



# プロジェクトの ソース管理の使い方

---

© 2002-2008 Unify Corporation All rights reserved. Sacramento California, USA

No part of this tutorial may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual or otherwise without the prior written consent of Unify Corporation.

Unify Corporation makes no representations or warranties with respect to the contents of this document and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. Further, Unify Corporation reserves the right to revise this document and to make changes from time to time in its content without being obligated to notify any person of such revisions or changes.

The Software described in this document is furnished under a Software License Agreement. The Software may be used or copied only in accordance with the terms of the license agreement. It is against the law to copy the Software on tape, disk, or any other medium for any purpose other than that described in the license agreement.

The Unify Corporation Documentation Group values and appreciates any comments you may have concerning our documents. Please address comments to:

doc@unify.com

1-800-24 UNIFY or 1-800-GO-UNIFY;(916) 928-6400  
FAX (916) 928-6401

UNIFY and DataServer are registered trademarks of Unify Corporation. Unify NXJ is a trademark of Unify Corporation. Java and J2EE are registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. JReport is a trademark of Jinfonet Corporation. IBM, Lotus, Lotus Notes, Cloudscape, and WebSphere are trademarks of International Business Machines Corporation in the United States, other countries, or both. CAS AHL Technology and ecKnowledge are registered trademarks of CAS AHL Technology, Inc. in the U.S. and other countries. All other products or services mentioned herein may be registered trademarks, trademarks, or service marks of their respective manufacturers, companies, or organizations.

Name: Using Source Control for Project

Release: Unify NXJ 12

Last Revision: January 23, 2009 2:58 pm

---

# プロジェクトのソース管理の使い方

Unify NXJ ソース管理機能は、様々なソース管理システムで作業することを目指しています。現在は、CVS のみがサポートされています。別のソース管理システムを使用する場合は、アプリケーションデザイナのインポート とエクスポート コマンドを使用して、プロジェクトファイルをインポートまたはエクスポートすることができます。詳細については、[27 ページの「その他のソース管理システムを使用」](#)を参照してください。

**注：** CVS は Unify NXJ にバンドルされています。Unify NXJ をインストールする場合、CVS “ファイル” プロトコルを使用する CVS を構成します。CVS “ファイル” プロトコルを使用する代わりに、CVS “PServer” プロトコルを使用する場合は、CVS を再インストールする必要があります。CVS は、Unify NXJ に含まれています。CVS をインストールするには、<UNIFY\_HOME>%cvsnt%にある **cvsnt-installer.exe** を実行します。c:%Unify%NXJ%CVS のような <UNIFY\_HOME> の新しいサブディレクトリに CVS をインストールします。

## ソース管理とは？

ソース管理は、アプリケーションへの変更を管理するためのメソッドです。アプリケーションのソースファイルの履歴を記録するためにソース管理を使用します。ソースファイルを変更した後、ソース管理のリポジトリの中に変更が反映されているかを確認します。リポジトリは、ソース管理にあるすべてのファイルとディレクトリの完全なコピーを格納します。通常、リポジトリにあるどんなファイルにも直接アクセスしてはいけません。その代わりに、作業ディレクトリ中のファイルのコピーを自分用に持つために、ソース管理 コマンドを使用することができます。その後、そのコピーしたファイルを使って作業を行います。

---

変更が完了するたびに、リポジトリの中に戻ってそれが反映されているかどうかを確認します。開発者が何を、いつ変更したかという情報、さらにその他の情報を正確に記録するのと同様に、加えられた変更をシステムは格納します。

ソース管理に配置されたアプリケーションは、繰り返し一貫した方法で異なるバージョンのアプリケーションを検索や構築することができます。

ソース管理は複数の開発者が同じプロジェクトで作業することを可能にします。それぞれの開発者が別々のディレクトリで作業してアプリケーションの実行を行うとき、ソース管理システムはその作業をマージします。プロジェクトの公式なバージョンは、ソース管理システムの中で確認されるものになります。プロジェクトのビューはローカルです。プロジェクトの開発チームの他のメンバーがプロジェクトにアクセスする前に、プロジェクトの変更をチェックインする必要があります。

この章は、ユーザがアプリケーション開発環境のソース管理の使い方の基本的な手順に精通していると仮定して説明致します。

## NXJ アプリケーションのソースとは？

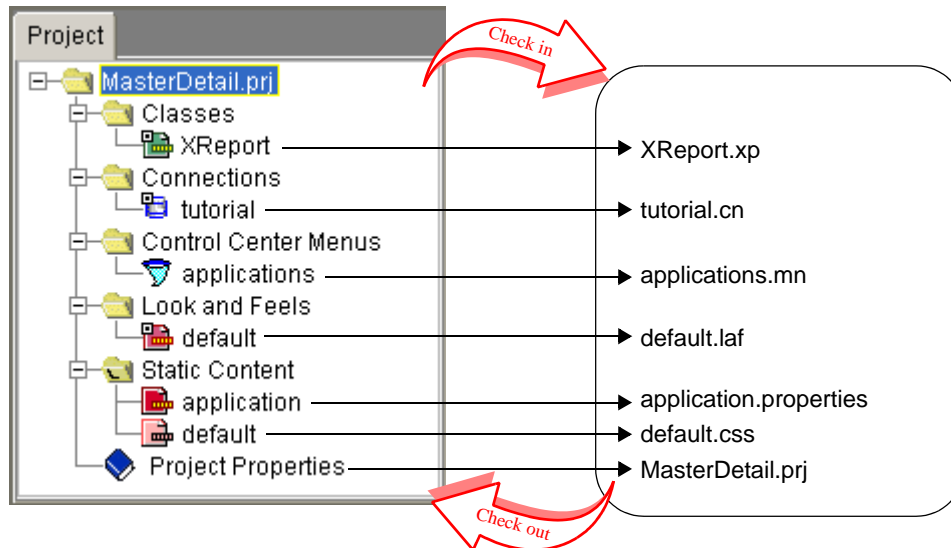
NXJ アプリケーションのソースは、NXJ プロジェクトディレクトリに存在するファイルです。これらのファイルは、アプリケーションデザイナーでプロジェクトのプロジェクトタブに表示されるオブジェクトに対応しています。つまり、フォームクラス、Java クラス、Static Content ファイル、プロパティファイルおよびその他のオブジェクトです。これらのオブジェクトのみが、ソース管理の対象となります。

フォームクラスがソース管理に配置される場合、それはアプリケーションデザイナーによっていくつかの特別な処理を必要とします。フォームクラスは、複数のソースファイル（.jsp ファイル、.fj ファイル、.fe ファイル、.fx ファイル）で表示されます。これらのファイルは同期を取りつづけないため、アプリケーションデザイナーはチェックインを実行する前に、単一の .xp ファイルにこれらのファイルを連結します。同様に、.xp ファイルをチェックアウトするとき、アプリケーションデザイナーは複数のファイルのオリジナルにフォームクラスを戻します。

下図は、プロジェクトのオブジェクトがリポジトリでどのように表示されるかを示しています。

アプリケーションデザイナーでの  
プロジェクトのオブジェクト

リポジトリにおける  
等しいファイル



プロジェクトタブのフォルダが、リポジトリの中に等しいエントリを持っていない点に注意してください。必要に応じて、フォルダはアプリケーションデザイナーによって、リポジトリに自動的に管理されます。

## ソース管理の概要

Unify NXJ のソース管理機能は、アプリケーションデザイナーからチェックアウト、マージ、チェックインのような、一般的なソース管理ユーザ操作を実行するように設計されています。アプリケーションデザイナーは、内部でソース管理システムと相互に作用します。ソース管理システムを直接使用してこれらの操作を実行したい場合、最初にプロジェクトファイルをエクスポートする必要があります。

例えば、新しいレベルを作成するようなソース管理の管理者の操作は、ソース管理システムによって提供されるツールを使用して行われます。詳細については、ソース管理システムのドキュメントを参照してください。

このセクションは、実行する必要がある共通のソース管理作業、および作業実行中にアプリケーションデザイナー上での動作を要約します。

## 共通のソース管理の作業

このセクションは、ソース管理に関連するプロジェクト - レベルの作業を要約します。

- 
- ソース管理からプロジェクトを取得することで、プロジェクトを作成する  
この作業は、プロジェクトチームに参加する開発者のためにあります。

ファイル > **新規プロジェクト** を選択して、ソース管理からプロジェクトを取り出す を選択します。新規プロジェクトオプションウィザードは、ソース管理システムオプション用のプロンプトを出し、次にプロジェクトファイルをチェックアウトします。プロジェクトの作業用コピーは、アプリケーションデザイナーで作成されて開かれます。

この作業手順に関する説明については、[8 ページの「プロジェクトを取得して作成」](#)を参照してください。

- プロジェクトをソース管理に追加する

この作業は、プロジェクトチームが利用可能とされる必要のあるプロジェクトを開発した開発者のためにあります。

ファイル > **ソース管理システム > プロジェクトをソース管理に追加** を選択します。アプリケーションデザイナーは、ソース管理システムオプション用とチェックインコメントのプロンプトを出し、次にソース管理システムで新しいリポジトリにプロジェクトを配置します。プロジェクトの作業用コピーは、アプリケーションデザイナーに開かれたままです。

