アプリケーションにおける ActiveWorkflow プロセスの使用

Release 12

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 or (916) 928-6400
FAX (916) 928-6401

Name: Using ActiveWorkflow Processes in Applications

Release: Unify NXJ 12

Last Revision: January 5, 2009 4:32 pm

アプリケーションにおける ActiveWorkflow プロセスの使用

NXJ ActiveWorkflow は、ビジネスプロセスの自動化と管理を行うことを目的とした機能です。この自動化の部分は、ActiveWorkflow プロセスを NXJ フォームアプリケーションと緊密に統合することを前提としています。

このドキュメントは、NXJ フォームアプリケーションが NXJ ActiveWorkflow プロセスと連携する方法を説明しています。

- ActiveWorkfow アクティビティと NXJ フォームの関連付け
- ActiveWorkflow オペランドとしてのエントリポイントフォーム
- エントリポイントフォームに渡される暗示的なパラメータ
- NXJ ソースから ActiveWorkflow エンジンの EJB メソッドの起動
- NXJ フォームアプリケーションから AcriveWorkflow プロセスの開始
- ActiveWorkflow プロセスにステータスとオペランドの追加
- NXJ ActiveWorkflow Application のセッションメカニズム

ActiveWorkflow アクティビティと NXJ フォームの関連付け

アクティビティフォームは、ActiveWorkflow エンジン上のアクティビティからそれに関するデータを受け取るフォームです。アクティビティフォームは、通常のフォームと同様ですが、エントリポイントやズームフォームを使用することはできません。アクティビティフォームは、各アクティビティのアクティビティワークページとして利用されます。

NXJ アプリケーションは、実行中、1 つのアクティビティフォームのみを持つことができます。つまり、アクティビティフォームは、他のアクティビティフォームに次フォームで移ることができません。アクティビティフォーム実行中は、NXJ フォームあるいは "to do" リストを表示している他のウィンドウは使用できません。アクティビティフォームの構築の詳細については、『ActiveWorkflow チュートリアル』を参照してください。

アクティビティフォームを定義するには、以下の手順で行います。

タスク 1: プロジェクトに ActiveWorkflow プロセス定義を追加する

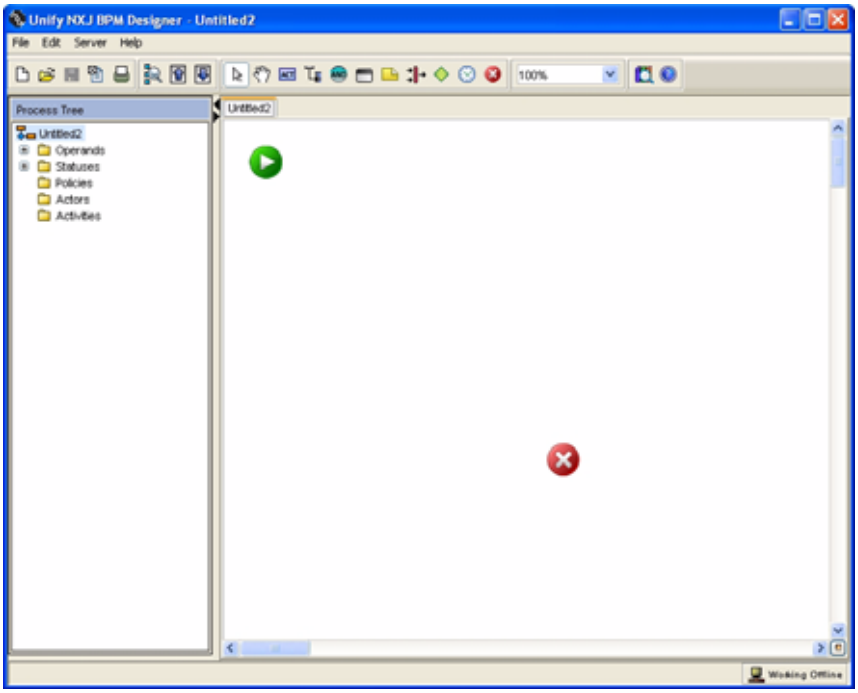
アクティビティフォームを定義する前に、NXJ フォームを適用する ActiveWorkflow プロセスを作成しなければなりません。プロジェクトに、新規の ActiveWorkflow プロセスを作成したり、既存の ActiveWorkflow プロセスを追加することができます。

Unify NXJ 開発ツールで、ActiveWorkflow プロセスは、Process Definitions と呼ばれるプロジェクトフォルダに格納されます。ActiveWorkflow プロセスは、ActiveWorkflow デザイナで作成されます。

新規 ActiveWorkflow プロセスの作成

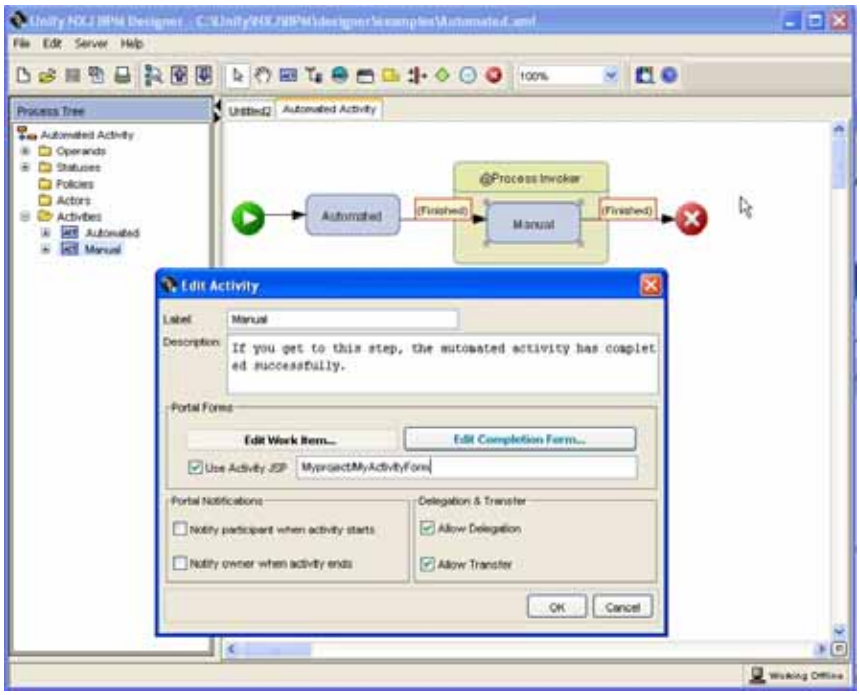
1. Unify NXJ 開発ツールで、プロセス定義を含む必要のあるプロジェクトを開きます。
2. プロジェクトパネルで、Process Definitions フォルダを右クリックして、**新規作成** > **Process** を選択します。

