェクトに使用されるかどうかを示します。

- `fol|folder|folders`

  プロセス ルールで使用されるフォルダを表示します。

- `ft|folder_temp|folder_temps|folder_template|folder_templates`

  プロセス ルールで使用されるフォルダ テンプレートを表示します。

- `i|info|information`

  指定したテンプレートのすべてのプロパティを表示します。

- `matching`

  プロセス ルールのバージョン一致文字列を表示します。

- `members`

  プロセス ルールで使用されるタスク、フォルダ、およびフォルダ テンプレートを表示します。

- `pb|prep_baseline`

  *prep* ベースラインがプロセス ルールに対して適したベースラインであるかどうかを示します。

`-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。

関連トピック

- process_rule コマンドの例

## **process_rule** コマンドの例

以下の操作の例について説明します。

- フォルダおよび／またはフォルダ テンプレートの追加
- 2 つのプロセス 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 ルールの修正
- プロセス ルールの管理データベースの設定
- プロセス ルールに関する情報の表示

### フォルダおよび／またはフォルダ テンプレートの追加

- 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"

Added the following folder(s) to process rule 2.1:Insulated Development
   Folder 3
Added 1 folder to process rule 2.1:Insulated Development.
```

- フォルダ テンプレート `xxx` 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"

   Added the following folder template(s) to process rule 2.1:Insulated
Development
      Folder template integration tested tasks for release %release
   Added 1 folder template to process rule 2.1:Insulated Development.
```

### 2 つのプロセス ルールの比較

- 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"

   Baseline selection for process rule toolkit/2.0:Collaborative Development
    Baseline Selection Mode: Latest Baseline Projects
    Prep Allowed:            No
    Versions Matching:
    Release Purposes:

   Baseline selection for process rule toolkit/2.0:Insulated Development
    Baseline Selection Mode: Latest Baseline Projects
    Prep Allowed:            No
    Versions Matching:
    Release Purposes:

   New Projects use process rule toolkit/2.0:Collaborative Development by default
   New Projects use process rule toolkit/2.0:Insulated Development by default

   Folder Templates and Folders only in process rule toolkit/
2.0:Collaborative Development
    Template All Completed Tasks for Release %release for Collaborative
projects from Database %database

   Folder Templates and Folders only in process rule toolkit/2.0:Insulated
Development
    Template Integration Tested Tasks for Release %release from Database
%database
    Template Master Integration Tested Tasks for Release %release

   Folder Templates and Folders in both process rules
   Template Assigned Or Completed Tasks for %owner for Release %release from
Database %database
    Template Miscellaneous Tasks for %owner for Release %release
```

### プロセス ルールのコピー

- 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"
```

### プロセス ルールの削除

- 2.1:Shared プロセス ルールを削除する。

```
ccm pr -delete "2.1:Shared"
```

### 現在のプロセス ルールの一覧表示

- 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
```

### プロセス ルールの修正

- 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編集する。

```
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 ルールのための名前 `Build_1234_int` を持つベースラインを選択する。

```
ccm process_rule -modify -bn Build_1234_int "2.0:Insulated Development"
```

### プロセス ルールの管理データベースの設定

- `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 ルールに関する情報の表示

- `2.1:Insulated Development` プロセス ルールのすべてのプロパティを表示する。

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

## **project_grouping** コマンド

### 表記

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

```
ccm pg|project_grouping
          -at|-add_task|-add_tasks task_spec|all_removed
         project_grouping_spec
```

プロジェクト グルーピングから他のプロジェクト グルーピングへのタスクのコピー

```
ccm pg|project_grouping
          -ct|-copy_tasks
         project_grouping_spec1
         project_grouping_spec2
```

プロジェクト グルーピングとそのメンバー プロジェクトの削除

```
ccm pg|project_grouping
          -d|-delete [-m|-members|-nm|-no_members]
         project_grouping_spec
```

プロジェクト グルーピングの一覧表示

```
ccm pg|project_grouping -l|-list
         [-r|-release release]
         [-purpose purpose_spec]
         [-o|-owner owner_spec]
         [-f|-format "format_string"]
         [-ns|-no_sort]
         [-u|-un_numbered]
         [-sep separator_char]
         [-nf]
```

プロジェクト グルーピングのベースラインとタスクの再表示

```
ccm pg|project_grouping
          -rbt|-refresh|-refresh_baseline_and_tasks
         project_grouping_spec
```

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

```
ccm pg|project_grouping
          -rt|-remove_task|-remove_tasks task_spec|all|all_added
         project_grouping_spec
```

### プロジェクト グルーピングの自動再表示モードの設定

```
ccm pg|project_grouping
          -ar|-auto_refresh_baseline_and_tasks|-thaw
          project_grouping_spec
ccm pg|project_grouping
          -no_ar|-no_auto_refresh_baseline_and_tasks|-freeze
          project_grouping_spec
```

### プロジェクト グルーピングのプロパティの表示

```
ccm pg|project_grouping -sh|-show keyword
          project_grouping_spec
          [-f|-format "format_string"]
          [-ns|-no_sort]
          [-u|-un_numbered]
          [-sep separator_char]
          [-nf]

ccm pg|project_grouping -sh|-show
          i|info|information|
          n|name
          r|release|
          p|purpose|
          created_in
          o|owner|
          ar|auto_refresh_baselines_and_tasks
          rtime|refresh_time
          project_grouping_spec
```

## 説明と用途

プロジェクト グルーピングは、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するために使用します。プロジェクト グルーピングのタスクとベースライン 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、またはプロジェクト

グルーピングが My CM/7.1a Insulated Development Projects などのローカル データベースに作成されている。

- *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、またはプロジェクト グルーピングが Mary's CM/7.1a Insulated Development Projects などのローカル データベースに作成されている。
- 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 用に初期化されている、またプロジェクト グルーピングが My CM/7.1a 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ない。
- *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 用に初期化されている、またプロジェクト グルーピングが Mary's CM/7.1a 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ない。

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/7.1a 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/7.1a 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 グルーピングは必ず、プロセス ルールを 1 つだけ持っています。

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。しかし、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、『ビルド マネージャ ガイド』を参照してください。

`project_grouping` コマンドは以下の目的で使用します。

- 特定のプロジェクト グルーピングの情報、または関連するプロジェクト、オブジェクトおよびタスクを表示する。
- プロジェクト グルーピングを一覧表示する。
- プロジェクト グルーピングを修正する。
- 既存のプロジェクト グルーピングを削除する。

ユーザーはプロジェクトの作成後、ベースラインとタスクの更新、タスクの追加と削除、グルーピングの更新、グルーピングへのタスクのコピー、自動更新オプションの指定、およびグルーピングの削除を行うことができます。

## オプションと引数

*`project_grouping_name`*

*`project_grouping_name`* はプロジェクト グルーピングに割り当てられた名前です。

`-at|add_task|add_tasks`

指定した 1 つまたは複数のタスクをプロジェクト グルーピングに追加します。

タスクが `Removed Tasks` 内にある場合、タスクは `Removed Tasks` から削除されて、`Saved Tasks` に追加されます。その場合、指定されたタスクが依存している必須タスクは、`include_required_tasks` オプションの設定に関わらず、追加されません。

タスクが `Removed Tasks` と `Saved Tasks` のいずれにもない場合、タスクは `Added Tasks` に追加されます。`include_required_tasks` オプションが TRUE の場合、必須タスクが計算され、`Added Tasks` に追加されます。

文字列 `all_removed` が *`task_spec`* に指定されている場合、削除されたタスクがすべて元に戻され、必要なタスクの計算は行われません。

処理中にこのコマンドを取り消す場合は、必ず操作の状態に関するメッセージを確認してください。取り消し前に追加されたタスクがあるかどうかで、結果が異なります。

`-ar|-auto_refresh_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の更新を行うことを指定します。

`-ct|-copy_tasks` *`project_grouping_spec1 project_grouping_spec2`*

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` オプションが使用され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。

`-delete`

指定した *`project_grouping_name`* を持つプロジェクト グルーピングを削除します。

`-no members` を指定するか、または `-members` を指定しない場合は、プロジェクト グルーピングは内部にプロジェクトを持たない場合にのみ、削除されます。内部にプロジェクトを持つ場合、コマンドは失敗します。

`-members` を指定した場合、プロジェクト グルーピングは関連するすべてのプロジェクトとともに削除されます。またどのプロジェクトやプロジェクト グルーピングでも使用されていない関連フォルダもすべて削除されます。

`-description "`*`project_grouping_description`*`"`

オプションで、プロジェクト グルーピングの詳細な説明を提供します。説明の長さと内容には制限はありません。*`project_grouping_description`* に 1 つまたは複数の空白が含まれる場合は、二重引用符で囲む必要があります。

`-f|-format "`*`format_string`*`"`

コマンドの出力フォーマットを指定します。デフォルトのフォーマットは、`-format` と一緒に使用するオプション（たとえば、`-list` または `-show`）とこれらのオプショ

ンのキーワード引数によって異なります。デフォルトの出力フォーマットの詳細については、-format と一緒に使用できるオプションの説明を参照してください。

このフォーマットには、テキストとキーワードの組み合わせを含むことができます。キーワードは、各オブジェクトについての特定のデータに置き換わります。たとえば、ユーザー sue が所有するオブジェクトに関する情報が表示される場合、キーワード %owner は sue に置き換わります。

-list

データベース内のすべてのプロジェクト グルーピングを一覧表示します。-release、-purpose または -owner を指定すると、指定と一致するプロパティを持つプロジェクト グルーピングのみが一覧表示されます。

デフォルト フォーマットは %displayname ですが、-format オプションを指定すると無効となります。

このコマンドは、返されるプロジェクト グルーピングのリストの選択セットを設定します。

-nf

出力を縦型フォーマットで表示しないことを指定します。

-no_ar|-no_auto_refresh_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。

-ns|-no_sort

コマンドの出力をソートしないように指定します。

-purpose *purpose_spec*

一覧表示するプロジェクトの目的を指定します。

-rbt|-refresh_baseline_and_tasks

プロジェクト グルーピングの自動再表示モードの設定に関わらず、ベースラインとタスクを更新してプロジェクト グルーピングに保存します。プロジェクト グルーピ

ングが凍結されている場合でも、ベースラインとタスクが更新されます。ただし、プロジェクト グルーピングの自動更新設定は変更されません。

`-rt|-remove_task|-remove_tasks` *`task_spec`*`|all|all_added`

プロジェクト グルーピングから 1 つまたは複数のタスクを削除します。

タスクが `Added Tasks` 内にある場合、タスクは `Added Tasks` から削除されます。タスクが `Saved Tasks` 内にある場合、タスクは `Saved Tasks` から削除されて、`Removed Tasks` に追加されます。いずれの場合も、依存タスクが計算されて削除されることはありません。

`all` を指定すると、すべての `Added Tasks` が削除され、すべての `Saved Tasks` が `Removed Tasks` フォルダに追加されます。タスクの削除操作は、各 `Added Task` および各 `Saved Task` に適用されます。

`all_added` を指定すると、`Added Tasks` からすべてのタスクが削除されます。タスクの削除操作は、各 Added Task に適用されます。

`-sep` *`separator_char`*

`-format` オプションと一緒にのみ使用します。*`format string`* 内で複数の入力フィールドを区切るために使用する区切り文字を指定します。この文字を指定すると、コマンド出力はカラム内のフィールドが空白文字で区切られて表示されます。

`-show`

指定した *`project_grouping_name`* と関連付けられたプロパティを表示します。

`i|info|information`

このオプションを指定すると、`name`、`release`、`purpose`、`owner` および `created_in` 情報が表示されます。

`n|name`

このオプションを指定すると、プロジェクト グルーピングの名前が表示されます。

`r|release`

このオプションを指定すると、プロジェクト グルーピングのリリース値が表示されます。

`p|purpose`

このオプションを指定すると、プロジェクト グルーピングの目的が表示されます。

`o|owner`

このオプションを指定すると、プロジェクト グルーピングの所有者名が表示されます。

`created_in`

このオプションを指定すると、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が表示されます。

`rtime|refresh_time`

このオプションを指定すると、ベースラインとタスクを最後に計算（および保存）した時刻が表示されます。

-- 以下のキーワードは、`-show:` と一緒に使用できます。

`proj|projects`

このオプションを指定すると、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が表示されます。デフォルト フォーマットは以下のとおりです。

```
%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`

このオプションを指定すると、ベースライン名が表示されます。デフォルト フォーマットは以下のとおりです。

```
%displayname: %description
```

`fo|folders`

このオプションを指定すると、プロジェクト グルーピングのフォルダが表示されます。デフォルト フォーマットは以下のとおりです。

```
%displayname
```

`ar|auto_refresh_baselines_and_tasks`

このオプションを指定するとプロジェクト グルーピングの自動再表示モードが表示されます。自動更新は、`TRUE`（有効または凍結解除）と `FALSE`（無効または凍結）のいずれかです。

`st|saved_tasks`

このオプションを指定すると、プロジェクト グルーピングの Saved Tasks 内のすべてのタスクが表示されます。Saved Tasks とは、ヘッダー「ベースラインで処理されたタスク」の下にある ベースラインとタスク タブの Rational Synergy プロジェクト グルーピング プロパティ ダイアログに表示されているタスクです。これらのタスクは、プロジェクト グルーピングのプロセス ルールから決定されたタスクです。

デフォルト フォーマットは以下のとおりです。

```
%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。

`at|added_tasks`

このオプションを指定すると、プロジェクト グルーピングの Added Tasks 内のすべてのタスクが表示されます。Added Tasks とは、ユーザーが手動で追加したタスクです。これらのタスクは、手動追加タスク タブの Rational Synergy プロジェクト グルーピング プロパティダイアログに表示されます。

デフォルト フォーマットは以下のとおりです。

```
%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。

`rt|removed_tasks`

このオプションを指定すると、プロジェクト グルーピングの Removed Tasks 内のすべてのタスクが表示されます。Removed Tasks とは、チェックボックスをチェックしていない Rational Synergy ベースラインとタスク タブに表示されるタスクです。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 フォーマットは以下のとおりです。

```
%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。

`all_tasks`

このオプションを指定すると、ネット タスク（Saved Tasks と Added Tasks）が表示されます。デフォルト フォーマットは以下のとおりです。

```
%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`

このオプションを指定すると、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が表示されます。デフォルト フォーマットは以下のとおりです。

```
%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。

`automatic_tasks`

このオプションを指定すると、プロジェクト グルーピングに必然的に関連付けられたプロセス ルールからの自動タスクが表示されます。デフォルト フォーマットは以下のとおりです。

```
%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|-un_numbered`

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）。

## 例

- `My CM/6.3 Collaborative Development` という名前を持つプロジェクト グルーピングのベースラインとタスクを再表示する。

  ```
  ccm project_grouping -refresh "My CM/6.3 Collaborative Development"
  ```

- `My Fav/5.1 Visible Projects` という名前のプロジェクト グルーピングについての情報を表示する。

  ```
  ccm pg -show info "My Fav/5.1 Visible Projects"
  ```

関連トピック

- update_properties コマンド
- process_rule コマンド

# **project_purpose** コマンド

## 表記

### プロジェクトの目的の作成

```
ccm project_purpose -cr|-create -n|-name "purpose name"
                    -stat|-status status
                    [-ms|-member_status member_status]
```

### プロジェクトの目的の削除

```
ccm project_purpose -d|-delete [-y] purpose_specs
```

### プロジェクトの目的の修正

```
ccm project_purpose -m|-modify [-n|-name "new name"]
                    [-ms|-member_status new_member_status] purpose_spec
```

### プロジェクトの目的の表示

```
ccm project_purpose -s|-sh|-show [-stat|-status status]
                    [-r|-role [role]] [purpose_spec]
```

## 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、削除、修正または表示できます。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 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。

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、削除できます。プロジェクト目的を編集するには、*ccm_admin* ロールを持っている必要があります。

プロジェクト目的テーブルは以下に影響を与えます。

- ダイアログボックス プロジェクトのコピー、プロジェクトの作成、プロパティ（プロジェクトまたは製品の場合）の 目的 フィールドに表示されるオプション。
- コマンド `ccm copy_project`、`ccm create`、`ccm create -type project` および `ccm properties` に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のデフォルト サブセットは、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
```

また、データベースが DCM 対応の場合は、次の目的も持っています。

```
Master Integration Testing:  master_integrate:    prep
```

このプロジェクト目的データベースは、1 行につき 1 つの目的を定義します。行のフォーマットは、以下のとおりです。

```
purpose_label:   status:   member_status
```

`purpose_label` は、ダイアログの Purpose フィールドに表示されるオプションとコマンドの -purpose オプションで指定する目的です。

`status` は、その目的を持ったプロジェクトの状態です。

`member_status` は、新規プロジェクトの `member_status` 属性が持つ値です。`member_status` 値は目的と同様とします。たとえば、`Test Integration` の目的を作成する場合、メンバーの状態は `test_int` などのように同様の値に設定します。この値は CLI コマンドで目的の代わりに使用できます。

オブジェクトの状態を使用して更新する場合、すべての `member_status` 値は、データベースで定義された実際の値である必要があります。タスクを使用して更新する場合、`member_status` 値には、更新時に *prep*（準備）プロジェクトや *shared*（共有）プロジェク

トを識別できるような一意の単語を使用できます。たとえば、このオプションを以下の値に設定して、システム テストの 4 つの一意レベルを定義できます。

```
Insulated Development:    working:     working
Integration Testing:      prep:        integrate
System Testing 1:         prep:        sqa1
System Testing 2:         prep:        sqa2
System Testing 3:         prep:        sqa3
System Testing 4:         prep:        sqa4
```

## オプションと引数

`-cr|-create`

プロジェクトの目的を作成します。このオプションには、`-name`、"*purpose name*" および `-status` も必要です。

また、特定の値が必要な場合は、`-member_status` オプションも指定できます。メンバーの状態を指定しないと、入力した `-status` オプションのデフォルト値として `member_status` が使用されます。

`-d|-delete` *purpose_spec*

*purpose_spec* で指定したプロジェクト目的を削除します。

プロジェクト目的を削除すると、以下のことが発生します。

- 削除した目的でプロジェクトのチェックアウト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。

`-m|-modify`

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名を指定します。

`-ms|-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似ている必要があります。たとえば、Test Integration の目的を作成する場合、メンバー状態は test_int などのように同様の値に設定します。

-n|-name *"purpose name"*

プロジェクトの目的に名前を付けます。このオプションは、作成するプロジェクト目的の名前を指定する場合は -create オプションと一緒に使用し、修正するプロジェクト目的の新規名を指定する場合は -modify オプションと一緒に使用します。

*purpose_spec*

"*purpose name*" または member_status です。

-modify オプションと一緒に使用した場合は、変更するプロジェクト目的を指定します。

*purpose_specs*

複数の *purpose_spec* を使用することを指定します。*purpose_spec* の間には少なくとも 1 つの空白文字が必要です。

-show オプションと一緒に使用した場合は、*purpose_specs* で指定したプロジェクト目的を表示します。

-r|-role [*role*]

-show オプションと一緒に使用したが、ロールを指定していない場合は、現在のロールで使用可能な目的が表示されます。-show オプションを指定して、ロールも指定している場合は、指定ロールで使用可能な目的が表示されます。

-s|-sh|-show [*purpose_specs*]

プロジェクトの目的を表示します。*purpose_specs* オプションを指定せずにこのオプションを使用した場合は、すべてのプロジェクト目的が表示されます。*purpose_specs* オプションと一緒に使用した場合は、*purpose_specs* で指定したプロジェクト目的が表示されます。

-stat|-status *status*

状態を指定します。-show オプションと一緒に使用した場合は、特定の状態が設定されたプロジェクト目的が表示されます。-create オプションと一緒に使用した場合

は、プロジェクト目的の状態を指定します。project_purpose_list からの値が、各目的オプションで使用されます。

-y

-delete オプションと一緒に使用した場合は、確認メッセージを表示せずにプロジェクト目的を削除します。

## 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
```

- 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"
Are you sure that you want to delete purpose 探est2 Purposeí? (Yes/No)
[No] Yes
```

- 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"
```

- *developer* ロールのユーザーのプロジェクト目的を表示する。

```
ccm set role developer
ccm project_purpose -s -r
 Purpose                Member Status    Status
 Insulated Development  working          working
 Shared Development     shared           shared
 Visible Development    visible          visible
 Collaborative Development
```

# **properties** コマンド

## 表記

### プロパティの表示

```
ccm prop|properties|info [-s] [-n] file_spec [file_spec...]
ccm prop|properties|info [-s] [-n] -p |
                         -project project_spec [project_spec...]
```

### 任意の一連の属性の表示

```
ccm prop|properties|info [-sep separator_char] [-nf]
                         -f|-format "format_string" file_spec [file_spec...]
ccm prop|properties|info [-sep separator_char] [-nf]
                         -f|-format "format_string"
                         -p|-project project_spec [project_spec...]
```

## 説明と用途

properties コマンドにより、1 つ以上のオブジェクトに関する情報が得られます。

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。

オブジェクト状態を使用してプロジェクトを構成する場合、exclude_status と member_status の 2 つの属性がコマンド ラインから表示されます。そうでない場合、baseline 属性が表示されます。

ccm properties コマンドを使用してオブジェクト バージョンの情報を表示する詳細については、選択セットの参照形式 を参照してください。

## オプションと引数

*file_spec*

情報を表示するファイルまたはディレクトリの名前を指定します。

-f|-format "*format_string*"

プロパティの出力フォーマットを指定します。

キーワードを指定したり、キーワードの配置を変更できます。たとえば、%objectname を指定すると、オブジェクト参照形式（*object_name-version*:*type*:*instance*）でクエリ情報が返されます。または、%displayname を使

用すると、タイプとインスタンスを含まない形式（*object_name-version*）でクエリ情報が返されます。

必須文字列には、以下のようにキーワードと文字テキストを使用します。

```
%displayname owned by
```

キーワードには、組み込まれたもの（`%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。

`-n`

オブジェクトのプロパティを表示するときに、ラベル（デフォルト モード）やヘッダー（単一行モード）を表示しません。

`-f` オプションを使用している場合には、このオプションを使用できません。

`-nf`

縦型フォーマットの出力を行いません。

このオプションは、`-f` オプションと一緒にのみ使用できます。

`-p` *project_spec*

情報を表示するプロジェクトの名前を指定します。

`-s`

情報を 1 行で表示します。

`-sep` *separator_char*

`-format` オプションと一緒にのみ使用して、`format_string` 内で複数の入力フィールド間を区切るために使用する、区切り文字を指定します。この文字を指定すると、カラム内のフィールドは空白で区切って表示されます。

`-sep` は出力には影響しません。`-nf` を使用してカラムを抑止しない限り、カラムは常に空白によって区切られます。

## 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 os_ico-jeff
name           : os_ico
version        : john
owner          : john
status         : working
type           : project
create_time    : Tue Aug 13 11:09:33 1998
modify_time    : Thu Aug 15 10:41:35 1998
platform       : <void>
release        : 2.1
task           : 6
member_status  : working
exclude_status : public shared visible
```

  プロジェクトには member_status 属性と exclude_status 属性があります。

- タスクを使用して更新を行う task_ico-2 プロジェクトの情報を表示する。

```
ccm prop -p task_ico-2
name           : task_ico
version        : 2
owner          : sue
status         : working
type           : project
create_time    : Tue Aug 13 11:09:33 1998
modify_time    : Thu Aug 15 10:41:35 1998
platform       : <void>
release        : 4.6
task          : 6
```

- 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" *
```

# **query** コマンド

## 表記

```
ccm query [-ns|-no_sort]


ccm query [-n|-name name] [-ns|-no_sort]
          [-o|-owner owner] [-s|-state state]
          [-t|-type type] [-v|-version version]
          [-i|-instance instance]
          [[-task task_number] [-db database_ID]]
          [-f|-format "format_string"] [-u] [-nf]
          [-sep separator_char] ["query_expression"]
```

## 説明と用途

query コマンドにより、クエリ式を使用してデータベース内のオブジェクトを検索します。クエリ結果は、-ns（ソートなし）を指定した場合を除いて、ソートして表示されます。

> 注記：クエリ関数にソート機能があっても、そのクエリ関数を他のクエリ演算子と結合して複合クエリを行うと、ソート機能は失われます。

クエリ結果は選択セットに入れられるので、選択セット参照構文（"*@listed_object_number*"）を使用することにより以降のコマンドで引数として使用できます。

-db オプションは、データベースが DCM 用に初期化されている場合にのみ使用可能です。

## オプションと引数

-db *database_ID*

このオプションは、-task 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、指定データベースに作成されている指定タスクに関連付けられているオブジェクトを検出します。デフォルトでは、任意のデータベースで作成されたタスクに関連付けられているオブジェクトを検出します。

-f|-format "*format_string*"

クエリの出力フォーマットを指定します。

デフォルト フォーマットは次のとおりです。

```
%displayname %status %owner %type %project %instance %task
```

ここで、%displayname キーワードは、デフォルトの区切り文字で区切られた名前とバージョンの 2 つのキーワードで構成され、%name-%version のようになります。

他のキーワードを指定したり、キーワードの配置を変更できます。たとえば、%objectname を指定すると、オブジェクト参照形式（*object_name-version*:*type*:*instance*）でクエリ情報が返されます。または、%displayname を使用すると、タイプとインスタンスを含まない形式（*object_name-version*）でクエリ情報が返されます。

必須文字列には、以下のようにキーワードと文字テキストを使用できます。

```
%object %displayname is owned by %owner
```

キーワードは、組み込まれたもの（%fullname、%displayname、%objectname）、あるいは %modify_time や %status など任意の既存属性または疑似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。

-i|-instance *instance*

インスタンス番号 *instance* を持つすべてのオブジェクトを検索します。

-n|-name *name*

*name* という名前のすべてのオブジェクトを検索します。

-nf

縦型フォーマットの出力を行いません。

-ns|-no_sort

出力のソートを行いません。

-o|-owner *owner*

*owner* が所有するすべてのオブジェクトを検索します。

"*query_expression*"

クエリの式を指定します。クエリ式については、クエリ式 を参照してください。

-s|-state *state*

*state* 状態にあるすべてのオブジェクトを検索します。

-sep *separator_char*

-format オプションと一緒にのみ使用します。*format_string* 内で複数の入力フィールド間を区切るために使用する、区切り文字を指定します。この文字を指定すると、カラム内のフィールドは空白で区切って表示されます。

-task *task_number*

指定したタスクと関連付けられているオブジェクトのみを検索します。

-t|-type *type*

タイプが *type* であるすべてのオブジェクトを検索します。

-u

このコマンド出力の自動番号付けを抑止します（"un-numbered"）。

-v|-version *version*

バージョンが *version* であるすべてのオブジェクトを検索します。

## 例

- 名前が foo.c、所有者が *mary* 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 mary
1) foo.c-1    integrate bob nasub1 csrc 1 1
2) foo.c-1.2  working   bob nasub1 csrc 1 4
3) foo.c-2    working   bob 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 *sue* のすべてのオブジェクトを一覧表示する。

```
ccm query -n foo.c -o sue -task 4
1) foo.c-1.2  working   sue csrc 1 4
```

- 名前が `brochure.doc`、所有者が *fiona* であるすべてのオブジェクトのうち、最後に修正されたオブジェクトの名前と時刻を一覧表示する。

  ```
  ccm query -n brochure.doc -o linda -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

# **reconcile** コマンド

## 表記

```
ccm rwa|reconc|reconcile]
             [-t|-task task]
             [-c|-comment string]
              [-r|-recurse|-no_recurse]
              [-iu|-ignore_uncontrolled|-cu|-consider_uncontrolled]
              [-mwaf|-missing_wa_file|-imwaf|-ignore_missing_wa_file]
              [-if file_spec]
              [-rpt|-report file_spec]
              [-s|-show]
              [-udb|-update_db|-uwa|-update_wa]
              [-p|-project] project_spec [project_spec...]
```

## 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。reconcile コマンドは、可能であれば、差分を自動的に解決します。解決できない場合、ファイル間にコンフリクトがあるものと判定して、何も行いません。以下の場合に、ファイルを自動的に更新します。

- ファイルをチェックアウトしてそのファイルを自分のワークエリア内で変更した場合、データベースは自動的に更新される。データベースからワークエリアを更新する `-update_wa` オプションを使用した場合、この処理は行われない。
- ファイルをチェックアウトして、別のワークエリアからデータベースを変更した場合、このワークエリアは自動的に更新される。

リコンサイル操作によってデータベースとワークエリアのどちらも自動的に更新できない場合、ファイルにコンフリクトがあると判定します。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もとに置かなかった。
- 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。ワークエリア コンフリクトの詳細については、ワークエリア コンフリクトを参照してください。

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 データベースとの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 セッションを確立したときに、`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>` 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、使用前にワークエリアを完全にリコンサイルしてください。

## オプションと引数

`-c|-comment` *`string`*

コメント文字列を指定します。

`-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。リコンサイル プロセス中に、ソース管理下にないファイルがあるかどうか、ワークエリアが

チェックされます。管理されていないファイルが見つかった場合、-update_db オプションも指定されていれば、それらのファイルがデータベース内に自動的に作成されて、管理下に置かれます。ファイル拡張子を使用して、ファイルのタイプが判定されます。

-consider_uncontrolled と -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。

-if|-ignore_types

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、-consider_uncontrolled オプションと一緒に使用する必要があります。

指定項目を区切るためには、カンマ、セミコロン、または空白文字を使用します。

-imwf|-ignore_missing_wa_file

ワークエリア内に存在しないファイルをリコンサイルしないよう指定します。プロジェクトのすべてのファイルをワークエリア内で必要とするのではないときに、このオプションを使用します。

-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。-ignore_uncontrolled と -consider_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、管理さ対象外のファイルは無視されます。

-mwaf|-missing_wa_file

データベースをワークエリアと比較して、ワークエリアに不足しているファイルを探します。リコンサイル プロセス中に、ワークエリア内のファイルがデータベース中のファイルと比較されます。-update_wa オプションが指定されていると、データベース内にあるがワークエリア内にはないファイルがワークエリアにコピーされます。データベースとワークエリアに同じファイルを含めるようにするために、このオプションを使用します。ただし、サイズの大きいプロジェクトについては、相当な時間がかかる可能性があります。

-nr|-no_recurse

リコンサイル対象に指定したプロジェクトに属するサブプロジェクトを、リコンサイルしないよう指定します。

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

リコンサイルするプロジェクトを指定します。

`-rpt|-report`

リコンサイル プロセスに関するテキスト レポートを生成します。

`-r|-recurse`

リコンサイル対象に指定したプロジェクトに属するサブプロジェクトを、リコンサイルするよう指定します。

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。

`-s|-show`

コンフリクトを解決せずに、そのまま表示します。これはデフォルト設定です。

`-t|-task` *`task`*

チェックアウトするオブジェクトを、指定されたタスクに関連付けます。

カレント（デフォルト）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。

`-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ワークエリアに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

- 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
  ```

- `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"
  ```

- `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¥bhoskins¥ccm_wa¥proj1-1¥src
  ```
  UNIX:
  ```
  ccm reconcile -missing_wa_file -update_wa /users/bhoskins/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
  ```

- データベースを更新せずにプロジェクト `proj1` およびそのサブプロジェクトをリコンサイルし、`*.tmp` ファイルを無視し、レポートを生成する。

  ```
  ccm reconcile -recurse -report conflict.txt -if *.tmp -project proj1-1
  ```

レポート例の一部を下に示します。

```
Reconcile Report: 06/24/99 10:02:06

Project:
   proj1-1

Options:
   Conflict Handling: Select
   Ignore Files: *.tmp
```

```
    Generate Report: c:¥temp¥conflict.txt
    Recurse Hierarchy

Conflict Summary:
    34 Project(s) reconciled
    129 Directories reconciled
    1 Work Area Change(s) to Working Object(s)
    1 File(s) Missing from Work Area
    105 Uncontrolled File(s)
    2 File(s) Ignored
    235 File(s) not in Conflict
--------------------------------------------------------------------------
    344 Total Files Examined

Conflict Details:
1 Work Area Change(s) to Working Object(s)
--------------------------------------------------------------------------
Conflict:        Work Area file: 'file.c-1' does not match the Database
File:            d:¥users¥joe¥ccm_wa¥ccm46¥proj1-1¥proj1¥x1¥file.c
File Type:       Source File
Project Name:    proj1-1
Object Name:     file.c-1
Object Status:   working
Work Area Time:  06/23/98 14:34:04
Database Time:   06/19/98 12:39:33


1 File(s) Missing from Work Area
--------------------------------------------------------------------------
Conflict:        'file.h-1' is missing from the Work Area
File:            d:¥users¥joe¥ccm_wa¥ccm46¥proj1-1¥proj1¥x1¥file.h
File Type:       Source File
Project Name:    proj1-1
Object Name:     file.h-1
```

- UNIX: `ico_june16-1` プロジェクトおよびそのサブプロジェクトをリコンサイルする。

  この例では、Rational Synergy 製品の外で 4 つのオブジェクトを修正する必要があることを想定しています。それらのオブジェクトを修正できるように、4 つのオブジェクトのコピーをワークエリア内に作成しています。2 つのオブジェクト `bufcolor.c` と `clear.c` は *working* 状態です。あと 2 つのオブジェクト `drawbuf.c` と `concat.c` は、*integrate* 状態です。これらのファイルを修正した後で、コマンド ラインからプロジェクトをリコンサイルしました。

```
ccm reconcile -p ico_june16-1
Work area reconciliation starting...
recursing hierarchy, conflicts will be automatically updated
Updating '/users/linda/ccm_wa/ccmint15/ico_june16-1'...
Updating database with file '/users/linda/ccm_wa/ccmint15/ico_june16-1/
ico_june16/src/bufcolor.c'...
Updating database with file '/users/linda/ccm_wa/ccmint15/ico_june16-1/
ico_june16/src/clear.c'...
Creating new members for project ico_june16-1 ...
Creating version 2 of concat.c-1:csrc:1 ...
Updating database with file '/users/linda/ccm_wa/ccmint15/ico_june16-1/
ico_june16/src/concat.c'...
Creating version 2 of drawbuf.c-1:csrc:1 ...
Updating database with file '/users/linda/ccm_wa/ccmint15/ico_june16-1/
ico_june16/src/drawbuf.c'...
2 new OV(s) successfully created.
concat.c-2:csrc:1
drawbuf.c-2:csrc:1
Reconciliation complete.
```

- 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 ~linda/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/linda/ccm_wa/ccmint15/ico_june16-1'...
  Discarding changes to '/users/linda/ccm_wa/ccmint15/ico_june16-1/
  ico_june16/src/colname.c'..
  Discarding changes to '/users/linda/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

- resync コマンド
- sync コマンド

- work_area コマンド
- ワークエリア コンフリクト

## reconfigure コマンド

`reconfigure` コマンドは update コマンド の別名です。

## reconfigure_properties コマンド

`reconfigure_properties` コマンドは update_properties コマンド の別名です。

## reconfigure_template コマンド

`reconfigure_template` コマンドは process_rule コマンド の別名です。

# **relate** コマンド

## 表記

```
ccm relate -s|-show [-l]
           [-l|
           -fmt|-format "format_for_from_object::format_for_to_object"]
           [n|-name relation_name]
           [-f|-from file_spec1] [-t|-to file_spec2]
ccm relate -n|-name relation_name
           -f|-from file_spec1 -t|-to file_spec2
```

## 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たりできます。

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。これらの関係を示しているテーブルについては、関係 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。

## オプションと引数

`-f|-from` *`file_spec1`*

関係を表示または作成する元のオブジェクトを指定します。

`-fmt|-format "`*`format_for_from_object`*`::`*`format_for_to_object`*`"`

`-s|-show` オプションの出力フォーマットを指定します。

`relate` コマンドのデフォルトの出力は以下のとおりです。

*`'from'_object_info relationship_name 'to'_object_info rel_create_time`*

各オブジェクトデフォルトのフォーマットは以下のとおりです。

```
%displayname %status %owner %type %project %instance %task
```

ここで、`%displayname` キーワードは、デフォルトの区切り文字で区切られた名前とバージョンの 2 つのキーワードで構成され、`%name-%version` のようになります。他のキーワードを指定したり、キーワードの配置を変更できます。

必須文字列には、以下のようにキーワードと文字テキストを使用できます。

```
%object %displayname is owned by %owner
```

キーワードには組み込まれたもの（%fullname、%displayname、%objectname）、あるいは既存の属性の名前（%modify_time や %status など）があります。

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。

各オブジェクトに別々のフォーマットを指定できます。それには、"*format_for_'from'_object*::*format_for_'to'_object*" 引数を使用します。

たとえば、task_in_folder 関係を持つ各フォルダ（'from' オブジェクト）の説明、そのフォルダ内の各タスク（'to' オブジェクト）のタスク概要を示すフォーマットを指定するには、以下のコマンドを使用します。

```
ccm relate -s -n task_in_folder -fmt "%description::%task_synopsis
```

各フォルダの出力は下記のようになります。

*folder_object_description* task_in_folder *task_synopsis*

このオプションを -l オプションと一緒に使用してはいけません。-l オプションは以下を指定することと同じです。

```
ccm relate -show -sep / -fmt "(%objectname/%status/%owner)" -f Makefile-
2:makefile:17
(Makefile-2:makefile:17 test steveh) successor (Makefile-3:makefile:17
working sue) Fri Nov 19 10:09:13 2004
ccm relate -sep / -fmt "(%objectname/%status/%owner)"
```

-l

関係の表示の際に、長い出力フォーマットを使用します。

-n|-name *relation_name*

作成または表示する関係の名前を指定します。

-s|-show

指定されたオブジェクト間の関係を表示します。

`-sep` *`separator_character`*

`-format` オプションと一緒にのみ使用します。*`format_string`* 内で複数の入力フィールド間を区切るために使用する、区切り文字を指定します。この文字を指定すると、カラム内のフィールドは空白で区切って表示されます。

`-t|-to` *`file_spec2`*

関係を表示または作成する先のオブジェクトを指定します。

## 例

- `clear-2` を `clear-1` の後継にする。

  ```
  ccm relate -n successor -f clear-1 -t clear-2
  ```

- `clear-2` と `clear-1` の関係を表示する。

  ```
  ccm relate -s -f clear-1 -t clear-2
  ```

- `clear-1` から任意のオブジェクト バージョンへの、データベース内のすべての関係を表示する。

  ```
  ccm relate -s -f clear-1
  ```

- `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

- history コマンド
- unrelate コマンド

## **release** コマンド

### 表記

#### データベースの管理

```
ccm release  -cdb|controlling_database
             [-local|-handover dbid| -accept dbid]
             [-component componentname | releasename]
```

#### 作成

```
ccm release  -c|-create [-from releasename] releasename
             [-baseline|-bl releasename]
             [-description|-desc description |
                 -description_file|-desc_file description_file]
             [-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]
             [-purposes purpose_spec |
                  -purposes_file purposes_file]
             [-phase phase]
             [-process process_name]
```

#### 削除

```
ccm release -d|-delete releasename [-force]
            [-old] releasename [[-force]
```

#### 区切り文字

```
ccm release -delimiter [-preview] old_delimiter new_delimiter
```

#### リスト

```
ccm release -l|-list [-active|-inactive] [-component componentname]
            [-old]
```

### 修正

```
ccm release  -m|-modify releasename
             [-baseline|-bl releasename]
             [-description|-desc description|
               -description_file|-desc_file description_file]
             [-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]
             [-purposes purpose_spec |
               -purposes_file purposes_file]
             [-phase phase]
```

### 名前変更

```
ccm release -rename oldreleasename newreleasename
            [-all|-local|-dbid dbids|-database_id dbids]
            [-force]
            [-preview]
            [-nocheck]
```

### 表示

```
ccm release -s|-show (information|active|allow_dcm_transfer|baseline|
               create_time|description|groups|included_releases|
               manager|modifiable_in|owner|parallel_check_out|
               parallel_check_in|phase|phase_log|purposes) releasename
```

## 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。

## オプションと引数

`-accept` *`dbid`*

指定された 1 つまたは複数のリリースについて、指定したデータベースから更新を受け付けることを指定します。

`-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。このオプションを指定しないと、アクティブなリリースと非アクティブなリリースの両方が、リストまたは修正されます。

`-baseline` *`releasename`*

新しいリリースの作成時にベースライン リリースを使用することを指定します。

`-component` *`componentname`*

リリース中のコンポーネント名を指定します。そのコンポーネント名に合致するすべてのリリース定義が修正されます。componentname は空の文字列 "" であってもかまいません。それは、コンポーネント名がヌルであるすべてのリリースを意味します。`controlling_database` オプションとともに使用して、他のデータベースからアップグレードされたすべてのリリースの制御を受け取ることができます。あるいは、`-list` オプションとともに使用します。

`-cdb|-controlling_database`

指定されたデータベースへ引き渡すか、あるいは指定されたデータベースからのみ更新を受け入れるように、DCM を設定します。

`-c|-create`

新しいリリースを作成します。

`-d|-delete releasename [-force]`

既存のリリース定義を削除します。そのリリース定義に後継のリリース定義がある場合は、それらの履歴は自動的に潰されます。リリースのプロセス ルール以外に、指

定されたリリース名を使用しているオブジェクトがあり、-force オプションが指定されていない場合は、この操作は失敗します。-force オプションが指定されていれば、そのリリース名を使用しているオブジェクトがある場合でも、警告が出されますが、操作は行われます。

-d|-delete -old releasename [-force]

アップグレード時に変換されなかったリリース定義を削除します。このコマンドは、不要となったリリースを削除するために使用します。指定されたリリース名を使用しているオブジェクトがあり、-force オプションが指定されていない場合は、この操作は失敗します。-force オプションが指定されていれば、そのリリース名を使用しているオブジェクトがある場合でも、警告が出されますが、操作は行われます。

-d|-delimiter

リリース名に使用されているリリース区切り文字を表示または変更します。

-description

リリースの 1 行の説明を指定します。複数行の説明を入力する必要がある場合には、代りに description_file オプションを使用します。

-description_path

使用する説明が格納されているファイルへのパスを指定します。

-force

確認メッセージを表示しないようにして、名前変更または削除の操作を強制的に実行します。このオプションは、-delete オプションまたは -rename オプションと一緒にのみ、使用できます。

-from *releasename*

新しいリリースの作成時にコピーすべきリリース名を指定します。

-groups

この定義を修正または削除する権限のあるグループを（既存のグループ セキュリティに基づいて）表示します。

-handover

選択したデータベースによってデータベースの管理を行うことを指定します。リリースがローカルに管理されている場合にのみ、このオプションを使用できます。

`-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。このオプションを指定しないと、アクティブ リリースと非アクティブ リリースの両方が、一覧表示または修正されます。

`-included_releases`

リリースに含めるべきリリース名を 1 つ以上指定します。この文字列には、複数のリリース名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名は使用できません。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。

`-included_releases_file`

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。

`-l|-list`

リリース名を一覧表示します。

`-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ことを指定します。

`-manager`

リリース名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。

`-m|-modify`

既存のリリースを修正します。

`-nocheck`

ユーザーが *ccm_admin* ロールを持たない場合、データベースが保護されていない場合、あるいはデータベースで他のセッションが実行されている場合でも、`-rename` コマンドを起動できるようにします。名前を変更するオブジェクトが何もない場合は、このコマンドはリリースの名前を変更しようとします。名前を変更するオブジェクトがある場合は、このコマンドはエラーとなります。クエリを実行してリリース内

でどのオブジェクトが参照されたかを判定できるまで、このオプションはオブジェクトの名前を変更するために必要なチェックを遅らせます。

`-phase` *`phasename`*

リリースのフェーズを示すリリース フェーズ値を指定します。この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。デフォルト値は、`New`、`Requirements Definition`、`Function Definition`、`Implementation`、`Validation`、および `Released` です。

`-preview`

更新されるオブジェクト数の概要を示すために、`-rename` オプションを実行できるようにします。

`-process` *`process_name`*

作成中のリリースのプロセスを指定できるようにします。指定され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 ルールは、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。

`-process` オプションは、`-purpose` オプションまたは `-purposes_file` オプションと一緒には使用できません。

`-purposes`

リリースに含めるべき目的名を 1 つ以上指定します。指定された目的のいずれかに、複数の汎用プロセス ルールがある場合、デフォルト プロセス ルールからのプロセス ルールが使用されます。各目的名は、プロジェクト目的テーブルに定義されている、有効な目的である必要があります。この文字列には、複数の目的名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目的名は使用できません。あるいは、`purposes_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。

`-purposes_file`

含めるべき目的が格納されているファイルへのパスを指定します。目的名を指定する際は、1 件ずつ復帰改行します。

`-releasename`

リリース名を指定します。ほとんどの場合、`releasename` には、`CM/6.4` のように、コンポーネント名、リリース区切り文字、および コンポーネント リリース値が含ま

れます。componentname が "None" である場合、リリース名は 6.4 のようなコンポーネント リリース値になります。これにより、下位互換性が保たれます。

リリース コンポーネント名の最大長は 64 文字です。

-r|-rename

既存のリリースの名前を変更します。指定された *oldreleasename* のリリース定義が存在する場合には、*newreleasename* に変更されます。*oldreleasename* の古いリリース テーブル エントリが存在する場合、そのエントリが削除されて、*newreleasename* の定義が存在しなければ作成されます。古いリリース名を参照しているすべてのオブジェクトは、新しいリリース名を参照するように更新されます。

新しいリリース名を使用したオブジェクトが存在する場合は、リリースがマージされることを意味します。この場合のリリースのマージは、-force オプションが指定されていないと失敗し、エラー メッセージが表示されます。一般には、-force が指定されている場合のリリースのマージは、警告は出されますが実行されます。

-s|-show information *release_name*

指定されたリリースのすべてのプロパティを表示します。

-s|-show property *release_name*

指定されたリリースの特定のプロパティを表示します。以下のプロパティ キーワードを使用できます。