この作業手順に関する説明については、[11 ページの「ソース管理にプロジェクトを追加」](#)を参照してください。

- リポジトリの変更でユーザの変更をマージする

プロジェクトの作業用コピー上で処理するとともに、ファイルの確認をした後、リポジトリバージョンに作成された変更をマージします。マージが終了された後、プロジェクトの作業用コピーは、リポジトリの変更と組み合わせられる変更を含めます。テスト後、必要に応じて、チェックインを実行します。

ファイル > **ソース管理システム > 自動マージ** を選択します。アプリケーションデザイナーが変更をマージできない場合、変更の衝突を解決するために相違ダイアログが表示されます。また、手動でマージ操作を実行することもできます。それにより、各変更を操作し衝突を解決することを可能にします。

この作業手順に関する説明については、[22 ページの「自動的に変更をマージする」](#)と [14 ページの「リポジトリで変更のマージ作業」](#)を参照してください。

- プロジェクトに変更をチェックインする

この作業は、プロジェクトの作業用コピーに変更を完了した開発者が、プロジェクトチームのメンバーがこれらを利用できるようにするためのものです。

アプリケーションデザイナーに変更されたオブジェクトがある場合、変更されたオブジェクトを探して選択するために、**編集 > 更新済みを選択** を選択します。次に、**ファイル > ソース管理システム > チェックイン** を選択します。チェックインウィザードは、情報をチェックインするためのプロンプトを発行します。

この作業手順に関する説明については、[17 ページの「ソース管理のプロジェクトに更新をチェックインする」](#)を参照してください。

- プロジェクトの最新バージョンをチェックアウトする

この作業は、リポジトリにあるプロジェクトの作業コピーを更新するためのものです。

最新をチェックアウト コマンドは、プロジェクトの各オブジェクトの最新バージョンをチェックアウトします。ブラウザパネルのプロジェクトタブで、プロジェクトフォルダを右クリックして、最新をチェックアウト を選択します。

この作業手順に関する説明については、[20 ページの「ソース管理からプロジェクトの最新バージョンをチェックアウトする」](#)を参照してください。

また、以下の項目についても本章で説明致します。

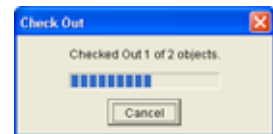
- 変更の再確認
- 非アクティブレベルにマージ
- ログファイルに基づいてチェックイン
- 非アクティブレベルにチェックインを適用

## アプリケーションデザイナーのソース管理に関連する動作

このセクションは、アプリケーションデザイナーにおける、ソース管理操作実行時の表示方法を説明します。

- 処理状況ダイアログ

ソース管理操作が進行中のときは常に、アプリケーションデザイナーは、**取消** ボタンのある処理状況ダイアログを表示します。



保存処理が終了する前であれば、**取消** ボタンをクリックすることで、保存をキャンセルすることができます。取り消しをすることについての詳細は、[25 ページの「操作の取り消し」](#)を参照してください。

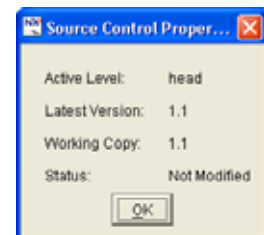
- タイトルバーでのアクティブレベルの識別

カレントプロジェクトがソース管理を使用している場合、アプリケーションデザイナー ウィンドウのタイトルバーは、アクティブレベル名を表示します。デフォルトでは、すべてのチェックインとチェックアウトの操作に対する状態が、タイトルバーに表示されます。

- ソース管理情報のポップアップ

単一のオブジェクト（フォルダ以外）上で右クリックして、プロパティ を選択してオブジェクトについての以下の情報を表示します。

**有効レベル**：アクティブレベルの名称。



**最新バージョン**：リポジトリ内のオブジェクトのカレントバージョン番号。番号は、非公開のオブジェクトや新規作成のオブジェクトでは“なし”と表示されます。

**作業用コピー**：オブジェクトの作業用コピーのカレントバージョン番号。これは、最新のチェックアウト、あるいは最新のマージのバージョン番号のより高いほうのどちらかです。

**状況**：オブジェクトの状態。状態の値については、以下の表を参照してください。

状況	意味
New	これは、新しい作業用オブジェクトであり、チェックインされていません。
Modified	作業用オブジェクトは変更されていますが、チェックインされていません。
Not Modeified	作業用オブジェクトが最後にマージ、チェックイン、チェックアウトされてから変更されていない状態。つまり、作業用オブジェクトは、リポジトリのオブジェクトの最新バージョンと同一です。
Private	作業用オブジェクトは、非公開です。非公開オブジェクトは、“private_folder”というフォルダにあり、ソース管理操作の間は無視されます。
Changed in Source Control	作業用オブジェクトは変更されていませんが、リポジトリバージョンは最後のマージ、チェックイン、チェックアウトから変更されています。
Deleted	作業用オブジェクトは削除されますが、削除はチェックインされません。
Removed	作業用オブジェクトは完全に削除されますが、削除はチェックインされません。

- ブラウザパネルのプロジェクトタブのアイコン

アイコンは、リポジトリバージョンに対して作業用オブジェクトが変更されているかどうかを示します。フォルダのアイコンは、フォルダのどのオブジェクトが変更されたかを示すために変わります。



## プロジェクトを取得して作成

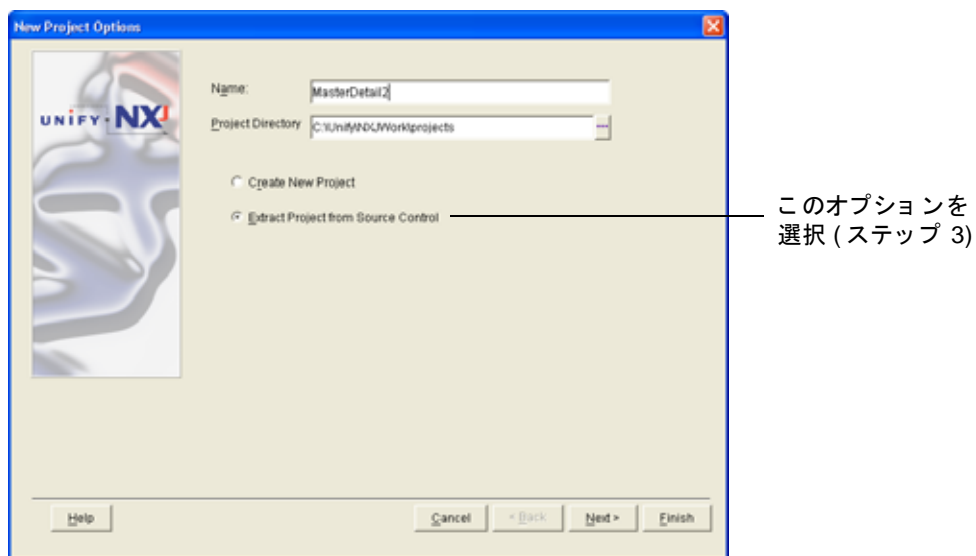
リポジトリに配置されたプロジェクト上で作業を始めるには、アプリケーションデザイナーで利用可能な作業用コピーのプロジェクトを取得する必要があります。プロジェクトを取得するステップは、以下のとおりです。

