



# ASTERIA WARP

## バージョン管理連携機能

第1版:2011年1月25日

ASTERIA では、オープンソースのバージョン管理システムである Apache Subversion と連携することで、作成したフローや関連するファイルをバージョン管理することができます。

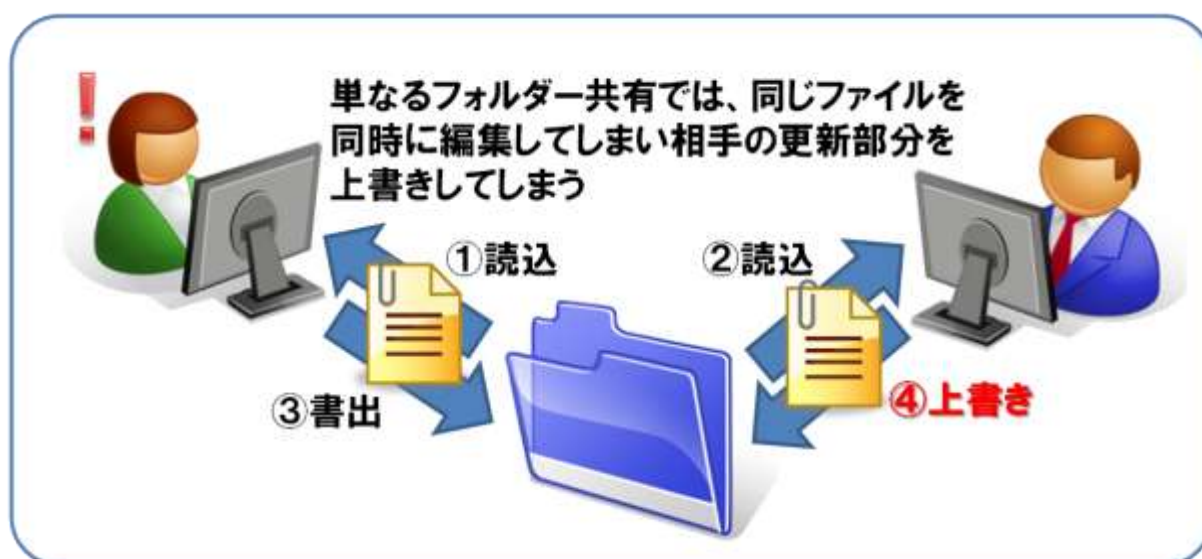
本書では、バージョン管理の概要、および ASTERIA WARP 上でのバージョン管理連携機能を使用した開発手順についてご紹介します。

## 目次

1.バージョン管理とは.....	3
2.ASTERIA におけるバージョン管理連携 .....	4
3.Apache Subversion.....	7
3.1.リポジトリ.....	7
3.2.作業コピー .....	7
3.3.リビジョン .....	8
3.4.競合、マージ、ロック .....	9
3.4.1.両者の修正をマージする .....	10
3.4.2.ファイルを「ロックが必要なファイル」とする.....	10
4.Subversion サーバーのセットアップ .....	12
4.1.インストールマシンの準備 .....	12
4.2.ダウンロード.....	12
4.3.Subversion のバージョン .....	12
4.4.インストール.....	13
4.5.リポジトリの作成.....	14
4.6.ユーザーの作成.....	14
4.7.svnserve をサービスとして設定する .....	16
4.8.リポジトリ内のフォルダー構成を決める.....	17
5.ASTERIA 側の設定.....	18
6.ASTERIA の Subversion 操作.....	21
6.1.フローデザイナーからの操作 .....	21
6.2.フローサービス管理コンソールからの操作.....	22
6.3.flow-ctrl による操作.....	23
付録. バージョン管理関連のトピック.....	24
付録.1.開発機と本番機でのリポジトリの共有 .....	24
付録.2.他の Subversion クライアントとの併用.....	24
付録.3.ブランチについて .....	25

# 1.バージョン管理とは

バージョン管理とは、日々更新されるファイルを、いつ、誰が、どのような修正をおこなったのかを管理することです。主にソフトウェア開発の場面で行われますが、バージョン管理が有効な場面はそれだけに限りません。例えば、Excel ワークブックや PowerPoint ドキュメントなどの業務で使用するファイルを更新する際にも元の状態を残しておきたいことがあります。そうした際に元のファイルを別の場所にバックアップとしてコピーしておき、それからファイルの修正を開始するということが誰もが経験のあることでしょう。



こうした手作業でのコピー&バックアップも一種のバージョン管理です。しかし、このような手作業での管理では更新履歴や差分を参照することができず、日々の業務の中で管理が煩雑になりがちです。少しファイルの数が多くなってくると、結局どれがいつのファイルかわからなくなったという経験を持つ人も多いのではないのでしょうか。

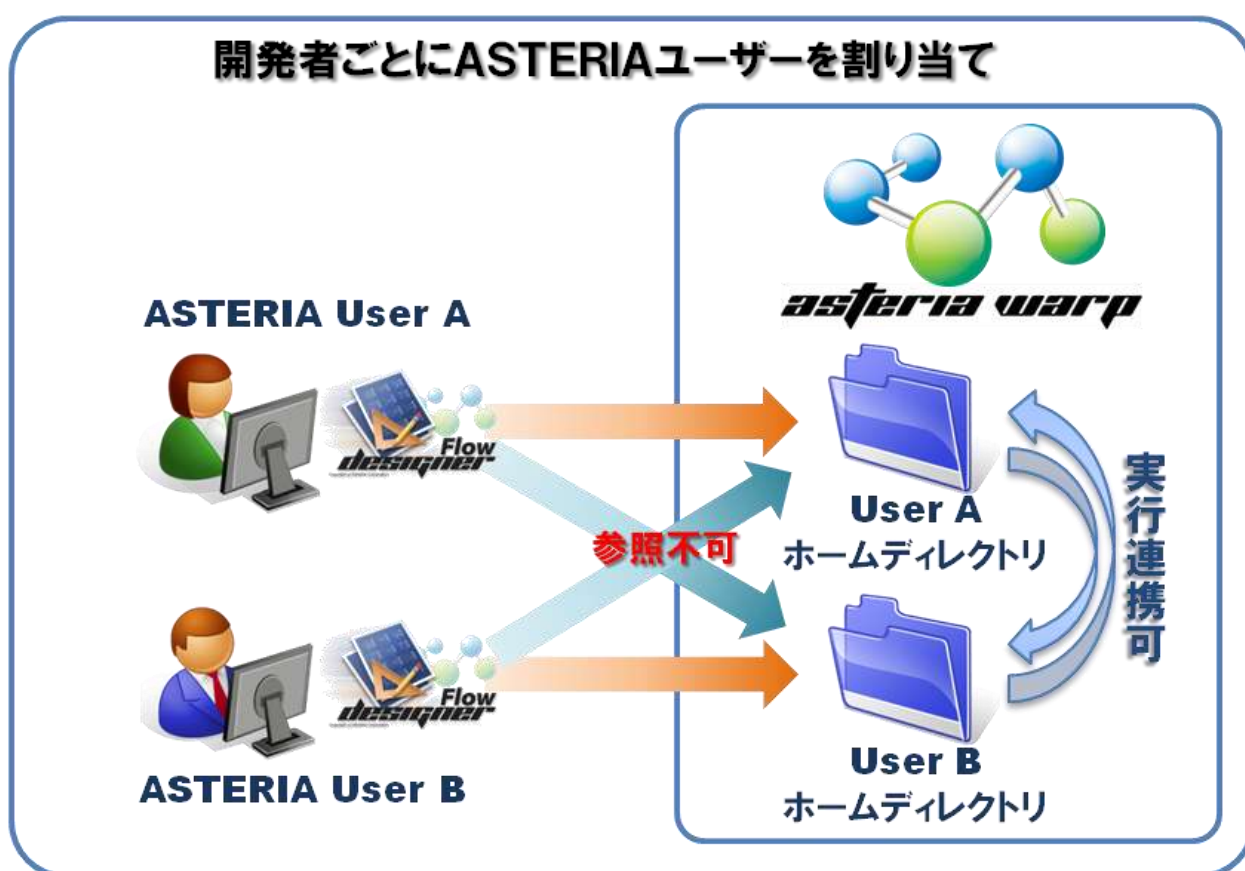
こうしたファイルの更新管理を自動で行ってくれるのがバージョン管理システムです。ファイルの更新時には更新内容をコメントとして記録することができ、そのコメントを手掛かりにいつでも過去のバージョンのファイルを取り出すことができます。また、テキスト形式のファイルであれば更新前後の差分を確認することも可能になります。