```
active
allow_dcm_transfer
baseline
create_time
description
groups
included_releases
information
manager
modifiable_in
owner
parallel_check_out
parallel_check_in
phase
phase_log
purposes
```

## 例

- 指定されたリリース名が使用するオブジェクトがある場合でも、Sweet　7.1a のリリース定義を削除する。

  ```
  ccm release -delete "Sweet/7.1a" -force
  ```

- alphabets 1 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.doc¥

  UNIX:
  ccm release -create "alphabets/2.0" -from "alphabets/1.0" -
  description_file  /usr/tom/alphabets_2/features
  ```

- 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 "S Sweet" -active -noallow_dcm_transfer
  ```

- Widget release 5.2 において、リコンフィギュア時にリリース 5.1 と 5.0 からのリリース情報を使用する。この操作は主としてオブジェクトベースの CM に使用されます。

  ```
  ccm release -m "Widget/5.2" -included_releases "5.1,5.0"
  ```

- Kit Ten 製品向けにアップグレードされていない、リリース名を表示する。

  ```
  ccm release -list -inactive "Kit Ten"
  ```

- Synergy 内のすべてのアクティブ リリースを表示する。

  ```
  ccm release -list -active "Synergy"
  ```

- Synergy 7.1a に関する情報を表示する。

  ```
  ccm release -show information "Synergy/7.1a"
  ```

- リリース情報を編集して、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 sue -phase Implementation
  ```

- 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 -handover A1 -component releasename
  ```

# **resync** コマンド

## 表記

```
ccm resync -f file_spec
```

## 説明と用途

resync コマンドにより、ワークエリアからの 1 つ以上のオブジェクトでデータベースを更新します。UNIX 上では、プロジェクト内でファイル コピーを使用している場合、またはリンクを削除してそのリンクをファイル コピーで置き換えた場合にのみ、この操作を行います。

どのユーザーでもこのコマンドを実行できます。

## オプションと引数

*-f file_spec*

同期を取り直すファイルを指定します。

## 例

clear.c の同期を取り直す（ワークエリアからのファイルでデータベースを更新する）。

```
ccm resync clear.c
```

関連トピック

- reconcile コマンド
- sync コマンド

# **set** コマンド

## 表記

```
ccm set [option [value]]
```

## 説明と用途

set コマンドにより、Rational Synergy のオブジェクトを設定し、オプションの値を表示し、オプションを一覧表示できます。

引数を指定しないと、set コマンドは Rational Synergy のオプションとその値を一覧表示します。新しい値を指定せずにオプション名を指定すると、オプションの現在の値が表示されます。たとえば、自分のロールを知るためには、ccm set role と入力すると、自分のロールを示す値が表示されます。

オプションの初期値は初期設定ファイルに設定されています。

設定できるオプションには、make_format、use_format、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。

## オプションと引数

*option value*

設定または表示するオプションの名前を指定します。必要に応じて、設定する値も指定できます。

## 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
  ```

- use_format オプションの値を設定する。

  ```
  ccm set use_format "%displayname %status %owner %task %platform %release"
  ```

- text_editor の値を表示する。

  Windows:
  ```
  ccm set text_editor
  NOTEPAD.EXE %filename
  ```

UNIX:
```
ccm set text_editor
vi %filename
```

## 警告

Rational Synergy のオプションの中には、暗黙的に設定され（`wa_type`、`sync_on_derive` など）、`set` コマンドでは修正できないものがあります。

関連トピック

- unset コマンド

# **show** コマンド

## 表記

```
ccm show -p|-projects [-o|-owner owner]
        [-n|-name name] [-v|-version version]
        [-s|-state state] [-f|-format "format_string"]
        [-task task_number]
ccm show -t|-types
ccm show -mar|-migrate_auto_rules
```

## 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できます。

-p オプションと一緒に使用すると、show コマンドにより、他のオプションで設定した基準を満たすプロジェクトが一覧表示されます。

-t オプションと一緒に使用すると、show コマンドにより、データベース内で定義されているオブジェクト タイプが一覧表示されます。

-mar オプションと一緒に使用すると、show コマンドにより、タイプ定義に基づいて自動的に生成されるマイグレーション ルールが一覧表示されます。

## オプションと引数

-f|-format "*format_string*"

出力フォーマットを指定する、置換文字列を指定します。"Name: %name, Type: %type" というように、テキストと Rational Synergy のキーワードの両方を使用できます。デフォルトでは、このコマンドの出力フォーマットは、%displayname、%status、%owner、%type、%project、%instance、%task を表示します。

*"format_string"* には、以下のようにキーワードと文字テキストを使用します。

```
Name: %displayname Owner: %owner
```

キーワードには、組み込まれたもの（%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。

`-mar|-migrate_auto_rules`

タイプ定義に基づいて自動的に生成される、マイグレーション ルールを表示します。

`-n|-name` *`name`*

名前が *`name`* であるオブジェクトのみを表示します。

`-o|-owner` *`owner`*

*`owner`* が所有するオブジェクトのみを表示します。

`-p|-projects`

データベース内のプロジェクトを表示します。

`-s|-state` *`state`*

状態が *`state`* であるオブジェクトのみを表示します。

`-task` *`task_number`*

指定タスクに関連付けられているオブジェクトのみを表示します。

`-t|-types`

データベース内のタイプを表示します。

`-v|-version` *`version`*

バージョンが *`version`* であるオブジェクトのみを表示します。

## 例

- 状態が *integrate* であり、所有者が *mike* 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 mary
  1) projY-1   integrate mary project projY 1 2
  2) projY-2   integrate mary project projY 1 7
  3) projY-2.1 integrate mary project projY 1 8
  ```

- 状態が *integrate* であり、所有者が *mary* であり、タスク 8 に関連している、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 mary -task 8
  1) projY-2.1 integrate mary 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
```

# **soad** コマンド

## 表記

### オブジェクト リストの プレビュー／作成

```
ccm soad -preview -scope "scope_name" [arg1 [arg2 [arg3 [arg4 [arg5 ]]]]]
                [-so|-save_offline]
                [-sort|-nosort]
                [-f|-format "format"]
                [-v|-verbose]
```

### オブジェクト リストを使用して削除

```
ccm soad -delete [-pn|-package_name "package_name"]
                [-path "path"]
                [-v|-verbose]
```

### スコープを使用して削除

```
ccm soad -delete -scope "scope_name" [arg1 [arg2 [arg3 [arg4 [arg5 ]]]]]
                [-so|-save_offline]
                [-pn|-package_name "package_name"]
                [-path "package_path"]
                [-v|-verbose]
```

## 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。

## 説明と用途

soad コマンドにより、オブジェクト リストのプレビューと作成、オブジェクトのオフライン保存、オブジェクトの削除を行うことができます。保存操作により、DCM 転送セットが作成されます。それを用いて、後でオブジェクトをリストアできます。

以下に、オフライン保存と削除を行いたいタイプのデータをに示します。

- 特定開発者の不要な「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プロジェクト

SOAD コマンドのロールと制限は以下のとおりです。

スコープ ロール

どのユーザーも利用可能なスコープを参照できます。各スコープに定義されているロールにより、そのスコープを使用してオフライン保存と削除を行えるかどうかが決まります。

**Rational Synergy** ルール

Rational Synergy ルールに従って、*working* 状態の任意のプロジェクトまたはユーザーが所有しているオブジェクトを削除できます。ビルド マネージャとして作業している場合には、*prep* プロジェクトおよびオブジェクトを削除できます。*ccm_admin* ロールの場合には、working 以外の状態の任意のオブジェクトまたは他のユーザーによって所有されているオブジェクトを、オフライン保存と削除できます。*ccm_admin* ロールのユーザーだけが、オフライン保存できます。

ワークエリア

ワークエリアが見えており書き込み可能であれば、プロジェクトまたはオブジェクトを削除できます。ビルド マネージャまたは *ccm_admin* ロールの場合、ワークエリアが見えず書き込み禁止であっても、プロジェクトとオブジェクトを削除できます。ただし、先にワークエリアを手作業で整理する必要があります。

## オプションと引数

`-delete`

スコープによって指定されたデータベース オブジェクトを削除します。

削除されるオブジェクトのスコープは、以下のいずれかの方法で決まります。

- 付随する `-scope` オプションから
- オブジェクト リスト（選択セット）の前のプレビューから

以下のいずれかの条件に当てはまる場合、オブジェクトは削除される前に保存されます。

- コマンドに `-so` オプションが付けられている。
- 以前の `ccm soad -preview` コマンドに、`-so` オプションが使用された。

どちらの場合でも、そのオブジェクト名とインスタンスの最後に残っているバージョンと思われるオブジェクトは、削除されません（データベース内に保持されます）。

`-f|-format "format"`

Rational Synergy キーワードを使用して、オブジェクト リストのデフォルトの出力フォーマットを変更できるようにします。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。

このオプションは、`-preview` オプションと一緒にのみ、使用できます。

`-path "package_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見え、*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 : 自分のパッケージ名を定義する場合には、`%date` キーワードを含む名前にしてください。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。

`-preview`

選択されたスコープおよびオプションの引数を使用して、オブジェクト リスト（選択セット）を作成し、それから見つかったオブジェクトを一覧表示します。ただし、該当のオブジェクト名およびインスタンスの最後に残っているバージョンと思われるオブジェクトは除外されます（データベース内に保持されます）。

他の `ccm soad` コマンドなど、他の Rational Synergy コマンドへの入力として、このオブジェクト リストを使用できます。

注記 : プレビューの後で、`ccm soad -delete` コマンド用のオブジェクト リストを使用する前にクエリを行った場合、プレビューの結果は上書きされません。

`-scope "scope_name" [arg1 [arg2 [arg3 [arg4 [arg5 ]]]]]`

オフライン保存またはオブジェクトの削除に使用された、スコープ（修正された「クエリ」）を指定します。

*arg1* から *arg5* までの引数が必須なのは、指定されたスコープに適している場合のみです。たとえば、スコープ「`My working projects and products for a specified release`」にはリリース値 *arg1* 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。

`-so|-save_offline`

`-preview` オプションと一緒に使用した場合、オブジェクト リストを作成します。ただし、該当のオブジェクト名およびインスタンスの最後に残っているバージョンと思われるオブジェクトは除外されます（データベース内に保持されます）。

`-delete` オプションと一緒に使用した場合、オブジェクト リストを作成し、それからオブジェクトを削除する前にオブジェクトを DCM パッケージに保存します。ただし、該当のオブジェクト名およびインスタンスの最後に残っているバージョンと思われるオブジェクトは除外されます（データベース内に保持されます）。

`-sort|-nosort`

プレビューの出力をアルファベット順にソートするか、またはソート機能を無効にします。デフォルトでは、出力はソートされます。

`-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、メッセージを生成します。

## 例

- プロジェクトをオフライン保存し、プロジェクト内のオブジェクトを削除する。

  1. 削除するオブジェクトのリストをプレビューする。

     ```
     ccm soad -pr -scope "scope_name" [arguments] -so
     ```

     プレビューの結果をチェックして、正しいことを確認する。

     > 注意！プロジェクトをオフライン保存して削除する前に、必ずオブジェクト リストをプレビューしてください。結果が正しくない場合には、別のスコープを選択するか、スコープを編集するか、新しいスコープを作成します。

  2. （オブジェクト リストを使用して）オブジェクトをオフライン保存して削除する。

     ```
     ccm soad -delete [-pn "package_name"] [-path "path"]
     ```

- 指定された SOAD 転送セット内のオブジェクトをリストアする。

  ```
  ccm dcm -rec -dir dir_path -ts "soad_scope_name"
  ```

  > 注記 : 別のデータベースへ、またはデータベース ID が変更されているデータベースへ、オブジェクトをリストアできます。そのためには、-dbid *database_id* オプションを指定します。

関連トピック

- soad_scope コマンド

## **soad_scope** コマンド

### 表記

#### スコープの作成

```
ccm soad_scope -c|-create "scope_name"
               [-roles role1 role2... roleN]
               [-parameters [label1 [|label2 [|label3 [|label4 [|
                 label5 ]]]]]]
               [-object object_spec | -query "query_expression"]
               [-expand|expansion_rules "expand_rules"]
               [-exclude|-exclusion_rules "exclude_rules"]
               [-exclude_query|-exclusion_query "query_expression"]
               [-pn|-package_name "package_name"]
```

#### スコープの編集

```
ccm soad_scope -m|-modify "scope_name"
               [-roles role1 role2... roleN]
               [-parameters [label1 [|label2 [|label3 [|label4 [|
               label5 ]]]]]]
               [-object object_spec | -query "query_expression"]
               [-expand|expansion_rules "expand_rules"]
               [-exclude|-exclusion_rules "exclude_rules"]
               [-exclude_query|-exclusion_query "query_expression"]
               [-pn|-package_name "package_name"]
```

#### スコープの一覧表示

```
ccm soad_scope -list
                [-s|-scope]
                [-expand|-expansion_rules]
                [-exclude|-exclusion_rules]}
```

#### スコープの表示

```
ccm soad_scope -show "scope_name"
```

#### スコープの削除

```
ccm soad_scope -d|-delete "scope_name"
```

### 前提条件

なし

## 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、削除できます。

> 注意！ 新しいスコープを作成するときは、既存のスコープから開始して、すべての除外ルールを保存し、テスト データを使用してスコープを検証してください。

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。

## オプションと引数

**-d|-delete "*scope_name*"**

指定したスコープを削除します。

**-exclude|-exclusion_rules *'rule1'* | *'rule2'* | *...'ruleN'***

**-list** オプションと一緒に使用して、除外ルールを一覧表示します。

**-modify** オプションまたは **-create** オプションと一緒に使用して、1 つまたは複数の除外ルールを以下のように指定します。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。

上記で説明したスコープのテキストについては、リリースベースのスコープ を参照してください。

**-expand|-expansion_rules *'rule1'* | *'rule2'* | *...'ruleN'***

**-list** オプションと一緒に使用して、拡張ルールを一覧表示します。

**-modify** オプションまたは **-create** オプションと一緒に使用して、1 つまたは複数の拡張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。

上記で説明したスコープのテキストについては、リリースベースのスコープ を参照してください。

`-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')
```

`-l|-list`

すべてのスコープを表示します。

`-list` オプションには、`-scope`、`-expand`、`-exclude` のいずれかを指定する必要があります。

`-m|-modify "`*`scope_name`*`"`

指定したスコープを編集します。

`-object` *`object_spec`*

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有効な 4 部構成のオブジェクト名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。

上記で説明したスコープのテキストについては、リリースベースのスコープ を参照してください。

`-parameters [`*`label1`* `[|`*`label2`* `[|`*`label3`* `[|`*`label4`* `[|`*`label5`* `]]]]]`

`-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_expression`*`"`

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、現在のすべて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 `*`role1 role2... roleN`*

スコープの使用を許可されさている 1 つまたは複数のロールを指定します。デフォルトでは、*ccm_admin* のロールを持つユーザーのみが、このスコープを使用できます。

`"`*`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にもなります。空白文字やその他の文字も含め、これが URL 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%20is%20my%20test%20scope.xml` となります。

`-show "`*`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

## 例

- 特定のスコープの詳細を表示する。

  ```
  ccm soad_scope -show "scope_name"
  ```

- 新しいスコープを作成する。
  1. 利用可能なスコープ名を一覧表示する（既存の名前を使用することを避けるため）。

     ```
     ccm soad_scope -list
     ```

  2. 必要に応じて、展開ルールの選択肢を表示する。

     ```
     ccm soad_scope -list -expansion_rules
     ```

  3. 必要に応じて、除外ルールの選択肢を表示する。

     ```
     ccm soad_scope -list -exclusion_rules
     ```

  4. 新しいスコープを定義する。

     > 注意！ 除外ルールは一般に、リリース全体を削除するスコープに関して必要です。

     ```
     ccm soad_scope -create "scope_name"
     [-roles role1 role2... roleN]
     -object object_spec | -query "query_expression"
     [-expand_rules "expand_rules"]
     [-exclude_rules "exclude_rules"]
     [-exclude_query "query_expression"]
     [-parameters [label1 [|label2 [|label3 [|label4 [|label5 ]]]]]]
     [-package_name "package_name"]
     ```

  5. 新しいスコープを検証する。

     ```
     ccm soad_scope -show "scope_name"
     ```

- スコープを編集する。
  1. 利用可能なスコープ名を一覧表示する（既存の名前を使用することを避けるため）。

     ```
     ccm soad_scope -list
     ```

  2. 必要に応じて、展開ルールの選択肢を表示する。

```
ccm soad_scope -list -expand
```

3. 必要に応じて、除外ルールの選択肢を表示する。

   ```
   ccm soad_scope -list -exclude
   ```

4. 新しいスコープを定義する。

   注意！ 除外ルールは一般に、リリース全体を削除するスコープに関して必要です。

   ```
   ccm soad_scope -modify "scope_name"
   [-roles role1 role2... roleN]
   -object object_spec | -query "query_expression"
   [-expansion_rules "expand_rules"]
   [-exclusion_rules "exclude_rules"]
   [-exclude_query "query_expression"]
   [-parameters [label1 [|label2 [|label3 [|label4 [|label5 ]]]]]]
   [-package_name "package_name"]
   ```

5. 更新されたスコープを検証する。

   ```
   ccm soad_scope -show "scope_name"
   ```

- スコープを削除する。

  ```
  ccm soad_scope -delete "scope_name"
  ```

関連トピック

- soad コマンド

# **source** コマンド

## 表記

```
ccm source filename
```

## 説明と用途

`source` コマンドにより、*`filename`* ファイル内で見つかった Rational Synergy コマンドを実行します。*`filename`* に含まれている Rational Synergy コマンドには「`ccm`」接頭辞は付けません。`source` コマンドは、頻繁に使用するコマンド シーケンスを実行するのに便利です。

## オプションと引数

*`filename`*

Rational Synergy コマンドが格納されているファイルの名前を指定します。

## 例

- `ccm_product_cleanup`　ファイル内の一連のコマンドを実行します。これは、データベース内で使用されていないフローティング製品を選択して削除するものです (このスクリプトを実行するには、*ccm_admin* ロールを持っている必要があります)。

  ```
  ccm source ccm_product_cleanup
  ```

  `ccm_product_cleanup` ファイルには、以下のコマンドが含まれています。

  ```
  set role ccm_admin
  query -type executable "not is_bound()"
  collapse @
  query -type library "not is_bound()"
  collapse @
  set role developer
  ```

# **start** コマンド

## 表記

```
ccm start [-nogui] [-q] [-d database_pathname] [-f filename]
          [-h engine_hostname] [-m] [-r initial_role]
          [-p project_spec] [-pw password][-u pathname]
          [-n user_name] (Windows only) [-home homedir]
          [-rc] (UNIX only)
```

## 説明と用途

`start` コマンドにより、エンジンとインターフェイスを起動して、Synergy Classic または CLI のセッションを開始できます。Start ダイアログまたはコマンドラインで適切な情報を入力すると、起動の状況を表す進捗バーが表示されます。セッションの起動後で、このインターフェイス セッションの一意の ID である Rational Synergy アドレス（`CCM_ADDR`）が、`ccm_ui.log`、Message View、およびコマンド ウィンドウ（Windows）またはセッションを起動したシェル（UNIX）に表示されます。

セッションが CLI のみである場合、ホーム ディレクトリにアクセス可能なローカル ネットワーク上の任意のマシンから、コマンド プロンプト（Windows）またはシェル（UNIX）から入力したすべての Rational Synergy コマンドは、この Rational Synergy セッションによって実行されます。

`CCM_INI_FILE` 環境変数を `ccm.ini` ファイル（Windows）または `.ccm.ini` ファイル（UNIX）のパスに設定した場合、そのファイルが起動および以降の Rational Synergy コマンドに使用されます。

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。

使用する接続のタイプがパフォーマンスに影響する可能性があります。`ipcshm` プロトコルのような共有メモリ接続を通じて Rational Synergy サーバー マシンがエンジン プロセス（エンジン ホスト）を実行する場合、リモート接続（`soctcp` または `tlitcp` プロトコル）に比べてパフォーマンスがよくなる可能性があります。

> 注記 : UNIX クライアントで Synergy Classic または CLI を実行しているが、起動時に `-h` オプションを使用してエンジン ホストを指定しなかった場合、エンジン プロセスはローカル マシン上で起動されます。パフォーマンスを向上するため、`ccm start -h` *server_name* を使用して、Rational Synergy サーバー マシン上でエンジン プロセスを実行するように指定できます。

Windows のオプションと引数については、オプションと引数 (Windows) を参照してください。

UNIX のオプションと引数については、オプションと引数（UNIX）を参照してください。

## オプションと引数 (Windows)

`-d` *`database_pathname`*

データベースの絶対パスを指定します。デフォルトは `ccm.ini` ファイルに定義されています。

`-f` *`filename`*

`ccm_test.ini` のような、別の `ccm_test.ini` ファイルを指定します。代替初期設定ファイルに関しては、フル パス名を使用する必要があります（例、`c:¥ccm¥new_inits¥ccm.ini`）。パス名に空白文字を含めることはできません。

`-h` *`engine_hostname`*

エンジンを実行するマシンを指定します。

`-home` *`homedir`*

ユーザーのホーム ディレクトリへのパスを指定します。

`-m`

複数のセッションを許可します。`CCM_ADDR` が設定されているコマンド プロンプトから実行するすべての Rational Synergy コマンドでは、そのアドレスで指定されたインターフェイス プロセスを使用します。そのアドレスはセッション起動時に表示されます。

例：`set CCM_ADDR=murray:2775`

`murray:2775` は、セッション起動時に表示される `CCM_ADDR` の値です。

このオプションを使用して起動したセッションでは、コマンドを実行できるようにするため `CCM_ADDR` を設定する必要があります。

`-n` *`user_name`*

指定されたユーザーに関して、セッションの起動をスクリプト化できます。

`-nogui`

`-nogui` を指定すると、Rational Synergy は GUI のサポートなしで起動されます。このモードでは、CLI のみ利用可能です。

`-p` *`project_spec`*

起動時のフォーカス プロジェクトの名前とバージョンを指定します。起動 プロジェクトを指定しないと、前のセッションからのフォーカス プロジェクトとプロジェクト履歴がデフォルトの設定により自動的にスタックされます。

`-pw` *`password`*

パスワードを指定します。このコマンドは通常、ダイアログを立ち上げずに、スクリプトのセッションを起動したいときに使用します。

`-q`

quiet モードでセッションを起動します。このオプションを使用すると、以下のようになります。

- Synergy Classic セッションの起動時に、Startup 画面が表示されない。
- stdout へ出力されるのは、インターフェイス プロセスの `CCM_ADDR` のみ。

ESD が有効になっている場合、このオプションを使用するためには、ユーザー名とパスワードを入力するか、あるいは `.ccmrc` ファイルが存在する必要があります。そうでない場合、Synergy サーバーを信頼されるホスト用に構成する必要があります。信頼されるホストの詳細については、該当する『管理者ガイド』を参照してください。

`-r` *`initial_role`*

起動時に割り当てるロールを指定します。指定するロールは自分に許可されたものである必要があります。それ以外の場合は、セッションは起動されません。

`-u` *`pathname`*

リモート クライアント セッションの実行時に、データベース情報をコピーする先のパス名を指定します。しかし、自分の PC からそのデータベース パスにアクセスできる場合は、このオプションを省いてもかまいません。

デフォルトは c:¥temp¥ccm です。start コマンドに -u オプションを使用するか、または ccm.ini ファイルの [Options] セクションで新しいパスに ui_database_dir を設定して、この場所を変更できます。

## 例

- 指定されたエンジンとデータベースを使用して、Synergy Classic を起動する。

  ```
  ccm start -h cwi -d ¥¥dbserver1¥ccmdb¥myproject
  ```

- quiet モードでセッションを起動する（スプラッシュ画面を表示しない）。

  ```
  ccm start -q -pw password -h engine_hostname -d database_path
  ```

start コマンドの詳細については、警告を参照してください。

## オプションと引数（**UNIX**）

`-d database_pathname`

データベースの絶対パスを指定します。デフォルトは .ccm.ini ファイルに定義されています。

`-f filename`

.ccm_test.ini のような、別の .ccm.ini ファイルを指定します。代替初期設定ファイルに関しては、フル パス名を使用する必要があります（例、/users/sue/new_inits/.ccm.ini）。パス名に空白文字を含めることはできません。

`-h engine_hostname`

エンジンを実行するマシンを指定します。

`-home homedir`

ユーザーのホーム ディレクトリへのパスを指定します。

`-m`

複数のセッションを許可します。CCM_ADDR が設定されているシェルから実行するすべての Rational Synergy コマンドは、そのアドレスで指定されたインターフェイス プロセスを使用します。セッションの起動時にそのアドレスが表示されます。

例：`export ccm ADDR=murray:2775:`

murray:2775 は、セッションを起動したときに表示される、CCM_ADDR の値です。

このオプションを使用して起動したセッションでは、コマンドを実行できるようにするため CCM_ADDR を設定する必要があります。

-nogui

-nogui を指定すると、Rational Synergy は GUI のサポートなしで起動されます。このモードでは、CLI のみ利用可能です。

Rational Synergy コマンドの出力はシェル内に表示されます。

-p *project_spec*

起動時のフォーカス プロジェクトの名前とバージョンを指定します。起動 プロジェクトを指定しないと、前のセッションからのフォーカス プロジェクトとプロジェクト履歴がデフォルトの設定により自動的にスタックされます。

-pw *password*

パスワードを指定します。このコマンドは通常、ダイアログを立ち上げずに、スクリプトのセッションを起動したいときに使用します。

ファイアウォールを通じて接続するセッションでは、このオプションを使用して、有効なパスワードを指定する必要があります。パスワードを指定しないと、ccm start コマンドを続行するためにパスワードを入力するよう促されます。

-q

quiet モードでセッションを起動します。このオプションを使用した場合、stdout への出力は、インターフェイス プロセスの CCM_ADDR のみになります。

-r *initial_role*

起動時に割り当てるロールを指定します。指定するロールは自分に許可されたものである必要があります。それ以外の場合は、セッションは起動されません。

-rc

リモート クライアントとしてセッションを起動するよう指定します。ファイアウォールを通じて接続する Rational Synergy セッションに関しては、リモート クライアント モードが自動的に有効に設定されます。

リモート クライアント セッションでは、インターフェイスから見えない Rational Synergy データベース（たとえば、インターフェイス ホスト上で NFS にマウントされていないデータベース）にアクセスできます （ただし、データベースはエンジンから見える必要があります)。

最初にリモート クライアント セッションを起動するときに、データベース ファイルの一部が `/tmp/ccm/database_path` と呼ばれるローカル ディレクトリにコピーされます。start コマンド上で `-u` オプションを使用するか、または `.ccm.ini` ファイルの [Options] セクションで新しいパスに `ui_database_dir` を設定して、この場所を変更できます。

リモート クライアント セッションで作業するときは、シンボリック リンクされたデータベース ファイルを使用するのではなく、自分のワークエリア内のファイルのコピーを使用します。

`-u` *`pathname`*

リモート クライアント セッションの実行時に、データベース情報をコピーする先のパス名を指定します。このオプションは `-rc` オプションと一緒にのみ使用されます。

デフォルトは `/tmp/ccm` です。`start` コマンドに `/-u` オプションを使用するか、または `.ccm.ini` ファイルの `[Options]` セクションで新しいパスに `ui_database_dir` を設定して、この場所を変更できます。

## 例

- デフォルトの Project View 起動を使用して、Synergy Classic を起動する。

  ```
  ccm start
  ```

- 指定されたエンジン ホストとデータベースを使用して、Rational Synergy を起動する。

  ```
  ccm start -h remoteHP -d /mnt/dev/ccmdb/myproject
  ```

- 別名を作成するか、または `-q` オプションを用いたスクリプトを使用して別のセッションを起動し、そのアドレスを設定する。

  ```
  alias ccmstart  export CCM_ADDR=`ccm start -m -q $*`
  ```

  または

  ```
  #!/bin/sh
  export CCM_ADDR=`ccm start -m -q -nogui`
  ```

注記：Rational Synergy のコマンド スクリプトには、この方法を使用してください。

## 警告

追加の Rational Synergy セッションを起動し、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、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* セッションを他のユーザーが実行している *ccm_root* セッションから区別できるようになります。

## 環境変数

CCM_ADDR

## ファイル

ccminit （Windows）または .ccminit （UNIX）- （起動時に実行すべき一連のコマンド、たとえば alias r update）

ccm_ui.log（ユーザー インターフェイス ログ ファイル）

ccm_eng.log（エンジン ログ ファイル）

ccm.ini（初期設定ファイル - Windows）または .ccmi.ini （初期設定ファイル - UNIX）

関連トピック

- stop コマンド

# **status** コマンド

## 表記

```
ccm status
```

## 説明と用途

`status` コマンドにより、セッションに関する情報を表示します。表示される情報には、各インターフェイス プロセスのアドレスと使用するデータベースなどがあります。ワークエリア内にいる場合は、現在のプロジェクトの名前も表示されます。

## オプションと引数

なし

## 例

- 現在のユーザーの状態を表示する。

```
ccm status

Rational Synergy sessions for user mary:

Graphical Interface @ toto:2531 (current session)
Database: /users/mb/devccmdb/test/db

Current project: 'rainbow platform_name 2.2'
```

- 実行中のセッションの状態を表示する。

```
choochoo[121]: ccm status

Rational Synergy sessions for user npoulin:

Graphical Interface @ choochoo:34721:192.187.201.84
Database: /vol/dbserver.2/ccmdb/test_ccm51new

command Interface @ choochoo:34732:192.187.201.84
Database: /vol/dbserver.2/ccmdb/test_ccm51new

command Interface @ choochoo:34749:192.187.201.84
Database: /vol/dbserver.2/ccmdb/test_ccm51new
```

- セッションが実行されていないときに、状態を表示する。

```
choochoo[131]: ccm status
Rational Synergy sessions for user mary:

    No sessions found.
```

関連トピック

- monitor コマンド

# **stop** コマンド

## 表記

```
ccm stop|quit
```

## 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。

## オプションと引数

なし

## 例

- Rational Synergy の現在のセッションを終了する。

```
ccm stop
```

関連トピック

- start コマンド

# **sync** コマンド

## 表記

```
ccm sync [-r|-recurse] [-nr|-no_recurse]
         [-p|-project] project_spec [project_spec...]
```

## 説明と用途

sync コマンドにより、プロジェクトのワークエリアを作成または更新します。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、ホーム ディレクトリの下の ccm_wa の後ろにデータベース名を続けたものです。ワークエリアの同期を強制的に取るために、sync コマンドを使用します。

> 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。

プロジェクトを作成したとき、および check out 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、ワークエリアの同期を強制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。

- ワークエリアのパスの変更時に work_area コマンドが失敗した場合

  work_area コマンドを使用するか、Work Area Properties ダイアログを使用するか、またはプロジェクトを移動させてワークエリアのパスを変更すると、Rational Synergy はワークエリア パスを新しいロケーションに合わせて変更しようとします。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起動します。
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。

プロジェクトのワークエリアが既に存在しており、同じタイプのものであれば、reconcile コマンド を使用してそのワークエリアを更新できます（たとえば、データベースから切り離して作業した後など）。reconcile コマンドは sync コマンドと似ていますが、より多くのコンフリクト処理オプションを備えています。

## オプションと引数

注記 : CLI から 同期を中止するには、任意の時点で Ctrl + C を押します。

**Project View** を開いた状態で（Synergy Classic）CLI から同期を実行すると、**Work Area Update Status** ダイアログが表示されます。

同期処理を中止するには、**Work Area Update Status** ダイアログの **Stop** ボタンをクリックします。

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、問題の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。

`-nr|-no_recurse`

プロジェクトの同期において、プロジェクト階層の再帰処理を行いません。指定されたプロジェクトだけを同期させます。

`-p|-project` *`project_spec`* `[`*`project_spec`*`...]`

同期させるプロジェクトを指定します。

`-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。これはデフォルト設定です。

`-s|-static`

既存の静的ワークエリアを、データベース上の現在のデータで更新します。静的ワークエリアとは、静的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できます。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-mary` 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-mary
  ```

- 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。

- save_to_wastebasket
- wastebasket
- wa_path_template
- sync_output

関連トピック

- reconcile コマンド
- resync コマンド
- work_area コマンド

# **task** コマンド

## 表記

### タスクの割り当て

```
ccm task -as|-assign task_specs -t|-to resolver -q|-quiet
```

### オブジェクト、既存タスク、変更依頼とのタスクの関連付け

```
ccm task -a|-associate|-relate task_spec
         {[-obj|-object file_spec [file_spec...]] | [-fixes task_spec] |
         [-cr|-change_request|-problem]  change_request_spec]}
```

### オブジェクト、既存タスク、変更依頼に対するタスクの関係解除

```
ccm task -d|-disassociate|-unrelate task_spec
         {[-obj|-object file_spec [file_spec...]] | [-fixes task_spec] |
         [-change_request|-problem]  change_request_spec]}
```

### タスクの完了

```
ccm task -complete|-ci|-checkin task_spec|default
         [-c|-comment "string"]
         [-time|-time_actual task_duration] -y
```

### タスクのコピー

```
ccm task -cp|-copy -s|-synopsis "string"
         [-rel|-release release]
         [-p|-priority priority]
         [-r|-resolver resolver]
         [-sub|-subsystem subsystem]
         [-plat|-platform platform]
         [-time|-time_estimate time_estimate]
         [-date date_estimate] [-no_objects]
         [-register]
         [-description "description"]
         [-descriptionedit]
         [-descriptionfile file_path]
         [-def|-default] [-q|-quiet]
         task_specs
```

### タスクの作成

```
ccm task -cr|-create -s|-synopsis "string"
         [-p|-priority priority]
         [-plat|-platform platform]
         [-r|-resolver resolver]
         [-rel|-release release]
         [-sub|-subsystem subsystem]
         [-time|-time_estimate time_estimate]
         [-date date_estimate]
         [-description "description"]
         [-descriptionedit]
         [-descriptionfile file_path]
         [-def|-default] [-q|-quiet]
```

### タスクの修正

```
ccm task -fix -s|-synopsis "string"
         [-rel|-release release]
         [-p|-priority priority]
         [-r|-resolver resolver]
         [-sub|-subsystem subsystem]
         [-plat|-platform platform]
         [-time|-time_estimate time_estimate]
         [-date date_estimate] [-register]
         [-exclude]
         [-description "description"]
         [-descriptionedit]
         [-descriptionfile file_path]
         [-def|-default] [-q|-quiet]
         task_specs
```

### タスクの変更

```
ccm task -mod|-modify
         {[-p|-priority priority]
         [-plat|-platform platform]
         [-r|-resolver resolver]
         [-rel|-release release]
        [-s|-synopsis "string"]
         [-sub|-subsystem subsystem]
         [-time|-time_estimate time_estimate]
         [-date|-date_estimate date_estimate]}
         [-description "description"]
         [-descriptionedit]
         [-descriptionfile file_path]
         task_specs -q|-quiet
```

タスクのクエリ

```
ccm task -qu|-query query_spec
         [-f|-format "format_string"] [-ns|-no_sort] [-u]
         [-in_rel|-in_release] [old_project_spec] project_spec
         [-not_in_rel|-not_in_release] project_spec
```

オブジェクトとタスクへのタスクの関連付け

```
ccm task -rel|-relate task_spec
         {[-obj|-object file_spec[ file_spec...]] | [-fix task_spec]}
```

カレント（デフォルト）タスクの設定または解除

```
ccm task -def|-default [task_spec|None]
```

タスク情報の表示

```
ccm task -sh|-show
         { i|info|information [-v|-verbose]| obj| objs| objects |
          -fix| -related [-all] | fixed_by | [-v|-verbose]}
         [-f|-format "format_string"] [-ns|-no_sort] [-u] task_specs
ccm task -sh|-show
         {p|priority |
         plat|platform |
         r|resolver |
         rel|release |
         s|synopsis |
         sub|subsystem |
         time|time_estimate |
         date|date_estimate |
         description |
         status_log
         cr|change_request|change_requests|prob|problem|problems}
         task_specs
```

別の状態へのタスクの遷移

```
ccm task -st|-state task_assigned | completed | excluded
         [-r|-resolver resolver]|
         [-description "description"]
         [-descriptionedit]
         [-descriptionfile file_path]
         task_specs
```

### オブジェクト、タスクからのタスクの関係解除

```
ccm task -unrelate task_spec
         {[-obj|-object file_spec[ file_spec...]] |
         [-fix task_spec]}
```

## 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オブジェクトの関係を作成（関連付け）または解除（関連解除）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。

## オプションと引数

**-a|-associate**

指定されたタスクを 1 つまたは複数のタスク、オブジェクト、変更依頼と関連付けます。

既存タスクによってタスクを修正する場合に、このコマンドを使用します。新しいタスクによってタスクを修正する場合は、-fix を参照してください。

オブジェクトまたは他のタスクを関連付けようとしているタスクは、自分が割り当て、書き込み可能である必要があります。変更依頼を関連付ける場合は、変更依頼が書き込み可能であり、タスクとの関連付けが可能な状態にある必要があります。

コマンド実行が成功した場合は戻り値 `0` が返され、成功しなかった場合はゼロ以外の値が返されます。

`-as|-assign`

1 つまたは複数の指定タスクを担当者に割り当てます。

このオプションを使用するには、割り当て者または PT 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。

表示される出力メッセージ数を減らすには、このオプションを `-quiet` オプションと一緒に使用します。

コマンド実行が成功した場合は戻り値 `0` が返され、成功しなかった場合はゼロ以外の値が返されます。

`-c|-comment "`*`string`*`"`

チェックインするタスクと関連オブジェクトに、コメントとして *`string`* を追加します。

`-cr|-change_request|-problem`

操作対象の変更依頼 ID を指定します。この引数の構文については、変更依頼の指定を参照してください。変更依頼はカンマまたは空白文字で区切ることができます。

`-ci|-checkin`

カレント（デフォルト）または指定タスクをチェックインします。Rational Synergy では、「タスクのチェックイン」は「タスクの完了」と呼び方が変更されました。

`-y` オプションを使用した場合、チェックインしようとするタスクがどのオブジェクトとも関連付けられていなければ、確認メッセージは表示されません。

コマンド実行が成功した場合は戻り値 `0` が返され、成功しなかった場合はゼロ以外の値が返されます。

-complete

カレント（デフォルト）または指定タスクを完了します。-y オプションを使用した場合、完了タスクがどのオブジェクトとも関連付けられていなければ、確認メッセージは表示されません。

コマンド実行が成功した場合は戻り値 0 が返され、成功しなかった場合はゼロ以外の値が返されます。

-cp|-copy

タスクをコピーします。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。

コピーされたタスクの説明は元のタスクからコピーされません。タスクの説明を追加するには、-description または -descriptionfile を使用します。テキスト エディタを起動するには、-descriptionedit を使用します。

元のタスクに関連しているオブジェクトをコピーしたタスクに関連付けたくない場合は、-no_objects を使用します。コピーされたタスクの状態を *registered*（登録済み）に設定するには、-register を使用します。コピーしたタスクを完了するユーザーを指定するには、-resolver を使用します。

デフォルトでは、-register を指定しないと、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。

表示される出力メッセージの数を減らすには、-quiet を使用します。

-cr|-create

指定したプロパティを持つタスクを作成します。

既存のタスク説明にテキストを追加するには、-description または -descriptionfile を使用します。既存のタスク説明を表示するためにテキスト エディタを起動するには、-descriptionedit を使用します。表示される出力メッセージの数を減らすには、-quiet を使用します。

コマンド実行が成功した場合は戻り値 `0` が返され、成功しなかった場合はゼロ以外の値が返されます。

`-date|-date_estimate` *`date_estimate`*

タスクの完了日を予測します。

このオプションは、`-create`、`-copy`、`-fix`、`-modify` のいずれかのオプションと一緒にのみ使用できます。

`-show` オプションも「`date_estimate`」の値を持つことができます（ハイフンなし）。

`-def|-default`

カレント（デフォルト）タスクを指定したタスク番号または `None` に設定するか、またはカレント タスクを表示します。Rational Synergy では、「デフォルト タスク」は「カレント タスク」と呼び方が変更されました。

このオプションを `-create` オプションと一緒に使用すると、新しいタスクが作成され、ユーザーに割り当てられ、それから Rational Synergy Classic の現在のセッションのカレント タスクとして設定されます。他のユーザーにタスクを割り当てるために `-resolver` オプションを使用すると、エラー メッセージが表示され、タスクは作成されません。

コマンド実行が成功した場合は戻り値 `0` が返され、成功しなかった場合はゼロ以外の値が返されます。

> 注記：選択するカレント タスクは *assigned*（割り当て済み）状態で、かつユーザーがそのタスクの *resolver*（担当者）である必要があります。

カレント タスクが設定されていないのに、`-default` と入力すると、デフォルトが設定されていないことを示すメッセージが表示されます。カレント タスクが設定されていない場合、このコマンドの戻り値は 1 です。

`-description "`*`description`*`"`

タスクの説明を追加します。

`-descriptionfile` *`file_path`*

タスクの説明を記述します。

`-descriptionedit`

タスクの説明を表示するために、テキスト エディタを起動します。

`-d|-disassociate`

指定タスクまたは変更依頼と 1 つまたは複数のタスクまたはオブジェクトとの関連を解除します。既存のタスクと関係を解除するために、このコマンドを使用します。

関係を解除するタスクは書き込み可能である必要があります。

コマンド実行が成功した場合は戻り値 `0` が返され、成功しなかった場合はゼロ以外の値が返されます。

`-fix`

新しいタスクを作成して、作成したタスクと修正するタスクを関連付け、タスクとそのタスクが修正したタスクとの間の関係を解除します。

修正タスクを作成すると、タスク間に関係が設定されます。これにより、Rational Synergy では、プロジェクトが一方のタスクを使用しているが他方のタスクを使用していないことを検出できます （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。

修正タスクとの関係を設定してタスクを修正するために、このコマンドを使用します。

既存のタスクによってタスクを修正する場合は、`-relate` または `-a|-associate` オプションを参照してください（どちらのコマンドも機能は同じですが、`-relate` を使用してください)。もし修正タスクにバグがある場合、最初の修正タスクを修正するための新しい修正タスクを作成します。

また、`-unrelate` オプションを使用することにより、タスクの関係を解除できます。修正関係を間違って設定したときに、2 つのタスク間の修正関係を解除します。

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。

修正タスクの詳細説明は元のタスクからコピーされません。タスクの詳細説明を記述するには、`-description` または `-descriptionfile` を使用します。テキスト エディタを起動するには、`-descriptionedit` を使用します。

修正タスクの状態を *excluded*（除外）に設定するには、`-exclude` を使用します。修正タスクの状態を *registered*（登録済み）に設定するには、`-register` を使用します。修正タスクを完了するユーザーを指定するには、`-resolver` を使用します。

デフォルトでは、`-register` を指定しないと、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。

表示される出力メッセージの数を減らすには、`-quiet` を使用します。

値（概要、担当者、プラットフォーム、リリースなど）を変更するには、`-modify`、`-associate`、`-disassociate` のいずれかのコマンドを使用します。