1. リポジトリ名と場所を取得します。

リポジトリ名、場所、および作業する予定のプロジェクトのレベル名を確認して下さい。

2. **ファイル > 新規プロジェクト** を選択します。

新規プロジェクトオプションパネルが表示されます。



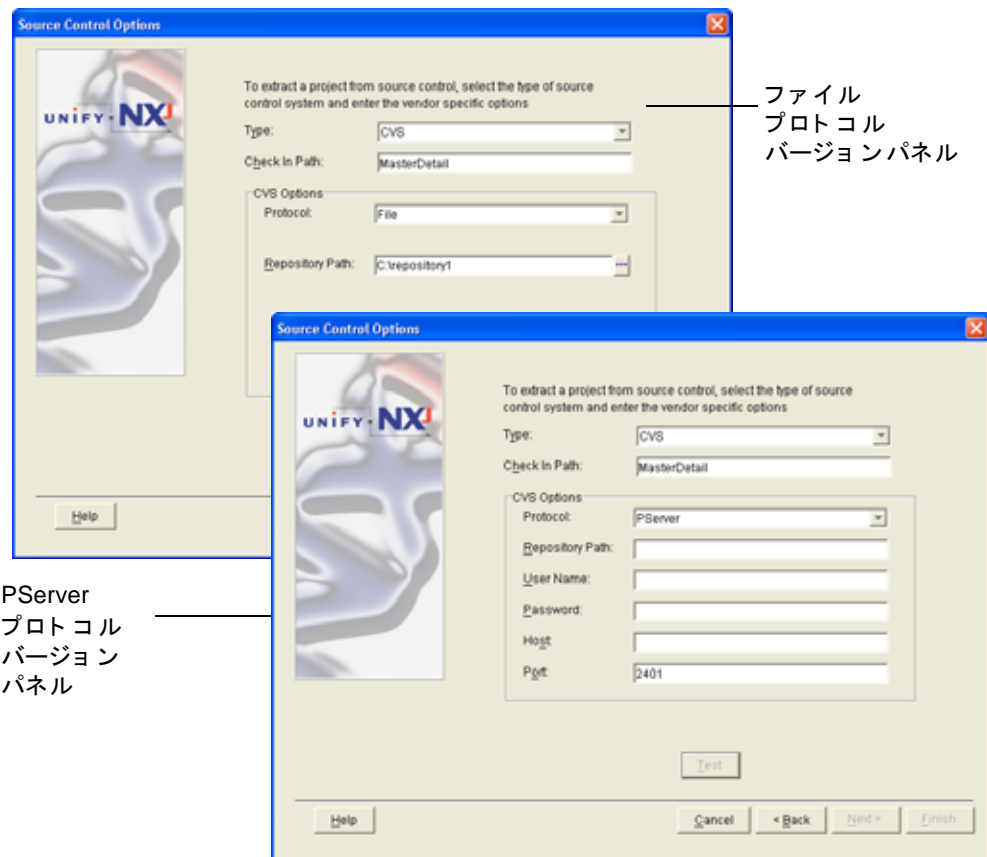
以下の情報を入力します。

**名称:** プロジェクト名。プロジェクト名は、リポジトリ名と一致していなければなりません。

**プロジェクトディレクトリ:** 作業用プロジェクトディレクトリが作成される名前および位置。『開発者ガイド』の第3章「プロジェクト」を参照してください。

3. ソース管理からプロジェクトを取り出す を選択します。
4. 次へ ボタンをクリックします。

ソース管理システムオプションパネルが表示されます。



以下の情報を入力します。

**タイプ:** ソース管理システムのタイプ。デフォルトで、このフィールドは CVS が設定されています。

**チェックインパス:** ファイルが、チェックインで配置されることになっているリポジトリの場所。

パネルの上記以外の項目は、使用しているソース管理システムに特有のフィールドを含みます。CVS の場合、表示されるフィールドは、リポジトリにアクセスするために使用されるプロトコルのタイプに依存します。

ファイルプロトコルの場合、プロトコルフィールドのドロップダウンリストからファイルを選択します。リポジトリパスフィールドで、リポジトリへの絶対パスを指定します。選択 ボタンをクリックして、ファイルシステムからブラウズしてファイルを選択します。他の開発者がリポジトリにアクセスする場合、ファイルがオペレーティングシステムアクセス経由でアクセス可能であることを確認する必要があります。例えば、ドライブは共有する必要があります。

PServer プロトコルの場合、プロトコルフィールドのドロップダウンリストから、PServer を選択します。リポジトリパスフィールドで、サーバがデータベースを検索するためのパスを指定します。これは、必須フィールドです。

ユーザ名フィールドは、CVS コマンドを実行するユーザ名を入力します。デフォルトは、アプリケーションデザイナーが現在実行しているオペレーティングシステムのユーザ名です。このフィールドは、オプションです。ユーザが指定されていないか、指定された値が有効なユーザでない場合、リポジトリへのアクセスが操作で必要とされるときに、ログインダイアログが表示されます。

パスワードフィールドは、CVS コマンドを実行するためのパスワードを入力します。デフォルトは空白です。ユーザ名フィールドの値と同様に、パスワード情報が必要な場合は、ログインダイアログが表示されます。

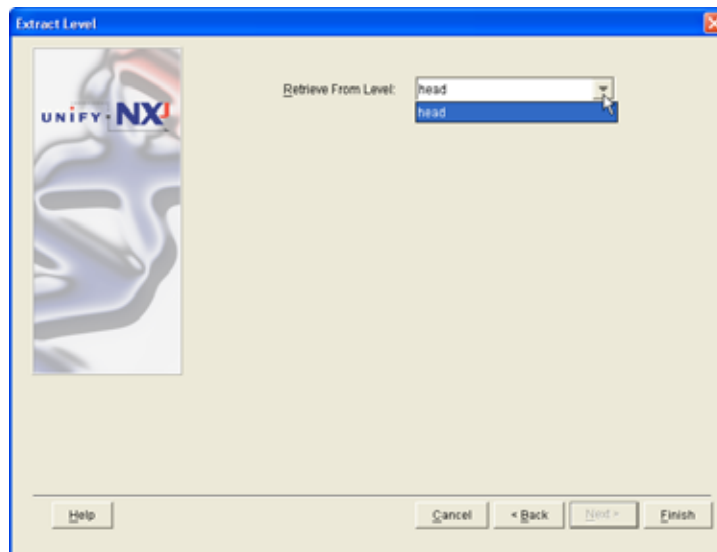
ホストフィールドは、CVS サーバを含むホスト名を入力します。これは、必須フィールドです。作業をしているホスト名がデフォルトになります。

ポートフィールドは、CVS サーバと通信するためのポートを入力します。

5. **次へ** ボタンをクリックします。

ソースレベルパネルが表示されます。

取得するレベルリストボックスでは、取得したいプロジェクトファイルのレベルを選択します。ドロップダウンリストは、ブランチと非ブランチの両方を含めたリポジトリのプロジェクト定義ファイルの指定されたレベルをすべて含んでいます。[24 ページの「非アクティブレベルにチェックインを適用」](#)を参照してください。



6. **完了** ボタンをクリックします。

プロジェクトの作業用コピーが、アプリケーションデザイナーに開きます。

## ソース管理にプロジェクトを追加

この作業を開始する前に、リポジトリを作成するためのファイルシステムの場所を確認します。これは、プロジェクトソースファイルが格納される場所です。他の Unify NXJ プロジェクト定義ファイルを含まない場所を選択します。

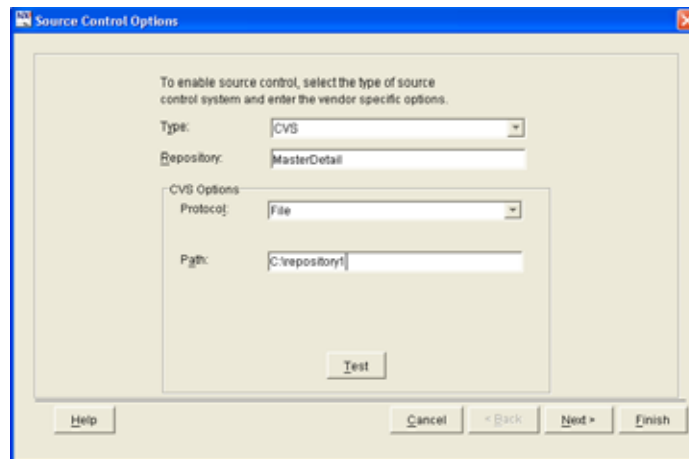
チェックインは、チェックインウィザードを使用して行います。チェックインウィザードを使用している間は、戻る ボタンを使って前のパネルに戻ることができます。テスト、次へ、完了 ボタンはすべての必須フィールドが指定された場合に、利用可能となります。

リポジトリに作業用プロジェクトを設定するステップは、以下のとおりです。

1. プロジェクトを準備します。

チェックインを実行する前に、プロジェクトがチェックインに適した状態かどうか確認して下さい。例えば、すべてのコンポーネントがテストを終えている状態です。

2. **ファイル > ソース管理システム > プロジェクトをソース管理に追加** を選択します。ソース管理システムオプションパネルが表示されます。

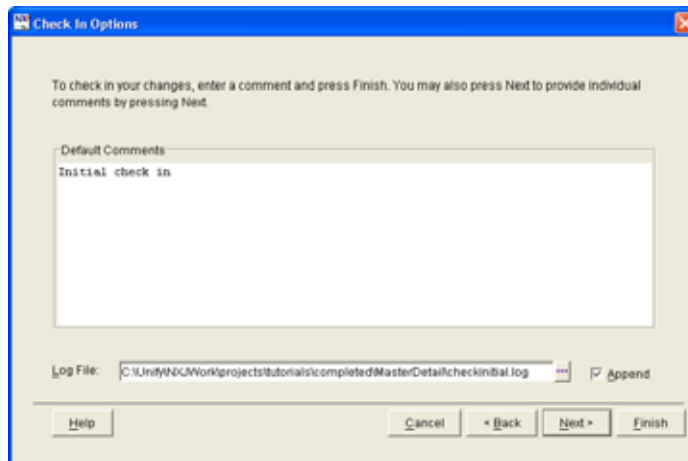


各フィールドに適切な値を入力します。フィールドについての詳細は、[9 ページのステップ 4](#) を参照してください。

3. **テスト** ボタンをクリックして、ソース管理システムへのアクセスをテストします。テスト ボタンは、読み取り専用操作を実行することで、入力された値を確認しようとします。

CVS において、パスフィールドに指定されたディレクトリがない場合、アプリケーションデザイナーは、それを作成するかどうかのプロンプトを出します。

4. **次へ** ボタンをクリックすると、チェックインオプションパネルにアクセスします。チェックインオプションパネルは、チェックインのデフォルトコメントと、チェックイン操作を記録するために使用されるログファイルを指定することができます。