バージョン管理システムは一人で使用する場合でも便利ですが、複数の人間が更新する可能性のあるファイルではさらに有効です。その典型的な例がソフトウェア開発です。バージョン管理システムを使用したソフトウェア開発では、いつ、誰が、どのような修正を行ったのかをすべて把握することができ、また仮に複数の人が同時に同じファイルを修正してしまったとしても、自動的なマージや競合の警告といった形でバージョン管理システムがそれぞれの更新を取り成します。

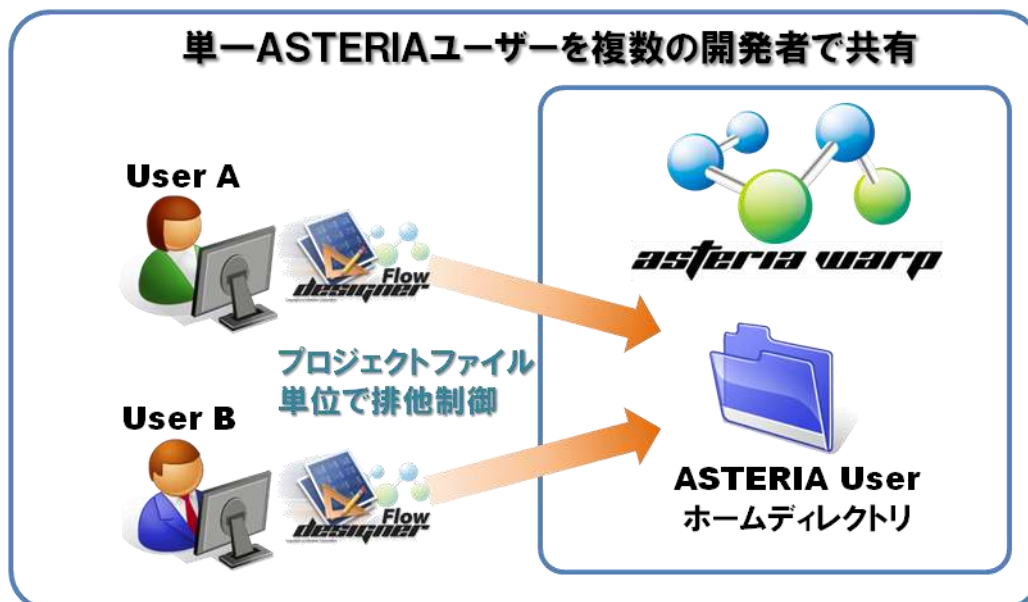
## 2.ASTERIA におけるバージョン管理連携

ASTERIA WARP において、開発者が作成したファイルなどは、全て ASTERIA WARP サーバー上に保存されます。特に作成したプロジェクトファイルや外部変数の定義ファイル、ユーザーコネクションなどは、各 ASTERIA ユーザーアカウントのホームディレクトリーに保存されます。

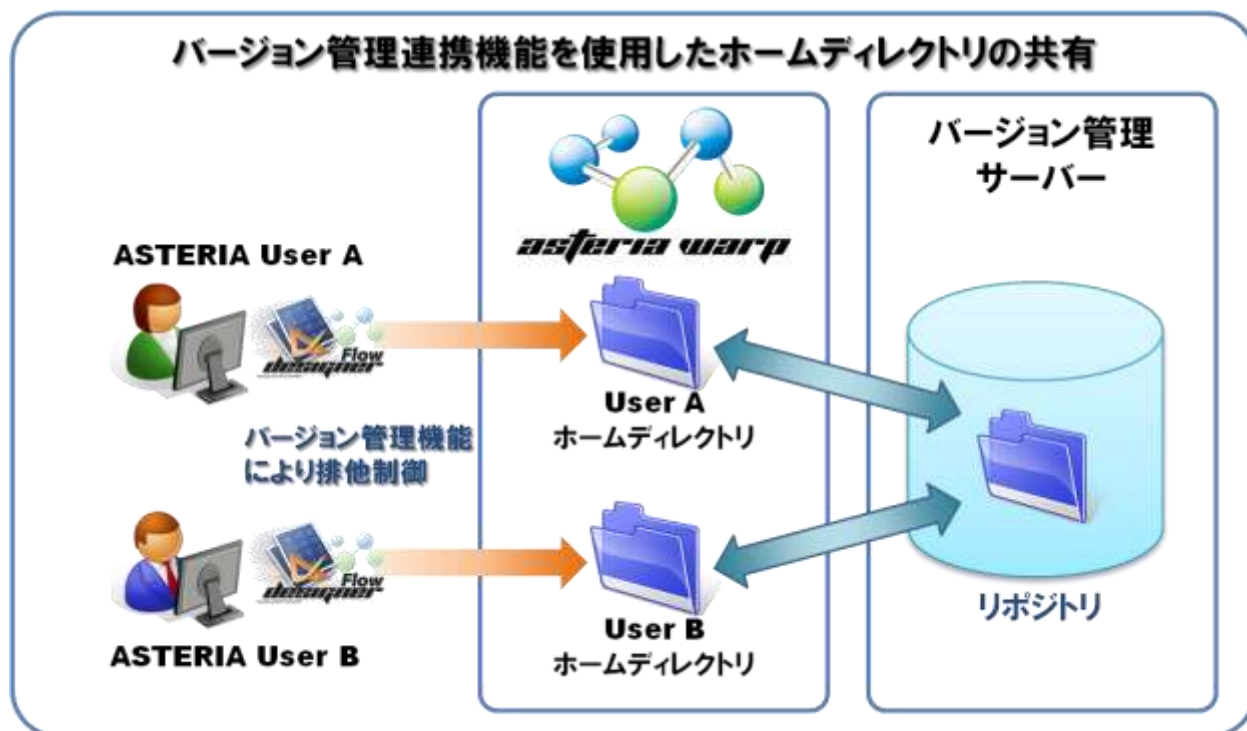
ASTERIA 上で複数のユーザーによりシステムを開発する場合、ASTERIA ユーザーを分け、ホームディレクトリーを分けることで、互いに競合させることなくフローの開発が可能となります。しかし、各ユーザーのフローを相互に参照することはできなくなります(実行は可能です)。



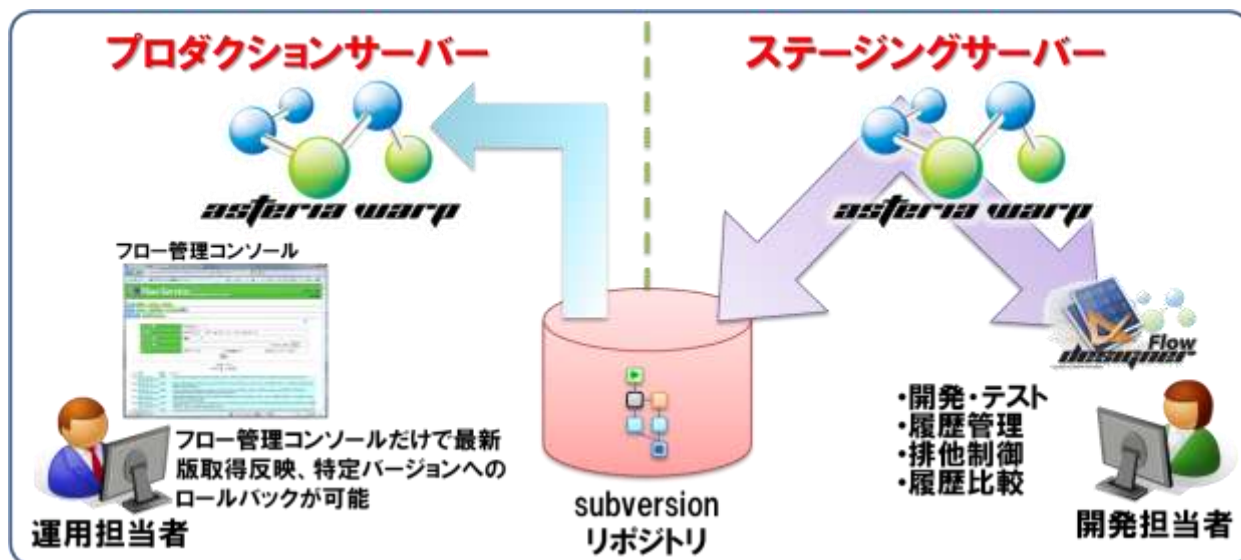
チーム全体でプロジェクト・フローを共有しながら開発するためには、単一の ASTERIA ユーザーを複数の開発者で共有しながら作成する必要があります。この場合、プロジェクトファイル単位で排他制御されますので、上書き保存などの危険性はなく、他ユーザーの開いているプロジェクトファイルをリードオンリーで参照することも可能となります。しかし、誰がどのように編集したかや、プロジェクトファイルへのアクセス権の制御が困難となります。



