管理コンソール
ユーザーズガイド

Borland AppServer™ 6.6

Borland®

Borland Software Corporation
20450 Stevens Creek Blvd., Suite 800
Cupertino, CA 95014 USA
www.borland.com

2006 年 6 月 8 日

著者：Borland Software Corporation

発行：ボーランド株式会社

PDF

目次

第 1 章

Borland AppServer の概要

Borland AppServer（AppServer）は，企業環境で分散エンタープライズアプリケーションを構築，配布，および管理するためのサービスとツールのセットです。

AppServer は，J2EE 1.4 標準の最新のインプリメンテーションとして，EJB 2.1，JMS 1.1，Servlet 2.4，JSP 2.0，CORBA 2.6，XML，SOAP などの最新の業界標準をサポートします。Borland は 2 つのバージョンの AppServer を提供しています。これには，Java メッセージサービス（JMS）管理用の最先端のエンタープライズメッセージングソリューション（Tibco と OpenJMS）が含まれます。AppServer で必要な機能とサービスのレベルを選択できます。また，必要であれば，ライセンスのアップグレードも簡単に行うことができます。詳細は『インストールガイド』の「第 3 章 Borland AppServer の Windows へのインストール」，または「第 4 章 Borland AppServer の Solaris または HP-UX へのインストール」を参照してください。

AppServer を使用すると，J2EE 1.4 プラットフォーム標準を実装する分散 Java および CORBA アプリケーションを安全に配布し，そのすべての面を管理できます。

AppServer では，インストールごとのサーバーインスタンスの数は無制限です。そのため，同時接続ユーザーの数は無制限です。

AppServer は次のコンポーネントを備えています。

- J2EE 1.4 のインプリメンテーション
- Apache Web サーバーバージョン 2.2
- Borland Security（AppServer のセキュリティを確保するフレームワーク）
- AppServer に付属する最新の集中管理型 JMS 管理ソリューション（Tibco と OpenJMS）
- 分散コンポーネント（AppServer の外部で開発されたアプリケーションを含む）の強力な管理ツール

AppServer の機能

AppServer には次の機能があります。

- BAS プラットフォームをサポートします。AppServer でサポートされるプラットフォームのリストについては，http://support.borland.com/kbcategory.jspa?categoryID=389 を参照してください。
- クラスタリングトポロジーを完全にサポートします。
- VisiBroker ORB インフラストラクチャとシームレスに統合されます。
- Borland JBuilder 統合開発環境と統合されます。
- 他の Borland 製品（Borland Together ControlCenter, Borland Optimizeit Profiler および ServerTrace など）と幅広く統合されます。
- AppServer を使用して，既存のアプリケーションを Web サービスとして公開したり，新しいアプリケーションや追加の Web サービスと統合することができます。Borland Web サービスサポートは，Apache Axis 1.2 テクノロジー（SOAP 1.2 をサポートする次世代 Apache SOAP サーバー）に基づきます。

Borland AppServer のマニュアル

AppServer のマニュアルセットは次のマニュアルで構成されています。

- *Borland AppServer インストールガイド* — AppServer をネットワークにインストールする方法について説明します。このマニュアルは，Windows または UNIX オペレーティングシステムに精通しているシステム管理者を対象としています。
- *Borland AppServer 開発者ガイド* — 各動作環境における分散オブジェクトベースアプリケーションのパッケージング，配布，および管理の詳細が記載されています。
- *Borland 管理コンソールユーザーズガイド* — Borland 管理コンソール GUI の使い方が記載されています。
- *Borland セキュリティガイド* — VisiSecure for VisiBroker for Java および VisiBroker for C++ など，AppServer のセキュリティを確保するための Borland のフレームワークについて説明しています。
- *Borland VisiBroker for Java 開発者ガイド* — Java による VisiBroker アプリケーションの開発方法について記載されています。Visibroker ORB の設定と管理，およびプログラミングツールの使用方法について説明します。また，IDL コンパイラ，スマートエージェント，ロケーションサービス，ネーミングサービス，イベントサービス，オブジェクトアクティベーションデーモン（OAD），Quality of Service（QoS），およびインターフェースリポジトリについても説明します。
- *Borland VisiBroker VisiTransact ガイド* — Borland による OMG Object Transaction Service 仕様のインプリメンテーションおよび Borland Integrated Transaction Service コンポーネントについて説明します。

通常，マニュアルにアクセスするには，AppServer 製品とともにインストールされるヘルプビューアを使用します。ヘルプは，スタンドアロンのヘルプビューアからアクセスすることも，AppServer GUI ツールからアクセスすることもできます。どちらの場合も，ヘルプビューアを起動すると独立したウィンドウが表示されるため，このウィンドウからヘルプビューアのメインツールバーにアクセスしてナビゲーションや印刷を行ったり，ナビゲーションペインにアクセスすることができます。ヘルプビューアのナビゲーションペインには，すべての AppServer ブックとリファレンス文書の目次，完全なインデックス，および包括的な検索を実行できるページがあります。

PDF 形式の『Borland AppServer 開発者ガイド』と『Borland 管理コンソールユーザーズガイド』は，http://info.borland.com/techpubs/appserver からオンラインで入手できます。

スタンドアロンヘルプビューアからの AppServer オンラインヘルプトピックへのアクセス

製品がインストールされているコンピュータでスタンドアロンのヘルプビューアからオンラインヘルプにアクセスするには，次のいずれかの手順を実行します。

Windows

- ［スタート｜すべてのプログラム｜ Borland AppServer ｜ Help Topics］の順に選択します。
- または，コマンドプロンプトを開き，製品のインストールディレクトリの \bin ディレクトリに移動し，次のコマンドを入力します。

```
help
```

UNIX コマンドシェルを開き，製品のインストールディレクトリの /bin ディレクトリに移動し，次のコマンドを入力します。

```
help
```

ヒント UNIX システムにインストールするときの指定で，PATH エントリのデフォルトに bin を含まないようにします。カスタムインストールオプションを選択して PATH エントリのデフォルトを変更せず，PATH に現在のディレクトリのエントリがない場合は，./help を使用してヘルプビューアを起動できます。

AppServer GUI ツール内からの AppServer オンラインヘルプトピックへのアクセス

AppServer GUI ツール内からオンラインヘルプにアクセスするには，次のいずれかの方法を使用します。

- Borland 管理コンソールで［Help ｜ Help Topics］を選択します。
- Borland Deployment Descriptor Editor（DDEditor）で［Help ｜ Help Topics］を選択します。

［Help］メニューには，オンラインヘルプ内のいくつかの文書へのショートカットもあります。ショートカットの 1 つを選択すると，ヘルプトピックビューアが起動し，［Help］メニューで選択した項目が表示されます。

マニュアルの表記規則

AppServer のマニュアルでは，文中の特定の部分を表すために，次の表に示す書体と記号を使用します。

表記規則	用途
italic	新規の用語およびマニュアル名に使用されます。
`computer`	ユーザーやアプリケーションが提供する情報，サンプルコマンドライン，およびコードです。
`bold computer`	本文では，ユーザーが入力する情報を示します。サンプルコードでは，重要なステートメントを強調表示します。
[ ]	省略可能な項目。
...	繰り返しが可能な直前の引数。
|	二者択一の選択。

プラットフォームの表記

AppServer マニュアルでは，次の記号を使用してプラットフォーム固有の情報を示します。

記号	意味
Windows	サポートされているすべての Windows プラットフォーム
Win2003	Windows 2003 のみ
WinXP	Windows XP のみ
Win2000	Windows 2000 のみ
UNIX	すべての UNIX プラットフォーム
Solaris	Solaris のみ

Borland サポートへの連絡

ボーランド社は各種のサポートオプションを用意しています。それらにはインターネット上の無償サービスが含まれており，大規模な情報ベースを検索したり，他の Borland 製品ユーザーからの情報を得ることができます。さらに Borland 製品のインストールに関するサポートから有償のコンサルタントレベルのサポートおよび高レベルなアシスタンスに至るまでの複数のカテゴリから，電話サポートの種類を選択できます。

Borland のサポートサービスの詳細や Borland テクニカルサポートへの問い合わせについては，Web サイト `http://support.borland.com` で地域を選択してください。

ボーランド社のサポートへの連絡にあたっては，次の情報を用意してください。

- 名前
- 会社名およびサイト ID
- 電話番号
- ユーザー ID 番号（米国のみ）
- オペレーティングシステムおよびバージョン
- Borland 製品名およびバージョン
- 適用済みのパッチまたはサービスパック
- クライアントの言語とそのバージョン（使用している場合）
- データベースとそのバージョン（使用している場合）
- 発生した問題の詳細な内容と経緯
- 問題を示すログファイル
- 発生したエラーメッセージまたは例外の詳細な内容

オンラインリソース

ネットワーク上の次のサイトから情報を得ることができます。

ワールドワイドウェブ： http://www.borland.com/jp/

オンラインサポート： http://support.borland.com（ユーザー ID が必要）

Web サイト

定期的に http://www.borland.com/jp/products/appserver/index.html をチェックしてください。AppServer 製品チームによるホワイトペーパー，競合製品の分析，FAQ の回答，サンプルアプリケーション，最新ソフトウェア，最新のマニュアル，および新旧製品に関する情報が掲載されます。

特に，次の URL をチェックすることをお勧めします。

- http://www.borland.com/downloads/download_appserver.html（AppServer ソフトウェアおよび他のファイル）
- http://support.borland.com（AppServer の FAQ）

Borland ニュースグループ

AppServer を対象とした数多くのニュースグループに参加できます。Enterprise Server などの Borland 製品のユーザーによるニュースグループへの参加については，http://www.borland.com/newsgroups を参照してください。

メモ これらのニュースグループはユーザーによって管理されているものであり，ボーランド社の公式サイトではありません。

第 2 章

Borland 管理コンソールの使い方

ここでは，Borland 管理コンソールの概要，起動方法，機能などについて説明します。

Borland 管理コンソールの概要

Borland 管理コンソールは，ネットワーク上の管理ハブや設定の表示，ハブや管理オブジェクトの起動と終了，設定の編集，コンポーネントベースのエンタープライズアプリケーションを構築，配布，管理するサービスやツールの管理を行う操作の中心として機能するグラフィカルユーザーインターフェースです。また，配布されたモジュールの表示，配布プロパティの設定，およびパフォーマンスの監視を行うこともできます。管理コンソールは，任意のマシンで実行して，分散システムの表示や変更に使用できます。

Borland 管理コンソールの起動とログイン

管理コンソールを起動するには，次の手順にしたがいます。

1 管理コンソールを起動するには，次のいずれかの方法を使用します。

Windows ［スタート］メニューから［すべてのプログラム｜ Borland AppServer ｜ Management Console］を選択します。
または，コマンドプロンプトを起動し，次のコマンドを入力します。

```
console
```

メモ console コマンドを使用するには，パスシステム変数に <install_dir>¥bin ディレクトリが含まれている必要があります。または，パスを明示的に入力します。

UNIX コマンドシェルを開き，次のコマンドを入力します。

```
console
```

メモ console コマンドを使用するには，パスシステム変数に <install_dir>/bin ディレクトリが含まれている必要があります。または，パスを明示的に入力します。

［Administration Security Credentials］画面が表示されます。

図 2.1　デフォルトの認証情報が設定された［Administration Security Credentials］画面

2 ［User］，［Password］，および［Realm］を入力します。
Borland のデフォルトの認証情報設定を使用する場合は，次のパスワードを入力します。

```
admin
```

3 ［OK］をクリックします。
管理コンソールが表示されます。

管理コンソールのビュー

［Management Console］ウィンドウの左端の列のボタンを使用して，次のビューにアクセスします。

- ［Hubs］- エージェントと設定にアクセスできます。特に，Borland AppServer（AppServer）設定から AppServer パーティションの使い方，Borland AppServer サービスの使い方，および J2EE コンポーネントの設定の表示にアクセスできます。
- ［Installations］- ローカルマシンへのインストールを操作するビューです。
- ［VisiBroker］- VisiBroker コンソールを開きます。

管理コンソールのメニュー

管理コンソールには，次のメインメニューがあります。

- ［Console］
- ［View］
- ［Wizards］
- ［Tools］
- ［Help］

[Console] メニュー

次の表では，[Console] メニューのコマンドについて説明します。

表 2.1 [Console] メニューのコマンド

[Console] メニューのコマンド	説明
Refresh	管理コンソールに表示される状態情報を手動で更新します。
Preferences	管理コンソールの設定や環境設定に使用する [Preferences] ダイアログボックスを表示します。このダイアログでの環境設定については，11 ページの「管理コンソールの基本設定」を参照してください。
Set Identity	ハブやエージェントを検出する際に認証で使用される認証情報（ユーザー ID，パスワード，領域）を設定します。
Exit	管理コンソールを終了します。

[View] メニュー

次の表では，[View] メニューのコマンドについて説明します。

表 2.2 [View] メニューのコマンド

[View] メニューのコマンド	説明
Messages	エラーウィンドウを表示／非表示にします。
Tool Bar	[Management Console] ウィンドウの上部にあるツールバーを表示／非表示にします。
Status Bar	[Management Console] ウィンドウの下部にあるステータスバーを表示／非表示にします。

[Wizards] メニュー

次の表では，[Wizards] メニューのコマンドについて説明します。

表 2.3 [Wizards] メニューのコマンド

[Wizards] メニューのコマンド	説明
Deployment Wizard	J2EE モジュールを配布するためのウィザードを起動します。
Merge Wizard	複数の J2EE モジュールを 1 つのモジュールにマージするためのウィザードを起動します。
Verify Wizard	アーカイブファイルの正当性と整合性を確認し，アプリケーションの配布に必要な要素がすべて所定の位置にあるかどうかを確認するためのウィザードを起動します。
XML Migration Wizard	J2EE バージョン 1.2 からバージョン 1.3 に，または J2EE バージョン 1.3 からバージョン 1.2 にモジュールを変換するためのウィザードを起動します。
Stub Generation Wizard	さまざまな種類のアーカイブに保存できるスタブクラスを生成するためのウィザードを起動します。
Remove Stubs Wizard	アーカイブからスタブクラスを削除するためのウィザードを起動します。
Apply Patch Wizard	アーカイブに 1 つ以上のパッチを適用するためのウィザードを起動します。
Jar Wizard	JAR ファイル内のオブジェクトを圧縮または抽出するためのウィザードを起動します。

[Tools] メニュー

次の表では，[Tools] メニューのコマンドについて説明します。

表 2.4　[Tools] メニューのコマンド

[Tools] メニューのコマンド	説明
Archive Tool	アーカイブツールを起動します。詳細については，第 11 章「アーカイブツールの使い方」を参照してください。
Borland Log Tool	Borland ログツールを開きます。
Deployment Descriptor Editor	DDEditor を開きます。詳細については，第 9 章「配布デスクリプタエディタの使い方」を参照してください。
JDataStore Explorer	JDataStore エクスプローラを起動します。詳細については，http://info.borland.com/techpubs/jdatastore/ の JDataStore ドキュメントを参照してください。
Launcher Generation Wizard	起動プログラム生成ウィザードを起動します。
License Manager	ライセンスマネージャを起動します。詳細については，第 12 章「ライセンスマネージャの使い方」を参照してください。
Optimizeit Profiler	管理コンソールの基本設定に設定している場合は，Optimizeit Profiler ビューアを起動します。Optimizeit のクライアント設定については，13 ページの「Tools」を参照してください。
Optimizeit ServerTrace	管理コンソールの基本設定に設定している場合は，Optimizeit ServerTrace ビューアを起動します。Optimizeit のクライアント設定については，13 ページの「Tools」を参照してください。
TCP Monitor	TCP モニタを起動します。
Tibco Admin Console	Tibco Management Console を起動します。JMS の詳細については，37 ページの「Borland AppServer サービスの使い方」を参照してください。
Chainsaw Log4J Browser	Chainsaw Log4J ブラウザを起動します。
JNDI Browser	JNDI ブラウザを起動します。詳細については，第 10 章「JNDI ブラウザの使い方」を参照してください。
Web Services Explorer	Web サービスエクスプローラを起動します。
VisiBroker Properties File Editor	VisiBroker プロパティファイルを編集するためのエディタを開きます。

[Help] メニュー

このメニューでは，オンラインヘルプの各文書，AppServer の開発者サポート Web サイト，および Borland 管理コンソールのバージョン情報画面にアクセスできます。

[About] 画面には，Borland ソフトウェアの重要情報を表示するタブがあります。

バージョン情報画面のタブ	説明
About	Borland Deployment Platform のバージョン番号と著作権情報を表示します。
General System Information	AppServer が検出した，オペレーティングシステム，Java バージョン，Java ベンダー，Java コンパイラなどのシステム環境設定を表示します。
Java Properties	AppServer が使用している Java 仮想マシンのプロパティ設定を表示します。

管理コンソールの基本設定

管理コンソールの基本設定では, SmartAgent ポートや, コンソールに表示されるパフォーマンス情報のデフォルトのポーリング間隔など，管理コンソールによって使用される設定，操作，および表示のオプションを指定できます。

管理コンソールの基本設定を行うには，次の手順にしたがいます。

1 ［Management Console］を起動し，［Console］メニューの［Preferences］を選択します。［Preferences］ダイアログボックスに，基本設定に必要なオプションを含む次のカテゴリが表示されます。

- ［General］: ユーザーインターフェースのオプションと設定を制御します。
- ［Security］: システム管理とセキュリティ環境の管理に関連するプロパティを提供します。
- ［Discovery］: ツリーに表示されるハブを指定します。
- ［State］: 状態情報の更新を設定します。
- ［Logs］: ログファイルと監査ファイルの内容の更新と表示を設定します。
- ［Tools］: 管理コンソールによって起動されるツール（Optimizeit など）の設定情報です。
- ［VisiBroker］: VisiBroker コンソールのユーザーインターフェースオプションを設定します。

2 タブを選択して移動し，必要に応じて設定値を変更します。

3 設定が完了したら，［OK］をクリックします。

以下の節で，環境設定の詳細について説明します。

General

［General］タブでは，管理コンソールのさまざまなユーザーインターフェース要素を設定します。このタブには，次のオプションがあります。

- **［Look and feel］:** コンソールウィンドウの表示形式と動作を設定します。選択できるオプションは, Metal, CDE/Motif, Windows（Microsoft Windows プラットフォームのみ），Windows Classic（Microsoft Windows プラットフォームのみ）および Borland です。
- **［Tab memory］:** 管理コンソールで使用する表示の状態情報を指定します。次のオプションがあります。
  - **［Don't remember last visited tab pane］:** 右側に［General］タブを表示して，ツリーの各ノードを開くように管理コンソールに指示します。
  - **［Remember last visited tab pane by type］:** 最後に開いたタブと同じ種類のタブ（同様のノード）でノードを開くように管理コンソールに指示します。たとえば，［Properties］タブが表示された状態で，［Properties］タブを持つ別のノードをクリックすると，管理コンソールはまずそのノードの［Properties］タブを表示します。
  - **［Remember last visited tab pane by type and name］:** 先のコンソールセッションでノードを選択したときに, 最後に開いたタブに展開されたそのノードを開くように管理コンソールに指示します。このノードは，ノードが選択されたときに，最後に表示されていたタブに展開されます。
- **［Sound beep on errors］:** 管理コンソールでエラーが発生すると，アラームが鳴ります。
- **［Enable debug output］:** 管理コンソールの下部にある［Errors］ペインでデバッグ情報を報告するように管理コンソールに指示します。

- **［HTML Browser Setup］:** 管理コンソールで使用する Web ブラウザのパスを指定します。インストールされている Web ブラウザを検索するには，［Browse］ボタンを使用します。

Security

［Security］ダブでは，システム管理とセキュリティ環境の管理に関連する設定を提供します。このタブには，次のオプションがあります。

- **［Default realm］:** 各 AppServer と対話するために，コンソールで使用する認証領域の名前を指定します。
- **［Default user］:** 各 AppServer と対話するために，コンソールで使用するユーザー名を指定します。
- **［Enable Security］:** 管理コンソールでのセキュリティの処理方法を指定します。
  - サーバーのセキュリティがオンになっているかどうかにかかわらず，管理コンソールはエージェントと通信できます。ただし，セキュリティがオンになっているエージェントから要求を受信する場合，管理コンソールは，認証のためにユーザーのログイン認証情報（領域，ユーザー名，パスワード）をエージェントに渡した後に，そのエージェントのサービスにアクセスできます。
  - このチェックをはずすと，管理コンソールは，セキュリティがオンになっていないサーバーとだけ通信します。

Discovery

［Discovery］タブでは，管理コンソールのナビゲーションペインに表示される管理ハブを設定します。このタブには，次のオプションがあります。

- **［Autodiscover hubs on local subnet］:** ロケーションサービスやネーミングサービスがローカルエリアネットワーク上の指定の管理ポートを使用して，自動的に管理ハブを検出し，管理コンソールに表示できるようにします。この値はできるだけ変更しないでください。

メモ　このタブに表示される管理ポート番号は，スマートエージェント（別名 osagent）ポートとは異なります。

- **［Autodiscover hubs via proxy hubs］:** ロケーションサービスやネーミングサービスが，指定されたプロキシサーバーでホストされている管理ポートを使用して，すべての管理ハブを自動的に検出します。
- **［Enter an explicit list of hubs］:** 管理コンソールに表示される管理ハブを手動で指定します。
- **［Show hub/agent host in name］:** 管理コンソールのナビゲーションツリーで，ハブやエージェントの表示名の隣にハブやエージェントが稼動しているマシンのホスト名を表示する場合は，このチェックボックスをチェックします。

State

［State］タブには，管理コンソールの状態情報の更新を制御する設定があります。また，管理コンソールに表示されるオブジェクトの状態を表すアイコンの凡例もあります。このタブには，次のオプションがあります。

- **［Enable polling for events］:** 設定，パーティション，サービスの状態（実行中，停止中など）を自動的に更新するように管理コンソールに指示します。
  次の設定を使用して，管理コンソールがオブジェクトの状態を検証する頻度や，状態を管理コンソールの［Hubs］ビューのナビゲーションペインで更新する頻度をミリ秒単位で指定します。

  - **[Background polling interval]**：コンソールが，ユーザーと対話していないときに［Hubs］ビューのオブジェクトの状態を確認する頻度を指定します。
  - **[Foreground polling interval]**：ユーザーが，ユーザー名のオブジェクトの状態が変更されるような処理（停止，起動，オブジェクトの再起動など）を実行したときにオブジェクトの状態を確認する頻度を指定します。
  - **[Number of foreground cycles]**：指定のフォアグラウンドポーリング間隔内で，管理コンソールがオブジェクトの状態をチェックする回数を指定します。
- **[Display license warnings and events]**：これをチェックすると，管理コンソールは，ライセンス機能に関連する警告とイベントを表示します。
- **[Enable background refreshes]**：これをチェックすると，管理コンソールは，自動的にナビゲーションツリー内の変更に関する情報（設定，パーティション，サービスの追加や削除など）の表示を更新します。このチェックボックスのチェックをはずすと，管理コンソールのポーリング動作によるオーバーヘッドが減少します。このチェックボックスのチェックをはずし，コンソールの［Refresh］メニューコマンドまたは［Refresh］ツールバーボタンをクリックすれば，管理コンソールに表示された情報を手動で更新できます。
  - **[Refresh every]**：管理コンソールがナビゲーションペインの状態をチェックして表示する頻度をミリ秒で指定します。
- **[State Legend]**：さまざまなオブジェクト状態を表すために管理コンソールで使用されるアイコンを表示します。

Logs

［Logs］タブには，ログファイルと監査ファイルの内容の更新と表示を制御する設定があります。このタブには，次のオプションがあります。

- **[Polling]**：
  - **[Show at most *n* lines]**：表示するメッセージ行（ログエントリ）の最大数を設定します。500 行と指定すると，ログファイルの最新の 500 行が表示されます。
  - **[Enable polling of log file contents]**：ログファイルの統計情報を生成するように管理コンソールを設定します。このチェックをはずすと，統計値のポーリングは行われません。
  - **[Polling interval]**：ログファイルの更新間隔をミリ秒単位で設定します。
- **[Filters]**：
  - **[Filters kept in history]**：管理コンソールで使用するために履歴に保持するログフィルタの数を設定します。
  - **[Clear History]**：管理コンソールで以前に使用したすべてのログフィルタの履歴をクリアします。

Tools

［Tools］の基本設定のタブを使用して，ローカルの Optimizeit Profiler クライアントまたは Optimizeit ServerTrace クライアントがインストールされている場所の絶対（完全修飾）パスを指定します。このパスを設定すると，管理コンソールから Profiler ビューアまたは ServerTrace ビューアを起動できる管理コンソールのツールバーボタンおよびツールメニューオプションが有効になります。

［Path］フィールドに Optimizeit 実行可能ファイルのフルパスを入力するか，［Browse］をクリックしてローカルの Optimizeit インストールを選択します。

サーバー側の Optimizeit と相互運用するように管理コンソールを設定する方法については，第 13 章「Optimizeit Profiler と ServerTrace の使い方」を参照してください。

メモ　管理コンソールを使ってリモートサーバーを管理している場合は，リモートサーバーでスナップショットを生成するために，サーバーが実行されているマシンにも Optimizeit をインストールする必要があります。

VisiBroker

［VisiBroker］タブを使用して，VisiBroker コンソールのユーザーインターフェースオプションを指定します。このタブには，次のオプションがあります。

- **［Remove Stale Service Reference］:** このオプションを有効にすると，VisiBroker コンソールメインビューの My Services フォルダに設定されたすべての古い VisiBroker サービスリファレンスを削除できます。
- **［Enable DNS Lookup］:** このオプションを有効にすると，VisiBroker コンソールに表示するすべての DNS 関連情報をコンソールで解決できます。たとえば，IP アドレスは登録されたホスト名としてコンソールに表示されます。

メモ　DNS ルックアップは，負荷のかかるリソース操作なので，VisiBroker コンソールではこのオプションを提供して DNS ルックアップの有効／無効を制御します。

第 3 章

パーティションの使い方

ここでは，管理コンソールを使用して，Borland AppServer（AppServer）のパーティションを操作する方法を説明します。ここでは，次の作業について説明します。

- 「パーティションの作成，クローンの作成，パーティションの削除」
- 「パーティションへのモジュールとライブラリの配布」
- 「パーティションの設定」
- 「パーティション情報の表示」
- 「パーティションのパフォーマンスの調整」
- 「ログファイルへのパーティションのスタックトレースのダンプ」
- 「パーティションと Optimizeit Profiler または ServerTrace の実行」

パーティションの作成，クローンの作成，パーティションの削除

管理コンソールを使用して，アプリケーションのパーティションの作成，クローン作成，および削除を行うことができます。パーティションは，テンプレートから作成したり，既存のパーティションからクローンとして作成することができます。パーティションは，管理コンソールの［Hubs］ビューのナビゲーションペインに，親設定の子ノードとして表示されます。

新しいパーティションの作成

新しいパーティションを作成するには，次の手順にしたがいます。

1 ナビゲーションペインで，新しいパーティションが所属する［Configuration］を選択します。

2 ［Configuration］を右クリックし，［Add Managed Object］を選択します。
管理オブジェクトの［Managed Object Template Gallery］が開きます。

3 AppServer カテゴリまたは OpenJMS Partition カテゴリで，次のパーティションテンプレートから選択します。

- **[AppServer 5.2.1 Partition]：**AppServer 5.2.1 のデフォルトの partition.config を使って管理パーティションを生成します。
- **[Standard Partition]：**管理パーティションを作成します。
- **[Explicitly Pathed Partition]：**既存のパーティションへのパスを作成します。既存のパーティションを現在の設定の管理下に移動する場合は，このテンプレートを使用します。
- **[JBuilder Partition]：**非管理パーティションを作成します。**JBuilder パーティションは，JBuilder によるローカルサーバーのデバッグに使用されます。デバッグ目的で JBuilder を AppServer とともに使用すると，自動的に作成されます。JBuilder パーティションをほかの目的で使用しないでください。**
- **[Partition with Embedded OpenJMS]：**OpenJMS を含むパーティションを生成します。

4 パーティションテンプレートを選択して，[Add] をクリックします。

[Add From Template] ダイアログボックスが表示されます。

メモ　[Add From Template] ダイアログボックスに表示される情報は，前の手順で選択したテンプレートによって異なります。

5 ダイアログで必要な情報（太字）を入力します。
このダイアログには，選択したテンプレートによって異なるプロパティが表示されます。テンプレート間で共通のプロパティは次のとおりです。

- **[Name]：**[Name] フィールドに一意のパーティション名を指定します。この名前は，ダイアログのほかの部分で文字列置換変数 `${mo.name}` の値として使用され，パーティションの表示名として使用されます。表示名を実際のパーティション名と違う名前にする場合は，[Display Name] フィールドの値を変更します。
- **[Management Agent]：**ドロップダウンリストから有効な管理エージェントを選択して，パーティションを作成するホストを選択します。文字列置換変数 `${hub.name}` で表される現在の管理エージェントを使用する場合，このフィールドを変更する必要はありません。
- **[Display Name]：**オプションで，管理コンソールに表示されるフレンドリ名を入力できます。文字列置換変数 `${mo.name}` には，[Name] フィールドの値が表示されます。
- **[Smart Agent Port]：**このフィールドに有効なポート番号を入力して，デフォルトの osagent（スマートエージェント）ポート番号を変更できます。
- **[HTTP Connector Port]：**このフィールドでデフォルトの HTTP コネクタポート番号を変更できます。
- **[Data Directory]：**明示的にパスが指定されるパーティションの場合は，既存のパーティションのパスを入力する必要があります。

6 その他のパーティション設定プロパティを表示する場合は，[Show hidden properties] チェックボックスをチェックします。

7 終了したら，[OK] をクリックします。

既存のパーティションのクローン作成

既存のパーティションのクローンを作成するには，次の手順にしたがいます。

1 ナビゲーションペインから，クローンを作成するパーティションを選択します。

2 パーティションを右クリックし，[Clone] を選択します。
[Clone Managed Object] ダイアログが表示されます。

3 ［Target Configuration］ドロップダウンリストから，新たにクローンとして作成されたパーティションをターゲットにする設定を選択します。

4 ［New Name］フィールドに，クローンとして作成されるパーティションに指定する名前を入力します。
この名前は，ダイアログのほかの部分で文字列置換変数 ${mo.name} の値として使用され，クローンとして作成されるパーティションの表示名として使用されます。表示名を実際のパーティション名と違う名前にする場合は，［New Display Name］フィールドの値を変更します。

5 必要に応じて，［Description］に入力します。この情報は省略できます。

6 ［Data Directory］（${config.path}/mos/${mo.name}）は，エージェントからの相対パスでパーティションの場所を識別します。

7 ［Target Agent］ドロップダウンリストから，クローンとして作成されるパーティションをホストする管理エージェントを選択します。

8 ［Target Group］ドロップダウンリストから，クローンとして作成されるパーティションが所属するグループを選択します。

9 終了したら，［OK］をクリックします。

パーティションの削除

パーティションを削除するには，次の手順にしたがいます。

1 ナビゲーションペインから，削除するパーティションを選択します。

2 パーティションを右クリックし，［Remove］を選択します。

パーティションへのモジュールとライブラリの配布

パーティションにモジュールを配布するには，次の手順にしたがいます。

1 ナビゲーションペインから，配布先のパーティションを選択します。

2 パーティションを右クリックし，［Deploy modules］を選択します。
モジュールとライブラリの配布ウィザードが表示されます。

3 モジュールを追加するには，［Add］をクリックします。
［Add J2EE Module］ダイアログボックスが表示されます。

   a パーティションに配布するモジュールを参照し，［OK］をクリックします。

   b この手順を繰り返して，すべてのアプリケーションモジュールを追加します。
   間違えた場合は，そのモジュールをリストで強調表示し，［Remove］をクリックすると削除できます。

4 チェックボックスを使用して，追加のオプションを選択します。
次のオプションがあります。

   - **［Restart partitions on deploy (cold deploy)］:**「コールド」デプロイを実行し，配布操作が完了したらパーティションを再起動します。実行中のパーティションへの「ホット」デプロイを実行し，パーティションを再起動しない場合は，この項目をチェックしません。
   - **［Verify deployment descriptors］:** Borland 固有のデスクリプタを含むすべての配布デスクリプタが適切に作成されているかどうかを確認する検証ツールを実行します。これは推奨のオプションです。
   - **［Generate stubs］:** 配布時にアプリケーションスタブを生成する場合は，このボックスをチェックします。これは推奨のオプションです。

5 ［Advanced Options］をクリックして，この配布の詳細なプロパティを設定します。詳細については，18 ページの「配布の詳細オプション」を参照してください。

6 ［Next］をクリックして続行します。
ウィザードのステップ 2 が表示されます。

7 モジュールの配布先になるパーティションを選択します。
選択可能なパーティションが自動的にリストに表示されます。パーティションが表示されない場合は，［Refresh List］をクリックしてください。〔Shift〕キーまたは〔Ctrl〕キーを押したままクリックすると，複数のパーティションを選択できます。

8 終了したら，［Finish］をクリックします。

9 モジュールの配布中は，［Deploying Modules］ダイアログに進行状況が表示されます。

10 終了したら，［Close］をクリックします。

配布の詳細オプション

配布ウィザードのステップ 1 では，スタブ生成と検証ツールの詳細オプションを設定できます。詳細設定を行うには，次の手順にしたがいます。

1 配布ウィザードのステップ 1 で，［Advanced Options］をクリックします。
［Advanced Deployment Options］ダイアログが表示されます。

2 ［Stub Generator］タブで次のオプションを設定できます。

- **［Generate stubs］:** このチェックボックスをチェックすると，配布時にスタブの生成が有効になります。
- **［Classpath］:** ドロップダウンリストからクラスパスを選択したり，［Edit］をクリックし，クラスパスのリストにアーカイブを追加します。
- **［Edit］:** ［Edit］をクリックするとクラスパスエディタが開き，そこでアーカイブとパスの追加および削除ができます。
- **［Java2IIOP arguments］:** このフィールドに `java2iiop` コマンドライン引数を入力します。有効なコマンドライン引数のリストについては，［More Info］をクリックするか，「java2iiop」を参照してください。
- **［More Info］:** ［More Info］をクリックすると，スタブジェネレータのコンパイルフラグの使い方が表示されます。
- **［Javac arguments］:** このフィールドに `javac` コマンドライン引数を入力します。

3 ［Verifier］タブでは，チェックボックスを使って検証のレベルを選択できます。

- **［Verify deployment descriptors］:** Borland と標準のすべてのデスクリプタを確認する場合は，このボックスをチェックします。
- **［Show all warnings］:** デスクリプタに関するすべての問題のログ情報を受け取るには，このボックスをチェックします。
- **［Use strict (pedantic) checks］:** デスクリプタの厳密なチェックを行います。

4 終了したら，［OK］をクリックします。

追加モジュールのホスト

パーティションのフットプリント上にない「配布前の」モジュールをホストすることもできます。さらに，「開かれたアーカイブ」と呼ばれるパス（アーカイブ形式に変換されていないアプリケーション）をホストすることもできます。

パーティションでアーカイブをホストするには，次の手順にしたがいます。

1 ナビゲーションペインから，配布先のパーティションを選択します。

2 パーティションを右クリックし，［Host additional module］を選択します。
［Host Additional Module］ダイアログボックスが表示されます。