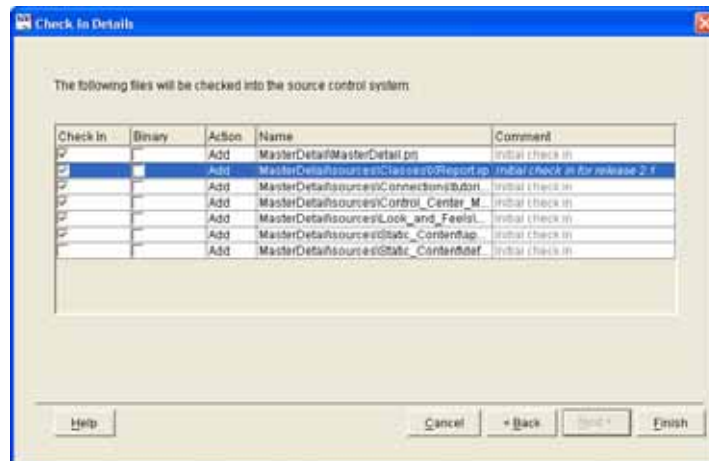


パネルには、以下のフィールドがあります。

**デフォルトコメント**：チェックインする個々のオブジェクトに適用されるデフォルトコメント。各オブジェクトのコメントを次のパネルでオーバーライドすることができます。

5. **次へ** ボタンをクリックすると、チェックインの詳細パネルにアクセスします。

チェックインの詳細パネルは、プロジェクトのパブリックオブジェクト（フォルダはオブジェクトとしてみなされません）の行を含んでいます。



各行は、以下のフィールドを含みます。

**チェックイン**：オブジェクトをチェックインするかどうかの指定。デフォルトでは、チェックボックスは、オブジェクトのチェックインを行なうという設定になっています。オブジェクトのチェックインを行わない場合は、このチェックボックスをクリアにします。

**バイナリ**：オブジェクトをバイナリとしてチェックインすることを指定。オブジェクトが既存の NXJ ファイルタイプ、またはファイルが以前にチェックインされたことがある場合は、バイナリチェックボックスはあらかじめセットされており、変更することはできません。

---

他のオブジェクトで .gif や .bmp 等のイメージファイル、他のバイナリファイルの場合は、バイナリチェックボックスはセットする必要があります。

**アクション**：実行するソース管理アクションのタイプを指定します。プロジェクトをソース管理に追加 コマンドの場合、値は常にリポジトリにオブジェクトを追加することを指示する“追加”です。

**名称**：リポジトリのオブジェクト名と場所。これらは、リポジトリに格納されるファイル名です。

**コメント**：チェックインを説明するコメント。コメントフィールドをクリックして、コメントダイアログにアクセスします。デフォルトコメントを上書き、変更して OK ボタンをクリックします。

コメントがデフォルトコメントと異なる場合、コメントダイアログは黒字の斜体でテキストを表示します。

コメントを指定しない場合、前のパネルからのデフォルトコメントが使用されません。

6. **完了** ボタンをクリックして、チェックインを終了します。

チェックイン操作が終了したとき、アプリケーションデザイナーのソース管理ツールバー ボタンは利用できるようになり、アプリケーションデザイナーのタイトルバーは、有効レベルを表示します。

ユーザは、アプリケーション開発を続けることができます。変更をチェックインする準備が整っている場合、[17 ページの「ソース管理のプロジェクトに更新をチェックインする」](#)を参照してください。

## リポジトリで変更のマージ作業

別の開発者がファイルを変更して、リポジトリにそれをチェックインした場合、その他の作業用コピーは、リポジトリと同期が取れないため、マージ操作が必要とされます。マージ操作を行わない場合、ユーザは新しい変更をチェックアウトすることができますが、現在使用している作業用コピーに対する変更を失うこととなります。

マージには、手動と自動の 2 種類があります。手動マージは、相違ダイアログを使って各変更をマージします。自動マージは、アプリケーションデザイナーによって実行され、リポジトリの変更がプロジェクトの作業用コピーに適用されます。いくつかの変更は、自動マージすることができません。例えば、コードの同じ行がリポジトリと作業用コピーで変更される場合です。これらの変更については、手動マージを使う必要があります。

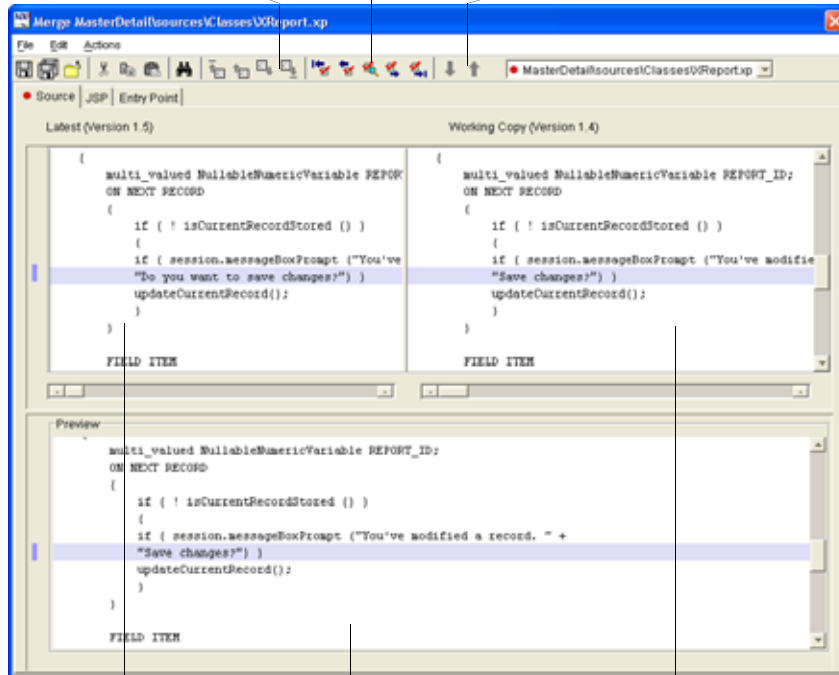
### 相違ダイアログ

相違ダイアログは、変更を処理するためのコマンドで、3 つの領域と独自のツールバーを含みます。

変更の左側、右側、プレビューパネルバージョンの適用

変更のセットをスクロールする

マージする変更を持つオブジェクトをスクロールする



リポジトリバージョン      マージの結果表示領域      作業用バージョン

ダイアログが最初に表示される時、最初の変更部分にカーソルが合わされます。リポジトリバージョンは左に表示され、作業用バージョンは右に表示されます。プレビューパネルは、オリジナルと作業用コピーを選択してマージしたファイルを表示します。変更は、以下のようにハイライトされます。

- ピンクのハイライトは、削除を示します。
- 緑のハイライトは、挿入を示します。
- 青のハイライトは、変更を示します。

フォームの変更は3つのパネルで提供され、1つがNXJプログラミング言語ソースを示し、もう1つはJSPソースを表示し、そしてもう1つはエントリフォームの情報を表示されている事に注意して下さい。JSPソースのタブは、タグに直接埋め込まれる拡張属性（例えば、findable）を持ちます。

バイナリオブジェクトでは、左上のパネルは“Old Binary Data”を表示し、右上のパネルは“New Binary Data”を表示します。これらが含まれる行は、ファイル間の唯一の違いとしてハイライトされます。






別の方法で相違ダイアログを使用することができます。一般に、以下の作業を実行します。

- 変更を解決する
- 別の変更に移動する
- 解決された変更を保存する




## 変更を解決する




適用コマンドを使って、オブジェクトの左側のバージョン、あるいは右側のバージョンのどちらかを適用します。また、手動でプレビューパネルの内容を変更することもでき、プレビューパネルバージョンで適用することができます。

オブジェクトがバイナリオブジェクトの場合、その後プレビューパネルは読み取り専用になります。

ボタン	メニューアクセス	説明
	アクション > 左側のすべてを適用	プレビューパネルに、このタブが付けられたパネル上に残っているすべての変更の左側バージョンをコピーします。
	アクション > 左側を適用	プレビューパネルに、左側バージョンのカレントの変更をコピーします。
	アクション > 右側のすべてを適用	プレビューパネルに、現在のタブ上に残っているすべての変更の右側バージョンをコピーします。
	アクション > 右側を適用	プレビューパネルに、右側バージョンのカレントの変更をコピーします。
	アクション > プレビューを適用	プレビューパネルにある現在の変更のバージョンを適用します。

変更を適用した後、相違ダイアログは“次の変更”ダイアログを表示し、または、“前の変更”ボタンなどを使い、別の変更箇所に移動することができます。

ツールバー ボタン	メニューコマンド	目的
	アクション > 最初の変更	オリジナル、作業用コピー、プレビューの各パネルの中で最初に見つかった変更箇所に移動します。
	アクション > 最後の変更	オリジナル、作業用コピー、プレビューの各パネルの中で最後に見つかった変更箇所に移動します。
	アクション > 次の変更	オリジナル、作業用コピー、プレビューの各パネルの中で次に見つかった変更箇所に移動します。

ツールバー ボタン	メニューコマンド	目的
	アクション > 次のオブジェクト	3つのすべてのテキストエリアで、未解決の次のオブジェクトに移動します。
	アクション > 前の変更	オリジナル、作業用コピー、プレビューの各パネルの中で1つ前に見つかった変更箇所移动到移動します。
	アクション > 前のオブジェクト	3つのすべてのテキストエリアで、未解決の前のオブジェクトに移動します。