ActiveWorkflow デザイナが起動され、新規の空のプロセスが開きます。



3. 通常、ActiveWorkflow ユーザガイドに説明されているように ActiveWorkflow プロセスを定義します。このプロセスのアクティビティに関連付けられるアクティビティフォームの名前を指定しなければなりません。

   アクティビティフォームの名前を指定するには、ActiveWorkflow デザイナの各アクティビティをダブルクリックして アクティビティの編集ダイアログ にアクセスします。Activity JSP の使用フィールドにアクティビティに関連付けられるアクティビティフォームの名前を入力します。



4. **ファイル** > **保存** を選択して、プロセス定義を保存します。

   プロセスは、デフォルト名で保存されます。

5. Unify NXJ 開発ツールで、プロセスの名前を変更する場合、プロジェクトタブのそのプロセスを右クリックして **名前の変更** を選択します。

6. ActiveWorkflow デザイナ上で、**プロセステンプレートのアップロード**ボタンををクリックして、ActiveWorkflow エンジンにプロセステンプレートをアップロードします。

7. 必要に応じて、**ファイル** > **閉じる** を選択して、ActiveWorkflow デザイナを閉じます。

既存の ActiveWorkflow プロセスの追加

1. Unify NXJ 開発ツールで、ActiveWorkflow アクティビティフォームを含む必要のあるプロジェクトを開きます。

2. プロセスを記述している XML ファイルを指定します。

   デフォルトで、ActiveWorkflow デザイナは、次のパスにプロセスを作成します。

   ```
   <UNIFY_HOME>\BPM\designer\examples.<processName>.xml
   ```

3. XML ファイルをプロジェクトのプロセスフォルダにコピーします。デフォルトでは、次のパスです。

   ```
   <UNIFY_WORK>\projects\<ProjectName>\sources\Process_Definitions
   \<myProcess>.xml
   ```

4. Unify NXJ 開発ツールのプロジェクトタブで、Process Definitions フォルダを右クリックしてファイルの追加を選択します。
5. ステップ 2 で追加した <myProcess>.xml ファイルをブラウズして選択します。

   プロセスが、プロジェクトを利用できるようになります。
6. このプロセスのアクティビティに関連付けるアクティビティフォームをマップします。

   詳細については、5 ページのステップ 3 を参照してください。

タスク 2: アクティビティフォームの作成

アクティビティフォームを作成するには、アクティビティウィザードを使用します。アクティビティフォームを作成するために、ウィザードを使用する 2 つの方法があります。デフォルトでは、アクティビティのオペランド情報からフィールドを自動生成したフォーム作成があります。あるいは、空のアクティビティフォームを作成して、どのフィールドを生成するかを選択します。

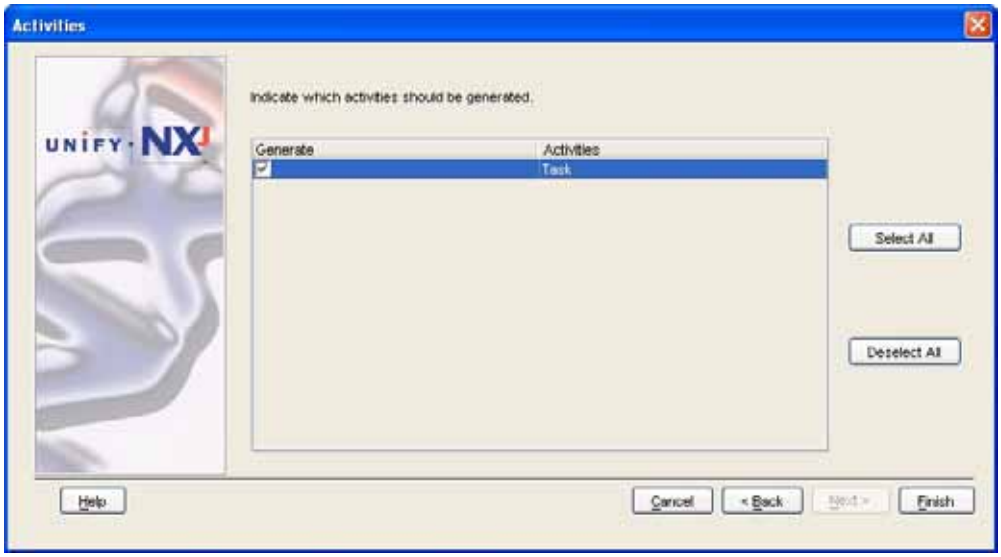
デフォルトによるアクティビティフォームの作成

1. **ファイル** > **デフォルトフォームの作成**を選択します。

   デフォルトフォーム作成ウィザードが表示されます。
2. **プロセス**ボタンをクリックします。
3. プロセスのドロップダウンリスト内で、これから作成するフォームを関連付けたいプロセスを選択します。

   タスク 1 で定義されたプロセスになります。
4. **次へ**ボタンをクリックします。

   プロセスで定義された各アクティビティが一覧されます。



5. **アクティビティパネルで、フォームを生成する各** Activity **の作成チェックボックスをチェックします。**
6. **完了ボタンをクリックします。**

   デフォルトフォームの作成ウィザードは、指定した各アクティビティのフォームとプロセスのオペランドのコントロールをフォーム上に作成します。

   次のテーブルは、オペランドのタイプによって生成されるコントロールのタイプを示しています。

ActiveWorkflow オペランド タイプ	生成された コントロール	生成された データ タイプ	生成されたプロパティ
Checkbox	Checkbox	String	
Currency	Text field	Amount	Display justify: right
Database Object	Text field	String	
Date	Text field	Date	
Email	Text field	String	
Entry Point Form	Text field	String	エントリポイントの URL
File	File chooser	Binary	
Long Text	Text area	Text	
Number	Text field	Amount	Display justify: right
Pick list	Dropdown list	String	**リストオプションは、オペランド定義から実行時に受け取る**
Short Text	Text field	String	
Title	Text field	String	
URL	Link	URL Link	
User	Text field	String	