3 アーカイブをホストするには，［Select File］を選択します。開かれたアーカイブをホストするには，［Select Directory］を選択します。

4 ホストするアーカイブまたは開かれたアーカイブを指定するには，［Browse］をクリックします。

5 オプションで，ホストされるモジュールにわかりやすいモジュール名を指定できます。

6 終了したら，［OK］をクリックします。

パーティションの設定

ナビゲーションペインでパーティションを選択したときに内容ペインに表示されるプロパティなどのパーティションのプロパティを設定できます。パーティション，パーティションのプロパティ，統計情報の収集の設定，JMX エージェントの設定，ログの設定，JMX クライアントの設定，時間ルール，管理の詳細オプションなどの一般情報を指定できます。

パーティションのプロパティを編集するには，次の手順にしたがいます。

1 ナビゲーションペインでパーティションを選択し，右クリックします。

2 ［Properties］を選択します。
［Partition Properties］ダイアログが表示されます。

3 以下で説明するタブを使用して，設定を編集します。

4 変更が完了したら，［OK］をクリックします。

一般プロパティ

［General］タブのプロパティでは，パーティションの一般情報を指定できます。

次のオプションを編集できます。

- **［Display name］:** 管理コンソールに表示されるパーティションの名前を入力します。
- **［Data directory］:** パーティションのフットプリントの場所（エージェントからの相対パス）を入力します。
- **［Version］:** 管理システムによって作成および管理されるバージョン番号。
- **［Vendor］:** 管理オブジェクトのベンダー。標準パーティションのベンダーは，Borland Software Corporation です。
- **［Description］:** パーティションの説明（オプション）。

パーティション設定のプロパティ

［Partition Settings］タブのプロパティを使用して，パーティションのコマンドライン引数の指定と，JPDA デバッグの設定を行います。

次のオプションを設定できます。

- **［Arguments］:** パーティション実行可能ファイルのコマンドライン引数をスペース区切りリストで入力するか，［Edit］ボタンを押してコマンドライン引数をリストに保存できます。
- **［Enable JPDA remote debugging］:** パーティションでデバッグを有効にするかどうかを指定します。

- **[JPDA debugging transport address]:** パーティションに接続するために JPDA デバッガによって使用されるポート。ランダムにポートを割り当てる場合は，このフィールドを空白にします。
- **[Suspend partition until debugger attaches]:** チェックされた場合は，デバッガがアタッチに成功するまでパーティションを「実行中」としてマークしません。

統計情報のプロパティ

[Statistics] タブのプロパティでは，統計のタブに表示される統計情報を管理コンソールから収集できます。統計情報の収集は，デフォルトで有効になります。統計情報の収集を無効にすると，パーティションのパフォーマンスが向上します。

有効にすると，統計情報の収集について次の設定を行うことができます。

- **[Enable Agent Statistics]:** このボックスをチェックして，パーティションの統計情報エージェントを有効にします。
  [Statistics level] ドロップダウンリストを使用して，ログのレベルを設定できます。また，[Snapshot period] フィールドに値を入力して，ポーリング間隔を設定できます。
- **[Enable Agent Statistics Reaping]:** このボックスをチェックして，保存されている統計情報を定期的に削除し，ディスクスペースを確保します。
  [Reap older than] に値を入力して，統計情報を保存する期間を指定します。[Reap period] に値を入力して，統計情報ログを削除する頻度を設定できます。

JMX エージェントのプロパティ

[JMX Agent] タブのプロパティを使用すれば，パーティションに実装される JMX MBean サーバー，RMI-IIOP アダプタと HTTP アダプタ，および MLet サービスの一部のオプションを設定できます。

次のオプションを編集できます。

- **[Enable JMX]:** このチェックボックスをチェックすると，JMX MBean サーバーが有効になります。
  MBean サーバーは，JMX のエージェント仕様レベルで定義されるインターフェースとファクトリオブジェクトです。このオプションは，JMX コンソールを起動するために有効にする必要があります（第 14 章「JMX コンソールの使い方」を参照）。
- **[Enable HTTP Adaptor]:** このチェックボックスをチェックすると，HTTP アダプタが有効になります。
  HTTP アダプタは，HTML 3.2 準拠のブラウザまたはアプリケーションを使ってパーティションを管理するための HTTP プロトコルのアダプタです。
  この設定タブでは，HTTP アダプタが監視するポートの番号（デフォルト値 8082）を設定し，XSLT プロセッサを有効にできます。Web ブラウザを使ってパーティションを監視する場合は，XSLT プロセッサを有効にして，HTTP アダプタの出力を未編集の XML から HTML に変換する必要があります。ホスト名（デフォルト名 `localhost`）を設定するには，partition.xml を編集する必要があります（詳細については，「<jmx> 要素」を参照）。
- **[Enable RMI-IIOP adaptor]:** このチェックボックスをチェックすると，RMI-IIOP アダプタが有効になります。
  RMI-IIOP アダプタは，JMX クライアントフレームワークに基づくので，管理アプリケーションが RMI を使って MBean サーバーと通信する場合に役立ちます。
- **[Enable mlet service]:** このチェックボックスをチェックすると，MLet サービスが有効になります。
  MLet サービスによって，MBean サーバーの JVM 内の MBean クラスとリソースを 1 つの操作で簡単にリモートホストからロードして登録できます。

JMX エージェントのプロパティ設定の詳細については，「<jmx> 要素」を参照してください。JMX コンソール（パーティションに関連付けられている MBean の監視に使用できる JMX クライアント）の使い方については，第 14 章「JMX コンソールの使い方」を参照してください。

JDK のプロパティ

［JDK］タブのプロパティでは，パーティションに JDK のプロパティを設定できます。

次のオプションを編集できます。

- **［Select the JDK to be used by this partition］:** リストから適切な JDK をクリックして選択します。
- **［Heap and Thread Stack Sizes］:** ［Initial heap size］フィールドに値を入力して，初期ヒープサイズを設定します。
  ［Maximum heap size］フィールドに値を入力すると，最大ヒープサイズを制御できます。VM によるスレッドの管理方法を制御するには，［Java thread stack size］フィールドを使用します。
- **［Java VM Type］:** 適切なラジオボタンを選択して，Java VM タイプを選択します。値は，［Server］，［Client］，［Other］です（ドロップダウンリストからカスタマイズ）。

その他の JDK オプション

［Advanced Configuration］をクリックして，編集ウィンドウに `partition_server.config` ファイルを開きます。必要に応じて，ファイルにエントリを追加します。終了したら，［OK］をクリックします。

パフォーマンス調整のヒント

［Performance Tuning Hints］をクリックして，パーティションのパフォーマンスの最適化に関するヒントを取得します。終了したら，［OK］をクリックします。パフォーマンスの調整の詳細については，25 ページの「パーティションのパフォーマンスの調整」を参照してください。

VisiBroker のプロパティ

［VisiBroker］タブのプロパティでは，パーティションで使用される VisiBroker ORB のプロパティの一部を調整できます。

次のオプションを編集できます。

- **［Select a server connection manager］:** ドロップダウンリストからサーバー接続マネージャを選択します。
- **［Listener port］:** サーバー接続マネージャのリスナーポートを設定します。
- **［Connection Pool］:** これらのフィールドで，許容される最大接続数と最大接続アイドル時間を設定します。
- **［Thread Dispatcher Pool］:** これらのフィールドで，許容されるスレッドの範囲と最大スレッドアイドル時間を設定します。

その他の VisiBroker オプション

［Advanced］をクリックして，編集ウィンドウに `vbroker.properties` ファイルを開きます。必要に応じてファイルを編集します。終了したら，［OK］をクリックします。

セキュリティのプロパティ

［Security］タブのプロパティでは，パーティションのセキュリティ情報を設定できます。

セキュリティを有効にするには，ドロップダウンリストから［Security profile］を選択します。パーティションで SSL を有効にするには，ssl_enabled オプションを選択します。ssl_enabled を選択した場合は，次のオプションも設定できます。

- **［SSL Listener Settings］:** SSL リスナーポートを設定し，そのポートを介して接続するクライアントの信頼を有効にします。
- **［SSL Connection Pool］:** これらのフィールドで，許容される最大接続数と最大接続アイドル時間を設定します。
- **［SSL Thread Dispatcher Pool］:** これらのフィールドで，許容されるスレッドの範囲と最大スレッドアイドル時間を設定します。

ログ設定のプロパティ

［Log Settings］タブのプロパティでは，パーティションにログのプロパティを設定できます。

次のオプションを編集できます。

- **［Partition log format］:** ドロップダウンリストからログ形式（XML またはテキスト）を選択します。
- **［Trace Level Settings］:** トレースが提供されている各サービスにトレースレベル（必要最小限（minimal）または詳細（verbose））を選択します。

時間ルールのプロパティ

［Time Rules］タブでは，パーティションの実行と停止の時刻に関するルールを設定できます。

次のプロパティを編集できます。

- **［Default state］:** パーティションのデフォルトの状態を実行中にするか，停止状態にするかをこのドロップダウンリストで指定します。
- **［Rules］:** パーティションの管理オブジェクトの可用性タイマーのルールを設定します。

詳細設定のプロパティ

［Advanced］タブのプロパティでは，パーティションの管理方法に関する詳細情報を設定できます。

2 種類の管理情報（「管理オブジェクトの設定」と「管理アクションの設定」）を設定できます。

管理オブジェクトの設定

次の管理オブジェクトの設定を編集できます。

- **［Local restart］:** チェックされると，リモートハブが要求を送信するのではなく，ローカルエージェントがパーティションを再起動します。
- **［Escalate stop］:** チェックされると，パーティションの停止処理がタイムアウトになった場合に，強制終了に移行します。
- **［Ping policy］:** ［Always］に設定されると，パーティションは，実行中かどうかにかかわらず常にステータスを ping されます。［Not-when-stopped］に設定されると，パーティションは，実行中とわかる場合にだけ ping されます。
- **［Ping strategy］:** この操作に使用する AppServer アクションストラテジ。
- **［Ping interval］:** パーティションの状態をチェックする間隔（秒）。

- [**Max failure retries**]: 管理オブジェクトが起動，停止などの操作を再試行できる最大回数。
- [**Failure retry interval**]: 管理オブジェクトが操作を再試行する間隔。

管理アクションの設定

ここには，パーティションの起動，停止，強制終了を設定する 3 つのタブがあります。起動，停止，強制終了の各操作に対して，次の項目を設定できます。

- [**Strategy**]: この操作に使用する AppServer アクションストラテジ。
- [**First ping delay**]: ping を開始するまでの時間。最初の ping 操作に遅延時間を設定しない場合は，このフィールドを空白にします。
- [**Ping interval**]: 操作が成功したかどうかを判断するためにパーティションの状態をチェックする間隔。
- [**Retry interval**]:（停止と強制終了のタブのみ）再試行の間隔。
- [**Max retries**]: 最初に操作が失敗してから管理オブジェクトが操作を再試行する最大回数。
- [**Timeout**]: 成功するまで操作を続行する時間。

パーティション情報の表示

ナビゲーションペインでパーティションを選択すると，管理コンソールの右側の内容ペインにさまざまな情報が表示されます。ここでは，内容ペインに表示されるタブと，タブに表示される情報について説明します。

[General] タブ

[General] タブには，パーティションに関する基本的な情報が表示されます。一般プロパティ，パーティションのプロパティ，セキュリティ設定，Web コンテナルートコンテキストの 4 つのカテゴリの情報が表示されます。

一般プロパティ

- [**Display Name**]: 管理コンソールに表示されるパーティションの名前。
- [**Name**]: パーティションの論理名。
- [**Agent name**]: パーティションのホストエージェント名。
- [**Data directory**]: パーティションのフットプリントの場所（エージェントからの相対パス）。
- [**Version**]: ユーザーがオプションで指定したバージョン番号。
- [**Vendor**]: ユーザーが指定したパーティションのベンダー。
- [**Description**]: ユーザーがオプションで指定した説明。

パーティションのプロパティ

- [**Path of partition on server**]: パーティションのフットプリントの物理的な場所。
- [**Verify all modules when loaded**]: パーティションの起動時に検証ツールを実行するかどうかを指定するフラグ。

セキュリティのプロパティ

- [**Security profile**]: パーティションで基本的なセキュリティ（認証／承認）を有効にするかどうかを指定するフラグ。

- [Secure transport enabled]：RPC が SSL を実装するかどうかを指定するフラグ。

Web コンテナルートコンテキスト

[General] タブのこのプロパティでは，Web コンテナのルートコンテキストが表示されます。

[Properties] タブ

[Properties] タブには，JPDA リモートデバッグとサーバー接続マネージャ設定に関する情報が表示されます。

仮想マシン

- [Enable JPDA remote debugging]：パーティションでデバッグを有効にするかどうかを指定します。
- [JPDA debugging transport address]：パーティションに接続するために JPDA デバッガによって使用されるポート。
- [Suspend partition until debugger attaches]：true の場合は，デバッガがアタッチに成功するまでパーティションを「実行中」としてマークしません。

サーバー接続マネージャ設定

- [Server connection manager name]：パーティションにサービスを提供するサーバー接続マネージャ。
- [Listener port]：サーバー接続マネージャが着信接続を監視するポート。
- [Connection Pool]：パーティションへの接続の許容される最少数と最大数の範囲を表示します。
- [Dispatcher Pool]：パーティション内のスレッドの許容される最少数と最大数の範囲を表示します。ほかに，最大スレッドアイドル時間も表示されます。

[XML] タブ

[XML] タブには，パーティションの管理プロパティを定義する XML データブロックが表示されます。このデータブロックは，`configuration.xml` ファイルから抜粋されます。

[Class Loading] タブ

[Class Loading] タブには，パーティションのクラスローディングポリシーとクラスローダー階層が表示されます。

[Logs] タブ

[Logs] タブには，log4j，stderr，stdout のログが表示されます。タブの左上隅にあるドロップダウンリストを使用して，各ログの内容を表示します。log4j ログには，フィルタリングの機能があります。

[Status] タブ

パーティションの統計情報エージェントを有効にしている場合は，[Status] タブにパーティションに関する情報が表示されます。統計情報の収集を有効にするには，「パーティションの設定」の 20 ページの「統計情報のプロパティ」を参照してください。

［JDBC Pool States］タブ

パーティションの統計情報エージェントを有効にしている場合は，［JDBC Pool States］タブに JDBC プールに関する情報が表示されます。統計情報の収集を有効にするには，「パーティションの設定」の 20 ページの「統計情報のプロパティ」を参照してください。

パーティションのパフォーマンスの調整

パーティションには，パフォーマンスのチューニングウィザードがあり，パーティションの操作に関する環境設定を行うことができます。パフォーマンスの調整の詳細プロパティも設定できます。

パーティションを調整するには，次の手順にしたがいます。

1 ナビゲーションペインから，調整するパーティションを選択します。

2 パーティションを右クリックし，［Performance Tuning］を選択します。［Performance Tuning Wizard ｜ Step 1］が表示されます。
このステップで，統計情報の調整の使用モデルを選択できます。ここで選択した内容は，ウィザードの次のステップにある統計情報の収集の設定に反映されます。次のモデルから選択します。

- **［Developer］:** 最大レベルの統計情報の収集を有効にします。
- **［System test］:** 中間レベルの統計情報の収集を有効にします。
- **［Production］:** 最小レベルの統計情報の収集を有効にします。
- **［Best performance］:** このオプションを選択すると，統計情報の収集が無効になり，パーティションのパフォーマンスが向上します。
- **［Custom］:** 目的の設定が上記の定義済みの使用モデルに該当しない場合は，このオプションを選択します。上記の使用モデルをカスタマイズすることもできますが，その場合，使用モデルは*カスタム*と表示されます。

3 終了したら，［Next］をクリックします。［Performance Tuning Wizard ｜ Step 2］が表示されます。
このステップでは，パーティションが操作に関する統計情報を収集する方法を選択します。統計情報は，ディスクに書き込まれ，内容ペインのパーティションの［Statistics］タブに表示されます。パーティションの統計情報の詳細については，「<statistics.agent> 要素」を参照してください。パフォーマンスを最大にするには，統計情報の収集を無効にします。このステップでは，次のオプションを指定できます。

- **［Enable Agent Statistics］:** このボックスをチェックして，パーティションの統計情報エージェントを有効にします。［Statistics level］ドロップダウンリストを使用して，ログのレベルを設定できます。また，［Snapshot period］フィールドに値を入力して，ポーリング間隔を設定できます。
- **［Enable Agent Statistics Reaping］:** このボックスをチェックして，保存されている統計情報を定期的に削除し，ディスクスペースを確保します。［Reap older than］に値を入力して，統計情報を保存する期間を指定します。［Reap period］に値を入力して，統計情報ログを削除する頻度を設定できます。

4 続行するには，［Next］をクリックします。ステップ 3 が表示されます。
不要または使用されていない配布モジュールをパーティションで有効にすると，パーティションの起動時間が長くなり，メモリのフットプリントが増加し，ガベージコレクションにかかる時間が長くなります。無効にするモジュールをリストから選択します。〔Shift〕キーまたは〔Ctrl〕キーを押したままクリックすると，複数のモジュールを選択できます。

5 続行するには，［Next］をクリックします。ステップ 4 が表示されます。
アプリケーションで必要がないパーティションサービスを無効にできます。パーティ

ションサービスを無効にするには，チェックボックスのチェックをはずします。有効にするには，チェックボックスをチェックします。

6 続行するには，［Next］をクリックします。ステップ 5 が表示されます。
VM パラメータをアプリケーションに合わせて調整すれば，パフォーマンスを向上できます。このステップで，次の設定を行うことができます。

- **［Select the JDK to be used by this partition］:** リストから JDK をクリックして選択します。
- **［Heap and Thread Stack Sizes］:** ［Initial heap size］フィールドに値を入力して，初期ヒープサイズを設定します。［Maximum heap size］フィールドに値を入力すると，最大ヒープサイズを制御できます。VM によるスレッドの管理方法を制御するには，［Java thread stack size］フィールドを使用します。
- **［Java VM Type］:** 適切なオプションを選択します。有効な値は，［Server］，［Client］，［Other］です（ドロップダウンリストから選択）。

さらに，一部の詳細設定（26 ページの「パフォーマンスのチューニングの詳細オプション」を参照）を行い，パフォーマンス調整のヒントを参照することもできます。

7 終了したら，［Next］をクリックします。ステップ 6 が表示されます。
パフォーマンスに影響を与える VisiBroker パラメータの一部を設定できます。このステップで，次の設定を行うことができます。

- **［Connection Pool］:** 適切なダイアログボックスで，許容される最大接続数と最大接続アイドル時間を設定します。
- **［Dispatcher Pool］:** ［Minimum number of threads］フィールドと［Maximum number of threads］フィールドで，許容されるスレッド数の範囲を設定できます。［Maximum thread idle time］フィールドを設定します。
- **［Security］:** パーティションで基本的なセキュリティ（認証／承認）を有効にするかどうかを指定するフラグ。

8 続行するには，［Next］をクリックします。最後のステップが表示されます。
このステップでは，J2EE 固有のプロパティを調整できます。次の設定を設定できます。

- **［EJB Container Tuning］:** ［Session passivation timeout］フィールドを設定します。Bean 内呼び出しで PBV を使用するには，チェックボックスをチェックします。
- **［Message-driven bean thread pool］:** ［Minimum number of threads］フィールドと［Maximum number of threads］フィールドに値を設定して，スレッドプールの範囲を設定します。［Maximum thread idle time］フィールドを設定します。
- **［Web Container Tuning］:** ドロップダウンリストから HTTP サービスコネクタを選択し，プロセッサと接続タイムアウトについて設定します。

9 終了したら，［Finish］をクリックします。

パフォーマンスのチューニングの詳細オプション

パフォーマンスのチューニングの詳細オプションにアクセスするには，次の手順にしたがいます。

1 パフォーマンスチューニングウィザードのステップ 4 で，［Advanced Configuration］をクリックします。

2 編集ウィンドウに `partition_server.config` ファイルが表示されます。

3 必要に応じてファイルを編集します。終了したら，［OK］をクリックします。

ログファイルへのパーティションのスタックトレースのダンプ

トラブルシューティングのために，ログファイルにパーティションのスタックトレースをダンプできます。

ログファイルにパーティションのスタックトレースをダンプするには，次の手順にしたがいます。

1 ナビゲーションペインでパーティションを選択します。

2 パーティションを右クリックし，［Dump stack to log］を選択します。

3 スタックトレースは，次の場所にある stdout.log にダンプされます。

```
<install_dir>/var/domains/<domain-name>/configurations/<configuration-name>/
    mos/<partition-name>/adm/logs/<partition_name>.stdout.log
```

パーティションと Optimizeit Profiler または ServerTrace の実行

Optimizeit ServerTrace と Optimizeit Profiler が統合された管理コンソールの設定と使い方については，第 13 章「Optimizeit Profiler と ServerTrace の使い方」を参照してください。

第 4 章

パーティションサービスの使い方

ここでは，パーティションサービスの使い方について説明します。パーティションサービスは，管理コンソールの［Hubs］ビューのナビゲーションペインに，親パーティションの子ノードとして表示されます。

パーティションサービスの起動

パーティションサービスは，パーティションが起動すると自動的に起動します。パーティションサービスの使用を停止するには，ナビゲーションペインでサービスのアイコンを右クリックし，［Disable］を選択します。

パーティションサービスの設定

各パーティションサービスのプロパティの設定ダイアログにアクセスするには，次の手順にしたがいます。

1 ナビゲーションペインでサービスのアイコンをクリックし，［Properties］を選択します。

2 終了したら，［OK］をクリックします。

各サービスで設定できるプロパティの詳細については，次の節を参照してください。

- 「接続サービスのプロパティの設定」
- 「EJB コンテナのプロパティの設定」
- 「JDataStore サーバーのプロパティの設定」
- 「ネーミングサービスのプロパティの設定」
- 「セッションストレージサービスのプロパティの設定」
- 「トランザクションマネージャのプロパティの設定」
- 「Web コンテナのプロパティの設定」

メモ 存続期間インターセプタマネージャのプロパティは，管理コンソールでは設定できません。

接続サービスのプロパティの設定

次の接続サービスのプロパティを設定できます。

- **［Trace level］:**［Trace level］ドロップダウンリストから，イベントログの詳細度を指定します。
- **［Use pass by value for calls］:** チェックすると，コネクタがパラメータを値で渡すことができます。チェックしない場合は，すべての呼び出しをリファレンスで行う必要があります。
- **［Enable statistics gathering］:** チェックしない場合は，コネクタによって収集される統計情報量が減少し，パフォーマンスが向上します。デフォルトは true です。
- **［Use java serialization (not IIOP)］:** チェックすると，IIOP のシリアライゼーションではなく，Java のシリアライゼーションを使ってコネクタがシリアライズされます。

EJB コンテナのプロパティの設定

次の EJB コンテナのプロパティを設定できます。

設定：

- **［Trace level］:**［Trace level］ドロップダウンリストから，イベントログの詳細度を指定します。
- **［Passivation timeout］:** ステートフルセッション Bean に対して，Bean の状態が永続化するまでの許容時間を指定します。
- **［Passivation factory name］:** このコンテナに対して，非アクティブ化の永続性を管理する JSS サービス名を指定します。

フラグ：

- **［Use pass by value for intra bean calls］:** チェックすると，ローカル Bean を呼び出すときに EJB がパラメータを値で渡すことができます。設定しない場合は，すべての呼び出しをリファレンスで行う必要があります。
- **［Use java serialization (not IIOP)］:** チェックすると，IIOP のシリアライゼーションではなく，Java のシリアライゼーションを使ってステートフルセッション Bean がシリアライズされます。
- **［Enable statistics gathering］:** デフォルトでは，チェックされます。チェックをはずすと，コンテナによって収集される統計情報量が減少し，パフォーマンスが向上します。
- **［Enable SIDL］:** チェックすると，SIDL（Simplified Interface Description Language）を使って EJB にアクセスできます。

メッセージ駆動型 Bean スレッドプール：

- **［Minimum number of threads］:** スレッドプールに存在できるメッセージ駆動型 Bean のディスパッチャスレッドの最小数を設定します。
- **［Maximum number of threads］:** スレッドプールに存在できるメッセージ駆動型 Bean のディスパッチャスレッドの最大数を設定します。
- **［Maximum thread idle time］:** メッセージ駆動型 Bean のスレッドをプールから解放するまでアイドル状態にしておくことができる最長時間を設定します。

JDataStore サーバーのプロパティの設定

次の JDataStore サーバーのプロパティを設定できます。

- **［Port］:**JDBC リクエストの監視に使用するポートを設定します。

- [**Socket timeout**]：IO の例外が発生するまで JDS で accept() に対する呼び出しをブロックする時間を指定します。
- [**Temp Directory**]：すべての DataStore 接続で使用する一時ディレクトリを設定します。ディレクトリが指定されていない場合は，現在のディレクトリが使用されます。

ネーミングサービスのプロパティの設定

次のネーミングサービスのプロパティを設定できます。

- [**Backing store**]：永続的データを格納するプラグイン可能バックストアの種類を指定します。指定できる種類は，リレーショナルデータベース，または LDAP ディレクトリサーバーです。サポートされているバックストアアダプタの種類は次のとおりです。
  - [**InMemory**]：インメモリアダプタ。
  - [**JDBC**]：リレーショナルデータベース用の JDBC アダプタ。この設定では，ログイン名，パスワード，URL，JDBC ドライバの場所，プールサイズの値を入力する必要があります。
  - [**Dx**]：DataExpress アダプタ。この設定では，ログイン名，パスワード，URL を入力する必要があります。
  - [**JNDI**]：LDAP のみ。この設定では，ログイン名，パスワード，URL，初期ファクトリ，認証の値を入力する必要があります。

セッションストレージサービスのプロパティの設定

次のセッションストレージサービスのプロパティを設定できます。

- [**Factory name**]：セッションストレージサービスがスマートエージェントに登録するときに使用する名前を指定します。ファクトリ名を指定しない場合は，Borland AppServer（AppServer）は，サーバー名とパーティション名を組み合わせた <server name>/<partition_name> のような値を使用します。
- [**Working directory**]：セッションストレージサービスが，DataStore ファイル usback エンドバックエンドデータベースを検索するディレクトリを指定します。デフォルトの場所は，<install_dir>/var/servers/<server name>/partitions/<partition name> のように稼動中のパーティションディレクトリになります。
- [**Persistent store**]：セッションストレージサービスがバックエンドデータサービスとして使用する JDateStore ファイル名を指定します。デフォルト値は，jss_storage です。
- [**User name**]：セッションサービスが JDataStore バックエンドデータサービスと通信するときに使用するユーザー名を指定します。デフォルト値は，default-user-name です。
- [**Max idle**]：セッションがアクセスされずに，バックエンドデータベースに存続する最小時間を秒単位で指定します。このプロパティが 0（デフォルト）に設定されている場合は，セッションは制限なくデータベースに存続します。
- [**Soft commit**]：チェックすると（デフォルト），セッションストレージサービスはソフトコミットモードを有効にし，JDataStore バックエンドデータベースを使用します。このチェックボックスのチェックをはずすと，セッションストレージサービスのパフォーマンスが向上しますが，システムがクラッシュした場合は，最近コミットされたトランザクションがロールバックされてしまいます。詳細については，http://www.borland.com/techpubs/jdatastore にある JDataStore オンラインドキュメントを参照してください。
- [**Debug**]：チェックすると，デバッグ情報を有効にしてセッションストレージサービスが実行されます。デフォルトでは，このオプションはチェックされていません。

詳細については,『*開発者ガイド*』で「Java セッションサービス（JSS）」の「JSS の管理と設定」を参照してください。

トランザクションマネージャのプロパティの設定

次のトランザクションマネージャのプロパティを設定できます。

- **［Allow Unrecoverable Completion］:** チェックすると，トランザクションサービスが, 対象となるデータベースアプリケーションで正常に完了したかどうかにかかわらず，トランザクションをコミットするように指示します。［Allowing unrecoverable transactions］は，異種データベーストランザクションシステムが含まれる配布構成のテストに役立ちますが，リカバリ操作およびロールバック操作を使用できないので，本稼動のサーバーにはお勧めしません。このオプションは，デフォルトでオフです。
- **［Debug］:** チェックすると，デバッグ情報を有効にしてトランザクションサービスが実行されます。このオプションは，デフォルトでオフです。

Web コンテナのプロパティの設定

デフォルトでは，Web コンテナは，HTTP サービスと IIOP（Internet Inter-ORB Protocol）サービスをホストします。［Configure Web Container］ダイアログでは，Web サービスの XML 設定ファイルを変更できます。このダイアログには，検索しやすい階層表示で XML ファイルの要素と属性が示されます。これらの属性の詳細については，「Borland Web コンテナの IIOP 設定の変更」を参照してください。

パーティションサービス情報の表示

各パーティションサービスのプロパティと統計情報を表示するには，ナビゲーションペインでサービスのアイコンをクリックします。

以下では，管理コンソールを使って表示できるパーティションサービスのプロパティについて説明します。

- 「接続サービスの情報」
- 「EJB コンテナの情報」
- 「JDataStore サーバーの情報」
- 「ネーミングサービスの情報」
- 「セッションストレージサービスの情報」
- 「トランザクションマネージャの情報」
- 「Web コンテナの情報」

接続サービスの情報

23 ページの「［General］タブ」で説明されている一般のプロパティのほかに，接続サービスの［General］タブには，次のプロパティが表示されます。

- **［Trace level］:** イベントログの詳細度を指定します。
- **［Use pass by value for calls］:** `true` の場合は，コネクタがパラメータを値で渡すことができます。設定しない場合は，すべての呼び出しをリファレンスで行う必要があります。
- **［Enable statistics gathering］:** このプロパティを `false` に設定すると，コネクタによって収集される統計情報量が減少し，パフォーマンスが向上します。デフォルトは `true` です。

- **[Use java serialization (not IIOP)]**：true の場合は，IIOP のシリアライゼーションではなく，Java のシリアライゼーションを使ってコネクタがシリアライズされます。

JCA 概要統計

[JCA Summary Statistics] タブには，コネクタの統計情報が表示されます。このタブには 4 つのグラフがあり，それぞれ送信状態，送信要求，受信状態，受信要求のレベルの推移が表示されます。統計情報の収集の有効化と設定の詳細については，20 ページの「統計情報のプロパティ」を参照してください。

EJB コンテナの情報

23 ページの「[General] タブ」で説明されている一般のプロパティのほかに，[Properties] タブに次の EJB コンテナのプロパティが表示されます。

設定：

- **[Trace level]**：イベントログの詳細度を指定します。
- **[Passivation timeout]**：ステートフルセッション Bean に対して，Bean の状態が永続化するまでの許容時間を指定します。
- **[Passivation factory name]**：このコンテナに対して，非アクティブ化の永続性を管理する JSS サービス名を指定します。

フラグ：

- **[Use pass by value for intra bean calls]**：true の場合は，ローカル Bean を呼び出すときに EJB がパラメータを値で渡すことができます。設定しない場合は，すべての呼び出しをリファレンスで行う必要があります。
- **[Enable statistics gathering]**：デフォルトでは，true に設定されます。このプロパティを false に設定すると，コネクタによって収集される統計情報量が減少し，パフォーマンスが向上します。
- **[Use java serialization (not IIOP)]**：true の場合は，IIOP のシリアライゼーションではなく，Java のシリアライゼーションを使ってステートフルセッション Bean がシリアライズされます。
- **[Enable SIDL]**：このプロパティを設定すると，SIDL（Simplified Interface Description Language）を使って EJB にアクセスできます。

メッセージ駆動型 Bean スレッドプール]：

- **[Minimum number of threads]**：スレッドプールに存在できるメッセージ駆動型 Bean のディスパッチャスレッドの最小数を指定します。
- **[Maximum number of threads]**：スレッドプールに存在できるメッセージ駆動型 Bean のディスパッチャスレッドの最大数を指定します。
- **[Maximum thread idle time]**：メッセージ駆動型 Bean のスレッドをプールから解放するまでアイドル状態にしておくことのできる最長時間を指定します。

コンテナサービス

[Container Services] タブには，EJB コンテナがホストする EJB に提供するメインサービスに関する統計データが表示されます。

このタブには 2 つのグラフがあり，それぞれ CMP JDBC とトランザクションのレベルの推移が表示されます。トランザクショングラフには，次の統計情報が表示されます。

- **[Begun Transactions]**:最後のタイムスライスから開始されたトランザクションの数。
- **[Rolledback Transactions]**:最後のタイムスライスからロールバックトランザクションの数。

- [Committed Transactions]：最後のタイムスライスからコミットされたトランザクションの数。

CMP JDBC グラフには，次の統計情報が表示されます。

- [CMP JDBC Query]：最後のタイムスライスからの照会の数。
- [CMP JDBC Update]：最後のタイムスライスからの更新の数。

メモ　統計情報の収集の有効化と設定の詳細については，20 ページの「統計情報のプロパティ」を参照してください。

Bean 要求

[Bean Requests] タブには，EJB 要求動作集計が表示されます。このタブには 4 つのグラフがあり，それぞれエンティティ Bean，メッセージ駆動型 Bean（MDB），ステートフルセッション Bean（SFSB），ステートレスセッション Bean（SLSB）の要求の数の推移が表示されます。各グラフには，最後のタイムスライスから EJB コンテナに行われた各タイプの Bean 要求の数が表示されます。さらに，SFSB のグラフには，次の統計情報が表示されます。

- [Active]：アクティブな SFSB の数。
- [Activated]：最後のタイムスライスからアクティブ化された SFSB の数。
- [Passivated]：最後のタイムスライスから非アクティブ化された SFSB の数。非アクティブ化は，`partition.xml` 内の EJB コンテナの `ejb.sfsb.passivation_timeout` 属性によって制御されます。この期間アクセスされなかったすべての SFSB は，非アクティブ化されます。

Bean 状態

[Bean States] タブには，Bean タイプごとに各タイムスライスの EJB 状態集計が表示されます。このタブには 4 つのグラフがあり，それぞれエンティティ Bean，メッセージ駆動型 Bean（MDB），ステートフルセッション Bean（SFSB），ステートレスセッション Bean（SLSB）の状態の推移が表示されます。

エンティティ Bean のグラフには，プール状態にあるエンティティ Bean の数と，準備完了状態にあるエンティティ Bean の数が表示されます。MDB のグラフには，準備完了状態にある MDB の数が表示されます。SFSB のグラフには，非アクティブ状態にある SLSB の数と，準備完了状態にある SLSB の数が表示されます。SLSB のグラフには，準備完了状態にある SLSB の数が表示されます。

JDataStore サーバーの情報

23 ページの「[General] タブ」で説明されている一般のプロパティのほかに，JDataStore サービスの [General] タブには，次のプロパティが表示されます。

- [Port]：JDBC リクエストの監視に使用するポートを設定します。
- [Socket timeout]：IO の例外が発生するまで JDS で accept() に対する呼び出しをブロックする時間を指定します。
- [Temporary directory]：すべての DataStore 接続で使用する一時ディレクトリを指定します。ディレクトリが指定されていない場合は，現在のディレクトリが使用されます。

ネーミングサービスの情報

23 ページの「[General] タブ」で説明されている一般のプロパティのほかに，ネーミングサービスの [General] タブには，次のプロパティが表示されます。

