著作権

限定的権利条項

商標または登録商標

BEA WebLogic Server 6.1 リリース ノート

パート番号	マニュアルの日付	ソフトウェアのバージョン
860-001003-010	2003 年 12 月 2 日	BEA WebLogic Server バージョン 6.1

BEA WebLogic Server™

BEA WebLogic Express™

BEA WebLogic Server 6.1 リリース ノート

BEA WebLogic Server バージョン 6.1
マニュアルの日付：2003 年 12 月 2 日

目次

このマニュアルの内容

対象読者 xiv
e-docs Web サイト xiv
このマニュアルの印刷方法 xiv
サポート情報 xv
表記規則 xvi

1. WebLogic Server 6.1 の機能と変更点

WebLogic Server 6.1 サービス パック 6 1-1
EJB に関する変更 1-2
WebLogic Server 6.1 サービス パック 5 1-2
WebLogic Server 6.1 サービス パック 4 1-2
システム管理に関する変更 1-2
beasvc.exe に対する -delay 引数の動作の変更 1-3
アプリケーションを強制的に更新するための新しいパラメータ 1-3
JVM に対するデフォルト エンコーディングの設定 1-4
JDBC に関する変更 1-5
WebLogic Server 6.1 サービス パック 3 1-6
システム管理に関する変更 1-6
ACL の厳密な適用 1-6
WebLogic jDriver のサポートに関する変更 1-7
Oracle 9.0.1 Thin Driver の動作確認 1-7
JMS に関する変更 1-7
WebLogic と Tuxedo の相互運用性 1-9
WebLogic Server 6.1 サービス パック 2 1-9
互換性に関する問題 1-9
WebLogic Server 6.1 SP02 クライアント用の起動スクリプト スイッチ 1-10
SAX パーサとエンコーディング エラーの検出 1-10
動作確認 1-10
プロキシ サーブレットに対する変更 1-11

新しいセキュリティ クラス 1-11
サーブレットと JSP のエンコーディングに関する変更 1-12
WebLogic 6.1 SP02 と 5.1 の相互運用性 1-12
WebLogic Server 6.1 サービス パック 1 1-13
WebLogic Server 6.1 (G.A. リリース) 1-13
WebLogic API に関する注意事項 1-14
アプレット 1-14
Beanshell コード 1-15
クラスタ 1-15
WebLogic Server 6.1 でのデプロイメント 1-15
エンタープライズ JavaBean (EJB) 1-16
Hypertext Transfer Protocol (HTTP) 1.1 1-17
J2EE コネクタ 1-18
Java Database Connectivity (JDBC) 1-19
JavaMail 1-19
Java Transaction API (JTA) 1-20
JMS 1-20
JSP 1-21
セキュリティ 1-22
サーブレットと Web アプリケーション 1-23
SNMP 1-24
システム管理 1-24
WebLogic Tuxedo Connector 1-27
XML 1-28
Web サービス 1-29
WebLogic Server™ の現在のリリースには、WebLogic Web サービスという新機能が追加されています。Web サービスは、Web および XML の標準規格の新しいファミリであり、プログラミング モデル全体にわたる、柔軟に結合されたアプリケーション同士の相互運用を実現します。WebLogic Server 6.1 の Web サービス実装では、J2EE プログラミング モデルに対する簡単な拡張機能が提供されており、J2EE アプリケーションを Web サービスとして公開できます。WebLogic Web サービスは標準に準拠しており、WebLogic Server 以外のサーバでホストされる Web サービスと相互運用できます。Java クライアント アプリケーションからでも、非 Java クライアント アプリケーション (Microsoft SOAP Toolkit クライアントなど) からでも、WebLogic Web サービスを呼び出すことができます。

WebLogic Web サービスを開発および呼び出す方法の詳細については、『WebLogic Server Web サービス プログラマーズ ガイド』を参照してください。非推奨となった機能と API 1-29
マニュアルとサンプル 1-30
WebLogic Server ツアー 1-31
J2EE と規格について 1-31
Java Development Kit 1-31
J2EE 1.2 の機能に加えて J2EE 1.3 の機能を備える WebLogic Server 6.1 1-32
J2EE 1.2 認定の WebLogic Server 6.1 1-35
J2EE 1.2 および 1.3 の製品 CD インストーラ 1-35
規格のサポート 1-35