カスタムオペランドは、フィールドコントロールとしてサポートがされません。フィールドコントロールは、フォームスクリプトからオペランドとしてアクセスできます。Javadoc の com.unify.nxj.bpm.engine.requests と『ActiveWorkflow プログラミングガイド』を参照してください。

アクティビティウィザードは、キャンセルボタンとプロセスの各ステータスコードに対応したボタンも作成します。キャンセルボタンは、CANCEL ACTIVITY コマンドが設定されており、ステータスコードに対応したボタンには、COMPLETE ACTIVITYnnn コマンドが設定されています。（この nnn はステータスコードを表わします。）

各フィールドの Required と Updateable プロパティは、オペランドの required と read-write プロパティに従って生成されます。

以下のアクティビティフォームに関連するプロパティは、アクティビティウィザードによって自動で設定されます。アクティビティフォームのメンテナンス時には、これらのプロパティを変更してアクティビティフォームを再定義することができます。

プロセス：フォームが割り当てられたプロセス。これは、プロパティの Workflow グループの Form プロパティです。

アクティビティ：フォームが割り当てられたアクティビティ。これは、プロパティの Workflow グループの Form プロパティです。

オペランド：フィールドがバインドされたアクティビティのオペランド。これは、プロパティの Workflow グループの Control-level プロパティです。

オペランド設定によるアクティビティフォームの作成

1. Unify NXJ 開発ルールのブラウザパネルのプロジェクトタブで、Classes フォルダを右クリックして、**新規作成** > **Activity form** を選択します。

   空のフォームが作成されます。

2. ツールバーから、**Activity ウィザード**ボタンをクリックします。

   ActiveWorkflow オプションが表示されます。

3. **プロセス**フィールドで、フォームに関連付けるアクティビティを含むプロセスの名前を選択します。

   プロセス定義に関する詳細については、タスク 1 を参照してください。

   使用するプロセスがそこにない場合、Activity ウィザードをキャンセルします。トラブルを解決するために、タスク 1 を正しく行ったかを確認してください。

4. **Activity** フィールドで、アクティビティを選択します。

5. **次へ** ボタンをクリックして、デフォルトフィールドパネルを表示します。

   デフォルトで、すべてのオペランドが含むの列に一覧されています。

6. フィールドを生成したくないオペランドがあれば、除外の列に移動します。

7. **次へ**ボタンをクリックして、デフォルトボタンパネルを表示します。

8. フォームに含めないボタンがあれば、除外の列に移動します。

このパネルは、アクティビティのステータスコードの値を表示して、ユーザがそのステータスコードに対するボタンが必要かどうかを決めることができます。

9. **完了**ボタンをクリックします。

   7 ページのステップ 6 に説明されているように、ActiveWorkflow オプションがフォームを生成します。

タスク 3: 必要に応じてフォームを構築する

タスク 2 で作成されたフォームは、実行できる状態にあります。必要に応じて、レイアウト、プロパティ、フォームのソースプログラムを変更することができます。

アクティビティフォームは、マスタ / 詳細関係のマスタとして使用されます。アクティビティフォームが、マスタ / 詳細関係のマスタとして使用される場合、アクティビティのオペランドをマスタキーとして使用することができます。オペランドは、マスタキーの選択肢に表示されます。“顧客名” のようなオペランドがあり、顧客情報を表示したいような場合には、この機能が役立ちます。これをするためには、アクティビティフォームを作成して、フォームにデータビューを追加して、顧客名オペランドをマスタキーとしたマスタ / 詳細関係を設定します。

ソースプログラミングスクリプトから、フォーム上に表示されていないターゲットフィールドにアクセスするのと同じ方法で、フォーム上に表示されていないオペランドにアクセスすることができます。フォームに表示されていないターゲットオペランドは、フォームに表示されていないターゲットフィールドが暗示的に宣言されているのと同様に暗示的に宣言されています。

新規のフィールドコントロールは、アクティビティで定義された ActiveWorkflow オペランドに割当てることができます。ActiveWorkflow オペランドに割当てるフィールドコントロールで、Workflow グループの Operand プロパティを設定します。

アクティビティフォームは、ボタンまたはリンクに割当てることができるアクティビティに関連する一組のコマンドを持っています。

コマンド	動作
CANCEL_ACTIVITY	CANCEL_ACTIVITY は、現在のアクティビティをキャンセルします。ActiveWorkflow エンジンへの影響は、あたかもフォームがアクティビティに対してまったくアクションしなかったようになります。コマンドは、以下のように作用します。 • ON CANCEL ACTIVITY イベントセクションを起動します。イベントセクションが reject 操作を行ってから、残りのコマンドをスキップします。 • 前フォーム操作をして、to-do リストに戻ります。アクティビティフォームが to-do リストと同じウィンドウに表示されている場合、ウィンドウをリセットして to-do リストを表示します。アクティビティフォームが別のウィンドウに表示されている場合は、アクティビティフォームのウィンドウを閉じて、to-do リストのボタン、リンク等を利用可能にします。前フォーム処理では、ON PREVIOUS FORM イベントセクションの実行はしません。CANCEL_ACTIVITY コマンドと PREVIOUS_FORM コマンドは、実行されるイベントセクションの名前が違うだけでそれ以外は同じです。

コマンド	動作
COMPLETE_ ACTIVITY_*nnn*	各アクティビティフォームは、COMPLETE_ACTIVITY_nnn というパターンの名前のコマンドをサポートします。この nnn は、アクティビティに有効なステータスコードを表します。Complete Activity Form は、以下のような処理をします。 • BEFORE COMPLETE ACTIVITY イベントセクションを起動します。イベントセクションは reject 操作を行う場合、AFTER COMPLETE ACTIVITY イベントセクションを実行して、残りのコマンドをスキップします。 • BEFORE UPDATE OPERANDS イベントセクションを起動します。イベントセクションが reject 操作を行う場合、AFTER UPDATE OPERANDS と AFTER COMPLETE ACTIVITY イベントセクションを実行してから残りのコマンドをスキップします。 • ActiveWorkflow エンジンに変更されたオペランドを書き出します。 • AFTER UPDATE OPERANDS イベントセクションを起動します。 • ActiveWorkflow エンジンにステータスコードを書き出します。 • AFTER COMPLETE ACTIVITY イベントセクションを起動します。 • 前フォーム操作をして、to-do リストに戻ります。アクティビティフォームが to-do リストと同じウィンドウに表示されている場合、ウィンドウをリセットして to-do リストを表示します。アクティビティフォームが別のウィンドウに表示されている場合は、アクティビティフォームのウィンドウを閉じて、to-do リストのボタン、リンク等を利用可能にします。前フォーム処理では、ON PREVIOUS FORM イベントセクションの実行はしません。 コマンド名は、ステータスコードに基づいて生成されます。NXJSession.queueCompleteActivityCommand メソッドを使用して、ステータスコード名を指定します。また、NXJ は、NXJActivityForm.getCompleteActivityCommand メソッドを使用して、COMPLETE ACTIVITY nnn コマンドの NXJ コマンドインスタンスを入手します。