- **[Backing store]**：永続的データを格納するプラグイン可能バックストアの種類を指定します。指定できる種類は，リレーショナルデータベース，または LDAP ディレクトリサーバーです。サポートされているバックストアアダプタの種類は次のとおりです。
  - ［InMemory］：インメモリアダプタ
  - ［JDBC］：リレーショナルデータベース用の JDBC アダプタ
  - ［Dx］：DataExpress アダプタ
  - ［JNDI］：LDAP のみ
- **[JDBC driver]**：バックストアとして使用するデータベースへアクセスするために必要な JDBC ドライバを指定します。デフォルトは，Java DataStore JDBC ドライバです。サポートされているその他のドライバは，Sybase，Oracle，Borland Interbase，および IBM DB2 です。
- **[URL]**：アクセスするデータベースの URL を指定します。
- **[Login name]**：データベースに関連付けられているログイン名を指定します。デフォルトは `VisiNaming` です。
- **[Password]**：データベースに関連付けられているパスワードを指定します。デフォルトは `VisiNaming` です。
- **[Pool size]**：バックストアとして JDBC アダプタを使用する際の，AppServer 接続プールのデータベース接続数を指定します。
- **[JNDI Initial Factory]**：この JNDI アダプタプロパティは，JNDI 初期ファクトリを指定します。
- **[JNDI Authentication]**：この JNDI アダプタプロパティは，JNDI バックサーバーがサポートしている JNDI 認証のタイプを指定します。

詳細については，『*VisiBroker for Java 開発者ガイド*』の「VisiNaming サービスの使い方」を参照してください。

セッションストレージサービスの情報

23 ページの「[General] タブ」で説明されている一般のプロパティのほかに，セッションサービスの［General］タブには，次のプロパティが表示されます。

- **[Factory name]**：セッションサービスがスマートエージェントに登録するときに使用する名前を指定します。ファクトリ名を指定しない場合は，AppServer は，サーバー名とパーティション名を組み合わせた次のような値を使用します。
  <server name>/*<partition_name>*.
- **[Working directory]**：セッションサービスが，DataStore ファイル usback エンドバックエンドデータベースを検索するディレクトリを指定します。デフォルトは，次のような作業中のパーティションディレクトリです。*<install_dir>*/var/servers/*<server name>*/partitions/*<partition name>*.
- **[Persistent store]**：セッションサービスがバックエンドデータサービスとして使用する JDateStore ファイル名を指定します。デフォルト値は，`jss_factory` です。
- **[User name]**：セッションサービスが JDataStore バックエンドデータサービスと通信するときに使用するユーザー名を指定します。デフォルト値は，`default-user-name` です。
- **[Max idle]**：セッションがアクセスされずに，バックエンドデータベースに存続する最小時間を秒単位で指定します。このプロパティが 0（デフォルト）に設定されている場合は，セッションは制限なくデータベースに存続します。
- **[Soft commit]**：このプロパティが `true`（デフォルト）に設定されている場合，セッションサービスは，Soft Commit モードを有効にして，JDataStore バックエンドデータベースを使用します。このプロパティを `true` に設定すると，セッションサービスの

パフォーマンスが向上しますが，システムがクラッシュした場合は，最近コミットされたトランザクションがロールバックされてしまいます。詳細については，http://www.borland.com/techpubs/jdatastore にある JDataStore オンラインドキュメントを参照してください。

- **[Debug]：**このプロパティを `true` に設定すると，デバッグ情報を有効にしてセッションサービスを実行するように指示します。デフォルトは `false` です。

詳細については，『*開発者ガイド*』で「Java セッションサービス（JSS）」の「JSS の管理と設定」を参照してください。

トランザクションマネージャの情報

23 ページの「[General] タブ」で説明されている一般のプロパティのほかに，トランザクションサービスの［General］タブには，次のプロパティが表示されます。

- **[Allow unrecoverable completion]：**`true` に設定すると，トランザクションサービスが，対象となるデータベースアプリケーションで正常に完了したかどうかにかかわらず，トランザクションをコミットするように指示します。［Allow unrecoverable completion］は，異種データベーストランザクションシステムが含まれる配布構成のテストに役立ちますが，リカバリ操作およびロールバック操作を使用できないので，本稼動のサーバーにはお勧めしません。デフォルトは `false` です。
- **[Debug]：**このプロパティを `true` に設定すると，デバッグ情報を有効にしてトランザクションサービスを実行するように指示します。デフォルトは `false` です。

Web コンテナの情報

23 ページの「[General] タブ」で説明されている一般のプロパティのほかに，Web コンテナサービスの［General］タブには，次のプロパティが表示されます。

- **[Enable naming]：**このプロパティを `true`（デフォルト）に設定すると，Web アプリケーションに提供されているデフォルトの JNDI ネーミングコンテキストを表すために，JNDI（Java Naming and Directory Interface）インプリメンテーションを使用するように Borland Web コンテナに指示します。
- **[Debug]：**このプロパティを `true` に設定すると，デバッグ情報を有効にして Borland Web コンテナサービスを実行するように指示します。デフォルトは `false` です。

コンテナ統計情報と Servlet/JSP 統計情報

コンテナ統計情報とサーブレット /JSP 統計情報のタブには，Web コンテナの統計情報が表示されます。統計情報の収集の有効化と設定の詳細については，20 ページの「統計情報のプロパティ」を参照してください。

第 5 章

Borland AppServer サービスの使い方

Borland AppServer（AppServer）サービスは，AppServer でホストされるすべてのアプリケーションで使用できるサービスです。次のサービスがあります。

- Apache 2.2 Web Server
- スマートエージェント
- JMS サービス
- 2PC トランザクションサービス

各サービスに対して，右側のペインにサービスに関する詳細情報のタブが表示されます。

- ［General］：表示名，名前，エージェント，説明，バージョン，ベンダーが表示されます。2PC トランザクションサービスでは，タイムアウト，スリープ，キャッシュの各プロパティも表示できます。これについては，下の「2PC トランザクションサービス」で説明します。
- XML：`configuration.xml` に含まれる AppServer サービス管理オブジェクトの設定を表示します。
- ログ：サービスに関連するエラーメッセージと状態メッセージを表示します。

これらのパネルは読み取り専用です。情報を設定するには，以下のサービスの設定に関する節を参照してください。

メモ　各 AppServer サービスは，「管理オブジェクト」です。*管理*設定は，詳細なプロパティのタブで設定できます。管理オブジェクトの詳細については，「管理オブジェクト」を参照してください。

Apache 2.2 Web Server

AppServer には，Apache Web Server バージョン 2.2 が組み込まれています。Apache Web Server は，堅固な市販製品クオリティの HTTP プロトコルリファレンスインプリメンテーションです。サードパーティのモジュールを追加することで，Apache Web Server は高度に設定および拡張することができます。Borland では，Apache Web Server のインプリメンテーションに IIOP コネクタモジュールを追加しました。このコネクタモジュー

ルによって，Apache と Tomcat Web コンテナ（下記参照）がインターネットインター ORB プロトコル（IIOP）を介して通信できるようになるため，まったく新しい方法で CORBA のパワフルな機能性を Web アプリケーションに追加できます。AppServer での Apache インプリメンテーションの詳細については,『Borland AppServer *開発者ガイド*』の「Web コンポーネント」を参照してください。

Apache 2.2 Web Server の設定

Apache 2.2 Web Server を設定するには，次の手順にしたがいます。

1 管理コンソールの［Hubs］ビューで［Configurations］に移動します。

2 目的のサービスを含む設定を展開します。

3 ［Apache 2.2］ノード（ラベル付けされた Apache）を右クリックし，［Properties］を選択します。

次の項目が Apache のプロパティ設定パネルに表示されます。

- **［General］:** このタブでは，オブジェクト型とオブジェクト名のリストが表示されます（設定不可）。右側のペインの［General］タブにもすべてのプロパティのリストが表示されます。こちらは設定できます。
- **［**`httpd.conf`**］:** メインの Apache サーバー設定ファイルです。ここには，サーバーに指示を与える設定指示文が含まれています。
- **［mime.types］:** このファイルは，特定のファイル拡張子に対してクライアントに送信されるインターネットのメディア型（アプリケーション，テキスト，イメージファイルなど）を制御します。
- **［magic］:** `mod_mime_magic` Apache モジュールの magic（16 進数）データです。
- **［Uri Map File］:** このファイル内のエントリは, URI を WebClusters.properties ファイルで設定されている Web クラスタ（CORBA インスタンス）にマッピングします。
- **［Web Clusters File］:** このファイル内のエントリは，要求にサービスを提供する Web コンテナに Web クラスタ名を関連付けます。この場合の Web コンテナは，`ReqProcessor` IDL を実装する実際の CORBA サービスです。
- **［Advanced］:** このタブでは，ローカルの再起動，強制終了に移行，ping ポリシー，ping 間隔などの管理オブジェクトの設定を行うことができます。また，開始，停止，および強制終了の管理アクションの設定（ストラテジ run-apache-process，ping インターバル，タイムアウトなど）を行うことができます。

スマートエージェント

スマートエージェントは，AppServer で使用している VisiBroker ORB が提供する分散ディレクトリサービスです。スマートエージェントは, クライアントプログラムとオブジェクトインプリメンテーションの両方で使用する機能で，ローカルサーバーネットワークにあるホストの少なくとも 1 つで起動する必要があります。

メモ Web Services Edition を使用し，Web サーバーや Web コンテナの通信を HTTP などの Web プロトコルを介して行う場合，スマートエージェントを使用する必要はありません。

スマートエージェントの設定

スマートエージェントを設定するには，次の手順にしたがいます。

1 管理コンソールの［Hubs ］ビューで［Configurations］に移動します。

2 目的のサービスを含む設定を展開します。

3 ［Smart Agent］ノード（ラベル付けされた osagent）を右クリックし，［Properties］を選択します。

次の項目がスマートエージェントの設定パネルに表示されます。

- **[General]**：このタブでは，オブジェクト型とオブジェクト名のリストが表示されます（設定不可）。右側のペインの［General］タブにもすべてのプロパティのリストが表示されます。こちらは設定できます。
- **[Settings]**：このタブでは，スマートエージェント `${osagent.port}` 引数を設定できます。さらに，スマートエージェントの開始，停止，監視などの制御オプションを設定できます。
- **[Address File]**：このタブでは，リモートスマートエージェントの IP アドレスが指定されたファイルを編集できます。
- **[Local Address File]**：このタブでは，デフォルトの監視アドレスが指定されたファイルを編集できます。
- **[Advanced]**：このタブでは，ローカルの再起動，強制終了に移行，ping ポリシー，ping 間隔などの管理オブジェクトの設定を行うことができます。また，開始，停止，および強制終了の管理アクションの設定（ストラテジ run-process，ping インターバル，タイムアウトなど）を行うことができます。

JMS サービス

AppServer は，標準 JMS 接続性をサポートし，現在は Tibco メッセージサービスがバンドルされています。Tibco，SonicMQ などの JMS プロバイダの設定をカスタマイズするには，「JMS プロバイダの接続性」を参照してください。Tibco の詳細については，<install_dir>¥jms¥tibco¥doc¥html にあるドキュメントを参照してください。

メモ Tibco Management Console は，管理コンソールの［Tools］メニューから起動します。

JMS サービスを設定するには，次の手順にしたがいます。

1 管理コンソールで，［Configurations］に移動します。

2 目的のサービスを含む設定を展開します。

3 ［JMS Service］ノード（Tibco 用に tibco というラベルが付いている）を右クリックし，［Properties］を選択します。

次の項目が JMS サービスの設定パネルに表示されます。

- **[General]**：このタブでは，オブジェクト型とオブジェクト名のリストが表示されます（設定不可）。右側のペインの［General］タブにもすべてのプロパティのリストが表示されます。こちらは設定できます。
- **[Settings]**：このタブでは，JMS ホームディレクトリ，JMS サーバー URL などの JMS 設定を設定できます。
- **[tibjmsd.conf]**：このタブでは，Tibco JMS デーモンが表示されます。設定可能な Tibco プロセスのすべてのリストです。パラメータの構文は name = value です。
- **[users.conf]**：このタブでは，`<user-name>:[password]:<description>` などのすべてのユーザーを定義する users.conf ファイルを編集できます。パスワードは*常に*システムによってのみ設定されます。テキストエディタで手動で入力してはなりません。テキストエディタでユーザーを追加する場合は，パスワードを空にする必要があります。
- **[groups.conf]**：このタブでは，次のようなすべてのグループを定義する groups.conf ファイルを編集できます。

```
<group-name1>:[<description>]
          <user-name1>
    <user-name2>
       ...
    <user-nameN>
```

- **[topics.conf]**：このタブでは，`<topic-name> [<property1,property2,...>]`（プロパティは，ファイルにリストされている変数）などのすべてのトピックを定義する topics.conf ファイルを編集できます。

- **[queues.conf]：**このタブでは，<queue-name> [<property1,property2,...>]（プロパティは，ファイルにリストされている変数）などのすべてのキューを定義する queues.conf ファイルを編集できます。
- **[ad.conf]：**このタブでは，すべてのユーザーまたはグループのトピックまたはキューに対する次のようなすべてのアクセス許可が記載されている ad.conf ファイルを編集できます。

```
TOPIC=<topic> USER=<user>   PERM=<permissions>
TOPIC=<topic> GROUP=<group> PERM=<permissions>

QUEUE=<queue> USER=<user>   PERM=<permissions>
QUEUE=<queue> GROUP=<group> PERM=<permissions>
```

- **[factories.conf]：**このタブでは，次のような JNDI の接続ファクトリが定義されている factories.conf ファイルを編集できます。

```
[<factory-name>]
type=topic|queue|generic|xatopic|xaqueue|xageneric
url=<url-string>
clientID=<client-id>

<ssl-prop1=value>
<ssl-prop2=value>
  ...
<ssl-propN=value>
```

メモ SSL プロパティはオプションです。

- **[routes.conf]：**このタブでは，AppServer Tibco Enterprise for JMS サーバーとその他の Tibco Enterprise for JMS サーバー間の経路が次のように定義されている routes.conf ファイルを編集できます。

```
 [<route-name>]
# url=<url-string>
#
# <ssl-prop1=value>
# <ssl-prop2=value>
```

- **[bridges.conf]：**このタブでは，宛先間のブリッジが定義されている bridges.conf ファイルを編集できます。ブリッジは，ソースの宛先に送信されたメッセージをターゲットの宛先にブリッジ（再送）します。メッセージをターゲットの宛先に直接送信するアプリケーションと同様に機能します。
- **[transports.conf]：**このタブでは，RV トランスポートと RVCM トランスポートの設定が定義されている transports.conf ファイルを編集できます。

メモ このファイルで定義されたトランスポートは，tibjmsd.conf が tibrv_transports=enabled パラメータを含む場合にだけ有効になります。

- **[tibrvcm.conf]：**このタブでは，[<transport-name>] <listener-name> <rv cm subject> のような Anticipated Tibco RVCM（リスナープログラム）リスナー設定が含まれる tibrvcm.conf ファイルを編集できます。
- **[Advanced]：**このタブでは，ローカルの再起動，強制終了に移行，ping ポリシー，ping 間隔などの管理オブジェクトの設定を行うことができます。また，開始，停止，および強制終了の管理アクションの設定（ストラテジ run-process，ping インターバル，タイムアウトなど）を行うことができます。

2PC トランザクションサービス

AppServer には，CORBA OTS 仕様に基づく VisiTransact Transaction Manager という 2 フェーズコミットトランザクション管理サービスがあります。トランザクションと VisiTransact の詳細については，『Borland AppServer *開発者ガイド*』の「トランザクション管理」を参照してください。

2PC トランザクションサービスの［General］タブには，一般プロパティのほかに，次のような 2PC トランザクションサービスのプロパティが表示されます。

- **［Name］:** スマートエージェントに Transaction Service のインターフェースを登録する際に使用されるインスタンス名を設定します。デフォルトは <server_name>_ots です。
- **［Default transaction timeout］:** 選択したトランザクションサービスインスタンスのデフォルトのトランザクションタイムアウト値を秒単位で設定します。指定しない場合，デフォルトは 600 秒になります。
- **［Default max transaction timeout］:** 選択したトランザクションサービスインスタンスの最大のトランザクションタイムアウト値を秒単位で設定します。指定しない場合，デフォルトは 3600 秒になります。
- **［Log Sleep］:** トランザクションが同期ログ書き込み操作から戻るまでの待機時間（ミリ秒単位）を設定します。デフォルト値は 0 です。同時に実行されるトランザクションが 4 つ以下のマルチスレッド環境では，このデフォルト値が推奨されます。同時に実行されるトランザクションが 5 つ以上のマルチスレッド環境では，10（ミリ秒）が推奨されます。同時に実行されるトランザクション数がきわめて多い場合は，10（ミリ秒）以上が推奨されます。この値を大きくすると，個別のスレッドは遅延しますが，同時処理が過密な場合のスループットは向上します。
- **［Log Cache］:** ログキャッシュのサイズを指定します（キロバイト）。デフォルトは 64 KB です。キャッシュサイズが小さすぎると，ディスクが強制的にフラッシュされる回数が増加します。通常の環境では，デフォルト値が適しています。
- **［Purge old log transaction records］:** トランザクションレコードをログから削除するかどうかを指定します。false（デフォルト）では，既存のトランザクションレコードがログから削除されません。true では，これまでにトランザクションマネージャを起動して取得された既存のトランザクションレコードがログから削除されます。true に設定すると，トランザクションサービスのこのインスタンスを前回実行したときのすべてのライブレコード（完了を保留しているトランザクションのレコードなど）は失われます。さらに，このプロパティを true に設定すると，トランザクションサービスを再起動するときにトランザクションサービスログのディレクトリを変更できます。

2PC トランザクションサービスを設定するには，次の手順にしたがいます。

1. 管理コンソールの［Hubs］ビューで［Configurations］に移動します。
2. 目的のサービスを含む設定を展開します。
3. ［Transaction Service (2PC)］ノード（ots というラベル）を右クリックし，［Properties］を選択します。

次の項目がトランザクションサービスの設定パネルに表示されます。

- **［General］:** オブジェクト型とオブジェクト名がリストされます（設定不可）。表示名，エージェント名，データディレクトリ，バージョン，ベンダー，および説明も表示されます。
- **［Settings］:** このタブでは，［factory name］，［default transaction timeout］，［default max transaction timeout］，［log directory］，［log sleep］，［log cache］，［purge old transaction records］などのプロパティを設定します。
- **［Advanced］:** このタブでは，ローカルの再起動，強制終了に移行，ping ポリシー，ping 間隔などの管理オブジェクトの設定を行うことができます。また，開始，停止，および強制終了の管理アクションの設定（ストラテジ run-process，ping インターバル，タイムアウトなど）を行うことができます。

第 6 章

J2EE コンポーネントの設定の表示

ここでは，J2EE コンポーネントの概要と，J2EE コンポーネントを管理コンソールで表示する方法について説明します。

アーカイブ属性の表示

配布されたモジュールにはさまざまな属性があり，管理コンソールで表示できます。属性は，［General］，［Details］，［References］，［XML］の各タブに表示されます。モジュールには，アーカイブの種類に応じて異なるプロパティが表示されます。

配布されたモジュールの属性を表示するには，次に手順にしたがいます。

1 管理コンソールの［Hubs］ビューで［Configuration］に移動します。

2 設定内のパーティションを展開し，［Deployed Modules］グループのメンバーを選択します。管理コンソールの内容ペインのタブに，配布されているモジュールの情報が表示されます。

［General］タブ

［General］タブには，モジュールのプロパティがリストされます。プロパティには，モジュール名，表示名，モジュールのタイプ（Web またはアプリケーション），J2EE のバージョン，モジュールのバージョン，および説明があります。

モジュールのストレージと移動性の設定は，パーティションの外部でパスを指定されたアーカイブのホスト（サーバーパスからホスト）と，パーティション上の開かれたアーカイブのホスト（開かれたアーカイブとして保存）に関連します。詳細については，17 ページの「パーティションへのモジュールとライブラリの配布」を参照してください。

- **［Stored as exploded archive］:** 通常，開かれた（展開された）アーカイブをパーティションに配布することはできません。この機能を使用して，パーティションは，サーバーのローカルディレクトリで開かれたアーカイブをホストできます。このディレクトリは，ファイルシステムから可視である必要があります。明示的にパスが指定されるアーカイブと同様に，開かれたアーカイブは配布できる状態である必要があります。つまり，配布する前にすべての必要なスタブとスケルトンが生成されている必要があります。開かれたアーカイブには，明示的にパスが指定されたアーカイブだけを指定でき，パーティションのツリーの［Hosted Modules］フォルダに配布されます。パーティションでホ

ストされる特定のアーカイブが開かれることを示す場合に，［Stored as exploded archive］ボックスをチェックします。

開かれたアーカイブのホストを使用する典型的な例は WAR の場合で，実行中のアプリケーションが自分自身を変更できるようになります。追加または変更された JSP はすべて認識され，［Hosted Modules］フォルダでパーティションによって動的にホストされます。

- **［Hosted from server path］:** この機能を使用して，パーティションは，サーバーのローカルディレクトリにある任意のアーカイブをホストできます。これを「明示的にパスが指定されたアーカイブ」と呼びます。これらのアーカイブは，サーバーのファイルシステムから可視である必要があります（localhost にある必要はありません）。これらのアーカイブはパーティションには配布されません。そのかわり，パーティションはアーカイブをホストするように明示的に指示されます。明示的にパスが指定されたアーカイブは，パーティションのツリーの［Hosted Modules］フォルダに置かれます。

  ホストされるアーカイブは，すべてのスタブとスケルトンが生成された配布できる状態である必要があります。通常，スタブとスケルトンは配布の際に生成されます。これらのアーカイブはホストされるため，ホストするアーカイブに対してスタブ生成ウィザードを使用する必要があります。必要なスタブとスケルトンが生成されると，アーカイブをパーティションに追加できます。

  モジュールがパーティションのリポジトリに配布されると，［Deployed］オプションが選択されます。

追加の Web モジュールのプロパティセクション（WAR）

［Deployed Modules］フォルダ内の WAR には，［General］タブに追加の Web モジュールプロパティが表示されます。選択した WAR のコンテキストルートとすべての URL が［Additional Web Module Properties］に表示されます。［URL］ウィンドウから URL を選択し，［Test URL］ボタンをクリックすると，URL アドレスをテストできます。

［Details］タブ

［Details］タブには，子モジュール（JAR，WAR など），meta-inf，マニフェスト（XML ファイル）などのアーカイブの内容が表示されます。列ヘッダーをクリックすると，名前やタイプ，日付で並べ替えることができます。任意の内容をダブルクリックすると，ファイルの内容が表示されます。

［References］タブ

［References］タブは，特定の配布モジュールにある読み取り専用の表示領域です。タブの上部のツールを使用して，すべての EJB リファレンスをさまざまな方法でグラフィカルに表示できます。このタブは，配布デスクリプタエディタ（DDEditor）にもあります。

［XML］タブ

［XML］タブを開くと，ナビゲーションツリーに別のペインが開きます。このペインでいずれかのデスクリプタファイルをクリックすると，内容ペインにその内容が表示されます。［XML］タブには，標準デスクリプタ（web.xml など）と Borland 固有のデスクリプタ（web-borland.xml など）を含むモジュールの XML 配布デスクリプタが表示されます（読み取り専用）。標準の配布デスクリプタおよびベンダー固有の配布デスクリプタの詳細については，第 9 章「配布デスクリプタエディタの使い方」を参照してください。

メモ このビューの XML は編集不可です。デスクリプタを編集するには，配布デスクリプタエディタ を使用します。ただし，XML の内容ペインで右クリックすれば，ジャンプ，コピー，および貼り付けを行うことができます。

ライブラリモジュールの［XML］タブ

ライブラリモジュールの［XML］タブは，初期化やクリーンアップなどの期限付きのパーティション操作やパーティション存続期間インターセプタに関連するタブです。パーティション存続期間イベントに関連付けられた特別な関数呼び出しに応答したり対話するクラスを登録するためにライブラリモジュールで必要になる XML が表示されます。このようなコードをホストできるのはライブラリモジュールだけなので，ユーザーは，イベントの関数に渡されるクラス名とパラメータをライブラリモジュールの XML ファイルで宣言します。ライブラリモジュールの XML やパーティション存続期間インターセプタの詳細については，『**Borland AppServer 開発者ガイド**』の「パーティション存続期間インターセプタ」を参照してください。

［Axis Properties］タブ

［Axis Properties］タブには，Axis サービスの要素（サービス，ハンドラ，型マッピング，トランスポート，チェイン，およびグローバル設定）が表示されます。詳細については，47 ページの「Web サービス配布デスクリプタプロパティの表示」を参照してください。

［Summary Statistics］タブ

配布に関する統計を表示する場合は，管理コンソールの［Summary Statistics］タブが有効になります。

パーティションからのモジュールの削除

配布されたモジュールをパーティションから削除するには，次の手順にしたがいます。

1 削除する配布されたモジュールのノードを管理コンソールで選択します。

2 削除するモジュールを右クリックし，コンテキストメニューから［Remove］を選択します。

メモ　モジュールを完全に削除してしまうと，そのモジュールはパーティションから削除されます。

Enterprise Archive（EAR）について

管理コンソールを使用すると，Enterprise Archive（EAR）としてパッケージされているエンタープライズアプリケーションアーカイブの表示，管理ができます。EAR は，配布可能な単位で，サーバーの中の 1 つのアプリケーションを表します。各 EAR は JAR ファイルとよく似た方法でパッケージされ，アプリケーションを構成するモジュールと配布デスクリプタのすべてを含んでいます。

EAR の表示

配布された EAR の属性を表示するには，次の手順にしたがいます。

1 管理コンソールを開きます。

2 配布されたモジュールを含む設定を選択します。

3 ナビゲーションツリーで，パーティションを表すノードを展開します。

4 ［Deployed Modules］ノードを展開し，表示する EAR を選択します。

EAR に関連付けられている JAR ファイルや WAR ファイルがあれば，それらも表示されます。EAR は複数の JAR と WAR を格納できます。

1 EAR 属性を表示するには，［Deployed Modules］フォルダノードから EAR を選択します。

2 ［Details］タブ，［XML］タブ，または［References］タブを選択して，EAR の属性を表示します。

3 作業ペインでファイルを開くには，そのファイルをダブルクリックします。

EAR のロード順序の指定

複数の EAR を特定の順序でロードする場合は，partition.xml ファイルを変更して，その順序を指定する必要があります。たとえば，a.ear，b.ear，c.ear，d.ear，e.ear の 5 つの EAR があり，e を d より先に，d を c より先にというようにロードする場合は，該当する partition/properties フォルダにある partition.xml ファイルを次のように変更する必要があります。

```
<archives ear.repository.path="ears" war.repository.path="wars"
ejbjar.repository.path="ejb_jars" dar.repository.path="dars"
rar.repository.path="rars" lib.repository.path="lib"
classes.repository.path="classes">
    <archive name="e.ear" disable="false" order="1"/>
    <archive name="d.ear" disable="false" order="2"/>
    <archive name="c.ear" disable="false" order="3"/>
    <archive name="b.ear" disable="false" order"=4"/>
    <archive name="a.ear" disable="false" order="5"/>
</archives>
```

上の EAR は，e，d，c，b，a の順序でロードされます。

順序が指定されていない EAR は，最後に（アルファベット順で）ロードされます。同じ順序が指定されている ERA は，アルファベット順でロードされます。このしくみは，EAR に対してのみ実装されています。したがって，他のアーカイブのロード順序を指定するには，そのアーカイブを EAR に格納し，partition.xml を適切に変更する必要があります。

Web アーカイブについて

管理モジュールでは，Web アーカイブ（WAR）を表示して管理できます。ただし，WAR は EAR 内にカプセル化されているとは限りません。また，Web オブジェクトだけを表します。一般に WAR には次の項目が含まれます。

- サーブレットの Java クラスファイルとヘルパークラス
- JSP ページとそのヘルパークラス
- 静的 HTML 文書
- アプレットとそのクラスファイル
- XML 配布デスクリプタ：web.xml ファイルおよび Borland 固有の web-borland.xml ファイル

WAR の表示

配布された WAR の属性を表示するには，次の手順にしたがいます。

1 管理コンソールを開きます。

2 配布されたモジュールを含む設定を選択します。

3 ナビゲーションツリーで，パーティションを表すノードを展開します。

4 ［Deployed Modules］ノードを展開し，表示する WAR を選択します。

［Details］タブ，［References］タブ，［XML］タブ，および［Summary Statistics］タブに，WAR の属性を表示できます。配布する WAR を作成する方法については，第 9 章「配布デスクリプタエディタの使い方」を参照してください。

Web サービス配布デスクリプタプロパティの表示

管理コンソールでは，WAR ファイルにパッケージされた Web サービス配布デスクリプタ（WSDD）のプロパティを表示できます。

管理コンソールから server-config.wsdd ファイルを表示するには，次の手順にしたがいます。

1 管理コンソールを開きます。

2 配布されたモジュールを含む設定を選択します。

3 ナビゲーションツリーで，パーティションを表すノードを展開します。

4 ［Deployed Modules］ノードを展開し，表示する Web サービスを含む WAR を選択します。

5 内容ペイン下部の［Axis Properties］タブをクリックします。server-config.wsdd ファイルのプロパティ要素が読み取り専用レイアウトでマッピングされます。

6 ［Elements］ドロップダウンリストから，表示する要素のタイプを選択します。

要素を選択すると，関連付けられているパラメータが隣のテーブルに自動的に表示されます。ドリルダウンして［Parameters of Element］詳細を表示できます。

ライブラリについて

AppServer のライブラリには，モジュール間で共有できる JDBC ドライバなどのクラスが含まれており，JAR ファイルまたは ZIP ファイルとして格納されています。モジュールでこれらのライブラリを使用する場合は，サーバーのクラスパスに格納されます。

たとえば，モジュールがデータベースにアクセスするために一連のドライバが必要な場合，ドライバを JAR に入れて "Installed Libraries" フォルダに配布します。そのデータベースにアクセスするすべてのモジュールが，このフォルダからドライバにアクセスできるようになります。

ライブラリの表示

インストールされているライブラリを管理コンソールで表示するには，次の手順にしたがいます。

1 管理コンソールを開きます。

2 配布されたライブラリを含む設定を選択します。

3 ナビゲーションツリーで，パーティションを表すノードを展開します。

4 ［Deployed Modules］ノードを展開して，表示するライブラリを選択します。

ライブラリの属性の表示

ライブラリにはさまざまな属性があり，それらを表示することができます。表示できる属性には，ライブラリ名，タイプ，データサイズ，パーセント（圧縮率），圧縮済サイズ，パスなどの値があります。

ライブラリの属性を表示するには，表示するライブラリを選択します。右側のペインにライブラリの属性が表示されます。

メモ　ライブラリの配布の詳細については，『Borland AppServer *開発者ガイド*』のデータソースを参照してください。

リソースアダプタについて

リソースアダプタは，サードパーティのエンタープライズ情報システム（EIS）のデータにアクセスするための標準的な信頼できる方法です。リソースアダプタは，リソースアダプタアーカイブ（RAR）としてパッケージされて，Borland AppServer（AppServer）に配布されます。リソースアダプタを使用して，サーブレット，JSP ページ，エンタープライズ Bean，J2EE アプリケーションクライアント，および CORBA クライアントから EIS にトランザクション対応のアクセスを行うことができます。RAR は，Sun の J2EE 1.3 仕様で指定されている XA_TRANSACTION モードを使用します。

リソースアダプタの表示

配布された RAR の属性を表示するには，次の手順にしたがいます。

1 管理コンソールを開きます。

2 配布されたモジュールを含む設定を選択します。

3 ナビゲーションツリーで，パーティションを表すノードを展開します。

4 ［Deployed Modules］ノードを展開して，表示する RAR を選択します。WAR は EAR ファイル内にある場合もあります。

リソースアダプタ情報の表示

RAR に関連付けられた次の情報を表示できます。

- **［Attributes］:** RAR にはさまざまな属性があり，それらを表示することができます。表示できる属性には，RAR プロパティと値，RAR 内のクラスの詳細などがあり，［Details］タブに表示されます。
- **［State］パネル :** 接続ファクトリインスタンスのカウント（円グラフ，折れ線グラフ，または表），接続ファクトリ状態図，接続ファクトリ状態プロパティ（編集可能）を表示します。

RAR の属性を表示するには，次の手順にしたがいます。

1 RAR を選択します。

2 作業ペインに表示する属性の種類を選択します。

RAR の［State］パネルを表示するには，次の手順にしたがいます。

1 RAR を選択します。

2 RAR のファクトリアイコンを選択します。

JAR 情報の表示

管理モジュールでは，Java アーカイブ（JAR）を表示して管理できます。JAR は，WAR または EAR 内のモジュールの場合もあります。EJB は，JAR ファイル内にカプセル化され，表示されます。

通常，JAR の内容は次のとおりです。

- Java クラスファイル
- 標準の XML ファイル
- Borland 固有またはベンダー固有の XML ファイル

JAR の表示

配布された JAR の属性を表示するには，次の手順にしたがいます。

1 管理コンソールを開きます。

2 配布されたモジュールを含む設定を選択します。

3 ナビゲーションツリーで，パーティションを表すノードを展開します。

4 ［Deployed Modules］ノードを展開し，表示する JAR を選択します。JAR は WAR ファイルや EAR ファイルに含まれる場合もあります。

［Details］タブ，［XML］タブ，または［References］タブに WAR の属性を表示できます。配布する JAR を作成する方法については，第 9 章「配布デスクリプタエディタの使い方」を参照してください。

EJB 情報の表示

管理モジュールでは，EJB を表示して管理できます。EJB は，JAR ファイル内にカプセル化され，表示されます。JAR は，WAR または EAR 内のモジュールの場合もあります。

EJB に関連付けられた次の情報を表示できます。

- **［State］パネル：**Bean インスタンスのカウント，Bean 状態図，Bean 状態プロパティを表示します。

エンタープライズ Bean の種類

Borland の EJB コンテナでは，次の 3 種類のエンタープライズ Bean がサポートされています。

- **［Stateless Session beans］：**これらの Bean の状態は，必要に応じてクライアントまたはデータベースなどの外部に保存されます。このようなセッション Bean は，状態を保持しないため，特定のクライアントに結び付けられることがありません。これらの Bean については，任意の使用可能なインスタンスを使ってクライアントにサービスを提供できます。
- **［Stateful Session beans］：**これらの Bean の状態は，エンタープライズ Bean によって保持されます。これは，アプリケーションサーバーがクライアントと Bean の組を管理することを意味します。エンタープライズ Bean の各インスタンスは，クライアントのかわりに作成され，そのクライアントにとってプライベートなリソースになります。ステートフルセッション Bean はデータベースやファイルなどの永続的リソースにアクセスできますが，エンティティ Bean とは異なり，実際のデータを表すわけではありません。
- **［Entity beans］：**これらの Bean は，永続的ストレージに格納されているデータをオブジェクトとして表現する永続的オブジェクトです。レコードがデータベースの中で存続するのと同様に，エンティティ Bean は EJB コンテナの中で存続します。ステートフルセッション Bean とは異なり，エンティティ Bean には複数のクライアントが同時にアクセスできます。この同期処理は EJB コンテナによって管理されます。

次の Bean 属性は，配布デスクリプタエディタ（DDEditor）を使用して，表示および編集できます。

- ［Bean Name］
- ［Bean Type］
- ［Home Name］
- ［Remote Name］
- ［Local］

- ［Local Home］
- ［JNDI Home］
- ［JNDI Local Home］
- ［Bean Class］
- ［Session Type］
- ［Transaction Type］
- ［Storage Timeout］

ホストされるモジュール

ホストされるモジュールは，パーティションのホームディレクトリ <install root> 以外の場所でパーティションによってホストされるモジュールです。これらのモジュールは，サーバーの任意のローカルファイル（サーバーのファイルシステムから可視のファイル）内に置くことができます。localhost にある必要はありません。ホストされるモジュールは，ロードされると，［Hosted Modules］フォルダに表示されます。