`-fixes`

`-associate|-relate` と一緒に使用した場合は既存の 2 つのタスクの間の関係を設定し、`-disassociate|-unrelate` と一緒に使用した場合は既存の 2 つのタスクの間の関係を解除します。

`-f|-format "`*`format_string`*`"`

コマンドの出力フォーマットを指定します。デフォルトのフォーマットは、`-format` と一緒に使用するオプション、(`-query` または `-show`)、とこれらのオプションのキーワード引数によって異なります。デフォルトの出力フォーマットについては各オプションの説明を参照してください。

必須文字列には、以下のようにキーワードと文字テキストを使用します。

```
%displayname %owner
```

キーワードには、組み込まれたもの（%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。

-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* を指定しないと、差し引くタスクはありません。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。

-m|-modify

以下のサブオプションを任意の組み合わせで使用して、タスク プロパティを変更できます。

-s|-synopsis "*synopsis*"
-p|-priority *priority*
-r|-resolver *resolver*
-sub|-subsystem *subsystem*
-plat|-platform *platform*
-time|-time_estimate *time_estimate*
-date|-date_estimate *date_estimate*
-rel|-release *release*
-q|-quiet
*task_specs*

既存の詳細説明にテキストを追加するには、-description または -descriptionfile を使用します。既存の詳細説明を表示するためにテキスト エディタを起動するには、-descriptionedit を使用します。表示される出力メッセージ数を減らすには、このオプションを -quiet と一緒に使用します。

-modify オプションには、複数のサブオプションを使用できます。

コマンド実行が成功した場合は戻り値 0 が返され、成功しなかった場合はゼロ以外の値が返されます。

`-not_in_rel|-not_in_release` *`project_spec`*

リリース用に設定されているが、リリースに含まれなかったすべてのタスクを検索します。これは、*`project_spec`* とリリース値が合致する完了済みのタスクのリストを入手し、それから階層内のすべてのオブジェクトとその祖先に関するタスクを差し引いて決定されます。

`-ns|-no_sort`

コマンドの出力をソートしません。

`-obj|-object` *`file_spec`*`[ `*`file_spec`*`...]`

プロジェクトのメンバーであるファイルまたはディレクトリの名前を指定します。

複数の *`file_spec`* を指定する場合、使用する値同士の間に少なくとも空白を 1 つ入れる必要があります。

`-associate` または `-disassociate` オプションを使用する場合、このオプションへの引数として、選択セットを使用できます。

`-p|-priority` *`priority`*

作成するタスクの優先度を指定します。優先度には `high`、`medium`、low、any があります。

このオプションは、`-create` または `-modify` オプションと一緒にのみ使用できます。

`-plat|-platform` *`platform`*

タスクに関連する変更を適用するプラットフォームを指定します。プラットフォームの選択肢は *`CCM_HOME`*`¥etc¥om_hosts.cf` ファイル（Windows）または `$CCM_HOME/etc/om_hosts.cf` ファイル（UNIX）に定義されています。タスクが複数のプラットフォームに適用される場合は、タスクにプラットフォーム値を設定しないでください。

このオプションは、`-create` または `-modify` オプションと一緒にのみ使用できます。

`-problem`

`-cr|-change_request|-problem` を参照してください。

`-qu|-query` *query_spec*

*query_spec* を使用して実行したクエリによって見つかったタスクを表示します。`-format` オプションと一緒にこのコマンドを実行すると、出力に一覧表示されるタスクが選択セットに入れられます。

`-format` オプションを使用して、出力のフォーマットを制御できます。デフォルトフォーマットは次のとおりです。

```
Task %nnnn: %task_synopsis
```

自分以外のデータベースでクエリを行う場合に、`dbid` オプションを使用して出力のフォーマットを制御できます。

```
Task %dbid#nnnn: %task_synopsis
```

ここで、*dbid* はデータベース ID、`#` は DCM 区切り文字、*nnnn* はタスク番号です。

*query_spec* の構文は以下のとおりです。

```
[-cus|-custom query_expression]
[-db|-database_id database_id]
[-plat|-platform platform]
[-rel|-release release]
[-sub|-subsystem subsystem]
[-ts|-scope|-task_scope task_scope]
```

ここで、*task_scope* は以下のいずれかです。

```
user_defined
all_my_assigned
all_my_completed
all_my_tasks
all_completed
all_tasks
```

`-subsystem`、`-release`、`-platform`、`-database_id` を指定しないと、その値は `Any` であるとみなされます。

*query_expression* は query コマンド コマンドの *query_expression* と同じです。

`-q|-quiet`

表示される出力メッセージ数を減らします。

`-relate`

指定されたタスクと 1 つまたは複数のタスクまたはオブジェクトとの間に関係を設定します。

既存タスクによってタスクを修正する場合に、このコマンドを使用します。新しいタスクによってタスクを修正する場合は、`-fix` を参照してください。

オブジェクトを関連付ける相手のタスクは書き込み可能である必要があります。

既存タスクによってタスクを修正するには、`-fix` オプションを使用します。オブジェクトをタスクに関連付けるには、`-object` オプションを使用します。

コマンド実行が成功した場合は戻り値 `0` が返され、成功しなかった場合はゼロ以外の値が返されます。

`-rel|-release` *`release`*

タスクに関連する変更を適用するリリースを指定します。`ccm release` コマンドを使用して、リリースの選択肢を表示できます。

このオプションは、`-create` または `-modify` オプションと一緒にのみ使用できます。

`-r|-resolver` *`resolver`*

タスクを解決する担当のユーザーを指定します。データベース ユーザーの中の任意のユーザーを指定できます。

このオプションを使用するには、割り当て者または PT 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。

このオプションは、`-create` オプションまたは `-modify` オプションと一緒にのみ使用できます。また、タスクが書き込み可能である必要があります。タスクの作成時または修正時にこのオプションを使用すると、指定した担当者にタスクが自動的に割り当てられます。

`-s|-synopsis "`*`string`*`"`

タスクの説明であり、必須です。文字列は引用符で囲む必要があります。

このオプションは、`-create` または `-modify` オプションと一緒にのみ使用できます。

-sh|-show

指定したタスクのプロパティを表示します。このオプションを info、objs、prob キーワードまたはそのバリアントと一緒に使用すると、選択セットには出力に一覧表示されるフォルダ、オブジェクト、プロジェクト、タスクが含まれます。また、info キーワードにより、カレント タスクによって修正されたタスクおよびカレント タスクが修正するタスクが表示されます。

-f オプションを使用して、コマンドの出力フォーマットを変更します。-u によって出力の自動番号付けを抑止し、-ns によってソートを抑止します。

カレント タスクが修正するタスクを表示するには、-show に -fix オプションを付けて使用します。カレント タスクを修正するタスクを表示するには、-show に fixed_by オプションを付けて使用します。タスクを修正するかまたはタスクによって修正される完了済みタスクのリストを表示するには、-show に related オプションを付けて使用します。直接関係するタスクを修正するかまたは直接関係するタスクによって修正されるタスクも表示されます。すべての状態のタスクを表示するためには、-show related と一緒に -all オプションを使用します。

-format オプションを使用している場合、以下のいずれかのキーワードを使用できます。

```
i|info|information
obj|objs|objects
cr|change_request|change_requests|prob|problem|problems
```

指定されたタスクの影響を受けるすべての関連タスクとオブジェクトを表示するには、-verbose オプションと一緒に -show info を使用します。*task_spec* によって指定されたタスクの task_description 属性を表示するには、-description オプションと一緒に -show info を使用します。*task_spec* によって指定されたタスクの status_log 属性を表示するには、-status_log オプションと一緒に -show info を使用します。

task -show information のデフォルト出力フォーマットは以下のとおりです。

```
Task %displayname: %task_synopsis
```

ここで、

DCM が無効な場合、%displayname は %name です。
DCM が有効な場合は *<database_ID><DCM_delimiter>*%name です。
%task_synopsis はタスクの説明です。

これらの行の後に、いくつかのタスクのプロパティに関する追加情報が表示されます。

`task -show objects` のデフォルト出力フォーマットは以下のとおりです。

```
%objectname %status %owner
```

ここで、

`%objectname` はオブジェクトの *name-version:type:instance* です。
`%status` はオブジェクトの状態です。
`%owner` はオブジェクトの所有者です。

`-format` オプションを使用しない場合は、以下のいずれか 1 つのキーワードを使用してタスクのプロパティを表示できます。

```
cr|change_request|change_requests|prob|problem|problems
p|priority
plat|platform
r|resolver
rel|release
s|synopsis
sub|subsystem
time|time_estimate
```

コマンド実行が成功した場合は戻り値 `0` が返され、成功しなかった場合はゼロ以外の値が返されます。

### `-sub|-subsystem` *subsystem*

タスクが属するサブシステムを指定します（例、Any、GUI code、CLI code、または documentation）。サブシステムの指定に空白が含まれる場合は、引用符で囲んでください。

このオプションは、`-create` または `-modify` オプションと一緒にのみ使用できます。

### `-st|-state task_assigned | completed | excluded`

開発者がタスクを *completed*（完了）状態から *excluded*（除外）状態へ遷移できるようにします。また、開発者がタスクの担当者である場合は、*excluded*（除外）状態から *completed*（完了）状態へ遷移できるようにします。ビルド マネージャ、PT アドミニストレータ、または *ccm_admin* ロールを持つユーザーが、*completed*（完了）状態

から *excluded*（除外）状態へ、または *excluded*（除外）状態から *completed*（完了）状態へ遷移できるようにします。

`-resolver` *resolver* オプションを使用できるのは、`-state task_assigned` オプションを使用する場合のみです。

`-t|-to` *`resolver`*

1 つ以上のタスクを割り当てる担当者を指定します。

`-assign` オプションと一緒にこのオプションを使用する場合、タスクに書き込み可能である必要があります。

*`task_spec`*

操作対象のタスクの ID を指定します。この引数の構文については、タスクの指定 を参照してください。複数のタスク仕様は、カンマまたは空白文字で区切ります。

注記：*`task_spec`* が指定可能な場合は、*`task_spec`* を含むファイル名を指定できます。

`-time|-time_actual` *`task_duration`*

タスクを完了させるために必要な、実時間を指定します。

このオプションは、`-checkin` オプションと一緒にのみ使用できます。

`-time|-time_estimate` *`time_estimate`*

タスクを完了させるために必要な、予定時間を指定します。

このオプションは、`-create` または `-modify` オプションと一緒にのみ使用できます。

`-u`

このコマンド出力の自動番号付けを抑止します。

`-unrelate`

指定されたタスクと 1 つまたは複数のタスクまたはオブジェクトとの間の関係を解除します。

タスクまたはオブジェクトとの関係を解除する相手のタスクは、書き込み可能である必要があります。

既存のタスクとそれを修正するタスクとの間の関係解除するには、`-fix` オプションを使用します。

コマンド実行が成功した場合は戻り値 `0` が返され、成功しなかった場合はゼロ以外の値が返されます。

`-y`

タスクの完了時に関連するオブジェクトが何もなかった場合の、確認メッセージ抑止します。

関連トピック

- task コマンドの例

## **task** コマンドの例

以下の操作の例について説明します。

- タスクの関連付け
- タスクの割り当て
- タスクの完了（チェックイン）
- タスクのコピー
- タスクの作成
- タスクとオブジェクトの関連付けを解除
- タスクと変更依頼の関連付けを解除
- タスクの修正
- タスクの変更
- タスクのクエリ
- カレント（デフォルト）タスクの設定または解除
- タスク情報の表示
- タスクの遷移

### タスクの関連付け

- 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
  ```

### タスクの割り当て

- タスク 54、60-63、および 74 をユーザー *lseyer* に割り当てる。

  ```
  ccm task -as 54,60-63,74 to lseyer

  Assigned task 54
  Assigned task 60
  Assigned task 61
  Assigned task 62
  Assigned task 63
  Assigned task 74
  ```

### タスクの完了（チェックイン）

- 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 -checkin 40 -comment "The problem is fixed."

Object version is already associated with task.
Archiving CALC.H-4:source
Checked in 'CALC.H-4' to 'integrate'
Object version is already in the 'integrate' state: 'CALC.C-6'
Object version is already associated with task.
Archiving MSGS.H-5:source
Checked in 'MSGS.H-5' to 'integrate'
Summary:
1  skipped
2  succeeded
0  failed
Task '40' checked in.
```

- カレント タスクを完了（デフォルト タスクをチェックイン）する

```
ccm task -ci default
```

### タスクのコピー

- タスク 40 をコピーする。ただし、新しい概要、リリース、担当者、説明を付与し、このタスクに関連付けられたオブジェクトに上書きコピーはしないよう指定する。

```
ccm task -copy 40 -synopsis "Fix GUI color problem" -release 2.0 -
resolver bob -no_objects -description "check RGB module"

Task hawaii#50 created.
```

### タスクの作成

- 新しいタスクを作成して、概要（名前）を `Entanglement methods` とする。

```
ccm task -create -synopsis "Entanglement methods"

Task 44 created.
```

### タスクとオブジェクトの関連付けを解除

- タスク 35 とオブジェクト `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
```

### タスクと変更依頼の関連付けを解除

- 変更依頼 6569 からタスク 10668 の関連付けを解除する。

```
ccm task -d 10668 -change_request 6569
```

### タスクの修正

- 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) と修正されるタスク (12) との間の関係を解除する。

```
ccm task -unrelate 25 -fixes 12
```

### タスクの変更

- タスク 68 のリリースを 4.1 に変更する。

```
ccm task -modify -release 4.1 68

Changed release of task 68
```

### タスクのクエリ

- 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
```

### カレント（デフォルト）タスクの設定または解除

- カレント（デフォルト）タスクを表示する。

```
ccm task -default

The current task is not set.
```

- カレント（デフォルト）タスクを設定する。

```
ccm task -default 26

The current task is set to:
26:  Close box no longer active
```

- カレント（デフォルト）タスクを解除する。

```
ccm task -default None
```

```
   The current task has been cleared.
```

### タスク情報の表示

- タスク 31 の情報を表示する。

```
   ccm task -show info 31

   Task:    31
    Synopsis:   Wrong window receives message
    State:      completed

    Resolver:        john
    Release:         3.1
    Priority:        high
    Subsystem:       <Uninitialized>
    Platform:        <Uninitialized>
    Database Id:     M

   Task Description:
   The wrong window receives the event message when users abort an
operation. Currently, the Show window receives the abort message. The Home
window should receive this message.

Status Log:
Mon Aug 16 15:57:09 1999: Status set to 'registered' by mary in role assigner
Mon Aug 16 15:57:14 1999: Status set to 'task_assigned' by mary in role
assigner
Tue Aug 17 11:16:55 1999: Status set to 'completed' by bill in role developer
```

- 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 68 に関連する変更要求を表示する。

```
   ccm task -show change_request 68

   1) Change request 5
   2) Change request 6
```

- タスク 4 と 5 に関連付けられたオブジェクトを表示する。

```
ccm task -show objects 4,5

1) MAIN.C-2:csrc:1  integrate  john
2) MAIN.H-4:incl:1  integrate  john
3) UTIL.C-7:csrc:1  integrate  john
4) MSGS.H-9:incl:1  integrate  john
```

### タスクの遷移

- 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

Changed state of task KJG461#94 to excluded
```

- 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

Changed state of task KJG461#94 to completed
```

## type コマンド

type コマンドは cat コマンド の別名です。

# **typedef** コマンド

## 表記

```
ccm typedef type_name -d|-description type_description
            -s|-super_type super_type_name
           [-f|-file_extension file_extensions]
            [-fw file_extensions] [-fu file_extensions]
            [-mm match_regular_expressions]
            [-mmw match_regular_expressions]
            [-mmu match_regular_expressions]
            [-mi TRUE|FALSE]
            [-miw TRUE|FALSE] [-miu TRUE|FALSE]

ccm typedef -i|-import type_name [-image] -dir from_path [-force]
ccm typedef -e|-export type_name -dir to_path [-force]
            [-ef|-export_format export_format]
```

## 説明と用途

`typedef` コマンドにより、現在のデータベースに新しいタイプを追加、あるいは既存のタイプを更新します。タイプにマイグレーション ルールを指定すると、タイプ定義の一部として格納され、自動生成されるルールに使用されます。新しいタイプを作成すると、Rational Synergy はそれに対応するマイグレーション ルールを自動的に作成して保存します。

`-i` オプションを使用すると、このコマンドは指定されたタイプをファイル システムからデータベースへインポートします。ただし、前に `-e` オプションを使用して Rational Synergy のデータベースからエクスポートされたタイプでなければなりません。

`-e` オプションは指定されたタイプをデータベースからファイル システムへエクスポートします。タイプをエクスポートする先のディレクトリは既存のもので、*ccm_root* によって書き込み可能である必要があります。

`typedef` コマンドを使用するためには、タイプ開発者である必要があります。

## オプションと引数

`-d|-description` *`type_description`*

追加するタイプを説明します。説明に空白が含まれる場合には、全体を引用符で囲みます。

`-dir` *`from_path`*`|`*`to_path`*

タイプのインポート元またはタイプのエクスポート先のディレクトリへのフル パスを指定します。パスに空白が含まれる場合には、全体を引用符で囲みます。

-e|-export *type_name*

指定されたタイプをデータベースからファイル システムへエクスポートします。

-force オプションを指定せず、パスが既に存在する場合、パスは更新されません。-force オプションを指定した場合、エクスポートされたタイプの内容は上書きされます。

-ef|-export_format *export_format*

export_format が XML または CCM45SP2 のどちらかであることを指定します。デフォルトは XML てす。フォーマットが XML である場合、タイプ定義は type.xml という名前の 1 つの *XML* ファイルによって表されます。このファイルには、cvtype オブジェクトおよび任意の attype オブジェクトの定義が格納されています。

-f|-file_extension *file_extensions*

タイプのデフォルトのファイル拡張子を指定します。ファイル拡張子なしの場合は、-f "" を使用します。

-force

既存のタイプを新しいタイプで強制的に上書きし、新しい属性があれば追加します。このオプションは、-import オプションまたは -export オプションと一緒にのみ使用できます。

> 注記：新しいタイプに存在しないので既存タイプの属性を削除したい場合には、代りに -image オプションを使用します。

-fu file_extensions

UNIX クライアントで使用する、タイプのデフォルトのファイル拡張子を指定します。これは、1 つまたは複数のファイル接尾辞を少なくとも 1 つの空白で区切ったリストです。各接尾辞の先頭にはピリオド（'.'）を付けます。

-fw file_extensions

Windows クライアントで使用する、タイプのデフォルトのファイル拡張子を指定します。これは、1 つまたは複数のファイル接尾辞を少なくとも 1 つの空白で区切ったリストです。各接尾辞の先頭にはピリオド（'.'）を付けます。

-g

適切なダイアログを呼び出します。

`-i|-import` *`type_name`*

指定されたタイプをファイル システムからデータベースへインポートします。

> 注記：既存タイプにインポートするためには、タイプがエンジンから見える必要があります。既存タイプを上書きするためには、一緒に `-force` オプションまたは `-image` オプションを使用する必要があります。
>
> 注意！ `-force` オプションまたは `-image` オプションを使用してインポートした場合、データベース内の指定タイプのすべてのオブジェクトに変更が適用されます。

`-image`

指定されたタイプのすべてのオブジェクトに関して、すべての属性をインポート ディレクトリ内の属性で置き換えます。新しい属性は追加され、変更されたプロパティは置き換えられ、削除された属性はデータベースから除去されます。このオプションは、`-import` オプションと一緒にのみ使用できます。

`-mi TRUE|FALSE`

現在のクライアントに関して、指定されたタイプのオブジェクトをマイグレーション時に無視すべきかどうかを指定します。値 TRUE は無視、値 FALSE は無視しないこと（デフォルト）を意味します。

`-miu TRUE|FALSE`

UNIX クライアントに関して、指定されたタイプのオブジェクトをマイグレーション時に無視すべきかどうかを指定します。値 TRUE は無視、値 FALSE は無視しないこと（デフォルト）を意味します。

`-miw TRUE|FALSE`

Windows クライアントに関して、指定されたタイプのオブジェクトをマイグレーション時に無視すべきかどうかを指定します。値 TRUE は無視、値 FALSE は無視しないこと（デフォルト）を意味します。

`-mm match_regular_expressions`

現在のクライアントに関して、マイグレーション ルールでファイルを照合するために使用する正規表現を指定します。値は、1 つまたは複数の空白で区切られた、1 つまたは複数の正規表現のリストです。正規表現の詳細については、マイグレーション ルールを参照してください。

`-mmu match_regular_expressions`

UNIX クライアントに関して、マイグレーション ルールでファイルを照合するために使用する正規表現を指定します。値は、1 つまたは複数の空白で区切られた、1 つまたは複数の正規表現のリストです。正規表現の詳細については、マイグレーション ルールを参照してください。

`-mmw match_regular_expressions`

Windows クライアントに関して、マイグレーション ルールでファイルを照合するために使用する正規表現を指定します。値は、1 つまたは複数の空白で区切られた、1 つまたは複数の正規表現のリストです。正規表現の詳細については、マイグレーション ルールを参照してください。

`-s|-super_type` *`super_type_name`*

追加するタイプのスーパー タイプを指定します。スーパー タイプとは、タイプが特性を継承する元のタイプです。データベース中に定義されているタイプを一覧表示するには、`show` コマンドを使用します。

## 例

- HTML ファイル用のオブジェクト タイプを作成する。ただし、`ascii` スーパータイプから特性を継承し、Rational クライアントのカレント タイプには `.html` 拡張子を使用する。

  ```
  ccm typedef html -d "Hypertext Markup Language" -s ascii -f ".html"
  ```

- JPEG ファイルのオブジェクト タイプを作成する。ただし、バイナリ タイプから特性を継承し、Windows と UNIX の両方のクライアントの接尾辞として .jpeg、.jpg、および .jpe を使用する。

  ```
  ccm typedef jpeg -d "JPEG Image" -s binary -fw ".jpeg .jpg .jpe" -fu
  ".jpeg .jpg .jpe"
  ```

- ファイルを一覧表示するオブジェクト タイプを作成する。ただし、1) ascii タイプから特性を継承し、2) Windows の接尾辞を「.lst」とし、3) UNIX の接尾辞を「.lis」とし、4) UNIX クライアントでのマイグレーション時にデフォルトで無視する。

  ```
  ccm typedef list -d "Listing file" -s ascii -fw ".lstî -fu ".lis" -miu
  TRUE
  ```

- MS Word ドキュメントのオブジェクト タイプを作成する。ただし、1) バイナリ タイプから特性を継承し、2) Windows クライアントで .doc と .dot を認識するマイグレーション照合ルールを適用する。

  ```
  ccm typedef msword -d "MS Word" -s binary -mmw ".*[Dd][Oo][CcTt]"
  ```

- Windows :
  pascal タイプを現在のデータベースから c:¥ccm¥exported types ディレクトリにエクスポートする。

  ```
  ccm typedef -export pascal -dir "c:¥ccm¥exported types"
  ```

  UNIX :
  pascal タイプを現在のデータベースから /mnt/ccm/exported types ディレクトリにエクスポートする。

  ```
  ccm typedef -export pascal -dir "/mnt/ccm/exported types"
  ```

- Windows :
  fmdoc タイプを c:¥ccm¥types_to_import ディレクトリからカレント データベースにインポートする

  ```
  ccm typedef -import fmdoc -dir c:¥ccm¥types_to_import
  ```

  UNIX :
  fmdoc タイプを c/mnt/ccm/types_to_import ディレクトリからカレント データベースにインポートする

  ```
  ccm typedef -import fmdoc -dir /mnt/ccm/types_to_import
  ```

## 警告

Rational Synergy に組み込まれているタイプは、typedef コマンドを使用して変更できません。

# **unalias** コマンド

## 表記

```
ccm unalias alias_name
```

## 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` コマンドを使用して、現在のセッションの別名のみを削除します。

## オプションと引数

*`alias_name`*

削除する別名の名前を指定します。

## 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

# **unrelate** コマンド

## 表記

```
ccm unrelate -n|-name rel_name -f|-from file_spec1
             -t|-to file_spec2
```

## 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。これらの関係については、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ためには、(以下に定義する) 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。

## オプションと引数

-f|-from *file_spec1*

*file_spec1* を起点とする関係を削除します。

-n|-name *rel_name*

*rel_name* によって指定される関係を削除します。

-t|-to *file_spec2*

*file_spec2* を終点とする関係を削除します。

## 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

- relate コマンド

## undo_reconfigure コマンド

`unreconf|undo_reconfigure` コマンドは undo_update コマンド の別名です。

# **undo_update** コマンド

## 表記

```
ccm unupd|undo_update|unreconf|undo_reconfigure
                        [-v|-verbose] [-r|recurse] file_spec
ccm unupd|undo_update|unreconf|undo_reconfigure
                        [-v|-verbose] [-r|-recurse]
                        -p|-project project_spec
ccm unupd|undo_update|unreconf|undo_reconfigure
                        [-v|-verbose] [-r|-recurse]
                        -pg|-project_grouping project_grouping_spec
```

## 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（リコンフィギュア）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。

更新の取り消し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、更新の取り消し処理を続けたい場合に有用です。更新の取り消し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。

`undo_update` コマンドを使用した場合、最後の更新のみが取り消されます。つまり、更新の取り消しを 2 回以上実施した場合、最後の取り消しのみが取り消し可能です。

## オプションと引数

*`file_spec`*

更新を元に戻すディレクトリを指定します。

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

更新を元に戻すプロジェクトを指定します。

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

プロジェクト グルーピング内のすべてのプロジェクトの更新を元に戻すよう指定します。プロジェクト グルーピングのベースラインとタスクには変更されません。

`-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対象にして、`toolkit-mary` という名前のプロジェクトに対して行った更新を元に戻す。

  ```
  ccm undo_update -recurse -project toolkit-mary
  ```

関連トピック

- update コマンド

# **unset** コマンド

## 表記

`ccm unset` *`option`*

## 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。

## オプションと引数

*`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
```

## 警告

Rational Synergy には、暗黙的に設定され、`unset` コマンドでは設定を解除できない変数があります。

関連トピック

- set コマンド

# **unuse** コマンド

## 表記

```
ccm unuse [-t|-task task_number]
          [-d|-delete] [-r|-replace]
          file_spec [file_spec...]
ccm unuse [-t|-task task_number]
          [-d|-delete] [-r|-replace]
          -p|-project project_spec [project_spec...]
ccm unuse -delete -force file_spec [file_spec...]
```

## 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。Rational Synergy では、使用解除は切り取り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。初期化ファイル内の `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。

以下の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
```

## オプションと引数

`-d|-delete`

ディレクトリ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。

プロジェクトを切り取って削除することはできません。プロジェクトに対してこの操作を行った場合、プロジェクトは切り取られますが、削除はされません。プロジェクトを削除するためには、`delete` コマンドを使用します。

*`file_spec`*

使用解除するオブジェクトを指定します。

`-force`

`-force` オプションは確認メッセージを抑止し、削除操作を強制的に実行します。

`-g`

適切なダイアログを呼び出します。

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

切り取るプロジェクトを指定します。

サブプロジェクトを削除する場合、そのサブプロジェクトを含むディレクトリに移動する必要があります。`-p` オプションを含める必要はありません。

`-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_number`*

`unuse -delete` コマンドにより読み出し専用ディレクトリからオブジェクトを削除した場合に、新たにチェックアウトされたディレクトリにタスク番号を関連付けます。

カレント（デフォルト）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。

## 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
  ```

- 最上位プロジェクトの `ico_jan4-1` から、サブプロジェクトの `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
  ```

- プロジェクトからオブジェクトを削除して置き換える。

  `ccm unuse -delete -replace` *`object_version`*

  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。

関連トピック

- use コマンド

# **update** コマンド

## 表記

```
ccm u|update|update_members|reconf|reconfigure
                        [-v|-verbose] [-r|-recurse]
                        [-rs|-replace_subprojects | -ks|-keep_subprojects]
                        file_spec
ccm u|update|update_members|reconf|reconfigure
                        [-v|-verbose] [-r|-recurse]
                        [-rs|-replace_subprojects | -ks|-keep_subprojects]
                        -p|-project project_spec
ccm u|update|update_members|reconf|reconfigure
                        [-v|-verbose] [-r|-recurse]
                        [-rs|-replace_subprojects | -ks|-keep_subprojects]
                        [-pg|-project_grouping]
                        project_grouping_spec
```

## 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、リコンフィギュアおよびメンバーの更新は、「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すると、プロジェクト グルーピング内の一連のプロジェクトを同時には更新されないマルチフェーズ ビルドを、より容易に実行できるようになります。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、この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この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が可能になります。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または目的を変更する。
- `ccm update_properties` コマンドを使用して、プロセス ルールベースの更新と手動更新とを切り替える。
- `ccm update_properties` コマンドを使用して、オブジェクト ステータス　ベースの更新とタスクベースの更新とを切り替える。
- `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。

## オプションと引数

*`file_spec`*

更新するディレクトリを指定します。

`-ks|-keep_subprojects`

サブプロジェクトを維持するよう指定します。サブプロジェクトを置き換えるように replace_subproj が変更されている場合を除き、これが、現在のセッションのデフォルトです。`-ks` または `-rs` オプションを指定せずに更新コマンドを使用すると、既存のデフォルトが使用されます。

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

更新するプロジェクトを指定します。

`-pg|-project_grouping`

プロジェクト グルーピング全体を更新するよう指定します。プロジェクト グルーピングの設定に応じて、更新プロパティが最新の状態に更新されます。プロジェクト グルーピングの詳細については、project_grouping コマンドを参照してください。

*`project_gouping_spec`*

更新するプロジェクト グルーピングを指定します。

これは、4 部名称として、または `All CM/6.4 Integrations Testing Projects` や `Linda's CM/6.4 Collaborative Development Projects` のようなグループのデフォルトの表示名として、入力できます。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。

`-r|-recurse`

サブプロジェクトも更新するよう指定します。

`-rs|-replace_subprojects`

サブプロジェクトを新しいサブプロジェクトで置き換えてよいことを指定します。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、それらの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。`-ks` または `-rs` オプションを指定せずに更新コマンドを使用すると、既存のデフォルトが使用されます。

`-v|-verbose`

詳細な更新メッセージを表示するよう指定します。

## 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

## **update_properties** コマンド

### 表記

#### 更新プロパティの比較

```
ccm up|update_prop|update_properties|rp|reconf_prop|reconfigure_properties -
comp|-compare
       project_spec1 project_spec2
```

#### タスク／フォルダの追加

```
ccm up|update_prop|update_properties|rp|reconf_prop|reconfigure_properties -
a|-ad|-add
       [-t|-task|-tasks task_specs] [-y] [-related]
       [-fol|-folder|-folders folder_specs]
       [-r|-recurse] [-q|-quiet] project_specs
```

#### タスク／フォルダの削除

```
ccm up|update_prop|update_properties|rp|reconf_prop|reconfigure_properties -
rem|-remove
       [-t|-task|-tasks task_specs] [-related]
       [-fol|-folder|-folders folder_specs]
       [-r|-recurse] [-q|-quiet] project_specs
```

#### ベースライン、タスク、フォルダ、オブジェクトの表示

```
ccm up|update_prop|update_properties|rp|reconf_prop|reconfigure_properties -
sh|-show
       {at|all_t|all_tasks|
       b|bl| baseline_project
       fol|folders|
       obj|objects|
       t|tasks|
       tf|t_and_f|tasks_and_folders}
       [-auto|-automatic|-no_auto|-no_automatic]
       [-f|-format "format_string"]
       [-ns|-no_sort] [-u] [-r|-recurse] [-v|-verbose]
       project_spec
```

更新プロパティの表示

```
ccm up|update_prop|update_properties|rp|reconf_prop|reconfigure_properties -
sh|-show
       i|info|information
       [-v|-verbose] [-r|-recurse] project_specs
```

更新メソッドの設定

```
ccm up|update_prop|update_properties|rp|reconf_prop|reconfigure_properties -
ru|-reconf_using
       {os|obj_stat|object_status | t|tasks | template | manual}
       [-r|-recurse] [-q|-quiet] project_specs
```

特定プロジェクトのベースラインの設定

```
ccm up|update_prop|update_properties|rp|reconf_prop|reconfigure_properties
       -mb|-modify_baseline_project {baseline_spec|base_project_version}
       [-r|-recurse] -q|-quiet project_specs
```

更新プロパティの更新

```
ccm up|update_prop|update_properties|rp|reconf_prop|reconfigure_properties -
refresh [-recurse]
       project_specs
```

更新プロパティのサブプロジェクトへの保存

```
ccm up|update_prop|update_properties|rp|reconf_prop|reconfigure_properties -
ats|-apply_to_subprojs
       [-no_baseline] [-no_tasks_and_folders] project_specs
```

特定プロジェクトの有効なベースライン プロジェクトの表示

```
ccm up|update_prop|update_properties|rp|reconf_prop|reconfigure_properties -
vb|-valid_baseline_projects
       [-f|-format "format_string"] [-ns|-no_sort] [-u] project_spec
```

## 説明と用途

update_properties コマンドにより、1 つまたは複数のプロジェクトに関する更新プロパティを表示および設定し、2 つのプロジェクトの更新プロパティを比較できるようにします。

プロジェクトの更新プロパティは、以下の要素で構成されます。

プロジェクトの更新プロパティ

- プロセス ルールを使用して更新するようにプロジェクトが設定されている場合、プロジェクトのプロジェクト グルーピングのベースラインとタスク

- プロセス ルールを使用するようにプロジェクトが設定されていない場合、プロジェクトのベースラインとタスク

プロジェクトを指定しない場合、コマンドを実行するワークエリアに関連付けられたプロジェクトにコマンドが適用されます。

update_properties コマンドを使用して、以下の操作を行います。

- 更新プロパティを比較する。
- プロジェクトにタスク／フォルダを追加する。
- プロジェクトからタスク／フォルダを削除する。
- 1 つまたは複数のプロジェクトのベースライン、タスク、フォルダ、オブジェクトを表示する。
- プロジェクトの更新プロパティを表示する。
- プロジェクトの更新メソッドを設定する。
- 特定プロジェクトのベースラインを設定する。
- サブプロジェクトに更新プロパティを適用する。
- ベースライン、フォルダ、タスクを更新する。
- 特定プロジェクトの有効なベースラインを表示する。

## オプションと引数

-a|-add

指定されたタスクまたはフォルダを *project_spec* の更新プロパティに追加します。自分が書き込み可能なプロジェクトにのみ、このオプションを使用できます。

*excluded*（除外）状態にあるタスクをフォルダに追加すると、タスクの状態を示す確認メッセージが表示されます。このオプションをオフにするには、-y オプションを使用します。

表示される出力メッセージの数を減らすには、-quiet を使用します。指定したタスクまたはフォルダを再帰的に更新プロパティに追加するには、-r を使用します。

指定したフォルダまたはプロジェクトにタスクを追加するには、-add、-tasks と一緒に -related を使用します。*excluded*（除外）状態にあるタスクをフォルダに追加すると、タスクの状態を示す確認メッセージが表示されます。また、修正したタスクを追加せずに完了タスクをフォルダに追加すると、関連するタスクを追加するかを尋ねるメッセージが表示されます。可能な限り関連タスクを追加してください。

コマンド実行が成功した場合は戻り値 `0` が返され、成功しなかった場合はゼロ以外の値が返されます。

`-ats|-apply_to_subprojs`

更新プロパティの適用範囲を、指定したプロジェクト階層の全プロジェクトに設定します。単独では、この引数は指定したプロジェクトの更新プロパティをそのサブプロジェクトに保存するために使用できます。

`-auto|-automatic`

自動タスクを表示するよう指定します。

このオプションと一緒に、以下のいずれか 1 つを使用する必要があります。

- `ccm up -show tasks`
- `ccm up -show all_tasks`
- `ccm up -show tasks_and_folders`

このオプションは、`ccm.ini` ファイル (Windows) または `.ccm.ini` ファイル (UNIX) に設定されている値を上書きします。このオプションのデフォルトは `FALSE` です。つまり、自動タスクはデフォルトでは表示されません。

`-comp|-compare` *`project_spec1 project_spec2`*

2 つのプロジェクトの更新プロパティを比較します。

`-f|-format "`*`format_string`*`"`

コマンドの出力フォーマットを指定します。デフォルトのフォーマットはコマンドに使用する他のオプションによって異なります。

このフォーマットには、テキストとキーワードの組み合わせを含むことができます。キーワードは、表示されるときに各オブジェクトについての特定の情報に置き換わります。たとえば、ユーザー *sue* が所有するオブジェクトが表示される場合は、キーワード `%owner` は `sue` に置き換わります。

任意の既存の属性名をキーワードとして使用できます。また、`%displayname`、`%task_number` など、多数のキーワードがあらかじめ定義されて組み込まれています。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。

`-fol|-folder|-folders` *`folder_spec`*

追加、一覧表示、削除するフォルダの ID を指定します。この引数の構文については、フォルダの指定 を参照してください。

> 注記：*`folder_spec`* が指定可能な場合は、*`folder_spec`* を含むファイル名を指定できます。

`-mb|-modify_baseline` *`base_project_spec|base_project_version`*

*`project_spec`* で指定したプロジェクトについて、ベースラインを *`base_project_spec`* プロジェクトまたは *`base_project_version`* プロジェクト バージョンに設定します。

表示される出力メッセージの数を減らすには、このオプションを `-quiet` と一緒に使用します。

自分が書き込み可能なプロジェクトにのみ、このオプションを使用できます。

コマンド実行が成功した場合は戻り値 `0` が返され、成功しなかった場合はゼロ以外の値が返されます。

`-no_auto|-no_automatic`

表示しない自動タスクを指定します。

このオプションと一緒に以下のいずれか 1 つを使用する必要があります。

- `ccm up -show tasks`
- `ccm up -show all_tasks`
- `ccm up -show tasks_and_folders`

このオプションは、`ccm.ini` ファイル (Windows) または `.ccm.ini` ファイル (UNIX) に設定されている値を上書きします。このオプションのデフォルトは FALSE です。つまり、自動タスクはデフォルトでは表示されません。

`-no_baseline`

プロジェクト階層のベースライン プロジェクトがサブプロジェクトに適用されないよう指定します。このオプションは `-apply_to_subprojs` オプションと一緒に使用します。

`-no_tasks_and_folders`

プロジェクト階層のタスクとフォルダがサブプロジェクトに適用されないよう指定します。このオプションは `-apply_to_subprojs` オプションと一緒に使用します。

`-ns|-no_sort`

コマンドの出力をソートしません。

`-quiet`

出力メッセージ数を減らします。このオプションはスクリプト作成時に便利です。

*`project_specs`*

複数の *`project_spec`* です。

`-r|-recurse`

指定したプロジェクトだけでなく、プロジェクト階層内のサブプロジェクトにも現在のコマンドを適用します。

`-refresh`

プロセス ルールとの整合性を保つため、ベースライン、フォルダ、タスクを更新します。フォルダを更新するとクエリも実行されます。

`-recurse` オプションを使用した場合のみ、サブプロジェクトにも更新が適用されます。

このオプションは、ユーザーが持つロールによる更新プロパティの修正を許可するプロセスルールによって更新されたプロジェクトに対してのみ有効です。

`-related`

`-related` は、`-tasks` と `-add` または `-remove` と一緒に使用します。これにより、以下のいずれかが可能になります。

- 指定した修正タスクのすべての関連タスクを、指定したフォルダまたはプロジェクトに追加する。

または

- 指定した修正タスクのすべての関連タスクを、指定したフォルダまたはプロジェクトから削除する。

`-rem|-remove`

指定したタスクまたはフォルダを、指定したプロジェクトの更新プロパティから削除する。自分が書き込み可能なプロジェクトにのみ、このオプションを使用できます。

表示される出力メッセージの数を減らすには、このオプションを `-quiet` と一緒に使用します。指定したタスクを再帰的に更新プロパティから削除するには、`-recurse` を使用します。

コマンド実行が成功した場合は戻り値 `0` が返され、成功しなかった場合はゼロ以外の値が返されます。

`-ru|-reconf_using`

指定したプロジェクトの更新に使用するメソッドを指定します。このオプションは、タスクベース（`t|tasks|template|manual`）またはオブジェクト ステータス ベース（`os|obj_stat|object_status`）です。

表示される出力メッセージの数を減らすには、このオプションを `-q` と一緒に使用します。プロジェクトの更新メソッドを再帰的に変更するには、`-r` を使用します。

`-sh|-show`

*`project_spec`* プロジェクトの更新プロパティ に含るオブジェクトの指定タイプを表示します。選択セットには、出力に一覧表示されるフォルダ、オブジェクト、プロジェクト、タスクが入れられます。

`-show` オプションと一緒に、以下のいずれか 1 つの引数を使用する必要があります。

`at|all_t|all_tasks -auto|-automatic | -no_auto|-no_automatic`
直接的にまたは間接的にプロジェクトの更新プロパティ内にある（間接的とは、プロジェクトの更新プロパティ内のフォルダにタスクが含まれていることを意味します）、すべてのタスクを表示します。

`b|bl|baseline_project`
プロジェクトのベースラインを表示します。

`fol|folders`
プロジェクトの更新プロパティ内にあるフォルダを表示します。

`obj|objects`
直接的にまたは間接的にプロジェクトの更新プロパティ内にあるタスクのオブジェクトを表示します。

`t|tasks -auto|-automatic | -no_auto|-no_automatic`
直接的にプロジェクトの更新プロパティ内にあるタスクを表示します。

tf|t_and_f|tasks_and_folders -auto|-automatic | -no_auto|-no_automatic
直接的にプロジェクトの更新プロパティ内にあるタスクとフォルダを表示します。

-f オプションを使用して、コマンドの出力フォーマットを変更します。-u によって出力の自動番号付けを抑止し、-ns によってソートを抑止します。指定された更新プロパティを再帰的に表示するには、-r を使用します。

プロジェクトの更新プロパティを構成するベースライン プロジェクト、タスク、フォルダに関する情報を表示するには、info サブオプションを使用します。更新プロパティを再帰的に表示するには、info サブオプションと一緒に、-r を使用します。詳細なメッセージを表示するには、-v info サブオプションを使用します。

ccm up -show tasks と ccm up -show all_tasks のデフォルト出力フォーマットは以下のとおりです。

```
Task %displayname: %task_synopsis
```

ここで、

DCM が無効な場合、%displayname は %name です。
DCM が有効な場合は <*database_ID*><*DCM_delimiter*>%name です。
%task_synopsis はタスクの説明です。

ccm up -show tasks、ccm up -show all_tasks、または ccm up -show task_and_folder を実行しており、かつ ccm.ini ファイル（Windows）または .ccm.ini ファイル（UNIX）で show_auto_tasks が TRUE に設定されている場合、自動タスクが出力に表示されます。

up -show baseline_project のデフォルト出力フォーマットは以下のとおりです。

```
%displayname
```

up -show folder のデフォルト出力フォーマットは以下のとおりです。

```
Folder %displayname: %description
```

ここで、

DCM が無効な場合、%displayname は %name です。
DCM が有効な場合は <*database_ID*><*DCM_delimiter*>%name です。
%description はフォルダ名です。

up -show objects のデフォルト出力フォーマットは以下のとおりです。

```
%objectname %status %owner %task
```

ここで、

`%objectname` はオブジェクトの *name-version:type:instance* です。
`%status` はオブジェクトの状態です。
`%owner` はオブジェクトの所有者です。
`%task` はオブジェクトに関連付けられたタスクです。

`up -show tasks_and_folders` のデフォルト出力フォーマットは以下のとおりです。

```
%displayname %description %task_synopsis
```

`-t|-task|-tasks` *task_specs*

追加または削除するタスクの ID を指定します。この引数の構文については、タスクの指定 を参照してください。

`-u`

このコマンド出力の自動番号付けを抑止します（"un-numbered"）。

`-vb|-valid_baselines`

*project_spec* で指定したプロジェクトのベースラインとして使用可能な、候補のプロジェクト バージョンを表示します。

`-f` オプションを使用して、コマンドの出力フォーマットを変更します。`-u` によって出力の自動番号付けを抑止し、`-ns` によってソートを抑止します。デフォルト出力フォーマットは `%displayname` です。

`-v|-verbose`

追加の更新プロパティ メッセージを出力することを指定します。

`-y`

確認メッセージを抑止するには、`-y` を使用します。このオプションはスクリプト作成時に便利です。

関連トピック

- update_properties コマンドの例

## **update_properties** コマンドの例

以下の操作の例について説明します。

- タスク／フォルダの追加
- 更新プロパティの比較
- タスク／フォルダの削除
- ベースライン、タスク、フォルダの情報の表示
- 更新メソッドの設定
- 有効なベースラインの表示／設定
- ベースライン、タスク、フォルダ、オブジェクトの表示