コマンド	動作
UPDATE_ OPERANDS	UPDATE_OPERANDS コマンドは、ActiveWorkflow エンジンに変更されたオペランドを書き出します。このコマンドは、以下のように動きます。 • BEFORE UPDATE OPERANDS イベントセクションを起動します。 • イベントセクションが reject 操作を行う場合、AFTER UPDATE OPERANDS イベントセクションを実行してから、残りのコマンドをスキップします。 • ActiveWorkflow エンジンに変更されたオペランドを書き出します。 • AFTER UPDATE OPERANDS イベントセクションを起動します。
CANCEL_ZOOM, EXIT, NEXT_FIELD, NEXT_FORM, PREVIOUS_FIELD, PREVIOUS_FORM, ZOOM	通常の NXJ フォームにおける動作と同じです。NXJ Forms Processing Javadoc で、NXJCommand に関する説明を参照してください。

アクティビティフォームは、スクリプトを書いている時に使用できるイベントセクションを持っています。

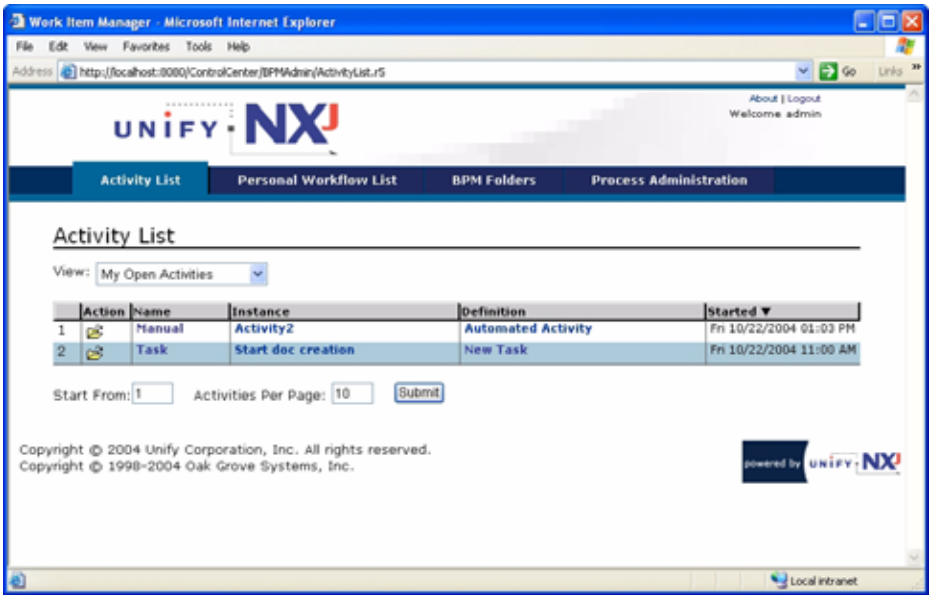
イベントセクション	動作
AFTER COMPLETE ACTIVITY	このイベントセクションは、COMPLETE_ACTIVITY_nnn コマンド実行中の前フォーム処理の直前に呼び出されます。ステータスコード nnn は、このセクションの引数として提供されます。
AFTER UPDATE OPERANDS	このイベントセクションは、COMPLETE_ACTIVITY_nnn と UPDATE_OPERANDS コマンドの一部として ActiveWorkflow エンジンにオペランドが書き込まれた後に呼び出されます。
BEFORE CANCEL ACTIVITY	このイベントセクションは、CANCEL_ACTIVITY コマンドの開始時に呼び出されます。ユーザが reject 操作を行う場合、コマンドの残りの部分はスキップされます。
BEFORE COMPLETE ACTIVITY	このイベントセクションは、COMPLETE_ACTIVITY_nnn コマンドの開始時に呼び出されます。ユーザが reject 操作を行う場合、コマンドの残りの部分はスキップされます。ステータスコード nnn はこのセクションの引数として提供されます。
BEFORE UPDATE OPERANDS	このイベントセクションは、COMPLETE_ACTIVITY_nnn と UPDATE_OPERANDS コマンドの一部としてオペランドが ActiveWorkflow エンジンに書き込まれる前に呼び出されます。ユーザが reject 操作を行う場合、コマンドの残りの部分はスキップされます。
ON CANCEL ACTIVITY	このイベントセクションは、CANCEL_ACTIVITY コマンドの開始時に呼び出されます。ユーザが reject 操作を行う場合、コマンドの残りの部分はスキップされます。

イベントセクション	動作
BEFORE_APPLICATION BEFORE_FORM ON NEXT FORM AFTER FORM RETURN AFTER ZOOM ON PREVIOUS FORM ON EXIT AFTER APPLICATION	通常の NXJ フォームと同様の動作をします。NXJ Forms Processing Javadoc で、NXJCommalnd クラスの説明を参照してください。