開発者の人数の多少に関わらず、バージョン管理連携機能を使うことで、両方の利点を得ることができます。また、修正履歴管理、修正のロールバックなど、システム開発統制を実現できます。



また、異なるサーバー間でバージョン管理サーバーを共有することで、本番サーバーとステージングサーバーの移行もスムーズに行うことが可能となります。



## 3. Apache Subversion

Apache Subversion (以下、Subversion) は、Windows や Linux などの複数のプラットフォームで動作するオープンソースのバージョン管理システムです。2000 年より開発が開始され、2009 年 11 月に Apache Incubator プロジェクトに移管されました。現在も活発に開発は継続されており 2011 年 1 月現在の最新版は 1.6.15 です。ライセンスは Apache License であり、誰でも無償で自由に使用することができます。

Subversion は svn、svnadmin などの複数のアプリケーションからなりますが、ここではそれらの個別アプリケーションのコマンドの詳細については説明を省略します。ASTERIA との連携では主に操作はフローデザイナーから行われるため、それらのコマンドを直接使用することはないからです。

ここでは、Subversion のモデルと考え方について説明し、実際に ASTERIA と連携するために必要なもつとも簡単な Subversion サーバーのセットアップ方法を紹介します。

### 3.1. リポジトリ

Subversion インストール後にまず最初にリポジトリを作成します。

リポジトリとは管理対象のすべてのファイルとその履歴、メッセージなどの保管庫となるものです。通常のファイルシステムと同じように複数のファイルやフォルダーをツリー構造で保持します。

リポジトリを作成すると conf、db などの複数のフォルダーがそこに作成されます。conf フォルダーにあるファイルは設定変更のため編集することがありますが、通常、それ以外のファイルは直接扱うことはありません。

リポジトリのあるホスト PC は Subversion におけるサーバーとなります。対して、このリポジトリに接続してファイルを出し入れするアプリケーションは Subversion クライアントと呼ばれます。Subversion クライアントには本体に同梱されているコマンドラインアプリケーション「svn」の他にも Windows エクスプローラーに統合できる「TortoiseSVN」や Eclipse プラグインの「Subversive」などいくつかの種類があり、ASTERIA もそうした Subversion クライアントの一種です。

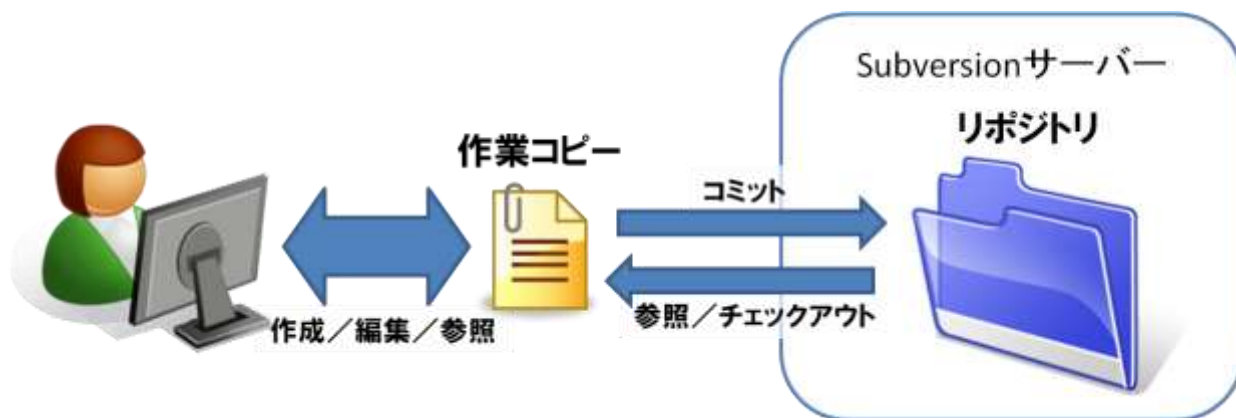
リポジトリ作成後は Subversion クライアントがリポジトリにファイルを追加、削除したり、更新したファイルを上書き保存したりします。Subversion クライアントがファイルを削除してもそのファイルが完全に失われることはありません。いつでもリポジトリから削除前のファイルを取り出すことができます。同様に、何度ファイルを上書き保存してもすべての版はリポジトリ内に残っているので、いつでも元に戻すことができます。

Subversion クライアントが、リポジトリが作成されたホスト(Subversion サーバー)と同一ホスト上で動作する場合は、これ以外に設定を行わなくても直接リポジトリに接続することができますが、ネットワークを介して別ホストからアクセスする場合は Apache Web サーバーまたは後述する svnserve を使用してリポジトリを公開する必要があります。

## 3.2.作業コピー

Subversion クライアントはリポジトリからファイルを取り出して作業を行います、そのファイルは「作業コピー」と呼びます。

ユーザーは自分のローカルマシンの中で作業を行う場所を決めて、そこにリポジトリ内のファイルツリーのすべて、または任意のフォルダー以下の一部をそこに取り出します。この操作を「チェックアウト」と呼びます。



チェックアウトによってリポジトリからユーザーのローカルフォルダーにコピーされたファイルが作業コピーです。作業コピーは文字通りリポジトリにあるファイルのコピーであり、作業コピーのファイルを編集したり削除したりしても、直ちにリポジトリには反映されません。作業コピーでの編集内容は Subversion クライアントから「コミット」という操作を行ってはじめてリポジトリに反映されます。

コミットは複数のファイルに対して一度に行うことができ、コミットする際に何のための修正であったかをコメントとして記録することができます。

繰り返しになりますが、作業コピーでどのような編集を行ったとしてもそのファイルはリポジトリでバージョン管理されているのでいつでも元に戻すことができます。また、ローカルマシン上で作業コピーが不要になった場合はいつでも削除できます。作業コピーがなくなってもリポジトリは何の影響も受けないからです。

### メモ

チェックアウトを行うとそのフォルダーとすべてのサブフォルダーに「.svn」という隠しフォルダーが作成されます。これは Subversion クライアントが作業コピーの状態を管理するために使用するファイルの置き場所です。

隠しフォルダーは設定によっては最初から表示されませんが、表示されている場合もこのフォルダー内のファイルを直接操作してはいけません。

## 3.3.リビジョン

リポジトリでは連番でリビジョン番号が管理されています。

リポジトリを新規に作成した際のリビジョン番号は0で、コミット操作が行われるごとに1ずつリビジョン番号は増えていきます。つまり、リポジトリ内のファイルの内容が変わるごとにリポジトリ全体のリビジョンが上がっていくこととなります。

**Subversion** クライアントの作業コピー上でファイルやフォルダーを選択して履歴を表示させると、そのファイル、またはそのフォルダー以下のファイルのリビジョンの一覧を、コミット時に入力したコメント付きで見ることができます。リビジョンの一覧では、そのファイルを、いつ、誰が、何のために更新したかを確認することができます。テキストファイルであればリビジョン間の差分を確認することも可能です。

過去のバージョンのファイルを作業コピーとして取得するなどのときにはこのリビジョンを指定して行います。

### メモ

正確には「インポート」(任意のフォルダー以下のファイルをまとめてリポジトリに取り込む)などのコミット以外の操作でもリビジョン番号があがるがありますが、考え方は同じです。

## 3.4.競合、マージ、ロック

ひとりのユーザーが一箇所の作業コピーだけを使用して作業を行っているのであれば、リポジトリに反映される内容はその作業コピーからのコミットだけです。この場合、作業コピーは常にリポジトリの最新リビジョンと同じ内容になりますし、コミットが他の誰かの変更と重なる心配はありません。

しかし複数の人間が同じリポジトリから作業コピーを取得して作業を行っている場合は、以下の様な状況が発生することがあります。

- ユーザーA がファイル 1 を編集開始
- すこし遅れてユーザーB が同じファイル 1 を編集開始
- ユーザーA がファイル 1 をコミット
- ユーザーB がファイル 1 をコミットしようとする。しかしこのファイルにはユーザーA の更新内容は含まれてないので。。

もしも最後の段階でユーザーB の変更がそのままコミットできてしまうと、ユーザーA の編集結果は誰も気がつかないうちに消えてしまうことになります。実際にはこの場合ユーザーB のコミット操作は「編集を行ったファイルが最新のリビジョンではない」という理由で失敗します。