一度、変更を解決すれば、ハイライトは3つのすべてのパネルからなくなり、ダイアログは次の変更順番に移動します。これらのパネルに、変更箇所がなくなった場合は、ダイアログは現在の位置にとどまります。現在のパネルに変更がある場合のみ、コマンドを利用することができます。

## 解決された変更を保存する

すべての変更を解決した場合、変更を保存しなければなりません。変更をチェックインする必要があります。終了するために、ダイアログを閉じます。

変更のすべてが解決されていない場合、変更をチェックインする前にマージ処理の部分をやり直す必要があります。

## ソース管理のプロジェクトに更新をチェックインする

プロジェクトに更新をチェックインするには、チェックインウィザードを使用します。

**注：** オブジェクトへの変更は、アプリケーションデザイナーによって必ず行われます。変更したオブジェクトをアプリケーションデザイナーが表示するとき、ソースコントロールで管理されているオブジェクトを理解しておく必要があります。詳細については、[27ページの「オブジェクトへの暗黙的な変更」](#)を参照してください。

プロジェクトにすべての更新をチェックインするステップは、以下のとおりです。

1. チェックインする必要があるオブジェクトを選択します。

チェックインするファイルを選択するいくつかの方法があります。ブラウザパネルのプロジェクトタブで、自分でオブジェクトを選択することができます。フォルダを選択する場合、変更されたフォルダの中のすべてのオブジェクトが必ず選択されます。プロジェクト名を選択する場合、プロジェクトのフォルダのすべてが必ず選択されます。

**編集 > 更新済みを選択** コマンドを選択して、アプリケーションデザイナーに変更済みのオブジェクトを見つけさせ、選択させることもできます。

完全に削除されたオブジェクトは、他の変更済みのオブジェクトと同様にチェックインを行なう必要があります。これらのオブジェクトのエントリは、色が薄く表示されます。

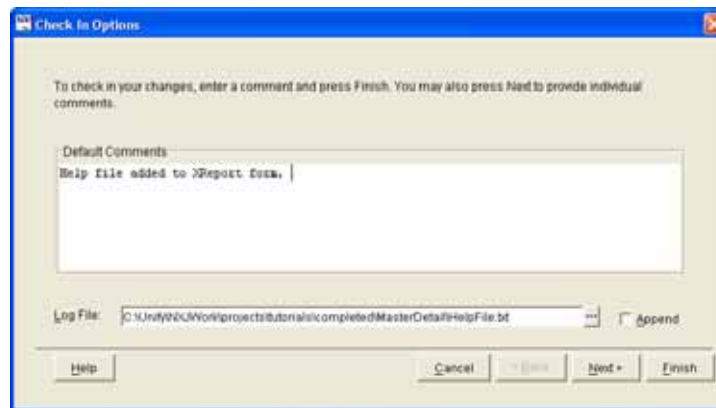
名前を変更したオブジェクトをチェックインしている場合、アプリケーションデザイナーはオブジェクトを新しい名前で確認して、次にそのソース管理から古い名前のオブジェクトを削除します。チェックインのコメントで、そのことを記述しておけば確認が容易になります。

2. **ファイル > ソース管理システム > チェックイン** を選択するか、または**チェックイン** ボタンをクリックします。

選択されたオブジェクトのいずれも保存されていない場合、変更を保存するかどうかのプロンプトが表示されます。

チェックインオプションパネルが表示されます。

3. チェックインオプションパネルで、デフォルトコメントを入力します。



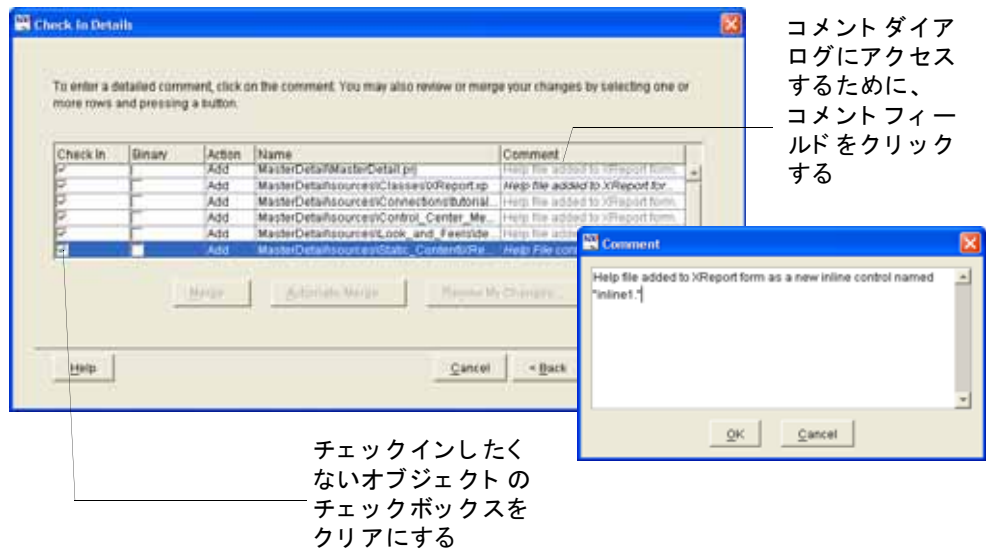
**ログファイル（オプション）**：このチェックイン操作のログを出力する新規、または既存のテキストファイルの名前です。ログファイルは、リポジトリの異なるレベルに対して、チェックイン操作を繰り返しアクセスすることができます。ログファイルの使い方についての詳細は、[23 ページの「ログファイルに基づく更新をチェックインする」](#)を参照してください。

ログのファイル名を入力するか、**選択** ボタンをクリックして、ファイル選択ダイアログにアクセスします。

**追加**：ログファイルのエントリをログファイルに追加して書き加えるかどうかを指定します。追加チェックボックスがクリアの場合、チェックイン操作中、ログファイルの内容は毎回クリアされて書き替えられます。

4. **次へ** ボタンをクリックすると、チェックイン詳細パネルにアクセスします。

このパネルでチェックインするオブジェクトを確認します。チェックイン詳細パネルは、ステップ 2 で選択したオブジェクトの各行を表示しています。



各行は、以下のフィールドを含んでいます。

**チェックイン**：オブジェクトをチェックインするかどうかの指定。デフォルトでは、チェックボックスはオブジェクトのチェックインを行なうという設定になっています。オブジェクトのチェックインを行わない場合は、このチェックボックスをクリアにします。

**バイナリ**：オブジェクトをバイナリとしてチェックインすることを指定。オブジェクトが既存の NXJ ファイルタイプ、またはファイルが以前にチェックインされたことがある場合は、バイナリチェックボックスはあらかじめセットされており、変更することはできません。

他のオブジェクトで .gif や .bmp 等のイメージファイル、他のバイナリファイルの場合は、バイナリチェックボックスはセットする必要があります。

**アクション**：以下のオプションを表示しています。

追加	リポジトリにオブジェクトを追加する。
更新	作業用コピーでリポジトリのバージョンを置き換える。
削除	リポジトリのバージョンを削除する。
失敗	チェックイン中にエラーがあった。

**名称**：リポジトリのオブジェクト名と場所。これらは、リポジトリに格納されるファイル名です。

**コメント**：チェックインを説明するコメント。コメントフィールドをクリックして、コメントダイアログにアクセスします。デフォルトコメントを上書き、変更して OK ボタンをクリックします。

コメントがデフォルトコメントと異なる場合、コメントダイアログは黒字の斜体でテキストを表示します。コメントを指定しない場合、前のパネルからのデフォルトコメントが使用されます。

---

パネルからマージ、自動マージ、変更を閲覧を起動することができます。マージ、自動マージ、変更を閲覧 ボタンはアクションが更新であるすべての選択された行に対してコマンドを実行します。

マージ操作についての詳細は、[14 ページの「リポジトリで変更のマージ作業」](#)を参照してください。

自動マージ操作についての詳細は、[22 ページの「自動的に変更をマージする」](#)を参照してください。

変更を閲覧操作についての詳細は、[21 ページの「チェックインする前に変更を確認する」](#)を参照してください。

#### 5. 完了 ボタンをクリックします。

削除されたオブジェクトをチェックインした場合、アプリケーションデザイナーは、プロジェクトタブのパネルから、オブジェクトを削除します。オブジェクトが完全に削除された場合、チェックインコマンドは、ファイルシステムからオブジェクトも削除します。

エラーがチェックインで発生した場合、チェックイン詳細パネルは、リフレッシュされて、以下のようにどのオブジェクトが影響されたかを表示します。

- オブジェクトのチェックインが成功した場合、その後チェックインの列のチェックボックスはクリアにされ、アクションの列がチェックインに設定されます。
- オブジェクトのチェックインが失敗した場合、その後、チェックインの列のチェックボックスはクリアにされ、アクションの列は失敗に設定されます。

単一レベルへのチェックインの場合、チェックイン操作は終了し、プロジェクトの作業用コピーが、アプリケーションデザイナーに開かれたままになります。

チェックインが複数のレベルにある場合、チェックインの適用ダイアログが表示されます。詳細については、[24 ページの「非アクティブレベルにチェックインを適用」](#)を参照してください。