ホストされるモジュールは，すべてのスタブとスケルトンが生成された配布できる状態である必要があります。通常，スタブとスケルトンは配布の際に生成されます。ホストするモジュールに対してスタブ生成ウィザードを使用できます。必要なスタブとスケルトンが生成されると，モジュールを［Hosted Modules］フォルダに追加できます。

開かれたアーカイブ

開かれたアーカイブは，「解凍された」状態にあるアーカイブです。通常，開かれた（展開された）アーカイブをパーティションに配布することはできません。「追加モジュールをホスト」機能を使用して，パーティションは，サーバーのローカルディレクトリ（サーバーのファイルシステムから可視のディレクトリ）にある開かれたアーカイブをホストできます。

ホストされるモジュールと同様に，開かれたアーカイブは配布できる状態である必要があります。配布する前に，すべての必要なスタブとスケルトンを生成する必要があります。開かれたアーカイブには，ホストされるモジュールだけを指定でき，［Hosted Modules］フォルダに配布されます。

開かれたアーカイブのホストを使用する典型的な例は WAR の場合で，実行中のアプリケーションが自分自身を変更できるようになります。追加または変更された JSP はすべて認識され，［Hosted Modules］フォルダでパーティションによって動的にホストされます。

追加モジュールや開かれたアーカイブを［Hosted Modules］でホストするには，次の手順にしたがいます。

1 パーティションまたは［Hosted Modules］ノードを右クリックし，［Host additional module］を選択します。

2 次のどちらかを行います。

- ホストされるモジュールの**ファイルを選択**します。モジュールが置かれているファイルを参照します。
- 開かれたアーカイブの**ディレクトリを選択**します。開かれたアーカイブが置かれているディレクトリを参照します。

3 モジュールの名前を変更する場合は，名前を入力します。デフォルト名の場合は，空白のままにします。

4 ［OK］をクリックします。

EJB コンテナについて

Borland AppServer は，パーティション内でホストされている統合された EJB コンテナサービスを提供します。標準のコンテナサービスに加えて，EJB コンテナは，トランザクションの範囲と永続性サービスを提供し，J2EE サービスおよび通信 API へのアクセスを可能にします。次の節では，コンテナに対する永続性のサポートについて説明します。

永続性のサポート

Sun Microsystems が公開した EJB 仕様バージョン 2.1 以降に準拠する AppServer の EJB コンテナには，永続性のサポートが組み込まれています。この永続性は，Bean 管理またはコンテナ管理です。

永続性を Bean で管理する場合は，`ejbLoad()` メソッドと `ejbStore()` メソッドをオーバーライドして，オブジェクトのフィールドのデータを永続的ストレージに保存したり，そこから読み込むためのカスタムコードを JDBC などを使って作成する必要があります。永続性を Bean で管理するには，エンタープライズ Bean を 1 つの永続化メソッドに結び付け，永続化のためのコードを作成して保守する必要があります。

永続性がコンテナ管理の場合，エンティティ Bean のフィールドを JDataStore のフィールド群にマッピングする Borland のツールを使用できます。エンティティ Bean は，配布デスクリプタのフィールド群の再マッピングにより，異なるデータソースに永続化されます。

永続性のサポートの詳細については，『*Borland AppServer 開発者ガイド*』の「Borland AppServer のエンティティ Bean と CMP 1.1」を参照してください。

メモ　［Deployed Modules］の下の JAR ファイルまたはアプリケーション（EAR）ファイルはすべて，EJB を含んでいることを表しています。パーティションサービスの 1 つとして表示される「EJB Container」は，EJB コンテナが提供するサービスを表します。このサービスには，データベースストレージ用のコンテナ管理の永続性（Container-Managed Persistence，CMP），オブジェクトを参照するネーミングサービス，Web コンテナなどがあります。

EJB サーバーには，複数のパーティションを保持できます。必要に応じて，オブジェクトがネットワークを介して配布されるように定義して設定できます。Oracle など，さまざまなデータソース用の EJB コンテナにカスタムパーティションを追加することで，それぞれを EJB 開発とは別にサポートできます。

EJB コンテナサービス属性の表示

EJB コンテナサービスには，表示および編集できる多くの属性があります。通常，コンテナサービスを表示するには，使用しているパーティションの［EJB Container］を選択します。サーバーのプロパティを編集するには，サービスを右クリックし，［Properties］を選択します。サービスの属性は，［General］，［Properties］，［Container Services］，［Bean Requests］，［Bean States］の各タブに表示されます。

メモ　ここでは，コンテナサービスの属性を表示する方法について説明します。EJB 属性の詳細については 49 ページの「EJB 情報の表示」を参照してください。

EJB コンテナサービス属性を表示するには，次の手順にしたがいます。

1. 管理コンソールを開きます。
2. EJB コンテナを含む設定を選択します。
3. ナビゲーションツリーで，パーティションを表すノードを展開します。
4. 表示する EJB コンテナノードをクリックします。

Web コンテナについて

Web コンテナは，Web アプリケーションの開発と配布をサポートするように設計されています。このバージョンの AppServer は，Web コンテナとして Tomcat 5.5.12 を提供します。Web コンテナのプロパティは，［General］，［Container Statistics］，［Servlet/JSP Statistics］の各タブにあります。

Web コンテナの表示

Web コンテナを表示するには，次の手順にしたがいます。

1 Borland 管理コンソールを開きます。

2 Web コンテナを含む設定を選択します。

3 ナビゲーションツリーで，パーティションを表すノードを展開します。

4 ［Deployed Modules］を展開します。

5 Web コンテナを選択します。

第 7 章

インストールビュー

［Installations］ビューを使用すれば，コンソールを実行するローカルホストに配布された Borland Deployment Platform コンポーネントを表示および設定できます。［Installations］ビューの使用目的は次のとおりです。

- ローカルインストールのプロパティと設定ファイルを表示および編集する
- ホストにインストールされている Borland AppServer (AppServer) に配布されたアーカイブを表示する
- ローカルホストに格納されているセキュリティプロファイルを表示および編集する
- パーティション，設定，および管理オブジェクトに使用する XML テンプレートを表示および編集する
- ローカルにインストールされているエージェントの情報がある場合は，それを表示する

警告 システムの設定には，管理コンソールの［Hubs］ビューを使用することをお勧めします。［Installations］ビューがオプションを提供できるのは，コンソールを実行するローカルホストにインストールされている Borland AppServer だけです。システムの設定に［Installations］ビューを使用する場合は，サーバーをシャットダウンしてから設定ファイル，プロパティファイル，またはその他のサービスを編集してください。サーバーをシャットダウンせずに編集すると，編集した内容が失われ，サーバーで予期しない結果が生じる場合があります。

［Installations］ビューには，次の領域があります。

- Borland AppServer 製品のすべてのインストールを表示するビュー。コンソールが実行されているインストールのインストールフォルダには，赤いドットが表示されます。
- 展開／折りたたみ可能なツリーブラウザが，Borland 管理コンソールと同じマシンにインストールされたサーバー，サービス，パーティション，モジュールを表示します。ツリーのオブジェクトを表示したり，修正するには，アイコン（ノード）を展開して目的のオブジェクトまで移動します。ツリーのオブジェクトにその他のオブジェクトが含まれている場合は，アイコンをダブルクリックして画面を展開することもできます。
- Borland 管理コンソールは，ツリーで選択されたオブジェクトに関連付けられた設定，プロパティ，セキュリティ設定，またはログファイルを発見したときに，ツリーの下のペインを開きます。
- 右側の編集ウィンドウには，選択したファイルまたはツリーノードの内容が表示されます。選択したコンポーネントを設定するダイアログが表示される場合があります。プロ

パティファイルと設定ファイルの場合，編集ウィンドウには変更または保存するファイルの内容が表示されます。

プロパティファイルと設定ファイルの使用

ツリーで選択したコンポーネントに関連するプロパティファイルまたは設定ファイルがある場合，該当する［Property Files］タブと［Configuration Files］タブが内容ペインに表示されます。ファイルを表示するには，次の手順にしたがいます。

1 表示するファイルがあるツリーのノードを選択します。ノードに設定可能なプロパティまたは設定ファイルがある場合は,［Property Files］タブまたは［Configuration Files］タブが内容ペインで使用可能になります。

2 該当するタブをクリックします。

3 構造ペインに，コンポーネントに関連する設定ファイルまたはプロパティファイルのリストが表示されます。

4 構造ペインのリストからファイルを選択します。内容ペインに内容が表示されます。デフォルトでは，［Configuration Files］タブまたは［Property Files］タブを選択すると，リストの最初のファイルの内容が選択されます。

5 編集ウィンドウでファイルを変更します。ファイルの正しい使い方に関する Borland のコメントをお読みください。コメントは緑のテキストで表示されます。ファイルの編集可能な部分は黒のテキストで表示されます。変更されたテキストは青で表示されます。

6 編集が終了したら，内容ペインのツールバーの左上にある［Apply Changes］アイコンをクリックします。

さまざまな Borland Deployment Platform ツールの起動プログラムが使用する設定ファイルも編集できます。(たとえば，Tibco JMS サービスまたは配布デスクリプタエディタの設定ファイルを編集できます)。ファイルを編集する場合は，次の手順にしたがいます。

1 ナビゲーションペインで,各自のインストールのルートノードを展開します (例:C:/BDP)。

2 ［Tool Configuration］ノードを選択します。

3 構造ペインに設定できるツールのリストが表示されます。

4 構造ペインのリストからファイルを選択します。内容ペインに内容が表示されます。デフォルトでは，［Tool Configuration］ノードを選択すると，リストの最初のファイルの内容が表示されます。

5 編集ウィンドウでファイルを変更します。ファイルの正しい使い方に関するコメントをお読みください。コメントは緑のテキストで表示されます。ファイルの編集可能な部分は黒のテキストで表示されます。変更されたテキストは青で表示されます。

6 編集が終了したら，内容ペインのツールバーの左上にある［Apply Changes］アイコンをクリックします。

ローカルアーカイブの使用

［Installations］ビューを使用すれば，コンソールを実行するローカルホストに配布されているアーカイブの内容を確認できます。［Archives］ノードを展開すると，ホストされているすべてのアーカイブがタイプ別に表示されます。特定のアーカイブに操作を実行することもできます。

アーカイブデータの表示

アーカイブ情報を表示するには，次の手順にしたがいます。

1 ナビゲーションペインで,各自のインストールのルートノードを展開します (例:C:/BDP)。

2 [Archives] ノードをクリックします。内容ペインに，ホストされているアーカイブの情報が表示されます。アーカイブの名前，タイプ，サイズ，およびその場所のパスが表示されます。

3 [Archives] ノードを展開して，Borland Deployment Platform が認識するアーカイブのタイプを表すフォルダを表示します。

4 ツリーを展開するか (推奨)，または内容ペインに表示されるフォルダをダブルクリックしてドリルダウンします。

5 個々のアーカイブファイルを選択してアーカイブの情報を表示します。内容ペインには，表示できるさまざまなアーカイブ特性のタブが表示されます。表示されるタブは次のとおりです。

- [General] タブ : モジュールの名前, タイプ, Java バージョンなどの情報を表示します。
- [Details] タブ : クラスファイルを含むアーカイブの内容を表示します。
- [References] タブ : アーカイブ内の Bean およびその他のコンポーネント間の参照をグラフィカルに表示します。
- [XML] タブ : アーカイブの配布デスクリプタを表示します。

6 アーカイブ情報は編集できません。これは読み取り専用です。

アーカイブ操作の実行

ツリーのノードを右クリックして表示されるコンテキストメニューから項目を選択して，個々のアーカイブに操作を実行することもできます。次のオプションがあります。

- [Remove module] : ホスト先パーティションからアーカイブを削除します。
- [Redeploy to another Partition] : アーカイブの現在のパーティションからアーカイブを削除し，選択した別のパーティションに再配布します。
- [Download copy of module] : モジュールのコピーを別の場所に格納します。このオプションによって，モジュールはホスト環境から削除されません。
- [Run] : モジュールを起動します。
- [Edit deployment descriptor] : 配布デスクリプタエディタを起動して，編集するモジュールの配布デスクリプタをロードします。
- [Edit with archive tool] : Borland Archive Tool を起動して，アーカイブ自身の構造を変更します。

セキュリティプロファイルの使用

コンソールのホストに格納されたセキュリティプロファイルの状態を表示できます。セキュリティプロファイルの状態を表示するには，次の手順にしたがいます。

1 ナビゲーションペインで,各自のインストールのルートノードを展開します (例:C:/BDP)。

2 [Security Profiles] ノードを展開します。使用可能なプロファイルが子ノードとして表示されます。

3 状態を確認するプロファイルをクリックします。内容ペインにプロファイルの一般的な状態と SSL の状態が表示されます。

個々のセキュリティプロファイルを設定するには，セキュリティプロファイルのノードを右クリックして表示されるコンテキストメニューから [Properties] を選択します。セキュリティプロファイルの設定については,「ドメインのセキュリティプロファイル」を参照してください。

テンプレートの使用

Borland AppServer は XML を使用して，管理するシステムコンポーネントの動作を記述します。この XML は，連係して動作するコンポーネントのセット全体（「設定」）の XML とともに表示されます。AppServer はテンプレートを使用して，一般的なコンポーネントのデフォルトの動作を記述します。テンプレートは，各自のニーズに合わせて編集し，カスタマイズされたテンプレートを使って管理オブジェクトや設定を作成できます。

メモ　カスタマイズしたテンプレートは，ローカルリポジトリだけに保存します。ただし，カスタマイズしたテンプレートを適用するには，コンソールの［Hubs］ビューを使用する必要があります。カスタマイズしたテンプレートを使用するには，テンプレートに対してローカルなホストで実行されているコンソールを使用する必要があります。

パーティションテンプレート

ローカルホストに存在するパーティションの内容を表示すれば，パーティションに配布されているモジュールを確認できます。パーティションのテンプレート情報を表示および編集するには，次の手順にしたがいます。

1 ナビゲーションペインで，各自のインストールのルートノードを展開します （例：C:/BDP）。

2 ［Templates］ノードを展開します。3 つの子ノード（［Configurations］，［Managed Objects］，および［Partitions］）が表示されます。

3 ［Partitions］ノードを展開します。使用可能なパーティションのテンプレートがツリーに表示されます。

4 個々のパーティションをクリックすると，内容が表示されます。

5 パーティションのテンプレートを編集するには，内容ペインで template.xml ファイルを探してクリックします。編集ウィンドウが表示されます。

6 変更が完了したら，［Save］をクリックします。

設定テンプレート

設定の XML テンプレートを表示および編集するには，次の手順にしたがいます。

1 ナビゲーションペインで，各自のインストールのルートノードを展開します （例：C:/BDP）。

2 ［Templates］ノードを展開します。3 つの子ノード（［Configurations］，［Managed Objects］，および［Partitions］）が表示されます。

3 ［Configurations］ノードを展開します。さまざまなタイプの設定を表す子ノードが［Empty］テンプレートのノードとともに表示されます。

4 子ノードを展開し，編集するテンプレートを探すか，または［Empty］テンプレートを選択します。

5 個々のテンプレートを選択します。内容ペインに XML が表示されます。

6 内容ペインで変更を行います。

7 編集が終了したら，内容ペインのツールバーの左上にある［Apply Changes］アイコンをクリックします。

管理オブジェクトテンプレート

個々の管理オブジェクトの XML テンプレートを表示および編集するには，次の手順にしたがいます。

1 ナビゲーションペインで，各自のインストールのルートノードを展開します （例：C:/BDP）。

2 ［Templates］ノードを展開します。3 つの子ノード（［Configurations］，［Managed Objects］，および［Partitions］）が表示されます。

3 ［Managed Objects］のノードを展開します。さまざまなタイプの管理オブジェクトを表す子ノードが表示されます。

4 子ノードを展開し，編集するテンプレートを探します。

5 個々のテンプレートを選択します。内容ペインに XML が表示されます。

6 内容ペインで変更を行います。

7 編集が終了したら，内容ペインのツールバーの左上にある［Apply Changes］アイコンをクリックします。

第 8 章

管理コンソールウィザードの使い方

ここでは，管理コンソールのウィザードを使用する方法について説明します。これらのウィザードは，スタブの生成，モジュールの配布，モジュールとライブラリのマージ，JAR 圧縮アプリケーションおよびパッチアプリケーション，検証，J2EE モジュールの移行などを管理します。

管理コンソールの［Wizards］メニューでは，次のウィザードが利用できます。

- ［Deployment Wizard］
- ［Merge Wizard］
- ［Verify Wizard］
- ［XML Migration Wizard］
- ［Stub Generation Wizard］
- ［Remove Stubs Wizard］
- ［Apply Patch Wizard］
- ［Jar Wizard］

［Wizards］メニューから使用できるウィザードのほかに，管理コンソールには次のウィザードもあります。

- ［Registration Wizard］
- 「EJB を Web サービスとしてエキスポート」

Deployment Wizard

［Deployment Wizard］は、モジュールの配布手順を示します。。J2EE モジュールをパーティションに配布するだけでなく，別の配布ウィザードでは手動の配布に適した JAR を作成できます。すぐには配布しないものの，配布可能な JAR を用意しておきたい場合に，この機能は有効です。

配布を開始する前に，実行時データ（EJB の JNDI 名など）を指定する必要があります。実行時データは Borland 固有の XML ファイルに格納されますが，そのファイルの名前はモジュールの種類によって異なります。たとえば，EAR の場合，Borland XML ファイルの名前は `application-borland.xml` になります。このファイルは，配布する J2EE モジュー

ル内に作成および格納する必要があります。それには，DDEditor を使って J2EE モジュールを開き，必要な XML 情報を入力すると簡単です。詳細については，第 9 章「配布デスクリプタエディタの使い方」を参照してください。

1 つ以上の J2EE モジュールをサーバーに追加するには，次の手順にしたがいます。

1 管理コンソールの［Wizards］メニューで［Deployment Wizard］を選択します。

2 配布ウィザードのステップ 1 では，［Add］をクリックしてディレクトリツリーを参照し，コンテナに配布するモジュールを検索します。

3 ［Add J2EE Module］ウィンドウで，追加するモジュールを選択して［OK］をクリックします。必要なだけモジュールを追加できます。ファイルを追加すると，ウィザードに展開可能なツリーブラウザが表示され，追加したモジュールとそのコンテンツが表示されます。モジュールを削除するには，ツリーにあるそのノードアイコンを選択して［Remove］をクリックします。

4 次の項目を指定します。

- **［Each module is independent (Don't share any generated stubs)］:** デフォルトでチェックされています。このオプションでは，配布に含まれるほかのモジュールから独立して各モジュールのスタブを生成できます。
- **［Restart Partitions on deploy (cold deploy)］:** デフォルトでチェックされていません。このオプションを使用すると，ライブラリ JAR または RAR を配布して，サーバーの再起動が必要な場合にサーバーのクラスパスに追加します。配布プロセスの一環としてこの処理を実行する場合は，このオプションをオンにしてください。
- **［Verify deployment descriptors］:** デフォルトでチェックされています。このオプションは，追加されたモジュールに提供されている配布デスクリプタが正しいかどうかを確認する検証ツールを実行します。
- **［Generate stubs］:** デフォルトでチェックされています。このオプションは，スタブジェネレータを実行します。スタブの生成が必要なのは，サーバーが予想している形式にモジュールをまとめるためです。EJB の場合，クライアント側のスタブとサーバー側のスタブの両方を生成する必要があります。

5 スタブを生成したり，配布デスクリプタを検証するために追加の情報を提供するには，［Advanced Options］をクリックして，［Advanced Deployment Options］ダイアログボックスを開きます。

- ［Stub Generator］タブには，java2iiop コンパイラと javac コンパイラのクラスパス情報とコマンドライン引数を追加するオプションが含まれます。

  アーカイブを追加したり，クラスパス依存性の場所を指定するには，［Edit］をクリックして［Classpath Editor］を開きます。コマンドライン引数を指定するには，［Java2IIOP arguments］フィールドと［Javac arguments］フィールドを使用します（Java2IIOP の使い方を表示するには，［More Info］をクリックします）。

  スタブの生成が必要なのは，サーバーが予想している形式にモジュールをまとめるためです。EJB の場合，クライアント側のスタブとサーバー側のスタブの両方を生成する必要があります。Borland Enterprise Server では，スタブは，サーバー上ではなく管理コンソールから生成されます。したがって，管理コンソールがスタブの生成時に必要なクラス，つまりコンパイル時に依存する EJB 内の Bean のホームインターフェースとリモートインターフェースを使用できる必要があります。複数のモジュールを配布する場合，デフォルトでは，すべてのモジュールがスタブジェネレータのクラスパスに入れられます。そのため，作成した Bean がほかの JAR 内の Bean を参照している場合は，複数の配布を行うには，そのクラスパスにそれらの JAR を追加する必要があります。引数を必要とするファイルがある場合は，ここに引数を入力します。使い方の詳細については［More Info］ボタンをクリックし，Java2IIOP ツールについては，「Java 対応プログラマツール」を参照してください。

メモ　ここでクラスパスを追加する場合は，サーバー側でもクラスパスを追加する必要があることに注意してください。

- ［Verifier］タブには，追加モジュール用の配布デスクリプタが有効かどうかを確認するオプションが含まれます。このオプションを選択した場合，（エラーを含む）すべての警告を表示したり，J2EE モジュールが J2EE 仕様に厳密に準拠しているかどうかに関する警告メッセージを生成します。

  オプションを選択すると，指定する必要がある情報が正確にわかります。

6 配布ウィザードのステップ 2 では，モジュールの配布先のパーティションを選択します。サーバーで使用可能なパーティションが変更されたばかりの場合は，［Refresh List］をクリックします。

7 ［Finish］をクリックして，配布プロセスを終了するようにウィザードに指示します。ウィザードはファイルを解析し，その結果を［Deploying Modules］ウィンドウに表示します。このとき，モジュールの妥当性が検査され，エラーや警告があれば表示されます。配布中のパーティションがモジュールのロードに失敗すると，エラーに［Partition's Log］タブが表示されます。

1 つ以上のファイルにエラーがある場合は，次の方法で対応できます。

- 配布ウィザードを中断し，DDEditor で問題を解決する。詳細については，第 9 章「配布デスクリプタエディタの使い方」を参照してください。
- IDE ツールを使って問題を解決する。

メモ モジュールを配布する前あるいは後に，配布デスクリプタエディタを使って配布情報を必要に応じて変更できます。たとえば，この DDEditor を使ってトランザクションポリシーやエンタープライズ Bean のセキュリティロールを指定したり変更することができます。

Merge Wizard

［Merge Wizard］では，コンポーネントとモジュールをマージして JAR と EAR を作成します。ウィザードは，EJB JAR および EAR ファイルを解析し，クライアントがこのファイルにアクセスするために必要な最小のクラス群を決定します。最小のクラス群を保持する JAR ファイルまたは EAR ファイルが生成されます。また，ウィザードは，選択したファイル用にライブラリファイルを作成します。

アプリケーションをアセンブルするには，別のアーカイブにあるコンポーネントと依存性をリンクするモジュール用に配布デスクリプタを編集する必要があります。すべての依存性は，配布前にリンクしなければなりません。たとえば，WAR ファイルにある EJB の記述は，EJB JAR ファイルにある記述と一致する必要があります。

EJB のマージ中にウィザードを使用して，JNDI リファレンスと配布デスクリプタの Bean JNDI 名を一致させる場合は，可能であれば［Convert EJB JNDI references to links］を選択してください。

それらの JAR ファイルが EJB 1.1 の配布デスクリプタを保持している場合は，それらの配布デスクリプタが統合されて 1 つの配布デスクリプタになります。

マージウィザードでは，次の処理を実行できます。

- 1.2 EAR を作成するためにモジュールをマージする。
- 1.3 EAR を作成するためにモジュールをマージする。
- ライブラリをマージする。

モジュールをマージして EJB クライアント JAR ファイルおよび J2EE EAR ファイルを生成するには，次の手順にしたがいます。

1 管理コンソールの［Wizards］メニューで［Merge Wizard］を選択します。

2 ウィザードで作成するモジュールの種類および生成されるファイルの J2EE バージョンを選択し，［Next］を選択します。

3 ［Add］をクリックしてディレクトリツリーを参照し，JAR または EAR ファイルを選択します。必要なだけファイルを追加できます。

4 マージするモジュールを選択し，［OK］をクリックします。（複数のファイルを削除するには，そのファイルを選択して［Remove］をクリックします。）

5 ［Next］をクリックします。

6 マージプロセスから出力ファイルを指定します。ファイル名を入力するか，あるいは［Browse］ボタンでディレクトリを参照してください。

7 ［Finish］をクリックします。

Verify Wizard

アーカイブファイルの正当性と整合性を確認し，アプリケーションの配布に必要な要素がすべて所定の位置にあるかどうかを確認するには，検証ウィザードを使用します。アーカイブを個別に確認してエラーがないことを確認した後で，アセンブリレベルの確認でアプリケーションに組み込まれるほかのリソースを確認します。たとえば，検証ウィザードはURI（Uniform Resource Identifiers）の存在と正当性は検証しますが，EJB リンクやJNDI リンクは検証しません。

アーカイブを検証するには，次の手順にしたがいます。

1 管理コンソールの［Wizards］メニューで［Verify Wizard］を選択します。

2 ［Add］をクリックしてディレクトリツリーを参照し，検証するモジュールを選択します。必要なだけモジュールを追加できます。

3 検証するモジュールを選択し，［OK］をクリックします。（複数のファイルを削除するには，そのファイルを選択して［Remove］をクリックします。）

4 ［Role］を選択します。

- **［Developer］:** 最も低い確認レベルです。すべての XML 構文と，現在のアーカイブの種類に関連した標準または独自のキーワードがチェックされます。アーカイブファイルの整合性はチェックされますが，このレベルでは外部リソースは確認されません。
- **［Assembler］:** アーカイブを個別に確認してエラーがないことを確認した後で，アプリケーションに組み込まれたほかのリソースを確認します。たとえば，このレベルでは URI の存在と正当性は検証しますが，EJB リンクや JNDI リンクまでは検証しません。
- **［Deployer］:**（デフォルト）すべてのチェックがオンになっています。このレベルでは，アプリケーションが配布される動作環境だけでなく，EJB リンクや JNDI リンクもチェックされます。

5 必要な場合は，次の追加のオプションを指定します。

- **［Classpath］:** クラスパス情報を示すことができます。アーカイブを追加したり，クラスパス依存性の場所を指定するには，［Edit］をクリックして［Classpath Editor］を開きます。
- **［Show all warnings / Use strict (pedantic) checks］:** エラーに加えてすべての警告を表示し，厳密な検証手順を使用して，各自の J2EE モジュールが J2EE 仕様に準拠しているかどうかに関する警告メッセージを生成します。

6 ［Finish］をクリックして，検証プロセスを開始します。

エラーが見つかると，新しいウィンドウに表示されます。

XML Migration Wizard

［XML Migration Wizard］は，J2EE バージョン 1.2 仕様の EJB を含む JAR ファイルを，J2EE バージョン 1.3 仕様の EJB に移行する場合に役立ちます。また，このウィザードには

J2EE 1.3 から J2EE 1.2 にモジュールを移行する機能もあります。このウィザードは, XML 配布デスクリプタだけを移行します。1.2 配布デスクリプタ（*.ser ファイル）に保存されている Borland 以外のベンダーの固有情報は，移行されません。

メモ　XML 移行ウィザードは，CMP 1.1 を CMP 2.0 に移行しません。

J2EE 1.2 から J2EE 1.3 へモジュールを完全に移行するには, さらに追加手順を行う必要があります。

1. EJB 2.0 仕様に合致するようにソースコードを変更し，新しいクラスファイルを生成する必要があります。変更の必要がある部分の例として，1.0 と 1.1 のバージョンで異なる ejbCreate() メソッドのシグニチャがあります。
2. 配布デスクリプタエディタを使用して，初めからなかった配布情報や，元の配布デスクリプタから移行されなかった配布情報（EJB JNDI 名，トランザクション属性など）を追加したり，変更する必要があります。

メモ　iastool コマンドを使って JAR ファイルを移行することもできます。詳細については,「iastool コマンドラインユーティリティ」を参照してください。

EJB 1.2 JAR ファイルを 1.3 JAR ファイルに移行したり，1.3 JAR ファイルを 1.2 JAR ファイルにダウングレードするには，次の手順にしたがいます。

1. 管理コンソールの［Wizards］メニューで［XML Migration Wizard］を選択します。
2. ［Migrage module to J2EE 1.2］，［Migrate module to J2EE 1.3］，または［Migrate module to J2EE 1.4］を選択します。
3. ［Module to migrate］の［Browse］をクリックして, 移行するモジュールを選択します。
4. ［Select File］ダイアログボックスでの選択作業が完了したら, ［OK］をクリックします。
5. プロセスから出力するファイルを指定するには，［Output filename］にファイル名を入力するか，［Browse］ボタンをクリックしてディレクトリを参照します。
6. ［Finish］をクリックします。
7. 結果は，新しいウィンドウに表示されます。

Stub Generation Wizard

スタブの生成が必要なのは，サーバーが予想している形式にモジュールをまとめるためです。EJB の場合，クライアント側のスタブとサーバー側のスタブの両方を生成する必要があります。スタブは，サーバー上ではなく管理コンソールから生成されます。したがって，管理コンソールがスタブの生成時に必要なクラス，つまりコンパイル時に依存する Bean のホームインターフェースとリモートインターフェースを使用できる必要があります。

スタブ生成ウィザードを使用すると，次の種類のアーカイブを作成できます。

- コンテナ以外のクライアントアプリケーション（たとえば，CORBA のサーバー／クライアント）での使用に適した 1 つのクライアントのライブラリ JAR ファイルを生成します。その結果，EAR モジュールが「フラット化」されるため，標準の Java クラスローダーも使用できるようになります。
- サーバーの配布に必要な, 生成したスタブクラス（クライアント側とサーバー側の両方）だけを含む 1 つの「スタブのみ」の JAR ファイルを生成します。
- サーバーに手動で配布するために必要なすべてのスタブを含む手動配布可能な　JAR ファイルを生成します。

コンテナ以外のアプリケーション用にクライアントスタブを作成する

コンテナ以外のアプリケーション用にクライアントスタブを作成するには，次の手順にしたがいます。

1 管理コンソールの［Wizards］メニューで［Stub Generation Wizard］を選択します。

2 ［Create a client library jar file］を選択して，［Next］をクリックします。

3 ［Add］をクリックしてスタブを生成するモジュールを選択し，［Next］をクリックします。

4 出力ファイルをプロセスから指定するには，ファイル名を入力するか，［Browse］ボタンをクリックしてディレクトリを参照し，ファイル名を入力します。

5 オプションで，［Edit］ボタンをクリックして［Class Path Editor］を表示すると，［Classpath］フィールドにアーカイブとクラスパスを追加できます。

- ［Add Archive］をクリックして，JAR を参照します。
- ［Add Path］をクリックして，パスを参照します。

複数のモジュールを配布する場合，デフォルトでは，すべてのモジュールがスタブジェネレータのクラスパスに入れられます。そのため，作成した Bean がほかの JAR 内の Bean を参照している場合は，複数の配布を行うには，そのクラスパスにそれらの JAR を追加する必要があります。

6 ファイルごとに必要な java2iiop コンパイラと javac コンパイラのコマンドラインを入力することもできます。Java2IIOP の使い方の詳細については，［More Info］ボタンをクリックしてください。

7 ［Finish］をクリックします。

「スタブのみ」ライブラリ JAR ファイルを作成する

このウィザードでは，生成したスタブクラスだけを含む 1 つの JAR ファイルを生成します。

生成したスタブクラスだけを含む「スタブのみ」の JAR ファイルを生成するには，次の手順にしたがいます。

1 管理コンソールの［Wizards］メニューで［Stub Generation Wizard］を選択します。

2 ［Create a 'stubs only' library jar file］を選択して，［Next］をクリックします。

3 ［Add］をクリックしてスタブを生成するモジュールを選択し，［Next］をクリックします。

4 サーバー側のスタブファイルを生成しない場合は，［Generate client stubs only］をチェックします。

5 出力ファイルをプロセスから指定するには，ファイル名を入力するか，［Browse］ボタンをクリックしてディレクトリを参照し，ファイル名を入力します。

6 JAR または Bean が依存するクラスパスを［Classpath］フィールドに入力するか，［Edit］をクリックして［ClassPath Editor］を表示して選択します。

7 ファイルごとに必要な java2iiop コンパイラと javac コンパイラのコマンドラインを入力することもできます。Java2IIOP の使い方の詳細については，［More Info］ボタンをクリックしてください。

8 ［Finish］をクリックします。

手動の配布に適した JAR を作成する

スタブ生成ウィザードは，サーバーへの手動配布に必要なすべてのスタブを含む JAR ファイルを生成します。配布可能な JAR を作成しておき，後で配布するような場合に，この機能が役に立ちます。

サーバー側で配布可能な手動配布用の JAR を作成するには，次の手順にしたがいます。

1 管理コンソールの［Wizards］メニューで［Stub Generation Wizard］を選択します。

2 ［Create archive files for manual deployment］を選択して，［Next］をクリックします。

3 ［Add］をクリックしてスタブを生成するモジュールを選択し，［Next］をクリックします。

4 出力ファイルをプロセスから指定するには，ファイル名を入力するか，［Browse］ボタンをクリックしてディレクトリを参照し，ファイル名を入力します。

5 JAR または Bean が依存するクラスパスを［Classpath］フィールドに入力するか，［Edit］をクリックして［ClassPath Editor］を表示して選択します。

6 ファイルごとに必要な java2iiop コンパイラと javac コンパイラのコマンドラインを入力することもできます。Java2IIOP の使い方の詳細については，［More Info］ボタンをクリックしてください。

7 ［Finish］をクリックします。

Remove Stubs Wizard

［Remove Stubs Wizard］を使用して，JAR ファイルから既存のスタブを削除できます。

JAR ファイルから既存のスタブを削除するには，次の手順にしたがいます。

1 管理コンソールの［Wizards］メニューで［Remove Stubs Wizard］を選択します。

2 ［Add］をクリックして，スタブを除去するモジュールを選択します。

3 ［Overwrite the original modules］を選択すると，修正された各 JAR には，このディレクトリで元のモジュールと同じ名前が付けられます。または，モジュールを上書きせず出力ファイルを指定する場合は，［Browse］ボタンをクリックしてディレクトリを参照してから，ファイル名を入力します。

4 ［Finish］をクリックします。

Apply Patch Wizard

［Apply Patch Wizard］を使用すると，1 つまたは一連のパッチを JAR ファイルに適用することができます。

パッチを JAR ファイルに適用するには，次の手順にしたがいます。

1 管理コンソールの［Wizards］メニューで［Apply Patch Wizard］を選択します。

2 パッチを適用する JAR ファイル名を入力するか，[Browse]ボタンをクリックしてディレクトリを参照してファイルを選択します。

3 ［Add］をクリックして，適用するパッチが含まれるモジュールを選択します。

4 このプロセスの結果作成された出力ファイル名を入力するか，［Browse］ボタンをクリックしてディレクトリを参照してファイルを選択します。

5 ［Finish］をクリックします。

Jar Wizard

JAR ファイル内のファイルを圧縮または非圧縮するには，JAR ウィザードを使用します。

JAR ファイルの圧縮と非圧縮を実行するには，次の手順にしたがいます。

1 管理コンソールの［Wizards］メニューで［Jar Wizard］を選択します。

2 ［Compress input jar］または［Decompress input jar］を選択します。

3 ［Input jar filename］に圧縮または非圧縮する JAR ファイル名を入力するか，［Browse］ボタンをクリックしてディレクトリを参照してファイルを選択します。