### タスク／フォルダの追加

- タスク 3、12-14、40-42 を `utility-mary` プロジェクトに追加する。

```
ccm update_prop -add -tasks 3,12-14,40-42 utility-mary

Adding tasks to utility-mary...
   Added task 3 to the update properties of project utility-mary
   Added task 12 to the update properties of project utility-mary
   Added task 13 to the update properties of project utility-mary
   Added task 14 to the update properties of project utility-mary
   Added task 40 to the update properties of project utility-mary
   Added task 41 to the update properties of project utility-mary
   Added task 42 to the update properties of project utility-mary

Added the following task(s) to project utility-mary
   Task 3
   Task 12
   Task 13
   Task 14
   Task 40
   Task 41
   Task 42
Added 7 tasks to utility-mary.
```

- DCM 用に初期化されたデータベースで、`frankfurt` データベースからインポートした、タスク 5 と 17 を `utility-mary` プロジェクトに追加する。

```
ccm update_prop -add -tasks frankfurt#5,frankfurt#17 utility-mary

Updating update properties of utility-mary...
   Added task 5
   Added task 17
Added 2 tasks.
```

- フォルダ 5-7 と 14 を utility-mary プロジェクトに追加する。

```
ccm update_prop -add -folders 5-7,114 utility-mary

Adding folders to utility-mary...
Added folder 5 to the update properties of project utility-mary
Added folder 6 to the update properties of project utility-mary
Added folder 7 to the update properties of project utility-mary
Added folder 114 to the update properties of project utility-mary

Added the following folder(s) to project utility-mary
      Folder 5
      Folder 6
      Folder 7
      Folder 114

Added 4 folders to utility-mary.
```

## 更新プロパティの比較

- utility-mary プロジェクトと toolkit-2.0_gui プロジェクトの更新プロパティを比較する。

```
ccm up -compare utility-mary utility-2.0

Baseline for utility-mary = utility-1.0
Baseline for utility-2.0 = utility-1.0
Tasks and Folders Only in Project utility-mary
     folder 1: All completed tasks for release 2.0
     folder 6: Assigned and completed tasks for mary for Release 2.0
     task   13: Insulated Development projects for mary for release 2.0
     task   14: Insulated Development products for mary for release 2.0

Tasks and Folders Only in Project utility-2.0
     folder 2: All tested tasks for release 2.0
     task   9: Integration Testing projects for release 2.0
     task   10: Integration Testing products for release 2.0

Tasks and Folders in Both Projects
```

## タスク／フォルダの削除

- タスク 3、12-14、40-42 を new-mary プロジェクトから削除する。

```
ccm update_prop -remove -tasks 3,12-14,40-42 new-mary

Removing tasks from new-mary...
   Removed task 3 from the update properties of project new-mary
   Removed task 12 from the update properties of project new-mary
```

```
   Removed task 13 from the update properties of project new-mary
   Removed task 14 from the update properties of project new-mary
   Removed task 40 from the update properties of project new-mary
   Removed task 41 from the update properties of project new-mary
   Removed task 42 from the update properties of project new-mary

Removed the following task(s) from project new-mary
   Task 3
   Task 12
   Task 13
   Task 14
   Task 40
   Task 41
   Task 42
Removed 7 tasks from new-mary.
```

- フォルダ 5-7 と 114 を new-mary プロジェクトから削除する。

```
ccm up -rem -folders 5-7,114 new-mary

Removing folders from new-mary...
Removed folder 5 from the update properties of project new-mary
Removed folder 6 from the update properties of project new-mary
Removed folder 7 from the update properties of project new-mary
Removed folder 114 from the update properties of project new-mary
Removed the following folder(s) from project new-mary
      Folder 5
      Folder 6
      Folder 7
      Folder 114

Removed 4 folders from new-mary.
```

- DCM 用に初期化されたデータベースで、frankfurt データベースからインポートした、フォルダ 37 を utility-mary プロジェクトから削除する。

```
ccm up -rem -folders frankfurt#37 utility-mary

Updating update properties of utility-mary...
Removed folder 'frankfurt#37: Nepal’s Tested Tasks'
Removed 1 folder.
```

## ベースライン、タスク、フォルダの情報の表示

- utility-mary プロジェクトの情報を再帰的に表示し、詳細なメッセージを表示する。

```
ccm up -show info utility-mary -v -r
```

### 更新メソッドの設定

- utility-mary プロジェクトの更新メソッドを、手動更新プロパティではなくテンプレートを使用するように設定する。

```
ccm up -ru template utility-mary -r

utility-mary has been set to update using the process rule that will
maintain its update properties automatically.
```

### 有効なベースラインの表示／設定

- utility-mary プロジェクトに使用可能なベースラインを表示する。

```
ccm up -vb utility-mary

1) utility-3.0
2) utility-3.1
```

- utility-mary プロジェクトのベースラインを utility-3.1 に設定する。

```
ccm up -mb utility-3.1 utility-mary

Set baseline to 'utility-3.1'
```

### ベースライン、タスク、フォルダ、オブジェクトの表示

- utility-mary プロジェクトのベースラインを表示する。

```
ccm up -sh bl utility-mary
```

- utility-mary プロジェクトのタスクを表示する。

  表示されるのは、プロジェクトの更新プロパティ に直接含まれているタスクのみです。

```
ccm up -sh tasks utility-mary

1) Task 5: mary’s insulated development projects
2) Task 6: mary’s insulated development products
3) Task 40: Auto-calculation gives incorrect result
4) Task 53: Download of images occurs too slow
```

- utility-mary プロジェクトのフォルダを表示する。

```
ccm up -sh folders utility-mary

1) Folder 111: mary’s Insulated Development Task Folder
2) Folder 146: mary’s Assigned Tasks
3) Folder 161: Tested Tasks for Release 3.2
```

- utility-mary プロジェクトのタスクおよびフォルダを表示する。

表示されるのは、プロジェクトの更新プロパティに直接含まれているタスクのみです。

```
ccm up -sh t_and_f utility-mary

1) 111 mary’s Insulated Development Task Folder <void>
2) 146 mary’s Assigned Tasks <void>
3) 161 Tested Tasks for Release 3.2 <void>
4) 40 <void> Auto-calculation gives incorrect result
5) 5 <void> mary’s insulated development projects
6) 53 <void> Download of images occurs too slow
7) 6 <void> mary’s insulated development products
```

- `utility-mary` プロジェクトのすべてのタスクを表示する。

  プロジェクトに関連付けられたすべてのタスクが表示されます。プロジェクトの更新プロパティに直接含まれているタスクと、フォルダを通じて間接的に含まれているタスクのどちらも表示されます。

  ```
  ccm up -sh all_tasks utility-mary

  1) Task 15: Correct spelling 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
  5) Task 5: mary’s insulated development projects
  6) Task 53: Download of images occurs too slow
  7) Task 6: mary’s insulated development products
  ```

- `utility-mary` プロジェクトのオブジェクトを表示する。

  ```
  ccm up -show objects utility-mary

  1) main.c-4:csrc:1 integrate  mary  26
  2) main.h-3:incl:1 integrate  mary  26
  3) msg.c-5:csrc:1  integrate  mary  19
  4) msg.h-4:csrc:1  integrate  mary  19
  ```

## update_template コマンド

`update_template` コマンドは process_rule コマンド の別名です。

# **use** コマンド

## 表記

```
ccm use [-t|-task task_number] [-r|-rules]
        file_spec [file_specÖ]
ccm use [-r|-rules]
        -p|-project project_spec [project_specÖ]
```

## 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。
- カレント 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。

  注記：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。初期化ファイル内の `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）。

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。

## オプションと引数

*`file_spec`*

使用するオブジェクト バージョンを指定します。

`-g`

適切なダイアログを呼び出します。

`-p|-project project_spec`

現在のディレクトリにプロジェクトを追加します。

`-r|-rules`

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_number`

use コマンドにより読み出し専用ディレクトリにオブジェクトが追加される場合、新たにチェックアウトされたディレクトリをタスク番号に関連付けます。

追加するオブジェクトをタスクと関連付けます。カレント（デフォルト）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、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。

## 例

- データベースにあるがプロジェクトにないオブジェクトを追加する。追加されるオブジェクトは書き込み可能である必要はありません。
  1. オブジェクト バージョンをクエリする。

     ```
     ccm query -n objectname
     ```

  2. 一覧表示されたクエリ項目から、使用するオブジェクト バージョンを確認する。
  3. オブジェクト バージョンをプロジェクトに追加する。

     ```
     ccm use @item_number
     ```

- `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
  ```

- 現在のバージョンではなく `display.c` のバージョン 2 を使用する。

  ```
  ccm use display.c-2
  ```

- 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
  ```

関連トピック

- delete コマンド
- unuse コマンド

# **users** コマンド

## 表記

```
ccm users
```

## 説明と用途

users コマンドにより、特定の Rational Synergy データベースの users リストを表示できます。デフォルト エディタを使用して、ユーザーとそのロールをファイルに追加、削除できます。

ユーザーのロールを設定するときには、最初のロールはユーザーが最も頻繁に実行するものにします （最初のロールとは、ユーザーがセッションを開始するときのデフォルトのロールです）。どのユーザーに対しても、可能な限り最初のロールとして *ccm_admin* を設定しないでください。

データベース ユーザーは、1 つのテーブルに 1 人 1 エントリで定義されています。1 つのデータベースに関して、1 つのユーザー リストが使用されます。ユーザーとその適切なロールを、以下のフォーマットで追加します。

*username* = *role(s)*

ユーザー名の次にユーザーがアクセス可能なロールを指定します。たとえば、ユーザーは 1 つのまたは複数のロールを持つことができます。以下にその例を示します。

```
user bob      =  developer ccm_admin;
user john     =  writer;
user mary     =  developer build_mgr ccm_admin;
```

注記：上記の例に示されているように、users リストの各エントリの末尾にはセミコロンを付けます。

1 人のユーザーが同じデータベース内で複数のロールを持っている場合、1 人のユーザーを 2 回リストしてはなりません。その代わりに、上記の例に示されているように、ユーザーのすべてのロールを列挙します（`user erin` と `user matt` の例を参照）。

このコマンドを使用するには、*ccm_admin* ロールを持っている必要があります。

## オプションと引数

なし

## 例

- データベースにユーザーを追加する。
  1. ユーザー リストを開きます。

     ```
     ccm users
     ```

  2. 既存の行をコピーしてユーザーをデータベースに追加し、新しいユーザーのユーザー名を置き換えます。
  3. リストを保存して閉じます。
- データベース内のユーザーのロールを変更する。
  1. ユーザー リストを開きます。

     ```
     ccm users
     ```

  2. ユーザー名の後ろにロールを追加または削除して、ユーザーのロールを変更します。
  3. リストを保存して閉じます。
- データベースからユーザーを削除する。
  1. ユーザー リストを開きます。

     ```
     ccm users
     ```

  2. 該当する行を削除して、データベースからユーザーを削除します。
  3. リストを保存して閉じます。

# **version** コマンド

## 表記

```
ccm version [-d|-dbschema] [-a|-all] [-c|-ccm] [-i|-informix]
```

## 説明と用途

`ccm version` コマンドにより、実行中の Rational Synergy のバージョンを表示します。

## オプションと引数

`-a|-all`

現在のデータベース スキーマ、Informix データベース サーバー、および Rational Synergy リリースのバージョンを表示します。

`-c|-ccm`

Rational Synergy リリースのバージョンを表示します。

スイッチを指定しないで `ccm version` と入力すると、Rational Synergy のリリース番号のみが表示されます。

`-d|-dbschema`

データベース スキーマのバージョンを表示します。

`-i|-informix`

データベース サーバーのバージョンを表示します。

## 例

実行中の現在のデータベース スキーマ、Informix データベース サーバー、および Rational Synergy リリースのバージョンを表示します。

```
ccm version -a
    INFORMIX Dynamic Server Version 10.00.UC5XA
    Rational Synergy Version 7.1a
    Rational Synergy Schema Version 0111
```

# **view** コマンド

## 表記

```
ccm view file_spec [file_spec...]
```

## 説明と用途

このコマンドを使用して、現在のディレクトリにないオブジェクトのソースを表示するため、または初期設定ファイル内の cli.text_viewer オプションによって設定されているビューアを起動します。

## オプションと引数

*file_spec*

表示するオブジェクトを指定します。

## 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

## **work_area** コマンド

### 表記

#### ワークエリア オプションの表示

```
ccm wa|work_area -show [-r|-recurse]
                 [-p|-project] project_spec [project_spec...]
```

#### ワークエリア オプションの変更

```
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] [-ntl|no_translation]
                 [-r|-recurse] [-nr|-no_recurse]
                [-setpath|-path|-set path]
                [-pst|-project_subdir_template template]
                 [-p|-project] project_spec [project_spec...]
```

#### ワークエリア パス内で指定文字列を持つすべてのプロジェクトの検索と表示

```
ccm wa|work_area -find "find_string" [-reg|-regexp] [-replace "new_string"]
                 -show
                 [-scope working|prep|shared|checkpoint|STATIC|ALL|DB]
                 [-p|-project] project_spec [project_spec...]
```

#### ワークエリア パス内で指定文字列の検索と置換

```
ccm wa|work_area -find "find_string" [-reg|-regexp] -replace "new_string"
                 [-scope working|prep|shared|checkpoint|STATIC|ALL|DB]
                 [-new|-visible]
                 [-p|-project] project_spec [project_spec...]
```

#### 指定データベース パスを持つすべてのプロジェクトの識別と表示

```
ccm wa|work_area -dbpath "old_database_path"
                 -show
                 [-scope working|prep|shared|checkpoint|STATIC|ALL|DB]
                 [-p|-project] project_spec [project_spec...]
                 [-ns|-nosync]
```

### データベース パスの識別と置換

```
ccm wa|work_area -dbpath "old_database_path"
                 [-scope working|prep|shared|checkpoint|STATIC|ALL|DB]
                 [-p|-project] project_spec [project_spec...]
```

## 説明と用途

work_area コマンドにより、ワークエリア オプションを表示して修正したり、ワークエリアとデータベースのパスを表示して更新できます。

ここでは、1 つまたは複数のプロジェクトと関連付けられたワークエリア情報を変更する方法について説明します。

### ワークエリア オプションの変更

以下に、プロジェクトで変更可能なワークエリア オプションを示します。

- ワークエリアを管理するかどうか（ファイル システムと同期させるか）-wa または -nwa
- サブプロジェクトを親プロジェクトに対して相対的とするかどうか -rel または -nrel
- ワークエリアをコピーベースとするか、リンクベースとするか -cb または -ncb
- ワークエリアの時刻を使用するかどうか -wat または -nwat
- ソースを ASCII ファイルまたはバイナリ ファイルのどちらでコピーするか -tl または -ntl
- ワークエリア パス（プロジェクトをファイル システムと同期させるか） -setpath *path* または -pst *path*
- オプションが静的な場合でも、ワークエリア内のファイルが読み出し専用に設定されていないか -mod または -nmod

ワークエリア パスのプロジェクト固有部分を変更することにより、プロジェクトのワークエリア パスを新しい場所に設定することもできます。この変更は、-pst|-project_subdir_template *template* オプションを使用して行います。

### 古くなったワークエリア パスの更新

プロジェクトのワークエリア パスはファイル システム内のプロジェクトへの絶対パスです。たとえば、ワークエリア パスには以下のようなものがあります。

Windows： c:¥ccm_wa¥tools¥ordtime-john
UNIX： /users/john/ccm_wa/tools/ordtime-john

データベースを新しいデータベースへコピーまたはアンパック（Windows では ccmdb cp または ccmdb unpack、UNIX では ccmdb_cp または ccmdb_unpack）したい場合、新しいデータベースのプロジェクトは、古いデータベースのプロジェクトが同期されていたワークエリア パスではなく、新しいワークエリア パスと同期させるでしょう。そこで、古いワークエリア パスを新しいもので置き換える必要が生じます。-find "*find_string*" -replace "*new_string*" オプションにより、この変更を行います。

-show オプションを使用して、ワークエリア パスの変更をリハーサルできます。

### 古くなったデータベース パスの更新

各プロジェクトには対応するデータベース パスがあります。たとえば、プロジェクトのデータベース パスには以下のようなものがあります。

Windows : `y:¥stanton¥ccmdb¥ccm_docs¥db`
UNIX : `/vol/stanton/ccmdb/ccm_docs/db`

データベースを新しいデータベース パスに移動すると、新しい場所でのプロジェクトのデータベース パスが古いままで正しくない可能性があります。したがって、古いデータベース パスを新しいデータベース パスに置き換える必要があります。 -dbpath "*old_database_path*" オプションを使用して、この変更を行います。

-show オプションを使用して、データベース パスの変更をリハーサルできます。

## オプションと引数

注記：すべてのオプションのデフォルトは指定プロジェクトに設定された値です。

`-cb|-copy_based`

ワークエリアに、リンクではなくファイルのコピーを格納するよう指定します。Windows では、ワークエリアはコピーベースです。

`-dbpath "old_database_path"`

Windows:
_ccmwaid.inf（Rational Synergy ワークエリア ID）ファイルに *old_database_path* を持つプロジェクトを識別し、そのプロジェクトの _ccmwaid.inf ファイルのデータベース文字列を現在のデータベースのパスに変更します。

このオプションは、データベースの移動時にデータベース パスを更新するために使用します。

UNIX：
`.ccmwaid.inf`（Rational Synergy ワークエリア ID）ファイルに *`old_database_path`* を持つプロジェクトを識別し、そのプロジェクトの `.ccmwaid.inf` ファイルのデータベース文字列を現在のデータベースのパスに変更します。

このオプションは、データベースの移動時にデータベース パスを更新するために使用します。

このオプションを使用するには、ワークエリアに対する書き込みアクセス権限が必要です。書き込み禁止プロジェクトのワークエリア ID ファイルを更新するには、*ccm_admin* ロールを持っている必要があります。

> 注記: リンクベースのワークエリアでは、新しいデータベースに対するシンボリック リンクを更新するために、ワークエリアは自動的に同期されます。また、リンクベースのワークエリアを更新する前に、コピーベースのワークエリアをすべて更新する必要があります。コピーベースのワークエリアは、リモート クライアント（`-rc` オプションを使用して起動したセッション）から更新します。
>
> ワークエリア パス名を更新した後で、`-rc` オプションを指定せずにセッションを再開して、リンクベースのワークエリアを更新します。

`-find "`*`find_string`*`"`

ワークエリア パス名に *`find_string`* が含まれるプロジェクトを検索します。このオプションは、古くなったワークエリア パスを更新するために使用します。

*`find_string`* 内で正規表現（例、`".*"`）を使用したい場合、`-reg` オプションを使用します。詳細については、正規表現を参照してください。

このオプションを使用するには、ワークエリアに対する書き込みアクセス権限が必要です。書き込み禁止プロジェクトのワークエリア パス名を更新するには、*ccm_admin* ロールを持っている必要があります。

UNIX では、リンクベースのワークエリアを更新する前に、コピーベースのワークエリアをすべて更新する必要があります。コピーベースのワークエリアは、リモート クライアント（`-rc` オプションを使用して起動したセッション）から更新します。

ワークエリア パス名を更新した後で、-rc オプションを指定せずにセッションを再開して、リンクベースのワークエリアを更新します。

-ncb|-not_copy_based

ワークエリアに、コピーではなくファイルへのリンクを格納するよう指定します。このオプションは Windows では使用できません。

-new

-find -replace と一緒に使用して、新しいデータベースであることを指定します。このオプションは 2 通りの状況を想定しています。find オプションによって指定されたワークエリアが可視でない場合、あるいは find オプションによって指定されたワークエリアが可視であるが無視すべき場合です。

このオプションを指定しないと、ワークエリアが可視であるプロジェクトに関してのみ、コマンドが機能します。可視ワークエリアの更新に関する情報については、-visible オプションを参照してください。

このオプションは、-find オプションと一緒にのみ使用できます。

-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

ワークエリアを親プロジェクトのワークエリアに対して相対的としません。UNIX ではサブプロジェクトにリンクを使用します。プロジェクトを初めて作成するとき、これがデフォルトとなります。

-ns|-nosync

ワークエリアを同期させないよう指定します。

-ntl|-no_translation

ソースをバイナリ ファイルとしてコピーするよう指定します。

`-nwa|-no_wa`

ワークエリアを管理しないことを指定します。このオプションを指定すると、ワークエリアがデータベースから切り離されます。

このオプションにより、データベースにのみに存在し、対応するワークエリアを持たないプロジェクトを保持することが可能になります （対応するワークエリアのないプロジェクトを「グループピング プロジェクト」といいます。）このオプションは、ワークエリアの使用が不要または不可能な、プロジェクトの論理グループを作成するために使用します。

- たとえば、以下の例ではワークエリアは不要です。

ビルド マネージャが、現在はまだ存在しないが近い将来導入予定であるプラットフォーム用のプロジェクトを作成するとします。新しい Windows OS 用のプラットフォームがいずれ必要になるので、このオプションを使用してワークエリアの管理をオフに設定します。プロジェクトを作成します。新しいプラットフォームのプロジェクトを使用できるようになったら、`-wa` オプションを使用して、新しいプロジェクトをワークエリアに同期させます。

- 以下のシナリオでは、ワークエリアは使用できません。

Windows XP プラットフォームと Solaris プラットフォームでリリースする予定の、ソフトウェアを開発しているとします。それぞれのプラットフォーム用のプロジェクトのワークエリアは、両方のプラットフォームからは見えない可能性があります。しかし、プロジェクトを両方とも表示したいので、それをグループ化することにします。これら両方をグループ化するプロジェクトをサブプロジェクトとして作成するためには、ワークエリアのないプロジェクトを使用する必要があります。

`-nwat|-no_wa_time`

新しいタイムスタンプを使用しないよう指定します。つまり、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初の修正時刻を反映して、ファイルのタイムスタンプを設定します。

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

指定したオプションを適用するプロジェクトを指定します。

変更を適用するプロジェクトの *`project_name-version`* を指定します。ただし、オプション名 `-p` を入力する必要はありません。

`-pst|-project_subdir_template] template`

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 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 *`atabase_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、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。

`-reg|-regexp`

*`new_string`* と *`find_string`* が正規表現であることを示します。

このオプションは、`-find` オプションと一緒にのみ使用できます。

`-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。プロジェクトが修正不可であれば、相対ワークエリアは複数のプロジェクト内で使用できます。修正可能な場合は、1 つのプロジェクト内でのみ使用できます。

`-replace "`*`new_string`*`"`

`-find` *`find_string`* オプションを使用して見つかったすべてのプロジェクトのワークエリア パス内の、*`find_string`* を *`new_string`* で置き換えます。

*new_string* に関して正規表現（例、".*"）を使用したい場合、-reg オプションを使用します。

このオプションは、-find オプションと一緒にのみ使用できます。

-scope

-find または -dbpath のいずれかのオプションを使用してプロジェクトを検索するための、初期基準を設定します。

scope（適用範囲）として、以下のいずれかの状態を取りことができます。

*working*（*working* 状態にあるすべてのプロジェクト）
*checkpoint*（*checkpoint* 状態にあるすべてのプロジェクト）
*prep* （すべての *prep*（準備）プロジェクト）
*shared*（すべての *shared* プロジェクト）

あるいは、以下のいずれかのキーワード（大文字と小文字を区別）を取りことができます。

STATIC（書き込み可能でないすべてのプロジェクト、たとえば、*integrate*、*test*、*sqa*、または *released* 状態）
ALL（状態を問わず、すべてのプロジェクト）
DB（所有または状態にかかわらず、現在のデータベースのすべてのプロジェクト）

適用範囲が *working*、*checkpoint*、または ALL である場合、プロジェクトは自分が所有している必要があります。

デフォルトの適用範囲は *working* です。

このオプションは、-find オプションまたは -dbpath オプションと一緒にのみ使用できます。

-set|-path|-setpath *path*

指定したプロジェクトのワークエリア パス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 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 内のデータベース名です。これがワークエリア パスです。ワークエリアのファイル システムの別の部分を使用する場合、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、このオプションを使用してワークエリア パスを変更できます。

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。

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、*ccm_admin* のロールを持っている必要があります。

-show

-find または -dbpath のどちらのオプションも使用していない場合に、指定したプロジェクトのワークエリア オプションを表示します。-find または -dbpath のいずれかのオプションを使用している場合、a) 適用範囲基準を満たすか、-p オプションによって指定されているか、または選択セット内にあり、b) 指定した文字列またはデータベース パスを含む、プロジェクトが示されます。

-replace と一緒にこのオプションを使用すると、この操作によって得られる結果が示されます。

-visible

可視のワークエリアのみを更新の対象とするよう指定します。

これはデフォルト設定です。インターフェイスから見えないためにスキップされたワークエリアについて、メッセージが表示されます。

ワークエリアが作成されていないデータベース内のプロジェクトのワークエリア パスの更新については、-new オプションを参照してください。

このオプションは、-find オプションと一緒にのみ使用できます。

-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> 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* ロール持ってい必要があります。

`-wat|-wa_time`

新しいタイムスタンプを使用するよう指定します。つまり、Rational Synergy によってファイルが修正された時刻ではなく、ファイルがワークエリアにコピーされた時刻を反映して、ファイルのタイムスタンプを設定します。

サード パーティの Make ツールを使用しているときにはこの設定を使用します。このオプションを使用しないと、ファイルの修正時刻がワークエリアに記録されます。これは適切に思われますが、ファイルの別バージョンがワークエリアに取り込まれたときには問題が生じます。このワークエリアに依存する製品よりファイルの修正時刻が古いと、サードパーティの Make ツールは製品の再ビルドが必要であると判断できません。

関連トピック

- work_area コマンドの例
- delimiter コマンド
- reconcile コマンド
- resync コマンド
- sync コマンド

## **work_area** コマンドの例

以下の操作の例について説明します。

- ワークエリア オプションの表示
- ワークエリア オプションの変更
- ワークエリア パス内で指定文字列の検索と置換
- データベース パスの識別と置換

### ワークエリア オプションの表示

- `ico_may2-1` プロジェクトのワークエリア オプションを表示する。

  ```
  ccm work_area -show -p ico_may2-1
  ```

### ワークエリア オプションの変更

- 指定したプロジェクトのパスを変更し、プロジェクトを同期させる。

  Windows:
  ```
  ccm work_area /setpath c:¥users¥linda¥new_wa¥database /p ico_feb1-1
  ```
  UNIX:
  ```
  ccm work_area -setpath /users/linda/new_wa/database -p ico_feb1-1
  ```

### ワークエリア パス内で指定文字列の検索と置換

- ワークエリアが見えていて、パスに文字 "-" が含まれる、すべての *working* プロジェクトを検索して、"-" を "~" に変換する。

  ```
  ccm wa -find "-" -replace "~"
  ```

  このコマンドはデータベースの区切り文字を変更する場合に適しています。すべてのワークエリアをカバーするためには、十分な数のセッションから、各ユーザーがこのコマンドを実行する必要があります。ユーザーのすべてのワークエリアが 1 つのセッションから見える場合には、1 セッションで十分です。しかし、ユーザーが Windows と UNIX の両方のワークエリアを使用している場合、それぞれのクライアントからこのコマンドを実行する必要があります。

- ワークエリアが見えていて、パスに文字 "-" が含まれる、すべての *prep* プロジェクトを検索して、"-" を "~" に変換する。

  ```
  ccm wa -find "-" -replace "~" -scope prep
  ```

  これはデータベースの区切り文字を変更する場合に適しています。すべてのビルド管理ワークエリアをカバーするためには、十分な数のセッションから、ビルド マネージャがこのコマンドを実行する必要があります。すべての *prep* ワークエリアが 1 つのセッションから見える場合には、1 セッションで十分です。しかし、Windows と UNIX

の両方のビルド管理ワークエリアがある場合、それぞれのクライアントからこのコマンドを実行する必要があります。

- 新たにコピーされたデータベースで元のデータベースのパスを使用している場合に、文字列 `platform` が含まれるすべてのワークエリア パスを検索して、文字列を `services` に変更する。

  ```
  ccm wa -find platform -replace services -new
  ```

  このコマンドは、新しい名前にアンパックまたはコピーされたデータベースに適しています。ここでは、すべての新しいワークエリアを作成するために、`-new` オプションを使用しています。古いワークエリアは、古いデータベース用だからです。移動されたデータベースなどで古いワークエリアを再使用する場合、まず `-dbpath` オプションを使用してワークエリア ID ファイルを更新して、このデータベースの古いワークエリアが見えるようにする必要があります。

- Windows で、正規表現を使用して、`¥+joe45` ディレクトリを削除することにより、ワークエリア パス `c:¥users¥joe¥+joe45¥hsaw~1` を `c:¥users¥joe¥hsaw~1` に短縮する （先行する特殊文字に注意してください）。

  ```
  ccm wa -find "¥+joe45¥¥¥" -reg -replace "" -p hsaw~1
  ```

### データベース パスの識別と置換

データベース `/vol/acrel5/ccmdb/ccm_platform` のワークエリアを持つすべての working プロジェクトを検索し、現在のデータベースへのパスでワークエリア ID ファイルを更新する。

```
ccm wa -dbpath "/vol/acrel5/ccmdb/ccm_platform"
```

このコマンドは、ファイル システム内の新しい場所に移動されたが、古いワークエリアが見えているデータベースに適しています。すべてのワークエリアをカバーするためには、十分な数のセッションから、各ユーザーがこのコマンドを実行する必要があります。ユーザーのすべてのワークエリアが 1 つのセッションから見える場合には、1 セッションで十分です。しかし、ユーザーが Windows と UNIX の両方のワークエリアを使用している場合、それぞれのクライアントからこのコマンドを実行する必要があります。

## デフォルト

`ccm.ini` ファイル（`Windows`）または `.ccm.ini` ファイル（UNIX）内で wa_path_template オプションと project_subdir_template オプションを設定するか、または work_area コマンド を使用できます。

## wa_snapshot コマンド

`wa_snapshot` コマンドは copy_to_file_system コマンドの別名です。

# 高度なトピック

以降のページでは、Rational Synergy の使用に関する高度な詳細情報を示します。

- コンフリクト検出

  コンフリクト検出では、Rational Synergy が検出するコンフリクトのタイプを示します。
- 日付形式

  日付形式では、Rational Synergy 内の日付と時刻の形式を設定し、変更する方法を説明します。
- マージ ツールの定義

  マージ ツールの定義では、デフォルト マージ ツールの代わりに他のマージ ツールを使用する方法を説明します。
- マイグレーション ルール

  マイグレーション ルールでは、マイグレーション ルールを説明し、詳細な例を示します。
- クエリ式

  クエリ式では、クエリ式の構築方法を詳細に説明します。
- 関係

  関係では、Rational Synergy で使用するオブジェクト間の関係を説明します。
- 共有プロジェクト

  共有プロジェクトでは、共有プロジェクトの概念と用途を紹介します。
- SOAD スコープ

  *SOAD* スコープでは、オフライン保存と削除（SOAD）ツールが、オフライン保存して削除するオブジェクトのリストを、スコープを使用してどのように作成するかを示します。
- トリガ

  トリガでは、オブジェクト状態変更や問題送出時など、データベースでの動作をユーザーに通知するために使用するプログラムの定義方法を説明します。

- ワークエリア

  ワークエリアでは、絶対と相対のワークエリア、コピーベースとリンクベースのワークエリアを説明し、例を示します。

- ワークエリア コンフリクト

  ワーク エリア コンフリクトでは、遭遇する可能性のあるワーク エリア コンフリクトの種類を示し、ワーク エリア コンフリクトの解決方法を説明します。

# コンフリクト検出

コンフリクト検出とは、ワークエリアまたはプロジェクトが期待したものから逸脱していることを警告するために Rational Synergy が用意している基準とその基準に基づいた処理方法のことです。Rational Synergy のコンフリクト検出は、プロジェクトのタスクと履歴の関係を追跡してプロジェクトが正しいメンバーをすべて含むことを確認します。しかし、基準からの逸脱が必要と思われる場合には、Rational Synergy は警告を出して、ユーザーがその後の処理を決定できるようにします。コンフリクトには以下に示すようにいろいろな種類があります。

- 開発者として作業している場合、ワークエリアとデータベースの同期がとれていないとき、またはファイル、ディレクトリまたはプロジェクトにパラレル バージョンがあるとき、コンフリクトが発生することがあります。Rational Synergy で作業時に発生するコンフリクト解決の具体的な情報については、オンライン ヘルプを参照してください。
- ビルド マネージャとして作業している場合、プロジェクトのメンバーとその更新プロパティの間にコンフリクトが発生することがあります。

コンフリクトの表示は、更新の直後に行うことを推奨します。なぜなら、その時点ではプロジェクト メンバーが プロジェクトの更新プロパティ と一致しているからです。

更新プロパティにクエリを使用するフォルダが含まれている場合、そのフォルダの内容を更新できるのは最後の更新の後です。プロジェクトの更新プロパティとプロジェクトのメンバーとの間の不一致はコンフリクトとして表示されるので、最近更新を行っていない場合、プロジェクトからさらにコンフリクトが示される可能性があります。

以下のトピックについて説明します。

- Rational Synergy が検出するコンフリクト
- タスクとオブジェクトの関係
- コンフリクトのタイプ
- 大規模コンフリクト検出

## **Rational Synergy** が検出するコンフリクト

作業中、コマンドラインまたはグラフィカル ユーザー インターフェイスからコンフリクトを表示できます。コンフリクトが発生していると、コンフリクトを引き起こしているオブジェクトのバージョンと関連タスクと、各オブジェクトのコンフリクトについて簡単な説明が表示されます。コンフリクト メッセージを下表に定義します。

下表とそれに続くコンフリクト検出の説明では、以下の定義を使用します。

- 「コンフリクト」は、次のいずれかの状況として定義します。

  プロジェクトに含まれないと指定されたタスクに関連付けられたファイルが含まれている。

  プロジェクトに含まれると指定したタスクに関連付けられたファイルが含まれていない。

  ファイルのタスク関係が期待したものに反する（たとえば、あるオブジェクトに関連付けられたタスクが存在しないか、複数存在するなど）。
- 「明示的」とは、「直接依頼されている」、つまりプロジェクトの更新プロパティに含まれていることを意味します。
- 「暗黙的」とは「間接的に依存または部分的に含まれる」こと、つまりプロジェクトの更新プロパティに含まれないことを意味します。

| コンフリクト メッセージ | コンフリクトのデフォルト表示 / 非表示 | 説明 |
|---|---|---|
| No task（タスクなし） | 表示 | オブジェクト バージョンが暗黙的にこのプロジェクトに含まれていますが、タスクと関連付けられていません （このオブジェクト バージョンを明示的に含めることができません。そのためにはタスクをプロジェクトの更新プロパティに含める必要があるからです）。 |
| Multiple Tasks（複数タスク） | 非表示 | オブジェクト バージョンは、プロジェクトに含まれ、複数のタスクに関連付けられています。 |
| Implicitly included（暗黙的に含まれる） | 表示 | オブジェクト バージョンは、明示的に指定されていませんが、プロジェクトに含まれています。 |
| Included by "use" operation?（"use" 操作で含まれているか） | 表示 | オブジェクト バージョンは明示的に指定されておらず、暗黙的にも要求されていないため、更新で選択されることはありません。 |

| コンフリクト メッセージ | コンフリクトのデフォルト表示 / 非表示 | 説明 |
|---|---|---|
| Implicitly required but before baseline（暗黙的に要求されているがベースラインより前） | 非表示 | オブジェクトバージョンは暗黙的に要求されていますが、ベースラインの直前バージョンです。（これは暗黙的に含まれるため、実際にはコンフリクトではありませんが、処理上の問題となる可能性を示しています）。 |
| Implicitly required but not included - newer（暗黙的に要求されているが含まれていない - より新しい） | 表示 | オブジェクト バージョンは、暗黙的に要求されていますが、プロジェクトに含まれていません。これは、現在選択されているバージョンよりも新しいバージョンのオブジェクトです。 |
| Implicitly required by multiple tasks - newer（複数のタスクによって暗黙的に要求されている - より新しい） | 表示 | オブジェクト バージョンは、暗黙的に要求されています。これは、プロジェクト内の別のファイルが複数のタスクに関連しているという理由で暗黙的に含まれるタスクに、関連付けられているからです。コンフリクトのあるオブジェクト バージョンはプロジェクトには含まれません。このオブジェクト バージョンは、現在プロジェクト内にあるファイルの後継バージョンです。 |
| Implicitly required but not included - parallel（暗黙的に要求されているが、含まれていない - パラレル） | 表示 | オブジェクト バージョンが暗黙的に要求されていますが、プロジェクトに含まれていません。そのバージョンは現在選択されているバージョンのパラレル バージョンで、マージが必要な場合があります。 |
| Implicitly required by multiple tasks - parallel（複数のタスクで暗黙的に要求されている - パラレル） | 表示 | オブジェクト バージョンは暗黙的に要求されています。これは、プロジェクト内の別のファイルが複数のタスクと関連しているという理由で暗黙的に含まれるタスクに、関連付けられているからです。コンフリクトのあるオブジェクト バージョンはプロジェクトには含まれません。このオブジェクト バージョンは、現在プロジェクト内にあるファイルのパラレル バージョンです。 |
| Explicitly specified but before baseline（明示的に指定されているが、ベースラインより前） | 非表示 | オブジェクト バージョンは、プロジェクトで明示的に指定されていますが、ベースラインの直前バージョンです （これは暗黙的に含まれるため、実際にはコンフリクトではありませんが、処理上の問題となる可能性を示しています）。 |

| コンフリクト メッセージ | コンフリクトのデフォルト表示 / 非表示 | 説明 |
|---|---|---|
| Explicitly specified but not included - newer（明示的に指定されているが含まれていない - より新しい） | 表示 | オブジェクト バージョンは、プロジェクトで明示的に指定されていますが、現在選択されているバージョンの後継バージョンのオブジェクトです。 |
| Explicitly specified but not included - parallel（明示的に指定されているが、含まれていない - パラレル） | 表示 | オブジェクト バージョンはプロジェクトで明示的に指定されていますが、プロジェクトに含まれていません。そのバージョンは現在のバージョンのパラレル バージョンで、マージが必要な場合があります。 |
| Explicitly specified but object not in project（明示的に指定されているが、プロジェクトにオブジェクトがない） | 非表示 | オブジェクト バージョンはプロジェクトで明示的に指定されていますが、プロジェクト内にそのバージョンが 1 つもありません。これは、プロジェクト階層全体で同じ更新プロパティが共有されているため、おそらく正常です。 |
| Implicitly required but object not in project（暗黙的に要求されているが、プロジェクトにオブジェクトがない） | 非表示 | オブジェクト バージョンがプロジェクトに含まれるタスクを通して暗黙的に要求されていますが、プロジェクト内にそのバージョンがありません。これは、プロジェクト階層全体で同じ更新プロパティが共有されているため、おそらく正常です。 |
| **task** オプションと一緒に表示された場合に、タスクに対して検出されるコンフリクト（メッセージとして表示） | | |
| Implicitly included（暗黙的に含まれる） | 表示 | タスクは暗黙的にプロジェクトに含まれます。 |
| Implicit task from explicit object（明示的オブジェクトからの暗黙的タスク） | 表示 | このタスクの関連ファイルには複数のタスクが割り当てられています。オブジェクトの関連タスクの少なくとも 1 つが明示的です（つまり更新プロパティに含まれます）が、このタスクはそうではありません。 |
| Implicitly required but not included（暗黙的に要求されているが含まれていない） | 表示 | タスクが暗黙的に要求されていますが、プロジェクトに含まれていません。 |

| コンフリクト メッセージ | コンフリクトのデフォルト表示 / 非表示 | 説明 |
|---|---|---|
| Explicitly specified but not included（明示的に指定されているが含まれていない） | 表示 | タスクはプロジェクトによって明示的に指定されていますが含まれていません。 |
| Explicitly specified but object not in project（明示的に指定されているが、プロジェクトにファイルが存在しない） | 非表示 | タスクがプロジェクトで明示的に指定されていますが、そのファイルがプロジェクトに含まれていません。これは、プロジェクト階層全体で同じ更新プロパティが共有されているため、おそらく正常です。 |
| Excluded task explicitly included（除外タスクが明示的に含まれている） | 表示 | 除外したタスクが、ベースラインとプロジェクトのプロジェクト グルーピングのタスクに含まれています。 |
| Excluded task implicitly included（除外タスクが暗黙的に含まれている） | 表示 | 除外したタスクがプロジェクトの更新プロパティに暗黙的に含まれています。 |
| Completed fix task not included（完了修正タスクが含まれない） | 表示 | 問題のあるタスクがプロジェクトの更新プロパティに含まれており、修正の完了している問題のないタスクが含まれていません。 |
| Assigned fix task not included（割り当て済み修正タスクが含まれない） | 非表示 | 問題のあるタスクがプロジェクトの更新プロパティに含まれており、修正のための `task_assigned`（タスク割り当て）された問題のないタスクが含まれていません。 |
| Task fixed by task not included（タスクによって修正されたタスクが含まれていない） | 表示 | 問題のあるタスクはプロジェクトの更新プロパティに含まれませんが、問題のないタスクは含まれます。 |

## タスクとオブジェクトの関係

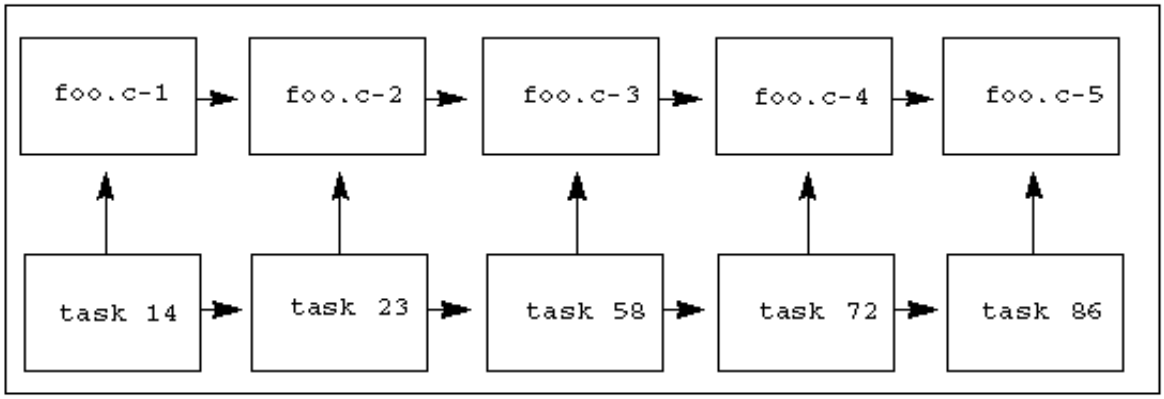
タスクと一連のオブジェクト バージョンは関係を持つことができます。Rational Synergy では一連のオブジェクト バージョンを 1 つのタスクと関連付けることができます。このタスクは、Rational Synergy にそれらのオブジェクト バージョンは一緒に使用する必要があり、互いの変更に依存することを知らせます。プロジェクトがタスクに関連する変更の一部のみ含む場合、そのプロジェクトはおそらくビルドに失敗します。あるいは、最悪の場合、実行に失敗します。

たとえば、関数のシグニチャを変更する場合は、その関数を呼び出すすべてのプログラムを更新してシグニチャを変更する必要があります。それらすべての変更は、まとめてプロジェクトに含めるか、全く含めないようにする必要があります。

### オブジェクトの履歴関係

タスクは履歴関係を持ちますが、それはオブジェクトの履歴関係とは異なります。オブジェクトの履歴は、通常、数的に連続します。タスクの履歴は、関連するファイルの履歴関係をベースにした単なる概念的な関係です。タスクは変更を完了するために必要なファイルをグループ化するため、タスクの履歴関係によって、現在の一連の変更は過去の一連の変更に依存することになります。

下図は、1 つのファイルと、オブジェクトの履歴を通してそのファイルに関連付けられる各タスクのバージョン履歴を示しています。




`foo.c` ファイルには 5 つのバージョンがあります。各バージョンは、異なるタスクと関連付けられています （各バージョンに関連するタスク番号がオブジェクト バージョンの下に表示されています)。

Rational Synergy は、オブジェクト バージョンの変更は、そのすべての直前オブジェクト バージョンの変更を含むものとみなします。したがって、上記の例では、バージョン 3 は、バージョン 2 と 1 の変更を含むものとみなします。