## ソース管理からプロジェクトの最新バージョンをチェックアウトする

プロジェクトのオブジェクトの最新バージョンをチェックアウトするには、プロジェクトフォルダをクリックして、最新をチェックアウト コマンドを選択するか、**最新をチェックアウト** ボタンをクリックします。



最新を  
チェックアウト

各オブジェクトの作業用コピーは、リフレッシュされます。これらのオブジェクトは、プロジェクトのリポジトリバージョンに関して、次のような同期がとられています。

- オブジェクトがリポジトリから削除された場合、作業用コピーが変更されていない限り、アプリケーションデザイナーは、プロジェクトの作業用コピーからオブジェクトを削除します。作業用コピーが変更されている場合、アプリケーションデザイナーはエラーメッセージを表示します。

- オブジェクトがリポジトリに追加された場合、名前が衝突しない限り、オブジェクトは、プロジェクトの作業用コピーに自動的に追加されます。

オブジェクトがリポジトリに追加されて、プロジェクトの作業用コピーで名前の衝突がある場合（同じ名前で新しいオブジェクトを作成し、それをチェックインしていないような場合）、その後、アプリケーションデザイナーは、オブジェクトを変更せずにエラーメッセージを表示します。

オブジェクトを選択してマージを行うことにより、リポジトリのバージョンでこの新しいオブジェクトをマージすることができます。あるいは、プロジェクトの外（または `private_folder` フォルダ）に新しいオブジェクトを移動させて、再び、“最新をチェックアウト” コマンドを発行します。

- プロジェクトディレクトリに実際に存在するファイルであるのに、アプリケーションデザイナーのブラウザパネルのプロジェクトタブにそのファイルがない場合、アプリケーションデザイナーは、既存のファイル名を変更し、エラーメッセージを表示してプロジェクトの作業用コピーにファイルを追加します。ファイルは、自動的に名前を変更されます。

## チェックインする前に変更を確認する

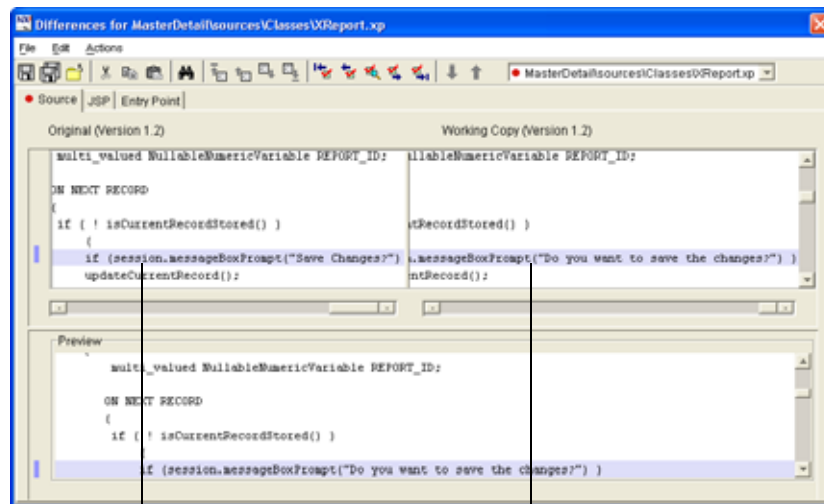
プロジェクトの作業用コピーに存在する変更を確認したい場合、**変更を閲覧** コマンドを使用します。



変更を閲覧

プロジェクトフォルダを選択したまま、**ファイル > ソース管理システム > 変更を閲覧** を選択するか、**変更を閲覧** ボタンをクリックします。相違ダイアログが表示されます。相違ダイアログは、開発者が変更したオブジェクトのセットと各オブジェクトの変更のセットを操作して、スクロールすることができるという点で、相違ダイアログに似ています。相違ダイアログについての詳細は、[14 ページの「リポジトリで変更のマージ作業」](#)を参照してください。

相違ダイアログと同様に、必要に応じてプレビューパネルを使用して、オブジェクトの作業バージョンを変更します。例えば、確認中に意図せずに行った変更を見つけた場合、変更のリポジトリバージョンを受け付けて変更を取り消すことができます。この場合、オブジェクトは前の状態を示しています。



リポジトリのバージョン

作業用バージョン

プレビューパネルの変更を終えるとき、**ファイル > 保存** を選択して変更を保存する必要があります。これにより、オブジェクトの作業用バージョンを更新します。

ウィンドウを閉じて、変更を閲覧ダイアログを終了します。

## 自動的に変更をマージする

自動マージ操作は、リポジトリの最新のバージョンとオブジェクトの作業用バージョンを比較して、どのような衝突があってもマージを行なおうとします。衝突がアプリケーションデザイナーでマージできなくて、そのため手動でマージする必要がある場合は、相違ダイアログが表示されます。[14 ページの「リポジトリで変更のマージ作業」](#)を参照してください。

自動マージ操作は、選択されたオブジェクトの自動マージ コマンドを発行するか、**ファイル > ソース管理システム > プロジェクトを自動的にマージ** を選択することにより、チェックインウィザードから起動することができます。

自動マージは、リポジトリにあるものとプロジェクトの作業用コピーとの同期をとります。

また、マージ処理はチェックインコマンドの一部として、非アクティブレベルに変更を適用しようとする場合に実行されます。

---

## ログファイルに基づく更新をチェックインする

“ログからマージ” コマンドを使ってログファイルに取り込まれた前のチェックインを適用できます。“ログからマージ” コマンドは、取り込まれたチェックイン操作からアクティブレベルに変更を自動的にマージしようとします。変更が自動的にマージされなかった場合、アプリケーションデザイナーは相違ダイアログを表示します。

ログファイルに基づく更新をチェックインするステップは、以下のとおりです。

1. ログファイルの位置を確認します。
2. **ファイル > ソース管理システム > ログからマージ** を選択します。  
ログからマージダイアログが表示されます。
3. ログファイルへの絶対パスを入力するか、選択します。  
デフォルトは、前のチェックインからのログファイルです。
4. **OK** ボタンをクリックします。  
アプリケーションデザイナーは、ダイアログを閉じて自動マージを開始します。
5. 変更をチェックインします。  
[17 ページの「ソース管理のプロジェクトに更新をチェックインする」](#)を参照してください。

このコマンドでは、オブジェクトの作業用リビジョン番号は変更しません。

## 非アクティブレベルでマージする

レベルへマージ コマンドは、非アクティブレベルへの作業用プロジェクトのオブジェクトにすべての変更を自動的にマージしようとし、次に有効レベルをリセットします。自動的にマージできない変更がある場合、アプリケーションデザイナーは 未解決の変更で相違ダイアログを表示します。

他のレベルに適用したいバグ修正がある場合、一般的にこの作業を実行します。

非アクティブレベルに変更をマージするステップは、以下のとおりです。

1. プロジェクトの作業用コピーが、チェックインしたい変更を持っていることを確認します。  
変更を閲覧 コマンドを使用してもかまいません。[21 ページの「チェックインする前に変更を確認する」](#)を参照してください。
2. **ファイル > ソース管理システム > レベルへマージ** を選択します。  
レベルへマージダイアログが表示されます。  
リポジトリは、ダイアログが表示されるために少なくとも 1 つの追加のレベル（有効レベルに加えて）を持っていなければなりません。

3. レベルヘマージフィールドで、チェックインしたいレベルを選択します。

4. **OK** ボタンをクリックします。

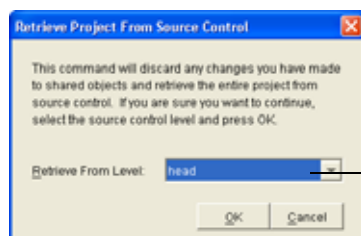
アプリケーションデザイナーは、指定されたレベルに変更をマージしようとします。有効レベルは、変更が自動的にマージすることができたかどうかに関係なく設定されます。

マージを中止するには、**取消** ボタンをクリックします。

有効レベルがリセットされます。

## ソース管理システムから非アクティブバージョンを チェックアウトする

ソース管理システムからプロジェクトを取得 コマンドは、ソース管理からカレントプロジェクトの特定のレベルを取り出します。(最新をチェックアウト コマンドが、最も高いレベルを取り出します。) ソース管理システムからの取得 コマンドは、プロジェクトを取り出すレベルを指定することができる ソース管理システムからプロジェクトを取得ダイアログを表示することにより開始します。



プロジェクト ( ブランチと非  
ブランチレベルの両方 ) のす  
べてのレベルの名前

**OK** ボタンをクリックして、取得を開始します。プロジェクトの作業用コピーの更新方法についての詳細は、[20 ページの「ソース管理からプロジェクトの最新バージョンをチェックアウトする」](#)を参照してください。

取り出されたレベルは、プロジェクトの新しい有効レベルになります。

## 非アクティブレベルにチェックインを適用

複数のレベルを持つリポジトリをチェックインする場合、チェックインウィザードを使用して、これらのレベルにチェックインを適用することができます。複数のレベルが検出された場合、チェックインウィザードの最後のパネルは、その他のレベルパネルにチェックインします。