タスク 4: アクティビティフォームの配備と利用

通常と同様に、フォームを配備します。

ActiveWorkflow ポータルの to-do リストからアクティビティフォームにアクセスすることができます。

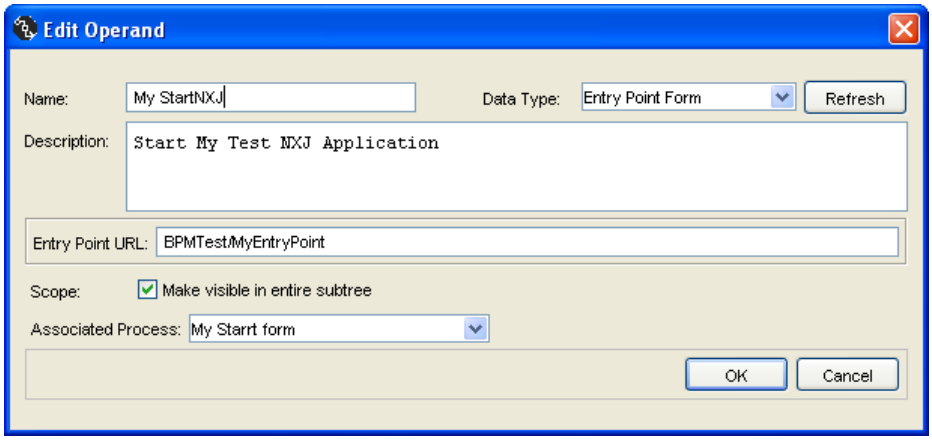


ActiveWorkflow オペランドとしてのエントリフォーム

NXJ フォームアプリケーションは、アプリケーションユーザの開始フォームとして使用されるエントリポイントフォームを定義する機能があります。このエントリポイントは、セキュリティ、変数、システム初期化パラメータの設定機能を提供します。この NXJ エントリポイントは、ActiveWorkflow アクティビティに対するオペランドとして動作します。ActiveWorkflow プロセス内から呼び出される NXJ フォームアプリケーションは、プロセス ID と認証トークンを識別するための HTTP リクエストパラメータを自動的に取

得します。プロセス ID は、プロセスの正しいインスタンスをユニークに識別するために使用されます。認証トークンは、NXJ フォームアプリケーションセキュリティによるユーザの識別をします。

NXJ フォームアプリケーションエントリポイントは、" エントリポイントフォーム " タイプとしてオペランドを定義したときに追加され、該当の ActiveWorkflow ポータルフォーム上に表示されるようになります。

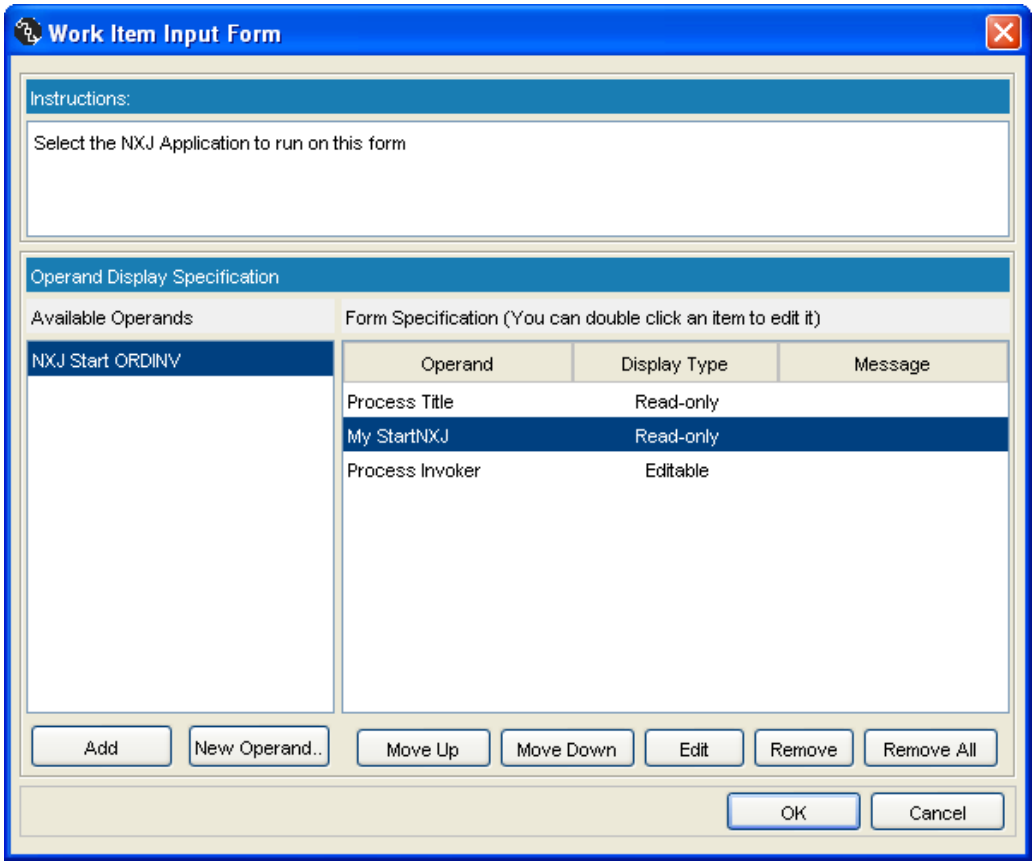


エントリポイントフォーム用のオペランドの編集

値は、必要とする NXJ エントリポイントへの URL として指定されます。URL は、完全で適切な URL として指定されます。

- パッケージ名 / エントリ名
- http://host:port/packages/package-name/entry-name

エントリポイントオペランドは、該当の ActiveWorkflow アクティビティに対する "Work Item Input Form" に追加することができます。



NXJ エントリフォームの Work Item Input Form

NXJ エントリフォームは、ポータルアクティビティフォーム内で利用可能なアプリケーションアイコンとして表示されます。このアイコンをクリックすると、NXJ アプリケーションが開いて、適切な HTTP リクエストパラメータが渡されます。

エントリポイントフォームに渡される暗示的パラメータ

NXJ フォームアプリケーションは、ActiveWorkflow プロセス ID と認証トークンを要求して、ActiveWorkflow プロセスにアクセスします。この情報は、bpmProcessID と bpmAuthToken として、startupProperties を使用して HTTP リクエスト内で NXJ アプリケーションに渡されます。

NXJ フォームアプリケーションは、ドキュメントに説明されている ActiveWorkflow java API のどれかを利用して、ActiveWorkflow プロセスと直接対話できます。以下の例は、ActiveWorkflow プロセスオペランドから呼び出されるエントリポイントフォームの NXJ