2. WebLogic Server 5.1 からバージョン 6.1 へのアップグレード

WebLogic Server コンフィグレーションのアップグレード : 主な手順 2-2
weblogic.properties ファイルの変換 2-2
WebLogic Server 6.x でのクラスのロード 2-3
起動スクリプトの修正 2-4
Oracle のアップグレード 2-4
JMS のアップグレード 2-5
JSP 2-14
JVM のアップグレード 2-15

3. WebLogic Server 4.5 および 5.1 アプリケーションのバージョン 6.x への移行

weblogic.properties ファイルと .xml ファイル 3-2
WebLogic Server 6.x アプリケーションの種類 3-3
移行の主な手順 3-3
weblogic.properties ファイルの .xml ファイルへの変換 3-4
weblogic.properties のマッピング表 3-5
Web アプリケーションの移行 3-26
Web アプリケーションのディレクトリ構造 3-27
XML デプロイメント記述子 3-28
WAR ファイル 3-28
Web アプリケーションのデプロイメント 3-29

セッションの移行 3-30
JavaServer Pages（JSP）とサーブレット 3-30
WebLogic Server 5.1 から WebLogic Server 6.1 への単純なサーブレットの移行 3-32
エンタープライズ JavaBean アプリケーションの移行と変換 3-33
EJB の移行に関する考慮事項 3-34
EJB の移行に関する推奨事項 3-36
1.0 EJBean を WebLogic Server 4.5.x から WebLogic Server 6.1 に移行する手順 3-38
1.1 EJBean を WebLogic Server 5.1 から WebLogic Server 6.1 に移行する手順 3-39
EJB 1.1 を EJB 2.0 に変換する手順 3-40
他の J2EE アプリケーション サーバからの EJB の移行 3-40
エンタープライズ アプリケーションの作成 3-41
J2EE クライアント アプリケーションについて 3-42
JMS アプリケーションの移行 3-43
移行およびデプロイメントの補足事項 3-44
スタンドアロンの HTML および JSP 3-44
アプリケーション サーバと管理対象サーバ 3-45
Java Transaction API（JTA） 3-45
Java Database Connectivity（JDBC） 3-45
RMI 3-46
FileServlet 3-47
インターナショナライゼーション（I18N） 3-47
セキュリティ 3-48
WAP アプリケーション 3-48
セッションの移行 3-49
Web コンポーネント 3-50
XML 6.0->6.1 パーサ 3-51
非推奨となった API および機能 3-51
削除された API および機能 3-51

4. WebLogic Server 6.0 アプリケーションの WebLogic Server 6.1 への移行

バージョン 6.0 からバージョン 6.1 へのアプリケーションの移行 4-2
JMS 4-3

EJB 2.0 4-4
EJB 2.0 セッション Bean と BMP エンティティ Bean のアップグレード 4-5
EJB 2.0 メッセージ駆動型 Bean（MDB）のアップグレード 4-5
EJB 2.0 CMP エンティティ Bean のアップグレード 4-5
JMX 4-6
Apache XML パーサ 4-6
Xalan XML パーサ 4-7
Web アプリケーション 4-7
セキュリティ 4-8
config\applications ディレクトリ 4-8
Solaris での WebLogic Server クラスタ 4-8
スレッド プールのサイズ 4-9

5. 注意事項

コネクタに関する確認済みの問題 5-3
Administration Console に関する確認済みの問題 5-3
コアに関する確認済みの問題 5-4
EJB に関する確認済みの問題 5-5
サンプルと Pet Store Demo に関する確認済みの問題 5-9
インストーラに関する確認済みの問題 5-14
JCA に関する確認済みの問題 5-16
JDBC と jDrivers に関する確認済みの問題 5-16
JMS に関する確認済みの問題 5-18
JSP に関する確認済みの問題 5-18
JVM に関する確認済みの問題 5-20
RMI に関する確認済みの問題 5-22
RMI-IIOP に関する確認済みの問題 5-22
セキュリティに関する確認済みの問題 5-23
サーバに関する確認済みの問題 5-25
サーブレットと Web アプリケーションに関する確認済みの問題 5-28
システム管理に関する確認済みの問題 5-31
ツールに関する確認済みの問題 5-35
Web サービスに関する確認済みの問題 5-35
WebLogic Tuxedo Connector に関する確認済みの問題 5-38