パネルは、有効レベルではない各レベル情報の行を表示します。ステータスの列には、以下の値があります。

- **適用済み**: チェックインが既にこのレベルに適用されていることを示します。

- **未適用**：このレベルへの変更を適用しようとしていないことを示します。
- **中止**：このレベルへの変更を適用しようとしたが、衝突が解決されていなかったことを示します。
- **失敗**：チェックインを適用するときにエラーがありました。

HEAD の 1 行と各ブランチレベルの 1 行があります。状態の列は、有効レベルに適用するため最初に設定され、他のすべてのレベルは適用されません。

別のレベルにチェックインを適用するには、行を選択して **チェックインを適用** ボタンをクリックします。選択された行の状態が、未適用 または 中止の場合、**チェックインを適用** ボタンは利用可能です。アプリケーションデザイナーは、選択されたレベルに変更をマージするために自動マージを行い、その後ファイルにチェックインします。どのオブジェクトも選択されたレベルに存在しない場合、アプリケーションデザイナーは、それを追加します。チェックインがオブジェクトのどれかを削除した場合、アプリケーションデザイナーは、選択されたレベルからこれらを削除します。

自動マージが衝突を検出した場合、アプリケーションデザイナーは相違ダイアログを表示します。ユーザは、衝突を解決し、変更を保存して、相違ダイアログを終了することができます。アプリケーションデザイナーは、チェックインを続けます。

アプリケーションデザイナーがチェックインを行おうとするとき、アプリケーションデザイナーがマージを実行するのは、他の開発者が同じオブジェクトでチェックインしているためです。この場合、アプリケーションデザイナーは変更を再度マージする必要があることを示したポップアップを表示します。

変更を適用し終わったら、**Done** ボタンをクリックします。**Done** ボタンで、ダイアログを閉じます。

## 操作の取り消し

アプリケーションデザイナーは、コマンドが取り消される前に実行された操作を反映しません。例えば、100 個のオブジェクトをチェックインしようとして、5 個のオブジェクトをチェックインした後にコマンドを取り消そうとした場合、残りのオブジェクトは、修正済として表示されます。

操作を繰り返す必要があります。

## ソース管理の関連付けを削除する

プロジェクトからソース管理に関連する情報を削除するには、**ファイル > ソース管理システム > ソース管理オプションのクリア** を選択します。ソース管理からプロジェクトを取得して、ソース管理の環境以外で作業をしたい場合に有用です。

---

## レベル

リポジトリのレベルは、異なるソース管理システムでは、違うものを意味します。CVSでレベルはタグ（順番にファイルの数字のレビジョンを割り当てられる記号の識別子である）です。CVS ツールを使用してタグを作成します。アプリケーションデザイナーは、これらを作成することはできません。

新規プロジェクト コマンドまたはソース管理システムからの取得 コマンドによってプロジェクトを抽出する場合、レベルのドロップダウンは、プロジェクト定義ファイル (.prj) 用のすべての CVS タグで作成されます。

ソース管理操作が、リポジトリの複数のレベルに既存するソースファイルに対して行われるとき、アプリケーションデザイナーはこれをユーザに通知し、他のレベルを扱うためのオプションを表示します。例えば、チェックインウィザードは、プロジェクト定義ファイルのすべてのブランチレベルのタグを示し、これらのレベルでチェックインすることを許可する追加のチェックインの適用ダイアログを表示します。

## レビジョン履歴のコメント

アプリケーションデザイナーのオブジェクトの作業用コピーで作業する場合、プロパティダイアログを使って、簡単にファイルの状態を決めることができます。アプリケーションデザイナー以外のファイルで作業する場合は、ファイルへのレビジョン履歴の記録方法を知っている必要があります。

アプリケーションデザイナーは、ファイルの終わりにファイルのレビジョン履歴とともに1行以上のコメント行を挿入します。

フォームの場合、アプリケーションデザイナーはスクリプトファイル (.fj) の終わりに、コメントを置きます。コメントは、以下のフォーマットです。

```
/*-----*
* BEGIN MODIFICATION HISTORY
*
* Revision #      Date      Time      Changes By
* -----
* $Log: $
* END MODIFICATION HISTORY
*-----*/
```

プロパティファイルフォーマット（コネクションの定義とアプリケーションメニュー）に格納されるプロパティファイルとその他のオブジェクトの場合、コメントは以下のフォーマットになります。

```
#####
## BEGIN MODIFICATION HISTORY
##
## Revision #      Date      Time      Changes By
## -----
## $Log: NXJ Application Designer.properties,v $
## END MODIFICATION HISTORY
#####
```

静的 HTML、JSP ページ、プロジェクト定義ファイルの場合、コメントは以下のようになります。

```
<!-------
* BEGIN MODIFICATION HISTORY
*
* Revision #      Date      Time      Changes By
* =====
* $Log: NXJ Application Designer.properties,v $
* END MODIFICATION HISTORY
*----->
```

## オブジェクトへの暗黙的な変更

アプリケーション開発やテスト中に実行する作業の一部は、オブジェクトに暗黙的な変更を引き起こします。変更はアプリケーションデザイナーで認識されるので、ユーザはその方法を知っている必要があります。以下のように、オブジェクトへの暗黙的な変更を起こすことができるいくつかの方法があります。

- エントリポイント追加コマンドを実行して“このエントリポイントをメニューに追加”チェックボックスをチェックして **OK** ボタンを押下するとき、アプリケーションデザイナーは暗黙のうちにメニューを更新します。
- プロジェクト > プロパティ コマンドを実行して、**OK** または **適用** ボタンを押下するとき、アプリケーションデザイナーは暗黙のうちにメニューを更新します。
- ローカライズされたフォームを編集するとき、タグの名前や値を変更するためのダイアログが表示されます。このダイアログは、Static Content フォルダの下のプロパティファイルの 1 つを暗黙のうちに変更します。

## その他のソース管理システムを使用

Unify NXJ ソース管理機能は、現在 CVS のみをサポートします。アプリケーションデザイナーは、他のソースコード管理システムに適切なフォーマットへプロジェクトをエクスポートし、アプリケーションデザイナーに以前エクスポートされたプロジェクトを戻すためのインポートを可能にします。プロジェクトをインポートやエクスポートしたり、

---

ソースコード管理システムを使うことで、NXJ アプリケーション開発者チームは、アプリケーションのプロジェクトを共有することができます。

プロジェクト全体よりも、プロジェクトの個別のオブジェクトのほうが開発者同士でエクスポート、インポート、共有するのに適しています。例えば、標準の look & feel 定義は、ソースコードコントロールシステムで保存され、すべてのプロジェクトによって使用されます。

このドキュメントは、以下の作業について記述します。

- プロジェクトをソースコード管理システムに配置する
- 個々のオブジェクトをソースコード管理システムに配置する
- ソースコード管理システムのオブジェクトを変更する
- プロジェクトのオブジェクトを削除する
- プロジェクトを抽出する

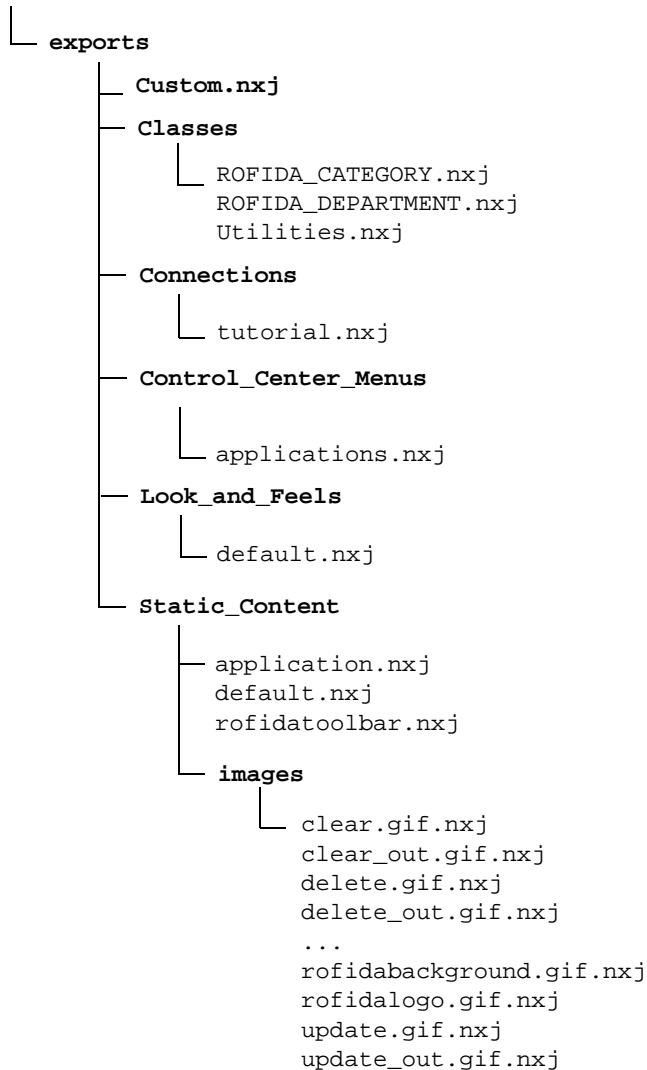
## プロジェクトをソースコード管理システムに設定する

ソースコード管理システムに NXJ プロジェクトをサブミットする前に、プロジェクトをエクスポートしておく必要があります。プロジェクトをエクスポートすると、プロジェクトの中に各オブジェクトの .nxj ファイルが作成されます。このファイルは、アプリケーションデザイナーでのオブジェクト名に、.nxj 拡張子がついたファイル名となります。 .nxj ファイルは、プロジェクトディレクトリ下の “exports” ディレクトリ内に各カテゴリ毎に作成されますが、.nxj ファイルはブラウザパネルには表示されません。これらは OS レベルで参照することができます。

