管理者ガイド

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

Name: Administrator's Guide

Release: Unify NXJ 12

Last Revision: January 23, 2009 2:59 pm

目次

1 はじめに

Unify NXJ は、NXJ アプリケーションを構築・配備・管理をするためのプラットフォームです。

*NXJ アプリケーション*は、ビジネスプロセスを自動化して、それらのプロセスにおけるデータへの安全なアクセスが保証された J2EE 準拠の Web アプリケーションです。例えば、NXJ アプリケーションは経費申請の承認プロセスに使用することができます。そのアプリケーションでは、従業員が申請書を提出してその進捗を確認することができたり、給与支払い担当者が従業員の提出した申請書を処理して、支払い処理に自動的に送ることができたりします。このように自動化されたプロセスは、申請書が正しく取り扱われることを確実にすることができます。また、セキュリティに関しては、従業員は申請の追加・確認に限定される一方で給与支払い担当者は追加・削除・更新・確認ができるといった形で保証することができます。

各 NXJ アプリケーションは ZIP ファイルでパッケージされ、J2EE アプリケーションサーバに配備されます。サポートされるアプリケーションサーバについては、『Unify NXJ がサポートする構成』に一覧されています。

1.1 Unify NXJ の環境

Unify NXJ は、開発環境と実行環境で動作します。NXJ アプリケーションは開発環境で作成されて、パッケージされます。これらのパッケージは、ユーザがアプリケーションにアクセスする実行環境において、アプリケーションサーバに配備されます。

管理者と配備担当者は実行環境で作業し、以下を使用します。

- NXJ インタラクションサーバ
  アプリケーションサーバと連携して NXJ アプリケーションの実行を調整する。

- 管理システム
  NXJ テクノロジに基づいてインストールされたいろいろなサーバサイド管理コンソールがあります。

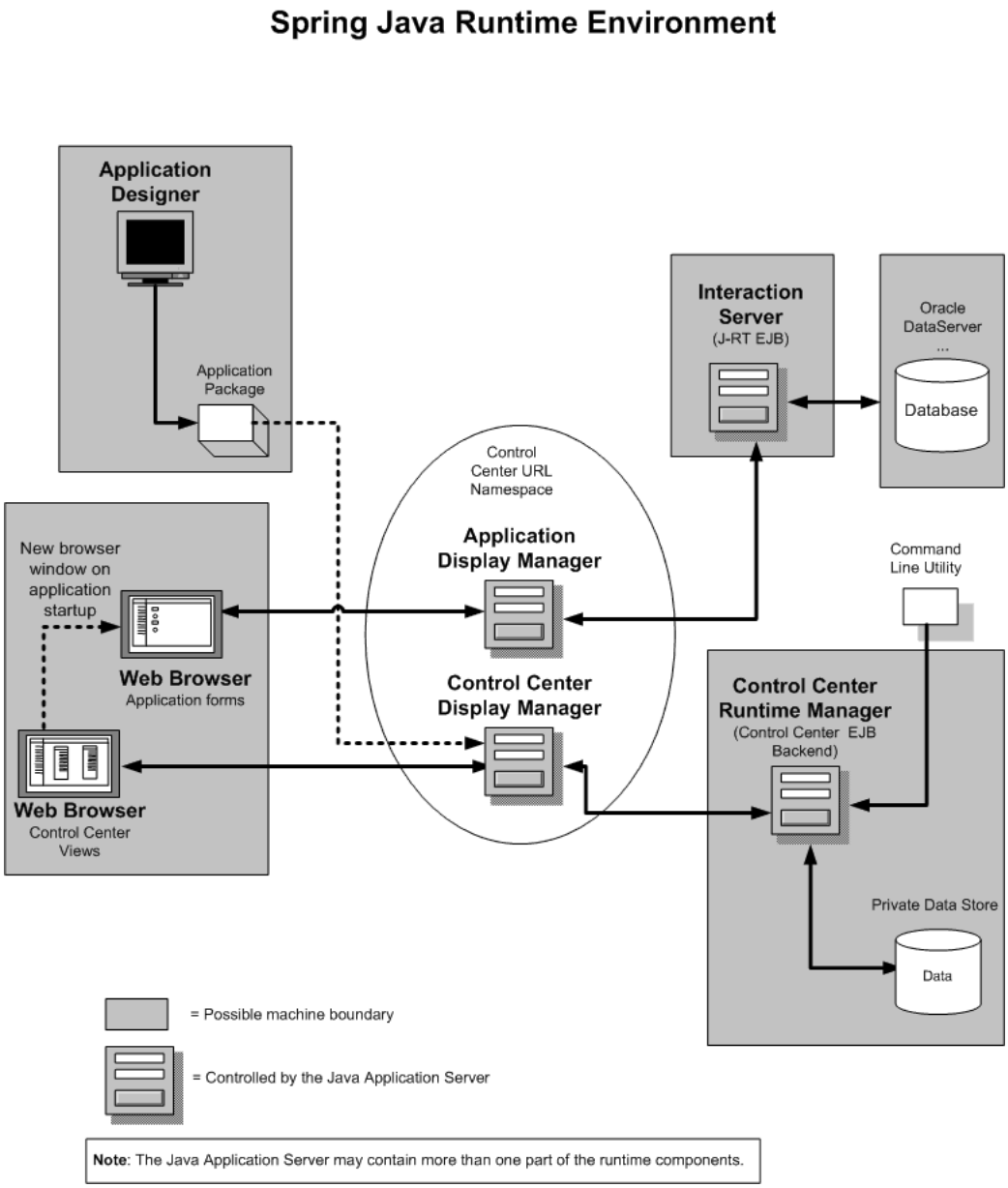
  Systinet Server Admin Console:
  ホストされた Web サービスを管理するために使用します。これは、http: //<host>: <por>/systinet/server/admin/console でアクセスすることができます。

  Business Process Admin Console:
  配備されたビジネスプロセスワークフローを管理するために使用します。これは、http://<host>:<port>/BPMAdmin でアクセスすることができます。

  Report Server:
  配備された NXJ レポートを管理するために使用します。これは、http: //<host>: <port>/jreport でアクセスすることができます。

エンドユーザは、ブラウザのアドレスバーに NXJ アプリケーションの URL を入力することにより、実行環境の NXJ アプリケーションにアクセスします。

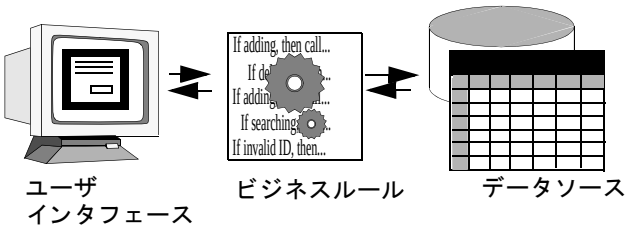
以下は、Unify NXJ でこれらのコンポーネントと他のコンポーネントの関係を示します。



このガイドは、これらのコンポーネントを使用する方法に関する情報を提供します。ガイドは、管理者や配備担当者、そして NXJ アプリケーションを配備して管理するために Unify NXJ を使用するその他のユーザを対象としています。

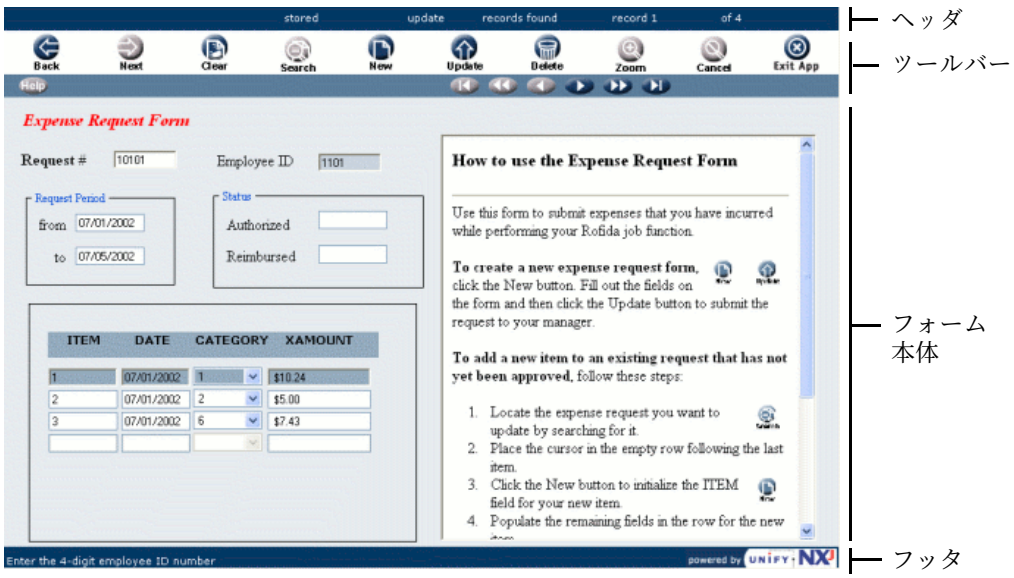
1.2 NXJ アプリケーションはどのように動作するか？

NXJ アプリケーションは、ユーザインタフェース、データソースへのリンク、ユーザインタフェースとデータソース（以下の図）間の相互作用を定義するビジネスルールから成ります。



1.2.1 ユーザインタフェース

ユーザインタフェースは、主に Java Server Pages（JSP）ページに表示される *フォーム*から成ります。ユーザは、Web ブラウザでフォームの URL を入力してフォームにアクセスします。各フォームは、コンポーネントと機能をドラッグ＆ドロップで配置できるグラフィカルなツールのアプリケーションデザイナで作成されます。各フォームは、一般的にデータソースからの *ターゲットテーブル*に関連付けられています。以下は、一般的なフォームの例です。



ヘッダやフッタはアプリケーションのステータスに関する情報を提供します。ツールバーは、NXJ アプリケーションをナビゲートしたり、データをナビゲートしたり、データベース操作を実行するコマンドボタンをユーザに提供します。

ユーザがフォームを操作したとき、NXJ インタラクションサーバはユーザのアクションをデータベースに発行される SQL 文に変換します。どんな結果でも、フォームに返され、適切なフィールドに表示されます。例えば、ユーザは検索条件を入力し、ツールバーにある検索ボタンをクリックすることでターゲットテーブルを検索することができます。NXJ インタラクションサーバは、この検索コマンドをフォームにレコードを返す SQL SELECT 文に変換します。

1.2.2 データソース

NXJ アプリケーションで使用される *データソース*は、一般的に Java DataBase Connectivity（JDBC）を通して接続されるデータベースです。開発者は、アプリケーションのプロパティの一部として NXJ アプリケーションデータソースを確立します。管理者と配備担当者は、配備で使用するデータソースを選択します。管理者と配備担当者は、実行時にアプリケーションサーバにおいて、データソース定義を設定する必要があるかもしれません。データソース定義についての詳細は、15 ページの「データソース定義」を参照してください。サポートされるデータベースベンダーの一覧は、『Unify NXJ がサポートする構成』にあります。

1.2.3 ビジネスルール

*ビジネスルール*は利息を計算したり、請求額の合計を計算したりといったカスタマイズされたロジックを使用して特定の作業を実行します。開発者は、組織のビジネスプロセスに合うように NXJ アプリケーションを作成するためにビジネスルールを使用します。例えば、NXJ アプリケーションは、新しい経費申請書のレコードがデータベースに追加されると、自動的に申請書番号を生成するビジネスルールを含むことができます。あるいは、従業員が新しい経費申請書を提出する際に、従業員 ID を確認するためにビジネスルールを使用できます。

1.2.4 配備担当者と管理者

NXJ *配備担当者*は、NXJ アプリケーション、スタンドアローン ActiveWorkflow プロセスとレポートを配備する NXJ ユーザの特別なタイプです。

NXJ *管理者*は、NXJ アプリケーション、スタンドアローン ActiveWorkflow プロセスとスタンドアローンレポートを配備、再配備、または削除する NXJ ユーザの特別なタイプです。

1.2.5 リソース

*リソース*とは、アプリケーションフォーム、静的コンテント、動的コンテント等、NXJ アプリケーションを通じてユーザがアクセスできる物です。

静的コンテントには、PDF、HTML ページ、イメージが含まれます。静的コンテント ファイルは、アプリケーションパッケージに含めるか、または管理者によって追加されます。一部の静的コンテントは、PDF ファイルのために Adobe Acrobat Reader のような別のビューワを必要とします。