こうした状況を「競合」と呼びます。

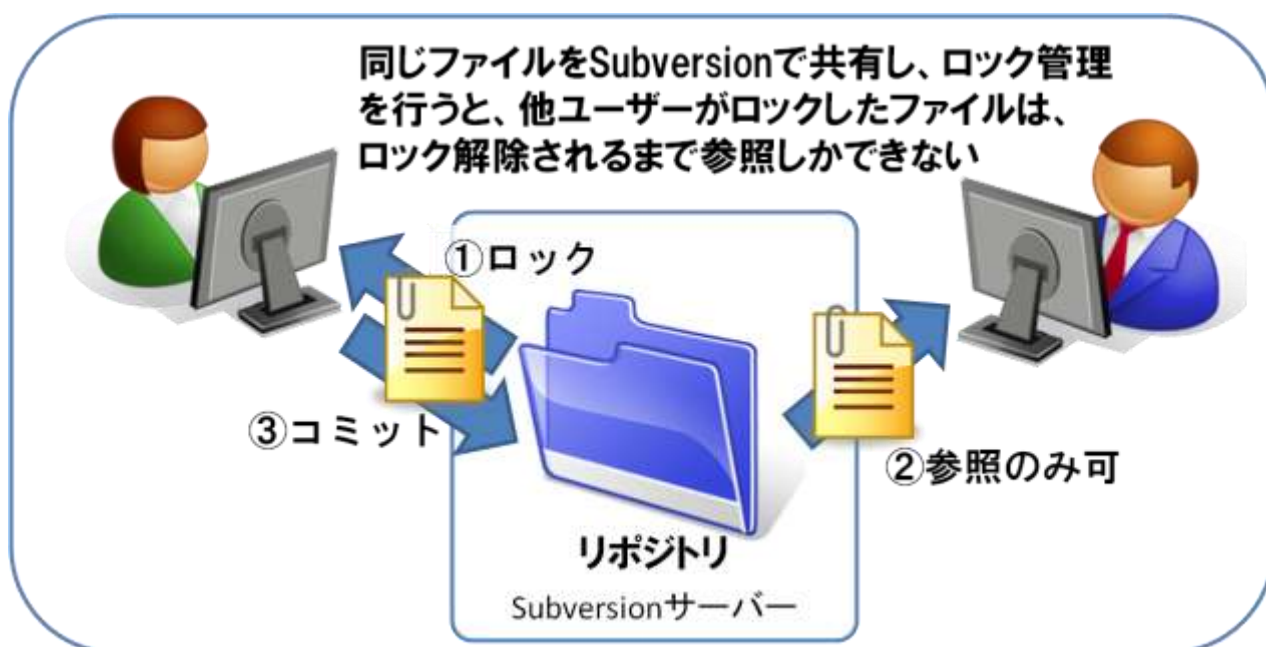
複数の人間が同一リポジトリで作業を行う場合、常に競合が発生する可能性があります。この問題に対しては次節以降で説明する2つの解決方法があります。

### 3.4.1.両者の修正をマージする

第一の方法は、両方の修正をマージして反映したファイルを作成し、それをコミットすることです。これをマージ法といいます。対象がテキストファイルで両者の修正箇所が重なっていない場合、Subversionはこのマージを自動的に行います。自動的なマージができない場合は手作業でファイルをマージすることになりますが、Subversionは修正箇所を強調したマージを補助するファイルを作成します。手作業でマージを行った場合はもちろん、自動的なマージが行われた場合もコミット前の最終的な確認は人間の目で行う必要があります。

### 3.4.2.ファイルを「ロックが必要なファイル」とする

Subversionでは個々のファイルに対して、「ロックが必要なファイル」という設定をすることができます。「ロックが必要なファイル」とは編集を行う前にそのファイルのロックを取得する必要があるファイルのことです。この場合ユーザーAがファイルを編集するためには、まずそのファイルのロックを取得する必要があります。そして、ユーザーAがロックを持っている状態ではユーザーBはロックを取得することができないので編集が行えません。ユーザーBがロックを取得できるのはユーザーAが編集を終えコミットしてロックを解放した後です。この方法を使用した場合、異なるユーザーが同一のファイルを編集し、競合が発生すること自体を回避できます。



## メモ

実際には「ロックが必要なファイル」となっていないファイルに対してもロックをかけることができますし、コミットの前にロックを解放することも可能です。ただし、そこは運用ルールとして徹底しなければ、オペレーションミスなどにより、結局競合が発生してしまうので注意が必要です。

どちらの方法にも一長一短がありますが、可能な場合はマージ法を取る方が効率が良いと言われています。ロックを取得するのは一手間増えますし、誰かがロックをかけたまま解放するのを忘れることもありえます。

バイナリファイルではマージ法は使えませんが、例えばイメージファイルに対してデザイナーが1名しかいないなどほとんど競合が発生しないことが分かっているのであれば、必ずしもすべてのバイナリファイルを「ロックが必要なファイル」とする必要はありません。そのあたりはバージョン管理の運用ポリシー次第となります。

むしろ重要なのは、競合の可能性を減らすために個々の作業者が普段からこまめにアップデートをかけて作業コピーの状態を最新リビジョンに保つことです。そしてそれ以上に重要なのは、作業者間でコミュニケーションを取ってお互いの作業を把握しておくことです。

## 4.Subversion サーバーのセットアップ

前章でバージョン管理の考え方について説明しました。次は実際に触ってみましょう。ここでは Windows マシンに Subversion サーバーをセットアップする手順を説明します。

### 4.1.インストールマシンの準備

Subversion はそれほど大きなリソースを必要とするソフトウェアではありません。日常的に使用しているパソコンにインストールしても問題なく動作します。

管理するファイルの量が多くなれば相応のディスクサイズを消費しますが、それについてだけ気をつければ ASTERIA サーバーが稼働しているマシンにインストールすることにも問題ありません。

### 4.2.ダウンロード

Subversion は、以下の Apache の Subversion サイトのリンクからダウンロードできます。

[Apache Subversion](http://subversion.apache.org/)

(<http://subversion.apache.org/>)

画面左の「Getting Subversion > Binary Packages」から次のページに移るとプラットフォームごとのダウンロード先がリストされたページが表示されます。

Windows 版では4つのダウンロード先が表示されており、それぞれのリンク先では独自にビルドした Subversion が配布されています。どれを選択しても基本的な機能には差がありません。VisualSVN のように独自に機能拡張されたものもありますが、ここでは SlikSVN 例に説明します。

#### メモ

ここで SlikSVN を選択したのは以下の理由からです。

- msi 形式で配布されているのでインストールが簡単
- 日本語ヘルプが充実している

もちろん他のサイトから入手したものを使用しても問題ありません。

なお、SlikSVN では Windows の 32bit 版および 64bit 版のインストーラーがダウンロード可能となっているので、お使いの OS に合わせたインストーラーをダウンロードしてください。

## 4.3.Subversion のバージョン

インストールの説明に入る前にここで Subversion のバージョンについて補足しておきます。

2011 年 1 月現在、Subversion の最新バージョンは「1.6.15」です。

ASTERIA は Subversion1.6 相当の Subversion クライアント機能を保持していますが、使用している機能は 1.5 までの機能としています。基本的には Subversion はサーバーとクライアントが異なるバージョンであっても動作しますので、バージョン 1.5 以降の Subversion サーバーであれば ASTERIA との連携で利用できます。特に理由がなければ、新規にインストールする場合は 1.6 系の最新版を推奨します。

## 4.4.インストール

SlikSVN のインストール手順を以下に記します。

ここでは、Windows プラットフォーム上で Slik-Subversion-1.6.xx-win32.msi を使用したインストール手順をご紹介します。

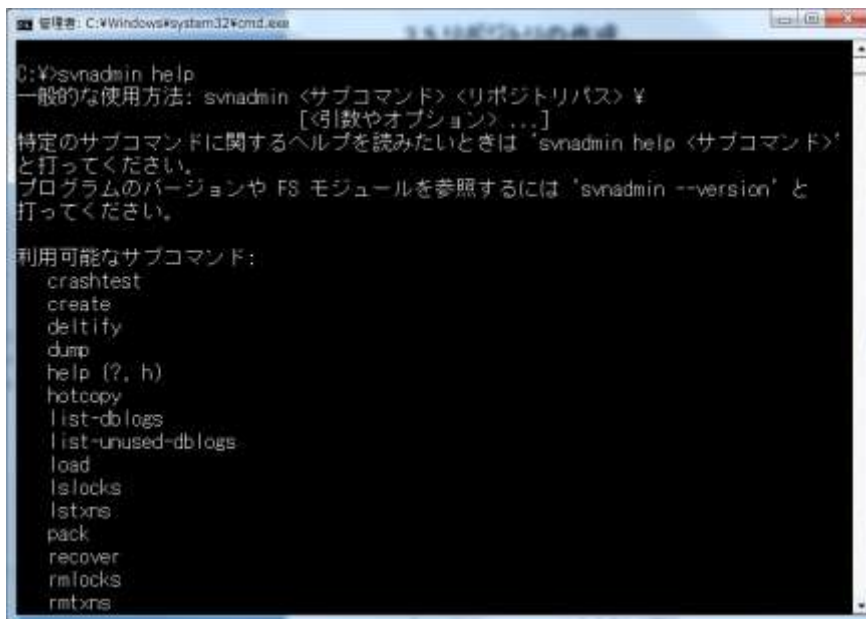
1. ダウンロードした「Slik-Subversion-1.6.xx-win32.msi」をダブルクリックしてインストーラーを起動
2. ライセンス確認画面でライセンスに同意して Next をクリック
3. インストールフォルダーの選択画面でインストール先を選択して Next をクリック  
(通常は変更する必要はありません。以下の説明では「c:\Programfiles\SlikSVN」にインストールされているものとして説明します。)
4. セットアップ種別の選択画面で「Custom」ボタンをクリック
5. カスタムセットアップ画面で「SvnServe」をインストールに含めて Next をクリック
6. Install ボタンをクリックしてインストールを開始
7. インストールの完了画面が表示されたら「Finish」ボタンをクリックしてインストーラーを終了してください。インストールに成功すると「c:\Program files\SlikSvn\bin」に Subversion のアプリケーションが配置されます。このパスはインストーラーによって環境変数「Path」に自動的に設定されますが、もしうまく設定されていない場合は手作業で設定を行ってください。