4 このプロセスの結果作成された出力ファイル名を入力するか，［Browse］ボタンをクリックしてディレクトリを参照してファイルを選択します。

5 ［Finish］をクリックします。

EJB を Web サービスとしてエキスポートウィザード

ステートレスセッション Bean を Web サービスとしてエキスポートするには，EJB を Web サービスとしてエキスポートウィザードを使用します。完了すると，ウィザードは，サービスを生成および配布するための要求を自動的にサーバーに送信します。

メモ 以下の例では，*install_dir*/examples/webservices/ejb ディレクトリにある Animal サンプルを使用しています。最初に ant ユーティリティを使用して，animal_ejb.jar ファイルと Bean を作成する必要があります。次に配布ウィザードまたは iastool ユーティリティ deploy ツールを使用して，任意のパーティションに animal_ejb.jar を配布する必要があります。

配布ウィザードの詳細については，59 ページの「Deployment Wizard」を参照してください。iastool deploy ツールの詳細については，『Borland AppServer *開発者ガイド*』の「deploy」を参照してください。ant ユーティリティと Web サービスのサンプルの使い方については，『Borland AppServer *開発者ガイド*』の「Borland Web サービスのサンプル」を参照してください。

1 管理コンソールで，［Hubs］ビューが表示されていない場合は，選択します。

2 配布するモジュールを検索し，ノードを展開して，モジュールが含まれる Bean を表示します。

3 ステートレスセッション Bean を右クリックし，［Export EJB as a Web Service］をクリックします。

4 作成および配布する Web サービスと WAR ファイルの名前を入力します。Animal の例では，Web サービス名は Animal，WAR 名は animal_ejb です。ここで，デフォルトの設定のまま Web サービスを作成するには，［Finish］をクリックします。または，情報を追加する場合は，［Next］をクリックします。

5 メソッドの選択画面で，Web サービスで利用するリモートインターフェースメソッドをリストから選択します。たとえば，Animal の例では，sleep メソッドと talk メソッドが選択されています。ここで，デフォルトの設定のまま Web サービスを作成するには，［Finish］をクリックします。または，情報を追加する場合は，［Next］をクリックします。

6 XML の編集画面では，必要に応じて XML 配布デスクリプタファイルを編集し，デフォルトの設定のまま Web サービスを作成するには，［Finish］をクリックします。または，情報を追加する場合は，［Next］をクリックします。

7 スタブ画面の生成に必要な追加オプションで，JAR または Bean が依存するクラスパスを［Classpath］フィールドに入力するか，［Edit］をクリックして［ClassPath Editor］を開きます。ファイルごとに必要な java2iiop コンパイラと javac コンパイラのコマンドラインを入力することもできます。Java2IIOP の使い方の詳細については，［More

Info］ボタンをクリックし，Java2IIOP については，『VisiBroker for Java *開発者ガイド*』の「Java 対応プログラマツール」を参照してください。

8 終了したら，［Finish］をクリックします。

9 処理が終了したら，［Close］をクリックします。

10 配布が成功したかどうかを確認するには，ナビゲーションツリーにある新しく作成された War ファイルをクリックし，管理コンソールの右側で内容ペインの下にある［Additional Web Module Properties］ボックスから URL を選択し，［Test URL］をクリックします。必要に応じて，内容ペインの右側にあるスクロールバーを使用して，［Additional Web Module Properties］ボックスを表示してください。

11 WSDL（Web Service Description Language）ファイルを確認するには，EJB の Web サービスに対応する WSDL ファイルの絶対アドレスを Web ブラウザのアドレスフィールドに入力します。たとえば，`services/Animal?wsdl` と入力して次のようなアドレスを指定します。

```
http://myserver:8080/animal_ejb/services/Animal?wsdl
```

第 9 章

配布デスクリプタエディタの使い方

Borland 管理コンソールには，配布デスクリプタエディタ（DDEditor）が搭載されています。このエディタを使用すると，J2EE アーカイブオブジェクトの配布デスクリプタファイル内の配布情報を変更したり，追加することができます。Borland AppServer（AppServer）では，全部で 3 種類の DDEditor が利用できます。Borland AppServer は J2EE 1.4 に準拠していますが，J2EE 1.3 デスクリプタとの下位互換性もあります。編集できるオブジェクトは次のとおりです。

デスクリプタ（一般的な XML ファイル，および Borland 固有 XML ファイルの生成）

- EAR（application.xml を生成）
- WAR（web.xml を生成）
- EJB 1.1 JAR（ejb-jar.xml を生成）
- EJB 2.0 JAR（ejb-jar.xml を生成）
- アプリケーションクライアント JAR（application-client.xml を生成）
- JNDI 定義（Borland 固有の製品）

アーカイブ

- J2EE 1.3 アプリケーション EAR
- J2EE 1.2 アプリケーション EAR
- Servlets 2.2 WARs
- Servlets 2.3 WARs
- JNDI 定義 - DAR（Borland 固有の製品）
- リソースアダプタアーカイブ（1.0）
- アプリケーションクライアント JAR（1.2 および 1.3）
- EJB JAR（1.1 および 2.0）

この章では，配布デスクリプタファイルに含まれるコンポーネント情報の概要と，Borland AppServer の配布デスクリプタエディタの使い方について説明します。また，配布デスクリプタファイル内の編集可能なデスクリプタ情報のリストを示します。

アーカイブディレクトリ構造

アプリケーションは, 指定されたディレクトリ構造内で開発されるので, J2EE サーバーで, アーカイブして配布することができます。クラス，ファイル，サーブレット，およびその他のリソースは，すべてディレクトリ内で階層的に整理されます。

配布デスクリプタについて

J2EE アプリケーションは，1 つ以上の J2EE コンポーネントと 1 つの J2EE アプリケーション配布デスクリプタで構成されます。配布デスクリプタは，アプリケーションのコンポーネントをいくつかのモジュールとして記述します。

配布デスクリプタは 1 つの XML ファイルであり，これにより，配布されるアーカイブ（モジュール）内の単位を記述します。配布デスクリプタで使用される情報は，AppServer の配布に必要です。配布デスクリプタがアプリケーションの配布方法をデプロイヤに指示します。J2EE 仕様では，`ejb-jar.xml` デスクリプタや `web-app.xml` デスクリプタなどの配布デスクリプタが定義されています。Borland は，追加の配布デスクリプタに，AppServer でコンポーネントを配布するために必要な情報を提供します。

AppServer の DDEditor ツールを使用すると，配布デスクリプタを編集できます。JBuilder などのコード生成ツールでは配布デスクリプタを自動的に生成できますが，自分でコードを作成することもできます。

使用の条件

DDEditor は，さまざまな種類のアーカイブやデスクリプタに対して使用できます。それぞれの種類について，使用上の注意事項があります。

一般

1 DDEditor は，XML ベースの配布デスクリプタを自動的に作成します。したがって，XML を学ぶ必要がありません。

2 XML を埋め込んだアーカイブファイルを編集したり, XML ファイルを直接編集できます。

3 XML ファイルを直接編集する場合，基本コードの編集機能は，DDEitor に含まれています。

- 〔Ctrl〕＋〔F〕＝検索と置換
- 〔Ctrl〕＋〔G〕＝行番号に移動

4 DDEditor は，DTD が指定する Sun のセマンティクス規則に準拠しています。DDEditor が入力データにこれらの規則を実装します。

5 DDEditor は，Borland 固有の拡張を別のファイルに自動的に設定します。

6 このマニュアルでは，DDEditor の左側のペインをナビゲーションペインと呼び，右側ペインを作業ペインと呼びます。

EAR

EAR は基本的な配布単位です。ただし，その中に入れるアーカイブがなければ，EAR にあまり使い道はありません。したがって，EAR を作成する場合は，次の基本的な手順にしたがってください。

1 新しい EAR を作成します。

2 WAR，アプリケーションクライアント，EJB JAR などの構成要素を EAR に格納するか，EAR の中でこれらを作成します。

WAR

WAR は Web アプリケーション用のアーカイブで，サーブレット，JSP，HTML などのオブジェクトをサポートします。

サーブレットで分散可能なフラグを設定すると，この Web アプリケーションのサーブレットと JSP が独立した仮想マシン内で動作したり，別のサーバーにフェイルオーバーする機能を持つことが Web コンテナ（Tomcat）に通知されます。したがって，Distributable とマークされたサーブレットは，次の制約やインプリメンテーションの要件を満たす必要があります。

- サーブレット間で情報を共有するようなサーブレットのコンテキストを使用することはできません。かわりに，セッションを使用する必要があります。
- そのセッション内で共有するオブジェクトは，`serializable` を実装する必要があります。シリアライズ可能でないオブジェクトをセッションに組み込むと，引数が無効であることを示す例外が生成されます。

WAR ディレクトリ構造

この階層のルートは，Web アプリケーションのドキュメントルートを定義します。このルートディレクトリにあるファイルは，すべて `WEB-INF` ディレクトリ（ルートディレクトリの下）にあるファイル以外のクライアントに対応します。Web アプリケーション名は，Web アプリケーションのコンポーネントに対する要求を解決するために使用します。

`WEB-INF` ディレクトリにあるすべての private ファイルを置き換えます。`WEB-INF` にあるファイルはすべて private で，クライアントに対応します。

WebApplicationName/：HTML ファイルや JSP ファイルなどの静的ファイルは，このディレクトリに保存します。このディレクトリは，Web アプリケーションのドキュメントルートです。

/WEB-INF/web.xml：これは，Web アプリケーションを設定する Web アプリケーション配布デスクリプタです。

/WEB-INF/web-borland.xml：これは，Borland 固有の配布デスクリプタで，`web.xml` ファイル内の指定されたリソースを，サーバー内の別の場所に配置されたリソースにマッピングする方法を定義します。また，このデスクリプタファイルを使用すると，JSP および HTTP セッション属性を定義することもできます。

/WEB-INF/classes：サーブレットやユーティリティクラスなどのサーバー側のクラスが格納されます。

/WEB-INF/lib：Web アプリケーションで使用する JAR ファイルが格納されます。JSP タグライブラリも含まれます。

EJB

EJB に対して配布デスクリプタエディタを使用する場合は，次の注意があります。

- XML を埋め込んだ EJB JAR ファイルを編集したり，XML ファイルを直接編集できます。
- AppServer の実行中に，DDEditor をスタンドアロンアプリケーションとして起動した場合，Borland パーティションに配布されている JAR ファイルを編集することはできません。
- ［Set Classpath］ボタンは，EJB JAR ファイルに対してしか機能しません。

- エンティティ Bean は，JDBC 1 と JDBC 2 のどちらのデータソースも使用できます。JDBC 1 データソースは，EJB JAR に対してローカルです。JDBC 2 データソースは任意の EJB で使用できます。

JDBC2 データソースの使い方については，『Borland AppServer **プログラマーズガイド**』を参照してください。

JNDI 定義

JNDI 定義に対して DDEditor を使用する場合は，アプリケーションから独立して，またはアプリケーションアーカイブ（EAR）の一部として JNDI 定義を配布できることに注意してください。

エディタの起動

DDEditor は，スタンドアロンアプリケーションとして実行するか，管理コンソールの一部として実行できます。

また，コンソールのナビゲーションツリーにある［Deployed Modules］ノードからファイルを選択して，DDEditor を起動することもできます。編集するモジュールを右クリックして，コンテキストから［Edit deployment descriptor］を選択します。プロセスウィンドウが表示され，DDEdiotr が起動します。

DDEditor を起動するには，次の手順にしたがいます。

1 インストール時の選択に応じて，次のいずれかの方法でエディタを起動します。

- Windows NT の場合は［スタート］ボタンをクリックし，［すべてのプログラム］の［Borland AppServer］プログラムグループにある［Deployment Descriptor Editor］を選択します。
- コンソールを起動して，［Tools］メニューの［DDEditor］を選択します。
- AppServer の `bin` ディレクトリをコマンド検索パスに追加したら，コマンドウィンドウを開いて次のように入力します。

```
ddeditor <filename>
```

ここで，**ファイル名**は編集する XML ファイル，またはアーカイブファイルの名前です。それ以外の場合は，`bin` ディレクトリに移動して，同じコマンドを入力します。

2 必要に応じて，編集する配布デスクリプタを次の手順で選択します。

3 既存の配布デスクリプタを開くには，［File］メニューの［Open File］を選択します。

4 配布デスクリプタまたはアーカイブを新しく作成するには，［File］メニューの［New］を選択します。

5 配布デスクリプタエディタのメインウィンドウが表示されたら，タブを使用して，デスクリプタ情報を検査または追加します。ナビゲーションパネル（Borland Management Console の左側）を使用して，ファイルの検査，作成，および名前変更を行うことができます。

次の節では，配布デスクリプタで編集できる情報の種類を説明します。

デスクリプタの追加

Borland DDEditor を使用すると，各種のアーカイブ用の配布デスクリプタのほか，JNDI 定義用の配布デスクリプタも作成できます。

配布デスクリプタは，Sun Microsystems による DTD（Document Type Definition）に準拠した XML ファイルです。配布デスクリプタは，コンテナがアーカイブを配布する方法を記述したプロパティのセットを保持します。

デスクリプタの情報の種類

配布デスクリプタ内の情報は，次の 2 種類に分類できます。

- **構造情報。**アーカイブファイル（および JNDI 定義）の構造とそのモジュールを記述します。アーカイブファイルの構造情報には，「表示名」，「EJB JAR 名」，「EJB クライアント JAR 名」，「承認ドメイン」などの一般情報が含まれます。エンタープライズ Bean の構造情報は，エンタープライズ Bean の外部依存関係を宣言します。構造情報は必須で，通常，変更できません。この情報を変更すると，エンタープライズ Bean が機能しなくなる可能性があります。
- **アプリケーションのアセンブリ情報。**アーカイブファイルやエンタープライズ Bean をより大きなアプリケーション配布単位にまとめる方法と，全体としてアプリケーションと対話する方法を記述します。アセンブリレベルの情報を変更しても，モジュールの機能が損なわれることはありません。

次の節では，さまざまな配布モジュールの「構造」情報および「アセンブリ」情報について説明します。

エンタープライズ Bean の構造情報

すべてのエンタープライズ Bean で，次の情報が必要です。

- エンタープライズ Bean クラス
- エンタープライズ Bean のホームインターフェース
- エンタープライズリモートインターフェース
- エンタープライズ Bean のローカルホームインターフェース
- エンタープライズ Bean のローカルインターフェース
- エンタープライズ Bean のホーム JNDI 名
- エンタープライズ Bean のローカルホーム JNDI 名
- 環境項目
- EJB へのリファレンス（別のエンタープライズ Bean を参照する場合）
- リソース（ファクトリ）へのリファレンス（データソースを使用する場合）
- セキュリティロールリファレンス
- セキュリティ ID

エンタープライズセッション Bean では，次の情報が必要です。

- セッション Bean の状態管理の方法
- セッション Bean のトランザクションの種類

エンタープライズエンティティ Bean では，次の情報が必要です。

- エンティティ Bean の永続性タイプ
- エンティティ Bean の主キークラス
- エンティティ Bean のリエントラントな動作

コンテナ管理の永続性を持つエンタープライズエンティティ Bean では，次の情報が必要です。

- コンテナ管理のフィールド
- コンテナ管理の関係

エンタープライズ Bean のアプリケーションアセンブリ情報

配布デスクリプタでは，次のアプリケーションアセンブリ情報も指定できます。

- 複数のエンタープライズ Bean リファレンスのバインド
- セキュリティロール
- メソッド許可
- セキュリティロールリファレンスのリンク
- 除外リスト
- エンティティ Bean のコンテナトランザクション

配布デスクリプタの作成

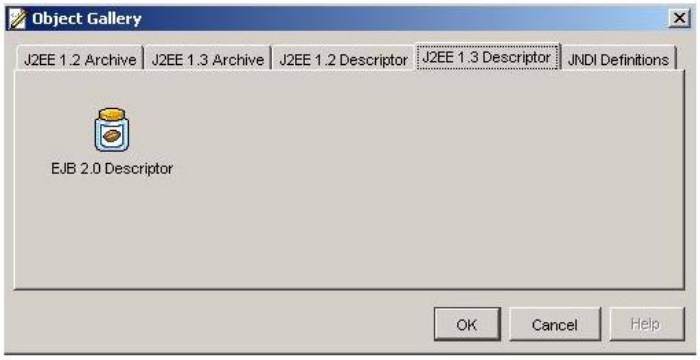
配布デスクリプタは，アーカイブの作成時に使用されるものと同じ情報を保持するため，ここでは詳細を説明しません。

定義デスクリプタを新規作成するには，次の手順にしたがいます。

1 DDEditor を開きます。

2 ［File］メニューから［New］を選択します。

3 ［Object Gallery］ウィンドウが開きます。このウィンドウにある 7 つのタブから，デスクリプタまたはアーカイブの種類を選択します。



4 デスクリプタにプロパティを割り当てます。

5 デスクリプタが完成したら，［File］メニューから［Save］を選択します。

6 フォルダを選択して，［OK］ボタンをクリックします。デスクリプタは，*Descriptor_Type*.xml として保存されます。JNDI デスクリプタ以外では，Borland 固有の XML ファイルも作成されます。

JNDI 定義デスクリプタの作成

JNDI 定義デスクリプタを作成する手順は，ほかのデスクリプタに似ています。

JNDI 定義デスクリプタを作成するには，次の手順にしたがいます。

1 DDEditor を開きます。

2 ［File］メニューから［New］を選択します。

3 Object Gallery で，［JNDI Definitions］タブを選択します。

4 ［JNDI Definitions］を選択して，［OK］をクリックします。

5 次のデータソースのいずれかを追加するには，次の手順にしたがいます。

- 新規 JDBC データソース
- 新規 JMS オブジェクト
- 新規 JNDI オブジェクト

ナビゲーションペインで jndi-definitions オブジェクトを右クリックし，オプションの 1 つを選択します。JNDI 名ウィンドウが表示されたら，（前に付加される名前を指定した後）名前を入力します。

6 デスクリプタが完成したら，［File］メニューから［Save］を選択します。デスクリプタは，jndi-definitions.xml として保存されます。

メモ 追加された［New JDBC Datasources］，［JMS］オブジェクト，および［JNDI］オブジェクトが［Standard XML］タブに表示されます。

JNDI 定義とデータソースアーカイブ（DAR）

Borland は, JDBC リソースファクトリと JMS 接続ファクトリの両方を配布するメカニズムを提供します。

AppServer の EJB コンテナにより，データソースが作成されます。コンテナで必要な作業は，アプリケーションが接続する必要があるリソースの情報を指定して，JNDI にデータソースをバインドすることだけです。データソースをバインドしたら，配布デスクリプタ内の <resource-reference> 要素を使用して，エンタープライズ Bean からデータソースを参照します。

データソースは，JNDI 定義モジュールの一部として配布されると，JNDI にバインドされます。定義モジュールは，ほかの J2EE 標準 Java アーカイブ型に似ており，拡張子 .dar で終わります。このモジュールは，DAR とも呼ばれます。DAR は，標準 J2EE モジュール型に追加して，EAR の一部としてパッケージしたり，スタンドアロンとして配布することもできます。データソースは独自のモジュールに格納されているので，AppServer の s コンソールのサーバーツリーに表示されます。（DDEditor を起動して）ツリーの表示を右クリックすれば，簡単にデータソースを有効化，無効化，および編集できます。

メモ DAR は，J2EE 仕様の一部では*ありません*。これは，Borland 固有のインプリメンテーションであり，接続ファクトリを簡単に配布したり管理することを目的としています。このアーカイブの種類内の接続ファクトリクラスはパッケージしないでください。接続ファクトリクラスは，個々のパーティションにライブラリとして配布する必要があります。

コンテナによって接続ファクトリクラスが構築されるので，DAR に提供するのは，jndi-definitions.xml と呼ばれる XML デスクリプタファイルだけです。ほかのデスクリプタと同様に，このファイルは，DAR の META-INF ディレクトリにあります。DAR は次のようになります。

```
META-INF/jndi-definitions.xml
```

デスクリプタファイルを含む DAR は, コンソールまたはコマンドラインユーティリティを使用したり, あるいは EAR の一部としてほかの J2EE モジュールを配布するように配布します。

JNDI 定義デスクリプタアーカイブ（DAR）の作成

JNDI 定義デスクリプタアーカイブ（DAR）を作成するには，次の手順にしたがいます。

1 DDEditor を開きます。

2 ［File］メニューから［New］を選択します。

3 Object Gallery で，［JNDI Definitions］タブを選択します。

4 ［JNDI Definitions Archive］を選択して，［OK］をクリックします。

5 次のデータソースのいずれかを追加するには，次の手順にしたがいます。

- 新規 JDBC データソース
- 新規 JMS オブジェクト
- 新規 JNDI オブジェクト

6 ナビゲーションペインで Untitled オブジェクトを右クリックし，オプションの 1 つを選択します。JNDI 名ウィンドウが表示されたら，（前に付加される名前を指定した後）名前を入力します。

7 アーカイブが完成したら，［File］メニューから［Save］を選択します。

8 アーカイブに名前を付け，.dar ファイルとして保存します。

データソースのプロパティを設定する

データソースを作成したら，そのデータソースのプロパティを表示および設定できます。データソースのプロパティを表示または設定するには，次の手順にしたがいます。

1 データベースを選択するには，たとえば，ナビゲーションペインで「データベース」（JDBC データソース用）を選択します。

2 JDBC データソースについては，Main，Pool Properties，および Driver Properties を設定できます。

- **［Main］:** このタブでは，よく使用される定義済みのデータソースタイプ（JDataStore，Oracle，Oracle.XA，Sybase，JDBC2 など）を選択したり，一覧にないデータソースを追加できます。また，定義済みのデータソースタイプを選択すると，そのソースに必要な，クラス名，最大プールサイズなどのその他のプロパティが提供されます。［Datasource type］，［Class name］，［Isolation Level］，［Max pool size］，［User］，［password］，［URL］などのフィールドを設定します。これらのプロパティも XML の一部で，［XML］タブで表示および編集できます。次に XML のサンプルを示します。

```
<driver-datasource>
        <jndi-name>datasources/myDriver</jndi-name>
        <datasource-class-name>oracle.jdbc.xa.client.OracleXADataSource</datasource-
class-name>
        <log-writer>False</log-writer>
        <property>
            <prop-name>driverType</prop-name>
            <prop-type>Enumerated</prop-type>
            <prop-value>thin</prop-value>
        </property>
        <property>
            <prop-name>portNumber</prop-name>
            <prop-type>Integer</prop-type>
            <prop-value>1521</prop-value>
        </property>
        <property>
```

3 **［Pool Properties］:** このタブでは，プールのプロパティである［Name］，［Type］，および［Value］を設定できます。［Type］によって，［Value］オプションが異なります。次の［Names］オプションがあります。

- ［initSQL］
- ［description］
- ［reuseStatements］
- ［busyTimeout］
- ［idleTimeout］
- ［queryTimeout］
- ［maxPoolSize］
- ［dialect］
- ［isolationLevel］
- ［resSharingScope］
- ［refreshFrequency］
- ［connectionType］
- ［waitTimeout］
- ［dbPingSQL］
- ［maxPreparedStatementCacheSize］

4 **［Driver Properties］:** このタブでは，ドライバのプロパティを設定および追加できます。

5 JMS オブジェクトについては，Main および Properties を設定できます。

- **［Main］:** このタブには，JMS オブジェクトのクラス名と，通常使用される設定済みの接続ファクトリ `QueueConnectionFactory` が表示されます。AppServer にバンドルされている Tibco JMS サービスプロバイダを使用する場合，クラス名は `com.tibco.tibjms.appserver.borland.TibjmsBorlandQueueConnectionFactory` と表示されます。
- **［Properties］:** JMS オブジェクトに必要なパラメータは，すべてここで事前に定義されます。これらのプロパティは XML `<,jndi-object>` 要素の一部で，［XML］タブで表示および編集できます。次に XML のサンプルを示します。

```
<jndi-object>
        <jndi-name>jms/message</jndi-name>
        <class-name>progress.message.jclient.QueueConnectionFactory</class-name>
        <property>
            <prop-name>brokerURL</prop-name>
            <prop-type>String</prop-type>
            <prop-value>localhost:2506</prop-value>
        </property>
        <property>
            <prop-name>sequential</prop-name>
            <prop-type>Boolean</prop-type>
            <prop-value>false</prop-value>
        </property>
        <property>
            <prop-name>loadBalancing</prop-name>
            <prop-type>Boolean</prop-type>
            <prop-value>true</prop-value>
        </property>
    </jndi-object>
```

メモ JMS サービスのベンダー固有の設定とプロパティ情報については，JMS サービスの設定に関する付録を参照してください。

メモ　2 つの JNDI 定義モジュールが，AppServer パーティションで配布済みで，サーバーのツールを使って簡単に設定できます。この 2 つのモジュールは，jdbc-datasources.dar および jms-resources.dar と呼ばれます。それぞれのモジュールにある設定済みの接続ファクトリを編集するには，サーバーツリーのモジュールを右クリックして，コンテキストメニューから［Edit deployment descriptor］を選択します。

メモ　JDBC データベースおよび接続ファクトリの定義の詳細については，『**Borland AppServer 開発者ガイド**』の「AppServer を使用したリソースへの接続：定義アーカイブ（DAR）の使い方」を参照してください。

EAR アーカイブの追加

J2EE 準拠のアプリケーションを作成する場合，EAR が配布の基礎となります。AppServer で配布用の EAR を作成する場合は，次の手順をお勧めします。

1. DDEditor を開きます。
2. ［File］メニューから［New］を選択します。
3. ［Object Gallery］ウィンドウから，［J2EE 1.3 Archive］タブの［1.3 Application Archive］，または［J2EE 1.2 Archive］タブの［1.2 Application Archive］のどちらかを選択します。
4. EAR アーカイブにプロパティを割り当てます。
5. EAR アーカイブが完成したら，［File］メニューから［Save As］を選択します。
6. アーカイブに名前を付け，拡張子を .ear にして，［OK］ボタンをクリックします。

EAR のプロパティ

いくつかの EAR アーカイブのプロパティが［Properties］ペインに表示され，作成後に編集できます。

［General］タブ

［General］タブを使用して，EAR アーカイブの一般情報を入力したり，変更を行います。次のような情報があります。

- **［Display Name］:** アプリケーションに割り当てられ，コンソールに表示される論理名。
- **［Application File Name］:** アーカイブで定義する EAR ファイルの名前と場所。
- **［Description］:** アプリケーションの目的および機能の概要。この情報は省略できます。
- **［Authorization domain］:** 承認ドメインは，一連の規則を定義して，ユーザーがロールに属すかどうかを判別します。アクセスの決定が実行されると（このマニュアルの 102 ページの「メソッド許可の割り当て」および 100 ページの「セキュリティロールとメソッド許可の追加」を参照），承認ドメインは，リソースへのアクセスに必要な論理ロールを一連の規則にマッピングします。特定のロールにユーザーが属するかどうかを判別する規則のデータベースに問い合わせます。規則は，ユーザーがロールに属するために必要な属性や対応する値を記述します。呼び出し元のセキュリティ属性が，アクセスに必要な規則を満たしているかどうかを判別するために必要な規則と，呼び出し元のセキュリティ属性とを一致させます。

  アプリケーションが使用する承認ドメインは，<authorization-domain> 配布デスクリプタ，または domain() メソッドによって決まります。authorization-domain タグを使用すると，JAR，WAR，または EAR を承認ドメインにバインドすることができます。このタグは，承認ドメインを定義するのではなく，J2EE アプリケーション内のリソースを特定の承認ドメインにバインドするだけです。

  authorization-domain 要素は Borland 固有の要素で，［Vendor XML］タブに表示されます。

[Context Roots] タブ

このタブを使用して，サーブレットに対して適切な URI 設定を指定し，適切な URI マッピングからの呼び出しであることを確認します。

- **[Web Uri]**：サーブレットの URI パターン（"helloweb.war" など）。
- **[Context Root]**：サーブレットを呼び出す Web アプリケーション名。

[Properties] タブ

このタブを使用すると，WAR に関連するさまざまな環境プロパティを設定できます。

- **[Name]**：環境変数の名前。
- **[Type]**：環境変数の型。
- **[Value]**：環境変数の値。

[XML] タブ

XML を直接編集することもできます。[Standard] タブと [Vendor] タブのどちらかを選択して編集を行います。作業が終わったら，[Apply Changes] をクリックしてアーカイブを保存します。

WAR 情報の追加

WAR アーカイブを作成するには，次の手順にしたがいます。

1. DDEditor を開きます。
2. [File] メニューから [New] を選択します。
3. [Object Gallery] ウィンドウから [J2EE 1.2 Archive] タブの [Servlets 2.2 Web Archive(WAR)]，または [J2EE 1.3 Archive] タブの [Servlets 2.3 Web Archive(WAR)]] のどちらかを選択します。
4. WAR アーカイブにプロパティを割り当てます。
5. WAR アーカイブが完成したら，[File] メニューから [Save] を選択します。
6. アーカイブに名前を付け，拡張子を .war にして，[OK] ボタンをクリックします。

WAR のプロパティ

WAR アーカイブのプロパティが [Properties] ペインに表示され，作成後に編集できます。

[General]

- **[Display Name]**：Web コンポーネントに割り当てられ，コンソールに表示される論理名（"HelloWebTier" など）。
- **[War File Name]**：アーカイブで定義する WAR ファイルの名前と場所（"helloweb.war" など）。
- **[Description]**：Web コンポーネントの目的および機能の概要。この情報は省略できます。
- **[Context Root]**：サーブレットを呼び出す Web アプリケーション名（"hello" など）。
- **[Authorization domain]**：承認ドメインに関する前の節を参照してください。
- **[Distributable]**：このコンポーネントが分散サービスコンテナ内に表示されるように適切にプログラムされているかどうかを示します。
- **[Session timeout]**：アクティブでないセッションを強制終了するまでの経過時間を指定します。

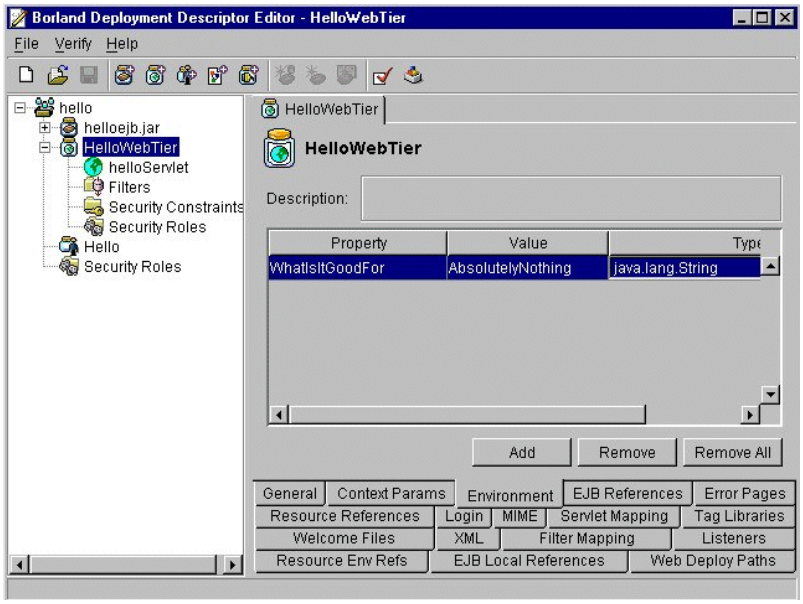
Context Params

- **[Context Parameter]**：サーブレットのコンテキスト初期化パラメータ。
- **[Value]**：サーブレットコンテキスト初期化パラメータの値。

Environment

- **[Description]**：環境変数の目的についての簡単な説明。
- **[Property]**：環境変数の名前。
- **[Value]**：環境変数の値。
- **[Type]**：環境変数の型。次のオプションがあります。
  - [java.lang.Boolean]
  - [java.lang.String]
  - [java.lang.Integer]
  - [java.lang.Character]
  - [java.lang.Double]
  - [java.lang.Byte]
  - [java.lang.Short]
  - [java.lang.Long]
  - [java.lang.Float]

図 9.1 Adding an environment property



EJB References

EJB リファレンスは，Bean を検索する JDNI ロケーションの仮名です。仮名は，使用中の Bean 内にある実際の JNDI ロケーションに対応しない場合もあります。コードは，ホームの仮名を介してホームを検索し，シンボリックリンク（リンクについては，以下参照）を使用して，仮名を JNDI ロケーションにバインドします。

Bean A が Bean B を使用する必要がある場合，EJB リファレンスで Bean B に関する必要なすべての情報を宣言するだけで，プログラム的なサンプルを使用できます。デプロイヤは，Bean A がもう 1 つ別の Bean として，Bean B を使用することを認識します。これにより，デプロイヤは，Bean A が依存しているクラスファイルと，バインドする必要がある

JNDI ロケーションを認識するようになるので便利です。コンテナのツールを使用すると，配布デスクリプタを容易に調べることができ，デプロイヤが正しく認識しているかどうかを検証できます。

```
...
<enterprise-beans>
    <session>
           <ejb-name>BeanA</ejb-name>
           <home>examples.AHome</home<
               ...
    </session>
    <session>
           <ejb-name>BeanB</ejb-name>
           <home>examples.BHome</home<
               ...
           <ejb-ref>
                 <description>
          この EJB 環境は Bean B が Bean A を使用することを定義
                 </description>
                 <ejb-ref-name>ejb/AHome</ejb-ref-name>
                 <ejb-ref-type>Session</ejb-ref-type>
                 <home>examples.AHome</home>
                 <remote>examples.A</home>
                 <ejb-link>BeanA</ejb-link>
           </ejb-ref>
       </session>
   </enterprise-beans>
```

- **［Description］:** EJB の機能についての簡単な説明。
- **［Name］:** コード内で使用する JNDI 名。
- **［IsLink］:** ローカルな Bean へのリファレンスがある場合にチェックします。そうでない場合，Bean は JNDI 名によって参照されます。
- **［Link］:** EJB リファレンスをターゲットエンタープライズ Bean にリンクします。Link 値は，ターゲットエンタープライズ Bean の名前です。IsLink をチェックした場合，この情報は必須です。IsLink をチェックしていない場合は，使用しません。
- **［Type］:** EJB の種類を指定します。次のオプションがあります。
  - ［Entity］
  - ［Session］
- **［Home］:** Bean のホームインターフェースの完全な名前。
- **［Remote］:** Bean のリモートインターフェースの完全な名前。
- **［JNDI Name］:** エンタープライズ Bean のホームインターフェースの JNDI 名。ISLink を指定した場合は使用されません。

EJB Local References

［EJB Local References］パネルでは，単一のアーカイブにあるほかのエンタープライズ Bean のホームに対するすべてのエンタープライズ Bean リファレンスを一覧表示します。

それぞれの EJB ローカルリファレンスでは，参照されるエンタープライズ Bean に対して参照しているエンタープライズ Bean が持つインターフェースの必要条件を記述します。リファレンスには，次の指定を入力します。

- **［Description］:** 参照される Bean の簡単な説明。この情報は省略できます。
- **［Name］:** 参照される Bean の名。［tool tip information］には，コード内で名前にアクセスする場合に利用する名前が表示されます。
- **［IsLink］:** これをチェックすると，ローカルエンタープライズ Bean へのリンクが設定され，［Type］フィールドに値が挿入されて読み取り専用になります。このチェックを