動的コンテントは、レポート等のように NXJ アプリケーションに組み込まれるロジックの結果、生成されるか変更されます。

1.2.6 セキュリティ

NXJ アプリケーションは、特定のフォーム、データとその他の機能へアクセスするユーザを制限することを保証します。

NXJ アプリケーションのセキュリティは、NXJ アプリケーションの機能に割り当てられるロールとユーザのセキュリティプロバイダーで定義されるユーザのグループ間の関係に基づきます。セキュリティについての詳細は、『Securing an NXJ Application』を参照してください。

2 NXJ リポジトリの構成

NXJ リポジトリは、カレントビジネスプロセスや完了プロセスの履歴のような情報を格納します。デフォルトでは、NXJ リポジトリは Unify NXJ にバンドルされている SQL Base リレーショナルデータベースにデータリポジトリを作成するように構成されています。

NXJ アプリケーションをテストする際には、実行環境をエミュレートできるよう、開発環境にデータリポジトリを構成してください。つまり、実行環境でデータリポジトリを Oracle 上に構成するのであれば、開発環境でも Oracle データベースを使用して下さい。

この章の残りのセクションでは、NXJ データリポジトリを他のデータベース上に構成する方法について説明します。以下のアプリケーションサーバ / データベースの組み合わせがサポートされています。

アプリケーションサーバ	データベース	参照
JBoss	IBM DB2	8 ページの「IBM DB2 データベース」
	Informix	9 ページの「Informix データベース」
	MS SQL Server	9 ページの「MS SQL Server データベース」
	Oracle	9 ページの「Oracle データベース」

アプリケーション サーバ	データベース	参照
WebSphere	IBM DB2 Informix Oracle	11 ページの「WebSphere アプリケーションサーバ」
WebLogic	Oracle MS SQL Server	11 ページの「WebLogic アプリケーションサーバ」
Oracle 10g & OC4J	Oracle	14 ページの「Oracle アプリケーションサーバ（OC4J or 10g）」

2.1 IBM DB2 データベース

DB2 データベースをリポジトリとして使用するには、以下の手順で NXJ リポジトリを構成する必要があります。

注： DB2 データベースを作成する手順は、“Type-2” CLI-based driver を使用することを前提としています。これには DB2 クライアント製品をインストールし、実際のデータベースへの “alias” を作成する必要があります。この alias はローカルアクセスする際のデータベース名として使用されます。NXJ からデータベースにアクセスするには、“Type-4” ドライバを使用します（データベース alias は使用しません）。

また、32KB ページサイズのテーブル領域が利用可能でなければなりません。これは、リポジトリデータベース中のあるテーブルの行サイズで非常に大きいものがあり、その行がデータベースページに収まる必要があるからです。User 領域および System 一時領域を作成して下さい。

1. DB2 “db2sql92” ユーティリティと repository-db2.sql スクリプト (<UNIFY_HOME>/lib/repository) を実行し**リポジトリテーブルを作成**します。

   ```
   db2sql92 -d ALIAS -a user/password < repository-db2.sql
   ```

   ALIAS はデータベース alias 名、user/password はテーブル作成のために必要なユーザ認証です。

   さらに、CLI-based ドライバを使用するには、ドライバの Jar ファイルをアプリケーションサーバのクラスパスに追加し、DB2 “bin” ディレクトリをアプリケーションサーバの PATH に追加する必要があります。

2.2 Informix データベース

Informix データベースを NXJ データリポジトリとして使用するには、以下の手順で構成する必要があります。

1. NXJ リポジトリデータを保持するテーブルをデータベースに作成します。

   dbaccess または同様のツールを用い、<UNIFY_HOME>/lib/repository ディレクトリにある “repository-informix.sql” スクリプトを実行します。このスクリプトは､同じディレクトリにある turbineUser-ifx.dat も使用しますので注意して下さい。

   dbaccess を実行するために必要な環境変数は以下のとおりです。

   - INFORMIXDIR　　Informix インストールディレクトリ ;
   - INFORMIXSERVER　Informix サーバインスタンス名 ;
   - PATH　　Informix “bin” ディレクトリをパスに含めます

   Informix 環境変数を設定後、dbaccess を実行します。

   ```
   dbaccess <repository> repository-informix.sql
   ```

   <repository> はデータベース名です。また “repository-informix.sql” および “turbineUser-ifx.dat” は現在のディレクトリに存在するものとします。

2.3 MS SQL Server データベース

MS SQL Server データベースを NXJ データリポジトリとして使用するには、以下の手順で構成する必要があります。

1. NXJ リポジトリデータを保持するテーブルをデータベースに作成します。

   osql または同様のツールを使用し、<UNIFY_HOME>/lib/repository にある "repository-mssql.sql" スクリプトを実行します。

   注 :　Microsoft SQL Server クライアントツールに含まれる "isql" ユーティリティはスクリプトを実行できません。CT-Library ベースの osql ユーティリティが代わりに使用できます。

2.4 Oracle データベース

Oracle データベースを NXJ データリポジトリとして使用するには、以下の手順で構成する必要があります。

1. NXJ リポジトリデータを保持するテーブルをデータベースに作成します。

   sqlplus または同様のツールを使用し、<UNIFY_HOME>/lib/repository ディレクトリにある "repository-oracle.sql" スクリプトを実行します。このスクリプトはリソース権限を持ったユーザで実行する必要があります。この権限は以下の方法で許可できます。

GRANT “RESOURCE” TO “<username>”;

テーブルスペースは 3MB 以上割り当てて下さい。

2.5 SQLBase

SQLBase でリポジトリテーブルを作成するには、Unify “sqltalk”ユーティリティを使用して SQLBase データベースに接続して、sql スクリプトを実行してください。: <UNIFY_HOME>/lib/ repository/repository-sqlbase.sql

2.6 JBoss アプリケーションサーバ

JBoss が、データソースを認識するために必要とする 2 つの構成項目があります。

1. 必要な jdbc データベースドライバ jar ファイルをインストールします。
2. データソース xml ファイルを設定します。

   データソース xml ファイルが、各サポートされたデータベースに対してどの様に構成されなければならないかを示すテンプレートファイルがあります。これらのテンプレートは、<UNIFY_HOME>/lib/repository ディレクトリにあります。

   これらは、<database>-NXJCCDS-ds.xml と名付けられます。ここで、<database> は db2、oracle、informix、mssql、または sqlbase のことです。

   例：
   db2 データソースを構成したい場合、.../jboss/server/default/deploy ディレクトリに db2-NXJCCDS-ds.xml をコピーします。新しくコピーされたファイルを編集して、以下の値を db2 に置き換えます。
   YOUR_HOST
   YOUR_PORT
   YOUR_DBNAME
   YOUR_USER NAME
   YOUR_PASSWORD

注： oracle データベースを使用する場合、以下の <attribute name=” Pad”>true</attribute> のように表示される”Pad" 属性をアンコメントする必要があります。この属性は、ディレクトリ ../jboss/server/default/conf に格納される jboss-service.xml ファイルにおいて構成されます。

2.7 WebSphere アプリケーションサーバ

WebSphere アプリケーションサーバを使用する場合、IBM DB2、Informix、MS SQL、Oracle データベースに NXJ データリポジトリを構成することができます。

使用する DBMS のバージョンは、本バージョンの Unify NXJ がサポートするバージョンでなければなりません。『Unify NXJ がサポートする構成』を参照して下さい。

以下の手順に従って、リポジトリとして他のデータベースを使用する NXJ リポジトリを構成してください。

2.7.1 リポジトリテーブルの作成

NXJ リポジトリデータを保持するリポジトリテーブルをデータベースに作成します。

8 ページの「IBM DB2 データベース」のステップ 1 を参照してください。

9 ページの「Informix データベース」のステップ 1 を参照してください。

9 ページの「Oracle データベース」のステップ 1 を参照してください。

2.8 WebLogic アプリケーションサーバ

WebLogic アプリケーションサーバに NXJ リポジトリを構成するには、アプリケーションサーバを構成する必要があります。

2.8.1 リポジトリテーブルの作成

NXJ リポジトリデータを保持するリポジトリテーブルをデータベースに作成します。

9 ページの「MS SQL Server データベース」のステップ 1 を参照してください。

9 ページの「Oracle データベース」のステップ 1 を参照してください。

2.8.2 WebLogic 開始スクリプトの更新

テキストエディタを使用して WebLogic サーバの開始スクリプトを編集します。通常、このスクリプトは startWebLogic.cmd（Unix/Linux では startWebLogic.sh）というファイルです。ファイルは、../bea/<user_projects>/Domains/<yourdomain> に格納されます。

1. （Oracle の場合）サーバの CLASSPATH に必要な jar ファイルを追加します。

   Windows:
   %UNIFY_HOME%/lib/jdbcDrivers/ojdbc14.jar

   Unix/Linux:
   ${UNIFY_HOME}/lib/jdbcDrivers/ojdbc14.jar

   注 : デフォルトの WebLogic 開始スクリプトの CLASSPATH には Oracle データベースドライバが含まれています。開始スクリプトを変更した場合、以下の jar を追加するようにしてください。
   ${UNIFY_HOME}/lib/jdbcDrivers/ojdbc14.jar

2. （MS SQL Server の場合）サーバの CLASSPATH に必要な jar ファイルを追加します。

   Windows
   %UNIFY_HOME%/lib/jdbcDrivers/jtds-1.1.jar

   Unix/Linux
   ${UNIFY_HOME}/lib/jdbcDrivers/jtds-1.1.jar

3. WebLogic サーバを再起動します。

2.8.3 WebLogic の構成を更新

WebLogic Server Console を使用して、以下の変更を正しいドメインに対して行って下さい。

1. NXJ データリポジトリテーブルを含むデータベースを使用するコネクションプールを作成します。

< Oracle データベースでは以下を実行します。>

   a. ナビゲーションツリーで、**Services Configurations** > **JDBC** を選択し、**Connection Pools** リンクをクリックします。

   b. **Configure a new JDBC Connection Pool** リンクをクリックします。

   c. Choose Database フォームに以下を入力し **Continue** をクリックします。

      Database Type: Oracle
      Database Driver: Oracle's Driver (Thin) Version 9.0.1, 9.2.0, 10

   d. “Define and test connection” フォームに以下を入力し、**Test Driver Configuration** をクリックします。

      Name: NXJDSPool
      Driver Classname: oracle.jdbc.OracleDriver
      URL: jdbc:oracle_clob:thin:@<host>:<port>:<sid>
      Database User Name: <username>
      Password: <password>
      Confirm Password: <password>

<host> は、データベースサーバ名です。
<port> は、データベースに接続用ポート番号です。
<sid> は、データベース sid です。
<username> は、データベースにログインするためのユーザ名です。
<password> は、そのユーザのパスワードです。

Test Driver Configuration をクリックすると、“Connection successful” というメッセージが表示されます。表示されない場合は、上記設定を確認して下さい。

注 : データベースドライバがサーバの CLASSPATH に含まれていない場合､WebLogic Server Console は “JDBC driver is not on the CLASSPATH” というメッセージを表示します。JdbcOraWrapperDriver が正しく Driver Classname に設定されている場合、JdbcOraWrapper.jar がサーバの CLASSPATH に設定されているか確認して下さい。(上記ステップ 2 を参照のこと)

e. **Create and Deploy** をクリックします。

< Microsoft SQL Server データベースでは以下を実行します。>

a. ナビゲーションツリーで、**Services** > **JDBC** を選択し、**Connection Pools** リンクをクリックします。

b. “Configure a new JDBC Connection Pools...” リンクをクリックします。

c. Choose Database フォームに以下を入力し **Continue** をクリックします。

Database Type: Other
Database Driver: Other

d. Define Connection プロパティに以下を入力し **Continue** をクリックします。

Name: NXJCCDSPool
Driver Classname: net.sourceforge.jtds.jdbc.Driver
URL: jdbc:jtds:sqlserver://<database host>:<database port>/<database name>
Database User Name: <username>
Password: <password>
Confirm Password: <password>

<database host> は、データベースサーバのホスト名です。
<database port> は、データベース接続用ポート番号です。
<database name> は、データベース名です。
<username> は、データベースにログインするためのユーザ名です。
<password> は、そのユーザのパスワードです。

e. Test data connection フォームにおいて、**Test Driver Configuration** ボタンをクリックすると、“Connection successful” というメッセージが表示されます。表示されない場合は、上記設定を確認して下さい。

f. **Create and Deploy** をクリックします。

2. NXJ データリポジトリテーブルを含むデータベースに対するデータソースを作成します。

a. ナビゲーションツリーで、**Services Configurations** > **JDBC** を開き、**Data Sources** をクリックします。

b. “Configure a new JDBC Data Source” リンクをクリックします。

c. “Configure the data source” フォームに以下の値を入力し、**Continue** をクリックします。

Name: NXJDS
JNDI Name: NXJDS
Honor Global Transactions: (check)
Emulate Two-Phase Commit for non-XA Driver: (check)

d. “Connect to connection pool” フォームに以下の値を入力し **Continue** をクリックします。

Pool Name: NXJDSPool

e. “Target the data source” フォームで、NXJ アプリケーションを配備するサーバを選択し、**Create** をクリックします。

2.8.4 WebLogic アプリケーションサーバの再起動

2.9 Oracle アプリケーションサーバ（OC4J or 10g）

NXJ アプリケーションをホストする Oracle アプリケーションサーバ（OAS）を構成するには、oracle NXJ リポジトリを示す oracle データソースを構成する必要があります。