ソースの一部です。このフォームは、ActiveWorkflow プロセスとプロセスのいくつかの値を設定します。

```
/*------------------ NXJ sample code ------------------------*/
import com.unify.nxj.bpm.engine.processMediation.*;
import com.unify.nxj.bpm.engine.requests.*;
import java.util.*;
FORM MyEntryPoint
{
// Create a BPMProxy object
   BPMProxy bpm = (BPMProxy)session.createBPMProxy();
   String authToken = null;
   BPMObjectId processId = null;
   BPMObjectId nProcessId = null;
   BPMObjectId statusId = null;
   BPMObjectId parentProcessId = null;
   BEFORE FORM
   {
      Properties props = session.startupProperties;
      if ( props != null )
      {
         String pid = props.getProperty("bpmProcessID");
         if ( pid != null )
            {
               processId = new BPMObjectId(pid);
               // display the ProcessID value to a screen field
               BPMPID=pid;
            } // if  ( pid != null )
            try{
               authToken = props.getProperty("bpmAuthToken");
               if ( authToken != null )
               {
                  // Set the current BPMProxy Authorization token
to the one
                  // passed into the NXJ Application
                     bpm.setAuthToken(authToken);
                  // Display the value to a screen field
                     AUTHTOKEN=authToken;
                } // if ( authToken != null )
             }catch (Exception NoAuth){
                AUTHTOKEN = "No Authorization token passed";
         }
      } // if ( props != null )
   } // BEFORE FORM
/*------------------ End NXJ sample code --------------------*/
```

この例では、ソースが、アプリケーションの startup プロパティから ActiveWorkflow プロセス ID を取得します。processId が存在する場合、ActiveWorkflow エンジンにリファレンスを付けて、フォームに表示するフィールドを設定します。

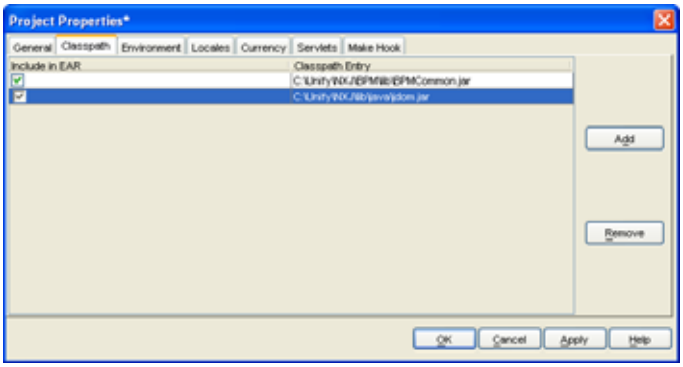
startup プロパティが ActiveWorkflow 承認トークンを含んでいる場合、これが取得され、セッションのために ActiveWorkflow エンジン内に承認トークンを設定することに使用されます。

以下の JAR ファイルが、プロジェクトのクラスパスに追加されていることに注意してください。

BPMCommon.jar
jdom.jar

JAR ファイルをプロジェクトのクラスパスに追加

1. **プロジェクト** > **プロパティ** > **クラスパス** を選択します。
2. **追加**ボタンをクリックして、新しいエントリを作成します。
3. <UNIFY_HOME>\NXJ\BPM\lib\BPMCommon.jar を表示して選択します。
4. **追加**ボタンをクリックして、新しいエントリを作成します。
5. <UNIFY_HOME>\NXJ\lib\java\jdom.jar をブラウズして選択します。
6. この結果、jar ファイルはアプリケーションパッケージに含まれて、"EAR に含める" チェックボックスをセットします。



7. **OK** ボタンをクリックして、クラスパスの変更を保存します。

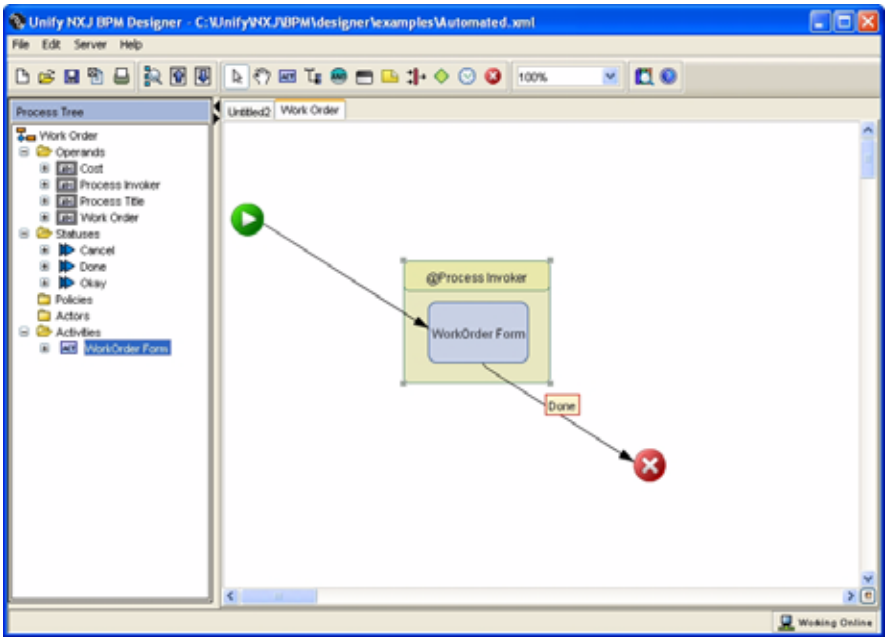
NXJ ソース内から ActiveWorkflow エンジン EJB メソッドを起動

ActiveWorkflow 機能は、NXJ フォームアプリケーション内から ActiveWorkflow プロセスにアクセスする API を持っています。この API は、NXJ リリースに含まれる Javadoc に記述されています。Unify NXJ 開発ツールメニューから **ヘルプ** > **Javadoc** を参照してください。

以下の 2 つの例は、NXJ フォームアプリケーションから ActiveWorkflow プロセスをスタートする方法とオペランドの更新と ActiveWorkflow ステータスの変更の方法を示しています。

NXJ フォームアプリケーションから ActiveWorkflow プロセスの開始

このサンプルは、Work **オーダと呼ばれる** ActiveWorkflow **プロセスを作成し、プロセスに必要な値を渡します。**



NXJ **フォームアプリケーションから** ActiveWorkflow **プロセスを開始するサンプル**

```
/*------------------ NXJ sample code ------------------------*/
import com.unify.nxj.bpm.engine.client.*;
import com.unify.nxj.bpm.engine.requests.*;
import com.unify.nxj.bpm.engine.*;
import com.unify.nxj.bpm.engine.processMediation.*;
import java.util.*;
import org.jdom.Element.*;
import com.unify.nxj.bpm.engine.client.BPMProxy;

FORM StartBPM
{
BPMProxy bpm = (BPMProxy)session.createBPMProxy();
BEFORE FORM
    {
// ActiveWorkflow Operand values
wo_no = "0012345";
cost=1234.56;
username = "alice";
    }
COMMAND start_BPMProcess
{
//setting up the authorization token required for interacting with
//the ActiveWorkflow Server
```