## 4.5.リポジトリの作成

インストール後に最初に行う作業はリポジトリの作成です。リポジトリの作成には「`svnadmin`」コマンドを使用します。まずはコマンドプロンプトを起動して

```
C:\>svnadmin help
```

と入力してください。インストールに成功してパスが正しく設定されていれば、コマンドのヘルプが表示されます。表示されない場合は環境変数「Path」の設定を見直してください。



```

C:\Windows\system32\cmd.exe
C:\>svnadmin help
一般的な使用方法: svnadmin <サブコマンド> <リポジトリパス> ¥
                    [<引数やオプション> ...]
特定のサブコマンドに関するヘルプを読みたいときは 'svnadmin help <サブコマンド>'
と打ってください。
プログラムのバージョンや FS モジュールを参照するには 'svnadmin --version' と
打ってください。

利用可能なサブコマンド:
  crashtest
  create
  deltify
  dump
  help (? , h)
  hotcopy
  list-dblogs
  list-unused-dblogs
  load
  lslocks
  lstxns
  pack
  recover
  rmllocks
  rmtxns
  
```

`svnadmin` ではサブコマンド「`create`」でリポジトリを作成します。リポジトリをどのフォルダーに作成するかを決めたらそのフォルダーに移動して `create` を実行します。「`c:\repos`」にリポジトリを作成する場合は次のような手順になります。

コマンドプロンプトで「`c:\>`」に移動

```
C:\>svnadmin create repos
```

と入力して実行

リポジトリを作成すると、エクスプローラー上でそのフォルダーが作成されていることが確認できます。

## 4.6.ユーザーの作成

今回はリポジトリの公開に `svnserve` を使用しますが、`svnserve` はデフォルトではユーザー認証不要の設定になっています。つまりリポジトリに接続すれば誰でもそこにあるファイルを読むことができます。

ASTERIA の仕様として Subversion に接続する際には認証が必須となっているので、認証を行うように変更します。

認証設定を行うひとつの方法としてパスワードファイルを使用する方法があります。これはパスワードファイルに記載されたユーザー名、パスワードの組でユーザー認証を行うもっとも簡単な設定方法です。

以下の手順でユーザー名とパスワードをリポジトリに登録します。

1. リポジトリ内の「conf」フォルダーにあるファイル「svnserve.conf」(先の例では、`c:\repos\conf\svnserve.conf`)をテキストエディタで開く  
 svnserve.confは、svnserveの設定ファイルです。「#」で始まる行はコメントであり、デフォルトでは何の設定も行われていません。

2. 以下の2行を修正します。

**修正前**

----

```
#anon-access=read
#auth-access=write
```

**修正後**

----

```
anon-access=none
auth-access=write
```

「anon-access」は認証なしの場合のアクセス権、「auth-access」は認証ありの場合のアクセス権です。これを修正して「認証なしの場合はアクセス権なし、認証ありの場合は読み書き可」に変更します。

3. password-db の設定行を変更します。

**修正前**

----

```
#password-db=passwd
```

**修正後**

----

```
password-db=passwd
```

「password-db」はパスワードファイル名の指定です。コメント解除してファイル「passwd」をパスワードファイルとして指定します。

4. リポジトリ内の「conf」フォルダーにあるファイル「passwd」(先の例では、`c:\repos\conf\passwd`)をテキストエディタで開く。3.で指定したパスワードファイルです。svnserve.xconfと同じく「#」で始まる行はコメント行です。  
 このファイルの[users]以下にユーザー名とパスワードを「<ユーザー名>=<パスワード>」の形式で指定します。

**行追加**

----

```
tarou=password
hanako=password
```

ここまでの設定によりリポジトリにアクセスする際の認証に passwd ファイルで指定したユーザー名とパスワードが使用されるようになります。

## メモ

この認証設定方法には

- パスワードが平文でファイルに保存される
- ユーザーが自分でパスワードを変更できない

などのセキュリティ上の欠点があります。

ユーザー認証の主目的がリポジトリの変更を行ったユーザーの識別であればこれでも十分ですが、セキュリティ上の問題がある場合は Apache や SSH を使用する別の認証設定を検討してください。

## 4.7.svnserve をサービスとして設定する

svnserve はリポジトリを公開するためのサーバーアプリケーションです。起動しておくとしリポジトリがあるサーバーと異なる PC から「svn://」という独自プロトコルを用いてリポジトリにアクセスできるようになります。

Subversion を恒常的に使用する場合、svnserve を Windows サービスに登録して常時起動することが可能です。svnserve を Windows サービスに登録するためのコマンドラインは以下の通りです。

```
C:¥>sc create svnserve
        binpath= "¥"C:¥Program Files¥SlikSvn¥bin¥svnserve.exe¥"
        --service --root c:¥repos" displayname= "Subversion"
        depend= tcpip start= auto
```

sc コマンドはオプションの指定方法やスペースのエスケープに独自の記法があります。インストールパスとリポジトリの作成パスが同じであれば上記から改行を取り除いた文字列をコピー & ペーストすることをお勧めします。

サービスの登録に成功したら Windows メニューの「コントロールパネル > 管理ツール > サービス」から「Subversion」という名前で表示されているサービスを開始します。

次回 OS 起動時からはサービスの設定により自動的に svnserve を起動します。

## 注意

### ・権限について

sc create コマンドは管理者にしか実行できません。

```
[SC]OpenSCManagerFAILED5:
```

というエラーが表示される場合はコマンドプロンプトを右クリックメニューの「管理者として実行」から起動し直してください。

### ・IPv6 について

IPv6 がデフォルトになっている環境で svnserve を実行すると IPv4 アドレスを使用する Subversion クライアントからの接続に失敗します。

外部の Subversion クライアントからの接続でエラーとなる場合は sc コマンドの引数「binpath」に「--listen-host0.0.0.0」を追加して登録し直してください。

## 4.8.リポジトリ内のフォルダー構成を決める

ここまでの設定で ASTERIA から Subversion に接続するための準備が整いましたが、その前にもうひとつ、リポジトリのフォルダー構成を先に決めておきます。

リポジトリ内のフォルダー構成の考え方は通常のファイル保存の場合と同じです。初期状態ではリポジトリ内は空の状態ですが、そこにユーザーが自由にフォルダーを作成してファイルを整理することができます。

フォルダー構成は後から変更することもできますが、その変更もバージョン管理によって履歴が残されます。できるだけフォルダーツリーの上位の構成については最初に決めた構成を使い続けることを推奨します。

このリポジトリを使用するのが単一の ASTERIA サーバーのみである場合は、リポジトリのルートに「asteriahome」というフォルダーを作成してそこに各ユーザーのホームディレクトリを追加していく方法を推奨します。複数の ASTERIA サーバーで共有するのであれば「<サーバー名>/asteriahome」とするのが良いでしょう。

ここでは、リポジトリ直下に「asteriahome」というフォルダーを作成するものとして説明を進めます。方法はいくつかありますが、ここでは svnserv を使用した、コマンドラインから svn プロトコル経由で直接 URL を指定してフォルダーを作成します。

```
C:¥>svn mkdir svn://localhost/asteriahome -m"Make asteriahome"
--username <username> --password <password>
```

「svn mkdir」コマンドを URL 指定で実行した場合、フォルダー作成と同時にコミットまで行われます。

svn プロトコルではサーバー名の指定によりリポジトリのルートディレクトリが参照されます。そこに「asteriahome」というフォルダーを作成するので作成対象のフォルダーの URL は「svn://localhost/asteriahome」となります。