注 : 現時点では、OAS を使用する場合、Oracle データベースのみがサポートされています。

2.9.1 リポジトリテーブルの作成

NXJ リポジトリデータをリポジトリテーブルをデータベースに作成します。

9 ページの「Oracle データベース」のステップ 1 を参照してください。

3 データソース定義

*データソース定義*は、DataSource オブジェクトを示し、それはデータベース等の特定のデータソースへの物理接続を表しています。アプリケーションサーバがデータソース定義を持っていない場合は、NXJ アプリケーションを配備することはできません。

開発環境のアプリケーションサーバは、アプリケーションデザイナで自動的に作成されるデータソース定義を使用します。実行環境が、開発環境において使用される同じアプリケーションサーバインスタンスを使用する場合は、新しいデータソース定義を作成する必要はありません。実行環境が異なるアプリケーションサーバを使用する場合は、そのサーバ上に新しいデータソース定義を作成しなければなりません。

このセクションは、以下のアプリケーションサーバ上でデータソース定義を作成するためのガイドラインを提供します。

- BEA WebLogic アプリケーションサーバ
- IBM WebSphere アプリケーションサーバ
- JBoss アプリケーションサーバ
- Oracle 10g アプリケーションサーバ

3.1 BEA WebLogic アプリケーションサーバ

Unify NXJ の使用に対する動作保証された BEA WebLogic のバージョンについては、『Unify NXJ がサポートする構成』を参照してください。

1. 該当する JDBC ドライバファイル（.zip または .jar）を含めるように、BEA WebLogic CLASSPATH を更新します。

CLASSPATH は、BEA WebLogic デフォルトサーバを起動するスクリプトに定義されています。このスクリプトは BEA WebLogic で提供されます。Windows では “setDomainEnv.cmd”、UNIX では “setDomainEnv.sh” がこのスクリプトに該当し、BEA WebLogic インストールディレクトリに格納されています。（デフォルト "weblogic81）

注 : Oracle JDBC ドライバを使用する場合は、CLASSPATH 定義文の先頭に .jar ファイルを追加して、WebLogic が提供する Oracle .jar ファイルを使用しないようにします。

2. アプリケーションに必要なデータソースの JDBC Connection Pool を新しく構成します。

   a. BEA WebLogic デフォルトサーバを起動します。

      これは NXJ アプリケーションが配備されたサーバです。

      サーバを起動させる方法はいくつかあります。”startWebLogic” や ”startManagedWebLogic” のスクリプトを実行します。あるいは Windows の場合、**スタート** > **プログラム** > **WebLogic** > **Start Default Server** を選択します。

   b. BEA WebLogic Server Console にて、**Services** > **JDBC** > **Connection Pools** を選択します。

   c. **Configure a new JDBC Connection Pool** をクリックします。

   d. General パネルで、データベースタイプに関する**フィールドエントリ**の値を入力します。

データベースタイプ	フィールド名	フィールドエントリ
IBM DB2	Name	<pool name>
	URL	**jdbc:db2:**<database name>
	Driver Classname	COM.ibm.db2.jdbc.app.DB2Driver
	Properties	<user-name>
	ACLName	ブランクのまま
	Password	**<password>**
	Open String Password	ブランクのまま
IBM Informix	Name	<pool name>
	URL	**jdbc:informix-sqli:**//<host>:<port>/<database name>: INFORMIXSERVER=<database server name> 説明 : <host>= ホスト名 ; <port>= データベースポート番号 ; <database server name>= データベースサーバ名
	Driver Classname	com.informix.jdbc.IfxDriver
	Properties	user=<user>
	ACLName	ブランクのまま
	Password	<password>

データベースタイプ	フィールド名	フィールドエントリ
	Open String Password	ブランクのまま
MS SQL Server 2000	Name	<pool name>
	URL	**jdbc:jtds:sqlserver:**//<host>:<port> 説明 : <host>= ホスト名 ; <port>= データベースポート番号
	Driver Classname	net.sourceforge.jtds.jdbc.Driver
	Properties	user=<user>
	ACLName	ブランクのまま
	Password	<password>
	Open String Password	ブランクのまま
MySQL	Name	<pool name>
	URL	**jdbc:mysql://**<host>**:**<port>**/**<dbname> 説明 : <host>= ホスト名 ; <port>= データベースポート番号 ; <dbname>= データベース名
	Driver Classname	com.mysql.jdbc.Driver
	Properties	user = <user >
	ACL Name	ブランクのまま
	Password	<password>
Oracle	Name	<pool name>
	URL	**jdbc:oracle:thin:**@<host>**:**<port>**:**<sid> 説明 : <host>= ホスト名 ; <port>= データベースポート番号 ; <sid> = データベース SID
	Driver Classname	oracle.jdbc.driver.OracleDriver
	Properties	user=<user>
	ACLName	ブランクのまま
	Password	<password>
	Open String Password	ブランクのまま
Sybase	Name	<pool name>
	URL	**jdbc:sybase:Tds:**<host>**:**<port> 説明 : <host>= ホスト名 ; <port>= データベースポート番号
	Driver Classname	com.sybase.jdbc2.jdbc.SybDriver

データベースタイプ	フィールド名	フィールドエントリ
	Properties	user=<user>
	ACLName	ブランクのまま
	Password	<password>
	Open String Password	ブランクのまま
Unify DataServer	Name	<pool name>
	URL	**jdbc:simba:**<host>:<port>**/**<DSN> 説明 : <host>= ホスト名 ; <port>= データベースポート番号 ; <DSN>= データソース名
	Driver Classname	simba.jdbc.SimbaDriver
	Properties	user=<user>
	ACL Name	ブランクのまま
	Password	<password>
	Open String Password	ブランクのまま
Unify SQLBase	Name	<pool name>
	URL	**jdbc:sqlbase:**<host>:<port>**/**<DSN> 説明 : <host>= ホスト名 ; <port>= データベースポート番号 ; <DSN>= データソース名
	Driver Classname	jdbc.gupta.sqlbase.SqlbaseDriver
	Properties	user=<user>
	ACL Name	ブランクのまま
	Password	<password>
	Open String Password	ブランクのまま

e. **作成** をクリックします。

接続プールが作成されます。

f. DataServer の場合、以下の方法でプロパティを設定します。

i. **接続**タブを開きます。

ii. **Prepared Statement Cache Size** プロパティを 0 に設定します。

iii. **適用** をクリックします。

g. **ターゲット**タブで、適切なサーバ名を**選択された**列に移動します。

デフォルトのサーバ名は “default” です。

h. **適用** をクリックします。

3. 接続プールの JDBC データソース定義を作成します。
   a. **Services** > **JDBC** > **Tx Data Sources** を選択します。
   b. “ **新しい JDBC Tx Data Source のコンフィグレーション** ” をクリックします。
   c. **コンフィグレーション**タブで、以下のように入力します。

      Name: <data source name>
      JNDI Name: <data source name>
      Pool Name: <pool name>

      その他の設定はデフォルトのままにしておきます。

   d. **作成** をクリックします。
   e. **ターゲット**タブで、適切なサーバを選択フィールドに移します
   f. **適用** をクリックします。

以上でデータソース定義が作成されました。サーバを再起動します。

3.2 IBM WebSphere アプリケーションサーバ

この章では、IBM WebSphere アプリケーションサーバに関しての手順を示します。Unify NXJ の使用に対する動作保証された IBM Websphere のバージョンについては、『Unify NXJ がサポートする構成』を参照してください。

1. WebSphere Application Server Administrative Console を開きます。

   データソースは NXJ アプリケーションを配備するホスト上に作成する必要があります。

   また、データソースにアクセス可能なユーザエイリアスのリストを指定する必要があります。

2. データベースに J2C 認証データエントリを作成します。
   a. **Security** > **JAAS Configuration** > **J2C Authentication Data** を選択します。
   b. **New** をクリックします。
   c. 新規エントリの **General プロパティ**で以下の情報を入力し、**OK** をクリックします。

      Alias: NXJDS_Credenitials
      User ID: <username>
      Password: <password>

   <username>/<password> には、NXJ データソースリポジトリテーブルを持つデータベースへログインする際に使用するユーザ名 / パスワードを指定します。

   これにより、<node>/NXJDS_Credenitials というエントリが作成されます。
   <node> は WebSphere サーバのノードです（通常、ホスト名です）。

3. データソースを作成します。

   データソース NXJDS は NXJ データリポジトリテーブルにアクセスします。

   Oracle データベースの設定

   a. ナビゲーションツリーで、**Resources** > **JDBC Providers** を選択します。
   b. **New** をクリックします。
   c. **JDBC Providers** ドロップダウンリストから **User-defined JDBC Driver** を選択し、**OK** をクリックします。
   d. 新しい JDBC Provider の **General プロパティ** テーブルへ以下の情報を入力し、**OK** をクリックします。

      Name: JdbcOraWrapper
      Description: JDBC provider for the NXJ Data Repository running on Oracle.
      Classpath: ${UNIFY_HOME}/lib/jdbcDrivers/ojdbc14.jar

      ${UNIFY_HOME}/lib/jdbcDrivers/JdbcOraWrapper.jar
      Implementation Classname: JdbcOraWrapperConnectionPoolDataSource

      ${UNIFY_HOME} は NXJ がインストールされたディレクトリを表します。または、WebSphere 変数 （UNIFY_HOME）を NXJ のインストールディレクトリを指すよう作成すると、上記のように指定することも可能です。WebSphere Administrative Console では、クラスパスの各項目は改行によって区切ることができます。

   e. **JdbcOraWrapper** をクリックします。
   f. **Data Sources** をクリックします。
   g. **New** をクリックします。
   h. データソースの **General プロパティ**テーブルに以下の情報を入力し、**OK** をクリックします。

      Name: NXJDS
      JNDI Name: NXJDS
      Database Helper Classname:
      com.ibm.Websphere.rsadapter.OracleDataStoreHelper
      Component-managed Authentication Alias: <node>/NXJDS_Credenitials
      Container-managed Authentication Alias: <node>/NXJDS_Credenitials

   i. **NXJDS** をクリックします。
   j. **Custom Properties** をクリックします。
   k. **New** をクリックします。
   l. **General プロパティ** テーブルに以下の情報を入力し、**OK** をクリックします。

      Name: URL
      Value: jdbc:oracle:thin:@<hostname>:<port>:<sid>
      <hostname> はデータベースのホスト名です。<port> はデータベースのポート番号です（Oracle では通常 1521）。<sid> はデータベースの SID です。

   Microsoft SQL Server データベースの設定

   a. **ナビゲーション**ツリーで、**Resources** > **JDBC Providers** を選択します。
   b. **New** をクリックします。

c. **JDBC Providers** ドロップダウンリストから **Microsoft JDBC driver for MSSQLServer 2000** を選択し、**OK** をクリックします。

d. JDBC Provider の **General プロパティ**テーブルへ以下の情報を入力し、**OK** をクリックします。

Classpath プロパティは MSSQLSERVER_JDBC_DRIVER_PATH（WebSphere 変数）を参照しています。このプロバイダでデータソースを使用するには、Environment -> Manage WebSphere Variables を選択し、Microsoft ドライバを含むディレクトリを指定します。または、Classpath プロパティに jar ファイルへの絶対パスを用いて指定します。

e. **Microsoft JDBC driver for MSSQLServer 2000** をクリックします。

f. **Data Sources** をクリックします。

g. **New** をクリックします。

h. データソースの **General プロパティ**テーブルに以下の情報を入力し、**OK** をクリックします。

Name: NXJCCDS
JNDI Name: NXJCCDS
Component-managed Authentication Alias: <node>/NXJCCDS server
（Value フィールドに入力します）

データベースが 1433 以外のポートを使用している場合、portNumber リンクをクリックし、ポート番号を Value フィールドに指定し、OK をクリックします。