以下の図は、Unify NXJ チュートリアル7のレッスン7で Custom プロジェクトをエクスポートした場合の NXJ プロジェクトファイルと作成される .nxj ファイルの名前と場所を示しています。フォームには関連する4つのファイルがありますが .nxj ファイルは一つのみであることに注意します。

<UNIFY\_WORK>¥projects¥tutorials¥Custom

プロジェクトでの  
オブジェクトの  
エクスポート  
バージョン



プロジェクトの **.nxj** ファイルをソース管理システム下に設定するときは、プロジェクト定義ファイル (**.prj**) もソースコードコントロール下に設定する必要があります。プロジェクト定義ファイルは、プロジェクト全体のプロパティであるだけでなく、プロジェクトの全オブジェクトのリストを含んでいます。**.prj** ファイルは、エクスポートを行いません。**.prj** ファイルは **.prj** フォーマットでソースコードコントロール下に設定されます。

**注：** **.nxj** ファイルは、インポートされるときにそのファイルが NXJ プロジェクトフォーマットに、どのようにリストアされるべきかを指定する Unify NXJ コマンドを含んでいます。このファイルをアプリケーションデザイナーの外部で使用するとき、NXJ コマンドの変更は行わないで下さい。

ソースコード管理システムに新規プロジェクトを設定するステップは、以下のとおりです。

1. **ファイル > 新規** を選択して、アプリケーションデザイナーで新規プロジェクトを作成します。
2. プロジェクトに必要なすべてのフォーム、コネクション、look & feel 定義、メニュー定義を作成します。

『開発者ガイド』を参照して、これらの作業を実行してください。これらのオブジェクトがソースコード管理システムに配置されるための定義方法に制限はありません。

3. **ファイル > すべて保存** を選択して、プロジェクトのオブジェクトの設定を保存します。
4. ブラウザパネルのプロジェクトタブで、<projectname>.prj エントリを選択します。
5. 右クリックして、オブジェクトのエクスポートを選択します。

[29 ページ](#)に示されるように、プロジェクトの各オブジェクトの .nxj バージョンは、プロジェクトの“Sources” ディレクトリ中の対応したディレクトリに書かれます。

6. ソースコード管理システムを使用して、.nxj ファイルと .prj ファイルをチェックインします。

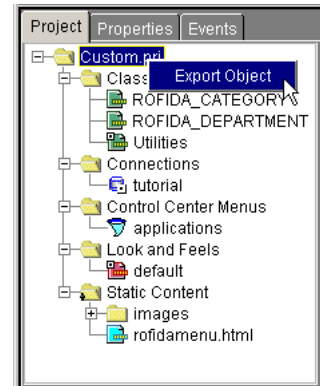
インポートを使ってプロジェクトを同じ構成に戻すためには、ディレクトリ構成を維持する必要があります。

.prj ファイルは、“Sources” ディレクトリと同じレベルでプロジェクトディレクトリにあります。

7. アプリケーションデザイナーで、プロジェクトを閉じます。

**注：** プロジェクトのエクスポートではロックされないで、プロジェクトは更新できます。エクスポートされたプロジェクトは、エクスポートコマンド発行時のプロジェクトファイルのスナップショットです。

ステップ 4 と 5



## 個々のオブジェクトをソースコード管理システムに設定する

プロジェクト全体ではなく、個別のオブジェクトやオブジェクトのフォルダを指定して、エクスポートすることができます。[30 ページ](#)のステップに従って、プロジェクトをソースコード管理システムに設定しますが、[ステップ 4](#)では、<projectname>.prj エントリを選択する代わりにエクスポートしたいフォルダやオブジェクトを選択します。選択したオブジェクトごとに .nxj ファイルが作成されます。

## ソースコード管理システムのオブジェクトを変更する

ソースコード管理システムにある NXJ オブジェクトを変更するステップは、以下のとおりです。

1. ソースコード管理システムを使って、変更したい .nxj ファイルや .prj ファイルをチェックアウトします。

関連するプロジェクトディレクトリ（exports\Classes 等）に、ファイルを正しく配置してください。古い .nxj ファイルは削除され、上書きされます。

2. アプリケーションデザイナーで、ブラウザパネルのプロジェクトタブでオブジェクトを選択して、右クリックしオブジェクトのインポートを選択して、ファイルをインポートします。

選択されたオブジェクトと同じ名前の .nxj ファイルを検索して、一致するディレクトリがスキャンします。検索されたファイルは .nxj フォーマットからプロジェクトフォーマットにコンバートされて、ブラウザパネルのプロジェクトタブのフォルダに配置されます。

.prj ファイルをインポートするには、プロジェクトディレクトリに .prj ファイルを配置します。

3. アプリケーションデザイナーの適切なツールを使用して、オブジェクトを変更します。

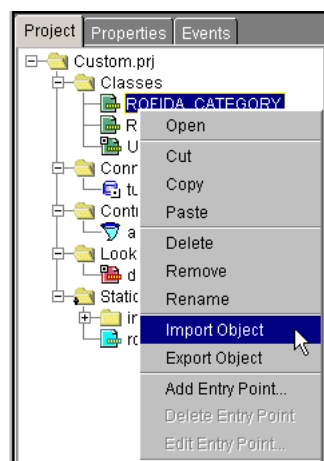
『開発者ガイド』を参照して、これらの作業を実行してください。これらのオブジェクトがソースコード管理システムに配置されるための定義方法に制限はありません。

4. **ファイル > 保存** を選択して、プロジェクトのオブジェクトを保存します。
5. ブラウザパネルのプロジェクトタブで、オブジェクトのエントリを選択します。
6. 右クリックをして、オブジェクトのエクスポートを選択します。

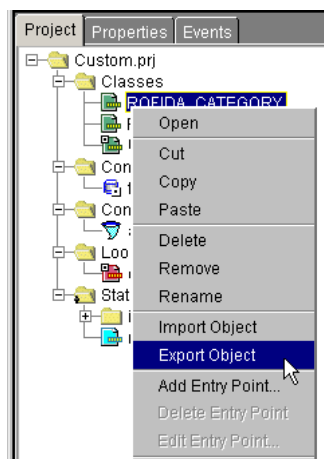
[29 ページ](#)に示されるように、オブジェクトの .nxj パージョンは、関連するディレクトリに書着込まれます。

7. ソースコード管理システムを使用して、.nxj ファイルと .prj ファイルをチェックインします。

ステップ 2



ステップ 5 と 6



## プロジェクトのオブジェクトを削除する

プロジェクトのオブジェクトが必要ではなくなり、プロジェクトから削除されることもあります。もしオブジェクトがエクスポートされている場合は、不注意にインポートし

---

てプロジェクトに追加されないように、その .nxj ファイルもまた削除しておく必要があります。

オブジェクトを削除するステップは、以下のとおりです。

1. オペレーティングシステムツールを使って、.nxj ファイルを削除します。
2. アプリケーションデザイナーで、一致するオブジェクトを以下のステップで削除します。
  - a. ブラウザパネルのプロジェクトタブで、オブジェクトを選択します。
  - b. 右クリックをして **削除** を選択します。  
オブジェクトに関する NXJ プロジェクトファイルが削除されます。
3. ソースコード管理システムからオブジェクトを削除します。

## プロジェクトを抽出する

プロジェクトを抽出することにより、ソース管理システムから .nxj ファイルのプロジェクトのセットを復元します。それからそれをアプリケーションデザイナーにインポートします。その後、作業を行うことができます。

プロジェクトを抽出するステップは、以下のとおりです。

1. ソースコード管理システムを使用して、.nxj ファイルと .prj ファイルをチェックアウトします。  
関連するプロジェクトディレクトリ (exports¥Classes 等) にチェックアウトするファイルを正しく配置してください。古い .nxj ファイルは削除され、上書きされます。
2. アプリケーションデザイナーのブラウザパネルのプロジェクトタブで、インポートにより入れ替えをしたいプロジェクトを開くか、**ファイル > 新規** を選択して、新しいプロジェクトを作成します。  
新しいプロジェクトを作成する場合には、ステップ 1 からのファイルを含んだディレクトリ名に一致するプロジェクト名を指定してください。
3. **ファイル > プロジェクトのインポート** を選択します。  
プロジェクトのインポート操作は、.nxj ファイルを検索してプロジェクトの "exports" ディレクトリ中の関連するディレクトリをスキャンし、その後、NXJ プロジェクトのフォーマットに変換して、適切なフォルダに配置します。  
プロジェクトディレクトリにあるその他のファイルは、.nxj バージョンでなくてもインポート操作によってプロジェクトから削除されます。(削除したオブジェクトは、ファイルシステムからは削除されませんが、**プロジェクト > ファイルの追加** コマンドによってプロジェクトに再度追加することができます。)