はずした場合，リファレンスの対象が JRA ファイルの外部にあるエンタープライズ Bean となり，［Type］フィールドに値を入力する必要があります。

- **［Link］:** EJB リファレンスをターゲットエンタープライズ Bean にリンクします。Link 値は，ターゲットエンタープライズ Bean の名前です。
- **［Type］:** 参照される Bean の予測される種類。
- **［Local Home］:** 参照される Bean のホームインターフェースの予測される Java の種類。
- **［Local］:** 参照される Bean のローカルインターフェースの予測される Java の種類。
- **［JNDI Name］:** 参照される Bean の JNDI 名。

Error Pages

- **［Type］:**エラーの種類。オプションには, "HTTP Error Code" および "Java Exception Type" があります。
- **［Error/Exception］:** エラー／例外ファイルの名前。
- **［Location］:** エラー／例外ファイルの場所。これは，Web アプリケーションのルートに対応しているので，「/」で開始する必要があります。

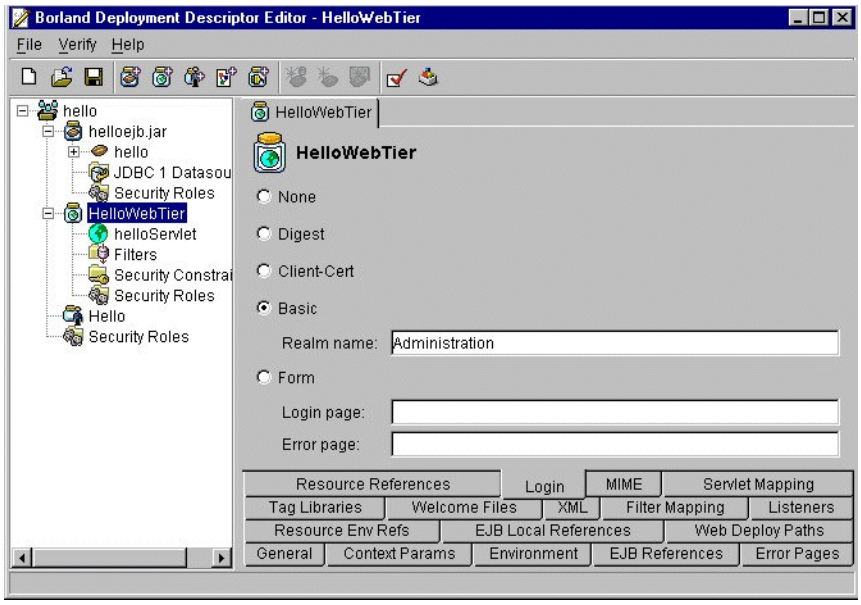
これらは，下位の要素として，<error-code> と <location> を持つ標準 XML 要素 <error-page> で表されます。

Login

このタブは，証明書を検証するための認証メソッドを選択するときに使用します。Web ブラウザクライアントがサーブレット／ JSP 層に接続されている場合, Web ブラウザのユーザーは，通常，認証をサーブレット／ JSP 層に提供し，サーブレット／ JSP 層は JAAS を使ってブラウザのユーザーを認証します。Web ブラウザは，証明書を提供し，次のいずれかのメソッドを使用して，Web コンポーネントへのログインを検証します。

- **None**
- **［Digest］:** Web クライアントは，特別なメッセージダイジェストを Web サーバーに送信します。このメッセージダイジェストは，ユーザーのパスワードと HTTP メッセージのシンボリック変換です。パスワード自体は，サーバーに送信されません。Web サーバーは，同じシンボリック変換を実行してメッセージダイジェストを再生しますが，このときサーバーは，永続的ストレージに保存されたユーザーのパスワードのセキュアコピーを使用します。メッセージダイジェストが一致すれば，ユーザーは認証されます。
- **［Client-Cert］:** クライアントは，X.509 証明書を使って ID を確立できます。オプションで，サードパーティがサーバーに扮して，サーバーを認証する X.509 証明書を受け取らないように確認することもできます。
- **［Basic］:** Web クライアントは，ユーザー名とパスワードを Web サーバーに送信します。サーバーは，ユーザー名とパスワードの永続的ストレージと証明書を比較します。認証する領域も指定する必要があります。
- **［Form］:** これは，基本的な認証に似ていますが，アプリケーションが，特別なログイン画面などのカスタマイズ可能なフォームを使用する点が異なります。ログインページとエラーページの URL も指定する必要があります。

図 9.2　フォームベースの認証のためにログインページとエラーページを設定する



MIME

これは，標準 XML 要素 <mime-mapping> を表します。mime-mapping 要素は，拡張子と MIME タイプのマッピングを定義します。2 つの対応する属性は，<extension> と <mime-type> です。

- **［Extension］:** MIME の種類を示すファイル拡張子。
- **［MIME Type］:**「text/plain」など，Web コンポーネントによって認識される定義済みの MIME タイプ。

Resource References

［Resource References］パネルには，すべてのエンタープライズ Bean リソースファクトリのリファレンスが一覧表示されます。これにより，アプリケーションアセンブラや Bean デプロイヤは，エンタープライズ Bean で使用するすべてのリファレンスを特定できます。

このパネルには，次の情報があります。

- **［Description］:** リソースリファレンスの説明です。この情報は省略できます。
- **［Name］:** エンタープライズ Bean のコードで使用する環境エントリの名前。
- **［Type］:** エンタープライズ Bean のコードが予想するリソースファクトリの Java 型。これはリソース**ファクトリ**の Java 型であり，リソースの Java 型ではありません。次の型を使用できます。
  - ［javax.sql.DataSource］
  - ［java.net.URL］
  - ［javax.mail.Session］
  - ［javax.jms.QueueConnectionFactory］
  - ［javax.jms.TopicConnectionFactory］

  指定できる配布設定（パラメータ）は，選択した Java 型によって異なります。
- **［Authentication］:** アプリケーション認証とは，エンタープライズ Bean がプログラムでリソースサインオンを実行することです。コンテナ認証とは，デプロイヤが供給するマッピング方針情報に基づき，コンテナが，Web アプリケーションのかわりに，リソースにサインオンすることです。

- **[Sharing Scope]:** <res-sharing-scope> 要素は，特定のリソースマネージャ接続ファクトリリファレンスを介して取得した接続が共有できるかどうかを指定します。この要素の値は，<Shareable> または <Unshareable> である必要があります。デフォルトは Shareable です。

Servlet Mapping

- **[Servlet]:** サーブレットの名前。
- **[URL Pattern]**: サーブレット名に URL をマッピングします。

Tag Libraries

<taglib> 要素は，JSP タグライブラリの記述に使用します。

- **[Library URI]:** パスの URI（web.xml マニュアルの場所に対応）は，onLoad で追加できます。
- **[Description File Location]:** <taglib-location> は，タグライブラリの Tag Library デスクリプタファイルのパスです。

Welcome Files

<welcome-file-list> 要素には，Welcome ファイル要素の順序一覧があります。<welcome-file> 要素には，index.html など，デフォルトの welcome ファイルとして使用するファイル名があります。

Web コンポーネントのアクセス時に最初に表示するページの URL を指定します。

XML

XML を直接編集することもできます。[Standard] タブと [Vendor] タブのどちらかを選択して編集を行います。作業が終わったら，[Apply Changes] をクリックしてアーカイブを保存します。

Filter Mappings

- **[Name]:** フィルタマッピングに使用するフィルタの論理名。Web アプリケーションの各フィルタ名は固有値です。
- **[URL Pattern]:** フィルタをサーブレットや URL パターンにマッピングするかどうかを指定します。
- **[Servlet Name]:** コンテナはフィルタマッピング宣言にしたがって，要求にどのような順番でどのフィルタを適用するかを判断します。コンテナは，通常の方法でサーブレットに対して要求 URI を照合します。適用するフィルタを決めるために，コンテナは，使用するスタイルに応じて，フィルタマッピング宣言を各フィルタマッピング要素の servlet-name または url-pattern のどちらかに一致させます。フィルタマッピングを起動する順番は，フィルタマッピングの要素リスト内の順番と同じです。フィールドに値を入力したら，生成された XML を見てフィルタマッピングが正しい順番になっているかどうかを確認することもできます。マッピングは，デスクリプタのマッピングタイプの値になります。

Listeners

[Listener Classes]: Web アプリケーションリスナー Bean として登録する必要があるアプリケーションで，1 つまたは複数のクラスを宣言します。この値は，リスナークラスの完全修飾クラス名である必要があります。

Resource Env Refs

<resource-env-ref> 要素には，アプリケーション環境内のリソースに関連付けられた管理対象のオブジェクトに対する Web アプリケーションのリファレンスの宣言があります。

[Resource Environment Ref Name]：リソース環境リファレンスの名前を指定します。このリファレンスは，Java クラスまたはインターフェースの完全修飾名である必要があります。このリファレンスは，Web アプリケーション内で一意である必要があります。

[Type]：サーブレットコードが予測するリソースマネージャの接続ファクトリの種類。これは，Java 言語クラスまたはインターフェースの完全修飾名である必要があります。

[JNDI Name]：リソースマネージャ接続ファクトリの JNDI 名。

Web Deploy Paths

［Web Deploy Paths］タブでは，Web アプリケーションを配布する場所を指定します（[Service]，［Engine]，［Host]）。Borland Web コンテナ（Tomcat ベース）では，ホストを，サービスの一部であるエンジンの一部とみなします。エンジンの下には複数のホストを設定でき，サービスの下には複数のエンジンを設定できます。Web アプリケーションは，これらホストの 1 つ以上に配布できます。この要素で指定したサービス，エンジン，ホストは，デフォルト値より優先します。ただし，この要素では，複数のエンジンを受け付けません。

デフォルトは，［Service］=HTTP，［Engine］=HTTP，［Host］=*（使用できるホストをすべて指定エンジンの下に配布）。

メモ Tomcat Web コンテナによる EAR の処理は，WAR の処理ほど簡単ではありません。IIOP コネクタで EAR の配布を処理するには，埋め込まれる WAR ファイルで IIOP コネクタへの配布を指定する必要があります。

それには，web-app_2_3-borland DTD（Borland 固有）で提供される <web-deploy-path> 要素を使用します。この要素は，Tomcat Web コンテナ「階層」内で WAR を配布する場所を指定します。

必要な XML は次のとおりです。これを自分で記述するか，DDEditor の［Web Deploy Paths］タブを使用します。

```
<web-deploy-path>
  <service>IIOP</service>
  <engine>IIOP</engine>
  <host>localhose</hose>
</web-deploy-path>
```

Web Services

DDEditor で server-config.wsdd を表示または編集するには，次のように操作します。

1 管理コンソールで，［Tools ｜ Deployment Descriptor Editor］を選択します。

Borland Deployment Description Editor が表示されます。

2 ［File ｜ Open File］を選択します。

［Open J2EE archive or descriptor］ウィンドウが表示されます。

3 WAR ファイルがあるディレクトリに移動します。

4 WAR ファイルをクリックし，［OK］をクリックします。

5 ナビゲーションツリーの最上位で，アーカイブの名前をクリックします。

6 内容ペイン右下の［XML］タブをクリックします。

7 ［Web Services］タブをクリックします。

server-config.wsdd ファイルの内容が表示されます。

8 編集します。

9 ［Apply Changes］をクリックします。

10 ［File ｜ Save］を選択します。

AppServer を使用した Web サービスの作成，編集，および配布デスクリプタの詳細については,『開発者ガイド』の「AppServer Web サービスプレビュー」を参照してください。

EJB JAR properties

EJB アーカイブのプロパティが［Properties］ペインに表示され，作成後に編集できます。

［General］

- **［Display Name］**：EJB JAR に割り当てられ，コンソールに表示される論理名。
- **［EJB JAR File Name］**：EJB Jar ファイル（読み取り専用）の名前と場所。
- **［Description］**：EJB の目的および機能の概要。この情報は省略できます。
- **［EJB Client JAR］**：EJB Jar ファイルの名前と場所。
- **［Authorization domain］**：JAR が所属する承認ドメイン名。承認ドメインは，通常，一連の定義済み役割に，リソースへのアクセスに必要な論理的な役割をマッピングします。
- **［Small icon］**：この EJB の識別に使用される 16 x 16 ピクセルのアイコンのファイル名。
- **［Large icon］**：この EJB の識別に使用される 32 x 32 ピクセルのアイコンのファイル名。

［XML］タブ

XML を直接編集することもできます。［Standard］タブと［Vendor］タブのどちらかを選択して編集を行います。作業が終わったら，［Apply Changes］をクリックしてアーカイブを保存します。

Properties

［Properties］パネル内の情報は，環境の属性やプロパティの設定に使用されます。プロパティは，オブジェクトを特定の環境で実行する方法を定義します。このパネルでプロパティを設定すると，オブジェクトのビジネスロジックをカスタマイズすることができます。これは，オブジェクトのソースコードにアクセスしたり，変更しないで，オブジェクトをアセンブルまたは配布したときに可能です。

- **［Name］**：環境変数の名前。
- **［Value］**：環境変数の値。
- **［Type］**：環境変数の型。次のオプションがあります。
  - Boolean
  - String
  - Integer

EJB Designer

EJB Designer の詳細については，107 ページの「EJB Designer」を参照してください。

アプリケーションクライアント JAR のプロパティ

アプリケーションクライアント JAR アーカイブのプロパティは［Properties］ペインに表示され，作成後に編集できます。

［General］

- **［Display Name］:** アプリケーションクライアントに割り当てられ，コンソールに表示される論理名。
- **［Application Client File Name］:** アプリケーションクライアントファイルの名前と場所。
- **［Description］:** アプリケーションクライアントの目的および機能の概要。この情報は省略できます。
- **［Small icon］:** アプリケーションクライアントの識別に使用される 16 x 16 ピクセルのアイコンのファイル名。
- **［Large icon］:** アプリケーションクライアントの識別に使用される 32 x 32 ピクセルのアイコンのファイル名。

Environment

前述の「Environment」を参照してください。

EJB References

- **［Description］:** EJB の機能についての簡単な説明。
- **［Name］:** コード内で使用する JNDI 名。
- **［IsLink］:** ローカルな Bean へのリファレンスがある場合にチェックします。そうでない場合，Bean は JNDI 名によって参照されます。
- **［Type］:** EJB の種類を指定します。次のオプションがあります。
  - ［Entity］
  - ［Session］
- **［Home］:** Bean のホームインターフェースの完全な名前。
- **［Remote］:** Bean のリモートインターフェースの完全な名前。
- **［JNDI Name］:** エンタープライズ Bean のホームインターフェースの JNDI 名。ISLink を指定した場合は使用されません。

Resource Referemces

［Resource References］パネルには，すべてのエンタープライズ Bean リソースファクトリのリファレンスが一覧表示されます。これにより，アプリケーションアセンブラや Bean デプロイヤは，エンタープライズ Bean で使用するすべてのリファレンスを特定できます。

［Resource Reference］パネルの詳細については，上の説明を参照してください。

- **［Authentication］:** アプリケーション認証とは，エンタープライズ Bean がプログラムでリソースサインオンを実行することです。コンテナ認証とは，デプロイヤが供給するマッピング方針情報に基づき，コンテナがリソースにサインオンすることです。

EJB リファレンスについては，次の点に注意してください。

- 参照先のエンタープライズ Bean は，宣言された EJB リファレンスと互換性のある型を持つ必要があります。
- すべての宣言された EJB リファレンスは，運用環境に存在するエンタープライズ Bean のホームにバインドする必要があります。

- ［Link］値を指定する場合は，そのエンタープライズ Bean リファレンスが参照先エンタープライズ Bean のホームにバインドする必要があります。

EJB ローカルリファレンス

94 ページの「［EJB Local References］パネル（EJB 2.0 のみ）」を参照してください。

［XML］タブ

XML を直接編集することもできます。［Standard］タブと［Vendor］タブのどちらかを選択して編集を行います。作業が終わったら，［Apply Changes］をクリックしてアーカイブを保存します。

Bean 情報の追加と変更

配布デスクリプタに新しい Bean を追加したり，既存の Bean を変更することができます。

メモ　ツールバーの［New Entity Bean］／［New Session Bean］／［New Session-Driven Bean］ボタンを使用するか，コンテキストメニューコマンドを右クリックして，Bean を追加します。

エンタープライズ Bean を追加するには，次の手順にしたがいます。

1 DDEditor を起動します。

2 追加する Bean の種類を選択します。

3 Bean を追加するには，ナビゲーションペインで JAR を右クリックし，コンテキストメニューから［New Entity Bean］／［New Session Bean］／［New Session-Driven Bean］を選択します。

メモ　［New Session-Driven Bean］は，EJB 2.0 でのみ追加できます。

名前を入力するダイアログボックスが表示されます。

4 エンタープライズ Bean の名前を入力して，［OK］をクリックします。

新しい Bean がナビゲーションペインに表示されます。

メモ　Bean 名は，Bean プロバイダがエンタープライズ Bean に割り当てる論理名です。各エンタープライズ Bean には，それぞれ 1 つの論理名があります。Bean の論理名と，その Bean に割り当てられる JNDI 名との間に構造上の関係はありません。Bean のデプロイヤは Bean の論理名を変更できます。

5 Bean を選択します。

エンタープライズ Bean のテンプレートが作業ペインに表示されます。このペインの下部にあるタブを使用して，Bean の情報を入力します。

クラスパス情報の設定

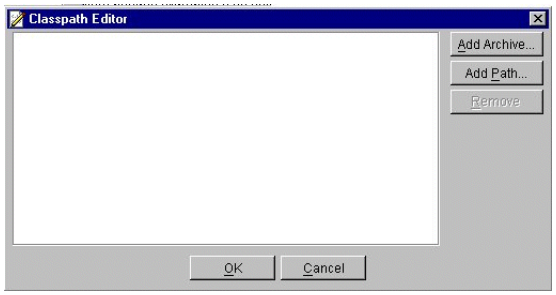
DDEditor を使ってクラスパスを設定するには，次の手順にしたがいます。

1 DDEditor を起動し，EJB JAR デスクリプタファイルまたはアーカイブを開きます。

2 ［Verify］メニューから［Set ClassPath］を選択するか，［Set ClassPath］ボタンを使用します。

Classpath Editor が表示されます。

図 9.3 Classpath Editor



3 表示されるボタンを使用して,1 つのクラスパスを 1 つ以上の JAR ファイルに追加したり，パスをクラスファイルに追加したり，アーカイブや JAR ファイルの追加や除去をすることができます。次に例を挙げます。

4 （JAR ファイル内ではなく）ファイルシステムにあるクラスの XML を編集する場合，［Add Path］ボタンでクラスパスにパスを追加する必要があります。

5 `META-INF` ディレクトリを収録しているディレクトリのクラスは，デフォルトでクラスパスに含まれます。

6 JAR にほかの JAR に対する依存性がある場合，JAR のクラスパスにそれを追加する必要があります。

7 また［JDBC 1 Datasource］パネルの［Test Connection］ボタンや［CMP 1.1］パネルの［Get Metadata］ボタンのクラスパスには，特定のドライバを追加できます。

8 完了したら，［OK］をクリックします。

Bean 情報の変更

Bean 情報を変更するには，次の手順にしたがいます。

1 DDEditor を起動して，デスクリプタファイルを開きます。

2 ナビゲーションペインで Bean を選択します。

Bean に関する詳細な説明が［Main］パネルに表示されます。このパネルの下部にあるタブを使用して，Bean の情報を表示および編集します。詳細については，次の節を参照してください。

エンタープライズ Bean の情報

ここでは，エンタープライズ Bean の配布デスクリプタファイルに作成して格納できる各種の情報について説明します。

［General］タブ

［General］パネルでは，エンタープライズ Bean（EJB 1.1 または 2.0）に関する全般的な情報を入力または変更します。次の情報を指定します。

- **［Bean class］:** Bean のビジネスメソッドを実装する Java クラスの完全な名前。この情報は必須です。
- **［Home interface］:** エンタープライズ Bean のホームインターフェースの完全な名前。この情報は必須です。
- **［Remote interface］:** エンタープライズ Bean のリモートインターフェースの完全な名前。この情報は必須です。
- **［Local home interface］:** 同じプロセス内にあるエンタープライズ Bean のローカルホームインターフェースの完全な名前。この情報は必須です。

- **［Local interface］:** 同じプロセス内にあるエンタープライズ Bean のローカルインターフェースの完全な名前。この情報は必須です。
- **［Home JNDI name］:** エンタープライズ Bean のホームインターフェースの JNDI 名。
- **［Local home JNDI name］:** エンタープライズ Bean のローカルホームインターフェースの JNDI 名。
- **［Description］:** Bean の目的および機能の概要。この情報は省略できます。

セッション Bean の場合は，次の項目も［General］パネルに表示されます。

- **［Session type］:** エンタープライズ Bean が Stateless（状態を保持しない）と Stateful（状態を保持する）のどちらであるかを指定します。
- **［Transaction type］:** Bean と Container のどちらでトランザクションポリシーを設定するかを指定します。
- **［Timeout］:** ステートフル Bean に対するトランザクションに適用されるタイムアウト。

エンティティ Bean の場合は，次の項目も［General］パネルに表示されます。

- **［Primary key class］:** エンティティ Bean の主キークラスの完全な名前。この情報は必須です。
- **［Reentrant］:** Bean がリエントラントである場合にチェックされます。ただし，予期しないマルチスレッドを防止するため，Bean をリエントラントにすることを Borland ではお勧めしていません。

  リエントラント要素は，別の Bean を介して Bean 自体を呼び出せるかどうかを命令します。たとえば，Bean A が Bean B を呼び出してから，Bean A をコールバックする場合は，Bean A はリエントラントです。これはマルチスレッドの例ですが，実際に Bean A にループバックするパスは 1 つだけです。

  別の Bean を介して Bean 自体を呼び出さない場合は，エンティティ Bean の［Reentrant］ボックスをチェックしないで，この値を「False」に設定して，予期しないマルチスレッドを防止してください。リエントラントの動作をサポートする場合は，エンティティ Bean の［Reentrant］ボックスをチェックしてこの値を「true」に設定すると，コンテナは 2 つのスレッドを Bean の中で一度に実行することができます。

メモ エンティティ Bean リエントラントを宣言する場合は，特に注意が必要です。通常，コンテナは，同一トランザクション内でのループバック呼び出しと，同一トランザクションコンテキスト内での同一エンティティ Bean に対する同時呼び出しを区別できません。

- **［Persistence type］:** ドロップダウンリストから，［container］または［Bean］を選択します。

［Environment］パネル

［Environment］パネル内の情報は，EJB 環境の属性やプロパティの設定に使用されます。プロパティは，EJB を特定の環境で実行する方法を定義します。このパネルでプロパティを設定すると，Bean のビジネスロジックをカスタマイズすることができます。これは，Bean のソースコードにアクセスしたり，変更しないで，Bean をアセンブルまたは配布したときに可能です。

各エンタープライズ Bean には，それぞれ独自の環境エントリ群が定義されています。同じエンタープライズ Bean のすべてのインスタンスは，同じ環境エントリ群を共有します。エンタープライズ Bean のインスタンスが Bean の環境を実行時に変更することはできません。

環境エントリを追加するには，次の手順にしたがいます。

1 ［Add］をクリックして，エントリを新しく作成します。

  新しい空の行がパネルに表示されます。

2 ［Property］列にプロパティを入力し，［Value］列にその値を入力します。

3 ［Type］のメニューでプロパティの型を選択します。

java.lang.String，java.lang.Integer，java.lang.Boolean，java.lang.Byte，java.lang.Short，java.lang.Long，java.lang.Double，java.lang.Character，または java.lang.Float をプロパティの型として選択できます。

4 必要に応じて，環境エントリの追加を繰り返します。

環境エントリについては，次の点に注意してください。

- Bean プロバイダは，エンタープライズ Bean のコードがアクセスするすべての環境エントリを宣言する必要があります。
- Bean プロバイダが保持している環境エントリの値は，後のアセンブリ時または配布時に変更できます。
- Bean プロバイダが設定した環境エントリの値は，アセンブリ時に変更できます。
- 配布時には，すべての環境エントリに有効な値が設定されていることを確認する必要があります。

［EJB References］パネル

［EJB References］パネルには，ほかのエンタープライズ Bean のホームへのエンタープライズ Bean リファレンスが一覧表示されます。

各 EJB リファレンスは，参照する側のエンタープライズ Bean が，参照される側のエンタープライズ Bean に対して要求するインターフェース要件を記述します。

先の 87 ページの「EJB References」を参照してください。

［Security Identity］パネル

［security identity］パネル（EJB 2.0）では，ほかの Bean からメソッドを呼び出す場合，Bean のセキュリティ ID を指定してセキュリティの伝達を制御します。EJB でメソッドを呼び出すと，コンテナは，スタブとスケルトン内のセキュリティコンテキストを暗黙的に渡して，セキュリティ情報を伝達します。たとえば，適切なセキュリティ ID で認証されたクライアントがあるとします。クライアントに呼び出された Bean A が Bean B を呼び出す場合は，Bean B がクライアントのセキュリティ ID を受け取るかどうか，また別のプリンシパルを受け取るかどうかを制御します。

このパネルには，Bean セキュリティ ID に対して 2 つのオプションがあります。

- **Use caller identity**
- **Run as**

次のサンプルは，2 つのオプションの使い方を具体的に示します。

［Use caller identity］: セキュリティ情報を配布デスクリプタで伝達する方法を制御します。次のサンプルコードは，Bean のセキュリティ ID を受け取り，呼び出されたその他すべての Bean へそのセキュリティ ID を伝達します。

```
...
<enterprise-beans>
      ...
      <session>
           <ejb-name>AccountAdministration</ejb-name>
           <home>examples.AccountAdministrationHome</home>
           ...
           <security-identity>
              <use-caller-identity/>
           </security-identity>
           ...
      </session>
```

```
        ...
    </enterprise-beans>
```

上記のように，Bean にセキュリティ ID を指定するには，[Specify Security Identity] チェックボックスをチェックします。必要に応じて識別用の簡単な説明を入れることもできます。次にドロップダウンリストから ID の種類［**Use caller identity**］を選択します。

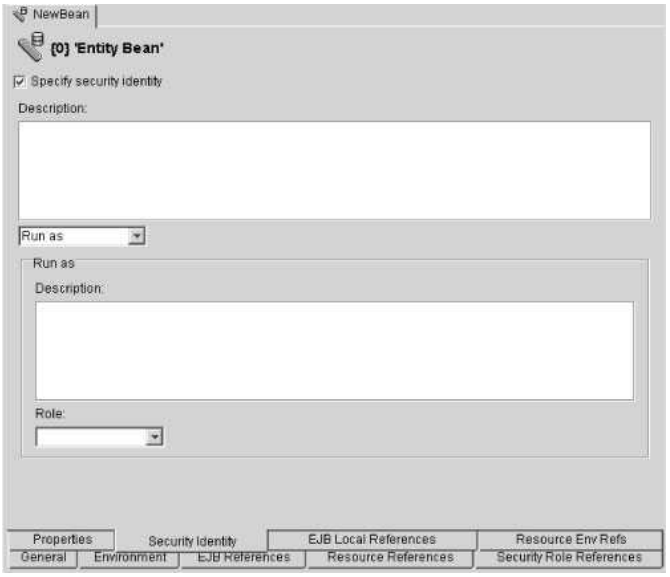
[Run as] : 上のサンプルとは対照的に，Bean を使用して，呼び出されたその他すべての Bean に「管理」などのロールを伝達するだけの場合もあります。この場合，Bean のセキュリティ ID を設定し，設定済みの <role-name> として実行します。次に，[Run as] のサンプルコードを示します。

```
...
<enterprise-beans>
       ...
       <session>
            <ejb-name>AccountAdministration</ejb-name>
            <home>examples.AccountAdministrationHome</home>
            ...
             <security-identity>
                <run-as>
                      <role-name>admin</role-name>
             </security-identity>
             ...
       <session>
<assembly-descriptor>
       ...
       <security-role>
                <description>
                   このロールはアカウント管理者用
                </description>
                <role-name>admin</role-name>
       </security-role>
          ...
</assembly-descriptor>
</enterprise-beans>
```

このセキュリティ ID オプションを選択した場合，表示された［**Role**］ドロップダウンリストでも run-as ロールを選択する必要があります。ロールは，［**Security Role References**］タブで定義されます。ロールは，[Security Roles] のナビゲーションペインで作成されます。

メモ　多くの Bean はセキュリティ ID を必要としないため，[Specify security identity] チェックボックスのチェックをはずしておく必要があります。

図 9.4　［Security Identity］パネル



[Security Role References] パネル

[Security Role References] パネルには，エンタープライズ Bean の宣言されたセキュリティロールと，その「抽象セキュリティロール」へのリンクがすべて一覧表示されます。デプロイヤは，さまざまなソース，Bean 開発者から Bean をアセンブルします。Bean 開発者は，セキュリティロールを別々に宣言している可能性があります。デプロイヤは，[Security Role References] パネルを使用して，宣言されたセキュリティロールリファレンスから，アプリケーションのアセンブラまたはデプロイヤが定義するセキュリティロールへの**リンク**を設定します。

たとえば，Bean の開発者は，AccountAdministration のロール名を administrator と定義する一方，デプロイヤはロール名として admin を使用します。この場合，デプロイヤは，アプリケーションが使用するセキュリティロールを生成する必要があります。次のサンプルコードで示すように，<role-link> 要素を使用して，この問題を解決します。

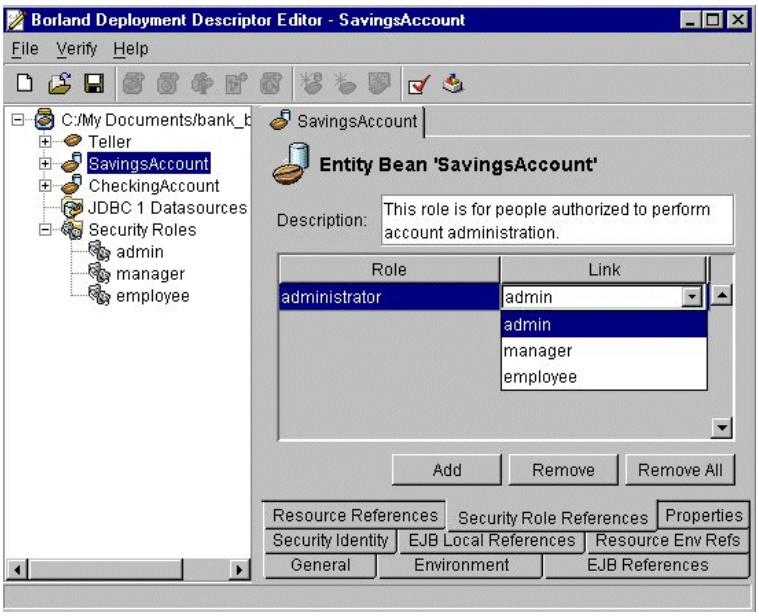
```
...
<enterprise-beans>
       ...
       <session>
            <ejb-name>AccountAdministration</ejb-name>
            <home>examples.AccountAdministrationHome</home>
               ...

            <security-role-ref>
                <description>
                    このロールはアカウント管理者用
                </description>

                <role-name>administrator</role-name>
             <role-link>admin</role-link>
            </security-role-ref>
               ...
       </session>
       <assembly-descriptor>
               ...
            <security-role>

                 <description>
                    このロールはアカウント管理者用
                 </description>
                 <role-name>admin</role-name>
            </security-role>
               ...
        </assembly-descriptor>
      </enterprise-beans>
```

セキュリティロールリファレンスのリンクを作成するには，まずセキュリティロールを作成する必要があります。**リンク**は，アプリケーションの配布に対するセキュリティロール名です。セキュリティロールの作成については，この節の 100 ページの「セキュリティロールの作成」を参照してください。

作成された配布ロールは，[Security Role References] パネルの［**Link**］フィールドの下に表示されます。

Bean 開発者が指定したセキュリティロール名にこのロールをリンクするには，Bean 開発者が［**Role**］フィールドで指定した名前を入力します。必要に応じて，ロールの説明を入力します。

図 9.5 ［Security role references］パネル



このパネルには，次の情報があります。

- **［Description］:** セキュリティロールの説明。この情報は省略できます。
- **［Role］:** Bean の開発者が指定するセキュリティロールの名前。
- **［Link］:** アプリケーション配布時に使用されるセキュリティロールの名前。通常ロールは，アプリケーションのアセンブラまたはデプロイヤが特定の運用環境を前提として定義します。

セキュリティロールリファレンスについては，次の点に注意してください。

- エンタープライズ Bean が使用するセキュリティロールリファレンスは，特定の運用環境に対応する共通名です。
- 必要に応じて，行を追加または削除できます。

［EJB Local References］パネル（EJB 2.0 のみ）

［EJB Local References］パネルでは，単一のアーカイブにあるほかのエンタープライズ Bean のホームに対するすべてのエンタープライズ Bean リファレンスを一覧表示します。

それぞれの EJB ローカルリファレンスでは，参照されるエンタープライズ Bean に対して参照しているエンタープライズ Bean が持つインターフェースの必要条件を記述します。リファレンスには，次の指定を入力します。

- **［Description］:** 参照される Bean の簡単な説明。この情報は省略できます。
- **［Name］:** 参照される Bean の名。［tool tip information］には，コード内で名前にアクセスする場合に利用する名前が表示されます。
- **［IsLink］:** これをチェックすると，ローカルエンタープライズ Bean へのリンクが設定され，［Type］フィールドに値が挿入されて読み取り専用になります。このチェックをはずした場合，リファレンスの対象が JRA ファイルの外部にあるエンタープライズ Bean となり，［Type］フィールドに値を入力する必要があります。
- **［Link］:** EJB リファレンスをターゲットエンタープライズ Bean にリンクします。Link 値は，ターゲットエンタープライズ Bean の名前です。この情報は省略できます。
- **［Type］:** 参照される Bean の予測される種類。
- **［Local Home］:** 参照される Bean のホームインターフェースの予測される Java の種類。

- **［Local］:** 参照される Bean のローカルインターフェースの予測される Java の種類。
- **［JNDI Name］:** 参照される Bean の JNDI 名。

［Resource References］パネル

［Resource References］パネルには，すべてのエンタープライズ Bean リソースファクトリのリファレンスが一覧表示されます。これにより，アプリケーションアセンブラや Bean デプロイヤは，エンタープライズ Bean で使用するすべてのリファレンスを特定できます。

このパネルには，次の情報があります。

- **［Description］:** リソースリファレンスの説明です。この情報は省略できます。
- **［Name］:** エンタープライズ Bean のコードで使用する環境エントリの名前。
- **［Type］:** エンタープライズ Bean のコードが予想するリソースファクトリの Java 型。これはリソース**ファクトリ**の Java 型であり，リソースの Java 型ではありません。次の型を使用できます。
  - ［javax.sql.DataSource］
  - ［java.net.URL］
  - ［javax.mail.Session］
  - ［javax.jms.QueueConnectionFactory］
  - ［javax.jms.TopicConnectionFactory］

  指定できる配布設定（パラメータ）は，選択した Java 型によって異なります。
- **［Authentication］:** アプリケーション認証とは，エンタープライズ Bean がプログラムでリソースサインオンを実行することです。コンテナ認証とは，デプロイヤが供給するマッピング方針情報に基づき，コンテナが，Web アプリケーションのかわりに，リソースにサインオンすることです。
- **［Sharing Scope］:** `<res-sharing-scope>` 要素は，特定のリソースマネージャ接続ファクトリリファレンスを介して取得した接続が共有できるかどうかを指定します。この要素の値は，`<Shareable>` または `<Unshareable>` である必要があります。デフォルトは `Shareable` です。