たとえば、バージョン 2 で関数のシグニチャを変更すると、バージョン 3、バージョン 4 およびその後のすべてのバージョンはそのシグニチャの変更を含みます。変更は、他の変更の上に重なっていきます。他の変更の一部を取り除く変更でも、その履歴バージョンの上に積み重なります。ただし、`foo.c-3` の変更といった場合に、その変更とは `foo.c-2` に対しての変更ということになります。foo.c-2 に変更を加えたのが foo.c-3 であるからです。

### タスク依存関係

さらに、バージョン 3 はバージョン 1 と 2 からの変更を含むため、バージョン 3 の関連タスクはバージョン 1 と 2 に関連するタスクに依存するとみなされます。したがって、この例ではタスク 58 はタスク 23 と 14 に依存します。

### 明示的に指定された更新プロパティ

`foo.c-4` を含むプロジェクト `myproj-bill` を見てみましょう。

プロジェクトの更新プロパティに、ベースライン プロジェクトとタスク リストが含まれていることを思い出してください （これにはタスク フォルダも含むことができますが、タスク フォルダはタスクの集まりです）。タスクをプロジェクトの更新プロパティ内で指定した場合（直接、またはタスク フォルダを使用して）、このプロジェクトは、プロジェクトにそのタスクに関連付けられているファイルを含むように明示的に指定したことになります。たとえば、`myproj-bill` プロジェクトの更新プロパティにタスク 72 と 23 が含まれる場合、それは、タスク 72 と 23 に関連付けられたオブジェクト バージョンを含む必要があることを明示的に指定しています。上記の図では、プロジェクトが明示的にタスク 72 と 23 を指定した場合、オブジェクト バージョン `foo.c-4` および `foo.c-2` も明示的に指定したことになっています。

`foo.c-4` は、`foo.c-2` からの変更を含み、タスク 72 はタスク 23 に依存することを忘れないでください。

プロジェクトを更新すると、明示的に指定されたオブジェクト バージョンがその候補となります。更新では、もっとも適切な候補が、通常は最新のものが選択されます。したがって、この例では、`myproj-bill` はタスク 72 と 23 を使用して候補リスト `foo.c-4` と `foo.c-2` を決定し、最新候補として `foo.c-4` を選択します。したがって、このプロジェクトは `foo.c-4` と `foo.c-2` の両方からの変更を含みます。同じように、タスク 72 と 23 の両方からの変更も含みます。

### 暗黙的に指定された更新プロパティ

myproj-bill プロジェクトは foo.c-4 を含むため、このプロジェクトは、その更新プロパティが明示的に指定したタスク 72 を含みます。また、foo.c-3 は foo.c-4 の直前バージョンなので、プロジェクトは foo.c-3 にも依存します。また、foo.c-3 に関連する タスク 58 にも依存します。

ただし、タスク 58（すなわち foo.c-3）が myproj-bill プロジェクトの更新プロパティに明示的に指定されていないが、変更はその履歴関係から含まれる場合、タスクとオブジェクト バージョンは両方ともプロジェクト内で暗黙的に指定されています。暗黙的に指定されたタスクに関連付けられたファイルは、プロジェクトに自動的には含まれないことに注意してください。

## コンフリクトのタイプ

ソフトウェア アプリケーションをリリースする準備をしていると仮定します。あなたが、リリースはタスク 72 と 23 を含む必要があることを指定しましたが、タスク 58 を指定しなかったとします。ビルドを行った後、準備しているアプリケーションにタスク 58 が含まれていることに驚くかもしれません。Rational Synergy は、あなたが依頼していないタスクが含まれることを警告できます。これをコンフリクトといいます。

コンフリクトにはいろいろな種類があります。プロジェクト内で `foo.c-5` を手動で使用したのに、プロジェクトの更新プロパティが明示的にタスク 86 を指定せず、明示的に指定したその他のタスクもタスク 86 に依存しない場合、それも一種のコンフリクトです。Rational Synergy は、あるオブジェクト バージョンがそのタスクを明示指定しないでプロジェクト内で使われているようだ、ということを警告できます。

もっと重大なコンフリクトもあります。

たとえば、チームは、どの変更がバグを修正したか知るのが困難なため、ファイルの 1 つのバージョンで 1 つ以上のバグを修正しないと決定します。さらに、あなたのチームは、各開発者は変更する各オブジェクトバージョンに1タスクのみ関連付ける必要があると決定します。この場合、もしリリース準備中のリリース内のオブジェクトバージョンが複数のタスクに関連付けられていたら、開発者にそれを止めさせるためにあらかじめ気が付く必要があります。もっとも、リリース準備をしているソフトウェアには必要な変更はすべて入っているので、さほど深刻なコンフリクトとはいえないでしょう。

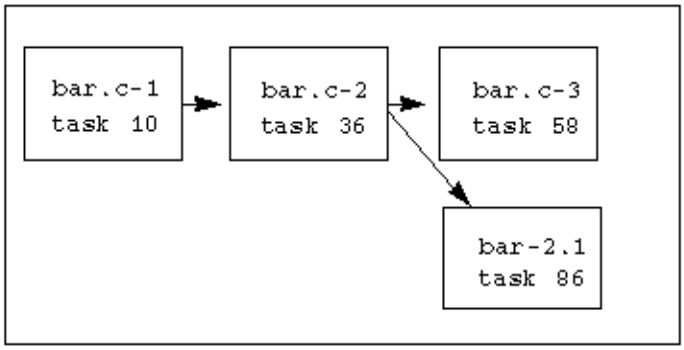
より重大なコンフリクトは、どのタスクとも関連付けられていない、暗黙的に含まれるオブジェクト バージョンです。

Rational Synergy は、どちらの種類のコンフリクトについても警告できます。

### パラレル コンフリクト

もっとも重要なコンフリクト検出の 1 つに、パラレル オブジェクト バージョンの検出があります。

プロジェクトが明示的に変更を指定しているが、それが含まれない場合、それは重大なコンフリクトです。たとえば、2 つのパラレル ファイルが 2 つの異なるタスクと関連付けられていて、両タスクとも明示的に指定されている状態を考えてみましょう。この例では、`myproj-bill` に `bar.c-3` が含まれると仮定します。`bar.c` ファイルは、下図に示す履歴関係とタスク関連があります。




myproj-bill プロジェクトの更新プロパティは、タスク 58 と 86 を含まれるべきだと指定します。しかし、プロジェクトはタスク 58 に関連付けられている bar.c-3 のみ含みます。パラレル ブランチであるため、タスク 86 に関連付けられている bar.c-2.1 も含むことは不可能です。bar.c のバージョンで、依頼した両方の変更を含むものはありません。これは、プロジェクトに含める必要があることがわかっているバージョンが欠落しているため、重大なコンフリクトです。

パラレル コンフリクトは変更の欠落を意味することもできますが、他の種類の変更の欠落もあります。

## 変更の欠落

bar.c-2.1 を手動で（ccm use を使って、または Use ダイアログから）myproj-bill プロジェクトに含めた場合にどうなるか考えてみましょう。タスク 58 と 86 両方の変更は、現在プロジェクト内にあるファイルのバージョンよりも新しいため、明示的に指定されているにもかかわらず欠落します。

この変更は、明示的に指定されていますが欠落しています。プロジェクトの更新プロパティに含まれるタスクがプロジェクトに含まれているかを確認して、欠落していることに気付くかもしれません。他の種類のコンフリクトは検出することがさらに困難です。

myproj-bill プロジェクトの更新プロパティを最新の情報に更新して、タスク 86 と 58 ではなく、タスク 86 と 36 を含めたとします。すると、タスク 58 は明示的に指定されなくなります。タスク 86 は foo.c-5 と関連付けられ、その直前バージョンはタスク 72 に関連付けられている foo.c-4 です。したがって、タスク 86 は暗黙的にタスク 72 を含みます。あなたのプロジェクトが foo.c-5 を含む場合、それは両方の変更を含み、何も問題ありません。しかし、bar.c はどうでしょうか。bar.c-2.1 はタスク 86 に関連付けられているため明示的に指定されており、bar.c-3 はタスク 58（プロジェクトに暗黙的に含まれる）に関連付けられているため暗黙的に指定されています。したがって、今回もあなたのプロジェクトが依頼したすべての変更を含む bar.c のバージョンがありません。

## 大規模コンフリクト検出

次に、その履歴に多数のバージョンを持ち、それぞれ数百のメンバーを持つプロジェクトと数百のタスクを含むリリースを見てみましょう。どんなにチームが注意しても、プロジェクトが大きくなるにつれてエラーの可能性も増えます たとえば、パラレル開発（マージの見落とし）によるものや人的エラー（オブジェクト／タスク関連付けの見落とし）によるものがあります。解決方法は、エラーを見つけてビルドを行う前に修正することです。Rational Synergy は、大規模プロジェクトのコンフリクトを検出できるので、チームは、問題が大きくなる（ビルドが遅延する）前に解決できます。

Rational Synergy は、すべての履歴関係とタスク関係の知識を使用してこれらのコンフリクトを検出します。全体では 24 種類のコンフリクトを検出することが可能であり、16 種類がデフォルトで表示されます。残りの 8 つは重大ではないので、デフォルトでは表示されません。余分なコンフリクトを表示すると、実際にソフトウェアの信頼性に影響するコンフリクトに集中することが難しくなります。デフォルトで表示されるコンフリクトの種類を変更する場合は、conflict_parameters を参照してください。これらのデフォルトの設定を変更するには、*ccm_admin* ロールを持っている必要があります。

Rational Synergy は、プロジェクトを分析してコンフリクトがあるか調べ、その後コンフリクトを表示します。コマンドラインからプロジェクト階層のコンフリクトを表示することもできますが、この操作は、階層ではなく単一のプロジェクトで利用できます（`ccm conflicts`）。

プロジェクトの大きさと特性によっては、コンフリクト検出に時間がかかることがあります。したがって、コンフリクト表示の最適なタイミングを知り得るのは、担当者のみでしょう。*prep*（準備）プロジェクトの更新後、毎回ビルド マネージャはコンフリクトを表示してください。開発者は、プロジェクトに問題を起こすパラレル バージョンや他のコンフリクトが含まれる疑いがなければ、コンフリクトを表示する必要がないかもしれません。

# 日付形式

以下のトピックでは、Rational Synergy のダイアログ、コマンド、レポートに表示される日時形式、および種々の関数へ日付／時刻入力形式の設定と操作について説明します。

- Rational Synergy が表示する日付
- Rational Synergy で入力可能な日付
- 環境変数の設定
- 一般情報

## Rational Synergy が表示する日付

Rational Synergy は、一部の Rational Synergy ダイアログと `ccm` コマンドに日付を表示します。デフォルトでは、日付は "Fri Oct 30 12:04:38 1998" または "Mittwoch, 6. Mai 1998, 17:13:20 Uhr" など、地域固有の形式で表示されます。

以下に、出力時の日付変換ルールを示します。

1. 環境変数 `CCM_DATETIME_FMT` を設定すると、Rational Synergy はその変数に指定された形式で `strftime()` を使用します。ただし、ほとんど（すべてではない）の日付／時刻変換はエンジンで行われるので、全体で形式を有効にするには、インターフェイスとエンジン プロセスの両方で環境変数を設定する必要があります。
2. 環境変数を設定せずに UNIX でプロセスを実行すると、`strftime()` は `%c` 形式で使用されます。これには Windows クライアントの UNIX エンジンも含まれます。この形式は、上記の地域固有の日付／時刻形式を作成します。
3. 環境変数を設定せずに Windows でプロセスを実行すると、コントロール パネルから開いた［地域と言語のオプション］ダイアログの［地域オプション］タブで、［長い形式］の日付形式と Windows で定義された時刻形式が使用されます。このタブには日付形式を設定するオプションはありますが、時刻形式を設定するオプションはありません。

## Rational Synergy で入力可能な日付

日付が入力可能なのは、クエリで時刻（"xxxxx"）演算子を使用する場合、`ccm attr` コマンドを使用して時刻値の属性を設定する場合、`ccm clean_cache` など特定のコマンドの場合です。

一部の UNIX クライアントでは、地域固有形式での入力時に日付を使用できます。地域固有形式の日付／時刻は、ほかの UNIX クライアントや Windows では入力できません。これらのシステムでは、日付と時刻は地域固有形式で表示されることがありますが、米国方式の順序で数値を入力する必要があります。地域固有形式の入力に対応するシステムはリリースによって異なっており、Rational Synergy README にリストがあります。環

境変数 CCM_NO_LOCALE_TIMES の設定により、地域固有形式での日付入力をすべてのシステムで無効にできます。

対応するシステムでは、地域固有の日付入力は、ルーチン strptime() により %c 形式で処理されます。Rational Synergy では、他に %c、%Ec、%c %Z、%x %X、%X %x、%x %X %Z、%X %x %Z、%x などの形式、Solaris 以外では %X も対処中です。

CCM_NO_LOCALE_TIMES を設定した場合、あるいはシステムが strptime() に対応していない場合、Rational Synergy は独自の日付／時刻変換を行います。Rational Synergy はいくつかの形式について対処中で、その多くは米国仕様に特有の言語、句読点、順序を使用するものです。

あらゆるシステムの入力で使用可能とすべき形式は、地域固有か否かにかかわらず、YYYY/MM/DD hh:mm:ss です。あるいは、相対的な日付／時刻形式を使用します。クエリ、ccm clean_cache などのコマンドでは相対的な日付／時刻形式を使用してください。

相対的な日付／時刻は以下の形式で指定します。

```
-d:h:m:s
+d:h:m:s
```

相対的な日付／時刻は、当日の午前 0 時からの相対値として解釈されます。以下に、日付／時刻形式のいくつかの例を示します。

| | |
|---|---|
| 'Thu May 2 12:15' | |
| 'April 10 1998' | (時刻はデフォルトで当日の午前 0 時) |
| '2001/08/21 11:36:15' | (2001 年 8 月 21 日の午前 11 時 36 分 15 秒) |
| '-0:0:0:0' | (当日の午前 0 時) |
| '+0:0:0:0' | (当日の 24 時) |
| '-1:18:0:0' | (一昨日の午前 6 時) |
| '-2:0:0:0' | (一昨日の午前 0 時) |
| '-2:8:0:0' | (3 日前の午後 4 時) |

## 環境変数の設定

ccm start を呼び出す前に変数を設定することによりクライアント(インターフェイス プロセス）の環境変数を設定できます。

Windows のエンジン用の環境変数を設定するには、ccm.ini ファイルに [Engine environment variables] セクションを追加し、そのセクションに必要な変数と値を指定します。

UNIX のエンジン用の環境変数を設定するには、スクリプト $CCM_HOME/bin/util/ccm_engine を編集します。

また、Rational Synergy コマンドの ccm env を使用して、インターフェイスまたはエンジンの環境変数を動的に設定することもできます。ccm env コマンドはデフォルトではイン

ストールされていませんが、Rational Synergy インストール エリアの
`extras¥contrib`（Windows）または `extras/contrib`（UNIX）ディレクトリにあります。

### 一般情報

使用可能な日付は 1970 年 1 月 1 日（GMT）の午前 0 時の 1 秒後から 2037 年の終わりまでです。たとえば、アメリカの太平洋標準時タイムゾーンでは、使用可能な最も早い時刻は 1969 年 12 月 31 日の午後 4 時から 1 秒後です。

> 注記：Rational Synergy が表示する日付が正確に同じ時刻に逆変換できるとは限りません。基本の OS コードが（`CCM_NO_LOCALE_TIMES` を使用している場合でも）常に秒単位まで正確であるとは限らず、タイムゾーン間の変換、夏時間と標準時間の変換が多くのシステムで不安定であることは知られています。

あるリリースのオペレーティングシステムでは、`strptime()` を使用できますが正しく機能しません。そのような場合は（たとえば、HP-UX 11.0）、`CCM_NO_LOCALE_TIMES` 環境変数を任意の値に設定してください。

## ISO 8601 形式を使用した日付形式

クエリで使用する日付／時刻文字列は、以下のパラメータを用いて "2006-08-21 T09:12:15-0100" の ISO 8601 形式で記述できます。

- 年は 4 桁の数字で指定する。
- 月、日、時、分、秒フィールドが存在しており、正確に 2 桁構成であること、必要に応じて 0 を付加していること。
- 年、月および日フィールドは 1 つのダッシュ（-）で区切る。
- 日付と時刻は 1 つの大文字の T で区切る。
- 複数の時刻フィールドは 1 つのコロン（:）で区切る。
- タイム ゾーン フィールドがあること。UTC（GMT）を表す 1 つの大文字 Z、または 1 つのプラス（+）またはマイナス（-）記号に続けて、UTC からずれを示す。
- UTC からのずれは時（2 桁）と分（2 桁）で、時と分の間はオプション区切り文字のコロン（:）を使用できます。したがって、-0500 と -05:00 は、どちらも有効なタイムゾーン値です。

# マージ ツールの定義

再定義または変更ができるのは対話型のマージと比較ツールだけです。以下に手順を示します。

- 前提条件
- コマンドのキーワード
- マージと比較コマンドを定義する場所

## 前提条件

新しいマージ ツールは以下の前提条件を満たす必要があります。

- 自分のシステムにインストールすること。
- マージ コマンドが "ancestor"（複数バージョンとなる直前の共通ファイル）、*file1*、*file2*（マージされるファイル）の 3 つの引数を使用可能であること。
- コマンドが他のファイルへ保存するための "save" 機能を持っていること。

対話型のマージ ツールがファイル出力オプションを持たない場合、出力を Rational Synergy が指定する場所に保存する必要があります。マージ中は、メッセージが Merge ダイアログとメッセージ ビューに表示され、Rational Synergy がファイルを保存する場所を示します。デフォルトでは、ファイルは `%file1`（指定された最初のファイル名）としてシステムのデフォルト一時ディレクトリに保存されます。たとえば、`%file1` が `foo.c-32` で、一時ディレクトリが `c:¥temp`（`Windows`）または `/tmp`（`UNIX`）である場合、Rational Synergy はマージ結果を `c:¥temp¥foo.c`（`Windows`）または `/tmp/foo.c (UNIX)` に保存します。Rational Synergy は保存後、保存したファイルの内容を新規の管理ファイルにコピーします。

指定された名前のファイルが保存ディレクトリに存在する場合、Rational Synergy はファイル名 `merge_#.out` を作成します。ただし、"#" は整数で、1 ファイルを保存するたびに 1 ずつ増加します。

## コマンドのキーワード

対話型のマージと比較ツールを定義するときは、下表に示すキーワードを使用します。これらのキーワードは、マージまたは比較時に展開されます。キーワードは記載されているとおりに正確に入力してください。

| キーワード | 説明 |
|---|---|
| %outfile | マージ出力が格納される出力ファイル |
| %ancestor | 共通祖先ファイル（存在する場合） |
| %file1 | マージ用に選択した最初のファイル |
| %file2 | マージ用に選択した 2 番目のファイル |
| %file1_label | 最初のファイルの名前とバージョン |
| %file2_label | 2 番目のファイルの名前とバージョン |

## マージと比較コマンドを定義する場所

以下の変更を行うことにより、異なる対話型マージ ツールを使用できます。

- オブジェクトのタイプ定義

  typedef コマンド を使用して、特定オブジェクト タイプの merge_cmd 属性を変更できます。

  タイプの変更は、そのタイプのオブジェクトにのみ影響します。

# マイグレーション ルール

マイグレーション ルールは、マイグレーション操作を実行するときに、ファイル特性を使用してファイル システム内のファイルを Rational Synergy データベースに対応付けます。Rational Synergy は、作成、リコンサイルおよびビルド操作を行うときもマイグレーション ルールを使用します。以下の該当する項目をお読みください。

- マイグレーション - Windows オペレーティング システム
- マイグレーション - UNIX オペレーティング システム

## マイグレーション - **Windows** オペレーティング システム

以下のセクションで、マイグレーション ルールの詳細について説明しています。

- マイグレーション ルール ファイル
- 自動生成されるルール
- ルールの構文と意味
- ルール ファイルの例
- バイナリ アーカイブ ファイルのマイグレーション
- マイグレーション ルールのトラブルシューティング

### マイグレーション ルール ファイル

任意の ASCII ファイルにマイグレーション ルールを定義できます。デフォルト ルールは *database_path*¥lib¥Windows¥migrate.rul ファイルにあります。

代替マイグレーション ルールファイルに任意の名前を付けて、以下のいずれかの方法で使用します。

- ルール ファイルを作成し、以下の構文でシステムまたは個人用の ccm.ini ファイルに名前を入力する。

```
migrate.options.rules_file: alternative_rules_filename
```

- ルール ファイルを作成し、Synergy Classic の Migrate Options ダイアログを使用してルール ファイルをロードする。
- ルール ファイルを作成し、コマンド ラインで以下の構文を使用してルール ファイルをロードする。

```
ccm migrate -rules alternative_rules_filename
```

複数の代替マイグレーション ルール ファイルを定義すると、ファイルの優先度は以下のようになります（最高から最低）。

1. Synergy Classic の Migrate Options ダイアログまたは ccm migrate コマンド オプションで指定されたファイル

2. 個人用の ccm.ini ファイルに指定されたファイル
3. システムの ccm.ini ファイルに指定されたファイル
4. デフォルト ルール

## 自動生成されるルール

デフォルトで、default migrate rules ファイルに INCLUDE_AUTO_RULES ディレクティブが入っています。このディレクティブには、実行中のクライアント タイプのタイプ定義で定義された自動生成ルールが含まれています。たとえば、Windows クライアントで接尾辞 .doc と .dot に一致する 2 つのパターンを持つ msword タイプを定義したとします。INCLUDE_AUTO_RULES ディレクティブが指定された場所で、自動生成されたルールには以下の 2 つのルールが含まれます。

```
MAP_FILE_TO_TYPE .*[Dd][Oo][Cc]$ msword
MAP_FILE_TO_TYPE .*[Dd][Oo][Tt]$ msword
```

自動生成ルールを使用する利点は、タイプ ベースのルールが該当するタイプ定義で定義されることです。さらに、タイプ定義があるデータベースからエクスポートされ、別のデータベースにインポートされた場合、該当するマイグレーション ルールも共にマイグレートします。これは特に、DCM を使用していて、複数のデータベース間でタイプ定義を複製する場合に便利です。

マイグレーション ルール ファイルはマイグレーション ルールの順序を完全に管理します。必要なら、INCLUDE_AUTO_RULES ディレクティブを削除して、マイグレーション ルール ファイルで明示的に各ルールを定義できます。また、ファイル内のディレクティブの場所を移動して、自動生成されたルールが含まれるポイントを変更することもできます。

自動生成ルールは、以下のプロセスで作成されます。

1. 定義済みの ascii タイプおよび binary タイプからの自動生成ルールがまず作成されます。
2. 次に、スーパー タイプとして ascii または binary を使用する各タイプが処理され、これによりタイプの継承ツリーで上から下に、また左から右に移動できるようになります。つまり、あるスーパー タイプの子である特殊タイプが、その親であるスーパー タイプの後にルールを生成します。
3. タイプのファイル照合リスト内のコメント以外の各入力について、以下の形式のルールが生成されます。

   MAP_FILE_TO_TYPE *regular_expression type*

4. Ignore on Migrate が TRUE に設定されている各タイプについて、以下の形式のルールが生成されます。

   MAP_TYPE_TO_IGNORE *type* TRUE

INCLUDE_AUTO_RULES ディレクティブによってインクルードされる自動生成ルールは、以下の手順で表示できます。

CLI から、`ccm show -mar` コマンドを使用します。

## ルールの構文と意味

マイグレーション ルール ファイル内の各ルールは、指定された特性を持つファイルに対して、属性を設定するか、または操作を実行します。

ルールの構文は以下のようになります。

*`mapping operand_1 operand_2`*

- *`mapping`* は実行されるマッピング動作です。

  マッピング動作はオブジェクト属性を設定するか、ignore または collapse 操作を行います。

- *`operand1`* はファイル特性です。

  ファイル特性は、マッピング動作に応じて正規表現、文字列、マクロとなります。

- *`operand2`* はマッピング動作で使用する値です。

  値は、マッピング動作の ignore と collapse については `TRUE` または `FALSE`、他のマッピング動作については属性値です。

複数ルールは、ルール ファイル内の順序で適用されます。コメントの前にはポンド記号を付けます。

> 注記：使用する Windows オペレーティング システムに応じて、ファイル名で大文字、小文字、またはその組み合わせが使用できます。

たとえば、以下のマイグレーション ルールでは、名前の最後が「`.c`」であるファイルについて、オブジェクト タイプを「`csrc`」、バージョンを「`1.0`」に設定します。

```
# C source files
MAP_FILE_TO_TYPE .*¥.c$ csrc
MAP_TYPE_TO_VERSION csrc 1.0
```

> 注記：パス内の円記号（¥）区切り文字は、マイグレーション ルールの適用時にスラッシュ（/）区切り文字に変換されます。したがって、マイグレーション ルールには標準の正規表現のエスケープ（¥）を含めることができます。

下表に、使用可能なマイグレーション ルールとオペランドを示します。

| マッピング | Operand 1 | Operand 2 |
|---|---|---|
| **to TYPE** | | |
| MAP_FILE_TO_TYPE | 文字列または正規表現 | 有効なタイプ |
| MAP_TYPE_TO_TYPE | 文字列または正規表現 | 有効なタイプ |
| MAP_MODE_TO_TYPE | 値またはマクロ | 有効なタイプ |
| **to VERSION** | | |
| MAP_FILE_TO_VERSION | 文字列または正規表現 | 有効なバージョン |
| MAP_TYPE_TO_VERSION | 文字列または正規表現 | 有効なバージョン |
| MAP_MODE_TO_VERSION | 値またはマクロ | 有効なバージョン |
| **to IGNORE** | | |
| MAP_FILE_TO_IGNORE | 文字列または正規表現 | TRUE または FALSE |
| MAP_TYPE_TO_IGNORE | 文字列または正規表現 | TRUE または FALSE |
| MAP_MODE_TO_IGNORE | 値またはマクロ | TRUE または FALSE |
| **to COLLAPSE** | | |
| MAP_FILE_TO_COLLAPSE | 文字列または正規表現 | TRUE or FALSE |
| MAP_TYPE_TO_COLLAPSE | 文字列または正規表現 | TRUE または FALSE |
| MAP_MODE_TO_COLLAPSE | 値またはマクロ | TRUE または FALSE |
| **to ATTRIBUTE** | | |
| MAP_FILE_TO_ATTRIBUTE | 文字列または正規表現 | TRUE または FALSE |
| MAP_KEY_TO_ATTR | 文字列 | 文字列 |

以下のセクションでは、ルールの使用方法について説明します。

- Map to TYPE
- Map to VERSION
- Map to IGNORE
- Map to COLLAPSE

- Map to ATTRIBUTE

## Map to TYPE

MAP_*_TO_TYPE ルールは type 属性を設定します。通常は、type 属性は以降のルールで使用できるため、これがファイルのために入力する最初のルールです。

オブジェクト タイプが %expand_pvcs の場合、migrate コマンドは一致ファイルを PVCS アーカイブとして扱い、マイグレーション操作でファイルから差分を抽出します。差分はオブジェクトになり、プレビューでオブジェクトのリストに表示されます。

> 注記：MAP_MODE_TO_* ルールでは、*operand_1*: %dir に 1 つの定義済みマクロを使用できます。

例

- 拡張子 .c を持つすべてのファイルについて、Rational オブジェクトの type 属性を csrc に設定する。

  ```
  MAP_FILE_TO_TYPE .*¥.c$ csrc
  ```

- 以下のルールのコメントを外すことにより、すべての PVCS アーカイブ ファイルを展開する。

  ```
  MAP_FILE_TO_TYPE   .*¥...[Vv]$ %expand_pvcs
  ```

- すべてのディレクトリについて、Rational オブジェクトの type 属性を dir に設定する。

  ```
  MAP_MODE_TO_TYPE %dir dir
  ```

## Map to VERSION

MAP_*_TO_VERSION ルールは version 属性を設定します。

> 注記：MAP_MODE_TO_* ルールでは、*operand_1*: %dir に 1 つの定義済みマクロを使用できます。

例

- 拡張子 .c を持つすべてのファイルについて、Rational オブジェクトの version 属性を 2 に設定します。

  ```
  MAP_FILE_TO_VERSION .*¥.c$ 2
  ```

> 注記：このオブジェクトのインスタンスが既にバージョン 2 で存在する場合、その次の適切なバージョンが使用されます。

## Map to IGNORE

MAP_*_TO_IGNORE ルールは、TRUE に設定されると、ファイルを無視します。

> 注記：ディレクトリに無視のマークが付くと、ディレクトリ階層内のすべてのファイルが無視されます。

例

- ファイル拡張子 .map を持つファイルを無視します。

  ```
  MAP_FILE_TO_IGNORE .*¥.map$ TRUE
  ```

- タイプが makefile のファイルを無視します。

  ```
  MAP_TYPE_TO_IGNORE makefile TRUE
  ```

- PVCS　ファイルを認識します（コメントアウトされたデフォルトマイグレーションルールのコメントを外すことにより）。

  これらのルールは通常、接尾辞が V または v で終わるファイル（.wav ファイルなど）が無視されないようにコメントアウトされます。2 番目のルールにより、展開されていない PVCS ファイルは無視されます。

  ```
  MAP_FILE_TO_TYPE   .*¥...[Vv]$ %expand_pvcs
  MAP_FILE_TO_IGNORE .*¥...[Vv]$ TRUE
  ```

## Map to COLLAPSE

MAP_*_TO_COLLAPSE ルールは、TRUE に設定されると、ディレクトリをマイグレートする前に破棄します。

一致するディレクトリのすべての子は、その親ディレクトリの子になります。この方法は一般に、アーカイブ ファイルをそのアーカイブ ディレクトリからチェックアウトされたバージョンのレベルまで上げるために使用します。

> 注意！ マイグレーションの場合、すべての COLLAPSE および %expand ルールは、プレビューの間だけ評価され、ルールが編集されると再評価されません。

例

- tmp ディレクトリを破棄します。

  ```
  MAP_FILE_TO_COLLAPSE .*/tmp$ TRUE
  ```

## Map to ATTRIBUTE

MAP_*_TO_ATTRIBUTE ルールは、指定された属性を指定値に設定します。

> 注記：テキスト属性のみを Rational Synergy 属性にマップできます。

例

- SYNERGY オブジェクトの `reviewer` 属性をファイルの PVCS 拡張リビジョン属性 `reviewed_by` に設定します。

  ```
  MAP_KEY_TO_ATTR reviewed_by reviewer
  ```

  注記：アーカイブ属性がオブジェクトに存在しない場合、Rational Synergy が属性をタイプ `text` として作成します。

## ルール ファイルの例

以下のマイグレーション ルール ファイルを使用するとします（左の数字は参照用です）。

1. `MAP_FILE_TO_TYPE .*¥.mk$ makefile`
2. `MAP_FILE_TO_TYPE [Mm]akefile[^/]*$ makefile`
3. `MAP_TYPE_TO_IGNORE makefile TRUE`
4. `MAP_FILE_TO_TYPE .*¥...[Vv]$ %expand_pcvs`
5. `MAP_FILE_TO_IGNORE .*¥...[Vv]$ TRUE`
6. `MAP_MODE_TO_TYPE %dir dir`

次に、以下のファイルを Rational Synergy データベースにマイグレートします。

```
dir1                <dir>
Makefile.joe
Makefile.jov
```

最初のファイル `dir1` はディレクトリです。このファイルに適用されるルールは以下のとおりです。

**6.** `MAP_MODE_TO_TYPE %dir dir`

2 番目のファイル `Makefile.joe` は make ファイルです。このファイルに適用されるルールは以下のとおりです。

**2.** `MAP_FILE_TO_TYPE [Mm]akefile[^/]*$ makefile`
**3.** `MAP_TYPE_TO_IGNORE makefile TRUE`

パターン "`[Mm]akefile[^/]*$`" が make ファイルのファイル名 `Makefile.joe` と一致するので、ルール 2 が適用されます。ファイル オブジェクトにタイプ `makefile` が与えられているので、ルール 3 が適用されます。その結果、タイプ `makefile` のファイルが無視されます。次の新規ルールを作成しても、同じ結果が得られます。

```
MAP_FILE_TO_IGNORE [Mm]akefile[^/]*$
```

3 番目のファイル `Makefile.jov` は PVCS アーカイブです。このファイルに適用されるルールは次のとおりです。

**4.** `MAP_FILE_TO_TYPE .*¥...[Vv]$ %expand_pvcs`
**5.** `MAP_FILE_TO_IGNORE .*\¥...[Vv]$ TRUE`

ルール 4 は、"*¥...[Vv]$" パターンがファイル名 Makefile.jov と一致するので適用されます。このルールは、マイグレーション操作でアーカイブ ファイル内のすべての差分を抽出し、プレビュー リストに個々の差分のオブジェクトを作成します。ルール 5 は、同じパターン マッチを行い、アーカイブ ファイルに **ignored** のマークを付けます。その結果、展開された PVCS ファイルについて新規オブジェクト バージョンが作成されますが、アーカイブ ファイル自身は無視されます。また、ルールはアーカイブ ファイルから展開された各ファイルに適用されるので、ルール **3** にしたがい、各 Makefile.jov ファイルにタイプ makefile が割り当てられます。

> 注記：マイグレーション ルールは、本番ファイルに適用する前に、必ずテスト ファイルでテストしてください。
>
> たとえば、MAP_FILE_TO_* ルールは、パターンをファイル名と比較します。ファイル名にはパス全体が含まれていて、パターン "makefile" がパスに入っているため、MAP_FILE_TO_TYPE make ファイル ルールは C:¥my_makefiles¥foo.c などのディレクトリをタイプ makefile にマップします。そのため、ルール MAP_FILE_TO_TYPE /makefile$ makefile（ファイル名の前とパスの最後にスラッシュを使用）は、ルールをディレクトリではなくファイルに確実に適用させたい場合に適しています。

## バイナリ アーカイブ ファイルのマイグレーション

アーカイブ ファイルにタイプを割り当てるとき、マイグレーション操作はまずアーカイブ ファイルからファイルを抽出してから、作成されたファイル名に対してマイグレーション ルールを適用します。作成されたファイル名が既存のどのタイプとも一致しない場合は、ascii が自動的に割り当てられます。アーカイブの内容が実際に ascii でない場合、マイグレーション操作でファイルをロードできません。

マイグレーション操作でアーカイブ ファイルを ascii としてロードさせない場合は、以下のいずれかの操作を行います。

- バイナリ アーカイブ ファイル名を、バイナリなどの非 ascii タイプにマッピングするルールを追加する（一般的な方法）。
- Preview Results リストから選択した項目について、バイナリ アーカイブ ファイルを非 ascii タイプに手動でマップする。
- タイプが正しいことを確認するために、ロードを実行する前にプレビュー結果をチェックする。

## マイグレーション ルールのトラブルシューティング

入力したルールに効果がない場合は、以下の操作を行います。

1. 変更されたルールが保存され、編集ルール セッションが終了したことを確認します。
2. ルールが、ルール リスト内の後続のルールにより無効とされていないことを確認します。
3. 他のルールが再度適用された場合、`%expand_pvcs` を使用する `MAP_*_TO_TYPE` が、編集後ではなく、プレビュー中 に適用されます。
4. `MAP_FILE_TO_*` ルールがフル ファイル パスを照合します。たとえば、`bitmap` という名前のディレクトリが存在していて、ルールが `MAP_FILE_TO_TYPE bitmap binary` の場合、ディレクトリとその階層内のすべてがタイプ `binary` に設定されます。これを回避するには、最初のオペランドを `/bitmap$` に設定します。

   たとえば、`bitmap` ファイルをタイプ `binary` として、また `bitmap` ディレクトリをタイプ `dir` としてマイグレートするには、以下のルールを使用します。

   ```
   MAP_FILE_TO_TYPE /bitmap$ binary
   MAP_MODE_TO_TYPE $dir dir
   ```

### マイグレーション - UNIX オペレーティング システム

以下のセクションで、マイグレーション ルールの詳細について説明しています。

- マイグレーション ルール ファイル
- 自動生成されるルール
- ルールの構文と意味
- ルール ファイルの例
- バイナリ アーカイブ ファイルのマイグレーション
- マイグレーション ルールのトラブルシューティング

### マイグレーション ルール ファイル

任意の ASCII ファイルにマイグレーション ルールを定義できます。デフォルト ルールは *`database_path`*`/lib/UNIX/migrate.rul` ファイルにあります。

代替マイグレーション ルールファイルに任意の名前を付けて、以下のいずれかの方法で使用します。

- ルール ファイルを作成し、以下の構文でシステムまたは個人用の `ccm.ini` ファイルに名前を入力する。

  `migrate.options.rules_file:` *`alternative_rules_filename`*

- ルール ファイルを作成し、Migrate Options ダイアログを使用してルール ファイルをロードする。
- ルール ファイルを作成し、コマンド ラインで以下の構文を使用してルール ファイルをロードする。

  `ccm migrate rules` *`alternative_rules_filename`*

複数の代替マイグレーション ルール ファイルを定義すると、ファイルの優先度は以下のようになります（最高から最低）。

1. Migrate Options ダイアログまたは `ccm migrate` コマンド オプションに指定されたファイル
2. 個人用の `.ccm.ini` ファイルに指定されたファイル
3. システムの `.ccm.ini` ファイルに指定されたファイル
4. デフォルト ルール

## 自動生成されるルール

デフォルトで、`default migrate rules` ファイルに `INCLUDE_AUTO_RULES` ディレクティブが入っています。このディレクティブには、実行中のクライアント タイプのタイプ定義で定義された自動生成ルールが含まれています。たとえば、Windows クライアントで接尾辞 `.doc` と `.dot` に一致する 2 つのパターンを持つ `msword` タイプを定義したとします。`INCLUDE_AUTO_RULES` ディレクティブが指定された場所で、自動生成されたルールには以下の 2 つのルールが含まれます。

```
MAP_FILE_TO_TYPE .*[Dd][Oo][Cc]$ msword
MAP_FILE_TO_TYPE .*[Dd][Oo][Tt]$ msword
```

自動生成ルールを使用する利点は、タイプ ベースのルールが該当するタイプ定義で定義されることです。さらに、タイプ定義があるデータベースからエクスポートされ、別のデータベースにインポートされた場合、該当するマイグレーション ルールも共にマイグレートします。これは特に、DCM を使用していて、複数のデータベース間でタイプ定義を複製する場合に便利です。

マイグレーション ルール ファイルはマイグレーション ルールの順序を完全に管理します。必要なら、`INCLUDE_AUTO_RULES` ディレクティブを削除して、マイグレーション ルール ファイルで明示的に各ルールを定義できます。また、ファイル内のディレクティブの場所を移動して、自動生成されたルールが含まれるポイントを変更することもできます。

自動生成ルールは、以下のプロセスで作成されます。

1. 定義済みの `ascii` タイプおよび `binary` タイプからの自動生成ルールがまず作成されます。
2. 次に、スーパー タイプとして `ascii` または `binary` を使用する各タイプが処理され、これによりタイプの継承ツリーで上から下に、また左から右に移動できるようになり

ます。つまり、あるスーパー タイプの子である特殊タイプが、その親であるスーパー タイプの後にルールを生成します。

3. タイプのファイル照合リスト内のコメント以外の各入力について、以下の形式のルールが生成されます。

   `MAP_FILE_TO_TYPE` *`regular_expression type`*

4. Ignore on Migrate が `TRUE` に設定されている各タイプについて、以下の形式のルールが生成されます。

   `MAP_TYPE_TO_IGNORE` *`type`* `TRUE`

`INCLUDE_AUTO_RULES` ディレクティブによってインクルードされる自動生成ルールは、以下の手順で表示できます。

CLI から、`ccm show -mar` コマンドを使用します。

## ルールの構文と意味

マイグレーション ルール ファイル内の各ルールは、指定された特性を持つファイルに対して、属性を設定するか、または操作を実行します。

ルールの構文は以下のようになります。

*`mapping operand_1 operand_2`*

- *`mapping`* は実行されるマッピング動作です。

  マッピング動作はオブジェクト属性を設定するか、ignore または collapse 操作を行います。

- *`operand1`* はファイル特性です。

  ファイル特性は、マッピング動作に応じて正規表現、文字列、マクロとなります。

- *`operand2`* はマッピング動作で使用する値です。

  値は、マッピング動作の ignore と collapse については `TRUE` または `FALSE`、他のマッピング動作については属性値です。

複数ルールは、ルール ファイル内の順序で適用されます。コメントの前にはポンド記号を付けます。

たとえば、以下のマイグレーション ルールでは、名前の最後が「`.c`」であるファイルについて、オブジェクト タイプを「`csrc`」、バージョンを「`1.0`」に設定します。

```
# C source files
MAP_FILE_TO_TYPE .*¥.c$ csrc
MAP_TYPE_TO_VERSION csrc 1.0
```

下表に、使用可能なマイグレーション ルールとオペランドを示します。

| マッピング | Operand 1 | Operand 2 |
|---|---|---|
| **to TYPE** | | |
| MAP_FILE_TO_TYPE | 文字列または正規表現 | 有効なタイプ |
| MAP_TYPE_TO_TYPE | 文字列または正規表現 | 有効なタイプ |
| MAP_MODE_TO_TYPE | 値またはマクロ | 有効なタイプ |
| **to VERSION** | | |
| MAP_FILE_TO_VERSION | 文字列または正規表現 | 有効なバージョン |
| MAP_TYPE_TO_VERSION | 文字列または正規表現 | 有効なバージョン |
| MAP_MODE_TO_VERSION | 値またはマクロ | 有効なバージョン |
| **to IGNORE** | | |
| MAP_FILE_TO_IGNORE | 文字列または正規表現 | TRUE または FALSE |
| MAP_TYPE_TO_IGNORE | 文字列または正規表現 | TRUE または FALSE |
| MAP_MODE_TO_IGNORE | 値またはマクロ | TRUE または FALSE |
| **to COLLAPSE** | | |
| MAP_FILE_TO_COLLAPSE | 文字列または正規表現 | TRUE または FALSE |
| MAP_TYPE_TO_COLLAPSE | 文字列または正規表現 | TRUE または FALSE |
| MAP_MODE_TO_COLLAPSE | 値またはマクロ | TRUE または FALSE |
| **to STATUS** | | |
| MAP_STATE_TO_STATUS | 文字列 | 有効な値 |
| **to ATTRIBUTE** | | |
| MAP_FILE_TO_ATTRIBUTE | 文字列または正規表現 | TRUE または FALSE |
| MAP_KEY_TO_ATTR | symbols （小文字の文字列） | 文字列 |

以下のセクションでは、ルールの使用方法について説明します。

- Map to TYPE

- Map to VERSION
- Map to IGNORE
- Map to COLLAPSE
- Map to STATUS
- Map to ATTRIBUTE

## Map to TYPE

MAP_*_TO_TYPE ルールは type 属性を設定します。通常は、type 属性は後続のルールで使用できるため、これがファイルのために入力する最初のルールです。

オブジェクト タイプが %expand_rcs または %expand_sccs の場合、migrate コマンンドトは一致ファイルを RCS または SCCS アーカイブとして扱い、マイグレーション操作でファイルから差分を抽出します。差分はオブジェクトになり、プレビューでオブジェクトのリストに表示されます。

> 注記：MAP_MODE_TO_* ルールでは、その最初のパラメータとして正の整数、8 進数、定義済みのマクロを使用できます。正の整数または 8 進数はファイル モードを示します。8 進数はゼロで始めます。たとえば、rwxr--r-- 権限を持つファイルは 8 進数 0100744 と一致します。ビット パターンの 0100000 は通常ファイルのコードです。
>
> 定義済みマクロ %dir、%link、%exec は、整数や 8 進数ほど厳密ではありません。 MAP_MODE_TO_TYPE 0100111 executable などのルールが権限 ----x--x--x を持つファイルとのみ一致するのに対し、ルール MAP_MODE_TO_TYPE %exec executable は、実行可能な権限（例、-rwx--x--x）を設定したファイルと一致します。

例

- 拡張子 .c を持つすべてのファイルについて、Rational オブジェクトの type 属性を csrc に設定する。

  ```
  MAP_FILE_TO_TYPE .*¥.c$ csrc
  ```

- モードが executable であるすべてのファイルについて、オブジェクトの type 属性を executable に設定する（ファイルのいずれかの権限 — ユーザー、グループ、その他 — が実行可能の場合、%exec は TRUE として評価されます）。

  ```
  MAP_MODE_TO_TYPE %exec executable
  ```