XML に関する確認済みの問題 5-40
ZAC に関する確認済みの問題 5-41

6. 解決済みの問題

WebLogic Server 6.1 サービス パック 6 のソリューション 6-1
クラスローダ 6-3
クラスタ 6-4
コネクタ 6-8
コンソール 6-9
コア 6-11
デプロイメント 6-26
EJB 6-27
JDBC 6-40
jDriver 6-47
JMS 6-48
JNDI 6-52
JSP 6-54
JTA 6-60
ノード マネージャ 6-64
OA&M (操作と管理) 6-65
プラグイン 6-74
RMI 6-83
RMI/IIOP 6-84
セキュリティ 6-86
サーブレット 6-91
SNMP 6-102
Web サービス 6-103
WTC-ATMI 6-105
XML 6-107
WebLogic Server 6.1 サービス パック 5 のソリューション 6-108
クラスタ 6-110
コネクタ 6-114
コンソール 6-115
コア 6-119
デプロイメント 6-134

EJB 6-135
インストーラ 6-148
JDBC 6-149
jDriver 6-156
JMS 6-159
JNDI 6-166
JSP 6-169
JTA 6-172
その他 6-176
プラグイン 6-177
RMI-IIOP 6-187
RMI 6-189
セキュリティ 6-190
サーブレット 6-191
WLS ツアー / サンプル 6-209
Web サービス 6-211
WebLogic Tuxedo 6-215
XML 6-217
WebLogic Server 6.1 サービス パック 4 のソリューション 6-218
クラスローダ 6-219
クラスタ 6-220
コンソール 6-222
EJB 6-224
サンプル 6-226
JDBC および jDriver 6-227
JMS 6-228
その他 6-230
プラグイン 6-239
サーブレット、JSP、Web アプリケーション 6-242
セキュリティ 6-249
システム管理 6-251
Web サービス 6-254
WebLogic Tuxedo 6-255
XML 6-256
WebLogic Server 6.1 サービス パック 3 のソリューション 6-257

クラスローダ 6-258
クラスタ 6-259
コンソール 6-260
コア 6-262
EJB 6-263
サンプル 6-268
JDBC 6-269
jDriver 6-271
JMS 6-272
JTA 6-273
その他 6-274
プラグイン 6-281
RMI over IIOP 6-285
セキュリティ 6-286
サーブレット、JSP、および Web アプリケーション 6-288
システム管理 6-296
Web サービス 6-299
WebLogic Tuxedo Connector 6-301
XML 6-302
WebLogic Server 6.1 サービス パック 3 に付属するサードパーティの JAR ファイル 6-303
WebLogic Server 6.1 サービス パック 2 のソリューション 6-305
EJB 6-306
サンプル 6-309
JDBC 6-310
jDriver 6-311
JMS 6-312
その他 6-313
プラグイン 6-318
RMI over IIOP 6-319
セキュリティ 6-320
サーブレット、JSP、および Web アプリケーション 6-321
システム管理 6-326
Web サービス 6-328
WebLogic Tuxedo Connector 6-330

XML 6-331
WebLogic Server 6.1 サービス パック 1 のソリューション 6-332
デプロイメント記述子エディタ 6-333
EJB 6-335
サンプル 6-339
JDBC 6-341
JMS 6-343
JTA 6-344
その他 6-345
プラグイン 6-349
RMI over IIOP 6-350
サーブレットと JSP 6-351
システム管理 6-354
Web サービス 6-357
XML 6-358
WebLogic Server 6.1 リリースのソリューション 6-360
デプロイメント記述子エディタ 6-361
EJB 6-362
サンプル 6-363
インターナショナライゼーション 6-364
JCA 6-365
JDBC 6-366
JMS 6-368
その他 6-372
プラグイン 6-376
サーブレットと JSP 6-378
Web サービス 6-381
XML 6-382

このマニュアルの内容

このマニュアルでは、BEA WebLogic Server の最新リリースを紹介します。新機能、サポートされる仕様、および確認済みの問題についての重要な情報を提供します。