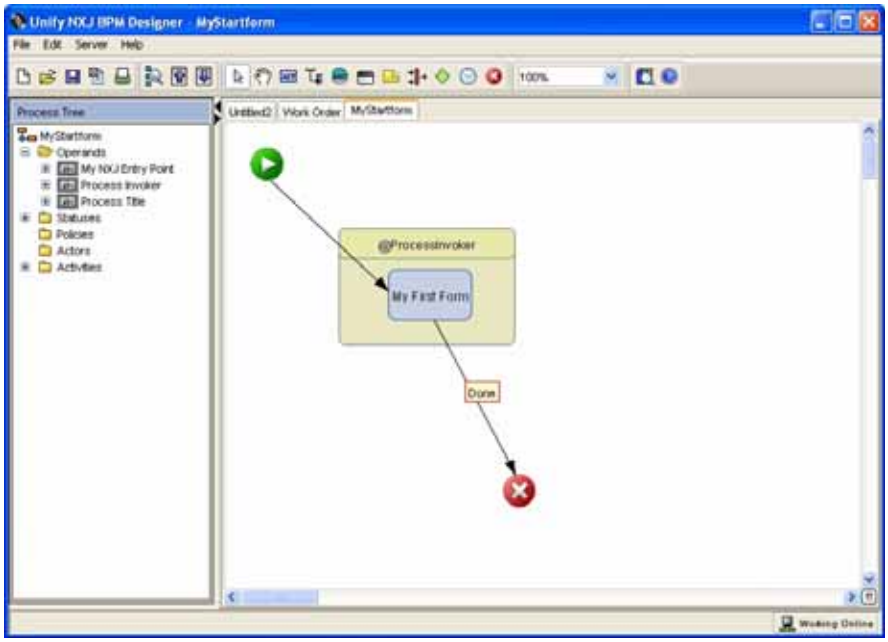
```
  try {
authToken = bpm.login( "admin", "admin");
}
catch (BPMProxyException BPMExcp) {
bpm = null;
authToken = "Did not authorize";
}
BPMObjectId processId =
bpm.cloneInstance(new LabelPath("def:/Work Order"));
Operand workOrder = (Operand)bpm.get(
             new LabelPath("operand://" +
             processId.getId() +
             "/Work Order"));
workOrder.setValue( wo_no.toString() );

Operand title = (Operand)bpm.get(
             new LabelPath("operand://" +
             processId.getId() +
             "/Process Title"));

title.setValue( "Work Order # " + wo_no );

Operand invoker = (Operand)bpm.get(
             new LabelPath("operand://" +
             processId.getId() +
             "/Process Invoker"));

invoker.setValue( username );

Operand costOper = (Operand)bpm.get(
             new LabelPath("operand://" +
             processId.getId() +
             "/Cost"));

costOper.setValue( cost.toString() );
HashSet objects = new HashSet();
objects.add(workOrder);
objects.add(title);
objects.add(invoker);
objects.add(costOper);
bpm.setObjects(objects);
bpm.startProcess( processId );
session.displayToMessageBox("Work Flow Initiated");
} // COMMAND start_BPMProcess
FIELD username
{

}
FIELD wo_no
{

}
FIELD cost
{

}
}
/*------------------ End NXJ sample code --------------------*/
```

前の例と同様に、次の JAR ファイルが、プロジェクトのクラスパスに追加されます。

BPMCommon.jar
jdom.jar

ActiveWorkflow プロセス内でステータスの追加とオペランドの更新

ステータスの追加は、プロセスの状態の変更を意味します。



ActiveWorkflow プロセスのサンプル

```
/*------------------ NXJ sample code ------------------------*/
import com.unify.nxj.bpm.engine.processMediation.Process;
import com.unify.nxj.bpm.engine.client.*;
import com.unify.nxj.bpm.engine.*;
import com.unify.nxj.bpm.engine.processMediation.*;
import com.unify.nxj.bpm.engine.requests.*;
import java.util.*;

FORM MyEntryPoint
{
// Create a BPMProxy object
    BPMProxy bpm = (BPMProxy)session.createBPMProxy();
    String authToken = null;
    BPMObjectId processId = null;
    BPMObjectId nProcessId = null;
    BPMObjectId statusId = null;
```

```
    BPMObjectId parentProcessId = null;

  BEFORE FORM
  {
    Properties props = session.startupProperties;
    if ( props != null )
    {
        String pid = props.getProperty("bpmProcessID");
        if ( pid != null )
        {
            processId = new BPMObjectId(pid);
            // display the ProcessID value to a screen field
            BPMPID=pid;
        } // if  ( pid != null )
        try{
            authToken = props.getProperty("bpmAuthToken");
            if ( authToken != null )
            {
            // Set the current BPMProxy Authorization token to the one
 // passed into the NXJ Application
                bpm.setAuthToken(authToken);
            // Display the value to a screen field
                AUTHTOKEN=authToken;
            } // if ( authToken != null )
        }catch (Exception NoAuth){
            AUTHTOKEN = "No Authorization token passed";
        }

    } // if ( props != null )
  } // BEFORE FORM

COMMAND addStatusDone
 {
    // Set up the query criteria
    QueryCriteria qc = new QueryCriteria();
    // only look for root processes
    qc.setMustBeRoot(true);
    // get the entire process tree
    qc.setDepth(-1);
    // get the operands as well
    qc.setIncludeOperands(true);
    // get the statuese
    qc.setIncludeStatuses(true);
    // create a new BPMObject to use to query for statuses
    BPMObjects bpmObject = new BPMObjects();
    try {
        bpmObject = bpm.query(qc,null);
    } catch(BPMProxyException rpe) {
            session.displayToMessageBox("Login failed: " + rpe);
    }
    try {
        parentProcessId = processId;
        Process demoProcess = (Process)bpm.get(processId);
        // look up the process tree for statuses and operands from the
 // parent
        while (demoProcess.getParent() != null) {
            parentProcessId = demoProcess.getParent();
            demoProcess = (Process)bpm.get(demoProcess.getParent());
            }
```

```
        // retrieving & Setting values of operand for the process
        Operand woCost = (Operand)bpm.get(new LabelPath("operand://" +
parentProcessId + "/Cost"));
        session.displayToMessageBox("Original Cost: " + woCost);
        woCost.setValue("5555.23");
        //update the cost operand in the process
        HashSet objects = new HashSet();
        objects.add(woCost);
        bpm.setObjects(objects);
    }catch (Exception excp) {System.out.println("Finished Error");}
        // Lookup all Status to process and find “Done”
    Set statuses = bpmObject.getStatuses(parentProcessId);
    if(statuses != null) {
        Iterator i = statuses.iterator();
        if(!i.hasNext()) {
            System.out.println("none");
        }
        //Search for the "Done" Status ID
        while(i.hasNext()) {
            Status statusIdx = (Status)i.next();
            String statusIdxDesc = statusIdx.toString();
            String statusIdxDesc1 = statusIdx.getId();
            if (statusIdxDesc.equals("Done")) {
                statusId = new BPMObjectId(statusIdxDesc1);
            }
        }
    }
    try  {
        bpm.addStatusToProcess(processId,statusId);
    } catch (Exception excp)
        {session.displayToMessageBox("Request Error Exception");}
}

    FIELD BPMPID
    {

    }

    FIELD AUTHTOKEN
    {

    }
}
/*------------------- End NXJ sample code ---------------------*/
```