4. **Resouces** グループで **JDBC Prividers** をクリックします。

JDBC Providers ページが表示されます。

5. **New** をクリックします。

Configuration 情報が表示されます。

6. General プロパティのドロップダウンリストから作成する JDBC Provider のタイプを選択して、**ApplyClick** ボタンを押下します。

以下の表は、Unify NXJ によってサポートされる様々なデータベースのタイプを示しています。残りの General プロパティは、JDBC プロバイダのタイプに基づいて表示されます。以下の表は、Unify NXJ での使用において動作保証された各データベースに対する推奨されるプロパティの値を記述しています。

プロバイダ	フィールドラベル	フィールドエントリ
DB2 Universal JDBC provider	Scope	データソースが適用されるノードの名前を指定します。
(Type 4)	Name	DB2 Universal JDBC Driver Provider
	Description	DB2 Universal JDBC Driver-compliant Provider
	Classpath	${DB2UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc.jar ${UNIVERSAL_JDBC_DRIVER_PATH}/ db2jcc_license_cu.jar ${DB2UNIVERSAL_JDBC_DRIVER_PATH}/ db2jcc_license_cisuz.jar

プロバイダ	フィールドラベル	フィールドエントリ
	Native Library Path	(ブランク)
	Implementation Classname	com.ibm.db2.jcc.DB2ConnectionPoolDataSource
DB2 Legacy JDBC provider	Scope	データソースが適用されるノードの名前を指定します。
(Type 2)	Name	DB2 Legacy CLI-based Type 2 JDBC Driver
	Description	DB2 JDBC2-compliant Provider
	Classpath	${DB2_JDBC_DRIVER_PATH}/db2java.zip
	Native Library Path	(ブランク)
	Implementation Classname	COM.ibm.db2.jdbc.DB2ConnectionPoolDataSource
Informix	このリリースではサポートされていません。	
MS SQL Server 2000	このリリースではサポートされていません。	
MySQL	このリリースではサポートされていません。	
Sybase JDBC Driver	Scope	データソースが適用されるノードを指定します。
	Name	Sybase JDBC Driver
	Description	Sybase JDBC Driver
	Classpath	${SYBASE_JDBC_DRIVER_PATH}/jconn2.jar
	Native Library Path	（ブランク）
	Implementation Classname	com.sybase.jdbc2.jdbc.SybConnectionPoolDataSource
Oracle JDBC Driver	Scope	データソースが適用されるノードを指定します。
	Name	Oracle JDBC Driver
	Description	Oracle JDBC Driver
	Classpath	${ORACLE_JDBC_DRIVER_PATH}/ojdbc14.jar
	Native Library Path	（ブランク）
	Implementation Classname	oracle.jdbc.pool.OracleConnectionPoolDataSource
Unify DataServer	このリリースではサポートされていません。	

7. **Apply** をクリックします。
8. **JDBC Providers** ページで、**Additional プロパティ**の下の **Data Sources** をクリックします。

Data Sources ページがデータベースドライバに表示されます。

9. **Data Sources** ページで **New** をクリックします。

   リソースのユニークな名前を入力して、それがまだクリアにされない場合は、Container managed persistence チェックボックスをクリアにします。

   Component-managed Authentication Alias では、実行時のデータベース認証のためのエイリアスを入力します。

   その他のフィールド値が、生成されます。

10. **Apply** をクリックします。

    新しく追加されたデータソースの構成が表示されます。

11. **Additinal プロパティ**グループで、**Custom プロパティ**をクリックします。

12. 以下の表で要約されるように、データベースのプロパティを設定します。

プロバイダタイプ	フィールドラベル	フィールドエントリ
DB2 JDBC provider (Type 4)	databaseName serverName portNumber	データベース名 サーバ名 ポート番号
DB2 JDBC provider (Type 2)	databaseName	データベース名
Informix JDBC Driver	このリリースではサポートされていません。	
Sybase JDBC Driver	databaseName	データベース名
	serverName	データベースサーバ名
	portNumber	データベースサーバのポート
Oracle JDBC Thin Driver	URL	データソースがコネクションを得るデータベース。
MSSQL Server 2000	このリリースではサポートされていません。	
Unify DataServer	このリリースではサポートされていません。	

13. **Apply** をクリックします。

14. データソース構成を**保存**します。

15. アプリケーションサーバを再起動します。

3.3 JBoss アプリケーションサーバ

JBoss アプリケーションサーバのデータソース定義を作成するには、開発環境で使用されるデータソース定義をコピーするか、または新しいデータソース定義を作成します。このセクションでは、この 2 つの作業について記述しています。

- 既存の JBoss データソース定義をコピーする方法
- 新しい JBoss データソース定義を作成する方法

Unify NXJ の使用に対する動作保証された JBoss のバージョンについては、『Unify NXJ がサポートする構成』を参照してください。

3.3.1 既存の JBoss データソース定義をコピーする

NXJ 開発環境から実行環境に JBoss データソース定義をコピーするステップは、以下のとおりです。

1. データソース定義を含む XML ファイルの場所を指定します。

   JBoss は、そのデータソース定義を XML ファイルに格納します。ファイル名とディレクトリは以下のとおりです。

   <UNIFY_WORK>¥jboss¥server¥default¥deploy¥UnifyNXJ-<connection name>-ds.XML.

   例えば、NXJ チュートリアルアプリケーションの場合、以下のファイルがあります。

   <C:¥Unify¥NXJWork¥jboss¥server¥default¥deploy¥UnifyNXJ-tutorial-ds.xml

2. プロダクション実行環境に XML ファイルをコピーします。

   ファイルは、ステップ 1 のように同等のパスですが、現在のプロダクション実行環境に配置する必要があります。従って、プロダクション実行環境のホストで以下のディレクトリにファイルを配置します。

   <JBOSS_HOME>¥server¥default¥deploy¥<yourXMLfile>

3. 必要に応じて、ファイルを編集します。

   必要であれば、ユーザ名とパスワードを入力することができます。また異なる JDBC ドライバを使用する場合は、ここに登録します。接続名を変更しないで下さい。例 :

```
 <?xml version="1.0" encoding="UTF-8" ?>
<datasources>
  <local-tx-datasource>
    <jndi-name>tutorial</jndi-name>
    <connection-url>jdbc:csv:C:/Unify/NXJWork/projects/
tutorials/databases/inet/db</connection-url>
    <driver-class>com.inet.csv.CsvDriver</driver-class>
    <user-name />
```

```
    <password />
  </local-tx-datasource>
</datasources>
```

4. JBoss インストールディレクトリに JDBC ドライバをコピーします。

   NXJ インストールは、サポートされる JDBC ドライバのサブセットのみを含みます。DBMS ドライバが含まれるかどうかを確認するために、『Unify NXJ 開発者ガイド』の第 3 章の 3.2 を確認してください。DBMS ドライバが含まれていない場合、それを取得します。

   JDBC ドライバの .zip あるいは .jar ファイルを以下にコピーします。

   <JBossInstallDir>/server/<serverName>/lib

   説明：<JBossInstallDir> は、JBoss がインストールされている場所で、<serverName> は、JBoss サーバの名称です。

   JDBC ドライバをコピーする場所は、『Unify NXJ 開発者ガイド』の第 3 章の「プロジェクトの作成」に記述されています。

5. DB2 データベースを使用する場合は、JBoss 起動スクリプトを変更する必要があります。Windows の場合は run.bat、UNIX の場合は run.sh を変更して DB2 クラスパスを追加します。

   Windows の run.bat では

   ```
   set JBOSS_CLASSPATH=%JBOSS_CLASSPATH%;%JAVAC_JAR%;%RUNJAR%
   ```

   の後に以下の行を追加します。

   ```
   set JBOSS_CLASSPATH=%JBOSS_CLASSPATH%;<SQLLIB>/java/
   db2java.zip
   ```

   UNIX の run.sh では

   ```
   JAVA_OPTS="$JAVA_OPTS -Dprogram.name=$PROGNAME"
   ```

   の前に以下の行を追加します。

   ```
   JBOSS_CLASSPATH=$JBOSS_CLASSPATH:<SQLLIB>/java/db2java.zip
   ```

   説明：<SQLLIB> は、DB2 がインストールされている場所です。

   この作業の実行後、サーバで JDBC ドライバクラスが利用できるようにするために JBoss サーバを再起動します。

3.3.2 新しいデータソース定義 XML ファイルを作成する

1. テキストエディタで -ds.xml ファイルを作成します。

   以下のテンプレートをエディタにコピーすることができます。-ds.xml ファイルの構文は以下のとおりです。

```
<datasources>
  <local-tx-datasource>
    <jndi-name>yourDataSourceName</jndi-name>
    <connection-url>yourJdbcURL</connection-url>
    <driver-class>yourJdbcDriverClass</driver-class>
    <user-name>yourUserName</user-name>
    <password>yourPassword</password>
  </local-tx-datasource>
</datasources>
```

説明 :

yourDataSourceName は、データソースの名称です。

yourJdbcURL は、JDBC URL です（以下の表を参照）。

yourJdbcDriverClass は、ドライバのクラス名です（以下の表を参照）。

yourUserName は、接続の確立に使用するユーザです。

yourPassword は、接続の確立に使用するパスワードです。

データベース名	ドライバ	URL
IBM DB2	COM.ibm.db2.jdbc.app.DB2Driver	jdbc:db2:<database name> 説明 : <database name> = データベース ID
IBM Informix	com.informix.jdbc.IfxDriver	jdbc:informix-sqli://<host>:<port>/<database name>:INFORMIXSERVER=<database server name> 説明 : <host>= ホスト名 ; <port>= データベースポート番号 <database name>= データベース名 <database server name>= データサーバ名
MS SQL Server 2000	net.sourceforge.jtds.jdbc.Driver	jdbc:jtds:sqlserver://<host>:<port> 説明 : <host> = ホスト名 <port> = データベースポート番号
MySQL	com.mysql.jdbc.Driver 以下のサイトからドライバをダウンロードして <UNIFY_HOME>¥lib¥jdbcDrivers¥ にインストールする必要があります。 http://www.mysql.com/downloads/api-idebc-stable.html	jdbc:mysql://<host>:<port>/<dbname> 説明 : <host>= ホスト名 ; <port>= データベースポート番号 ; <dbname>= データベース名
Oracle	oracle.jdbc.driver.OracleDriver	jdbc:oracle:thin:@<host>:<port>:<SID> 説明 : <host> = ホスト名 <port> = データベースポート番号 <SID> = データベース SID

データベース名	ドライバ	URL
SQL Base	jdbc.gupta.sqlbase.SqlbaseDriver	jdbc:sqlbase://<host>:<port>/<dbname> 説明 : <host>= ホスト名 ; <port>= データベースポート番号 <dbname>= データベースの名前
Sybase	com.sybase.jdbc2.jdbc.SybDriver	jdbc:sybase:Tds:<host>:<port> 説明 : <host> = ホスト名 <port> = データベースポート番号
Unify DataServer	simba.jdbc.SimbaDriver	jdbc:simba:<host>:<port>/<DSN> 説明 : <host> = ホスト名 <port> = データベースポート番号 <DSN> = データソース名

2. サーバの deploy ディレクトリにそれをコピーすることで、このファイルを配備します。

   <JBossInstallDir>/server/<serverName>/deploy

   説明 :

   <JBossInstallDir> は、JBoss がインストールされている場所です。

   <serverName> は、JBoss サーバの名称です。

3. JBoss サーバを再起動します。

   これで、データソースの定義が完了します。

3.4 Oracle Application Server 10g

Unify NXJ の使用に対する動作保証された OracleAS のバージョンについては、『Unify NXJ がサポートする構成』を参照してください。

データソースの定義方法は、使用している Oracle Application サーバのタイプ（OC4J または Enterprise Edition）によって異なります。

3.4.1 OC4J

1. コマンドウィンドウを開きます。
2. <Oracle installation directory>/j2ee/<server name> ディレクトリに移動します。

   説明 :

   <Oracle installation directory> は、Oracle9iAS Containers for J2EE（OC4J）がインストールされているディレクトリの名称です。

<server name> はアプリケーションサーバの名称で、デフォルトの名称は Unify NXJ アプリケーションサーバのデフォルトのインスタンスである “home” です。

3. アプリケーションサーバを起動するために、`java -jar oc4j.jar` コマンドを入力します。
4. 新しいコマンドウィンドウを開き、ステップ 2 にアクセスして <Oracle installation directory>/j2ee/<server name> ディレクトリに移動します。
5. config/rmi.xml ファイルを開き、rmi-server ポートを確認します。