このマニュアルの構成は次のとおりです。

- 第 1 章「WebLogic Server 6.1 の機能と変更点」では、WebLogic Server 6.1 の概要について説明します。WebLogic Server の最新リリースと付属のサービス パックに関する重要な情報を提供します。
- 第 2 章「WebLogic Server 5.1 からバージョン 6.1 へのアップグレード」では、WebLogic Server の以前のバージョンから WebLogic Server 6.1 へ移行するユーザ向けの情報を提供します。
- 第 3 章「WebLogic Server 4.5 および 5.1 アプリケーションのバージョン 6.x への移行」では、WebLogic Server の以前のバージョンから WebLogic Server 6.0 または 6.1 への移行について概説します。
- 第 4 章「WebLogic Server 6.0 アプリケーションの WebLogic Server 6.1 への移行」では、WebLogic Server 6.0 から 6.1 への移行について概説します。
- 第 5 章「注意事項」では、WebLogic Server 6.1 における確認済みの問題のリストを示します。
- 第 6 章「解決済みの問題」では、WebLogic Server 6.1 で修正されているバグについての情報を提供します。

対象読者

このマニュアルは、WebLogic Server 6.1 を使用するユーザを対象としています。

e-docs Web サイト

BEA 製品のドキュメントは、BEA の Web サイトで入手できます。BEA のホーム ページで [製品のドキュメント] をクリックします。

このマニュアルの印刷方法

Web ブラウザの [ファイル | 印刷] オプションを使用すると、Web ブラウザからこのマニュアルのメイン トピックを一度に 1 つずつ印刷できます。

このマニュアルの PDF 版は、Web サイトで入手できます。PDF を Adobe Acrobat Reader で開くと、マニュアルの全体（または一部分）を書籍の形式で印刷できます。PDF を表示するには、WebLogic Server ドキュメントのホーム ページを開き、[ドキュメントのダウンロード] をクリックして、印刷するマニュアルを選択します。

Adobe Acrobat Reader は、Adobe の Web サイト（http://www.adobe.co.jp）から無料で入手できます。

サポート情報

BEA のドキュメントに関するユーザからのフィードバックは弊社にとって非常に重要です。質問や意見などがあれば、電子メールで docsupport-jp@bea.com までお送りください。寄せられた意見については、ドキュメントを作成および改訂する BEA の専門の担当者が直に目を通します。

電子メールのメッセージには、ご使用のソフトウェア名とバージョン名、およびマニュアルのタイトルと作成日付をお書き添えください。本バージョンの BEA WebLogic Server について不明な点がある場合、または BEA WebLogic Server のインストールおよび動作に問題がある場合は、BEA WebSUPPORT（www.bea.com）を通じて BEA カスタマ サポートまでお問い合わせください。カスタマ サポートへの連絡方法については、製品パッケージに同梱されているカスタマ サポート カードにも記載されています。

カスタマ サポートでは以下の情報をお尋ねしますので、お問い合わせの際はあらかじめご用意ください。

- お名前、電子メール アドレス、電話番号、ファクス番号
- 会社の名前と住所
- お使いの機種とコード番号
- 製品の名前とバージョン
- 問題の状況と表示されるエラー メッセージの内容

表記規則

このマニュアルでは、全体を通して以下の表記規則が使用されています。

表記法	適用
〔Ctrl〕+ 〔Tab〕	同時に押すキーを示す。
斜体	強調または本のタイトルを示す。

等幅テキスト	コード サンプル、コマンドとそのオプション、Java クラス、データ型、ディレクトリ、およびファイル名とその拡張子を示す。等幅テキストはキーボードから入力するテキストも示す。
	例:
	`import java.util.Enumeration;`
	`chmod u+w *`
	`config/examples/applications`
	`.java`
	`config.xml`
	`float`

斜体の等幅テキスト	コード内の変数を示す。 例: `String CustomerName;`
すべて大文字のテキスト	デバイス名、環境変数、および論理演算子を示す。 例: LPT1 BEA_HOME OR
{ }	構文内の複数の選択肢を示す。

表記法	適用
[ ]	構文内の任意指定の項目を示す。 例： `java utils.MulticastTest -n name -a address [-p portnumber] [-t timeout] [-s send]`
|	構文の中で相互に排他的な選択肢を区切る。 例： `java weblogic.deploy [list|deploy|undeploy|update] password {application} {source}`
...	コマンドラインで以下のいずれかを示す。 ■ 引数を複数回繰り返すことができる。 ■ 任意指定の引数が省略されている。 ■ パラメータや値などの情報を追加入力できる。
. . .	コード サンプルまたは構文で項目が省略されていることを示す。