- すべてのディレクトリについて、Rational オブジェクトの type 属性を dir に設定する。

  ```
  MAP_MODE_TO_TYPE %dir dir
  ```

## Map to VERSION

MAP_*_TO_VERSION ルールは version 属性を設定します。

> 注記：MAP_MODE_TO_* ルールでは、その最初のパラメータとして正の整数、8 進数、定義済みのマクロを使用できます。正の整数または 8 進数はファイル モードを示します。8 進数はゼロで始めます。たとえば、rwxr--r-- 権限を持つファイルは 8 進数 0100744 と一致します。ビット パターンの 0100000 は通常ファイルのコードです。
>
> 定義済みマクロ %dir、%link、%exec は、整数や 8 進数ほど厳密ではありません。 MAP_MODE_TO_TYPE 0100111 executable などのルールが権限 ----x--x--x を持つファイルとのみ一致するのに対し、ルール MAP_MODE_TO_TYPE %exec executable は、実行可能な権限（たとえば、-rwx--x--x）を設定したファイルと一致します。

例

- 拡張子 .c を持つすべてのファイルについて、Rational オブジェクトの version 属性を 2 に設定する。

  ```
  MAP_FILE_TO_VERSION .*¥.c$ 2
  ```

  > 注記：このオブジェクトのインスタンスが既にバージョン 2 で存在する場合、その次の適切なバージョンが使用されます。

- タイプ link のすべてのファイルについて、version 属性を ver1 に設定する。

  ```
  MAP_MODE_TO_VERSION %link link
  ```

- ユーザー読み取りまたは書き込み専用のすべてのファイルについて、version 属性を pre1 に設定する。

  ```
  MAP_MODE_TO_VERSION 600 pre1
  ```

## Map to IGNORE

MAP_*_TO_IGNORE ルールは、TRUE に設定されると、ファイルを無視します。

> 注記：ディレクトリに無視のマークが付くと、ティレクトリ階層内のすべてのファイルが無視されます。

例

- tmp ディレクトリ内のすべてのファイルを無視する。

  ```
  MAP_FILE_TO_IGNORE /tmp/ TRUE
  ```

- ファイル拡張子 SCCS を持つファイルを無視する。

  ```
  MAP_FILE_TO_IGNORE .*/s\.[^/]+$ TRUE
  ```

- タイプが `makefile` のファイルを無視する。

  ```
  MAP_TYPE_TO_IGNORE makefile TRUE
  ```

## Map to COLLAPSE

`MAP_*_TO_COLLAPSE` ルールは、`TRUE` に設定されると、ディレクトリをマイグレートする前に破棄します。

一致するディレクトリのすべての子は、その親ディレクトリの子になります。この方法は一般に、アーカイブ ファイルをそのアーカイブ ディレクトリからチェックアウトされたバージョンのレベルまで上げるために使用します。

> 注意！ マイグレーションの場合、すべての `COLLAPSE` および `%expand` ルールは、プレビューの間だけ評価され、ルールが編集されると再評価されません。

例

- `SCCS` ディレクトリを破棄します。

  ```
  MAP_FILE_TO_COLLAPSE .*/SCCS$ TRUE
  ```

## Map to STATUS

`MAP_STATE_TO_STATUS` ルールは `status` 属性を設定します。

> 注記：このルールを使用するには *ccm_admin* ロールが必要で、さらに GUI で Use Status From Archive ON を設定するか、`migrate` コマンド ラインで `-meta_status` オプションを使用する必要があります。
>
> このマッピングは、RCS アーカイブからファイルをマイグレートする場合にのみ有効です。

例

- RCS ステータス `Exp` を持つすべてのファイルについて、オブジェクトの `status` 属性を *`released`* に設定します。

  ```
  MAP_STATE_TO_STATUS Exp released
  ```

## Map to ATTRIBUTE

`MAP_*_TO_ATTRIBUTE` ルールは、指定された属性を指定値に設定します。

注記：テキスト属性のみを Rational Synergy 属性にマップできます。

例

- オブジェクトの `release` 属性をファイルの RCS `symbols` 属性に設定する。

  ```
  MAP_KEY_TO_ATTR symbols release
  ```

  注記：アーカイブ属性がオブジェクトに存在しない場合、Rational Synergy が属性をタイプ `text` として作成します。

## ルール ファイルの例

以下のマイグレーション ルール ファイルを使用するとします（左の数字は参照用です）。

1. `MAP_MODE_TO_TYPE %exec executable`
2. `MAP_FILE_TO_TYPE .*¥.mk$ makefile`
3. `MAP_FILE_TO_TYPE [Mm]akefile[^/]*$ makefile`
4. `MAP_TYPE_TO_IGNORE makefile TRUE`
5. `MAP_FILE_TO_TYPE .*/s¥.[^/]+$ %expand_sccs`
6. `MAP_FILE_TO_IGNORE .*/s¥.[^/]+$ TRUE`
7. `MAP_FILE_TO_COLLAPSE .*/SCCS$ TRUE`
8. `MAP_MODE_TO_TYPE %dir dir`

次に、以下のファイルを Rational Synergy データベースにマイグレートします。

```
dir1              <dir>
Makefile.joe      -rw-r--r--
SCCS/             drwxr-xr-x
s.Makefile.joe    -r--r--r--
```

最初のファイル `dir1` はディレクトリです。このファイルに適用されるルールは以下のとおりです。

```
1. MAP_MODE_TO_TYPE %exec executable
8. MAP_MODE_TO_TYPE %dir dir
```

`a/` に実行可能ファイル ビットがセットされるので（"x"）、ルール **1** が適用されます。ディレクトリ ファイル ビットがセットされるので、ルール **8** が適用されます。2 番目のルールによって最初のルールが上書きされるので、作成されるファイル タイプは `dir` となります。

2 番目のファイル `Makefile.joe` は make ファイルです。このファイルに適用されるルールは以下のとおりです。

```
3. MAP_FILE_TO_TYPE [Mm]akefile[^/]*$ makefile
4. MAP_TYPE_TO_IGNORE makefile TRUE
```

パターン "[Mm]akefile[^/]*$" が make ファイルのファイル名 Makefile.joe と一致するので、ルール **3** が適用されます。ファイル オブジェクトにタイプ makefile が与えられているので、ルール **4** が適用されます。その結果、タイプ makefile のファイルが無視されます。次の新規ルールを作成しても、同じ結果が得られます。

```
MAP_FILE_TO_IGNORE [Mm]akefile[^/]*$
```

3 番目のファイル SCCS/ は SCCS ディレクトリです。このファイルに適用されるルールは以下のとおりです。

```
7. MAP_FILE_TO_COLLAPSE .*/SCCS$ TRUE
```

（ルール **8** も適用されますが、ファイルはすでにルール **7** によって縮小されています。）

パターン ".*/SCCS$" がディレクトリ名と一致するので、最初のルールが適用されます。このルールにより、SCCS ディレクトリはマイグレート せず、ディレクトリのメンバー ファイルが宛先ディレクトリにマイグレートします。

4 番目のファイル s.Makefile.joe は SCCS アーカイブです。このファイルに適用されるルールは以下のとおりです。

```
5. MAP_FILE_TO_TYPE .*/s¥.[^/]+$ %expand_sccs
6. MAP_FILE_TO_IGNORE .*/s¥.[^/]+$ TRUE
```

".*/s¥.[^/]+$" パターンがファイル名 s.Makefile.joe と一致するので、ルール **5** が適用されます。このルールは、マイグレーション操作でアーカイブ ファイル内のすべての差分を抽出し、プレビュー リストに個々の差分のオブジェクトを作成します。ルール **6** は、同じパターン マッチを行い、アーカイブ ファイルに **ignored** のマークを付けます。その結果、展開された SCCS ファイルについて新規オブジェクト バージョンが作成されますが、SCCS ファイル自身は無視されます。また、ルールはアーカイブ ファイルから展開された各ファイルに適用されるので、ルール **4** にしたがい、各 Makefile.joe ファイルにタイプ makefile が割り当てられます。

> 注記：マイグレーション ルールは、本番ファイルに適用する前に、必ずテスト ファイルでテストしてください。
>
> たとえば、MAP_FILE_TO_* ルールは、パターンをファイル名と比較します。ファイル名にはパス全体が含まれていて、パターン "makefile" がパスに入っているため、MAP_FILE_TO_TYPE make ファイル ルールは /user/mark/proj1/my_makefiles/foo.c などのディレクトリをタイプ makefile にマップします。そのため、ルール MAP_FILE_TO_TYPE /makefile$ makefile（ファイル名の前とパスの最後にスラッシュを使用）は、ルールをディレクトリではなくファイルに確実に適用させたい場合に適しています。

## バイナリ アーカイブ ファイルのマイグレーション

アーカイブ ファイルにタイプを割り当てるとき、マイグレーション操作はまずアーカイブ ファイルからファイルを抽出してから、作成されたファイル名に対してマイグレーション ルールを適用します。作成されたファイル名が既存のどのタイプとも一致しない場合は、ascii が自動的に割り当てられます。アーカイブの内容が実際に ascii でない場合、マイグレーション操作でファイルをロードできません。

マイグレーション操作でアーカイブ ファイルを ascii としてロードさせない場合は、以下のいずれかの操作を行います。

- バイナリ アーカイブ ファイル名を、バイナリなどの非 ascii タイプにマッピングするルールを追加する（一般的な方法）。
- Preview Results リストから選択した項目について、バイナリ アーカイブ ファイルを非 ascii タイプに手動でマップする。
- タイプが正しいことを確認するために、ロードを実行する前にプレビュー結果をチェックする。

## マイグレーション ルールのトラブルシューティング

入力したルールに効果がない場合は、以下の操作を行います。

1. 変更されたルールが保存され、編集ルール セッションが終了したことを確認します。
2. ルールが、ルール リスト内の後続のルールにより無効とされていないことを確認します。
3. ほかのルールが再度適用された場合、%expand_sccs と %expand_rcs を使用する MAP_*_TO_TYPE が、編集後ではなく、プレビュー中に適用されます。
4. MAP_FILE_TO_* ルールがフル ファイル パスを照合します。たとえば、core という名前のディレクトリが存在していて、ルールが MAP_FILE_TO_TYPE core binary の場合、ディレクトリとその階層内のすべてがタイプ binary に設定されます。これを回避するには、最初のオペランドを /core$ に設定します。

   たとえば、core ファイルをタイプ binary として、また core ディレクトリをタイプ dir としてマイグレートするには、以下のルールを使用します。

   ```
   MAP_FILE_TO_TYPE /core$ binary
   MAP_MODE_TO_TYPE $dir dir
   ```

# クエリ式

以下のトピックでは、Rational Synergy データベース クエリの構成方法を説明します。

- クエリのタイプ
- クエリ文節要素
- クエリの例

# クエリのタイプ

クエリ、つまり クエリ 式は、1 つ以上の クエリ文節 を持つ `ccm query` コマンドを使用して作成します。クエリ文節とは、クエリ式を構成する個々の検索条件です。

以下のセクションでは、種々のクエリ文節を用いたクエリ式の構成方法を説明します。

- 属性値文節を使用したクエリ
- 関数テスト文節を使用したクエリ
- 属性値と関数テスト文節の両方を使用したクエリ
- キーワードを使用したクエリ
- ネストされたクエリ

## 属性値文節を使用したクエリ

属性値に基づくクエリ文節は、一致する属性を持つ（または持たない）すべてのオブジェクトバージョンを検索します。

このタイプの文節の構文は、以下のように属性名（*attr_name*）、比較演算子（*relative_operator*）、および属性値（*constant*）で構成されています。

"*attr_name relative_operator 'constant'*"

例

- 状態が「*working*」のすべてのオブジェクト バージョンを検索する。

  ```
  ccm query "status='working'"
  ```

  同等の内蔵ショートカットは以下のとおりです。

  ```
  ccm query -s working
  ```

- バージョンが「2」であるすべてのオブジェクト バージョンを検索して表示する。

  ```
  ccm query "version='2'"
  ```

  同等の内蔵ショートカットは以下のとおりです。

  ```
  ccm query -v 2
  ```

## 関数テスト文節を使用したクエリ

関数テストをもとにしたクエリ文節は、関数結果に一致するすべてのオブジェクト バージョンを検索します。

このタイプの文節の構文は、以下のように関数（*function*）とその引数で構成されています。

"*function('function_arguments')*"

関数はあらかじめ定義されています。関数の説明については、関数定義を参照してください。

例

- 祖先 `ico-1:executable:2` を持つすべてのオブジェクト バージョンを検索する。

  ```
  ccm query "has_predecessor('ico-1:executable:2')"
  ccm delete @
  ```

- タイプ `wdt` のすべてのオブジェクト バージョンを検索して選択する。

  ```
  ccm query "type='wdt'"
  ```

  同等の内蔵ショートカットは以下のとおりです。

  ```
  ccm query -type wdt
  ```

## 属性値と関数テスト文節の両方を使用したクエリ

クエリ文節を組み合わせて検索を細かく指定できます。以下にクエリ文節を組み合わせる方法を示します。

"**not** *query_clause*"
"*query_clause* **and** *query_clause*"
"*query_clause* **or** *query_clause*"

例

- プロジェクトのメンバーではないすべてのオブジェクト バージョンを検索する。

  ```
  ccm query "not is_bound()"
  ```

- プロジェクトのメンバーで、2001 年 12 月 12 日より古い修正日付を持つすべてのオブジェクト バージョンを検索する。

  ```
  ccm query "is_bound() and modify_time < time('Fri Dec 12 2001')"
  ```

## キーワードを使用したクエリ

クエリ式には、日付に関する特定のキーワードを使用できます。下表に有効なキーワードを示します。

| キーワード | 説明 |
|---|---|
| `%today_begin` | 今日の 00:00:00 |
| `%today_end` | 今日の 23:59:59 |
| `%this_week_begin` | 今週の終わり 23:59:59<br>注記 1 を参照 |
| `%this_week_end` | 先週の始め 00:00:00<br>注記 1 を参照 |

| キーワード | 説明 |
|---|---|
| %last_week_begin | 先週の始め 00:00:00<br>注記 1 を参照 |
| %last_week_end | 先週の始め 23:59:59<br>注記 1 を参照 |
| %this_month_begin | 今月の始め 00:00:00 |
| %this_month_end | 今月の終わり 23:59:59 |
| %last_month_begin | 先月の始め 00:00:00 |
| %last_month_end | 先月の終わり 23:59:59 |
| %this_year_begin | 今年 1 月 1 日 00:00:00 |
| %this_year_end | 今年 12 月 31 日 23:59:59 |
| %today_minus<N>days | 今日から <N> 日前 00:00:00 |
| %today_plus<N>days | 今日から <N> 日後 00:00:00 |
| %today_minus<N>weeks | 今日から <N> 週間前 00:00:00 |
| %today_plus<N>weeks | 今日から <N> 週間後 00:00:00 |
| %today_minus<N>months | 今日から <N> 月前 00:00:00<br>注記 2 を参照 |
| %today_plus<N>months | 今日から <N> 月後 00:00:00<br>注記 2 を参照 |
| %today_minus<N>years | 今日から <N> 年前 00:00:00<br>注記 3 を参照 |
| %today_plus<N>years | 今日から <N> 年後 00:00:00<br>注記 3 を参照 |

注記 **1** : 週の始まりの日は、デフォルトでは日曜日です。モデル属性 `start_day_of_week` を設定してこのデフォルトを変更できます。値1は月曜、2は火曜、以下同様を意味します。

注記 **2** : 月を足し引きする場合、今日の日付が結果の月の日数より多い場合、有効日はその月の最終日になります。いずれの場合も時刻は 00:00:00 です。たとえば、今日が 2003 年 1 月 30 日の場合、`%today_plus1month` は、2003 年 2 月 28 日の 00:00:00 になります。

注記 **3** : 年を足し引きする場合、今日の日付が結果の年のその月の日数より多い場合、有効日はその月の最終日になります。 たとえば、今日が 2004 年 2 月 29 日の場合、`%today_plus1years` は、2005 年 2 月 28 日の 00:00:00 になります。

例

- 今日作成された `file1.c` という名のファイルをすべて表示する。

  ```
  ccm query "name=’file1.c’ and create_time > time(’%today_begin’)"
  ```

## ネストされたクエリ

ネストされたクエリとは、関数テストを使用するタイプで、引数の 1 つ以上がクエリ式であるようなクエリ式です。

クエリ関数は、通常以下の構文を使用します。

```
query_function('object_name'|
               'project_name'|
                  'type_name'|
                  'attr_name'|
            'privilege_name''
               [,sort_order])
```

評価結果が適切なタイプのオブジェクトになるクエリ式を使って、オブジェクト名引数、プロジェクト名引数、またはタイプ名引数を置き換え可能です。クエリのネストの深さは無制限です。

例

- `editor` という名前のすべてのプロジェクトのすべてのメンバーを検索する。

  ```
  ccm query "is_member_of(cvtype='project' and name='editor')"
  ```

- `toolkit-1.0`と同じメンバーを持つすべてのバージョン1.0プロジェクトを検索する。

  ```
  ccm query "has_member(is_member_of('1/project/toolkit/1.0')) and version
  = '1.0'"
  ```

- もっとも速い検索方法を使用して（none を指定して）`editor` というプロジェクト内のすべてのサブプロジェクトを検索する。

  ```
  ccm query "hierarchy_asm_members(cvtype='project' and name = 'editor', 'none')"
  ```

- 祖先としてオブジェクト `save.c-1`（タイプ `csrc`）を持つすべてのオブジェクトを検索する。

  ```
  ccm query "has_predecessor(cvtype='csrc' and name='save.c' and version='1')"
  ```

- プロジェクト `editor-fcheng` のディレクトリ `sources-1` で使用されているすべてのオブジェクトを検索する。

```
ccm query "is_child_of('sources-1:dir:1', cvtype='project' and
name='editor' and version='fcheng')"
```

- タスクのリリース値が 1.0 に設定されているタスクに関連するオブジェクトを含むすべてのプロジェクトを検索する。

```
ccm query "has_member(is_associated_cv_of(cvtype='task' and
release='1.0'))"
```

> 注記：Query ダイアログの Query フィールドでクエリ式を作成するとき、外側のクエリ式を二重引用符で囲む必要はありません。

## クエリ文節要素

クエリ文節は、個別の要素で構成されます。以下の要素を使用してクエリ文節を組み立てることができます。

- 関数
- 比較演算子
- 論理演算子
- 定数
- クエリ文節のグループ

### 関数

以下の関数引数および関数を使用して、関数ベースのクエリ文節を組み立てることができます。

- 関数引数
- 関数定義

#### 関数引数

関数引数を以下に示します。

**attr_name**

`is_product` や `platform` など、属性の名前を指定します。

**object_name**

オブジェクト バージョンのオブジェクト参照形式を指定します。

*`name-version:type:instance`*

**order_spec**

検索順序を指定します。*`order_spec`* に値 `depth` を使用すると、縦型検索が行われます。値 `breadth` を使用すると、横型検索が行われます。*`order_spec`* に値 none を使用すると、順序には意味がなくなり、検索は任意の順序で行うことができます（最速の方法が使用されます）。

```
none | depth | breadth
```

**privilege_name**

*read*（読み出し）や *write*（書き込み）など、権限の名前を指定します。

**project_name**

プロジェクトのオブジェクト バージョンの名前を指定します。

*project_name-version*

## 関数定義

クエリ関数を以下に示します。

**baseline (’*baseline_spec*’)**

データベースを検索し、指定した情報と一致するベースラインを探します。

**build (’*build_string*’)**

データベースを検索し、指定したビルド文字列を持つベースラインを探します。このクエリ関数は、"cvtype=’baseline’ や ’build=’build_string’"に展開され、これは、指定したビルド文字列を持つすべてのベースラインを返します。

**cr (’*cr_id*’)**

データベースを検索し、指定した番号を持つ変更依頼を探します。

**folder (’*folder_id*’)**

データベースを検索し、指定した番号を持つフォルダを探します。

**has_attr (’*attr_name*’)**

データベースを検索し、*attr_name* 属性（*is_product* や *platform* など）を持つすべてのオブジェクト バージョンを探します。

**has_child (’*object_name*’,’*project_name*’)**

データベースを検索し、プロジェクト *project_name* にオブジェクト *object_name* をメンバーとして持つすべてのディレクトリ オブジェクト バージョンを探します。

**has_member (’*object_name*’)**

データベースを検索し、指定したオブジェクト バージョンがメンバーになっているすべてのプロジェクト オブジェクト バージョンを探します。

**has_model (’*object_name*’)**

データベースを検索し、指定したモデル オブジェクト バージョンをモデルとして使用するすべてのオブジェクト バージョンを探します。

たとえば、Base Model を使用する場合、このクエリは has_model(base-1:model:base’) となります。

**has_no_*relationship* ()**

データベースを検索し、どのオブジェクトともその名前の関係を持たないオブジェクトを探します。

たとえば、has_no_successor は、子孫を持たないすべてのオブジェクトを返します。

**has_predecessor (*'object_name'*)**

データベースを検索し、指定したオブジェクト バージョンが直接祖先となっているすべてのオブジェクト バージョンを探します。

**has_priv (*'privilege_name'*)**

*privilege_name* は、*read*（読み出し）や *write*（書き込み）など、権限の名前を指定します。

**has_purpose (*'purpose_name'*)**

データベースを検索し、指定した目的を持つすべてのプロジェクトを探します。

**has_relationship (*'objectspec', 'operator', time*)**

関係作成時間が指定演算子（=、!=、>,<=、>、または >=）および時刻値と一致する指定オブジェクトへの指定関係を持つすべてのオブジェクトを検索します。たとえば、Rational Synergy は has_successor を使用して関係を表示します。

*relationship* は、associated_cv や associated_task など、定められた任意の関係を指します。

*has_relationship*('objectspec', 'operator', time) は、*is_relationship_of*('objectspec', 'operator', time) の逆クエリです。

関係の作成方法については、history コマンド を参照してください。

**has_type (*'type_name'*)**

データベースを検索し、タイプが *type_name* のすべてのオブジェクト バージョンを探します。 たとえば、クエリ has_type('csrc-1:cvtype:base') により、HelloWorld-1:csrc:1 が検索されます。

**hierarchy_project_members (*'project_name', order_spec*)**

データベースを検索し、*project_name* で指定したプロジェクト階層のすべてのプロジェクトを探します。*order_spec* 引数により、recursive_is_member_of で説明したように検索順序を指定します。

クエリにより、オブジェクト バージョン名の整列リストが返されます。このクエリを他のクエリと組み合わせて使用すると、結果の順序が変わる場合があります。

このクエリにより、*project_name* が返されます。

*order_spec* については、order_spec を参照してください。

**is_bound()**

データベースを検索し、いずれかのプロジェクトのメンバーであるオブジェクト バージョンを探します。このクエリは、プロジェクト名など他の限定オプション を指定する場合に便利です。

**is_child_of(*'object_name','project_name'*)**

データベースを検索し、プロジェクト *project_name* の *object_name* というディ レクトリのメンバーを探します。

**is_hist_leaf()**

データベースを検索し、履歴表示時にリーフ ノードであるオブジェクト（子孫を 持たないオブジェクト）を探します。

**is_hist_root ()**

データベースを検索し、履歴表示時にルート ノードであるオブジェクト（祖先を 持たないオブジェクト）を探します。

**is_member_of (*'project_name'*)**

データベースを検索し、指定プロジェクトのメンバーであるすべてのオブジェク トバージョンを探します。

**is_model_of (*'object_name'*)**

データベースを検索して、指定オブジェクト バージョンに関連するモデルオブ ジェクト バージョンを探します。

**is_no_*relationship* ()**

データベースを検索して、どのオブジェクトともその名前の関係のターゲットで はないすべてのオブジェクトを探します。

たとえば、is_no_successor は、子孫ではないすべてのオブジェクトを返します。

**is_predecessor_of (*'object_name'*)**

データベースを検索し、指定オブジェクト バージョンの直接祖先であるすべての オブジェクト バージョンを探します。

**is_*relationship*_of (*'objectspec', 'operator', time*)**

関係作成時間が指定演算子（=、!=、>,<=、>、または >=）および時刻値と一致す る指定オブジェクトから指定関係を持つすべてのオブジェクトを検索します。

例：

```
is_associated_cv_of ('task23-1:task:M', '>', time ('May 1, 2002') )
```

このクエリは、タスク M#23 の、2002 年 5 月 1 日以降に関連したすべての関連オ ブジェクトを探します。

has_*relationship*(*'objectspec'*, *'operator'*, *time*) は、is_*relationship*_of(*'objectspec'*, *'operator'*, *time*) の逆クエリです。

関係の作成方法については、history コマンド を参照してください。

**is_type_of (*'object_name'*)**

*object_name* の作成に使用したモデル内で *type* オブジェクト バージョンを検索します。

**recursive_is_member_of (*'project_name'*, *order_spec*)**

データベースを検索し、*project_name* で指定したプロジェクト階層のすべてのプロジェクトのすべてのメンバーを探します。

クエリにより、オブジェクト バージョン名のリストが返されます。プロジェクトでないものがリストの初めに表示され、階層のメンバーであるプロジェクトが後に続きます。検索順序に関係するのは、結果におけるプロジェクトの場所のみです。このクエリを他のクエリと組み合わせて使用すると、結果の順序が変わる場合があります。

このクエリでは、*project_name* は返されません。

*order_spec* については、order_spec を参照してください。

**task (*'task_id'*)**

データベースを検索し、指定した番号を持つタスク依頼を探します。

**versions_in_a_baseline (*'project_spec'*)**

ベースラインからプロジェクト バージョンを検索します。このクエリ関数は、"cvtype='project' と name='<project_name>' と instance='<project_subsystem>' と not is_no_project_in_baseline()'" に展開され、これはプロジェクトオブジェクトに対して任意のベースライン内にあるすべてのプロジェクトバージョンを返します。

## 比較演算子

クエリには比較演算子を使用できます。下表に、クエリ文節の構成に使用できる比較演算子を示します。

| 演算子 | 説明 |
|---|---|
| = | 属性の値が定数の値と同じである必要があります。 |

| 演算子 | 説明 |
|---|---|
| != | 属性の値が定数の値と同じであってはなりません。ただし、属性が必要です。 |
| < | 属性の値が定数の値より小さい必要があります。 |
| <= | 属性の値が定数の値より小さいか、同じである必要があります。 |
| > | 属性の値が定数の値より大きい必要があります。 |
| >= | 属性の値が定数の値より大きいか、同じである必要があります。 |
| match | 属性の値が定数の値と一致する必要があります。定数にはワイルドカード文字 "*" および "?" を含めることができます。ワイルドカード文字 "*" は任意の数の任意の文字、ワイルドカード文字 "?" は任意の 1 文字の代わりに使用します。match 演算子では、大文字と小文字が区別されます。<br>ワイルドカードは、属性の初めの 63 文字についてのみ使用できます。 |
| !match | 属性の値は定数となり得る値のいずれとも一致してはなりません。 |
| smatch | 属性の値は文字列と一致する必要があります。smatch 演算子では、大文字と小文字が区別されます。 |

## 論理演算子

クエリ文節を組み立てるのに使用できる論理演算子は **and**、**or**、**not** です。

*query_clause1* **and** *query_clause2:*

両方の *query_clause*s の条件と一致するオブジェクト バージョンのみが選択されます。

*query_clause1* **or** *query_clause2*:

いずれかの *query_clause* の条件と一致するオブジェクト バージョンが選択されます。

**not** *query_clause*:

*query_clause* の条件と一致しないオブジェクト バージョンのみが選択されます。

not 演算子は and 演算子に優先します。and 演算子は or 演算子に優先します。この優先ルールを無効にするには、クエリにかっこ「( )」を使用します。

## 定数

下表に、クエリ文節に使用できる定数を示します。属性タイプのクラスは指定した定数によって決まります。たとえば、定数が文字列の場合、文字列属性（文字列のサブタイプである属性）のみが一致します。

| 定数のタイプ | 説明 |
|---|---|
| 文字列 | 一重引用符で囲みます。 |
| 整数 | 範囲は 0 ～ +/-2147483647 です。 |
| `boolean` | `TRUE` または `FALSE` のいずれかです。 |
| `time` | 形式は `time('time_string')` です。詳細については、日付形式を参照してください。 |

## クエリ文節のグループ

かっこ「( )」を使用してクエリ文節をグループ化できます。深さに制限はありません。

## クエリの例

- `proj_ico-1` プロジェクトの *working* オブジェクト バージョン内のすべての `comment` 属性を検出し、変更する場合

  ```
  ccm query "status='working' and is_member_of('proj_ico-1')"
  ccm attr -m comment -v "increase surface complexity" @
  ```

- タスク 374 に関連付けられたすべての `executable`（実行形式）タイプのオブジェクトバージョンを探して表示する場合

  ```
  ccm query "type='executable' and is_associated_object_of('task374-
  1:task:probtrac')"
  ```

- a) 名前に web を含み、b) リリース値が 6.0 に設定されたプロジェクトのメンバーであるすべてのオブジェクトを探す場合 （ネストされたクエリを使用します）

  ```
  ccm query "is_member_of(release='6.0' and name match '*web*')"
  ```

- a) 名前が Advanced_Topics で、b) リリース値が 6.0 に設定されたプロジェクトのメンバーであるすべてのオブジェクトを探す場合 （ネストされたクエリを使用します）

  ```
  ccm query "is_member_of(release='6.0' and "name='Advanced_Topics'")"
  ```

- `toolkit-1.0` というプロジェクトのすべてのメンバーを探す場合

  ```
  ccm query "is_member_of('1/project/toolkit/1.0')"
  ```

- 最速の検索方法を使用して（none を指定）`editor-fcheng` というプロジェクト階層のすべてのメンバーを探す場合

  ```
  ccm query "recursive_is_member_of('1/project/editor/fcheng',none)"
  ```

- `csrc` オブジェクト、`save.c-2` を作成するために使用したモデル内の cvtype を検索する場合

  ```
  ccm query "is_cvtype_of('aave.c-2:csrc:1')"
  ```

- 所有者が `linda` で、過去 2 日間に修正されたすべてのオブジェクトを検索する場合

  ```
  ccm query -o linda "modify_time>time('-2:0:0:0')"
  ```

# 関係

Rational Synergy は、関係を使って、あるオブジェクトをデータベース内の他のオブジェクトに結び付けます。関係は、オブジェクト間で（たとえば、csrc オブジェクトは自身のチェックアウト元の後継）、オブジェクトとタスク間で（たとえば、タスクはオブジェクトに関連付けられています）、変更依頼とタスク間で（たとえば、タスクは変更要求を修正します）、またタスク間で（たとえば、タスクはタスクを修正します）維持されます。

ただし、オブジェクト という用語は、変更されたオブジェクトだけでなく、たとえば、プロジェクト、フォルダ、タスク、変更依頼など、Rational Synergy データベースに格納されているあらゆるオブジェクトを意味します。

以下のトピックでは、Rational Synergy 関係を説明します。

- 関係について
- ユーザー定義の関係
- 定義済みの関係
- 関連オブジェクトのクエリ

# 関係について

それぞれの関係には名前があり、その関係の性質を示します。関係名の例としては、successor、duplicate や associated_task があります。関係名は英字の任意の組み合わせです。

関係は一方向です。つまり、関係はあるオブジェクトから 別のオブジェクトに対するものです。たとえば、cosine.c-2 が cosine.c-1 の後継である場合、関係はバージョン 1 からバージョン 2 を示します。関係はいずれのオブジェクトからも示すことができます。たとえば、バージョン 1 は後継のバージョン 2 を持ち、バージョン 2 はバージョン 1 の後継です。

1 つのオブジェクトに任意の数の関係を含めることが可能です。異なるオブジェクトに対して同じ関係（同じ名前）の多数のインスタンスを持つことができます。たとえば、1 つの変更依頼には多数のタスクを関連付けられます。オブジェクトは関係のいずれか一端に置くことができます。たとえば、オブジェクトは後継オブジェクトを持つと共に他のオブジェクトの後継とすることが可能です。オブジェクトは多種類の関係を持つことができます。たとえば、フォルダ (task_in_folder) 内にあるタスクが、ソース オブジェクト（associated_cv）と関係付け、またそれを修正する他のタスク（fix）と関係付けることができます。

関係を持つオブジェクトを削除すると、Rational Synergy は自動的にその関係も削除します。

## ユーザー定義の関係

Rational Synergy は定義済みのすべての関係を維持するためのインスタンスを提供しますが、新しいタイプの関係が必要な場合は、relate コマンド を使用して関係を作成し、また unrelate コマンド でそれを解除できます。

チームでは、それぞれ異なる理由で新しいタイプの関係を必要とすることがあります。チームで有用な管理オブジェクト間の関係をトレースする場合は、新しいタイプの関係を作成すると便利です。たとえば、仕様とその対応するソース コード間に `associated_spec` という新しい関係を作成すれば、その関係をトレースできます。

CLI から 2 つのオブジェクトの関係を設定または設定解除するには、関係名、"from" オブジェクトと "to" オブジェクトを指定します。関係は、以下の例に示すように種々の方法で表示できます。

- "from" と "to" オブジェクトを指定することにより、2 つのオブジェクト間のすべての関係を表示する。
- "from" オブジェクトを指定することにより、ある 1 つのオブジェクトが関係する他のすべてのオブジェクトを表示する。
- "to" オブジェクトを指定することにより、ある 1 つのオブジェクトに関係している他のすべてのオブジェクトを表示する。
- "from"　オブジェクトと関係名を指定することにより、特定の関係を通してあるオブジェクトが関係しているすべての他のオブジェクトを表示する。

デフォルトでは、ユーザー定義の関係のセキュリティやセマンティックを管理するルールはありません。たとえば、デフォルトでは、任意の 2 つのオブジェクトが修正可能であるか否かにかかわらず、その間の関係を作成できます。関係を単対多、またはタイプ `problem` からタイプ `task` などのセマンティックに適合させたい場合でも、それを関係に定義する方法はありません。それに対し、関係を維持しこれらのルールを実施するためのスクリプトやプログラムを作成し、または Rational Synergy をカスタマイズできると便利です。

## 定義済みの関係

Rational Synergy は、すべての定義済みの関係を維持するためのインターフェイスを提供します。Tasks タブを使用して、タスク間の `fix` 関係を作成、削除、表示し、History ダイアログを使用してオブジェクト間の `successor` 関係（リンク）を作成、削除、表示し、また Properties タブを使用して変更依頼とタスク間、またはタスクとオブジェクト間の `associated_task` および `associated_cv` 関係（関連付け）を作成、削除、表示します。

定義済みの関係を維持するためには、提供されているダイアログとコマンドを使用してください。relate コマンド と unrelate コマンド は、ユーザー定義の関係をサポートする場合にのみ使用してください。

下表に、Rational Synergy に付随する定義済みの関係を示します。

| 名前 | 説明 |
|---|---|
| added_task_in_pg | プロジェクト グルーピングからタスク |
| associated_cv | タスクからソース オブジェクト（多対一） |
| automatic_task_in_pg | プロジェクト グルーピングからタスク（多対多） |
| baseline_in_pg | プロジェクト グルーピングからベースライン |
| baseline_project | プロジェクトからそのベースライン プロジェクト（多対一） |
| fix | 修正タスクからタスク（多対一） |
| folder_in_pg | プロジェクト グルーピングからフォルダ |
| folder_in_rp | プロジェクトからフォルダ（多対多） |
| folder_in_rpt | プロセス ルールからフォルダ（多対多） |
| folder_template | フォルダからフォルダ テンプレート（多対一） |
| folder_template_in_rpt | プロセス ルールからフォルダ テンプレート（多対多） |
| generic_pr | リリース固有のプロセス ルールから汎用プロセス ルール（多対一） |
| generic_pr_in_process | processdef から汎用プロセス ルール（多対多） |
| pr_in_release | processdef からプロセス ルール（多対多） |
| project_in_pg | プロジェクト グルーピングからプロジェクト |
| reconfigure_template | プロセス ルールからプロジェクト（多対多） |
| removed_task_in_pg | プロジェクト グルーピングからタスク |
| saved_task_in_pg | プロジェクト グルーピングからタスク |
| successor | 変更されたオブジェクトからその履歴後継（多対多） |
| task_in_folder | フォルダからタスク（多対多） |
| task_in_rp | プロジェクトからタスク（多対多） |

## 関連オブジェクトのクエリ

関連オブジェクトを表示するには、クエリを使用する方法もあります。query は関係のための 2 つのクエリ機能を提供します。それぞれ、各方向からのクエリを行います。クエリの構文を次に示します。

```
is_relationship-name_of(objectname)
has_relationship-name(objectname)
```

たとえば、cosine.c-1 の後継を見つけるには、次のコマンドを使用します。

```
ccm query "is_successor_of(’ cosine.c-1:csrc:1’ )"
```

後継として cosine.c-2 を持つオブジェクトを見つけるには、次のコマンドを使用します。

```
ccm query "has_successor(’ cosine.c-2:csrc:1’ )"
```

これらの機能は、ユーザー定義の関係でも使えます。たとえば、次に示すコマンドを使用して cosine.c-2 に関連付けられているオブジェクトを見つけられます。

```
ccm query "is_associated_spec_of(’ cosine.c-2:csrc:1’ )"
```

クエリの構成方法の詳細については、定義済みの関係 および クエリ文節要素 参照してください。

# 共有プロジェクト

共有プロジェクトは、*visible*、*shared*、その他の静的な状態にあるメンバーとの *shared*（共有）状態にある、プロジェクトです。共有 状態にあるプロジェクトでは、すべてのユーザーがプロジェクト内で変更を行うことができます。この特性は、プロジェクト内で 1 人のユーザーまたは特定ロールのユーザーのみが変更を行うことができる *working* 状態や *working* プロジェクトとは異なります。

以下のトピックでは、共有プロジェクトの概念と用途を説明します。

- 共有プロジェクトが役立つかどうかを判定する方法
- 共有プロジェクト手法
- パラレル開発と共有プロジェクト
- 共有プロジェクトのトラブルシューティング

このドキュメント中で参照されるダイアログとコマンドの詳細については、Rational Synergy ヘルプを参照してください。

# 共有プロジェクトが役立つかどうかを判定する方法

共有プロジェクトにはいくつかの利点があります。その大部分は、複数のユーザーが 1 つのプロジェクトを共有できることに関係しています。

しかし、その利点をもたらす同じ特徴が、制限にもなるので、注意する必要があります。利点だけではなく、制限も理解する必要があります。そうすれば、自分達のチームにとって共有プロジェクトが適切であるかどうかを判定できます。

- 共有プロジェクトの利点
- 共有プロジェクトの制限
- 共有プロジェクトを利用する最良の方法のシナリオ

## 共有プロジェクトの利点

共有プロジェクトを使用することには、以下の利点があります。

- 時間の節減

  小さな変更を行うためだけに 1 つの大きなプロジェクト（または複数のプロジェクト）の *working* バージョンをチェックアウトする必要はありません。共有プロジェクトの *working* バージョンを維持するのではない限り、保守または管理操作を行う必要ありません。その代わりに、プロジェクトとそのオブジェクトに直接アクセスできます。

- トレーニングの節減

すべてのユーザーが同じプロジェクトを共有しているので、個々のユーザーがプロジェクトを管理する方法（作成、更新、同期化、属性の設定など）を知っている必要はありません。ビルド マネージャなど、1 人のユーザーがプロジェクトを管理する責任を負えばよいのです。

- 共有開発機能

  他のユーザーが行った変更が、直ちに利用可能になります。たとえば、Bob と Sue が同じ共有プロジェクトで作業している場合、Bob が行った変更を保存すると、Sue はすぐにその結果を利用できるようになります。

- ディスク領域の節減

  すべてのユーザーが 1 つのプロジェクトを使用しているので、プロジェクトのワークエリアに複数コピーが必要ありません。それにより、ディスク領域が節減されます。

  しかし、リンクベースのワークエリアを使用している場合（UNIX において）、またはチームがワークエリア内に保有している管理対象外ファイルが少ない場合、ディスク領域の節減量は微細なものであることに、留意してください。

## 共有プロジェクトの制限

共有プロジェクトを使用することには、以下の制限があります。

- 個別開発が不能

  これは共有プロジェクトの最も重大な制約です。なぜならば、特に、製品のビルドを作成する場合などに、予想外の動作が発生する状況が起こりえるからです。

  共有プロジェクトにおいてビルドを行うことは、平行してビルドを行う結果となります。ユーザーはいつでも共有製品をビルドすることができ、新しい製品が共有プロジェクトに自動的に組み込まれるからです。そして、ユーザーのビルド結果は通知なく変更されることがあります。たとえば、1 人のユーザーが共有プロジェクト内のライブラリを変更して、変更が直ちにインクルードされ、他のユーザーの以降のビルドが影響を受けます。

  共有ファイルに対する不完全な変更も予期せぬ結果を生む可能性があります。なぜならば、作業中のファイルは、デバッグが完了していなくとも、共有プロジェクトのメンバーであるからです。

  最後に、共有ワークエリア内の管理対象外ファイルがビルドの結果に影響を及ぼすこともあります。たとえば、管理されていない再配置可能なオブジェクトが別々のユーザーによって同時にアクセスされたり、ファイルを壊したりビルド結果を変えたりする順序でアクセスされたりします。

  変更にすぐにアクセスできるという特徴が、他のユーザーに矛盾する変更や不完全な変更やエラーを、突然と前触れもなく引き起こすことにつながることに、留意する必要があります。

個別プロジェクト必要とする場合は、プロジェクトをチェックアウトして、*working* バージョンで作業を行います（パラレル開発と共有プロジェクト を参照）。

- 限定的なファイル システムのセキュリティ

  Rational Synergy では、共有プロジェクトのコピーベースのワークエリアに関して、ファイルに関するアクセス許可を全面的にはサポートしていません。すべてのユーザーがコピーベースのワークエリアにおいて、ファイル名を変更し、ファイルを移動し、ファイルを削除するアクセス権限を有しています。また、他のユーザー向けにチェックアウトされたオブジェクトのファイルも変更できます。

  コピーベースのワークエリアにおいては、ワークエリア内のファイルに加えた変更は Rational Synergy データベース内の管理されているファイルに影響を及ぼしません。しかし、ワークエリアからデータベースへ戻す際に、不必要な変更をリコンサイルしないように、注意する必要があります。

  Windows のユーザーは、ワークエリア用に非共有ドライブを使用することにより、不必要な変更を防ぐことができます（しかし、この方法では複数のユーザーが同じワークエリアを共有できません）。UNIX ユーザーは、ワークエリアをリンクベースにすることにより、ワークエリアに対する不必要な変更を防ぐことができます。なぜならば、他のユーザーはファイルへのリンクを変更できますが、ファイル自体を変更できるのはファイルの所有者だけだからです。

- パラレル開発は禁止

  共有プロジェクトにおいては、パラレル開発は禁止されます。共有プロジェクトからあるオブジェクトがチェックアウトされた後では、誰もそのオブジェクトをチェックアウトできません。

  別バージョン（`ccm use` コマンドまたは Use ダイアログ）を使用して、そのバージョンからチェックアウトしてはなりません。なぜならば、他のユーザーの変更を削除してしまう可能性があるからです。その代わりに、共有プロジェクトから *working* プロジェクトをチェックアウトして、パラレル バージョンで作業できます（パラレル開発と共有プロジェクトを参照）。

- 共通ワークエリア パスが必要

  共有プロジェクトのワークエリアは、すべてのユーザーが使用できるパスに設定する必要があります。また、このパスはすべてのマシンにとって同じでなければなりません。すべてのユーザーが多くのさまざまなマシンから同じワークエリアにアクセスするので、ネットワーク トラフィックが増大してパフォーマンスが低下する可能性があります。

## 共有プロジェクトを利用する最良の方法のシナリオ

共有プロジェクトはすべてのタイプの開発プロジェクトに適するわけではありません。共有プロジェクトが適する開発プロジェクトは通常、プロジェクトおよび開発環境につい

て、以下の特徴を備えています。共有プロジェクトを利用するかどうか決定する際には、あらゆる面にわたってデザインを検討する必要があります。

プロジェクトの特徴

- 各ユーザーの責任領域ごとにディレクトリが分けられている大規模なプロジェクト

  プロジェクトがこの構造に当てはまるなら、共有開発の利点を得られます。しかし、依然として、個別化されていないことに起因するリスクを下げる必要があります。

- 製品をビルドしないプロジェクト

  製品をビルドしないプロジェクトの例として、ドキュメントやウェブページを管理するプロジェクトや PowerBuilder のように統合を管理するプロジェクトがあります。