［Message-Driven Bean］パネル

EJB 2.0 アーキテクチャで MDB（メッセージ駆動型 Bean）をサポートするようになりました。メッセージ駆動型 Bean に対して，新しいメッセージ駆動型 Beay やデスクリプタを作成するように選択した場合，ナビゲーションペインで Bean をクリックすると［General Message-Driven Bean］パネルが表示されます。このパネルでは，次のメッセージ駆動型 Bean 情報を設定できます。

- **［Transaction Type］:** トランザクションを Bean 管理にするか，コンテナ管理にするかを指定します。Bean 管理のトランザクションを選択した場合は，次の Acknowlege mode のどちらかを選択する必要があります。
- **［Auto-acknowledge］:** メッセージ受信のための処理が正常終了，または返されたメッセージを処理するために `MessageListener` が呼び出された場合に，メッセージの受信を確認します。
- **［Message Selector］:** 関係のあるメッセージだけをヘッダーで指定するためにクライアントが使用する規則文字列。規則の構文は，SQL92 の条件式のサブセットに準拠します。詳細については，Sun Microsystems の JMS 仕様を参照してください。
- **［Destination］:** このプロパティは，メッセージ駆動型 Bean インスタンスがメッセージを消費する元となる JMS の送信先です。
- **［Connection Factory Name］:** メッセージブローカーとの接続を確立するために使用する接続ファクトリの JNDI 名。

- **[Destination Name]**: メッセージ駆動型 Bean が監視するキューまたはトピックの JNDI 名。
- **[Destination Type]**: 送信先がキュー (javax.jms.Queue), トピック (javax.jms.Topic), または指定されない（Not Specified）かどうかを指定します。
- **[Subscription Durability]**: MDB へのコンポーネントサブスクリプションが最初の接続の範囲を超えた状態で存続するかどうかを指定します。
- **[Initial Pool Size]**: 配布直後にコンテナで作成すべきメッセージ駆動型 Bean インスタンスの初期数を指定します。
- **[Maximum Pool Size]**: メッセージ駆動型 Bean インスタンスプールに作成し，格納できるメッセージ駆動型 Bean インスタンスの最大数を指定します。
- **[Wait Timeout]**: タイムアウトになるまでの時間を秒単位で指定します。

モジュールリファレンスエディタ

DDEditor の［References］タブには，モジュールリファレンスエディタがあります。このタブには，モジュール内の EJB JAR，WAR，およびクライアントの間の依存関係と，モジュール間のリンクまたはモジュール間の EJB の依存関係が表示されます。ここで, EAR, WAR，EJB JAR，およびクライアント JAR のレベルでリファレンスを追加したり編集することができます。

モジュールリファレンスエディタは，次のリファレンスを表示します。詳細については，凡例のヘルプを参照してください。

- ローカル EJB リンク
- リモート EJB リンク
- 外部ローカルリンクまたは JNDI リファレンス
- 外部リモートリンクまたは JNDI リファレンス
- 未解決のリンクまたは空の JNDI リファレンス
- リソースリファレンス

表示のオプションは次のとおりです。

- **[Module layout]**: EAR 内のモジュールを表示します。モジュール間の依存関係を表示します。
- **[Tiered layout]**: クライアント／サーバー層状レイアウトを表示します。左にクライアントが表示され，右にサーバー層（セッション Bean，エンティティ Bean，データリソース）が表示されます。依存関係が表示されます。
- **[Circular layout]**: WAR，セッション Bean，エンティティ Bean，クライアント MDB，リソース，およびこれらの依存関係が環状形式で表示されます。
- **[Toggle Visibility of External Views]**: 外部ノードリファレンスを表示するかどうかを切り替えることができます。［Show All］ボタンで，すべてのノードとリファレンスを表示できます。
- **[Legend]**: リファレンス凡例の表示を切り替えます。

  それぞれに，拡大，縮小，または全体表示したり，レイアウトを保存したり，デフォルトのレイアウトに戻すことができます。

モジュールリファレンスエディタの使い方

リファレンスを追加または編集するには，管理コンソールの［Deployed modules］ノードまたは［Hosted modules］ノードでモジュールを選択するか，DDEditor でモジュールを開きます。作業ペインで［References］タブを選択します。

モジュールリファレンスエディタを使用して，次の操作を行うことができます。

- EJB JNDI リファレンスの追加
- EJB リンクの追加
- リソースリファレンスの追加
- リソース環境リファレンスの追加
- EJB リファレンスの編集，およびリモートインターフェースとリモートホームインターフェースのリンク
- EJB リファレンスの編集，およびローカルインターフェースとローカルホームインターフェースのリンク
- EJB JNDI リファレンスからリンクへの変換。
- EJB リファレンスの削除

EJB JNDI リファレンスを追加するには，次の手順にしたがいます。

メモ　通常は，新しいまたは更新されたリファレンス／リンクに対して，自動的にリモートインターフェースとローカルインターフェースの情報が入力されます。省略符ボタンをクリックして，インターフェースを選択することもできます。

1. JNDI リファレンスを追加する EJB を右クリックし，［Add EJB JNDI Reference］を選択します。
2. `jndi:com/borland/examples/j2ee/hello/Hello` などの JNDI 名（場所）と，Reference1 などの Bean リファレンス名を入力します。
3. EJB がエンティティ Bean かセッション Bean かを選択します。
4. ［Remote］または［Local］を選択し，ホームインターフェースとリモートインターフェースまたはローカルホームインターフェースとローカルインターフェースを入力します。
5. ［OK］をクリックします。

EJB リンクを追加するには，次の手順にしたがいます。

1. リンクを追加する EJB を右クリックし，［Add EJB Link］を選択します。
2. リンク先の EJB に移動し，1 回クリックします。［New EJB Link］パネルが表示されます。
3. リンク先の EJB のリファレンス名を入力します（`ejb/local/CreditCard` など）。
4. リモートインターフェース情報またはローカルインターフェース情報は自動的に入力されます。省略符ボタンをクリックして，インターフェースを選択することもできます。
5. ［OK］をクリックします。

リソースリファレンスを追加するには，次の手順にしたがいます。

1. リソースリファレンスを追加する EJB を右クリックし，［Add Resource Reference］を選択します。
2. JNDI 名，リファレンス名，およびタイプを入力します（JDBC リソース，JMS 接続ファクトリなど）。［Type］ドロップダウンメニューから，データソース，JMS トピックまたはキュー，MDB，または URL を選択します。
3. 認証タイプを選択します。

- **[Application]:** エンタープライズ Bean がプログラムでリソースサインオンを実行することです。
- **[Container]**: デプロイヤが供給するマッピング方針情報に基づき，コンテナがアプリケーションのかわりにリソースにサインオンすることです。

4 共有スコープを選択します。

- **[Sharable]:** 特定のリソースマネージャ接続ファクトリリファレンスを介して取得した接続が共有できるかどうかを指定します。これは，<res-sharing-scope> 要素によって定義されます。デフォルトは Shareable です。
- **[UnSharable]:** 接続が共用できないことを示します。

5 説明はオプションです。

6 [OK] をクリックします。

リソース環境リファレンスを追加するには，次の手順にしたがいます。

1 リソース環境リファレンスを追加する EJB を右クリックし，[Add Resource Environment Reference] を選択します。

2 JNDI 名とリファレンス名を入力します。

3 [Type] ドロップダウンメニューから，JMS トピックまたはキューを選択します。

4 説明はオプションです。

5 [OK] をクリックします。

EJB JNDI リファレンスをリンクに変換するには，次の手順にしたがいます。

1 リンクに変換する JNDI リファレンス（点線で表示）を右クリックし，[Convert to Link] を選択します。

2 [To] ドロップダウンメニューから，リンク先の Bean を選択します。

インターフェース型情報に表示されるインターフェースを確認します。

3 インターフェース型情報に表示されているインターフェースを確認するか，省略符ボタンをクリックしてインターフェースを選択します。

4 [OK] をクリックします。

リファレンスを削除または編集するには，次の手順にしたがいます。

1 リファレンスまたはリンクを表す色付きの行を右クリックします。編集するには，行をダブルクリックします。

2 削除または編集を選択します。

3 [OK] をクリックします。

コンテナトランザクション

コンテナ管理のトランザクションを使用するエンタープライズ Bean は，コンテナによってトランザクションポリシーを設定してもらう必要があります。DDEditor を使用すると，コンテナ管理のトランザクション属性を設定し，これらの属性をエンタープライズ Bean のホームインターフェースとリモートインターフェースのメソッド群に関連付けることができます。

コンテナトランザクションの追加

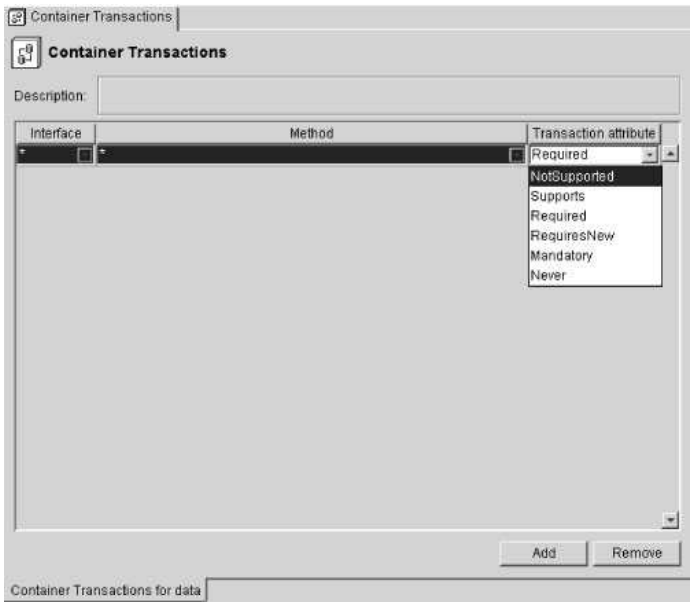
コンテナトランザクションを追加するには，次の手順にしたがいます。

1 DDEditor のナビゲーションパネルで，エンタープライズ Bean をクリックして，そのコンポーネントをすべて階層ツリーに表示します。

2 再び［navigation］パネルで，そのエンタープライズ Bean の［Container Transactions］をクリックします。

そのトランザクションの［Properties］パネルが表示されます。

図 9.6　新しいコンテナトランザクション



3 トランザクションの説明を入力します。

4 ［Interface］列のメニューから，Bean のインターフェースとして，［home］，［Remote］，［local］，［LocalHome］インターフェースのいずれかを選択するか，［*］で全部を選択します。

5 ［Method］列のメニューからメソッドを選択します。このメニューには，直前の列で指定したインターフェースのすべてのメソッドが一覧表示されます。特定のメソッドを選択するか，またはすべてのメソッドを表す［*］を選択します。

6 インターフェースとメソッドの各組み合わせに対して，［Transaction Attribute］列のメニューからトランザクション属性を選択します。

次のトランザクションポリシーがサポートされています。

表 9.1　トランザクションの属性

属性	構文	説明
Supports	TX_SUPPORTS	クライアントがトランザクションポリシーを持っている場合，Bean はそのトランザクションポリシーで呼び出されます。クライアントがトランザクションポリシーを持っていない場合，トランザクションコンテキストは設定されません。
NotSupported	TX_NOT_SUPPORTED	トランザクションコンテキストを使用することなく Bean が呼び出されます。
Never	TX_BEAN_NEVER	エンタープライズ Bean は，UserTransaction インターフェース（javax.jts）を呼び出しません。
Required	TX_REQUIRED	クライアントがトランザクションポリシーを持っている場合，Bean はそのトランザクションポリシーで呼び出されます。クライアントがトランザクションポリシーを持っていない場合，コンテナは新しいトランザクションを開始してから Bean のメソッドを呼び出します。メソッドが終了すると，コンテナがトランザクションをコミットします。
RequiresNew	TX_REQUIRES_NEW	この Bean には新しいトランザクションが必要です。この Bean は，常に新しいトランザクションの中で呼び出されます。
Mandatory	TX_MANDATORY	この Bean にはトランザクションコンテキストが必要です。クライアントがトランザクションポリシーを持っている場合，Bean はそのトランザクションポリシーで呼び出されます。クライアントがトランザクションポリシーを持っていない場合は，クライアントで例外（javax.transaction.TransactionRequiredException）が生成されます。

セキュリティロールとメソッド許可の追加

DDEditor では，配布デスクリプタ内にセキュリティロールを作成して編集できます。セキュリティロールを作成したら，エンタープライズ Bean のホームおよびリモートインターフェースのメソッドをこれらのロールに関連付けます。これにより，アプリケーションのセキュリティの概要が定義されます。

セキュリティロールについて

セキュリティロールは，EAR，WAR，および EJB JAR で見られる，論理的な役割を表す概念で，さまざまなセキュリティ環境で共通に使用されます。セキュリティロールを使用する利点は，環境ごとに独自の ID の一覧があるため，特定の ID を Bean にハードコードする必要がないことです。このため，Bean コードを再コンパイルしないで，アクセスコントロールを修正することもできます。配布時に，セキュリティロールは各運用環境で定義されている特定のユーザーグループやユーザーアカウントにマッピングされます。

ここでは，DDEditor を使ってセキュリティロールを作成し，エンタープライズ Bean のメソッド許可をロールに割り当てる方法について説明します。ただし，配布デスクリプタ内にセキュリティロールを定義するかどうかはオプションです。

セキュリティロールの作成

配布デスクリプタ内にセキュリティロールを作成するには，次の手順にしたがいます。

1. DDEditor の［navigation］パネルで，ロールを割り当てる EAR，WAR，または EJB JAR を開きます。
2. ［Security Roles］フォルダを右クリックし，コンテキストメニューで［New Role］を選択します。

   ダイアログボックスが表示されます。

3 新しいセキュリティロールの名前を入力し，［OK］をクリックします。

4 新しいセキュリティロールがナビゲーションペインに表示されます。

セキュリティロールを作成するには，次の手順にしたがいます。

1 ナビゲーションペインで［Security Role］を選択します。

そのロールのプロパティがパネルに表示されます。次のような情報があります。

［Deployment Role］: 標準の security-role-ref 要素は，security-role 要素にエリアスを定義し，アプリケーションが Bean コード内でハードコードされたロールを参照できるようにします。role-refs を使用すると，デプロイヤは，アプリケーションコードを更新するのではなく，role-refs を調整して，ロールを変更できます。

同様に，deployment-role 要素は，security-role 要素の別のエリアスです。この要素を使用すると，ユーザーは，承認ドメインで定義された配布ロールから独立して，配布デスクリプタ内でロールを保持できます。また，デプロイヤは，deployment-role を使用して，適切な承認ドメインで定義された既存のロールにセキュリティロールをマッピングできます。このとき，標準配布デスクリプタを変更する必要はありません。

メモ 承認ドメインにロールをバインドする場合は，deployment-role 要素を使用する必要はありません。security-role に deployment-role が定義されていない場合，security-role は承認ドメインで定義されるとみなされます。

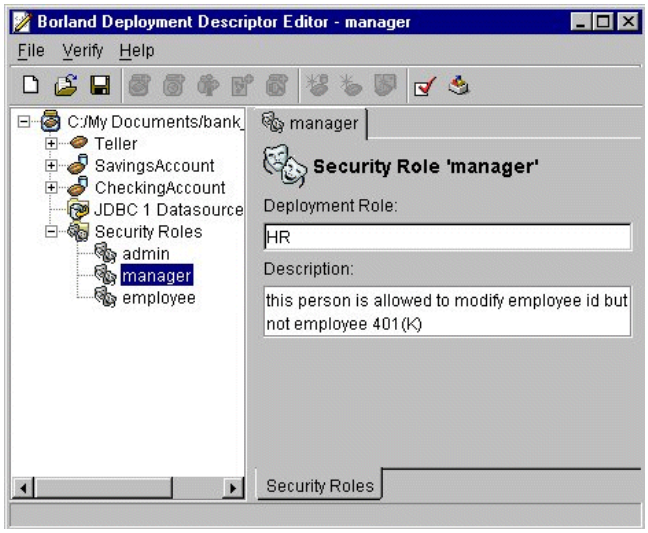
deployment-role 要素は Borland 固有の要素で，［Vendor XML］タブに表示されます。

［Description］: これは省略できますが，Bean のデプロイヤがロールの意味を理解できるように，記述することをお勧めします。

2 配布ロールを入力します。

3 ロールの説明を入力します。

図 9.7 ［Security roles ｜ properties］パネル



メソッド許可の割り当て

セキュリティロールを定義したら，エンタープライズ Bean のホームインターフェースおよびリモートインターフェースのメソッドの，どちらのメソッドをそのセキュリティロールから呼び出せるかを指定します。これにはメソッド許可が使用されます。

アプリケーションアセンブラまたは Bean 開発者が，JAR ファイル内のエンタープライズ Bean に対してセキュリティロールを定義する場合は，各セキュリティロールで起動できるホームインターフェースおよびコンポーネントインターフェースのメソッドを指定できます。ただし，セキュリティロールを Bean のホームインターフェースおよびリモートインターフェースのメソッドに関連付ける必要はありません。したがって，配布デスクリプタで定義されたセキュリティロールは，Bean インターフェース内のセキュリティロールと関連付けられない限り，これらのメソッドを呼び出すことはできません。

メソッド許可については，次の点に注意してください。

- 各 method-permission 要素には，1 つ以上のセキュリティロールの一覧（「ロール別」アクセスがある場合のみ）と 1 つ以上のメソッドの一覧が表示されます。表示されたすべてのセキュリティロールは，表示されたすべてのメソッドを呼び出すことができます。一覧にあるセキュリティはそれぞれ role-name 要素によって識別され，各メソッド（またはメソッドのセット）は method-name 要素によって識別されます。
- セキュリティロールやメソッドは，複数の method-permission 要素に表示される場合もあります。
- コンテナで呼び出される前に，承認するかどうかをチェックする必要がないメソッドを指定できます。これには，unchecked 要素を使用します。承認するかどうかをメソッドをチェックしない method-permission 要素は使用しません。
- メソッド許可が，特定のメソッドに対する unchecked 要素および 1 つ以上のセキュリティロールの両方を指定する場合は，承認するかどうかをメソッドはチェックされません。
- 呼び出されないメソッドセットを示すには，exclude-list 要素を使用します。この場合，exclude-list に含まれるメソッドにはアクセスが許可されないようにエンタープライズ Bean のセキュリティを設定する必要があります。
- 特定のメソッドが，exclude-list 要素とメソッド許可の両方で指定される場合は，メソッドにアクセスが許可されないようにエンタープライズ Bean のセキュリティを設定する必要があります。
- 同じ名前の複数のメソッドが，異なるパラメータを使用する場合は，method-params 要素を使用して，2 つを区別する許可を設定します。これは，直接 XML で，手動で編集する必要があります。

メモ 同じ名前で，異なるパラメータを持つ複数のメソッドは，[Method]プルダウンメニューに一覧表示されます。たとえば，bank_beans の SavingsAccount Bean には 2 つの create メソッドがあり，［Method］プルダウンメニューに create と create(String, float) が表示されます。

メソッド許可を割り当てるには，次の手順にしたがいます。

1. DDEditor のナビゲーションペインで，メソッド許可を割り当てる Bean を開きます。
2. Bean の階層メニューで［Method Permissions］をダブルクリックします。
3. ［Add］をクリックします。
4. ［Interface］列で，Bean のインターフェースとして［Local Home］か，［Local］または［Remote］，あるいはすべてを表す［*］を選択します。
5. ［Methods］プルダウンメニューからメソッドを選択します。このメニューには，直前の列で指定したインターフェースのすべてのメソッドが一覧表示されます。特定のメソッドを選択するか，またはすべてのメソッドを表す［*］を選択します。

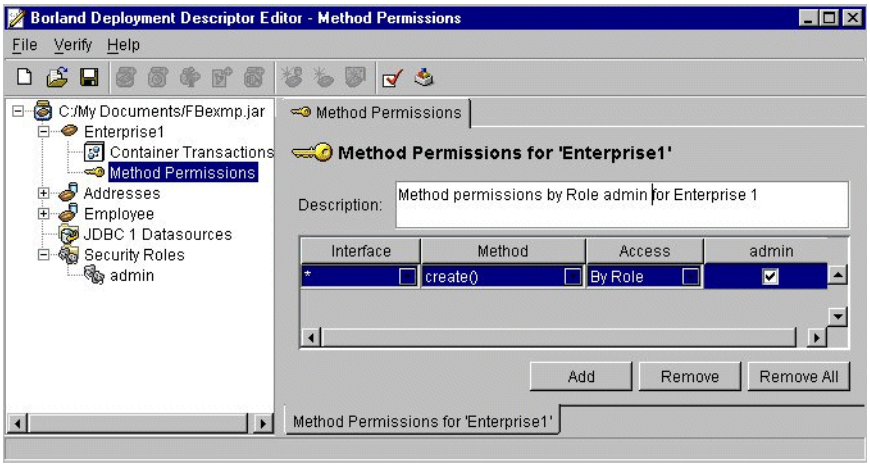
6 ［Access］プルダウンメニューから許可レベルを選択します。次の許可レベルがあります。

- **［By Role］:** 1 つ以上のセキュリティロールを選択して，選択したメソッドを呼び出すことができるロールを指定します。
- **［Unchecked］:** コンテナで呼び出される前に，承認するかどうかをチェックする必要がないメソッドを指定します。この要素は，メソッド許可内の `role-name` 要素のかわりに使用されます。
- **［Excluded］:** 呼び出されないメソッドを指定する `exclude-list` 要素の指定に使用します。

7 そのメソッドに関連付けるロールを選択します。

8 必要に応じて，Bean とメソッドの指定を繰り返します。

図 9.8　メソッド許可の指定



CMP 1.1 情報の追加

エンティティ Bean では，［General］タブで CMP 1.1 に永続性のタイプを設定して，ナビゲーションペインに CMP ノードを表示します。コンテナ管理の永続性情報を追加するには，次の手順にしたがいます。

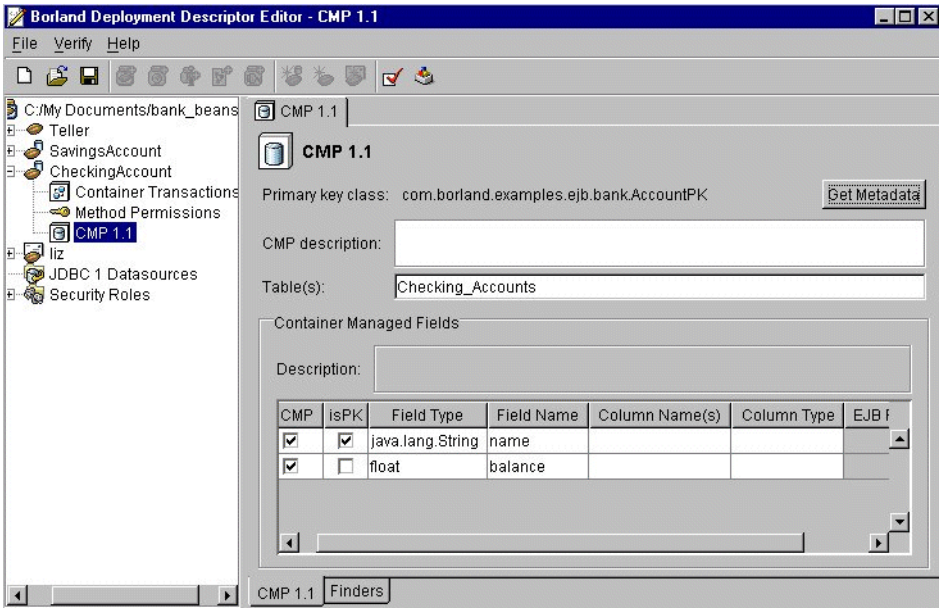
メモ　CMP バージョン (1.1, 2.0) を変更すると，以前行われた CMP の変更はすべて失われます。

1 DDEditor のナビゲーションパネルで，エンタープライズ Bean を展開して，そのコンポーネントをすべて階層ツリーに表示します。

2 CMP 1.1 要素をダブルクリックして，「CMP 1.1」ペインにプロパティを表示します。

メモ　複数のテーブルと複数の列を作業対象にできます。

図 9.9　［CMP 1.1］フィールド



Bean の永続性がコンテナ管理（1.1）の場合は，Bean のフィールドをデータベーステーブルの列にマッピングするために，次の項目が追加されます。

- **［Get Metadata］:** データベースに照会して，［Container Managed Fields］テーブルに表示される列の名前と型の一覧を取得します。
- **［CMP Description］:** コンテナ管理の永続性の説明。この情報は省略できます。
- **［Table(s)］:** Bean が参照するデータベーステーブルの名前。

コンテナ管理のフィールド

- **［Description］:** コンテナ管理のフィールドの説明。この情報は省略できます。
- **［isCMP］:** フィールドがコンテナ管理の場合にチェックされます。
- **［isPK］:** チェックボックスがチェックされている場合は，このフィールドが主キーであることを示します。また，主キークラスとこのフィールドのクラスの型が一致していることを示します。
- **［Field Type］:** フィールドのデータ型。型が EJBObject のコレクションまたはサブクラスである場合は，それに関連する EJB リファレンスが存在します。つまり，型は外部キーを参照します。コレクションの外部キーはいずれかの EJB リファレンスと一致します。そうでない場合，その型は EJB リファレンスのホームインターフェースと一致する必要があります。
- **［Field name］:** エンティティ Bean 内のフィールドの名前。

- **[Column name(s)]**: Bean の複合フィールド（location.street など）をデータベーステーブルの列にマッピングできます。ルートフィールド（location など）とサブフィールド（location.street など）のいずれかをマッピングできますが，両方をマッピングすることはできません。
- **[Column type]**: `CHAR(1)NOT NULL` などの列の型。
- **[EJB Reference]**: フィールドの型が EJB クラスである場合に，EJB リファレンスの一覧を選択肢として提示するメニューが表示されます。これらのリファレンスは，[EJB References] パネルで設定されます。有効なエントリは “None” です。

  このパネルには主キークラスが表示されますが，変更することはできません。[CMP Description] フィールドには，説明のためのテキストを自由に入力できます。

  DDEditor は JDBC を使用して，既存のテーブル内のメタデータを取得します。エンティティ Bean を既存のテーブルにフックすることができます。たとえば，購入したサードパーティのエンタープライズ Bean をデータベース内のテーブルと一緒に使用して，[Column Name] フィールドと [Column Type] フィールドの両方を格納する場合は，[Get Metadata] ボタンをクリックしてメタデータを取得します。

[Finders] パネル

[Finders] パネルは，コンテナ管理の永続性を持つ EJB エンティティ Bean に対してのみ表示されます。このパネルでは，CMP Bean が自身の検索メソッドを実行するときに使用する `WHERE` 節を指定します。

このパネルには次の項目があります。

- **[Method]**: `findAccountsLargerThan(float balance)` メソッド名とすべての引数の一覧です。
- **[Where Clause]**: `balance > :balance` など，コンテナがデータベースからレコードを取り出すときに使用する SQL の `WHERE` 節を指定します。
- **[Load State]**: `FIND` オペレーションが実行されるときに，コンテナが，すべてのコンテナ管理のフィールドを必ず事前にロードするように指定します。

CMP 2.0 情報の追加

CMP 2.0 の詳細については，107 ページの「EJB Designer」を参照してください。AppServer における CMP 2.0 の詳細については，『**開発者ガイド**』の「CMP 2.x の BES プロパティの使い方」を参照してください。

新しいデータソースの追加

DDEditor では，エンティティ Bean とコンテナに新しいデータソースを指定して，データトランザクションの分離レベルを設定できます。

新しいデータソースを追加するには，次の手順にしたがいます。

1. ナビゲーションペインで EJB JAR ファイルを選択し，それを開きます。
2. [JDBC 1 Datasources] フォルダを右クリックし，コンテキストメニューで [New JDBC1 Datasource] を選択します。

   ダイアログボックスが表示されます。
3. 新しいデータソースの JNDI 名を入力して，[OK] をクリックします。新しいデータソースがナビゲーションペインに表示されます。
4. データソースをダブルクリックします。

そのデータソースの［General］パネルが表示されます。

5 新しいデータソースに関する情報を入力します。次のような情報があります。

- **［URL］:** URL は，データベースの場所です。
- **［User name / Password］:** ソースにアクセスするために必要です。
- **［Driver class name］:** JDBC ドライバのクラス名。省略符ボタンをクリックすると，ウィンドウにドライバの一覧が表示されます。
- **［Test Connection］ボタン:** このソタンは，データソースへの接続をテストします。

メモ ドライバがクラスパス上にない場合は，データベースへの接続は失敗します。［Set Classpath］コマンドを使ってクラスパスにドライバを追加する必要があります。

データソースは，データソース名，URL ロケーション，およびそのデータソースにアクセスするためのユーザー名とパスワード（必要な場合）によって定義されます。このパネルには，JDBC ドライバのクラス名と JDBC のプロパティも表示されます。

- ポップアップメニューから分離レベルを選択します。詳細については，次の節を参照してください。

分離レベル

分離レベルとは，マルチユーザーデータベースの中にインターリーブされた複数のトランザクションが，互いの干渉を禁止される程度のことです。次のトランザクション違反が考えられます。

- **［Dirty Read］:** トランザクション t1 が行を変更し，トランザクション t2 がその行を読み取った後，t1 がロールバックを実行した場合です。t2 は存在しない行を読み取ったことになります。
- **［Non-Repeatable Read］:** トランザクション t1 が行を読み取り，トランザクション t2 がその行を更新した後，t1 が再び同じ行を読み取った場合です。トランザクション t1 は同じ行を 2 回読み取り，異なる値を取得します。
- **［Phantoms］:** トランザクション t1 が特定の検索条件を満たす行の集合を読み取り，次にトランザクション t2 がその検索条件を満たす 1 つ以上の行を挿入します。ここで再びトランザクション t1 が読み取りを繰り返すと，前回には存在しなかった行を読み取ることになります。これらの行はファントム（幻影）と呼ばれます。

配布デスクリプタで設定するトランザクション分離レベルは，これらのトランザクション違反のどれを許容するかを定義します。

表 9.2　分離レベルの属性

属性	構文	説明
Uncommitted	TRANSACTION_READ_UNCOMMITTED	3 つの違反をすべて許容します。
Committed	TRANSACTION_READ_COMMITTED	Non-Repeatable Read と Phantom を許容し，Dirty Read は許容しません。
Repeatable	TRANSACTION_REPEATABLE_READ	Phantom を許容し，ほかの 2 つは許容しません。
Serializable	TRANSACTION_SERIALIZABLE	3 つの違反をいずれも許容しません。

EJB Designer

DDEditor による EJB Designer のインプリメンテーションは，主に CMP 2.0 エンティティ Bean の永続性スキーマのデータベースへのマッピングを簡単にし，Borland 固有の Bean CMP プロパティを配布用に設定します。DDEditor の EJB Designer のインプリメンテーションでは，多くのプロパティが無効になっています。これは，プロパティがコード変更を必要とし，ソースで変更を行う必要があるからです。

EJB Designer を効果的に使用するには，CMP 2.0 プロパティを編集できるように，DDEditor にスキーマをインポートする必要があります。またはベンダー固有の XML をインポートしていない場合は，テーブルプロパティを手動で作成および編集する必要があります。

EJB Designer の詳細については，ホワイトペーパー Borland JBuilder マニュアル『DDEditor 6.6 Support for configuring EJB CMP 2.X Deployment Descriptor』を参照してください。

Borland 固有の配布デスクリプタ

Borland 固有の DTD の詳細については，オンラインヘルプの『DTD』ドキュメントを参照してください。

第 10 章

JNDI ブラウザの使い方

管理コンソールの JNDI ブラウザを使用して，JNDI を管理できます。JNDI を使用して，アプリケーションは，EJB，データソース，JMS 接続ファクトリ，送付先などのオブジェクトを名前で検索できます。JNDI は，CosNaming，LDAP，DNS，ファイルシステム，RMI レジストリなどの既存のネーミングサービスに対してシリアルプロバイダのインターフェースを定義します。シリアルプロバイダにはバインディングがあります。バインディングは，名前をオブジェクトに関連付け，オブジェクトを名前で検索できるようにします。これにより，コンポーネントどうしがアプリケーション内で互いの位置を調べることができます。

各ネーミングサービスプロバイダは, JNDI ブラウザの左側の列にあるファクトリアイコンからアクセスできます。JNDI ブラウザの各ネーミングサービスは，ナビゲーションツリーの各プロバイダノードのコンテキストメニューとツールバーアイコンを使って作成，設定，および削除できます。ツリー内の各サービスには，作業ペインに設定済みの一般プロパティが表示されます。ルートコンテキストとプロパティはここで設定します。さらに，各サービスに対して，新しいプロバイダを作成したりデフォルトのプロバイダを設定することができます。

Java リソースオブジェクトは，一般に CosNaming 名前空間に配布されます。JRO プロパティを表示するには，ツリーから Java リソースオブジェクトを選択し，［JRO Properties］タブをクリックします。このタブには，コンテナがインスタンス化した JRO に関連付けられた Java リソース定義オブジェクトのプロパティも表示されます。作業ペインの［Object Type］フィールドは，ツリーで選択されたオブジェクトが JRO かどうかを示します。

クラスパスの設定

JRO 情報を JNDI ブラウザの作業ペインに表示するには，クラスライブラリを有効にする必要があります。コンソールのクラスパスにライブラリを追加するには，次の手順にしたがいます。

1 ［File ｜ Set ClassPath］を選択します。Classpath Editor が表示されます。

2 コンソールのクラスパスにアーカイブまたは（開かれたアーカイブの）パスを追加できます。該当するボタンをクリックして，アーカイブまたは開かれたアーカイブのパスを追加します。ファイルシステムのブラウザが表示されます。

3 ファイルブラウザからアーカイブまたはパスを選択します。

4 アーカイブまたはパスの追加が完了するまで続行します。

5 ［OK］をクリックします。クラスパスが更新されます。

メモ　クラスパスを設定したら，［Refresh］ボタンをクリックする必要があります)。

サービスプロバイダビューの作成

デフォルトプロバイダのセットを作成できます。また，プロバイダを新規作成して作成時に設定することもできます。

メモ　CosNaming と Serial にはデフォルトのプロバイダが適用されます。

プロバイダのデフォルトを作成するには，次の手順にしたがいます。

1 ファクトリアイコンから該当するファクトリを選択します。

2 ナビゲーションツリーのファクトリノードを右クリックし，［Create Default Providers］を選択します。ファクトリの下のノードにデフォルトのプロバイダが表示されます。

新しいプロバイダを作成するには，次の手順にしたがいます。

1 ファクトリアイコンから該当するファクトリを選択します。

2 ナビゲーションツリーのファクトリノードを右クリックし，［New <factory> Provider］を選択します。プロバイダのプロパティの設定ダイアログが開きます。

3 プロバイダを設定します。110 ページの「サービスプロバイダビューの設定」で各プロバイダを設定するには，該当する設定のセクションを参照してください。

サービスプロバイダビューの設定

各プロバイダには，必ず設定する必要がある選択されたプロパティのセットがあります。特定のプロバイダの設定については，以下の該当するセクションを参照してください。

CosNaming

CosNaming Factory は，Borland AppServer（AppServer）上で実行中のサーバーのユーザーポート（スマートエージェントポート）をポイントします。デフォルトの VisiBroker プロバイダのツリーを展開すると，デフォルトの VisiBroker ルートコンテキストのエントリがデフォルトのスマートエージェントポート（14000）とともに表示されます。