1 WebLogic Server 6.1 の機能と変更点

ここでは、WebLogic Server 6.1 における変更点と機能について説明します。

現在のリリースは WebLogic Server 6.1 サービス パック 6 です。各サービス パックには、それより前のサービス パックで行われた変更と修正が含まれています。

以下の節では、WebLogic Server 6.1 の各サービス パックにおける主な変更点を示します。

- 1-1 ページの「WebLogic Server 6.1 サービス パック 6」
- 1-2 ページの「WebLogic Server 6.1 サービス パック 5」
- 1-2 ページの「WebLogic Server 6.1 サービス パック 4」
- 1-6 ページの「WebLogic Server 6.1 サービス パック 3」
- 1-9 ページの「WebLogic Server 6.1 サービス パック 2」
- 1-13 ページの「WebLogic Server 6.1 サービス パック 1」
- 1-13 ページの「WebLogic Server 6.1 (G.A. リリース)」

WebLogic Server 6.1 において現在確認されている問題に関する情報については、第 5 章「注意事項」を参照してください。

WebLogic Server 6.1 サービス パック 6

以下の節では、WebLogic Server 6.1 SP06 において行われたバグの解決と重要な変更について説明します。

- 6-1 ページの「WebLogic Server 6.1 サービス パック 6 のソリューション」
- 1-2 ページの「EJB に関する変更」

EJB に関する変更

EJB 1.1 の CMP Beans にフィールドの最適化機能が実装され、フィールドが更新されてもその値に変更がなかった場合には、データベースへの書き込みが行われなくなりました。最適化は、プリミティブ型のフィールドおよび不変のフィールドにのみ適用されます。

WebLogic Server 6.1 サービス パック 5

以下の節では、WebLogic Server 6.1 SP05 において行われたバグの解決と重要な変更について説明します。

- 6-108 ページの「WebLogic Server 6.1 サービス パック 5 のソリューション」

WebLogic Server 6.1 サービス パック 4

以下の節では、WebLogic Server 6.1 SP04 において行われたバグの解決と重要な変更について説明します。

- 6-218 ページの「WebLogic Server 6.1 サービス パック 4 のソリューション」
- 1-2 ページの「システム管理に関する変更」
- 1-5 ページの「JDBC に関する変更」

システム管理に関する変更

ここでは、WebLogic Server 6.1 SP04 において行われたシステム管理に関する重要な変更について説明します。

beasvc.exe に対する -delay 引数の動作の変更

WebLogic Server の Windows サービス プログラム beasvc.exe に対する -delay 引数の動作と使い方が、WebLogic Server 6.1 SP04 では変更されています。

以前は、管理対象サーバに対して beasvc.exe を実行するときに、-delay 引数が適用されました。Windows の Service Control Manager (SCM) は、管理対象サーバをサービスとして起動するときに、-delay 引数で指定された時間 (ミリ秒) だけ待った後、サービス ステータスを STARTED に設定して、サービスを起動しました。

WebLogic Server 6.1 SP04 では、-delay 引数は、各管理対象サーバに対してではなく、管理サーバのレベルで適用されます。ドメインの管理サーバに対して beasvc.exe を実行するときに、-delay 引数を使用します。引数は、その管理サーバに依存するすべてのサービスに対して適用され、これにはドメイン内の管理対象サーバも含まれます。

サービスのステータスと起動に対して -delay が及ぼす影響についても変更があります。WebLogic Server 6.1 SP04 では、Windows Service Control Manager (SCM) は管理対象サーバを直ちに起動して、サービス ステータスを SERVER_START_PENDING に設定し、指定された遅延時間が経過した後、サービス ステータスを STARTED に設定します。

アプリケーションを強制的に更新するための新しいパラメータ

管理サーバに対する新しい起動オプション -Dweblogic.management.forceApplicationCopy を指定すると、管理対象サーバは、デプロイされているアプリケーションの最新バージョンを、起動時に必ず取得します。この起動オプションの効果については、「起動時の強制的なアプリケーション更新」で説明します。「デフォルトのアプリケーション更新処理」では、-Dweblogic.management.forceApplicationCopy が false のときのデフォルト動作として、管理対象サーバが更新情報を取得する方法を説明します。