   この <port number> はステップ 6 で使用します。
6. 以下の java コマンドを入力し、デフォルトサーバにデータソースをインストールします。コマンドの大文字と小文字は区別されます。

```
% java -jar admin.jar ormi://<localhost>:<port number> admin
<administrative password>
-application default
-installDataSource
-className com.evermind.sql.OrionCMTDataSource
-url jdbc:oracle:thin:@<host>:<port>:<sid>
-connectionDriver oracle.jdbc.OracleDriver
-location <data source name>
-username <database username> -password <database password>
```

説明 :

<localhost> は、ORMI ホストです。

<port number> は、ORMI ポートです。

<administrative password> は、ORMI 管理パスワードです。

<host> は、データベースのホスト名です。

<port> は、データベースに接続するためのポート番号です。

<sid> は、データベースシステム ID です。

<data source name> は、作成するデータベース定義の名称です。

className は、データソースを実装するクラスの名称です。

url は、データベース接続の URL です。

connectionDriver は、このデータソースの JDBC ドライバのクラス名です。

location は、データソースオブジェクトの JNDI 論理名です。

username は、接続先スキーマの名称です（オプション）。

password は、接続先スキーマのパスワードです（オプション）。

以上でデータソースが作成されました。

3.4.2 Enterprise Edition

Appllication Server Control ページの J2EE Applications リンクをクリックします。

1. **default** をクリックします。
2. **Resources** リスト以下にある、**Data Sources** をクリックします。
3. このアプリケーションサーバインスタンス用のデータソース定義が既に存在する場合には、**Create Like** をクリックして下さい。データソース定義が無い場合は、**Create** をクリックします。

   定義するデータソースに応じてページの各項目を指定して下さい。詳細については、Oracle 10g のドキュメントを参照してください。
4. ページの一番下で、**Create** をクリックします。

   データソースが作成されます。

4 NXJ - アプリケーションサーバの構成

このドキュメントは、以下のアプリケーションサーバの NXJ フォームアプリケーションを配備するための構成ガイドラインを提供します。

- JBoss アプリケーションサーバ
- IBM WebSphere アプリケーションサーバ
- WebLogic アプリケーションサーバ
- Oracle 10g アプリケーションサーバ

4.1 JBoss - NXJ の構成

この説明は、スタンドアローン JBoss アプリケーションマシン用です。動作保証された JBoss アプリケーションサーバについては、『UNIFY NXJ がサポートする構成』を参照してください。

注 : NXJ Developer または NXJ Enterprise Developer にバンドルされている JBoss の場合、予め設定されています。

注 : jboss4.2.2GA を使用するために以下を設定します。
Windows インストール ー run.bat -b 0.0.0.0
Unix インストール ー run.sh -b 0.0.0.0

NXJ Developer と NXJ Enterprise Developer の 2 種類の構成がありますので、使用する製品の構成を行ってください。
（NXJ Developer につきましては、2010 年 1 月現在、日本国内では販売しておりません。NXJ Enterprise Developer に関する記載を参照してください。）

4.1.1 NXJ Developer の構成

NXJ のこのバージョンは、NXJ Enterprise Server をインストールする必要はありません。以下の構成は、jboss アプリケーションサーバに NXJ フォームアプリケーションの ear ファイルを配備する前に行う必要があります。

4.1.1.1 NXJ リポジトリデータソースの設定

第 2 章の「NXJ リポジトリの構成」を参照してください。

4.1.1.2 NXJ アプリケーションデータソースの設定

第 3 章のセクション 3.3「JBoss アプリケーションサーバ」を参照してください。

4.1.1.3 Unify NXJ Stateful SessionBean のコンテナを追加

1. NXJ Developer をインストールした箇所に行き、../Unify/NXJ/jboss/server/default/conf/standardjboss.xml を開きます。
2. このファイルの下部にスタンドアローン JBoss アプリケーションサーバ上の同等な standardjboss.xml に格納される“Unify NXJ Stateful SessionBean”コンテナをコピーします。

4.1.2 NXJ Enterprise Developer の構成

NXJ のこのバージョンは、使用するアプリケーションサーバと同じマシンに NXJ Enterprise Server をインストールする必要があります。NXJ Enterprise Developer で作成される NXJ アプリケーションが機能するためには、NXJ Enterprise Server が必要です。NXJ Enterprise Server をインストールして、以下を行います。

4.1.2.1 上記の NXJ Developer と同じステップを行う

4.1.2.2 アプリケーションサーバの startup スクリプトの編集

Windows 上では、これは run.bat です。UNIX ベースのシステムでは、run.sh と run.conf です。このファイルは、<UNIFY_HOME>/bin ディレクトリにあります。

以下の行を追加します。

run.bat
set UNIFY_HOME=C:\Unify\NXJ
set UNIFY_WORK=C:\Unify\NXJWork

set JAVA_OPTS=-Dunify.home=%UNIFY_HOME% -Dunify.work=%UNIFY_WORK% -Durl.protocol=file:/:%JAVA_OPTS%

注 : 上記の編集は、JBoss の Windows インストール用です。ここでは、C: \Unify\NXJ の下に NXJ Enterprise Server をインストールしました。

run.sh
UNIFY_HOME=/opt/Unify/NXJ
UNIFY_WORK=/opt/Unify/NXJWork

run.conf
JAVA_OPTS=”-Dunify.home=$UNIFY_HOME -Dunify.work=$UNIFY_WORK -Durl.protocol=file:/ $JAVA_OPTS”

注 : 上記の編集は、JBoss の Linux のインストール用です。ここでは、/opt/Unify の下に NXJ Enterprise Server をインストールしました。

4.2 WebSphere - NXJ の構成

この説明は、スタンドアローン WebSphere アプリケーションサーバマシン用です。動作保証された WebSphere アプリケーションサーバについては、『UNIFY NXJ がサポートする構成』を参照してください。

NXJ Developer と NXJ Enterprise Developer の 2 種類の構成がありますので、使用する製品の構成を行ってください。
（NXJ Developer につきましては、2010 年 1 月現在、日本国内では販売しておりません。NXJ Enterprise Developer に関する記載を参照してください。）

4.2.1 NXJ Developer の構成

NXJ のこのバージョンは、NXJ Enterprise Server をインストールする必要はありません。websphere アプリケーションサーバに NXJ フォームアプリケーション ear ファイルを配備する前に必要とする 1 つの構成項目しかありません。

4.2.1.1 NXJ リポジトリデータソースの設定

第 2 章の「NXJ リポジトリの構成」を参照してください。

4.2.1.2 NXJ アプリケーションデータソースの設定

第 3 章のセクション 3.2「IBM WebSphere アプリケーションサーバ」を参照してください。

4.2.2 NXJ Enterprise Developer の構成

NXJ のこのバージョンは、使用するアプリケーションサーバと同じマシンに NXJ Enterprise Server をインストールする必要があります。NXJ Enterprise Developer で作成される NXJ アプリケーションが機能するためには、NXJ Enterprise Server が必要です。NXJ Enterprise Server をインストールして、以下を行います。

4.2.2.1 上記の NXJ Developer と同じステップを行う

4.2.2.2 呼び出されたアプリケーションサーバの startup スクリプトの 1 つを編集

Windows 上では、setupCmdLine.bat を選択します。UNIX ベースのシステムでは、setCmdLine.sh です。このファイルは、<WAS_HOME>/AppServer/bin ディレクトリにあります。

以下の行を追加します。

setupCmdLine.bat
set UNIFY_HOME=C:\Unify\NXJ
set UNIFY_WORK=C:\Unify\NXJWork

set JAVA_OPTS=%JAVA_OPTS% -Dunify.home=%UNIFY_HOME% -Dunify.work=%UNIFY_WORK% -Durl.protocol=file:/

注 : 上記の編集は、WebSphere の Windows インストール用です。ここでは、C: \Unify の下に NXJ Enterprise Server をインストールしました。

setupCmdLine.sh
UNIFY_HOME=/opt/Unify/NXJ
UNIFY_WORK=/opt/Unify/NXJWork

JAVA_OPTS=”-Dunify.home=$UNIFY_HOME -Dunify.work=$UNIFY_WORK -Durl.protocol=file:/:$JAVA_OPTS”

注 : 上記の編集は、WebSphere の Linux のインストール用です。ここでは、/opt/Unify の下に NXJ Enterprise Server をインストールしました。

4.3 Weblogic - NXJ の構成

この説明は、スタンドアローン WebLogic アプリケーションサーバマシン用です。動作保証された WebLogic アプリケーションサーバについては、『UNIFY NXJ がサポートする構成』を参照してください。

NXJ Developer と NXJ Enterprise Developer の 2 種類の構成がありますので、使用する製品の構成を行ってください。
（NXJ Developer につきましては、2010 年 1 月現在、日本国内では販売しておりません。NXJ Enterprise Developer に関する記載を参照してください。）

4.3.1 NXJ Developer の構成

NXJ のこのバージョンは、NXJ Enterprise Server をインストールする必要はありません。WebLogic アプリケーションサーバに NXJ フォームアプリケーション ear ファイルを配備する前に必要とする 1 つの構成項目しかありません。

4.3.1.1 NXJ リポジトリデータソースの設定

第 2 章の「NXJ リポジトリの構成」を参照してください。

4.3.1.2 NXJ アプリケーションデータソースの設定

第 3 章のセクション 3.1「BEA WebLogic アプリケーションサーバ」を参照してください。

4.3.2 NXJ Enterprise Developer の構成

NXJ のこのバージョンは、使用するアプリケーションサーバと同じマシンに NXJ Enterprise Server をインストールする必要があります。NXJ Enterprise Developer で作成される NXJ アプリケーションが機能するためには、NXJ Enterprise Server が必要です。NXJ Enterprise Server をインストールして、以下を行います。

4.3.2.1 上記の NXJ Developer と同じステップを行う

4.3.2.2 アプリケーションサーバの startup スクリプトを編集

Windows 上では、startWLS.cmd を選択します。UNIX ベースのシステムでは、startWLS.sh です。このファイルは、<WL_HOME>/bin ディレクトリにあります。

以下の行を追加します。

setDomainEnv.cmd

```
set UNIFY_HOME=C:\Unify\NXJ
set UNIFY_WORK=C:\Unify\NXJWork

set JAVA_OPTS=%JAVA_OPTS% -Dunify.home=%UNIFY_HOME% -
Dunify.work=%UNIFY_WORK% -Durl.protocol=file:/
```

注 : 上記の編集は、weblogic の Windows インストール用です。ここでは、C: \Unify の下に NXJ Enterprise Server をインストールしました。

setDomainEnv.sh

```
UNIFY_HOME=/opt/Unify/NXJ
UNIFY_WORK=/opt/Unify/NXJWork

JAVA_OPTS="-Dunify.home=$UNIFY_HOME -Dunify.work=$UNIFY_WORK -
Durl.protocol=file:/:$JAVA_OPTS"
```

注： 上記の編集は、websphere の Linux のインストール用です。ここでは、/opt/Unify の下に NXJ Enterprise Server をインストールしました。

4.4 Oracle 10g - NXJ の構成

この説明は、スタンドアローン Oracle アプリケーションサーバマシン用です。動作保証された Oracle アプリケーションサーバについては、『UNIFY NXJ がサポートする構成』を参照してください。

NXJ Developer と NXJ Enterprise Developer の 2 種類の構成がありますので、使用する製品の構成を行ってください。
（NXJ Developer につきましては、2010 年 1 月現在、日本国内では販売しておりません。NXJ Enterprise Developer に関する記載を参照してください。）

4.4.1 NXJ Developer の構成

NXJ のこのバージョンは、NXJ Enterprise Server をインストールする必要はありません。oracle アプリケーションサーバに NXJ フォームアプリケーション ear ファイルを配備する前に必要とする 1 つの構成項目しかありません。

4.4.1.1 NXJ リポジトリデータソースの設定

第 2 章の「NXJ リポジトリの構成」を参照してください。

4.4.1.2 NXJ アプリケーションデータソースの設定

第 3 章のセクション 3.4「Oracle Application Server 10g」を参照してください。

4.4.2 NXJ Enterprise Developer の構成

NXJ のこのバージョンは、使用するアプリケーションサーバと同じマシンに NXJ Enterprise Server をインストールする必要があります。NXJ Enterprise Developer で作成される NXJ アプリケーションが機能するためには、NXJ Enterprise Server が必要です。NXJ Enterprise Server をインストールして、以下を行います。