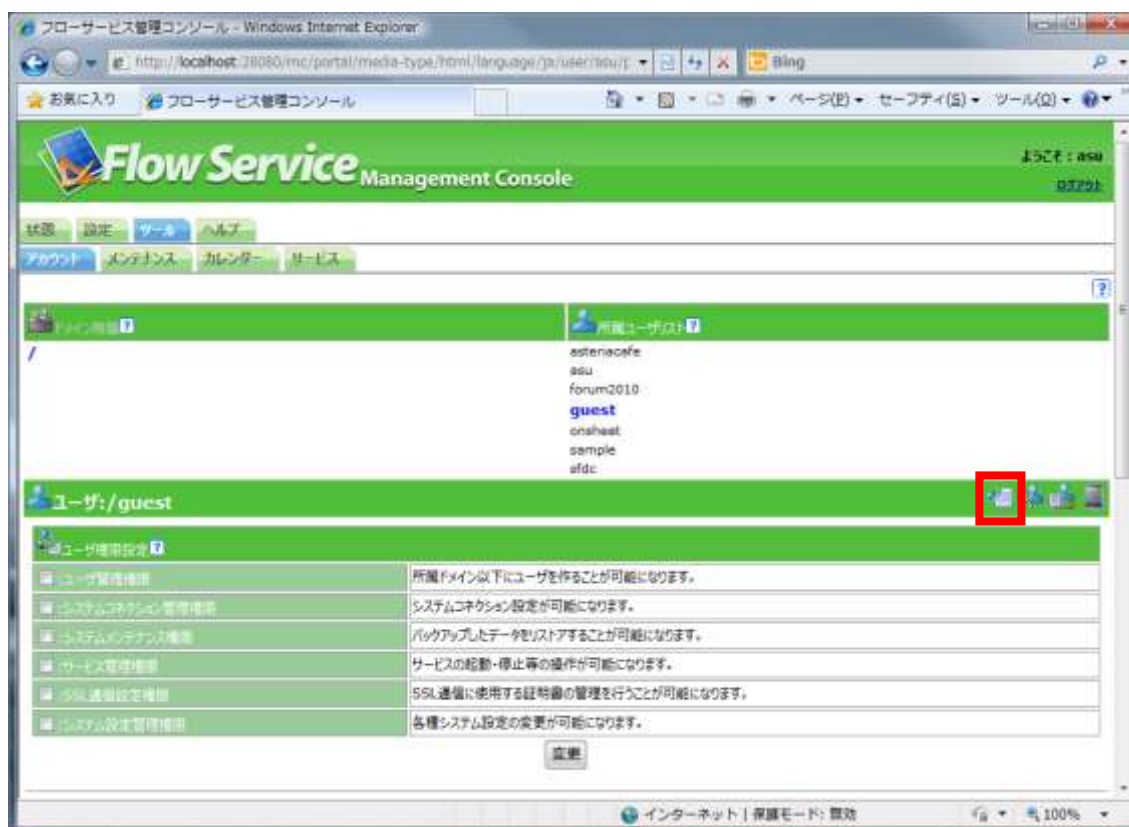
「-m」にはコミット時のコメントを指定し、「--username」と「--password」にはパスワードファイルで指定したユーザー名とパスワードを指定します。

### メモ

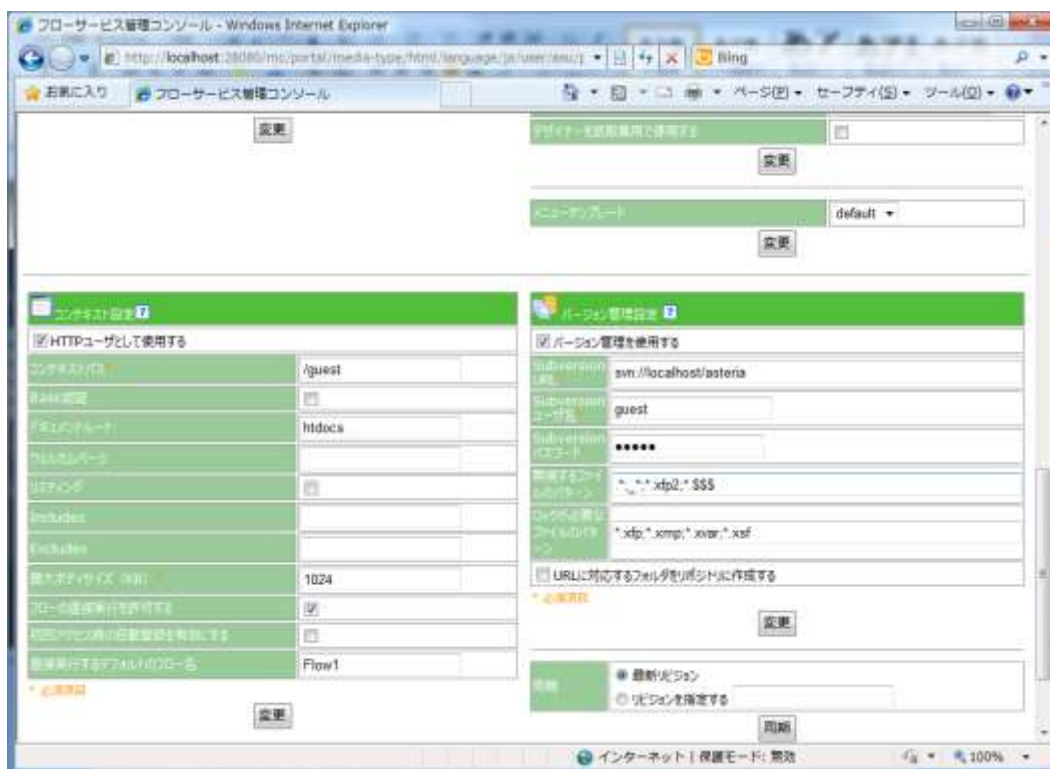
ここではその他の svn コマンドの解説は省略します。詳細は Subversion 関連の書籍または Web サイトを参照してください。

## 5.ASTERIA 側の設定

ASTERIA 側で Subversion を使用するためには、管理者がフローサービス管理コンソールで ASTERIA ユーザーと Subversion ユーザーの紐付けを行います。まず、ユーザー管理権限を持つ ASTERIA ユーザーで管理コンソールにログインし、バージョン管理を使用する ASTERIA ユーザーの設定画面を表示します。(メニューのツール>アカウント>所属ユーザーリストよりユーザーを選択)



ユーザー情報の画面で詳細設定のアイコンをクリックすると画面下部に「バージョン管理設定」が表示されます。



各設定項目の意味は以下のとおりです。

### バージョン管理を使用する

この ASTERIA ユーザーで作成したファイルをバージョン管理するかどうかの指定です。バージョン管理する場合はチェックします。

### SubversionURL

この ASTERIA ユーザーで作成したファイルを保管する Subversion のリポジトリパスを指定します。先ほど作成したリポジトリで「asteriahome」を ASTERIA 用のフォルダーと決めたので、例えば ASTERIA ユーザー「guest」のファイルを管理する URL の場合は

`svn://<SERVERNAME>/asteriahome/guest`

のように指定します。

この時点でリポジトリには「asteriahome/guest」というフォルダーは存在しないので、最後の「URLに対応するフォルダーをリポジトリに作成する」もチェックします。

### Subversion ユーザ名

Subversion のユーザー名を指定します。このユーザー名はパスワードファイルに指定したユーザー名です。ASTERIA ユーザーとは無関係であり ASTERIA ユーザー名とは異なるユーザー名でも構いません。ただし、デザイナーで編集時、コミットした際に履歴情報として登録されるユーザー名は、この Subversion ユーザー名となります。

## Subversion パスワード

Subversion のユーザーのパスワードを指定します。このパスワードはパスワードファイルに指定したパスワードです。

## 無視するファイルのパターン

バージョン管理を行わないファイルのパターンをセミコロンまたはカンマで区切り指定します。デフォルトでは「.\*;\_\*.xfp2;\*.\$\$\$」となっています。このパターンは

- 「.\*」「\_\*」 : ASTERIA の隠しフォルダー
- 「\*.xfp2」 : プロジェクトコンパイル時の中間ファイル
- 「\*.\$\$\$」 : フローデザイナーのロックファイル

を表しており、通常はこれらのパターンをこの設定から外す必要はありません。これ以外にバージョン管理する必要のないことがわかっているファイルパターンがあるならば追加で設定します。

## ロックが必要なファイルのパターン

競合対策として「ロックが必要なファイル」とするファイルのパターンをセミコロンまたはカンマ区切りで指定します。デフォルトでは「\*.xfp;\*.xmp;\*.xvar;\*.xsf」となっています。このパターンは