起動時の強制的なアプリケーション更新

-Dweblogic.management.forceApplicationCopy パラメータは、管理サーバに対する起動オプションとして、コマンドラインまたは起動スクリプトで指定できます。-Dweblogic.management.forceApplicationCopy を true に設定して

管理サーバを起動すると、ドメインの管理対象サーバの起動時に、その管理対象サーバにデプロイされているアプリケーションが、管理サーバから管理対象サーバにコピーされます。

デフォルトのアプリケーション更新処理

`-Dweblogic.management.forceApplicationCopy` を true に設定しないで管理サーバを起動すると、次の条件が両方とも満たされている場合にだけ、管理対象サーバの起動時にアプリケーションがコピーされます。

- アプリケーションが変更されて再デプロイされている。
- 管理対象サーバに、最新バージョンのアプリケーションがない。

管理サーバは、ドメイン内のどの管理対象サーバに最新バージョンのアプリケーションがあるのかを示す `StagedTargets` リストを保持しています。管理対象サーバは、起動時に、自分 (管理対象サーバ) のアプリケーションが最新バージョンかどうかを管理サーバに問い合わせます。アプリケーションのバージョンが更新されている場合は、それが管理対象サーバにコピーされて、管理サーバは `StagedTargets` リストにその管理対象サーバを追加します。

管理サーバがダウンすると、`StagedTargets` からすべての管理対象サーバが削除されます。管理サーバを再起動すると、ドメイン内の各管理対象サーバは、デプロイされているアプリケーションを管理サーバからコピーします。これにより、管理サーバがダウンしている間にアプリケーションが更新された場合でも、すべての管理対象サーバのアプリケーションは同じバージョンになります。

JVM に対するデフォルト エンコーディングの設定

WebLogic Server 6.1 SP04 と Sun JVM 1.3.1_06 を使用した場合、`-Dfile.encoding` 属性を使って VM に対するデフォルト エンコーディングを設定しようとすると失敗します。

`file.encoding` は読み取り専用のプロパティなので、コマンドラインまたは `java.lang.System` クラスのメソッドのどちらを使っても、このプロパティを変更することはできません。

詳細については、<http://developer.java.sun.com/developer/bugParade/bugs/4163515.html> および関連するバグを参照してください。

VM および実行時システムが使用するデフォルト エンコーディングを変更する方法としては、Java プログラムを実行する前に基になっているプラットフォームのロケールを変更することをお勧めします。

JDBC に関する変更

WebLogic Server 6.1 SP04 のリリースでは、WebLogic Server での Oracle 9.2.0 Thin Driver の動作が確認されています。WebLogic Server 6.1 SP04 のソフトウェア配布キットには、以下の新しい JDBC ドライバが収められています。

- WebLogic jDriver for Oracle 9.2.0。Oracle 8.1.7 用および 9.0.1 用のバージョンと共に収められています。
- Oracle 9.2.0 Thin Driver。WebLogic Server 6.1 の以前のリリースに付属していた 8.1.7 バージョンの代わりに収められています。

WebLogic jDriver for Oracle の使い方の詳細については、『WebLogic jDriver for Oracle のインストールと使い方』を参照してください。WebLogic Server での Oracle Thin Driver の使い方の詳細については、「WebLogic Server でのサードパーティ ドライバの使い方」を参照してください。

Oracle Thin Driver バージョン 9.2.0 では、以前のドライバでは利用できた一部の拡張メソッドが削除されています。これらのメソッドはサポートされなくなっており、WebLogic Server 6.1 SP04 には含まれていません。

クラス:

```
weblogic.jdbc.oracle.OracleConnection
```

メソッド:

```
isCompatibleTo816()
```

クラス:

```
weblogic.jdbc.oracle.OracleStatement
```

メソッド:

```
getWaitOption()
setWaitOption(int i)
setAutoRollback(int i)
getAutoRollback()
```

クラス：

```
weblogic.jdbc.oracle.OracleResultSet
```

メソッド：

```
getCURSOR(String s)
```

WebLogic Server 6.1 サービス パック 3

以下の節では、WebLogic Server 6.1 SP03 において行われたバグの解決と重要な変更について説明します。

- 6-257 ページの「WebLogic Server 6.1 サービス パック 3 のソリューション」
- 1-6 ページの「システム管理に関する変更」
- 1-7 ページの「WebLogic jDriver のサポートに関する変更」
- 1-7 ページの「Oracle 9.0.1 Thin Driver の動作確認」
- 1-7 ページの「JMS に関する変更」