  製品をビルドする場合には、ビルド プロセスをモジュール化する必要があります (つまり、別々のディレクトリを作成して、ある程度の個別性を持たせます)。ビルド前に製品が確実に削除されるように、make ファイル を変更する必要があるでしょう。そうしないと、ファイルアクセス権限により、あるユーザーが他のユーザーによって作成されたファイルを更新できなくなる可能性があります。

- ローカル マシンにワークエリアを必要としないプロジェクト

  すべてのユーザーが同じワークエリアにアクセスするので、プロジェクトはローカル ハード ドライブ上にではなく、共有ドライブ上に配置すべきです。データに接続しないで作業する場合には（たとえば、自宅でファイルを処理する場合、またはパフォーマンスを向上するためファイルをローカル ディスクへ移す場合）、ユーザーはローカル マシン上で作業する必要があります。

開発環境の特徴

- 協力して作業し、個別化されていないワークエリアによって引き起こされる問題をよく理解している、小規模なチーム
- Rational Synergy のすべての機能を使用するようにはトレーニングされていない、ユーザーのチーム

  ユーザーは、製品について多くの知識がなくとも、チェックインやチェックアウトのような、基本的な Rational Synergy の操作を行うことができます。しかし、プロジェクトを作成して管理する方法については、ある程度の知識が必要です。

- すべてのチームメンバーは個別化されていないワークエリアの重大性を理解する必要がある。

## 共有プロジェクト手法

このセクションでは、タスクベースの手法について説明します。共有プロジェクトと標準プロジェクトではライフサイクルが違います。また、共有プロジェクトと標準プロジェクトとでは、CM の処理動作が異なるものがあります。

- ライフサイクル
- 共有プロジェクトの作成
- 共有プロジェクトにおける作業
- 共有プロジェクトの更新

### ライフサイクル

共有プロジェクトでは、*shared*（共有）と *visible*（可視）2 つの状態のみが使用されます。オブジェクトのタイプに応じて、これらの状態を利用します。

#### プロジェクト

共有状態は共有プロジェクトに利用します。共有状態はプロジェクトのライフサイクルの *working*（作業中状）態と *prep*（準備）状態の間にあります。共有状態では、すべてのユーザーが共有プロジェクト内で作業できます。デフォルトでは、共有プロジェクトにおいては、作業中オブジェクトを使用しません。

#### オブジェクト

共有プロジェクトにおいては、チェックアウトされたオブジェクトに可視状態および共有状態を適用します。

可視状態は通常の（非製品）オブジェクトの共有開発に適用されます。オブジェクトの内容を変更できるのは、オブジェクトの所有者だけです。しかし、すべてのユーザーがそれを使用（自分の作業中プロジェクトに追加したり、共有プロジェクトに追加したり）できます。

共有状態は製品オブジェクトの共有開発に利用します。すべてのユーザーが共有オブジェクトを使用したり変更したりできます。

ライフサイクルに関するさらに詳細については、『IBM Rational Synergy の紹介』を参照してください。そこには、プロジェクト、オブジェクト、および製品のライフサイクルが詳しく説明されています。このドキュメントは、http://www.ibm.com/software/rational/support/ で入手できます。

### 共有プロジェクトの作成

どのユーザーも共有プロジェクトを作成できます。しかし、できるだけ、ビルド マネージャに共有プロジェクトの責任を持たせてください。その主な理由は、ビルド マネージャ

はプロジェクト管理（作成、更新プロパティの設定、更新の実行、など）に通じているからです。しかし、すべてのユーザーがプロジェクトに通じている環境では、チーム リーダーにも共有プロジェクトの責任を持たせられます。開発者が密接に協力している非常に小さなチームでは、個々の開発者に共有プロジェクトの責任を負わせてもよいでしょう。

#### **copy project** の使用

プロジェクトのライフサイクルは 作業中状態から始まります。しかし、プロジェクトを直接共有状態にコピーできます。そのためには、Copy Project ダイアログにおいて、プロジェクトの目的を **Shared Development** に設定します。コマンド ラインから共有プロジェクトを作成するには、`ccm copy_project` コマンドを使用し、`-purpose Shared Development` オプションを付加します。

Rational Synergy からプロジェクトの目的を **Shared Development** に変更できます。この場合も状態が変更されます。

#### **create** の使用

Create Project 操作を行うかまたは `ccm create` コマンドを使用して、作業中状態のプロジェクトを作成します。それから、そのプロジェクトを共有状態にチェックインします。次の状態を必ず共有に設定するようにします。その理由は、次の状態のデフォルトの設定は通常、ユーザーの役割に応じて、チェックポイント または準備中であるからです。

### 共有プロジェクトにおける作業

共有プロジェクトにおいて作業を行うときは、チェックアウトおよびチェックインの操作を行う必要があります。ただし、これらの操作は標準プロジェクトの場合とは異なります。

#### チェックアウト状態

通常のオブジェクトは共有プロジェクト内の可視状態にチェックアウトされます。製品オブジェクトは共有プロジェクト内の共有状態にチェックアウトされます 。サブプロジェクトは手作業で 共有状態にチェックアウトできます。

> 注記：ファイル アクセス許可を変更して書き込み可能にするのではなく、変更したいオブジェクトを必ずチェックアウトします。チェックアウトされたファイルだけを変更することにより、データの整合性を保証する役に立ちます。

#### ディレクトリの自動チェックイン

共有プロジェクトにおいては、あるディレクトリを自動的にチェックアウトするコマンドはすべて、そのディレクトリを自動式にチェックインもします。この機能により、

共有ディレクトリがすべてのユーザーに利用可能になります。したがって、あるディレクトリ メンバーへの変更が完了するのを待たずに、自分のディレクトリの変更を行うことができます。

たとえば、あるユーザーが web ディレクトリ中に新しいファイルを作成すると、create コマンドの結果として、そのディレクトリが自動的にチェックアウトされます。そのディレクトリは自動的にチェックインされるので、そのユーザーが自分のファイルを変更している間も、他の開発者は変更を行うことができます。そのディレクトリが自動的にチェックインされなかったとしても、そのユーザーが変更を完了するまで、そのディレクトリはおそらくチェックアウトされた（変更不可）状態を保ちます。

ディレクトリを自動的にチェックインするためには、タスクベースの共有プロジェクトで作業を行う前に、カレント タスクを設定する必要があります。カレント（デフォルト）のタスクが設定されていないと、関連するタスクがないままにディレクトリがチェックインされます。したがって、ユーザーが行った変更は更新時に他のプロジェクトによって選択されません。また、チェックインの際にタスクが必要であるが、カレント タスクが設定されていないと、ディレクトリの自動チェックインは失敗します。

このディレクトリの自動チェックイン機能を無効に設定できます。そのためには、ccm.ini ファイルに以下の行を登録します。

```
shared_project_directory_checkin = FALSE
```

### パラレル開発はなし

チェックアウトされた可視状態のオブジェクトからチェックアウトすることはできません。しかし、共有プロジェクトにおいては、そのオブジェクトの別バージョンを使用することにより（ccm use コマンドまたは Use ダイアログ）、チェックアウトされたオブジェクトを置き換えることができます。チェックアウトされたオブジェクトの別バージョンを使用すると、他のユーザーによってチェックアウトされたオブジェクトを未使用にすることになります。

この場合も、自分用にチェックアウトされたファイルだけを編集します。ファイルのアクセス権限を変更して書き込み可能にすることは避けます。

> 注意！ 他のユーザーのチェックアウトされたオブジェクトを切り捨ててはなりません。そのような操作を行うと、データが失われることがあります。

## 共有プロジェクトの更新

共有プロジェクトにおいて作業を行うときは、プロジェクトの更新操作を行う必要があります。ただし、これらの操作は標準プロジェクトの場合とは異なります。

### 更新

共有プロジェクトのバージョンが 1 つしか存在しない（そのプロジェクトの作業中のバージョンを保持しているユーザーがいない）場合、共有プロジェクトを管理する必要はほとんどありません。共有プロジェクト内のオブジェクトが他のプロジェクトにチェックアウトされていない限り、共有プロジェクトは常に最新であり、更新する必要はありません。したがって、共有プロジェクトを作成し、それからユーザーが Rational Synergy の基本的なコマンド（カレントタスクの設定、チェックアウト、作成、編集、タスクの完了（チェックイン））を使用してプロジェクトを更新することを許可できます。

共有プロジェクトの 作業中バージョンが存在する場合、作業中のプロジェクト内で行われた変更を収集するために、ビルド マネージャまたはチーム リーダーは共有プロジェクトの更新を行う必要があります。更新プロパティには、現在のリリースに関するすべてのタスクを問い合わせるように設定された、1 つのフォルダを含めます。このフォルダはすべてのユーザーが使用でき書き込める必要があります。

更新を行うと、多くのオブジェクトが更新される可能性があり、共有プロジェクトを利用している開発者に影響を及ぼします。したがって、その影響を少なくするために、プロジェクトを使用しているユーザーがほとんどいないときを選んで更新を行ってください。また、更新に続いて、ビルド マネージャは更新ログを確認して、予期せぬ結果が生じていないか、マージすべきパラレル バージョンが存在しないか、確認する必要があります。

すべてのユーザーに最新のプロジェクトを提供するには、以下のプロセスに従ってください。

1. プロジェクトを管理する責任を負うユーザー（多くの場合、ビルド マネージャ）がすべてのユーザーにタスクを完了させてオブジェクトをチェックインするよう、要請する。
2. ユーザーがタスクを完了する。
3. ビルド マネージャがプロジェクトを更新する。

### リコンサイル

すべてのユーザーは共有プロジェクト内でリコンサイル処理を行うことができます。ただし、各ユーザーが破棄できるのは、静的な状態にあるオブジェクト変更または自分に対してチェックアウトされたものだけです。他のユーザーに対してチェックアウトされたオブジェクトをリコンサイル処理を通じて変更することはできません。

単一のオブジェクトをリコンサイルするには、`ccm resync` コマンドを使用します。

## パラレル開発と共有プロジェクト

共有プロジェクトを使用するとき、以下のタイプのパラレル開発が一般的に行われます。

- 同時パラレル開発

  2 人以上の開発者が同時に同じオブジェクトを変更することがあるため（単一のオブジェクトが複数のユーザーにチェックアウトされる）、オブジェクトにはパラレルバージョンが存在します。

- バリアント用のパラレル開発

  2 つ以上のリリースまたはプラットフォームが必要な場合、プロジェクトの複数のバージョンが存在します。

## 同時パラレル開発

パラレル開発を行うために、または個別バージョンのプロジェクトで作業を行うために、開発者は共有プロジェクトの作業中バージョンをチェックアウトできます。プロジェクトの作業中バージョンにより、開発者は必要な個別開発環境が得られ、他の開発者は共有バージョンにより自分達のプロジェクト作業を続行できます。

たとえば、共有プロジェクトで作業しており、完成に数日を要する変更を行う場合、他の開発者によって行われた変更によって自分の作業が干渉されるのを防ぎたいことがあります。このような場合、共有プロジェクト内で作業する代りに、共有プロジェクトの作業中バージョンをチェックアウトします。これによって、元のプロジェクトとメンバーが同じ作業中プロジェクトが作成されます。新たにチェックアウトされた作業中プロジェクトには、可視オブジェクトと共有オブジェクトが含まれています。自分用の個別エリアを維持するために、修正可能なオブジェクトを作業中プロジェクトから除去する必要があります。

共有プロジェクトの作業中バージョンをチェックアウトするためには、以下の手順に従います。

1. 目的を個別開発または共同開発として、プロジェクトをチェックアウトします。
2. プロジェクトを更新します。

これらのステップにより、作業中プロジェクトは変更から個別化されます。更新を行うと、可視オブジェクトも 共有オブジェクトもプロジェクトから見えなくなります。作業中プロジェクトのチェックアウトについてさらに詳しい情報が必要な場合は、Rational Synergy のヘルプを参照してください。

作業中プロジェクトでのすべての変更を完了したら、ビルド マネージャが共有プロジェクトを更新して、変更を反映させます。

## 異なるプラットフォームまたはリリース用のパラレル開発

共有プロジェクトの状態の異なるパラレル バージョンがいくつかあり、それらのバージョンでは複数のプラットフォームまたはリリースをサポートしていることがあります。

たとえば、共有プロジェクト `airplane-hp_2.0s`（HP プラットフォーム上での airplane のリリース 2.0）に `airplane-hp_3.0s` （将来のリリース）、`airplane-hp_1.0p1`（前のリ

リースへのパッチ)、および airplane-aix_2.0s（現行リリースの別のプラットフォーム版）というパラレル プロジェクトがあるとします。バージョン hp_2.0s は共有されていますが、他のパラレル プロジェクトは共有されていません。この例では、airplane-hp_1.0p1 は準備中状態にあって、提供すべきパッチのリストを備えている必要があります。

共有プロジェクトの階層は platform 属性と release 属性を用いて維持されます。前の例を続けると、airplane プロジェクトは wing と fuselage と rudder というサブプロジェクトから構成されるものとします。airplane-hp_2.0s を更新するとき、wing-hp_3.0s ではなく、wing-hp_2.0s を選択する必要があります。release 属性により、各サブプロジェクトの合致するバージョンが識別されます。

しかし、同じプラットフォームおよびリリースの複数の共有プロジェクト階層がある場合、更新を行う際に間違ったサブプロジェクト バージョンを選択してしまう可能性があります。この問題を避けるためには、一意のプロジェクト目的を使用して、同じプラットフォームとリリースの値を持つ階層をチェックアウトします。

たとえば、以下のプロジェクト目的リストでは、airplane プロジェクトの 3 通りの共有プロジェクト階層に対応しています。具体的には、共有プロジェクトの汎用目的、構造エンジニアのチーム向けの共有プロジェクト、電気エンジニア向けの共有プロジェクトという階層になっています。

```
Shared Development: shared:
Shared - structural: shared: structural
Shared - electrical: shared: electrical
```

3 つのチームのために airplane プロジェクトの別バージョンを作成する代りに、3 つのチームは 3 つの別々の共有プロジェクトで作業しています。3 つのプロジェクト階層をすべて定期的に更新して、それらの間の同期を取ることができます。あるいは、各チームが他のチームからどのような頻度で変更を取り込むかも決定できます。

## 共有プロジェクトのトラブルシューティング

共有プロジェクトを利用して問題に遭遇した場合、以下の問題点に着目して検討してください。

### 不完全な変更が含まれるディレクトリ

最初の変更が完了する前に、2 番目のディレクトリのチェックアウトが行われると、ディレクトリの自動チェックインの際に、不完全な変更が含まれることがあります。たとえば、Bob が src ディレクトリの中から foo.c ファイルを削除するタスクを割り当てられたとします。Bob は foo.c ファイルを削除します。この処理はチェックアウトされて、直ちに src ディレクトリにチェックインされます。それから、Bob は他のプロジェクト内の foo.c への参照を削除します。Bob が参照の削除を完了する前に、Sue が src ディレクトリから他のディレクトリへファイルを移動します。src ディレクトリはチェックアウトされますが、Sue がタスクを完了すると、直ちにチェックバックされます。Bob がまだ変

更を完了していないのに、ビルド マネージャが、Sue と Bob の変更を使用して、準備中プロジェクトを更新します。

不完全な変更は準備中プロジェクト内の構成のコンフリクトのように見えます。ビルド マネージャが準備中プロジェクトを更新した後で、コンフリクトを表示すれば、この問題を回避できます。

## 過負荷の状態

共有プロジェクトにおいては、変更が終わった直後に、その変更の影響を受けるすべてのダイアログが Rational Synergy により更新されます。この動きにより、他のユーザーが処理を行っている最中に、インターフェイスの負荷が過剰になることがあります。

たとえば、Bob が Sue の プロジェクト エクスプローラ内にあるオブジェクトをチェックアウトしたときに、オブジェクトの更新中に待機中（砂時計）カーソルが表示されることがあります。待機中カーソルは通常それほど気になることはありません。この動作は、更新中にオブジェクトに対する変更を防止するものです。しかし、多数のオブジェクトを変更する操作（たとえば、更新）をユーザーが実行すると、変更の累積効果の影響で、カーソルがずっと待機中となって点滅し続け、どのダイアログも使うことができなくなります。

変更度合いが高い操作においてはユーザー インターフェイスが一時的に使用できなくなる可能性があるので、ビーク時にはそのような操作を実行することは避けるべきです。

# SOAD スコープ

*scope*（スコープ）とは、オフライン保存して削除するオブジェクトのリストを作成するために、オフライン保存と削除（SOAD）ツールによって使用される、修正版のクエリです。以下の高度なトピックでは、有効なオブジェクトのリストを作成するために、スコープが SOAD によってどのように使用されるかについて説明します。

- スコープの評価
- スコープの検証
- 定義済みのスコープ

## スコープの評価

オフライン保存と削除のスコープは以下のように評価されます。

1. 指定したオブジェクトが存在すれば、当初のオブジェクト リストに含められる。指定したオブジェクトが存在しなければ、エラー メッセージが出されて、評価は終了する。
2. クエリ式があれば評価され（スコープに固有の除外クエリがあれば含まれる）、見つかったオブジェクトがオブジェクト リストに追加される。
3. オブジェクト リストが空であれば、エラー メッセージが出されて、評価は終了する。
4. クエリ リスト中の各オブジェクトについて、適用可能な各展開ルールが適用される。展開ルールがタイプ固有である場合、展開対象のオブジェクトがそのタイプである場合にのみ、そのルールが適用される。無限の再帰または重複した展開を避けるために、どのオブジェクトが展開されたかが記録される。クエリベースの展開ルールが実行されたとき、スコープに固有の除外クエリがあれば、クエリ式に追加される（このルールは ACcent ベースの展開ルールには適用されない）。
5. オブジェクト　リスト内の各オブジェクトについて、タイプがグローバル除外タイプ（たとえば、モデル オブジェクト）のどれかに該当するものがあれば、リストから削除する。
6. オブジェクト リスト内に残っている各オブジェクトについて、オフラインで保存されており同じインスタンスの別バージョンを持たないオブジェクトがあれば、リストから削除する。
7. リストに残っており該当タイプに合致するオブジェクトに照らして、スコープに固有な除外ルールを順々に評価して適用する。クエリから返されたオブジェクトの中に、オブジェクト リストに含まれていないものがあれば、その評価対象のオブジェクトはリストから削除される。
8. リストに残っている各オブジェクトについて、そのオブジェクトを使用して、プロジェクトのクエリが実行される。オブジェクトの親プロジェクトがオブジェクト削除リスト中になければ、そのオブジェクトは削除される。

9. 最後に、ユーザーが *ccm_admin* ロールを持っていないために修正できないオブジェクトがあれば、リストから削除される。これにより、アドミニストレータ以外のユーザーが SOAD を使用して、自分達の *working*、*checkpoint*、*visible*、*public*、*prep* のいずれかのバージョンを削除することが可能になる。

この評価が終わった時点では、オブジェクト リストには削除の候補でありオフライン保存の対象としてもよい、すべてのオブジェクトが含まれています。

Rational Synergy Classic の Save Offline and Delete ダイアログ ボックスで Verbose チェック ボックスを選択するか、コマンド ラインで `-verbose` オプションを使用して、詳細なスコープ評価情報を入手できます。しかし、この種のトレースを作動させると、処理が遅くなることに留意してください。

最初のオブジェクト仕様またはクエリが評価された後で、オブジェクト リストを変更する働きをする、スコープ評価機能には以下のものがあります。

- グローバル除外
- キーワード
- 展開ルールと除外ルール

## グローバル除外

一般的なクエリベースのメソッドのみを用いてオプションを削除しようとすると、保存する必要のあるオブジェクトを誤って削除してしまうことがありえます。したがって、SOAD は以下のタイプのオブジェクトを除外します。

- モデル オブジェクト タイプ
- 除外されたプロジェクト内の候補オブジェクト
- オブジェクトの最後の静的なバージョン

### モデル オブジェクト タイプ

Rational Synergy データベースを運用するために必要なオブジェクトを削除してしまうことを防止するために、`soadf_excluded_types` という名前のモデル属性中にリストされているオブジェクト タイプを、SOAD は除外します。

出荷時には、以下のタイプがその中に含まれています。

```
model
mcomp
cvtype
attype
bstype
admin
tset
```

```
update_temp
folder_temp
```

### 除外されたプロジェクト内の候補オブジェクト

SOAD のセキュリティ ルールにより、削除されないプロジェクト内で使用されているオブジェクトを削除することは禁止されています。これはグローバルな制約であるため、このルールはすべてのスコープに適用されます。

### オブジェクトの最後の静的なバージョン

あるオブジェクト インスタンスの最後の（静的な）バージョンを SOAD は除外します。そのような制約がないと、オブジェクトが削除された後で、ユーザーが同じ名前とタイプの新しいオブジェクトを作成し、インスタンス値を再使用する可能性があります。これによって、`cluster_id` コンフリクトが発生するため、保存された元のオブジェクトをリストアできなくなります。

このルールを適用するために、SOAD は履歴のルートからの各バージョンの深さを判定し、最も深い静的オブジェクトを選択します。オブジェクトの静的バージョンがない場合、SOAD は最も深い非静的バージョンを見つけ出し、*prep* バージョンを検索し、次にその他のバージョンを検索します。

> 注記：このオプションを適用するためには、**Save Offline** オプションも使用する必要があります。

## キーワード

オブジェクト名、クエリ、展開クエリ、除外クエリ、パッケージ名の中で、キーワードを使用できます。

キーワードは "%" で始まるもので、以下の表に示すように展開されます。

> 注記：リテラル文字列を用いて、キーワードを定義できます。そのためには、文字列の前に、"%" を 2 つ付けます。たとえば、"%release" と読み替えられるキーワードを定義するには、"%%release" と指定します。

| キーワード | ルール内での展開 | オブジェクト名、クエリ名、パッケージ名内での展開 | 置換 |
|---|---|---|---|
| %1 | × | ○ | 最初のパラメータの値 |
| %2 | × | ○ | 2 番目のパラメータの値 |
| %3 | × | ○ | 3 番目のパラメータの値 |

| | | | |
|---|---|---|---|
| %4 | × | ○ | 4 番目のパラメータの値 |
| %5 | × | ○ | 5 番目のパラメータの値 |
| %user | ○ | ○ | ユーザー名 |
| %date | ○ | ○ | 現在の日付と時刻 |
| %object | ○ | × | 処理対象のオプションの名前 |

## 展開ルールと除外ルール

展開ルールは、当初リスト内に収録されているオブジェクトとの関係に基づいて、オブジェクト リストにオブジェクトを追加するものです。除外ルールは、当初リスト内に収録されているオブジェクトとの関係に基づいて、オブジェクト リストからオブジェクトを削除するものです。

SOAD には、定義済みの一連の展開ルールと除外ルールが用意されています。それらのルールは、ユーザーが必要とする展開および除外の働きを十分に提供する、強力なものです。GUI または CLI を使用して、展開ルールや除外ルールを変更することはできません。しかし、標準のテキスト エディタを使用して、それらのルールを変更できます。

展開ルールは、`soadf_expansion_rules` という名前の、モデル属性内に格納されています。除外ルールは、`soadf_exclusion_rules` という名前の、モデル属性内に格納されています。各エントリは、クエリ名、オブジェクト タイプ、クエリ タイプ、値から構成され、それらが ':' で区切られて、1 行に記載されています。

下表に、定義済みの展開ルールを示します。

| 展開ルール | オブジェクト タイプ | ルール タイプ | 説明 |
|---|---|---|---|
| Project's folders（プロジェクトのフォルダ） | プロジェクト | クエリ | 指定した プロジェクトの更新プロパティ 内で使用されている、すべてのフォルダを含めます。 |
| Project's tasks（プロジェクトのタスク） | プロジェクト | クエリ | 指定されたプロジェクトの更新プロパティ内で使用されている、すべてのタスクを含めます。 |
| Project's non-automatic tasks（プロジェクトの非自動タスク） | プロジェクト | クエリ | 指定されたプロジェクトの更新プロパティ内で使用されている、すべての非自動タスクを含めます。 |
| Project's baseline project（プロジェクトのベースライン プロジェクト） | プロジェクト | クエリ | 指定されたプロジェクトに関する、すべてのベースライン プロジェクトを含めます。 |
| Project's members（プロジェクトのメンバー） | プロジェクト | クエリ | 指定されたプロジェクトの直接のメンバーである、すべてのオブジェクトを含めます。 |
| Project's recursive members（プロジェクトの再帰メンバー） | プロジェクト | クエリ | 指定されたプロジェクトの直接のメンバーまたは再帰的メンバーである、すべてのオブジェクトを含めます。 |

（続き）

| | | | |
|---|---|---|---|
| Project's products（プロジェクトの製品） | プロジェクト | クエリ | 指定されたプロジェクトの直接のメンバーであ、すべての製品を含めます。 |
| Hierarchy project members（階層プロジェクト メンバー） | プロジェクト | クエリ | 指定されたプロジェクトの直接のメンバーまたは再帰的メンバーである、すべてのオブジェクトおよびプロジェクトを含めます。 |
| Projects using baseline project（ベースライン プロジェクトを用いたプロジェクト） | プロジェクト | クエリ | 指定されたプロジェクトのベースラインであるすべてのプロジェクトを含めます。 |
| Non-static projects using baseline project（ベースライン プロジェクトを使用した非静的なプロジェクト） | プロジェクト | クエリ | 指定されたプロジェクトのベースラインであるすべての非静的なプロジェクトを含めます。 |
| Project's baselines（プロジェクトのベースライン） | プロジェクト | クエリ | 指定されたプロジェクトがベースライン中にある、すべてのプロジェクトを含めます。 |
| Folder's tasks（フォルダトのタスク） | フォルダ | クエリ | 指定されたフォルダ内にある、すべてのタスクを含めます。 |
| Folder's non-automatic tasks（フォルダの非自動タスク） | フォルダ | クエリ | 指定されたフォルダ内にある、すべての非動タスクを含めます。 |
| Projects using folder（フォルダを使用しているプロジェクト） | フォルダ | クエリ | 指定されたフォルダが更新プロパティ内で使用されている、すべてのプロジェクトを含めます。 |
| Task's objects（タスクのオブジェクト） | タスク | クエリ | 指定されたタスクに関連する、すべてのオブジェクトを含めます。 |
| Task's baseline（タスクのベースライン） | タスク | クエリ | 指定されたタスクを使用する、すべてのベースラインを含めます。 |
| Projects using task（タスクを使用しているプロジェクト） | タスク | クエリ | 指定されたタスクが更新プロパティ内で使用されている、すべてのプロジェクトを含めます。 |
| Folders using task（タスクを使用しているフォルダ） | タスク | クエリ | 指定されたタスクを使用する、すべてのフォルダを含めます。 |

（続き）

| | | | |
|---|---|---|---|
| Baseline's projects（ベースラインのプロジェクト） | ベースライン | accent | 指定されたベースライン中にあり、なおかつベースラインが作成される前は存在していなかった、すべてのプロジェクトを含めます。 |
| Baseline's tasks（ベースラインのタスク） | ベースライン | クエリ | 指定されたベースライン内にある、すべてのタスクを含めます。 |
| Tasks associated with object（オブジェクトに関連するタスク） | | クエリ | 指定された関連オブジェクトがある、すべてのタスクを含めます。 |
| Project's project grouping（プロジェクトのプロジェクト グルーピング） | プロジェクト | クエリ | 指定されたプロジェクトを含む、すべてのプロジェクト グルーピングを含めます。 |

下表に、定義済みの除外ルールを示します。

| 除外ルール | オブジェクト タイプ | クエリ タイプ | 説明 |
|---|---|---|---|
| Baselines used by other project groupings（他のプロジェクト グルーピングに使用されているベースライン） | | クエリ | 他のプロジェクト グルーピングによって使用されている、すべてのベースラインを除外します。 |
| Baseline projects used by other projects（他のプロジェクトによって使用されているベースライン プロジェクト） | プロジェクト | クエリ | 他のプロジェクトによって使用されている、すべてのベースライン プロジェクトを除外します。 |
| Baseline projects used by other non-static projects（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 プロジェクト） | プロジェクト | クエリ | 他の非静的なプロジェクトによって使用されている、すべてのベースライン プロジェクトを除外します。 |
| Projects used by other baselines（他のベースラインによって使用されているプロジェクト） | プロジェクト | クエリ | 他のベースラインによって使用されている、すべてのプロジェクトを除外します。 |
| Projects that are the last static version（最後の静的なバージョンであるプロジェクト） | プロジェクト | accent | 含められているプロジェクトのうちの最後の静的なバージョンである、すべてのプロジェクトを除外します。 |
| Folder used by other projects（他のプロジェクトによって使用されているフォルダ） | フォルダ | クエリ | 更新プロパティ内で他のプロジェクトに使用されている、すべてのフォルダを除外します。 |
| Task used by other projects（他のプロジェクトによって使用されているタスク） | タスク | クエリ | 更新プロパティ内で他のプロジェクトに使用されている、すべてのタスクを除外します。 |

（続き）

| | | | |
|---|---|---|---|
| Task used by other folders（他のフォルダによって使用されているタスク） | タスク | クエリ | 他のフォルダによって使用されている、すべてのタスクを除外します。 |
| Task used by other baselines（他のベースラインによって使用されているタスク） | タスク | クエリ | 他のベースラインによって使用されている、すべてのタスクを除外します。 |
| Objects associated with other tasks（他のタスクに関連するオブジェクト） | | クエリ | 他のタスクに関する、すべてのオブジェクトを除外します。 |
| Objects associated with other non-automatic tasks（他の非自動タスクに関連するオブジェクト） | | クエリ | 他の非自動タスクに関する、すべてのオブジェクトを除外します。 |
| Attachments of any change request（任意の変更依頼の添付ファイル） | | クエリ | 変更依頼に関連す、すべての添付ファイルを除外します。 |
| Objects that are the last static version（最後の静的なバージョンであるオブジェクト） | | accent | 含められていオブジェクトのうちの最後の静的なバージョンである、すべてのオブジェクトを除外します。 |
| Project groupings containing other projects（他のプロジェクトを含むプロジェクト グルーピング） | プロジェクト_グルーピング | クエリ | 削除対象以外のプロジェクトを含む、プロジェクト グルーピングを除外します。 |

## スコープの検証

SOAD はスコープを以下のように検証します。

1. 新しいスコープを保存する前
2. 編集されたスコープを保存する前
3. スコープを評価する前

検証により、以下のことが保証されます。

- 参照されている展開ルールが存在すること
- 参照されている除外ルールが存在すること
- スコープ クエリの構文が有効であること
- スコープ除外クエリの構文が有効であること（ただし、依然として、評価時にクエリがエラーを起こす可能性がある）
- スコープ オブジェクト、スコープ クエリ、スコープ除外クエリ、スコープ パッケージ名内のキーワードが有効であること

> 注記：クエリの構文チェックによって判定されるのは、構文が有効かどうかだけです。クエリの関数名が有効かどうか、引数が有効化どうか、属性値が適切かどうかは、チェックされません。したがって、構文チェックを通っても、評価されたときにクエリ式がエラーとなる可能性は残ります。

## 定義済みのスコープ

定義済みのオフライン保存と削除のスコープは、*CCM_HOME*¥etc¥soad ディレクトリ（Windows）または $CCM_HOME/etc/soad ディレクトリ（UNIX）の、XML ファイルに格納されています。ファイル名は、スコープ名を URL 化し、空白を %20 で置き換え、拡張子 ".xml" を付けたものです。

> 注記：*ccm_root* としてログインすれば、テキスト エディタを使用して、スコープ ファイルを編集、作成、削除できますが、可能な限り GUI または CLI を使用してください。

以下に、定義済みのスコープの例を 2 つ示します。

- リリースベースのスコープ
- プロジェクト階層ベースのスコープ

## リリースベースのスコープ

以下に、"All projects and related objects for a specified release" という 定義済みのスコープの、XML ファイルの内容を示します。

```
<?xml version="1.0" encoding='ISO-8859-1'?>

<soadfscope version="1">
  <predefined>TRUE</predefined>
  <role></role>
  <parameter>
    <label>Release value</label>
  </parameter>
  <object></object>
  <query>release='%1' and cvtype!='problem'</query>
  <expansion_rule>Folder's tasks</expansion_rule>
  <expansion_rule>Project's folders</expansion_rule>
  <expansion_rule>Project's tasks</expansion_rule>
  <expansion_rule>Task's objects</expansion_rule>
  <exclusion_rule>Baseline projects used by other non-static projects
  </exclusion_rule>
  <exclusion_rule>Folders used by other projects</exclusion_rule>
  <exclusion_rule>Objects associated with other non-automatic tasks
  </exclusion_rule>
  <exclusion_rule>Projects used by other baselines</exclusion_rule>
  <exclusion_rule>Tasks used by other baselines</exclusion_rule>
  <exclusion_rule>Tasks used by other folders</exclusion_rule>
  <exclusion_rule>Tasks used by other projects</exclusion_rule>
  <exclusion_query></exclusion_query>
  <package_name>All projects and related objects for Release %1 saved
  on %date</package_name>
</soadfscope>
```

最初に、指定したリリースを有するすべてのオブジェクトのクエリを実行して、当初のオブジェクト リストを作成します。

次に、見つかった各プロジェクトに関して、展開ルールにより、以下のものをリストに追加します。

- 各プロジェクトのフォルダとタスク
- 各フォルダのタスク
- 各タスクに関連するオブジェクト

最後に、除外ルールを適用して、削除しないフォルダ、タスク、ベースラインに影響を及ぼさないようにするために、オブジェクト リストから以下のものを削除します。

- 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 プロジェクト
- 他のベースラインによって使用されているプロジェクト
- 他のプロジェクト内で使用されているフォルダとタスク
- 他のベースラインまたはフォルダによって使用されているタスク
- 他の非自動タスクに関連するオブジェクト

## プロジェクト階層ベースのスコープ

以下に、"Project hierarchy and related folders and tasks" という、定義済みのスコープの XML ファイルの内容を示します。

```
<?xml version="1.0" encoding='ISO-8859-1'?>

<soadfscope version="1">
  <predefined>TRUE</predefined>
  <role></role>
  <parameter>
    <label>Project objectname</label>
  </parameter>
  <object>%1</object>
  <query></query>
  <expansion_rule>Folder's non-automatic tasks</expansion_rule>
  <expansion_rule>Project's folders</expansion_rule>
  <expansion_rule>Project's non-automatic tasks</expansion_rule>
  <expansion_rule>Project's recursive members</expansion_rule>
  <exclusion_rule>Baseline projects used by other non-static projects
  </exclusion_rule>
  <exclusion_rule>Folders used by other projects</exclusion_rule>
  <exclusion_rule>Objects associated with other non-automatic tasks
  </exclusion_rule>
  <exclusion_rule>Projects used by other baselines</exclusion_rule>
  <exclusion_rule>Tasks used by other baselines</exclusion_rule>
  <exclusion_rule>Tasks used by other folders</exclusion_rule>
  <exclusion_rule>Tasks used by other projects</exclusion_rule>
  <exclusion_query></exclusion_query>
  <package_name>Project hierarchy %1 saved on %date</package_name>
</soadfscope>
```

最初に、当初のオブジェクト リストには、オブジェクト名によって指定されたプロジェクトだけが含まれています。

次に、指定したプロジェクトに関して、展開ルールにより、以下のものを再帰的にリストに追加します。

- プロジェクトのフォルダ、非自動タスク、および再帰的なメンバー
- 各フォルダの非自動タスク
- 各タスクに関連するオブジェクト

最後に、除外ルールを適用して、削除しないフォルダ、タスク、ベースラインに影響を及ぼさないようにするために、オブジェクト リストから以下のものを削除します。

- 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
- 他のベースラインによって使用されているプロジェクト
- 他のプロジェクトによって使用されているフォルダ
- 他のベースライン、プロジェクト、フォルダによって使用されているタスク
- 他の非自動タスクに関連するオブジェクト

# トリガ

## 通知トリガの使用

通知機能により、オブジェクトの状態が変化したときに、または変更依頼が提出されたときに、呼び出すプログラムを定義できます。プログラムの記述言語は問いません（シェルスクリプトまたはバッチ ファイルでも）。また、リテラル値／トリガが起動されたオブジェクトの属性値を渡すこともできます。

以下のトピックについて説明します。

- トリガ定義ファイルのフォーマットと説明
- プログラム
- メッセージ
- 例

## トリガ定義ファイルのフォーマットと説明

以下にトリガ定義ファイルの例を示し、その後で個々のファイルのコンポーネントを説明します。

```
trans_type
{
   type
   type
     .
     .
     .
   {
      status  program  arg1 arg2 ... argN;
      status  program  arg1 arg2 ... argN;
              .
              .
              .
   }
   .
   .
   .
}
.
.
.
```

コンポーネントの説明

`trans_type`

いつプログラムを呼び出すべきかを示します。

`pretransition`

遷移時になんらかの操作が行われる前に、プログラムが呼び出されます。これは、残りのトリガの処理に進む前に、非 PT のオブジェクトの条件を検証する場合に適用可能です。遷移前作業が失敗した場合は、遷移全体が失敗します。

`posttransition`

遷移が正常に完了した後で、プログラムが呼び出されます。これは非 PT オブジェクトの遷移に関して適用可能です。事後遷移の戻り状態は無視されます。

`pt_transition`

PT オブジェクトの遷移が完了した後で、プログラムが呼び出されます。PT 関連オブジェクトの遷移のために、プログラムは特別に設定されています。

`type`

オブジェクトのタイプを定義します。これは、有効なデータベース タイプである必要があります。'*' はすべてのタイプを示します。全タイプの遷移が定義された場合、特定遷移が呼び出される前に、それが実行されます。

`status`

オブジェクトの状態を定義します。有効な状態（*working*、*integrate*、*test*、*sqa*、*released*、*public* など）である必要があります。

`Arguments`

二重引用符で囲まれた引数はリテラルとして扱われ、引用符のないリテラルとしてトリガ プログラムに渡されます。

"=" 記号で始まる引数はテンプレート ファイルとして扱われます。そのファイルはメモリに読み込まれ、オブジェクトの属性値を使用してキーワードの置換が行われ、変更されたファイルがディスクに書き込まれます。その後で、ファイルのパスがプログラムに渡されます。使用後のファイルの除去は、プログラム側で行われます。

他の引数は遷移中のオブジェクトの属性名であるとみなされます。属性の値は通知プログラムに対する引数として渡されます。引数の名前の前に '+' が付いている場合、属性がないか値が空であればトリガは発動されません。オブジェクトの状態が変化した後で、作成または設定される属性もあります。したがって、トリガが呼び出された時点では、属性がまだ存在してないか期待される値を保持していない場合があります。

複数行のテキスト引数（属性とリテラル）を渡すことはできません。属性 'source' が指定された場合、ソース ファイルまたはキャッシュ ファイルへのパスが渡されます。

キーワード dbid、database、current_user は、それぞれデータベース ID、データベースへの絶対パス名、現在の CM セッションを実行しているユーザーの名前で置き換えられます。

## プログラム

プログラムは、オブジェクトが格納されるデータベースの notify ディレクトリ、または実行エリアの CCM_HOME/bin/util ディレクトリにあります。セキュリティ上の理由から、トリガ定義ファイルには相対パスまたは絶対パスを指定できません。その代わりに、notify ディレクトリにラッパー プログラムを記述し、それによって notify ディレクトリにないプログラムを呼び出すことができます。

トリガがインターフェイス（Trig_ui.def）で実行されている場合、プログラムは現在のユーザーによって実行されます。トリガがエンジン（Trig_eng.def）で実行されている場合、プログラムは ccm_root ユーザーによって実行されます。このことのセキュリティ上の意味を考えてください。

> 注記：CCM コマンドはトリガ付きのプロジェクトではサポートされていません。トリガによって ccm コマンドを実行すると、セッションがハングアップします。

## メッセージ

プログラムは、stdout、stderr、終了状態 コードによって、Rational Synergy セッションに情報メッセージまたは指示を返すことができます。

### stdout

```
ATTR_SET:<name>:<type>:<value>
ATTR_ADD:<name>:<type>:<value>
```

上記の両コードは、指定された属性がまだ存在しない場合に、指定した名前とタイプの属性を作成します。ATTR_SET は既存の値を上書きしますが、ATTR_ADD はテキスト属性に情報を追加します。

```
MSG:<string>
```

<string> 内のメッセージがユーザーに表示されます。GUI からトリガが起動された場合、メッセージはダイアログ ボックスに表示されます。CLI からトリガが起動された場合、メッセージはコマンド ラインに表示されます

APPLY_ATTRS_ON_FAIL

一般に、プログラムがエラー メッセージを返した場合は、返された属性は適用はされません。上記の文字列が返された場合、返された状態がなんであれ、属性は適用されます。

IGNORE_FAIL

上記で述べたように、グループ内のなんらかのプログラムが失敗した場合、トランザクション全体が失敗します。上記のメッセージを返すと、プログラムが失敗した場合でもトランザクションの処理は続行されます。これは、プログラムが失敗したことを否定するものではありません。全体の結果に影響を及ぼさないようにするだけです。

これが true に設定されると、その他の行はすべて MSG として扱われます。

FAILED

このメッセージが示された場合、プログラムが失敗したと判断され、終了状態は無視されます。

<string>

stdout へのその他の出力はすべて、ログ ファイルおよびメッセージ パネルに出力されます。これが MSG: とは異なる点は、GUI からトリガが起動された場合、MSG: 出力はポップアップ ダイアログに表示されることです。

## stderr

stderr の出力はすべて、ログ ファイルとメッセージ パネルに表示されます。GUI からトリガが起動された場合はポップアップ ダイアログに表示されます。

トランザクションに対して複数のプログラムが実行されている場合、すべての出力が結合されて、1 つのメッセージとして表示されます。

### 終了状態

終了状態がゼロ（0）である場合、正常に終了したものとみなされます。遷移前に上記以外の状態、あるいは FAILED メッセージが返された場合、遷移へ進むことはできません。

## 例

トリガは UNIX と Windows のどちらでも実行可能ですが、以下に示す例では、Windows でクライアントが稼働している状況でのトリガ機能を示すために、Windows のバッチファイルを呼び出しています。

```
#  Before ANY object gets checked in to integrate state,
#  run the pretrans.bat script, passing the owner, object name,
#  and object version
pretransition
    {
    *
        {
        integrate pretrans.bat owner name version;
        }
    }

#  After ascii and csrc objects are created, call the psttrans.bat script,
#  passing the owner, name and version.
posttransition
    {
    ascii
    csrc
        {
        working psttrans.bat owner name version;
        }
    }

#  After a problem has been verified, call the verified.bat script,
#  only if the submitter_email attribute is present and not empty.
#  Pass the script the name of a file that has the keyword expanded
#  contents of the verified.tpl template.
pt_transition
    {
    problem
        {
        verified verified.bat +submitter_email =verified.tpl;
        }
    }
```

```
#  Here is an example of the template you might use with the trigger
#  for email problem submissions, notifying the remote sender of the
#  new problem number.
#  The keywords you may use are any attributes of the object in question,
#  or one of the following special keywords:
#   %dbid           DCM database id
#   %database       Database path
#   %current_user   Current user name
#   %rfc822_date    Date in the RFC822 format Wed, 13 Oct 99 02:20:24 PDT
#  Attributes or keywords may be specified in the format %name or %{name}
To: %submitter_email
From: %current_user
Subject: Problem Receipt Notification
Date: %rfc822_date

Problem Receipt Notification
_____________________________________________________________________________

Your problem report:

%problem_synopsis

was received and created as number %problem_number
in database %{database} by %{enterer}.
```

# ワークエリア

ワークエリアとは、ファイル システム内の領域であり、Rational Synergy がプロジェクト階層を書き込む場所です。したがって、ワークエリア内のプロジェクトのファイル構造はプロジェクトのデータベース ファイル構造と対応付けられています。

Windows では、ワークエリアにはデータベース オブジェクトのコピーが格納され、Rational Synergy によってデータベースとの同期が維持されます。

UNIX では、ワークエリアはリンクベース（データベース オブジェクトへのリンクを保持する）またはコピーベース（データベース オブジェクトのコピーを保持する）のどちらかです。どちらのタイプのワークエリアも、Rational Synergy によってデータベースの同期が維持されます。