4.4.2.1 上記の NXJ Developer と同じステップを行う

4.4.2.2 アプリケーションサーバの startup スクリプトを編集

このファイルは opmnctl と呼ばれ、<ORACLE_HOME>/opmn/bin ディレクトリにあります。

以下の行を追加します。

windows:
set UNIFY_HOME=C:\Unify\NXJ
set UNIFY_WORK=C:\Unify\NXJWork

unix:
set UNIFY_HOME=/opt/Unify/NXJ
set UNIFY_WORK=/opt/Unify/NXJWork

4.4.2.3 JAVA オプションの編集

<ORACLE_HOME>/opmn/conf/ ディレクトリの下にある opmn.xml ファイルに以下の行を追加します。

windows:
set JAVA_OPTS=%JAVA_OPTS% -Dunify.home=%UNIFY_HOME% -Dunify.work=%UNIFY_WORK% -Durl.protocol=file:/

unix:
JAVA_OPTS="-Dunify.home=$UNIFY_HOME -Dunify.work=$UNIFY_WORK -Durl.protocol=file:/:$JAVA_OPTS"

5 NXJ ActiveSOA - アプリケーションサーバの構成

このセクションは、以下のアプリケーションサーバに NXJ ActiveSOA サーバを配備するためのガイドラインを提供します。

- JBoss アプリケーションサーバ
- IBM WebSphere アプリケーションサーバ
- WebLogic アプリケーションサーバ
- Oracle 10g アプリケーションサーバ

5.1 JBoss - NXJ ActiveSOA の構成

この説明は、スタンドアローン JBoss アプリケーションサーバマシン用です。動作保証された JBoss アプリケーションサーバについては、『UNIFY NXJ がサポートする構成』を参照してください。

注 : NXJ Developer または NXJ Enterprise Developer にバンドルされている JBoss の場合、予め設定されています。
NXJ Developer につきましては、2009 年 1 月現在、日本国内では販売しておりません。NXJ Enterprise Developer に関する記載をご参照ください。

注 : バンドルされた JBoss を使用しない場合は、NXJ ActiveSOA を構成する前に、4.1 章「JBoss - NXJ の構成」を必ずお読みください。

5.1.1 JBoss アプリケーションサーバをシャットダウン

5.1.2 JBoss アプリケーションサーバの systinet war の作成

1. NXJ Developer または NXJ Enterprise server で以下のファイルを編集します。

   <UNIFY_HOME>/wasp/conf/database.xml

   jboss hypersonic database プロパティファイルに絶対パスでエレメント <dbName> の値を置き換える必要があります。（例えば、<JBOSS_HOME>/server/default/data/hypersonic/localDB.properties for jboss 4.2.2.GA）

注 : hypersonic を使用しない場合、ユーザの選択する別のデータベースを使用することができます。

2. NXJ Developer または NXJ Enterprise server をインストールした所で、コマンドラインから <UNIFY_HOME>\NXJ\wasp\bin\ ディレクトリに行きます。
3. 以下のバッチスクリプトを実行します。

   J2eeIntegrate.bat --installation-type=jboss4

注 : これは、<UNIFY_HOME>/NXJ/ wasp/conf/porting/jboss/build ディレクトリ下に systinet.war ファイルを作成します。

4. systinet.war と呼ばれるフォルダに systinet war ファイルを展開します。次に、jboss サーバ配備ディレクトリの <JBOSS_HOME>\server\default\deploy にこれを配置します。

5.1.3 Production jboss サーバログイン - config.xml ファイルの構成

1. ファイル <JBOSS_HOME>/server/default/conf/login-config.xml を開いて、タグ <policy>.....</policy> の間に前のステップで実行中の J2eeIntegrate.bat --installation-type=jboss4 から与えられた出力から内容を貼り付けます。

注 : この出力は、<UNIFY_HOME>/NXJ/wasp/conf/instnotes.txt ファイルにあります。

5.1.4 “bin/run.conf”の編集

1. サーバサイドの JBoss インストールで、“bin/run.conf”ファイルを編集し、以下の JAVA_OPT エントリを追加します。

```
JAVA_OPTS="$JAVA_OPTS -Dwasp.location=/opt/jboss-4.2.2.GA/server/default/deploy/systinet.war"
```

（配備された systinet.war ディレクトリの現時点の位置を置き換えます）

5.1.5 アプリケーションサーバに“security-ng.jar”をコピー

注 : NXJ Enterprise Server を使用している場合、このステップは必要ありません。

NXJ Developer をインストールした場所に移動し、<JBOSS_HOME>/server/default/lib ディレクトリに <UNIFY_HOME>/wasp/lib/security-ng.jar をコピーします。

5.1.6 アプリケーションサーバに“nxjwaspjass.jar”をコピー

注 : NXJ Enterprise Server を使用している場合、このステップは必要ありません。

NXJ Developer インストールした場所に移動し、<JBOSS_HOME>/server/default/lib ディレクトリに <UNIFY_HOME>/NXJ/lib/nxjwaspjaas.jar をコピーします。

5.1.7 アプリケーションサーバに“jass.config”をコピー

注 : NXJ Enterprise Server を使用している場合、このステップは必要ありません。

NXJ Developer インストールした場所に移動し、<JBOSS_HOME>/server/default/conf ディレクトリに <UNIFY_HOME>/wasp/config/jaas.config をコピーします。

5.1.8 ‘jboss-service.xml’ ファイルの構成

<JBOSS_HOME>/server/default/conf/jboss-service. xml ファイルの‘CallByValue’属性を true に設定します。

5.1.9 ‘ear-deployer.xml’ ファイルの構成

<JBOSS_HOME>/server/default/ deploy/ear-deployer.xml ファイルの CallByValue’属性と‘Isolated’属性を true に設定します。

5.1.10 アプリケーションサーバ startup スクリプトの編集

Windows では、これは run.bat を、UNIX ベースのシステムでは run.sh と run.conf を構成する必要があります。

このファイルは、<JBOSS_HOME>/bin ディレクトリの下に格納することができます。

以下の行を追加します。

run.bat

```
set JAVA_OPTS=%JAVA_OPTS% -
Dorg.apache.axis.soap.SOAPFactoryImpl=com.systinet.saaj.soap.SOAPFactoryImpl
set JAVA_OPTS=%JAVA_OPTS% -
Djavax.xml.soap.MessageFactory=com.systinet.saaj.soap.DefaultMessageFactory
set JAVA_OPTS=%JAVA_OPTS% -
Djavax.xml.soap.SOAPFactory=com.systinet.saaj.soap.SOAPFactoryImpl

set JAVA_OPTS=%JAVA_OPTS% - Djava.security.auth.login.config=
%JBOSS_HOME%/server/default/conf/jaas.config
set JBOSS_CLASSPATH=%JBOSS_CLASSPATH%;%JBOSS_HOME%
\server\default\lib\security-ng.jar
set JBOSS_CLASSPATH=%JBOSS_CLASSPATH%;%JBOSS_HOME%
\server\default\lib\nxjwaspjaas.jar
```

注 : NXJ Enterprise を使用している場合、以下のように JBOSS_CLASSPATH リファレンスを変更する必要があります。
“security-ng.jar”:<UNIFY_HOME>\wasp\lib\security-ng.jar.
“nxjwaspjaas.jar”:<UNIFY_HOME>\lib\nxjwaspjaas.jar

run.sh

```
JBOSS_CLASSPATH=”$JBOSS_CLASSPATH:$JBOSS_HOME/server/default/lib/
security-ng.jar”
JBOSS_CLASSPATH=”$JBOSS_CLASSPATH:$JBOSS_HOME/server/default/lib/
nxjwaspjaas.jar”
```

注 : NXJ Enterprise を使用している場合、以下のように JBOSS_CLASSPATH リファレンスを変更する必要があります。
“security-ng.jar”: <UNIFY_HOME>\wasp\lib\security-ng.jar
“nxjwaspjaas.jar”: <UNIFY_HOME>\lib\nxjwaspjaas.jar

run.conf

以下の java オプションを追加します。

```
-Dorg.apache.axis.soap.SOAPFactoryImpl=com.systinet.saaj.soap.SOAPFactoryImpl
-Djavax.xml.soap.MessageFactory=com.systinet.saaj.soap.DefaultMessageFactory
-Djavax.xml.soap.SOAPFactory=com.systinet.saaj.soap.SOAPFactoryImpl
-Djava.security.auth.login.config=/opt/jboss-4.2.2.GA/server/default/conf/jaas.config
```

注 : 設定されている他の java オプションの次にこれらの java オプションを置きます。

5.1.11 joss アプリケーションサーバを起動して NXJ ActiveSOA admin console にアクセス

JBoss アプリケーションサーバを起動して、以下の URL にアクセスします。

http: //<host>: <port>/systinet/server/admin/console

注 : jboss アプリケーションサーバと同じマシン上のブラウザを起動している場合、この URL は通常以下の通りです。
http://localhost: 8080/ systinet/server/admin/console

注 : 以下のユーザとパスワードで systinet admin server にログインできます。
ユーザ：**admin**
パスワード：**changeit**

5.1.12 NXJ アプリケーション ear の配備と web サービスの関連付け

5.1.12.1 NXJ アプリケーション ear の配備

<NXJProject>\output\<NXJProjectName>.ear から NXJ アプリケーション ear を得て、joss 配備ディレクトリ <JBOSS_HOME>\server\default\deploy に配置します。

5.1.12.2 NXJ アプリケーションの関連する web サービスの配備

systinet コンソールに移動して、Deployment -> Deploy new package に移動します。
以下の設定をセットします。
Existing Contexts - （デフォルト）
Context name: packages/<NXJProjectName>
Disable service instances: チェックを外す
Select package to deploy: web サービス jar ファイルをブラウズして、以下の注意を参照します。

注 : NXJ web サービスの位置は以下の通りです。
<NXJProject>\output\webnxj\packagecontents\wasp\<servicename>.jar

5.2 Weblogic - NXJ ActiveSOA の構成

5.2.1 weblogic アプリケーションサーバをシャットダウン

5.2.2 weblogic アプリケーションサーバの systinet war の作成

1. NXJ Developer または NXJ Enterprise server で以下のファイルを編集します。

   <UNIFY_HOME>/wasp/conf/database.xml

   あなたの選択したデータベースを示すには、このファイルを構成する必要があります。

注 : デフォルトは、jboss hypersonic データベースです。

2. NXJ Developer または NXJ Enterprise server に行き、以下のファイルを編集します。

   <UNIFY_HOME>/wasp/conf/jaas.config

   “NamePasswordAN” 設定を変更します。

   NamePasswordAN{
   com.idoox.security.jaas.WasNamePasswordLoginModule required debug=true;
   com.idoox.security.jaas.NamePasswordLoginModuleNoAuth required debug=true;
   };.

3. NXJ Developer または NXJ Enterprise server インストールした場所に移動し、コマンドラインから <UNIFY_HOME>\NXJ\wasp\bin\ ディレクトリに行きます。

4. 以下のバッチスクリプトを実行します。

   J2eeIntegrate.bat --installation-type=weblogic

注 : これは、<UNIFY_HOME>/NXJ/ wasp/config/porting/weblogic/build ディレクトリ下に systinet.war ファイルを作成します。

5. 上記で作成された systinet war を抽出し、次のことを行います。

   次のように呼ばれる二重のコンテキストパラメータセクションを削除します。WEB-INF/web.xml の wasp.servlet.context は、WEB-INF/lib/wasp.jar/javax/xml/namespace/QName.class を削除します。

注 : wasp.jar を抽出する必要があります。

6. ステップ 5 を行った wasp.jar を zip ファイルにし、次にすべてのものを zip ファイルにしたものを systinet.war と名づけます。

5.2.3 アプリケーションサーバに“security-ng.jar”をコピー

注 : NXJ Enterprise Server を使用している場合、このステップは必要ありません。

NXJ Developer インストールした場所に移動し、<WL_HOME>/server/lib ディレクトリに <UNIFY_HOME>/wasp/lib/security-ng.jar をコピーします。

5.2.4 アプリケーションサーバに“nxjwaspjass.jar”をコピー

注 : NXJ Enterprise Server を使用している場合、このステップは必要ありません。

NXJ Developer インストールした場所に移動し、<WL_HOME>/server/default/lib ディレクトリに <UNIFY_HOME>/lib/nxjwaspjaas.jar をコピーします。

5.2.5 アプリケーションサーバに“j2ee_jndi_connerctor.jar”をコピー

注 : NXJ Enterprise Server を使用している場合、このステップは必要ありません。

NXJ Developer インストールした場所に移動し、<WL_HOME>/server/default/lib ディレクトリに <UNIFY_HOME>/wasp/conf/porting/dist/j2ee_jndi_connector.jar をコピーします。

5.2.6 呼び出されたアプリケーションサーバ startup スクリプトの編集

Windows では、setDomainEnv.cmd を選択し、UNIX ベースのシステムでは、setDomainEnv.sh を選択します。
このファイルは、<WL_HOME>/bin ディレクトリ以下にあります。