システム管理に関する変更

ここでは、WebLogic Server 6.1 SP03 において行われたシステム管理に関する重要な変更について説明します。

ACL の厳密な適用

WebLogic Server 6.1 SP03 では、JDBC に対する ACL の実装のエラーが解決されて、ACL が非常に厳密になりました。接続プールに対して ACL を定義した場合、アクセスは ACL の定義に「厳密に」従って (それ以上でも以下でもなく) 制限されます。詳細については、「パーミッション」を参照してください。

WebLogic jDriver のサポートに関する変更

以下のリストでは、WebLogic jDriver for Oracle で使用される Oracle のサポート対象バージョンに関する最近の変更について詳しく説明します。

- WebLogic Server 6.1 サービス パック 3 のリリース時点で、Oracle 8.1.6 用の WebLogic jDriver for Oracle はサポートされなくなりました。WebLogic jDriver for Oracle を使って Oracle 8.1.6 データベースに接続している場合は、Oracle のバージョン 8.1.7 または 9.0.1 にアップグレードする必要があります。そのためには、WebLogic Server が稼働しているマシンで、以下の手順を実行します。
  - Oracle クライアントのインストールを、バージョン 8.1.7 または 9.0.1 にアップグレードします。
  - アップグレードした Oracle クライアントのバージョンを示すように、`ORACLE_HOME` 環境変数を更新します。
  - 新しい Oracle クライアントへのパスおよび WebLogic jDriver for Oracle の適切なバージョンへのパスを含むように、PATH (Windows の場合、UNIX では異なります) を変更します。

  詳細については、『WebLogic jDriver for Oracle のインストールと使い方』の「WebLogic jDriver for Oracle の使用環境の設定」を参照してください。

Oracle 9.0.1 Thin Driver の動作確認

WebLogic Server 6.1 SP03 のリリースでは、Oracle 9.0.1 Thin Driver を WebLogic Server で使用した場合の動作が確認されています。WebLogic Server での Oracle Thin Driver の使い方の詳細については、「WebLogic Server でのサードパーティ ドライバの使い方」を参照してください。

JMS に関する変更

WebLogic Server 6.1 SP03 には、WebLogic メッセージング ブリッジが含まれています。

メッセージング ブリッジは、2 つのメッセージング プロバイダ間のメッセージの転送を行います。WebLogic メッセージング ブリッジを使うと、WebLogic JMS の個別の実装を含む任意の 2 つのメッセージング製品間に、ストア アンド フォワード (store-and-forward) のメカニズムをコンフィグレーションできます。

詳細については、『WebLogic Server 管理者ガイド』の「WebLogic メッセージング ブリッジの使い方」を参照してください。

WebLogic Server 6.1 SP03 には、新しい JMS メッセージ ページング機能が含まれています。

WebLogic の JMS メッセージ ページング機能を使うと、メッセージの負荷が指定されたしきい値に達した時点で、メッセージを仮想メモリから永続ストレージにスワップ アウトすることにより、メッセージ負荷がピークに達している間に貴重な仮想メモリを空けることができます。今日のエンタープライズ アプリケーションが必要とする大容量のメッセージ空間を備える WebLogic Server の実装にとって、この機能はパフォーマンスの点で大きなメリットがあります。

ページングの開始と終了のタイミングを決定するには、バイト ページングとメッセージ ページングという 2 つのメトリックを使用します。これらのメトリックはそれぞれが 1 つのページング モードの基になっており、JMS サーバと送り先 (トピックとキュー) の一方または両方で、個別に、または同時に、有効または無効にすることができます。

ページングは、Administration Console でコンフィグレーションします。JMS サーバ ノードのページング属性を使うことで、サーバに対するページング ストアを指定したり、バイト ページングやメッセージ ページングを有効にしたり、ページングを開始および停止するバイト / メッセージの最大 / 最小しきい値をコンフィグレーションしたりできます。同様に、送り先ノードのページング属性を使えば、JMS サーバにコンフィグレーションされているすべてのトピックやキューに対するバイト / メッセージ ページングをコンフィグレーションできます。送り先は、JMS サーバに対してコンフィグレーションされているページング ストアを使用します。また、JMS テンプレートを使用して複数の送り先をコンフィグレーションする場合は、テンプレート ノードのページング属性を使うことで、すべての送り先に対するメッセージ ページングを簡単にコンフィグレーションできます。