以下のトピックでは、ワークエリアの機能について説明します。

- ワークエリアの更新方法
- ワークエリアの場所
- 絶対ワークエリアと相対ワークエリア
- ワークエリア パスの更新

このドキュメントで参照されるコマンド リファレンスの詳細については、Rational Synergy ヘルプを参照してください。

# ワークエリアの更新方法

プロジェクトの作成時または変更時に、Rational Synergy は自動的かつユーザーに意識されることなく、ワークエリアを更新します。たとえば、プロジェクトにメンバーを追加すると、新しいファイルでワークエリアが更新され、プロジェクトからメンバーを削除すると、対応するファイルがワークエリアから削除されます。ワークエリアを手作業で更新することもできます。

以下のトピックでは、Rational Synergy によりワークエリアの更新方法について説明します。

- コピーベースのワークエリアの更新
- リンクベースのワークエリアの更新
- ワークエリアの変更または再作成
- 複数ワークエリアの更新

## コピーベースのワークエリアの更新

コピーベースのワークエリアには、プロジェクト内のすべてのオブジェクトに関するソースのコピーが収められています。各ファイルのコピーが少なくとも 2 箇所、ワークエリア内とデータベース内に存在します。

管理ファイルにアクセスするたびに、Rational Synergy がファイルをチェックし、変更されたかどうか判断します。オブジェクトをチェックアウトしてワークエリア内のファイルに対して変更加えた場合、次回そのオブジェクトにアクセスした際、Rational Synergy が変更を反映してデータベースを自動更新します。ワークエリア内でチェックアウトされていないファイルを変更すると、ワークエリア内でそのファイルにアクセスしたときに、Rational Synergy がコンフリクトの存在を通知します。ワークエリアのコンフリクトを解決するためには、reconcile コマンド またはワークエリアの同期操作を参照してください。

Windows では、ワークエリアはすべてコピーベースです。UNIX でワークエリアをコピーベースにするには、リモート クライアント セッション中にプロジェクトを作成するか（`ccm start -rc` を使用）、`ccm work_area -cb` オプションを使用するか、Properties ダイアログ ボックスで Symbolic Links オプションまたは Copies オプションを選択します。

以下のいずれかの要件が満たされる場合に、UNIX でコピーベースのワークエリアを使用します。

- クライアント ホストからデータベースへの NFS マウントがない。
- 社外での作業など、データベースと切断した状況で作業する必要がある。
- シンボリック リンクを適切に使用できないツールを使用している。

## リンクベースのワークエリアの更新

デフォルトでは、UNIX クライアント ユーザーは リンクベースのワークエリアで作業します。このワークエリアをリンクベースと呼ぶ理由は、ワークエリア内のファイルは Rational Synergy データベース内のファイルへのシンボリック リンクであるからです。リンクベースのワークエリア内のファイルを編集すると、データベース ファイルに変更が直ちに反映されます。なぜなら、ファイルのコピーではなく、ファイルそのものに対して作業しているからです。

UNIX クライアントで実行している場合、コピーベースまたはリンクベースのどちらかのワークエリアを使用できます。リモート クライアントを起動する UNIX クライアントでは、コピーベースのワークエリアを使用します。

> 注記：ワークエリアでシンボリック リンクを解除するツールまたはコマンドを使用する場合、プロジェクトを同期させて、ワークエリアでの変更を反映してデータベース オブジェクトを更新する必要があります。

## ワークエリアの変更または再作成

ワークエリアを検証または変更するために、以下の操作を行うことができます。

- ワークエリアとデータベースのコンフリクトがあるか調べる（ワークエリアの操作を行う前に）

管理ワークエリア ファイルをデータベース ファイルと比較したい場合、reconcile コマンド を参照してください。

- リンクベースからコピーベースに、またはその逆に変更する。

  ワークエリアをリンクベースからコピーベースに、またはその逆に変更したい場合、`ccm work_area -cb` オプションを使用するか、Properties ダイアログ ボックスの Symbolic Links オプションか Copies オプションを選択して、ワークエリアを自動的に変換します。リンクベースまたはコピーベースのワークエリアを変換する方法については、reconcile コマンド を参照してください。

- 削除する

  ワークエリアを削除できます（たとえば、不要な非管理ファイルを削除するため）。ワークエリアでの変更を反映してデータベースが更新されていることを確実にするため、削除する前に必ずワークエリアを同期させます。reconcile コマンドを参照してください。

- 再作成する

  ワークエリアからファイル削除し、データベース バージョンをワークエリアに戻したい場合、sync コマンド を使用して、ワークエリアを再作成できます。

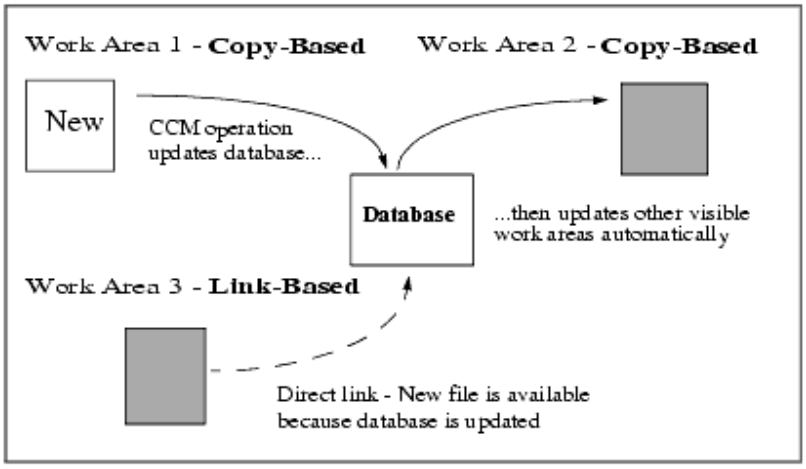
## 複数ワークエリアの更新

ヘッダー ファイルやライブラリのようなオブジェクトは、複数プロジェクトで使用できるため、それらのオブジェクトが複数のワークエリアに存在する可能性があります。Rational Synergy は、複数の場所にあるそれらのファイルを、アクセス時に更新することで同期しています。

コピーベースのワークエリアでは、ファイルが使用されるワークエリアにはそのコピーが置かれています。コピーベースのワークエリア内のファイルに変更を加えた場合は、ローカル コピーだけが変更されています。次回そのファイルにアクセスすると、Rational Synergy が変更を検出し、データベースと、ファイルが使用されている他の見えるワークエリアで、ファイルを更新します。

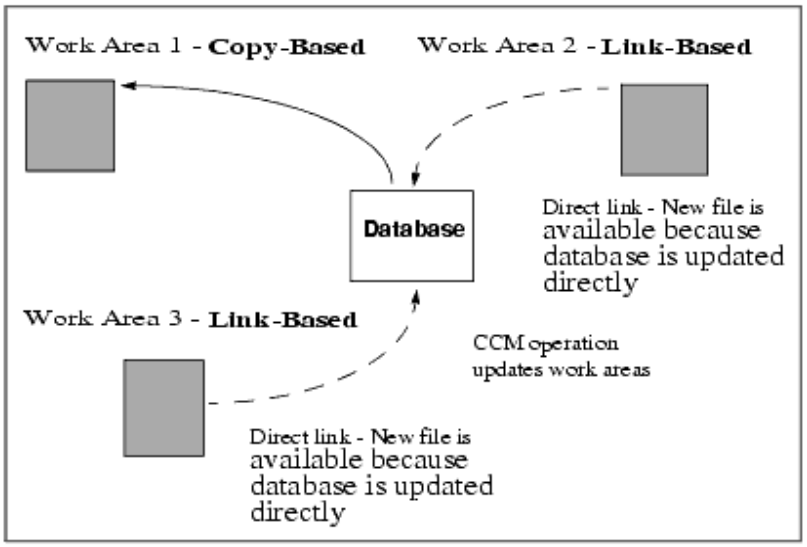
下図の例は、複数プロジェクトで使用されているコピーベースのファイルが更新されたときに、更新がどのように行われるかを示しています。四角は、単一ソース ファイルのコピーまたはソース ファイルへのリンクを表します。見えるワークエリアのみが更新されます。

図：


Work Area 1 - Copy-Based
Work Area 2 - Copy-Based
New
CCM operation
updates database...
Database
...then updates other visible
work areas automatically
Work Area 3 - Link-Based
Direct link - New file is available
because database is updated

2 番目の例は、複数プロジェクトで使用されているリンクベースのファイルが更新されたときに、どのように更新が行われるかを示しています。四角は、単一ソース ファイルのコピーまたはソース ファイルへのリンクを表します。リンクを通じて変更が行われた場合、ワークエリアが見えるクライアントからオブジェクトにアクセスするまで、コピーベースのワークエリア（たとえば、下図の Work Area 1）は更新されません。

図 :




複数プロジェクトで使用されているファイルがあり、同時に複数のワークエリアでそのファイルを変更した場合、Rational Synergy から変更されたファイルにアクセスしたときに、コンフリクトが存在することが通知されます。作業を続けるには、コンフリクトを解決する必要があります。

## ワークエリアの場所

各プロジェクト バージョンには単一の ワークエリアの場所があります。その場所を変更できますが、ワークエリアは必ず単一の場所に維持されます。

- 同じプロジェクトを複数の場所に対して同期させたい場合、ワークエリアごとにプロジェクトの別バージョンをチェックアウトする必要がある。
- このワークエリアを複数のユーザーが利用できるようにしたい場合、すべてのユーザーが利用可能な場所に対して、プロジェクトの同期させる。

プロジェクトの別バージョンをチェックアウトする場合、ワークエリアのファイルを一覧表示するコマンドを使用してそのすべてを確認できます。以下に、Windows ワークエリアと UNIX ワークエリアの出力例を示します。

Windows:

```
Directory of C:¥ccm_wa¥ccmdb219
.             <DIR>     02/01/99  3:04p .
..            <DIR>     02/01/99  3:04p ..
demo-1        <DIR>     02/01/99  3:04p demo-1
demo-2        <DIR>     02/07/99 11:23p demo-2
demo-3        <DIR>     02/21/99  9:15a demo-3
```

UNIX:

```
drwxr-xr-x   4 bill      develop       2048 Aug 31 14:37 ./
drwxr-xr-x   3 bill      develop        512 Aug 31 14:37 ../
-r--r--r--   1 bill      develop         63 Aug 31 14:37 .ccmwaid.inf
drwxr-xr-x   2 bill      develop       2560 Aug 31 14:37 demo-1/
drwxr-xr-x   2 bill      develop       1024 Aug 31 14:37 demo-2/
drwxr-xr-x   2 bill      develop        512 Aug 31 14:37 demo-3/
```

正しいプロジェクトに進むためには、方向を適切なプロジェクトおよびバージョン（*project-version*）に変更する必要があります。上記の出力例では、demo プロジェクトに demo-1、demo-2、demo-3 という 3 つのバージョンがあります。

単一場所のワークエリアを使用した場合、あるプロジェクトがローカル Windows ファイル システムに対して同期された場合、UNIX セッションからこのワークエリアが見えなくなります。逆も同様です。プラットフォームが異なる場合は、プロジェクトの別バージョンを使用してください。

## 絶対ワークエリアと相対ワークエリア

Rational Synergy では、絶対ワークエリアと相対ワークエリアの 2 種類に対応しています。以下のトピックでは、2 つのタイプのワークエリアについて説明します。

- 絶対ワークエリア
- 相対ワークエリア

### 絶対ワークエリア

絶対ワークエリアは個別のディレクトリ階層として存在します。つまり、絶対サブプロジェクトのワークエリア パスは親プロジェクトのパスの下にある必要はありません。

サブプロジェクトがどこにあっても同期が可能ですが、デフォルトでサブプロジェクトは以下のパスのホーム ディレクトリにあります。

Windows：*home_directory*¥ccm_wa¥*database_name*¥*project-version*

UNIX： ~/ccm_wa/*database_name*/*project-version*

この場所にあると、プロジェクトのワークエリアが同じディレクトリ構造になくても、他のプロジェクトが絶対サブプロジェクトを見つけて使用できます。

たとえば、bar-1 が ccm_tools データベースの foo-1 のサブプロジェクトであり、(UNIX のみ) シンボリック リンクを使用していないものとします。bar-1 が絶対ワークエリアであれば、以下のように見えるでしょう。

Windows:

```
c:¥ccm_wa¥ccm_tools
   foo-1¥
      foo¥
         a.c
         b.c

   bar-1¥
      bar¥
         c.c
```

UNIX:

```
/users/bill/ccm_wa/ccm_tools
    foo-1/
        foo/
            a.c
            b.c
    bar-1/
        bar/
            c.c
```

絶対プロジェクトをサブプロジェクトとして、複数回使用できます。Rational Synergy では、複数の開発者が共同使用する目的で、絶対プロジェクトを共有ファイル システムに置くことを想定しています。共有の製品、ライブラリ、ヘッダ ファイルを格納する外部プロジェクトなどにとって、これは有用です。

UNIX では、シンボリック リンクを使用している場合、絶対サブプロジェクトは親プロジェクトのワークエリア内のサブディレクトリに見えます。たとえば、絶対サブプロジェクトの場合、プロジェクト階層は以下のように見えるでしょう。

```
/users/bill/ccm_wa/ccm_tools
   foo-1/
      foo/
         a.c
         bar -> /users/bill/ccm_wa/ccm_tools/bar-1/bar
   bar-1/
      bar/
```

```
        c.c
```

Windows クライアントでは、サブプロジェクトが絶対であるとき、親プロジェクトのワークエリアを見ても、サブプロジェクトがサブディレクトリとして見えません。これは、Windows ではシンボリック リンクはサポートされていないからです。

work_area コマンド操作を使用して、サブプロジェクトを絶対から相対に変更できます。

## 相対ワークエリア

他のプロジェクトによって使用された場合、相対サブプロジェクトは親プロジェクトのワークエリア内にサブディレクトリであるかのように存在します。パフォーマンス上の理由からコードをサブプロジェクトに構成しなければならない場合、あるいは make ファイルまたはツールが相対パスを使用するよう記述されている場合に、これは便利です。

以下に、上記の例で示したものと同じプロジェクトに関する、ワークエリアの例を下に示します。この例では、`bar-1` サブプロジェクトは絶対ではなく相対です。

Windows:

```
c:¥ccm_wa¥ccm_tools
   foo-1¥
      foo¥
         a.c
         b.c
      bar¥
         c.c
```

UNIX:

```
/users/bill/ccm_wa/ccm_tools
    foo-1¥
        foo¥
           a.c
           b.c
           bar¥
               c.c
```

相対サブプロジェクトが静的（修正不可）であれば、複数のプロジェクトでサブプロジェクトとして使用できます。修正可能な相対サブプロジェクトは 1 回しか使用できません。なぜなら、それは親プロジェクトのワークエリア内に存在し、その場所に対してのみ同期されるからです。修正可能な相対プロジェクトを複数の場所で使用したい場合は、プロジェクトの複数バージョンを使用します。

Windows と UNIX のどちらのクライアント上でも、作成時のデフォルトでプロジェクトは絶対になっています。プロジェクトの新しいバージョンをチェックアウトしたときに、新バージョンのワークエリアが相対となるのは、相対プロジェクトからチェックアウトした場合だけです。それ以外の場合は、新しいバージョンのワークエリアは絶対となります。

プロジェクト階層内の make ファイルが、相対パスによってサブプロジェクトがサブディレクトリであるかのように、サブプロジェクトのメンバーを参照する場合、またはサブプロジェクトへのシンボリック リンクを使用できない場合、プロジェクトを相対にしておく必要があります。プロジェクト階層内の make ファイルが、サブプロジェクトが無関係のサブディレクトリ構造にあるかのように、サブプロジェクトのメンバーを参照する場合、プロジェクトは絶対とすることができます。

絶対プロジェクトと相対プロジェクトのいずれかまたは両者の組み合わせでも作業できるように、make ファイルを変更できます。あるいは、既存の make ファイルがワークエリアのディレクトリ構造を認識できるようにサブプロジェクトを絶対または相対に設定できます。

## ワークエリア パスの更新

ファイル システム内の別の場所に合わせてプロジェクトの同期を取りたい場合は、ワークエリア パスを変更する必要があります。また、データベースをコピーまたは移動したり、バージョン区切り文字を変更した場合は、そのパスの設定を変更する必要があります。

以下のトピックでは、ワークエリア パスと必要な変更について説明します。

- ワークエリア パスの要素
- ワークエリアの移動
- データベースの変更
- セキュリティと可視性の問題
- work_area コマンドの構文

### ワークエリア パスの要素

典型的なプロジェクト ワークエリア パスはユーザーのホーム ディレクトリ、ワークエリア サブディレクトリ（たとえば、`ccm_wa`）、データベース名、プロジェクトの名前とバージョン、およびプロジェクトのルート ディレクトリで構成されます。

典型的な構文は以下のようになります。

```
Windows: home_dir¥ccm_wa¥database_name¥project_name-version¥project_name
UNIX:    home_dir/ccm_wa/database_name/project_name-version/project_name
```

たとえば、以下に示すパスは、`ccmdb219` データベース内のユーザー `bill` の `baselib-bill` プロジェクトのものです。

```
Windows:  c:¥ccm_wa¥ccmdb219¥baselib-bill¥baselib
UNIX:    /users/bill/ccm_wa/ccmdb219/baselib-bill/baselib
```

## ワークエリアの移動

プロジェクトのワークエリアを移動するには、`ccm work_area -setpath` コマンド、または Properties ダイアログ ボックスを使用します。これらのインターフェイスでは、ワークエリア パスに *`project_name<version_delimiter>project_version`* が自動的に付加されます。ワークエリア パスのこの部分をユーザーが指定する必要はありません。

> 注記 : UNIX のリンクベースのワークエリアは、データベース パス内の対応するファイルへのシンボリック リンクを使用した、プロジェクトの管理オブジェクトです。データベースを移動する場合は、新しいデータベース パス内に位置するファイルを指すように、ワークエリア内のすべてのシンボリック リンクを更新する必要があります。

## データベースの変更

データベースの名前変更、移動、コピーのいずれかを行う場合（たとえば、`ccmdb cp` または `ccmdb unpack` コマンドを使用）、またはデータベースのバージョン区切り文字を変更する場合、ワークエリア パスを更新する必要があります。ワークエリア パスが古くなったプロジェクトを識別して更新するには、`work_area` コマンドの `-find` オプションを使用します。t

また、各ワークエリアには識別ファイル `ccmwaid.inf` （`Windows`）または `.ccmwaid.inf`（`UNIX`）があり、ここにプロジェクトのデータベースへのパス名が格納されています。Rational Synergy はこのファイルを使用して、1 つのデータベースのみが当該ワークエリアを更新するようにします。データベースを移動すると、ワークエリア ID ファイルが別のデータベース用のように見え、ワークエリアが使用できなくなります。ワークエリア ID ファイルを識別してデータベース ロケーションでの変更を確認するには、`work_area` コマンドの `-dbpath` オプションを使用します。

## セキュリティと可視性の問題

ワークエリア パスを変更するためには、プロジェクトの Rational Synergy 書き込みアクセス権限が必要です。静的プロジェクトを更新するためには、*ccm_admin* ロールを持っている必要があります。また、どのプロジェクトを更新する場合も、ワークエリアに対するファイル システムの書き込みアクセス権限が必要です。

新しいワークエリア パスは Rational Synergy クライアントから見えなければなりません。ワークエリアが Windows と UNIX の両方のファイル システム上に存在する場合、それらのワークエリアを更新するためには、別々の Rational Synergy セッションを使用する必要があります。Windows のワークエリアを更新するためには、Windows ファイル システムが見える Windows クライアント セッションを使用します。UNIX のワークエリアを更新するためには、UNIX ファイル システムが見える UNIX クライアント セッションを使用します。

ワークエリア ID ファイルを更新するためには、ファイル システム内のワークエリアに対する書き込みアクセス権限が必要です。

## **work_area** コマンドの構文

以下では、`-find` および `-dbpath` オプションの構文例のみを示します。詳細については、work_area コマンド を参照してください。

- 検出されたすべてのプロジェクトのワークエリア内で、*`find_str`* を *`new_str`* に置き換える（指定した適用範囲内、または `-p` オプションで指定した範囲内）。

  ```
  ccm work_area -find find_str -replace new_str
  ```

  `-reg` または `-regexp` も指定した場合、*`find_str`* と *`new_str`* はともに正規表現と解釈されます。

- *`old_path`* 引数で指定したデータベース パス名が格納されるワークエリア ID ファイルを持つプロジェクトを検索し、それらのファイルを現在のデータベースのパス名で更新する。

  ```
  ccm work_area -dbpath old_path
  ```

  *`old_path`* オプションは正規表現とは解釈されません。`-dbpath` オプションは、検出されたプロジェクトのワークエリア ID ファイルを、現在のデータベースのパス名で更新します。このオプションはデータベースを移動するときにのみ使用します。UNIX のリンクベースのワークエリア上でこのオプションを使用すると、新しいデータベース ロケーションに合わせてリンクを更新するために、ワークエリアが同期されます。`-nosync` オプションも指定した場合、同期は延期されますが、ワークエリアを使用する前に、手作業で同期させる必要があります。

新しいデータベースの存在を示すために、`-find` と `-replace` に、`-new` オプションを付加できます。これによって、`-find` によって指定された元のワークエリアはこのセッションでは見えないので無視すべきであることを意味します。新しいパス名にデータベースがアンパックされており、元のデータベースのワークエリアを無視したい場合、あるいは元のデータベースが存在しない場合に、このオプションが役立ちます。`-new` オプションを追加しなかった場合、このコマンドは、見えるワークエリアを持つプロジェクトにのみ適用されます。

どのプロジェクトが更新されたかを示すために、`-find` または `-dbpath` に `-show` オプションを付加できます。`-find` および `-replace` と一緒に `-show` を使用した場合、置換パス名が表示されます。正規表現（`-regexp`）を用いて `-find` と `-replace` を使用している場合、これは非常に役に立ちます。

### 例

- 見えるワークエリアとパスを持ち、`"-"` 文字を含む *working* プロジェクトをすべて検索し、`"-"` を `"~"` に変更する。

```
ccm work_area -find "-" -replace "~"
```

データベース区切り文字を "-" から "~" に変更した後で、このコマンドを使用できます。すべてのワークエリアを変更するのに十分なセッションから、このコマンドを実行する必要があります。すべてのワークエリアが 1 つのセッションから見える場合には、セッションは 1 つで十分です。しかし、Windows と UNIX の両方のワークエリアがある場合、それぞれのクライアントからこのコマンドを実行する必要があります。

- 見えるワークエリアとパスを持ち、"-" 文字を含む *prep* プロジェクトをすべて検索し、"-" を "~" に変更する。

  ```
  ccm work_area -find "-" -replace "~" -scope prep
  ```

  データベース区切り文字を "-" から "~" に変更した後で、このコマンドを使用できます。ビルド マネージャとして作業している間に、すべてのビルド管理ワークエリアを変更するのに十分なセッションから、このコマンドを実行する必要があります。すべての *prep* ワークエリアが 1 つのセッションから見える場合には、セッションは 1 つで十分です。しかし、Windows と UNIX の両方のビルド管理ワークエリアがある場合、それぞれのクライアントからこのコマンドを実行する必要があります。

- UNIX で、/vol/acrel5/ccmdb/ccm_platform データベースのワークエリアを持つ *working* プロジェクトをすべて検索し、現在のデータベース ロケーションへのパスでワークエリア ID ファイルを更新する。

  ```
  ccm work_area -dbpath /vol/acrel5/ccmdb/ccm_platform
  ```

  移動されたが古いワークエリア パスを使用可能なデータベースを更新するために、このコマンドを使用します。すべてのワークエリアを変更するのに十分なセッションから、このコマンドを実行する必要があります。すべてのワークエリアが 1 つのセッションから見える場合には、セッションは 1 つで十分です。しかし、Windows と UNIX の両方のワークエリアがある場合、それぞれのクライアントからこのコマンドを実行する必要があります。

- 文字列 platform を含むパスを持つ working プロジェクトをすべて検索し、文字列を services に変更する。

  ```
  ccm work_area -find platform -replace services -new
  ```

  新しい名前にアンパックまたはコピーされたデータベースを更新するために、このコマンドを使用します。-new オプションにより、新しいワークエリアがすべて作成されることに、注意してください。古いワークエリアは古いデータベースによって使用されているからです。古いワークエリア（たとえば、移動済みのデータベース）を再使用したい場合、データベースから、古いワークエリアが見えるようにするため -dbpath オプションを使用して最初にワークエリア ID ファイルを更新します。

## 正規表現の例

引数 *find_str* と *new_str* を正規表現として work_area コマンドに解釈させるためには、-reg または -regexp のいずれかのオプションを使用します。詳細については 正規表現 を参照してください。

正規表現は役に立ちますが、以下のような制約があります。

- Windows クライアントでは、ディレクトリ名に円記号が使われるが、正規表現内の円記号はエスケープ文字または置換指定の一部として解釈される可能性がある。
- 引数 *find_str* または *new_str* を引用符で囲むと、引用符が UNIX シェルと Rational Synergy コマンド ライン プロセッサの両方によって解釈され（Windows クライアント上でも）、コマンド ライン処理が混乱する可能性がある。

以下に、正規表現を使用したワークエリア パスの変更の例をいくつか示します。

- c:¥ccm_wa¥bill45¥ にあるワークエリア パスをすべて、c:¥ccm_wa¥ に短縮する。

  ```
  ccm wa /find "bill45¥¥¥¥" /replace "" /reg /p
   Checking work area paths for replacement...
   1 project(s) will be checked.
   Setting path for work area of 'hsai~1' to 'c:¥ccm_wa¥hsai~1'...
   1 project work area path(s) were updated:
       'hsai~1': 'c:¥ccm_wa¥hsai~1'
  ```

  パスからディレクトリを削除するときは、関連する円記号も含めます。前に付ける円記号の指定は簡単ですが、以下の例では後に円記号が付いています。

  ```
  bill45¥.
  ```

  Rational Synergy のコマンド プロセッサは bill45 の前に付いている開き引用符を見つけます。そして、次の引用符の前に円記号を検出すると、引用符を閉じ引用符ではなく引数の一部とすることを示すエスケープ文字であると解釈します。したがって、円記号と解釈させるには、円記号の前にエスケープ用の円記号をもう 1 つ付ける必要があります。また、Rational Synergy のコマンド処理の結果得られた式（bill45¥）は、正規表現プロセッサによって、2 つの連続した円記号を追加してエスケープしない限り、対応する置換構造文字のない円記号として誤って解釈されます。

  引用符で囲んだディレクトリを複数指定する場合、末尾の閉じ引用符の直前には末尾の円記号は 4 つのみ使用します。引数の途中に入れるディレクトリの円記号は 1 回でエスケープできます。それにより、正規表現プロセッサはバックスラッシュを置換構造指定の開始であると解釈しなくなります。

  ```
  ccm wa /find "ccm_wa¥¥bill45¥¥¥¥" /replace "" /reg /p hsaw~1
  Checking work area paths for replacement...
   1 project(s) will be checked.
   Setting path for work area of 'hsaw~1' to 'c:¥users¥bill¥hsaw~1'...
   1 project work area path(s) were updated:
       'hsaw~1': 'c:¥users¥bill¥hsaw~1'
  ```

- 後ろに付ける円記号を使用し、ワークエリア パスを c:¥ccm_wa¥+bill45¥hsaw~1 から c:¥ccm_wa¥hsaw~1 に短縮することで、先頭に特殊文字があるディレクトリをパスから削除する。

```
ccm wa /find "¥+bill45¥¥¥¥" /replace "" /reg /p hsaw~1
 Checking work area paths for replacement...
 1 project(s) will be checked.
 Setting path for work area of 'hsaw~1' to 'c:¥ccm_wa¥hsaw~1'...
 1 project work area path(s) were updated:
     'hsaw~1': 'c:¥ccm_wa¥hsaw~1'
```

- 前に付ける円記号を使用して、先頭に特殊文字があるディレクトリをパスから削除する。

```
ccm wa /find "¥¥¥+bill45" /replace ""  /reg /p hsaw~1
 Checking work area paths for replacement...
 1 project(s) will be checked. Setting path for work area of 'hsaw~1' to
 'c:¥ccm_wa¥hsaw~1'...
 1 project work area path(s) were updated:
     'hsaw~1': 'c:¥ccm_wa¥hsaw~1'
```

一般に、*find_str* または *new_str* に特殊文字または空白がない限り、(非 UNIX 環境においては) 引用符は必要はありません。以下に、引用符を使用しない正規表現の例を示します。

- ワークエリア パスを c:¥ccm_wa¥bill45¥junk~1 から c:¥temp¥ccm¥bill45¥junk~1 に変換する。

```
ccm wa /find users¥¥bill¥¥ccm_wa¥¥ /replace temp¥¥ccm¥¥ /reg /p junk~1
 Checking work area paths for replacement...
 1 project(s) will be checked.
 Setting path for work area of 'junk~1' to
 'c:¥temp¥ccm¥bill45¥junk~1'...
 1 project work area path(s) were updated:
     'junk~1': 'c:¥temp¥ccm¥bill45¥junk~1'
```

この場合、エスケープすべき唯一の文字は、円記号です。正規表現プロセッサは通常はこの文字を置換構造指定の開始と解釈するからです。

置換構造指定をエスケープしない 1 つの共通の理由は、UNIX ファイル システムでは大文字と小文字が区別されることです。

- UNIX で、ワークエリアを持つ 3 つのプロジェクトを返す ccm query コマンドを実行した後で、パス名の一部を小文字から先頭文字のみ大文字に変換する。

*ccm_admin* ロールのユーザーがディレクトリ命名の基準を変更した場合に、これが必要になることがあります。

```
 pc-1: /users/bill/ccm_wa/owner/pc-1
 pi-1: /users/bill/ccm_wa/static/pi-1
 pw-1: /users/bill/ccm_wa/owner/pw-
```

UNIX シェルが "*" などの特殊文字を処理することを防止するために、引数を引用符で囲んでいます。また、円記号もエスケープしています（この場合は必須ではありませんが）。かっこは後で置換を行うための最初と 2 番目の式を明示する働きをします。選択セット（@）演算子はクエリの結果を指しています。

```
ccm wa -find "/users/bill/ccm_wa/([^/]+)/(.*)" -replace ¥
"/users/bill/ccm_wa/¥¥u¥¥1/¥¥2" -reg @
    Checking work area paths for replacement...
    3 project(s) will be checked.
    Setting path for work area of ‘pc-1’ to ‘/users/bill/ccm_wa/Owner/pc-1’
    . . .
    Setting path for work area of ‘pi-1’ to ‘/users/bill/ccm_wa/Static/pi-1’
    . . .
    Setting path for work area of ‘pw-1’ to ‘/users/bill/ccm_wa/Owner/pw-1’
    . . .
    3 project work area path(s) were updated:
        ‘pc-1’: ’/users/bill/ccm_wa/Owner/pc-1’
        ‘pi-1’: ’/users/bill/ccm_wa/Static/pi-1’
        ‘pw-1’: ’/users/bill/ccm_wa/Owner/pw-1’
```

- Windows 上、ワークエリアを持つ 3 つのプロジェクトを返す ccm query コマンドを実行した後で、パス名の一部の大文字の頭文字を小文字に変換する。

  *ccm_admin* ロールのユーザーがディレクトリ命名の基準を変更した場合に、これが必要になることがあります。

  ```
  pc-1: c:¥bill¥ccm_wa¥Owner¥pc-1
  pi-1: c:¥bill¥ccm_wa¥Static/pi-1
  pw-1: c:¥bill¥ccm_wa¥Owner/pw-1
  ```

  シェルが '*' などの特殊文字を処理することを防止するために、引数を引用符で囲んでいます。また、円記号もエスケープしています（この場合は必須ではありませんが）。かっこは後で置換を行うための最初と 2 番目の式を明示する働きをします。選択セット（@）演算子はクエリの結果を指しています。

  ```
  ccm wa /find "¥¥bill¥ccm_wa¥¥([^¥¥]+)¥¥(.*)" /replace
 "¥¥bill¥ccm_wa¥¥¥1¥¥¥1¥¥¥2" /reg @
    Checking work area paths for replacement...
    3 project(s) will be checked.
    Setting path for work area of ‘pc-1’ to ‘C:¥bill¥ccm_wa¥owner¥pc-1’
    . . .
    Setting path for work area of ‘pi-1’ to ‘C:¥bill¥ccm_wa¥static¥pi-1’
    . . .
    Setting path for work area of ‘pw-1’ to ‘C:¥bill¥ccm_wa¥owner¥pw-1’
    . . .
    3 project work area path(s) were updated:
        ‘pc-1’: ‘C:¥bill¥ccm_wa¥owner¥pc-1’
        ‘pi-1’: ‘C:¥bill¥ccm_wa¥static¥pi-1’
        ‘pw-1’: ‘C:¥bill¥ccm_wa¥owner¥pw-1’
  ```

# ワークエリア コンフリクト

リコンサイル プロセスには 2 つのフェーズがあります。最初のフェーズはリコンサイル処理で、ワークエリアとデータベース ファイルを比較します。そして、自動的に解決できる場合は差分を解決します。コンフリクト状態にあると判定されたファイルについては、なにも措置は取られません。2 番目のフェーズはコンフリクト解決フェーズで、コンフリクトのリストをユーザーに提示します。その解決方法はユーザーが決定します。この処理は、バッチ モード（すべてのコンフリクトを同様の方法で解決する）または個別方式（選択されたファイルに応じて独自の解決法を選択する）で行うことができます。コンフリクトを未解決のまま残すこともできます。

リコンサイル処理により、ファイルが破棄、上書き、無視されることがあります。したがって、リコンサイルを実施したときになにが起こるかを、理解しておくことが重要です。コンフリクトを解決する方法を理解することも重要です。

コンフリクトがどのようにして検出され、コンフリクトを解決するための具体的な方法を選択したときになにが起こるかを理解できるように、以下に情報を提示します。以下のトピックについて説明します。

- コンフリクトのタイプ
- バッチ モードでコンフリクトを解決する方法
- コンフリクトの解決 - ワークエリアからバッチ モードでデータベースを更新
- コンフリクトの解決 - データベースからバッチ モードでワークエリアを更新
- コンフリクトの解決 - 手作業でのコンフリクトの選択と解決

## コンフリクトのタイプ

8 つのタイプのワークエリア コンフリクトを下に列挙して、それらが発生する状況を説明します。

1. 作業中のオブジェクトに対するワークエリアの変更

   作業中のオブジェクトがあるが、そのオブジェクトに対するワークエリアを変更します。ファイル システム内のファイルを変更したような場合です。

2. 作業中のオブジェクトに対するデータベースの変更

   作業中のオブジェクトがあるが、以下のいずれかによってデータベースが更新された場合です。
   - 作業中のオブジェクトが存在するワークエリア以外の場所
   - 2 つのワークエリアがあったとして、現在以外のワークエリア
   - Rational Synergy の管理外でデータベースのソース キャッシュ ファイルを更新

3. 静的なオブジェクトに対するワークエリアの変更

   ワークエリア内のファイルを書き込み可能モードに変更し、それから Rational Synergy の管理外でファイルを修正した場合です。

4. 静的なオブジェクトに対するデータベースの変更

   修正不可オブジェクトのデータベースのソース キャッシュ ファイルが Rational Synergy の管理外で更新された場合です。

5. 複数の場所でのオブジェクトの変更

   オブジェクト用のデータベースのソース キャッシュ ファイルおよびオブジェクトのワークエリア ファイルの両方が修正された場合です。オブジェクトは作業中または静的な状態のどちらかです。

6. ワークエリアからのファイルの欠落

   Rational Synergy の管理下にあるオブジェクト用のワークエリア ファイルがワークエリアから欠落しています。オブジェクトは作業中または静的な状態のどちらかです。

7. 非管理ファイル

   ワークエリア内のファイル／ディレクトリが Rational Synergy の管理下にありません。

8. ファイル エラー

   Rational Synergy の管理下にあるファイルがエラーを起こしています。このタイプのエラーは通常、リンクベースのワークエリアに関係しています。この種のコンフリクトに関しては、エラーを解決する方法を選択できません。この種のコンフリクトはリコンサイル プロセスによって特別に処理されます。取られる措置には、以下のような

ものがあります。
- データベースからワークエリアを更新する。
- ワークエリアからデータベースを更新する。
- ファイルをリンクし直す（UNIX のみ）。
- ファイルを削除する。

コンフリクトを解決しない選択肢もあります。

## バッチ モードでコンフリクトを解決する方法

コンフリクトが検出された後で、ケースバイケースまたはバッチ モードのいずれかで、それを解決できます。バッチ モードには、ワークエリアからデータベースを更新する方法と、データベースからワークエリアを更新する方法があります。バッチ モードを使用してコンフリクトを解決すると、すべてのコンフリクトが同じ方法で解決されます。たとえば、データベースからワークエリアを更新するオプションを選択した場合、すべてのワークエリア ファイルはデータベースからのファイルで更新されます。

下表に、バッチ モードから得られる結果をまとめています。

コンフリクト解決の結果 - バッチ モード

| コンフリクト タイプ | 結果 - ワークエリアからバッチ モードでデータベースを更新 | 結果 - データベースからバッチ モードでワークエリアを更新 |
|---|---|---|
| 1. 作業中のオブジェクトに対するワークエリアの変更 | ワークエリアからの変更でデータベース オブジェクトが更新される | データベースからのファイルでワークエリアのファイルが更新される |
| 2. 作業中のオブジェクトに対するデータベースの変更 | ワークエリアからの変更でデータベース オブジェクトが更新される | データベースからのファイルでワークエリアのファイルが更新される |
| 3. 静的なオブジェクトに対するワークエリアの変更 | データベース オブジェクトがチェックアウトされ、それからワークエリアからの変更で更新される | データベースからのファイルでワークエリアのファイルが更新される |
| 4. 静的なオブジェクトに対するデータベースの変更 | データベース オブジェクトがチェックアウトされ、それからワークエリアからの変更で更新される | データベースからのファイルでワークエリアのファイルが更新される |
| 5. 複数の場所でのオブジェクトの変更 | データベース オブジェクトがチェックアウトされ、それからワークエリアからの変更で更新される | データベースからのファイルでワークエリアのファイルが更新される |

コンフリクト解決の結果 - バッチ モード

| コンフリクト タイプ | 結果 - ワークエリアからバッチ モードでデータベースを更新 | 結果 - データベースからバッチ モードでワークエリアを更新 |
|---|---|---|
| 6. ワークエリアからのファイルの欠落 | ワークエリア内で見つからないデータベース オブジェクトはプロジェクト内で「未使用」にされる | 欠落しているファイルがデータベースからワークエリアにコピーされる |
| 7. 非管理ファイル | ワークエリアのファイルがデータベースに追加され、それからチェックアウトされる | 非管理ファイルがワークエリアから削除される |
| 8. ファイル エラー | 措置はコンフリクト タイプに応じて異なる ― エラーを解決することだけを指定可能 | 措置はコンフリクト タイプに応じて異なる ― エラーを解決することだけを指定可能 |

詳細については、コンフリクトの解決 - ワークエリアからバッチ モードでデータベースを更新 および コンフリクトの解決 - データベースからバッチ モードでワークエリアを更新 を参照してください。

## コンフリクトの解決 - ワークエリアからバッチ モードでデータベースを更新

バッチ モードでの `Update database from work area`（ワークエリアからのデータベースの更新）は、ワークエリアの状態からデータベースを更新することにより、すべてのコンフリクトを解決します。コンフリクトの内容が異なるごとに、コンフリクト解決方法があります（コンフリクトのタイプ を参照）。

以下に、ワークエリアからバッチ モードでデータベースを更新することによりコンフリクトを解決した場合に、なにが起こるかを示します。

1. 作業中のオブジェクトに対するワークエリアの変更

   データベース オブジェクトのソース属性が更新されます（ワークエリアの変更が受け入れられる）。

2. 作業中のオブジェクトに対するデータベースの変更

   ワークエリア ファイルの内容でデータベース オブジェクトのソース属性が更新されます（データベースの変更が破棄される）。

3. 静的なオブジェクトに対するワークエリアの変更

   データベース オブジェクトがチェックアウトされ、ソース属性が更新されます（ワークエリアの変更が受け入れられる）。

4. 静的なオブジェクトに対するデータベースの変更

   データベース オブジェクトがチェックアウトされ、ワークエリア ファイルの内容でソース属性が更新されます（データベースの変更が破棄される）。

5. 複数の場所でのオブジェクトの変更

   データベース オブジェクトがチェックアウトされ、ワークエリア ファイルの内容でソース属性が更新されます。

6. ワークエリアからのファイルの欠落

   欠けているワークエリア ファイル／ディレクトリに関連するデータベース オブジェクトについて、使用解除操作を実行します。

7. 非管理ファイル

   存在するプロジェクト内またはディレクトリのもとに、ファイル／ディレクトリを作成します。これらのファイルは作業中です。

8. ファイル エラー

   データベースからワークエリアを更新、ワークエリアからデータベースを更新、ファイルを再リンク（UNIX のみ）、ファイルを削除、などの措置がとられます。

   注記：オブジェクトをチェックアウトすることが可能でない場合、チェックアウト操作によってエラーが生じること

があります。この場合には、ワークエリア ファイルはそのままにします。

## コンフリクトの解決 - データベースからバッチ モードでワークエリアを更新

バッチ モードでの `Update work area from database` （データベースからのワークエリアの更新）では、データベースの状態からワークエリアを更新することにより、すべてのコンフリクトを解決します。コンフリクトの内容が異なるごとに、コンフリクト解決方法があります（コンフリクトのタイプ を参照）。

以下に、データベースからバッチ モードでワークエリアを更新することによりコンフリクトを解決した場合に、なにが起こるかを示します。

1. 作業中のオブジェクトに対するワークエリアの変更

   データベース オブジェクトのソース属性の内容でワークエリア ファイルが上書きされます（ワークエリアの変更が破棄される）。

2. 作業中のオブジェクトに対するデータベースの変更

   データベース オブジェクトのソース属性の内容でワークエリア ファイルが上書きされます（データベースの変更が受け入れられる）。

3. 静的なオブジェクトに対するワークエリアの変更

   データベース オブジェクトのソース属性の内容でワークエリア ファイルが上書きされます（ワークエリアの変更が破棄される）。

4. 静的なオブジェクトに対するデータベースの変更

   データベース オブジェクトのソース属性の内容でワークエリア ファイルが上書きされます（データベースの変更が受け入れられる）。

5. 複数の場所でのオブジェクトの変更

   データベース オブジェクトのソース属性の内容でワークエリア ファイルが上書きされます。

6. ワークエリアからのファイルの欠落

   欠落しているワークエリア ファイル／ディレクトリがデータベースからワークエリアにコピーされます。

7. 非管理ファイル

   非管理ファイル／ディレクトリがワークエリアから削除されます。

8. ファイル エラー

   データベースからワークエリアを更新、ワークエリアからデータベースを更新、ファイルを再リンク（UNIX のみ）、ファイルを削除、などの措置がとられます。

## コンフリクトの解決 - 手作業でのコンフリクトの選択と解決

リコンサイル処理では、コンフリクト状態にあるファイルを識別しますが、ユーザーが指示しない限り、コンフリクトを解決する措置が取られることはありません。

手作業選択モードの "`Select`" では、コンフリクトを解決する方法を選択できます。このモードを取ると、コンフリクトを解決するための種々の方法を使用できます。検出されたコンフリクトのタイプに応じて、ワークエリアからデータベースを更新する、データベースからワークエリアを更新する、コンフリクト状態にあるファイルをマージする、コンフリクトを無視する、といった方法を取ります。この方法を用いたコンフリクトの解決のさらに詳細については、具体的なタイプのコンフリクトの解決に関するヘルプを参照してください。

# Rational Synergy ヘルプへのリンク

以下のリンクにより、PDF 形式の Rational Synergy ヘルプを参照できます。

- 開発者用 Rational Synergy ヘルプ PDF
- チーム リーダーおよびビルドマネージャ用 Rational Synergy ヘルプ PDF
- Rational Synergy CLI ヘルプ、ウェブモード PDF
- Synergy Classic GUI ヘルプ、（UNIX、Windows）PDF

# 付録：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。

IIBM 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.html. をご覧ください。

# 索引

## 記号



## 数字



## A



## B



## C

## D



## E



## F



## G



## H



## I



## L



## M

## N



## P



## Q



## R



## S



## T

## U



## V



## W



## あ



## い



## え



## お

## か



## き



## く

## け



## こ

## さ



## し

## せ



## そ



## た



## つ



## て

## と



## は



## ひ



## ふ

## へ



## ま