- 「.xfp」 : フローのプロジェクトファイル
- 「\*.xmp」 : 関数コレクションのファイル
- 「\*.xvar」 : 外部変数セットのファイル
- 「\*.xsf」 : ストリーム定義セットのファイル

を表しています。これらのファイルはバイナリファイルではありませんが、競合時のマージを正確に行う方法がない(ファイルフォーマットが公開されていないためユーザーが手作業でのマージや自動マージの結果検証を行うことができない)ためにロックを必要としています。

これ以外に「\*.xls」(Excel ファイル)など他にバージョン管理したいバイナリファイルのファイルパターンがあるならば追加で設定します。逆に開発者が一人だけの場合など競合が発生しないことが分かっている場合は、この設定は空にしても問題ありません。

## URL に対応するフォルダーをリポジトリに作成する

はじめてバージョン管理を設定する場合など、この ASTERIA ユーザーに対応するフォルダーがリポジトリに存在しない場合にチェックすると、リポジトリに SubversionURL に指定した URL に対応するフォルダーが作成されます。

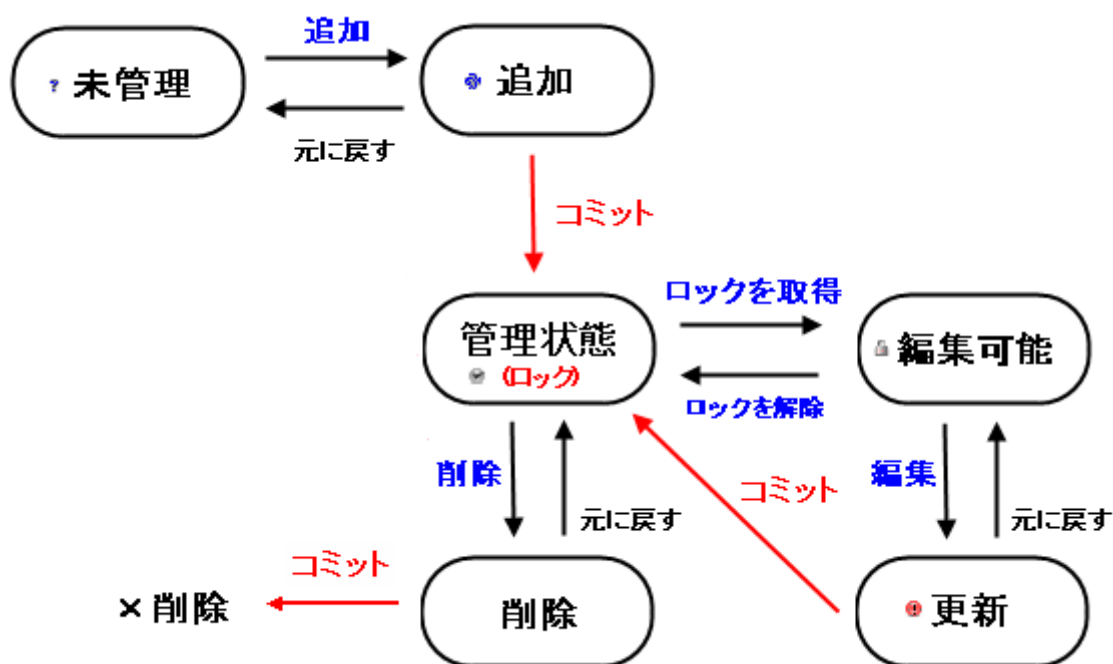
バージョン管理設定を行っても既にユーザーのホームディレクトリに存在している既存ファイルは自動的にリポジトリに追加されません。既存ファイルは未管理状態となるため必要に応じてデザイナーから追加、コミットしてください。

「無視するファイル」「ロックが必要なファイル」はここでのパターンマッチングによる自動設定以外にもファイルを選択して個別に設定することも可能です。

## 6.ASTERIA の Subversion 操作

ASTERIA からの Subversion 操作はフローデザイナー、フローサービス管理コンソール、flow-ctrl から行うことができます。フロー開発時には、ほとんどの操作はフローデザイナーから行います。

フローデザイナーでのバージョン管理のステータス遷移



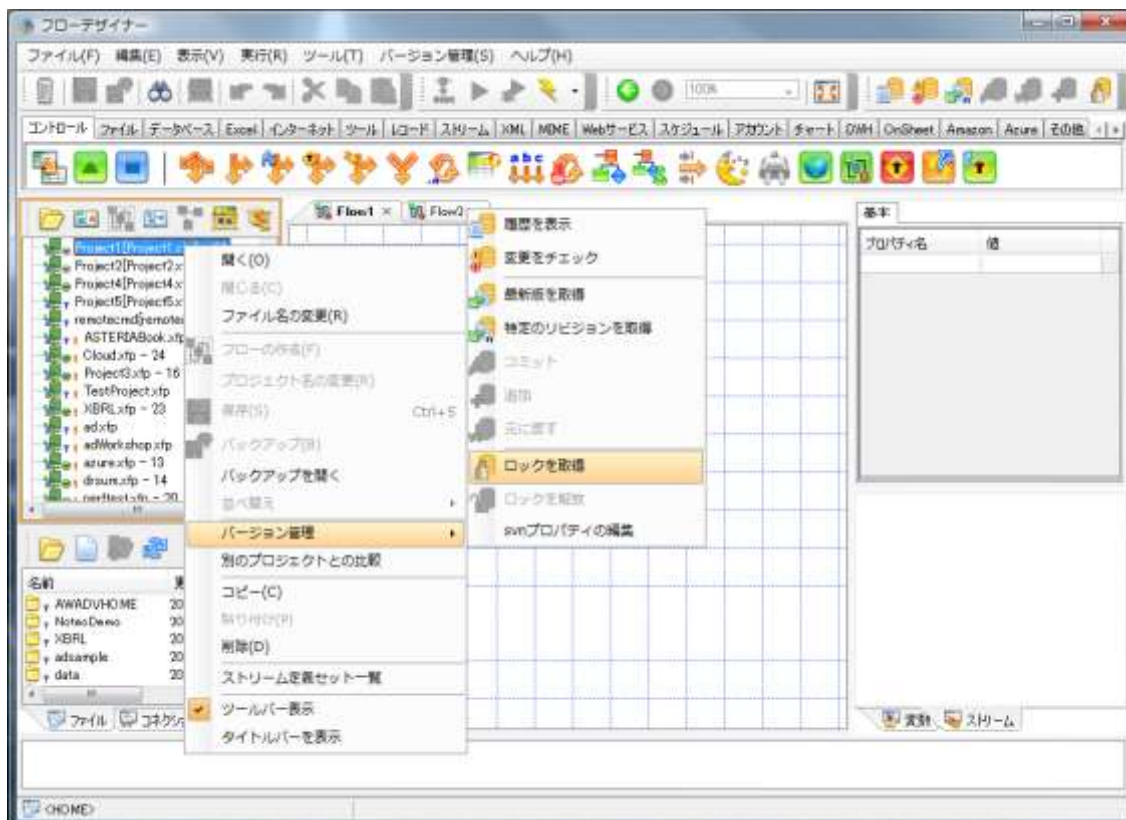
### メモ

クライアントとサーバーの関係を厳密に言えば Subversion クライアントなのは ASTERIA サーバー (FlowService)となります。フローデザイナーは、直接 Subversion サーバーとやりとりせず、ASTERIA サーバーを経由して Subversion へアクセスします。

### 6.1.フローデザイナーからの操作

バージョン管理が設定された ASTERIA ユーザーにフローデザイナーで接続するとツリーペインとファイルペインにバージョン管理の状態アイコンが表示されます。バージョン管理関連の操作はすべての以下のいずれかから実行でき、主要な操作はほとんどツールバーから実行できます。

- ツリーペインでノードを選択しての右クリックメニュー
- ファイルペインでファイルを選択しての右クリックメニュー
- バージョン管理ツールバー



フローデザイナーからは追加、コミットなどの ASTERIA がサポートするすべての Subversion クライアント操作が実行できます。具体的な操作方法についてはフローサービスマニュアルの「[フローデザイナー](#)」-「[バージョン管理](#)」以下のマニュアルを参照してください。