プロバイダを設定するには，次の手順にしたがいます。

1 ツリー内のプロバイダノードをクリックします。右側のペインに［CosNaming Provider Properties］が開きます。

2 次のプロパティを設定できます。

- ［Display name］
- ［Use java.naming.provider.url as display name］
- ［Root context］
- ［Use VisiBroker CORBA ORB］
- ［Smart agent port］
- ［Use default port］
- ［Set an identity for security checks］

3 ［CosNaming Provider Properties］ダイアログの下部にあるプロパティテーブルの隣にある［Add］ボタンをクリックして，追加のプロパティを設定することもできます。

このプロバイダを AppServer プロバイダとして使用する場合は，AppServer パーティションがネーミングサービスを見つけるためのプロパティも追加する必要があります。プロバイダを AppServer ネーミングサービスとして設定するには，次の手順にしたがいます。

1 ツリー内のプロバイダノードを右クリックし，［Configure］を選択します。［CosNaming Provider Properties］ダイアログが表示されます。

2 ダイアログのプロパティセクションで，［Add］ボタンをクリックします。［Add Jndi Property］ダイアログボックスが表示されます。

3 ドロップダウンリストから，SVCnameroot プロパティを選択します。

4 ［Value］フィールドに namingservice と入力します。

5 ［OK］をクリックします。［Add Jndi Property］ダイアログボックスが閉じます。

6 ［OK］をクリックします。

サブコンテキストの作成

CosNaming Factory では，プロバイダの下にサブコンテキストを作成できます。JNDI エントリは，ファイル／サブディレクトリシステムのように，サブコンテキストの下に階層順に配置されます。サブコンテキストを使用して，エントリを整理およびグループ化できます。

サブコンテキストを作成するには，プロパティの下のコンテキストフォルダを右クリックし，［Create subcontext］を選択します。

メモ CosNaming ビューでノードを削除すると，そのエントリにアクセスできなくなります。アーカイブを再配布すると，削除（または名前変更）された一連のエントリが復元されます。通常は，再配布したアーカイブを復元するために，パーティションを再起動する必要があります。

［Serial］

Serial は，Borland 固有の AppServer 5.x と 6.0 向けのネーミングサービスです。BES 6.5 では，これまでシリアルプロバイダからサービスを受けていた JRO が CosNaming からサービスを受けます。以前のバージョンの AppServer を使用している場合，または引き続き Serial 名前空間を使用する場合は，シリアルプロバイダビューを有効にし，ネーミングサービスをホストするパーティションのプロパティを設定すれば，プロバイダの使用を継続できます。

JNDI ブラウザでシリアルプロバイダを有効にするには，次の手順にしたがいます。

1 ［File ｜ BES Serial view］を選択します。

2 プロバイダ一覧に Serial アイコンが表示されます。

JRO でシリアルプロバイダを有効にするには，次の手順にしたがいます。

1 Serial をホストするパーティションのローカルホストで Borland 管理コンソールを開きます。

2 ［Installation］ボタンをクリックします。内容ペインにローカルビューが表示されます。

3 ローカルエージェントのノードを展開します。

4 ローカルエージェントの設定ノードを探して展開します。

5 Serial をホストするパーティションを選択します。

6 内容ペインで［Files］を選択します。構造ペインに設定ファイルとプロパティファイルのリストが表示されます。

7 構造ペインから partition_server.config ファイルを選択します。編集するファイルが内容ペインに開かれます。

8 ファイルの最後に次の行を追加します。

```
vmparam -DuseSerialForResRefs=true
```

9 内容ペインで［Save Icon］をクリックします。

10 パーティションを再起動します。

Serial を無効にするには，useSerialForResRefs の値を false に変更して，パーティションを再起動します。

プロバイダを設定するには，次の手順にしたがいます。

1 ツリー内のプロバイダノードを右クリックし，［Configure］を選択します。［Serial Provider Properties］ダイアログが表示されます。

2 次のプロパティを設定できます。

- ［Display name］
- ［Use java.naming.provider.url as display name］
- ［Root context］
- ［Use VisiBroker CORBA ORB］
- ［Smart agent port］
- ［Use default port］
- ［Set an identity for security checks］

3 ［Serial Provider Properties］ダイアログの下部にあるプロパティテーブルの隣にある［Add］ボタンをクリックして，追加のプロパティを設定することもできます。

このプロバイダを AppServer プロバイダとして使用する場合は，AppServer パーティションがネーミングサービスを見つけるためのプロパティも追加する必要があります。プロバイダを AppServer ネーミングサービスとして設定するには，次の手順にしたがいます。

1 ツリー内のプロバイダノードを右クリックし，［Configure］を選択します。［Serial Provider Properties］ダイアログが表示されます。

2 ダイアログのプロパティセクションで，［Add］ボタンをクリックします。［Add Jndi Property］ダイアログボックスが表示されます。

3 ドロップダウンリストから，SVCnameroot プロパティを選択します。

4 ［Value］フィールドに namingservice と入力します。

5 ［OK］をクリックします。［Add Jndi Property］ダイアログボックスが閉じます。

6 ［OK］をクリックします。

LDAP

LDAP ネーミングサービスは，ldap://directory.company.com などの LDAP サーバーをポイントします。ルートコンテキストを指定する必要があります（o=borland など）。

プロバイダを設定するには，次の手順にしたがいます。

1 ツリー内のプロバイダノードを右クリックし，［Configure］を選択します。［LDAP Provider Properties］ダイアログが表示されます。

2 次のプロパティを設定できます。

- ［Display name］

- ［Use java.naming.provider.url as display name］
- ［Root context］

3 ［LDAP Provider Properties］ダイアログの下部にあるプロパティテーブルの隣にある［Add］ボタンをクリックして，追加のプロパティを設定することもできます。

ファイルシステム

このネーミングサービスは，特定のディレクトリのショートカットを持つ Windows エクスプローラに似ています。ツリーに表示される表示名とファイルディレクトリを指定する必要があります。

メモ ファイルシステムビューでノードを物理的に削除すると，そのノードはマシンから削除されます。ただし，プロバイダを削除した場合は，JNDI プロバイダ定義だけが削除され，そのプロバイダがポイントするファイルやディレクトリは削除されません。

ディレクトリツリーに表示されるモジュール（EAR, JAR など）を編集するには，モジュールを右クリックし，アプリケーションアセンブリツールを起動します。アプリケーションアセンブリツールの詳細については，第 11 章「アーカイブツールの使い方」を参照してください。

ネーミング定義の .binding ファイルは，JMS サービスと JDBC データソースとしてツリーに表示されます。

プロバイダを設定するには，次の手順にしたがいます。

1 ツリー内のプロバイダノード（ドライブ C:¥ など）を右クリックし，［Configure］を選択します。［File System Provider Properties］ダイアログが表示されます。

2 次のプロパティを設定できます。

- ［Display name］
- ［Use root directory as display name］
- ［Root drive and directory］

RMI レジストリ

rmi://<your_host_name>:1099) のように，ブラウザで RMI サービスを参照すると，それらの RMI レジストリエントリが表示されます。選択するポートで RMI サーバーが実行されていることを確認します。そうでない場合は，エラーが発生します。

プロバイダを設定するには，次の手順にしたがいます。

1 ツリー内のプロバイダノードを右クリックし，［Configure］を選択します。［RMI Registry Provider Properties］ダイアログが表示されます。

2 次のプロパティを設定できます。

- ［Display name］
- ［Use java.naming.provider.url as display name］
- ［Root context］

3 ［RMI Registry Provider Properties］ダイアログの下部にあるプロパティテーブルの隣にある［Add］ボタンをクリックして，追加のプロパティを設定することもできます。

ネットワーク DNS

デフォルトのネットワーク DNS ネーミングサービスは，dns://dnshost:53/domain_name.com のように，ローカルマシンを DNS サーバーとみなします。

プロバイダを設定するには，次の手順にしたがいます。

1 ツリー内のプロバイダノードを右クリックし，［Configure］を選択します。［Network DNS Provider Properties］ダイアログが表示されます。

2 次のプロパティを設定できます。

- ［Display name］
- ［Use java.naming.provider.url as display name］
- ［Root context］

3 ［Network DNS Provider Properties］ダイアログの下部にあるプロパティテーブルの隣にある［Add］ボタンをクリックして，追加のプロパティを設定することもできます。

プロバイダの削除

プロバイダを削除するには，次の手順にしたがいます。

1 プロバイダノードを右クリックして，コンテキストメニューを開きます。

2 ［Delete］をクリックします。確認のダイアログが開きます。

3 ［はい］をクリックすると，このプロバイダノードが削除されます。

第 11 章

アーカイブツールの使い方

Borland AppServer（App Server）には，アーカイブツールが含まれています。このツールを使用すると，1 つ以上のサーバーで展開されるアーカイブについて，次のようなさまざまな作業ができます。

- 「アーカイブへのファイルの追加」
- 「アーカイブファイルの編集」
- 「アーカイブからのファイルの削除」
- 「アーカイブの新規作成」
- 「アーカイブの確認」

このツールを使用して，次のアーカイブファイルを開くことができます。

- エンタープライズアプリケーションリソース（EAR）
- WEB アプリケーションアーカイブ（WAR）
- リソースアダプタアーカイブ（RAR）
- EJB JARs
- クライアント JAR
- Zip ファイル
- データソースアーカイブ（DAR），JNDI 定義が保存される Borland 固有のアーカイブ形式

このアーカイブツールを使用して，次のようなアーカイブファイルを作成できます。

- エンタープライズアプリケーションリソース（EAR）
- EJB Jars
- WEB アプリケーションアーカイブ（WAR）
- クライアント JAR
- コネクタモジュール
- ライブラリファイル
- J2EE バージョン 1.2 アーカイブ（EAR，EJB，Web およびクライアント側の JAR）

このツールを使用して，アーカイブからファイルを抽出し，アーカイブファイルをアセンブリレベルで確認することもできます。

アーカイブツールには，数多くの実用的な使い方があります。アーカイブ内の既存のファイルを表示および編集できるだけでなく，アーカイブをすばやく収集することもできます。たとえば，JAR ファイルにイメージファイルを追加し忘れた場合，アーカイブツールを使用すれば，JAR を再ビルドしなくても，そのイメージファイルを JAR ファイルに追加できます。

アーカイブツールの起動

アーカイブツールは，Borland AppServer コンソールからしか起動できません。

1 コンソールを起動します。

2 ［Tools］メニューから［Archive Tool］を選択するか，ツールバーの［Launch the Borland archive tool］アイコンをクリックします。

アーカイブツールが起動されたら，（まだ名前の付いていない）アーカイブにファイルを追加できます。各アーカイブには，作成時からマニフェストファイルがあります。マニフェストファイルは，アーカイブツールによって自動的に生成されます。

アーカイブを開く

アーカイブを開くには，次の手順にしたがいます。

1 アーカイブツールを起動します。

2 ［File］メニューから［Open］を選択します。

3 使用するアーカイブファイルに移動して選択し，［OK］をクリックします。

アーカイブが開き，その内容が表示されます。

アーカイブに対する操作の実行

アーカイブを作成したら，そのアーカイブと中のファイルに対してさまざまな操作を実行できます。

アーカイブへのファイルの追加

ファイルは，新しいアーカイブまたは既存のアーカイブに追加できます。アーカイブにファイルを追加するには，次の手順にしたがいます。

1 アーカイブツールを起動します。

2 アーカイブを開き，［Module］メニューから［Add］を選択します。

3 アーカイブに追加するファイルを選択します。

- 複数のファイルを追加するには，［Add Files］をクリックします。
- ディレクトリ内のすべてのファイルを追加するには，［Add Directory］をクリックします。

4 追加するファイルまたはディレクトリに移動して選択し，［OK］をクリックします。

5 ［Relative path］（デフォルト）と［Absolute path］のどちらかをクリックします。J2EE アーカイブを構築する場合は，［Relative path］だけを使用します。

6 アーカイブに追加するファイルを圧縮したくない理由がある場合を除いて，[Compress entries］ボックスをオン（デフォルト）のままにしておきます。

7 ディレクトリを追加する場合，追加するディレクトリ内にあるサブディレクトリとファイルもすべて追加する場合は，[Include subdirectories］ボックスをチェックしたまま（デフォルト）にします。

8 項目の追加が完了したら，［OK］をクリックします。

アーカイブファイル内のテキストファイルの編集

アーカイブに追加したテキストファイルは，アーカイブから抽出しなくても編集できます。テキストファイルを編集するには，次の手順にしたがいます。

1 アーカイブを開きます。

2 編集可能なファイルをアーカイブ内でダブルクリックして，テキストエディタウィンドウで開きます。

3 必要に応じてファイルを編集し，［Save］をクリックします。

アーカイブからのファイルの削除

アーカイブからファイルを除去するには，次の手順にしたがいます。

1 削除するファイルを選択します。

2 ［Module］メニューから［Remove］を選択します。

アーカイブの保存

アーカイブファイルを保存するには，［File］メニューから［Save］を選択します。

作成済みのアーカイブを別の名前で保存するには，次の手順にしたがいます。

1 ［File］メニューから［Save As］オプションを選択します。

2 保存するファイルの名前と場所を指定します。

3 ［Save］をクリックします。

アーカイブファイルの編集

アーカイブ内のファイルの種類を問わずに編集するには，次の手順にしたがいます。

1 アーカイブを開きます。

2 ファイルを選択し，ツールバーの［View file contents］アイコンをクリックします。

3 必要な変更を入力します。

4 ［Save］をクリックします。

アーカイブの新規作成

新しいアーカイブファイルを作成するには，次の手順にしたがいます。

1 ［File］メニューから［New］を選択します。

2 新規モジュールウィザードで，作成するアーカイブの種類と，使用する J2EE バージョンを選択して，［Next］をクリックします。

新規モジュールウィザードにメッセージが表示され，選択した種類のアーカイブに適用する追加の詳細を提供するように求められます。求められる詳細は，アーカイブの種類によって異なります。

3 詳細を提供したら，［Finish］をクリックします。

新規モジュールウィザードが選択したアーカイブの種類に必要な基本的なファイル群を作成し，名称未設定のアーカイブとして開かれます。アーカイブファイルに対する変更で保存されていないものがある場合は，新しいアーカイブを開くときにアーカイブツールは変更されたアーカイブファイルを保存するようにメッセージを表示します。

モジュールの抽出

このアプリケーションアセンブリツールを使用して，アーカイブ内の 1 つ以上のファイルを解凍できます。

アーカイブ内のモジュールを抽出するには，次の手順にしたがいます。

1 アーカイブを開きます。

2 次のどちらかの手順にしたがいます。

- すべてのモジュールを抽出するには，すべてのモジュールを未選択のままにします。
- 選択したモジュールだけを抽出するには，1 つまたは複数のモジュールをクリックします。

3 ［Module］メニューから［Extract］を選択します。

4 ［Output directory］フィールドで，抽出したファイルを保存する場所を入力するか，［Browse］ボタンを使って保存する場所を指定し，［OK］をクリックします。

アーカイブの確認

アプリケーションアセンブリツールを使ってアーカイブファイルの正当性と整合性を確認したり，アプリケーションの配布に必要な要素がすべて所定の位置にあるかどうかを確認することができます。アーカイブを個別に確認してエラーがないことを確認した後で，アセンブリレベルの確認でアプリケーションに組み込まれるほかのリソースを確認します。たとえば，アーカイブツールは URI（Uniform Resource Identifiers）の存在と正当性は検証しますが，EJB リンクや JNDI リンクまでは検証しません。

サポートされているアーカイブの種類

サポートされているアーカイブの種類 : EAR，WAR，JNDI，RAR，クライアント JAR，DAR，および ZIP です。

アーカイブの確認プロセスでは，一般に次のようなチェックが行われます。

- XML 構文に対してコードが正しいかどうかをチェックするパスオーバー
- 標準または独自の XML デスクリプタの意味と，サポートされている各アーカイブの種類に対して必要なデスクリプタの準拠性の確認

確認は，常に最上位のモジュールからその下位モジュールへと順に階層的に行われ，最後にアーカイブ間のリンクがチェックされます。

アーカイブの確認

アーカイブを検証するには，次の手順にしたがいます。

1 アーカイブを開きます。

2 ［Module］メニューから［Verify］を選択します。

3 ［Verify Module］ダイアログで，検証ロールレベルを選択します。

- **［Developer］:** 最も低い確認レベルです。すべての XML 構文と，現在のアーカイブの種類に関連した標準または独自のキーワードがチェックされます。アーカイブファイルの整合性はチェックされますが，このレベルでは外部リソースは確認されません。
- **［Assembler］:** アーカイブを個別に確認してエラーがないことを確認した後で，アプリケーションに組み込まれたほかのリソースを確認します。たとえば，このレベルでは URI の存在と正当性は検証しますが，EJB リンクや JNDI リンクまでは検証しません。
- **［Deployer］:**（デフォルト）すべてのチェックがオンになっています。このレベルでは，アプリケーションが配布される動作環境だけでなく，EJB リンクや JNDI リンクもチェックされます。

4 必要に応じて，追加のオプションを指定して，［OK］をクリックします。［Verifying］ダイアログが開き，検証プロセスの状態が表示されます。

エラーと警告が見つかった場合は，［Verifying］ダイアログに表示されます。

第 12 章

ライセンスマネージャの使い方

管理コンソールのライセンスマネージャを使用して，Borland の製品ライセンスを追加，登録，および削除できます。ここでは，ライセンスマネージャの GUI 機能の使い方について説明します。

Borland AppServer では、**ノードライセンス**を使用します。このライセンスは，ライセンスが適用およびアクティブ化されるシステムに固定化されます。このライセンスを別のシステムにコピーすることはできません。また，別のシステムで実行されている AppServer 製品からこのライセンスにアクセスすることもできません。同じシステムにソフトウェアを再インストールした場合は，ライセンスを再びアクティブ化する必要があります。

ライセンスマネージャの起動

ライセンスマネージャは，管理コンソールから次のように起動できます。

1. 管理コンソールを起動します。
2. ツールメニューから［License Manager］を選択します。

ライセンスマネージャは，次のように実行可能ファイルを使って起動することもできます。

Windows
- Windows でライセンスマネージャを起動するには，<install_dir>¥bin ディレクトリの lmadmw 実行可能ファイルを実行します。

UNIX
- UNIX でライセンスマネージャを起動するには，<install_dir>/bin ディレクトリの lmadm 実行可能ファイルを実行します。

ライセンス情報の表示

ライセンスマネージャには，ローカルマシンの製品に適用したライセンスが表示されます。各ライセンスの詳細について確認するには，左側のナビゲーションツリーでライセンスノードをクリックします。右側の内容ペインには，製品の詳細と選択したノードに実行できる操作が表示されます。

ライセンスの追加

場合によっては，1 つのインストールに複数のライセンスを適用する必要があります。このようなケースは，オプション製品を購入してベース製品に追加する場合に起こります。同じ製品インストールに 2 番め，3 番めなどのライセンスを適用する場合も，最初の場合と同じです。

新しいノード製品ライセンスを追加する前に，次の手続きが必要です。

- **ライセンスごとの Serial Number と Key，またはアクティベーションファイル。**これらは Borland から電子メールで送信されます。
- **Borland Developer Network（BDN）のアカウント。**アカウントを持たない場合は，登録／使用許諾プロセス時に作成できます。
- **インターネットアクセス。** ライセンスをアクティブ化する方法としてダイレクト登録または Web ページ登録を使用する場合は，インターネットアクセスが必要です。登録は任意のシステムから実行できます。製品をインストールしたシステムである必要はありません。

製品ライセンスを追加するには，次の手順にしたがいます。

1 ［Serial］メニューから［Add］を選択します。［Add Serial Number］ダイアログが開きます。

2 ［Serial number］と ［Key］を入力して［OK］をクリックします。ライセンスマネージャのナビゲーションツリーで，新しい Serial Number が赤い文字で Unregistered serial numbers のリストに表示されます。

3 新しい Serial Number を選択したら，［Serial］メニューの［Register］を選択します。Borland Product Registration Wizard の指示にしたがって登録プロセスを進めます。ウィザードの使い方の詳細については，『Borland *AppServer インストールガイド*』の「Borland Product Registration Wizard」を参照してください。

ライセンスのインポート

場合によっては，ファイルからライセンスをインポートする必要があります。製品ライセンスをインポートするには，次の手順にしたがいます。

1 ［License］メニューから［Import］を選択します。［Import License］ダイアログが開きます。

2 Import License ナビゲーションツールを使ってライセンスファイルを探し，［OK］をクリックします。

第 13 章

Optimizeit Profiler と ServerTrace の使い方

Borland Optimizeit Profiler と ServerTrace は，アプリケーションサーバー（AppServer など）とともに使用するために管理コンソールに統合されています。

管理コンソールで Optimizeit Profiler または ServerTrace を使用するには，次の手順にしたがう必要があります。

1 「Profiler ／ ServerTrace のローカルクライアントのインストールと設定」

2 「サーバー側の Profiler ／ ServerTrace の設定」

Optimizeit Profiler と ServerTrace の機能の使い方については，127 ページの「管理コンソールでの Profiler ／ ServerTrace 機能の使い方」を参照してください。

メモ この統合では，Optimizeit ServerTrace バージョン 3 以降と Optimizeit Profiler 6.0 以降をサポートします。

Profiler ／ ServerTrace のローカルクライアントのインストールと設定

Profiler ／ ServerTrace のローカルクライアントを設定するには，次の手順にしたがいます。

1 管理コンソールが実行されているマシンに Optimizeit Profiler ／ ServerTrace をインストールします。

メモ リモートエージェント管理にコンソールを使用している場合は，エージェントが起動しているマシンにも Profiler ／ ServerTrace をインストールしてください。

2 Borland 管理コンソールの［Console］メニューから，［Preferences］を選択し，［Tools］タブを選択します。

3 ローカルマシン（管理コンソールが実行されているマシン）にインストールされたクライアントの実行可能ファイルの（完全修飾された）絶対パスを入力します。
Optimizeit Profiler の場合，Optimizeit.exe と EditFilter.exe へのパスが必要です。
Optimizeit ServerTrace の場合，ServerTrace.exe へのパスが必要です。パスを明示的に入力するか，または［Browse］ボタンをクリックして実行可能ファイルを指定します。

4 終了したら，［OK］をクリックします。

メモ　複数の BAS パーティションに対して Profiler を設定する際に，「ポート 1470 はすでに使用されています。監査システムは起動できませんでした。(The port 1470 is already used, the audit system cannot start.)」というエラーメッセージを受け取ることがあります。このエラーが発生した場合は，Audit System Selector ユーティリティを使用して，次のパーティションが新しいポートを選択するように明示的に別のポート番号を割り当ててください。ポートを変更するには，パーティションの optimizeit.xml ファイルにある次の audit-port エントリを編集します。

```
<audit-port>
   <port-range begin="1473" end="1483" />
</audit-port>
```

サーバー側の Profiler ／ ServerTrace の設定

サーバー側の Profiler ／ ServerTrace の設定を行うには，次の手順にしたがいます。

1 管理コンソールの［Hubs］ビューがアクティブになっていない場合は，これを選択します。

2 Profiler ／ ServerTrace が存在する AppServer パーティション（または，その他のアプリケーションサーバー）を表すノードのナビゲーションツリーを展開します。

3 パーティションノードを右クリックし，Optimizeit を選択してから［Configure］を選択します。

4 ［Mode］ドロップダウンリストで，適切な起動モードを選択します。

- **［Normal - Non Optimizeit］:** Profiler と ServerTrace がパーティションとともに起動しないデフォルトの起動モード。
- **［Profiler - Start with Optimizeit Profiler］:** パーティションを Optimizeit Profiler とともに実行する場合は，この起動モードを選択します。
- **［ServerTrace - Start with Optimizeit ServerTrace］:** パーティションを Optimizeit ServerTrace とともに実行する場合は，この起動モードを選択します。

5 選択したモードに応じて，［Profiler］タブまたは［ServerTrace］タブを選択します。

6 ServerTrace または Profiler のホームフィールドに，リモートマシン（パーティションが実行されているマシン）の Optimizeit Profiler または ServerTrace のインストール先ディレクトリを絶対（完全修飾）パスで入力します。次に例を示します。

```
C:¥Borland¥Optimizeit¥ServerTrace3¥
```

7 ［OK］をクリックすると，Optimizeit Profiler ／ ServerTrace アーカイブが配布されます。

8 パーティションノードを右クリックし，Optimizeit を選択してから［Configure］を選択します。

9 ServerTrace EJB ファイルを生成するには，ファイルメニューで EJB ファイルを選択し，［Generate］をクリックします。このファイルを生成すると，編集できるようになります。

10 Profiler/ServerTrace Master Configuration，Filter，Action，または EJB ファイルを編集するには，［Files］一覧からファイルを選択して［Edit］をクリックします。このファイルの設定方法については，Optimizeit Profiler または ServerTrace のマニュアルを参照してください。

メモ　ServerTrace を起動すると，SNMP エージェントも起動されます。この SNMP エージェントには 2 つのポート設定があります。このポート設定は，ServerTrace Master 設定ファイルで指定します。ServerTrace が使用するポートと，同じホスト上の別のシステムが使用するポートが競合する可能性があるため，未使用のポートを設定するようにしてください。デフォルトの SNMP ポートは，エージェントが 161，トラップが 162

です。optimizeit.xml ファイルにある SNMP の agentPort エントリと trapPort エントリを編集して，未使用のポート番号を設定します。次に例を示します。

```
<snmp agentPort="161" trapPort="162" confPath="adm/servertrace/straceSNMP" />
```

11 終了したら，［OK］をクリックします。

メモ パーティションの実行中に［OK］をクリックすると，パーティションを再起動するように求めるメッセージが管理コンソールに表示されます。［Yes］を選択すると，パーティションは再起動しますが，実行速度が遅くなります。

管理コンソールでの Profiler ／ ServerTrace 機能の使い方

ここでは，管理コンソールに統合された Optimizeit Profiler 機能と ServerTrace 機能にアクセスして使用する方法を説明します。用語と機能については，Optimizeit Profiler と ServerTrace のマニュアルを参照してください。次の機能がサポートされます。

- 「Hibernate と Wakeup」
- 「Generate Snapshot」
- 「Attach」
- 「Clear Statistics」
- 「View Snapshot」

Profiler ／ ServerTrace 機能へのアクセス

統合された Profiler ／ ServerTrace 機能にアクセスするには，次の手順にしたがいます。

1 パーティションノード（または，サポートされたアプリケーションサーバーのノード）を右クリックして，Optimizeit を選択します。

2 サブメニューから必要な機能を選択します。

Profiler へのパーティションプロセスのアタッチ

Optimizeit Profiler にパーティションプロセスをアタッチするには

1 左側ペインでパーティション名を右クリックし，［Optimizeit ｜ Configure］を選択します。

2 ［Mode］ドロップダウンメニューから［Profiler - Start with Optimizeit Profiler］を選択します。

3 ［Profiler home］テキストボックスに Optimizeit Profiler 実行可能ファイルのパスを入力します。

4 パーティションの再起動を求められたら，［Yes］をクリックします。

5 管理コンソールで，［Console ｜ Preferences］に移動します。

6 ［Tools］タブをクリックして前面に表示します。

7 左側ペインでパーティションノードを右クリックし，［Optimizeit ｜ Attach］を選択します。

8 ［Edit settings］ダイアログで，［Remote Application］ラジオボタンをクリックします。

9 アタッチするパーティションをホストしているコンピュータの名前とポート番号を入力します。

10 ［Attach］ボタンをクリックします。

Hibernate と Wakeup

Hibernate 機能を使用すると，ServerTrace を hibernate モードにできます。このモードでは，情報収集は継続しますが，レポート機能は停止します。このオプションは，パーティションが実行中で，ServerTrace に対して有効になっている場合に有効になります。

ServerTrace を hibernate モードにするには，[Optimizeit] サブメニューの [Hibernate] を選択します。ServerTrace を hibernate モードから戻すには，サブメニューの［Wakeup］を選択します。

Generate Snapshot

スナップショットの生成では，サーバーのスナップショットを生成できます。このオプションは，パーティションが実行中で，ServerTrace に対して有効になっている場合に有効になります。

スナップショットを生成するには，次の手順にしたがいます。

1 パーティションの Optimizeit を右クリックして表示されるメニューから［Generate Snapshot］を選択します。

2 スナップショットを必要とした理由，またはこのスナップショットを識別できるその他の情報を入力します。
［Reason］フィールドのテキストは，スナップショットのファイル名に使用されます。

3 オプションで，スナップショットのコメントを入力します。このコメントは，スナップショットのメタ情報に含められます。

Attach

Attach オプションを使用して，Profiler クライアントまたは ServerTrace クライアントを開くと，サーバーの現在のビューが表示されます。このオプションは，パーティションが実行中で，Optimizeit Profiler または ServerTrace に対して有効になっている場合に有効になります。

Clear Statistics

Clear Statistics を使用すれば，ServerTrace 監査システムのデータをクリアできます。新しいスナップショットに以前の情報は含まれません。以前に生成されたスナップショットは削除されません。このオプションは，パーティションが実行中で，ServerTrace に対して有効になっている場合に有効になります。

View Snapshot

View Snapshot を使用して，生成されたスナップショットを ServerTrace クライアントから開きます。このオプションは，パーティションが実行中で，ServerTrace に対して有効になっている場合に有効になります。

スナップショットを表示するには，次の手順にしたがいます。

1 パーティションの Optimizeit を右クリックして表示されるメニューから［View Snapshot］を選択します。

2 リストからスナップショットを選択し，［OK］をクリックします。

3 スナップショットをローカルで保存するには，次の手順にしたがいます。

a ［Save to local file］をオンにします。

b スナップショットの保存先ディレクトリのパスを入力します（または［Browse］を使用します)。

c ［OK］をクリックします。

第 14 章

JMX コンソールの使い方

JMX コンソール（MC4J 管理コンソール）は，Borland AppServer（AppServer）パーティションなどの JMX 対応オブジェクトに対するサードパーティの管理オプションとして提供されます。JMX コンソールを開始するには，パーティション（または，別の J2EE サーバー）を右クリックし，［Launch JMX Console］を選択します。

メモ　デフォルトの JDK ではなく，外部の JDK を使用している場合（HP-UX の場合など）は，管理コンソールから MC4J を起動する際に問題が発生する可能性があります。この問題を回避するには，<install_dir>/bin ディレクトリにある mc4j.config ファイルの次の行を変更して，正しい JDK を指示する必要があります。

```
javahome $var(installRoot)/jdk/jdk1.4.2
```

MC4J 管理コンソールは，AppServer パーティションに関連付けられた MBeans のツリー構造ビューを提供します。MC4J ユーザーガイドについては，http://mc4j.sourceforge.net/guide/index.html を参照してください。

JMX コンソールのスタンドアロンでの起動

MC4J 管理コンソールは，スタンドアロン（Borland 管理コンソールを実行しない）でも実行できます。AppServer でこのコンソールをスタンドアロンで実行するには，次の 2 つの方法があります。

- 「JMX サービス URL の使い方」
- 「BES スマートエージェントの使い方」

JMX サービス URL の使い方

1 メモ帳で jmxservice.url ファイルの内容をクリップボードにコピーします。jmxservice.url ファイルは次の場所にあります。

```
<install_dir>/var/domains/<domain_name>/configurations/<config_name>/mos/<mo_name>/
```

AppServer パーティションの jmxservice.url を探すには，設定にパーティションを追加した後に実行する必要があります。また，<domain_name> は **base** であり，<mo_name> は ***Partition**（**PetstorePartition** など）です。次に例を示します。

```
<install_dir>/var/domains/base/configurations/j2eeSample/mos/PetstorePartition/
```

2 コマンドプロンプトで次のように MC4J を起動します。

```
prompt% mc4j.exe
```

この実行可能ファイルは <install_dir>/bin/ ディレクトリにあります。

3 Connection Explorer で，［MC4J Connections］ノードノードを右クリックし，［Connect to Server］を選択します。ウィザードが表示されます。

4 ドロップダウンリストから［BorlandServer］を選択します。

5 接続の名前を入力します。

6 ［Server URL］フィールドに，ステップ 1 でコピーした jmxservice.url ファイルを貼り付けます。

7 管理ドメインがセキュリティで保護されている場合は，［Principle］フィールドと［Credentials］フィールドにそれぞれログインとパスワードを指定します。デフォルトで AppServer 管理ドメインはセキュリティで保護されており，ログイン名とパスワードは admin と admin になります。

8 ［Next］をクリックしてクラスパスをカスタマイズするステップに移動し，<install_dir>/lib/ から次の JAR を追加します。

- asrt.jar
- common.jar
- dom4j.jar
- jaas.jar
- jafa.jar
- jcert.jar
- jnet.jar
- jsse.jar
- lm.jar
- log4j.jar
- mail.jar
- sanct4.jar
- vbejb.jar
- vbjorb.jar
- vbsec.jar
- xercesImpl.jar
- xmlParserAPIs.jar
- xmlrt.jar

9 ［Finish］をクリックします。接続は，Connection Explorer に新しいノードとして表示されます。

Borland AppServer スマートエージェントの使い方

1 <install_dir>/bin/mc4j.config ファイルを編集し，プロパティ vmprop bes.no_osagent=false を設定します。

2 コマンドラインを使って osagent ポートを各自の管理ポートに設定し，osfind を使って各自の JMX エージェントの名前（ojms_JONDOE_openjms_JmxAgent など）を検索します。

3 （OSAGENT_PORT を正しく設定したシェルと同じシェルから）<install_dir>/bin/mc4j.exe を起動します。

4 Connection Explorer で，［MC4J Connections］ノードを右クリックし，［Connect to Server］を選択します。ウィザードが表示されます。

5 ドロップダウンリストから［BorlandServer］を選択します。

6 ステップ 2 の接続の名前を入力します。

7 ［Server URL］テキストフィールドは使用されないので変更する必要はありません。

8 管理ドメインがセキュリティで保護されている場合は，［Principle］フィールドと［Credentials］フィールドにそれぞれログインとパスワードを指定します。デフォルトで AppServer 管理ドメインはセキュリティで保護されており，ログイン名とパスワードは admin と admin になります。

9 ［Next］をクリックしてクラスパスをカスタマイズするステップに移動し，<install_dir>/lib/ から次の JAR を追加します。

- asrt.jar
- common.jar
- dom4j.jar
- jaas.jar
- jafa.jar
- jcert.jar
- jnet.jar
- jsse.jar
- lm.jar
- log4j.jar
- mail.jar
- sanct4.jar
- vbejb.jar
- vbjorb.jar
- vbsec.jar
- xercesImpl.jar
- xmlParserAPIs.jar
- xmlrt.jar

10 ［Finish］をクリックします。接続は，Connection Explorer に新しいノードとして表示されます。

HTTP アダプタを使って JMX 対応オブジェクトを監視する

パーティションの JMX エージェントの設定で HTTP アダプタと XSLT プロセッサが有効な場合は，Web ブラウザを使って AppServer を監視することもできます。HTTP アダプタのデフォルトの URL は，http://localhost:8082 です。

AppServer パーティションに提供されている MBean の装備の詳細については,「パーティションでの JMX のサポート」を参照してください。

索引

記号

数値

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W

X

あ

い

う

え

お

か

き

く

け

こ

さ

し

す

せ

そ

ち

つ

て

と

な

に

ね

は

ひ

ふ

へ

ほ

ま

め

も

ゆ

ら

り

ろ