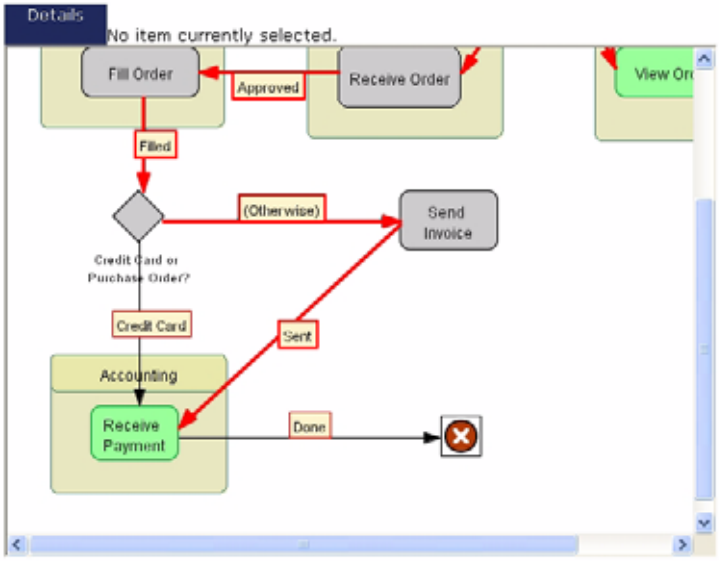
前の例と同様に、次の JAR **ファイルがプロジェクトのクラスパスに追加されます。**

BPMCommon.jar
jdom.jar

NXJ フォームからプロセスインスタンスの図にアクセス

ActiveWorkflow **エンジンは、プロセスインスタンスの最新の状態を表わすダイアグラムを表示するプロセスインスタンス詳細ビューを提供します。このダイアグラムは、最新**

のアクティビティ、あるいは、明るい緑で示されたアクティビティでプロセスマップを表示します。



ダイアグラムは、ActiveWorkflow 管理者から簡単にアクセスできる一方で、NXJ プログラミング言語文を使ってプロセスインスタンスダイアグラムを表示するコマンドを定義することができます。

この構文は、以下の通りです。

```
/*---------------------- NXJ sample code ----------------------*/
import com.unify.nxj.bpm.engine.*;
import java.util.*;

ACTIVITY FORM MY_FORM
{
// Create a BPMProxy object
BPMProxy bpm = (BPMProxy)session.createBPMProxy();
String authToken = null;
String pid = null;
BPMObjectId processId = null;

BEFORE FORM
{
Properties props = session.startupProperties;
if ( props != null )
{
pid = props.getProperty(“bpmProcessID”);
if ( pid != null )
{
processId = new BPMObjectId(pid);
// display the ProcessID value to a screen field
BPMPID=pid;
} // if ( pid != null )
try{
```

```
authToken = props.getProperty(“bpmAuthToken”);
if ( authToken != null )
{
// Set the current BPMProxy Authorization token to the one
// passed into the NXJ Application
bpm.setAuthToken(authToken);
// Display the value to a screen field
AUTHTOKEN=authToken;
} // if ( authToken != null )
}
catch (Exception NoAuth){
AUTHTOKEN = “No Authorization token passed”;
}

} // if ( props != null )
} // BEFORE FORM

COMMAND showMap
{
// pid and authToken is set by BEFORE FORM
com.unify.pub.NameValuePair[] nvp= new com.unify.pub.NameValuePair[2];
nvp[0] = new com.unify.pub.NameValuePair(“id”,pid);
nvp[1] = new com.unify.pub.NameValuePair(“bpmAuthToken”, authToken);
session.openURL(“/ControlCenter/BPMAdmin/ViewImageMap”,nvp,
”newWindow”);
};


FIELD AUTHTOKEN
{
}
FIELD BPMPID
{
}
/*------------------ End NXJ sample code ---------------------*/
```

showMap コマンドは、ActiveWorkflow アクティビティフォームのボタンや、インラインフレームに関連付けることができます。

前の例と同様に、次の JAR ファイルがプロジェクトのクラスパスに追加されます。

BPMCommon.jar
jdom.jar

NXJ ActiveWorkflow アプリケーションのセッションメカニズム

NXJ12 で、すべての NXJ ActiveWorkflow アプリケーションが BPM Admin コンソールから開始されない場合、BPMProxy を使用してログインする必要があります。
bpmAuthToken という新しい NXJSession プロパティは、auth トークンが BPMProxy オブジェクトに設定されることを許可して、BPMProxy オブジェクトに暗黙の内に設定さ

れ、また、bpmAuthToken の初期設定後に作成される BPMProxy オブジェクトにも暗黙の内に設定され、利用可能となります。

Example:

```
COMMAND logincmd
{
----if(session.login(user,pwd))
----{
-------BPMProxy bpm = (BPMProxy)session.createBPMProxy();
-------session.bpmAuthToken = bpm.login(user,pwd);
-------bpm.setAuthToken(session.bpmAuthToken);
----}
}
```

注: 完全な実施例を参照するには、../Unify/NXJWork/Projects/examples/ordmgmt にある NXJ プロジェクト “ordmgmt” を開いてください。