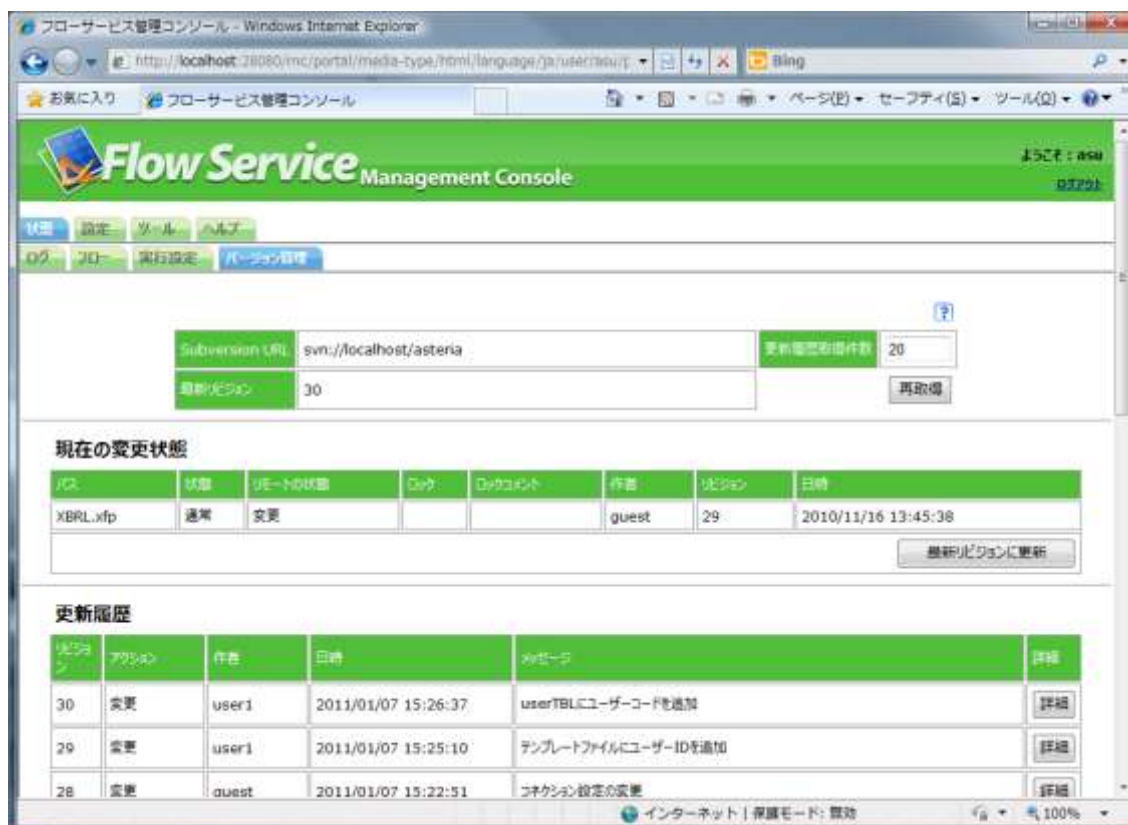
オンラインドキュメント URL:

[http://asteria.jp/documentation/warp/current/flow/designer/index\\_guide.html#svn.html](http://asteria.jp/documentation/warp/current/flow/designer/index_guide.html#svn.html)

## 6.2.フローサービス管理コンソールからの操作

バージョン管理が設定された ASTERIA ユーザーでフローサービス管理コンソールにログインした場合、メニューの状態>バージョン管理の画面では以下の操作を行うことができます。

- 現在の作業コピーの状態確認
- 更新履歴の確認
- 最新版または任意のリビジョンへの作業コピーの更新



フローサービス管理コンソールから行える操作はこの3つだけで、ファイルの追加、削除、コミットなどは行えません。具体的な操作方法についてはフローサービスマニュアルの[フローサービス管理コンソール\(FSMC\)オンラインヘルプ](#)を参照してください。

オンラインドキュメント URL:

[http://asteria.jp/documentation/warp/current/flow/mc/status.html#versioncontrol\\_top](http://asteria.jp/documentation/warp/current/flow/mc/status.html#versioncontrol_top)

## 6.3.flow-ctrl による操作

バージョン管理が設定された ASTERIA ユーザーで flow-ctrl にログインした場合、バージョン管理関連のコマンドとして以下のコマンドを提供します。

- svn stauts-現在の作業コピーの状態確認
- svn log-更新履歴の確認
- svn update-最新版または任意のリビジョンへの作業コピーの更新

flow-ctrl から行える操作はこの3つだけで、ファイルの追加、削除、コミットなどは行えません。具体的な操作方法についてはフローサービスマニュアルの[flow-ctrl コマンドリファレンス](#)を参照してください。

オンラインドキュメント URL:

<http://asteria.jp/documentation/warp/current/flow/designer/flow-ctrl.html#svn>

## 付録. バージョン管理関連のトピック

ここではバージョン管理を使用する上でのトピックをご紹介します。

### 付録.1.開発機と本番機でのリポジトリの共有

開発機としてステージングサーバーが導入されている場合、通常、本番機と開発機のホームディレクトリは同じ内容を共有します。こういった環境では、バージョン管理機能を使用することで、両環境で同じリポジトリを参照するように設定すれば、開発機で作成した成果物を本番機へスムーズに移行することができます。

バージョン管理を利用した開発サイクル

- ① 開発機でフロー、テンプレートファイルなどの作成・修正
- ② 開発機で検証の終わったフロー、テンプレートをリポジトリにコミット
- ③ 本番機でリポジトリから検証の終わったリビジョンの作成物を取得
- ④ 開発機では、次版に向けた開発や、保守を実施 → ②

さらに、開発機側のフローサービス管理コンソールのアカウント設定で「プロジェクト更新時にエクスポートファイルを作成する」をオンにし、そのエクスポートファイルもバージョン管理に含めておく事で実行設定の移行も簡便化できます。

具体的には開発機側でのリポジトリへのコミットが終わった後に本番機側で `flow-ctrl` にログインして以下のコマンドを実行します。

```
#リポジトリから最新版を取得する
```

```
svn update
```

```
#プロジェクトキャッシュのクリア
```

```
clear project-pool
```

```
#実行設定のインポート。「-r」を指定することで実行設定の総入れ替え
```

```
import triggers.xml -r
```

本番機では「プロジェクト更新時にエクスポートファイルを作成する」のチェックは外しておきます。

スケジュールが重複するなどの理由で開発機と本番機の実行設定が同期していない場合はこの方法はそのまま使用はできませんが、`regist` コマンドを使用したスクリプトと併用することである程度の自動化ができる場合があります。

## 付録.2.他の Subversion クライアントとの併用

ASTERIA サーバーが Windows マシン上で動作しており、そのマシンに TortoiseSVN がインストールされていると、ASTERIA ユーザーのホームディレクトリ以下にあるファイルをエクスプローラーで見た場合に TortoiseSVN の状態管理アイコンが表示されます。これは、バージョン管理設定を行ったホームディレクトリに Subversion の状態管理用の隠しフォルダー「.svn」が作成され、TortoiseSVN がそれを認識するためです。

このときにエクスプローラーの TortoiseSVN のメニューからバージョン管理の操作を実行するとほとんどの場合は問題なく実行できます。しかし、これは基本的にはやってはいけません。このときに TortoiseSVN が「.svn」フォルダー内のファイルを書きかえることがあり、不整合が発生する場合があります。

異なる Subversion クライアントでも「.svn」内のファイル仕様は基本的に共通ですが、バージョンが異なる場合などに不整合が発生して、一方のクライアントからはまったく読み書きできなくなることがあります。他の Subversion クライアントを併用する場合は ASTERIA ユーザーのホームディレクトリを直接参照せず、別のフォルダーに作業コピーをチェックアウトするようにしてください。

なお、万が一「.svn」フォルダーが壊れてしまった場合は以下の手順で復旧させることができます。

1. ユーザーのホームディレクトリを適当な名前でもリネーム
2. 同じ名前で空のユーザーホームディレクトリを作成する
3. フローサービス管理コンソールのバージョン管理設定で「変更」ボタンをクリック
4. リネームした元の出フォルダーからバージョン管理外のファイルをコピー

バージョン管理設定で「.svn」フォルダーがない状態から既にファイルのあるリポジトリを指定した場合、チェックアウトが実行されるのでこうした復旧が可能となっています。

## 付録.3.ブランチについて

ASTERIA の Subversion クライアントにはブランチ操作はありません。

ただし、ブランチ自体はサーバー側の機能なので別の Subversion クライアントを使用してブランチを作成することはできます。

作成したブランチに対してバージョン管理設定を行うことに問題はありますが、競合のマージの場合と同じく xfp などのフロー関連ファイルのマージは保証されません。

本書に記載された情報は、2011年1月現在の情報です。内容は、予告無しに変更することがあります。Infoteria、インフォテリア、ASTERIAは、インフォテリア株式会社の登録商標です。その他、各会社名、各組織名、各製品名は、各社、各組織の商標又は登録商標です。

©2011 Infoteria Corporation All Rights Reserved.