以下の行を追加します。

setDomainEnv.bat

```
set JAVA_OPTIONS=%JAVA_OPTIONS% -
Dorg.apache.axis.soap.SOAPFactoryImpl=com.systinet.saaj.soap.SOAPFactoryImpl
set JAVA_OPTIONS=%JAVA_OPTIONS% -
Djavax.xml.soap.MessageFactory=com.systinet.saaj.soap.DefaultMessageFactory
set JAVA_OPTIONS=%JAVA_OPTIONS% -
```

Djavax.xml.soap.SOAPFactory=com.systinet.saaj.soap.SOAPFactoryImpl
set JAVA_OPTIONS=%JAVA_OPTIONS% -
Djavax.rmi.PortableObject.narrow=com.idoox.wasp.jndi.
PortableRemoteObjectDelegate
set JAVA_OPTIONS=%JAVA_OPTIONS% -Djava.naming.security.principal=admin
set JAVA_OPTIONS=%JAVA_OPTIONS% -Djava.naming.security.credentials=changeit

set PRE_CLASSPATH=%PRE_CLASSPATH%;%WL_HOME%\server\default\
lib\security-ng.jar
set PRE_CLASSPATH=%PRE_CLASSPATH%;%WL_HOME%\server\default\
lib\nxjwaspjaas.jar
set PRE_CLASSPATH=%PRE_CLASSPATH%;%WL_HOME%\server\default\
lib\j2ee_jndi_connector.jar

注 : NXJ Enterprise Server を使用している場合、以下のように PRE_CLASSPATH リファレンスを変更する必要があります。

“security-ng.jar” :
<UNIFY_HOME>\wasp\lib\security-ng.jar.
“nxjwaspjaas.jar” :
<UNIFY_HOME>\lib\nxjwaspjaas.jar
“j2ee_jndi_connector.jar” :
<UNIFY_HOME>\wasp\conf\porting\dist\nxjwaspjaas.jar

setDomainEnv.sh
JAVA_OPTIONS=”$JAVA_OPTIONS -
Dorg.apache.axis.soap.SOAPFactoryImpl=com.systinet.saaj.soap.SOAPFactoryImpl”
JAVA_OPTIONS=”$JAVA_OPTIONS -
Djavax.xml.soap.MessageFactory=com.systinet.saaj.soap.DefaultMessageFactory”
set JAVA_OPTIONS=”$JAVA_OPTIONS -
Djavax.xml.soap.SOAPFactory=com.systinet.saaj.soap.SOAPFactoryImpl”
set JAVA_OPTIONS=”$JAVA_OPTIONS -
Djavax.rmi.PortableObject.narrow=com.idoox.wasp.jndi.
PortableRemoteObjectDelegate”
set JAVA_OPTIONS=$JAVA_OPTIONS -Djava.naming.security.principal=admin”
set JAVA_OPTIONS=$JAVA_OPTIONS -Djava.naming.security.credentials=changeit”

PRE_CLASSPATH=”$PRE_CLASSPATH:$WL_HOME/server/default/
lib/security-ng.jar”
PRE_CLASSPATH=”$PRE_CLASSPATH:$WL_HOME/server/default/
lib/nxjwaspjaas.jar”
PRE_CLASSPATH=%PRE_CLASSPATH%:%WL_HOME%/server/default/
lib/j2ee_jndi_connector.jar

注 : NXJ Enterprise Server を使用している場合、以下のように PRE_CLASSPATH リファレンスを変更する必要があります。

“security-ng.jar” :
<UNIFY_HOME>/wasp/lib/security-ng.jar.
“nxjwaspjaas.jar” :
<UNIFY_HOME>/lib/nxjwaspjaas.jar
“j2ee_jndi_connector.jar” :
<UNIFY_HOME>/wasp/conf/porting/dist/nxjwaspjaas.jar

5.2.7 admin user の作成

“Security Realms/Myrealm”の下の“Users and groups”で、admin group を削除してパスワード‘changeit’で admin user を作成します。

5.2.8 アプリケーションサーバに“systinet.war”を配備

アプリケーションサーバに systinet.war を配備します。

注 : これは、NXJ Developer または NXJ Enterprise Server をインストールした <UNIFY_HOME>/lib/systinet.war ファイルです。

5.2.9 weblogic アプリケーションサーバを開始して NXJ ActiveSOA 管理コンソールにアクセス

weblogic アプリケーションサーバを起動して、以下の URL にアクセスします。

http://<host>:<port>/systinet/server/admin/console

注 : jboss アプリケーションサーバと同じマシン上のブラウザを起動している場合、この URL は通常以下の通りです。
http://localhost:7001/systinet/server/admin/console

5.3 Oracle 10g - NXJ ActiveSOA の構成

5.3.1 Oracle 10g アプリケーションサーバの systinet war の作成

1. NXJ Developer または NXJ Enterprise Server で以下のファイルを編集します。
   <UNIFY_HOME>/wasp/conf/database.xml

   あなたの選択したデータベースを示すには、このファイルを構成する必要があります。

   注 : デフォルトは、jboss hypersonic データベースです。
2. NXJ Developer または NXJ Enterprise Developer インストールした場所に移動し、コマンドラインから <UNIFY_HOME>\NXJ\wasp\bin\ ディレクトリに行きます。
3. 以下のバッチスクリプトを実行します。

J2eeIntegrate.bat --modify-environment --installation-type=oracle

注 : これは、<UNIFY_HOME>/NXJ/ wasp/config/porting/oracle/build ディレクトリ下に systinet.war ファイルを作成します。

5.3.2 2 つの NXJ ActiveSOA 関連 jar ファイル用に shared library を Oracle 10g アプリケーションサーバに作成

1. Oracle Application Server Control url に行きます。
2. systinet.war ファイルを配備する場所の OC4J インスタンスを選択します。
3. ‘Administration’ リンクを選択します。
4. ‘Shared Libraries’ の ‘Go to task’ を選択します。
5. ‘Create’ ボタンをクリックします。
6. Shared Library Name: wasp_libraries
   Shared Library version: 1
   ‘Next’ をクリックします。
7. ‘Add’ ボタンをクリックして、security-ng.jar をブラウズします。

   注 : 場所 :<UNIFY_HOME>/NXJ/wasp/lib
8. ‘Add’ ボタンをクリックして、nxjwaspjaas.jar をブラウズします。

   注 : 場所 :<UNIFY_HOME>/NXJ/lib
9. これら jar の両方は、一度加えられて、‘Continue’ をクリックして、‘Next’ をクリックして、‘Finish’ をクリックします。

5.3.3 systinet.war ファイルの配備

1. Oracle Application Server Control url に行きます。

   注 : J2eeIntegrate ユーティリティを実行しているマシンのブラウザを起動します。
2. 配備したい OC4J インスタンスを選択します。
3. ‘Application’ リンクをクリックします。
4. Deploy ボタンをクリックして、systinet.war をブラウズします。

   注 : ディレクトリの場所 :<UNIFY_HOME>NXJ/wasp/config/porting/oralce/build
5. ‘Next’ をクリックします。
6. Application = systinet
7. ‘Next’ をクリックします。

8. ‘Configuring Class Loading’の‘Go to task’リンクを選択します。
9. wasp_libraries の import チェックボックスをチェックします。
10. ‘OK’をクリックします。
11. ‘Deploy’をクリックします。

5.3.4 Oracle 10g アプリケーションサーバファイルの更新

1. OC4J_HOME/webservices/lib ディレクトリからファイル orawsdl.jar を削除します。
2. OC4J_HOME/j2ee/ov4j_inst1/config/system-jazndata.xml ファイルへの jazn-loginconfig サブ要素の下に、<UNIFY_HOME>/wasp/conf/porting/jazn.data の内容を追加します。

5.3.5 NXJ ActiveSOA java オプションの追加

1. Oracle Application Server Control url に行きます。
2. systinet.war ファイルを配備する場所の OC4J インスタンスを選択します。
3. ‘Application’リンクをクリックします。
4. ‘Server Properties’の‘Go to task’リンクを選択します。
5. Command Line オプションの下の Start-parameters:Java Options で、‘Add Another Row’をクリックします。

   以下を入力します。

   -XX:MaxPermSize=128M
   -XX:AppendRatio=3
   -Djava.security.policy=/opt/oracleas_10.1.3/j2ee/oc4j_inst1/config/java2.policy
   -Djava.awt.headless=true
   -Dhttp.webdir.enable=false
   -Djava.security.auth.login.config=/opt/oracleas_10.1.3/j2ee/oc4j_inst1/applications/systinet/conf/jaas.config
   -Djava.naming.security.principal=admin
   -Djava.naming.security.credentials=changeit
   -Djavax.xml.soap.MessageFactory=com.systinet.saaj.soap.DefaultMessageFactory
   -Djavax.xml.soap.SOAPFactory=com.systinet.saaj.soap.SOAPFactoryImpl

   注： 上記のエントリのそれぞれに“Add Another Row”を行います。

5.3.6 Oracle 10g アプリケーションサーバを開始して、NXJ ActiveSOA 管理コンソールにアクセス

oracle 10g アプリケーションサーバを起動して、http: //<host>: <port>/systinet/server/admin/console にアクセスします。

注 : oracle 10g アプリケーションサーバと同じマシン上のブラウザを起動している場合、この URL は通常以下の通りです。
http://localhost:7777/systinet/server/admin/console

6 NXJ ActiveWorkflow - アプリケーションサーバの構成

6.1 NXJBPM.ear ファイルの構成

NXJ は、NXJ ActiveWorkflow ear ファイルを構成するために、“nxjbpmconfig.exe”と呼ばれるツールを提供します。このツールは、<UNIFY_HOME>/bin ディレクトリの下で、NXJ Enterprise Developer と NXJ Enterprise Server マシンの両方にあります。特に NXJ Enterprise Developer と NXJ Enterprise Server が異なるマシンにインストールされている場合、NXJ Enterprise Server マシンのこのツールを使用することを推奨します。一旦、開始されたならば、以下のステップを行います。

6.1.1 NXJ ActiveWorkflow ear ファイルを開く

File | Open を選択して、<UNIFY_HOME>/lib ディレクトリに移動します。
NXJBPM.ear ファイルを開きます。

6.1.2 Portal.properties の構成

左側のナビゲーションウィンドウで Portal.properties を選択します。以下の行を変更して保存します。

unify.home:
unify.work:
attachments.ds.name: java:/NXJDS
audit.ds.name: java:/NXJDS

注： JBoss の場合、java: /NXJDS（例　default）を使用します。JBoss 以外のアプリケーションサーバの場合、jdbc/NXJDS を使用します。

注： 一度、上記のプロパティが見つけられたら、関連するコメントを読んで、それらがどのように構成されるべきかを確かめてください。セットアップに適切な、他の構成があるかもしれないので、他の全てのコメントを読んでください。

6.1.3 hibernate.cfg.xml の構成

左側のナビゲーションウィンドウで hibernate.cfg.xml を選択します。以下の行を変更して保存します。

<property name=”dialect”>net.sf.hibernate.dialect.SQLBaseDialect</property>

<property name=”hibernate.transaction.manager_lookup_class”>
net.sf.hibernate.transaction.JBossTransactionManagerLookup</property>

<property name=”hibernate.jdbc.use_scrollable_resultset”>false</property>

<mapping resource=”SQLBase/AuthToken.hbm.xml”/>
<mapping resource=”SQLBase/HibACE.hbm.xml”/>
<mapping resource=”SQLBase/HibConditionalChange.hbm.xml”/>
<mapping resource=”SQLBase/HibMediationObject.hbm.xml”/>
<mapping resource=”SQLBase/HibOperand.hbm.xml”/>
<mapping resource=”SQLBase/HibOperandACE.hbm.xml”/>
<mapping resource=”SQLBase/HibPolicy.hbm.xml”/>
<mapping resource=”SQLBase/HibPolicyACE.hbm.xml”/>
<mapping resource=”SQLBase/HibProcess.hbm.xml”/>
<mapping resource=”SQLBase/HibProcessACE.hbm.xml”/>
<mapping resource=”SQLBase/HibBPMEvent.hbm.xml”/>
<mapping resource=”SQLBase/HibStatus.hbm.xml”/>
<mapping resource=”SQLBase/HibStatusACE.hbm.xml”/>
<mapping resource=”SQLBase/HibTimerSpec.hbm.xml”/>
<mapping resource=”SQLBase/Persistent.hbm.xml”/>

注： NXJ は、デフォルトデータベースリポジトリとして、SQLBase を仮定します。他のデータベースにする場合は、“SQLBase”というサブストリングを適切なデータベース名に変更してください。以下のデータベースがサポートされます。

1. “Pointbase”
2. “Oracle”
3. “MSSQLServer”
4. “Informix”
5. “DB2”
6. “Cloudscape”
7. “SQLBase”

8. “MySQL”

注 : 一度、上記のプロパティが見つけられたら、関連するコメントを読んで、それらがどのように構成されるべきかを確かめてください。セットアップに適切な、他の構成があるかもしれないので、他の全てのコメントを読んでください。

6.1.4 quartz.properties の構成

左側のナビゲーションウィンドウで quartz.properties を選択します。以下の行を変更して保存します。

org.quartz.dataSource.NXJDS.jndiURL= java:/NXJDS
org.quartz.dataSource.NXJDSUnmanaged.driver=jdbc.gupta.sqlbase.SqlbaseDriver
org.quartz.dataSource.NXJDSUnmanaged.URL=jdbc:sqlbase://localhost:2155/REPOSITORY
org.quartz.dataSource.NXJDSUnmanaged.user=SYSADM
org.quartz.dataSource.NXJDSUnmanaged.password=SYSADM

注 : JBoss の場合、java: /NXJDS（例　デフォルト）を使用します。JBoss 以外のアプリケーションサーバの場合、jdbc/NXJDS を使用します。

注 : NXJ は、デフォルトデータベースリポジトリとして、SQLBase を仮定します。他のデータベースにする場合は、“SQLBase”というサブストリングを適切なデータベース名に変更してください。

注 : 一度、上記のプロパティが見つけられたら、関連するコメントを読んで、それらがどのように構成されるべきかを確かめてください。セットアップに適切な、他の構成があるかもしれないので、他の全てのコメントを読んでください。

6.2 JBoss - ActiveWorkflow の構成

6.2.1 アプリケーションサーバの startup スクリプトの編集

Windows では、これは run.bat で、UNIX ベースのシステムでは、run.sh です。このファイルは、<UNIFY_HOME>/NXJ/jboss/bin ディレクトリの下にあります。以下の行を追加します。

run.bat
set UNIFY_HOME=C:\Unify\NXJ
set UNIFY_WORK=C:\Unify\NXJWork

注 : 上記の編集は、JBoss の Winodws インストレーション用で、C: \Unify\ の下の NXJ Enterprise Server にインストールされます。

run.sh
UNIFY_HOME=/opt/Unify/NXJ
UNIFY_WORK=/opt/Unify/NXJWork

注 : 上記の編集は、JBoss の Linux インストレーション用で、/opt/Unify の下の NXJ Enterprise Server にインストールされます。

6.2.2 NXJ リポジトリデータソースの設定

第 2 章の「NXJ リポジトリの構成」を参照してください。

6.2.3 ‘jboss-service.xml’ ファイルの構成

<JBOSS_HOME>/server/default/deploy/jbossservice.xml ファイルの ‘CallByValue’ 属性を true に設定します。

6.2.4 ‘ear-deployer.xml’ ファイルの構成

<JBOSS_HOME>/server/default/ deploy/ear-deployer.xml ファイルの ‘CallByValue’ 属性を true に設定します。

6.2.5 NXJBPM.ear ファイルの配備

jboss deploy ディレクトリに NXJBPM.ear ファイルをコピーします。

注 : NXJ Enterprise Server install から NXJBPM.ear ファイルを取得する必要があります。<UNIFY_HOME>/lib ディレクトリの下に格納されています。

6.3 WebSphere - ActiveWorkflow の構成

これは、NXJ12 リリースで動作保証されていません。

6.4 Weblogic - ActiveWorkflow の構成

6.4.1 JMS Connection factory の作成

1. General フォームで、以下を入力して **Create** をクリックします。

   Name: ConnectionFactory
   JNDI Name: ConnectionFactory

2. Trasaction フォームで、**XA Connection Factory Enabled** をチェックします。
3. Target and Deploy フォームで、配備するサーバをチェックして **Apply** をクリックします。

6.4.2 JMS Destination Queue の作成

1. General フォームで、以下を入力して **Create** をクリックします。

   Name: BPMQueue
   JNDI Name: queue/BPMMEvents

6.4.3 BPM commons-logging.jar を CLASSPATH 上に置く

Windows では setDomainEnv.cmd を、UNIX ベースのシステムでは、setDomainEnv.sh を選択します。このファイルは、<WL_HOME>/bin ディレクトリの下に格納することができます。以下の行を追加します。

setDomainEnv.bat
PRE_CLASSPATH=%PRE_CLASSPATH%;%UNIFY_HOME%\BPM\lib\commons-logging.jar

setDomainEnv.sh
PRE_CLASSPATH=”$PRE_CLASSPATH:$UNIFY_HOME/BPM/lib/commons-logging.jar”

6.4.4 NXJBPM.ear ファイルの配備

標準の weblogic ear 配備説明に従って、NXJBPM.ear ファイルを配備します。

6.5 Oracle 10g - ActiveWorkflow の構成

6.5.1 NXJ リポジトリデータソースの設定

第 2 章の「NXJ リポジトリの構成」を参照してください。

6.5.2 global jndi lookup の有効

1. ファイル <ORACLE_HOME>/j2ee/oc4j_inst*/config/server.xml を開きます。
2. テキストの以下の行を追加します。

   global-jndi-lookup-enabled=”true” - server.xml

6.5.3 BPM Events Queue の JMS destination の作成

1. Oracle Application Server Control に行きます。
2. BPMAdmin サーバを配備する OC4J インスタンスを選択します。
3. Administration リンクを選択します。
4. JMS Destinations の“Go to Task”を選択します。
5. New ボタンをクリックして、以下のように構成します。

   Destination Name: BPM Events Queue
   In Memory Persistence Only.
   JNDI Location: queue/BPMEvents.

6. OK をクリックします。

6.5.4 shared library の作成

1. Oracle Application Server Control に行きます。
2. BPMAdmin サーバを配備する OC4J インスタンスを選択します。
3. Administration リンクを選択します。
4. shared libraries の“Go to Task”を選択します。
5. Create ボタンをクリックします。
6. 以下のように構成します。

   Shared Library Name: BPM_Libraries
   Shared Library version: 1

7. Next をクリックします。
8. Add ボタンをクリックします。**commons-logging.jar** をブラウズします。

   注 : NXJ 開発またはサーバマシンの場所 :<UNIFY_HOME>/NXJ/BPM/lib
9. **log4j-1.2.6.jar** のためにステップ 8 を繰り返します。

   注 : NXJ 開発またはサーバマシンの場所 :<UNIFY_HOME>/NXJ/BPM/lib
10. Continue > Next > Finish をクリックします。

    注 : NXJActiveWorkflow アプリケーションを配備する場合は、shared library をインポートする必要があります。

6.5.5 NXJBPM.ear ファイルの配備

1. Oracle Application Server Control に行きます。
2. OC4J インスタンスを選択します。
3. Application リンクを選択します。
4. Deploy ボタンをクリックします。
5. Browse ボタンをクリックして、例えば、<UNIF_HOME>/NXJ/BPM/lib に格納される NXJBPM.ear を選択します。

   注 : NXJ Enterprise Server をインストールから NXJBPM.ear ファイルを使用する必要があります。
6. Continue をクリックします。
7. Next をクリックします。
8. Application Name に、“BPMAdmin”を設定して、‘Next’をクリックします。
9. “Go To Task”の下の pencil リンクをクリックすることによって、環境リファレンスをマップします。
   以下のように構成します。

   “NXJDS”の全てのリファレンスを“jdbc/NXJDS”に変更することによって、JNDI Location をマップします。

   OK をクリックします。
10. “Go To Task”の下の pencil リンクをクリックすることによって、class loading を構成します。
    以下のように構成します。

    BPM_Libraries の import チェックボックスをチェックします。

    “Add Another Row”をクリックして、パスとして <UNIFY_HOME>/BPM/samples/portal/lib を入力します。

    OK をクリックします。

11. advanced deployment plan Editing の下で、“Edit Deployment Plan”を選択します。
    以下のように構成します。

    左側のウィンドウで、BPM Process Manager > ejb > BPMServerEJB > Session > AdminService > env-entry を選択します。

    UNIFY_HOME と UNIFY_WORK の値が正しいことを確認します。
    ‘OK’をクリックして、‘Deploy’をクリックします。

7 NXJ ActiveReporting - アプリケーションサーバの構成

このセクションは、以下のアプリケーションサーバにおける、NXJ ActiveReporting のアプリケーションサーバの構成のガイドラインです。

- JBoss アプリケーションサーバ
- IBM WebSphere アプリケーションサーバ
- WebLogic アプリケーションサーバ
- Oracle 10g アプリケーションサーバ

7.1 JBoss - NXJ ActiveReporting の構成

NXJ Enterprise Server に行き、JBoss deployment ディレクトリに jreport.ear ファイルをコピーします。

注 : jreport.ear ファイルは、<UNIFY_HOME>/NXJ/lib ディレクトリの下に格納することができます。

注 : 一旦、配備されると、http://<host>:<port>/jreport（例 : http://qaes4.sac.unify.com:8080/jreport）にアクセスすることによって、report サーバにアクセスすることができます。

注 : Report server work ディレクトリは、通常、Windows の場合、c:\Document and Settings\<user>\.jreport にあり、UNIX ベースのマシンの場合、/home/<user>/.jreport にあります。このディレクトリを変更したい場合は、アプリケーションサーバの以下の java オプションを設定する必要があります。

Windows の場合
set JAVA_OPTS=%JAVA_OPTS% -Dreporthome=c:\report_work

UNIX の場合
JAVA_OPTS=”$JAVA_OPTS -Dreporthome=/opt/report_work”

7.2 WebSphere - NXJ ActiveReporting の構成

標準 websphere エンタープライズアプリケーションの配備説明に従って、jreport ear ファイルを配備します。

注 : jreport.ear ファイルは、<UNIFY_HOME>/NXJ/lib の下に格納することができます。

注 : 一旦、配備されると、http://<host>:<port>/jreport（例 : http://qaes4.sac.unify.com:9080/jreport）にアクセスすることによって、report サーバにアクセスすることができます。

注 : Report server work ディレクトリは、通常、Windows の場合、c:\Document and Settings\<user>\.jreport にあり、UNIX ベースのマシンの場合、/home/<user>/.jreport にあります。このディレクトリを変更したい場合は、アプリケーションサーバの以下の java オプションを設定する必要があります。

Windows の場合
set JAVA_OPTS=%JAVA_OPTS% -Dreporthome=c:\report_work

UNIX の場合
JAVA_OPTS=”$JAVA_OPTS -Dreporthome=/opt/report_work”

7.3 Weblogic - NXJ ActiveReporting の構成

標準 weblogic エンタープライズアプリケーションの配備説明に従って、jreport war ファイルを配備します。

注 : jreport.ear ファイルは、<UNIFY_HOME>/NXJ/lib の下に格納することができます。

注 : 一旦、配備されると、http://<host>:<port>/jreport（例 : http://qaes4.sac.unify.com:7001/jreport）にアクセスすることによって、report サーバにアクセスすることができます。

注 : Report server work ディレクトリは、通常、Windows の場合、c:\Document and Settings\<user>\.jreport にあり、UNIX ベースのマシンの場合、/home/<user>/.jreport にあります。このディレクトリを変更したい場合は、アプリケーションサーバの以下の java オプションを設定する必要があります。

Windows の場合
set JAVA_OPTS=%JAVA_OPTS% -Dreporthome=c:\report_work

UNIX の場合
JAVA_OPTS=”$JAVA_OPTS -Dreporthome=/opt/report_work”

7.4 Oracle 10g - NXJ ActiveReporting の構成

標準 oracle 10g エンタープライズアプリケーションの配備説明に従って、jreport war ファイルを配備します。

注 : jreport.ear ファイルは、<UNIFY_HOME>/NXJ/lib の下に格納することができます。

注 : 一旦、配備されると、http://<host>:<port>/jreport（例 : http://qaes4.sac.unify.com:8888/jreport）にアクセスすることによって、report サーバにアクセスすることができます。

注 : Report server work ディレクトリは、通常、Windows の場合、c:\Document and Settings\<user>\.jreport にあり、UNIX ベースのマシンの場合、/home/<user>/.jreport にあります。このディレクトリを変更したい場合は、アプリケーションサーバの以下の java オプションを設定する必要があります。

Windows の場合
set JAVA_OPTS=%JAVA_OPTS% -Dreporthome=c:\report_work

UNIX の場合
JAVA_OPTS=”$JAVA_OPTS -Dreporthome=/opt/report_work”