メッセージ ページングのコンフィグレーション方法の詳細については、WebLogic Server 管理者ガイド』の「JMS のチューニング」を参照してください。

WebLogic と Tuxedo の相互運用性

WebLogic Server 6.1 SP03 は、C++ クライアントとの相互運用に Tuxedo 8.0 C++ Client ORB を使用します。Tuxedo リリース 8.0 RP 56 以上が必要です。Tuxedo C++ Client ORB の入手方法については、BEA のサービス担当者にお問い合わせください。WebLogic Server と Tuxedo C++ Client ORB の相互運用方法の詳細については、『Tuxedo と Tuxedo クライアントを使用した RMI-IIOP』を参照してください。

WebLogic Server 6.1 サービス パック 2

以下の節では、WebLogic Server 6.1 SP02 において行われたバグの解決と重要な変更について説明します。

- 6-305 ページの「WebLogic Server 6.1 サービス パック 2 のソリューション」
- 1-9 ページの「互換性に関する問題」
- 1-10 ページの「動作確認」
- 1-11 ページの「プロキシ サーブレットに対する変更」
- 1-11 ページの「新しいセキュリティ クラス」
- 1-12 ページの「サーブレットと JSP のエンコーディングに関する変更」
- 1-12 ページの「WebLogic 6.1 SP02 と 5.1 の相互運用性」

互換性に関する問題

ここでは、WebLogic Server 6.1 SP02 の互換性に関する問題について説明します。

WebLogic Server 6.1 SP02 クライアント用の起動スクリプト スイッチ

デフォルトでは、WebLogic Server 6.1 SP02 のクライアントは、WebLogic Server 6.1 GA または 6.1 サービス パック 1 のサーバと相互運用できません。この問題は、6.1 SP02 クライアントだけでなく、6.1 GA サーバまたは 6.1 SP01 サーバに対するクライアントとして動作する 6.1 SP02 サーバでも発生します。

SP02 クライアントが 6.1 GA または 6.1 SP01 のサーバと対話するためには、WebLogic Server 6.1 SP02 の起動スクリプトに次のスイッチを追加する必要があります。

```
-Dweblogic.61compat=true
```

SAX パーサとエンコーディング エラーの検出

WebLogic Server の組み込み SAX パーサが、任意の文字セットで記述されたデプロイメント記述子ファイルを正しく解析するようになりました。

この障害対応を行う前は、英語だけを使用するアプリケーションの `.xml` ファイルに存在するエンコーディングの問題を検出できませんでした。したがって、エンコーディングの問題があってもこれまでは障害にならずに動作していた英語専用アプリケーションが、SP02 でのこの修正により障害が発生するようになる場合があります。この問題は、サポートされていないエンコーディング (encoding-not-supported) のエラーとして通知されます。

このエラーが発生した場合は、指定された `.xml` ファイルをチェックし、エンコーディングの名前と構文が正しいことを確認してください。

動作確認

- WebLogic Server 6.1 SP02 は、Sun の J2SE 1.3.1_01 での動作が確認されています。
- WebLogic Server 6.1 SP02 は、HP の JVM 1.3.1_01 での動作が確認されています。HP JVM 1.3.1_01 で必要なパッチについては、次の Web サイトを参照してください。

  http://www.hp.com/products1/unix/java/infolibrary/patches.html

- WebLogic Server 6.1 SP02 では、Oracle 9i での WebLogic jDriver for Oracle の使用がサポートされています。JDBC ドライバを使って Oracle 9i データベースに接続できるようになりました。WebLogic jDriver for Oracle の詳細については、『WebLogic jDriver for Oracle のインストールと使い方』を参照してください。

プラットフォーム サポートに関する最新の情報については、『動作確認状況』を参照してください。

プロキシ サーブレットに対する変更

WebLogic Server 6.1 SP02 には、HTTP 1.1 プロトコルをサポートする新バージョンの `HttpClusterServlet` と `HttpProxyServlet` が収められています。さらに、これらのサーブレットは、Apache、Netscape、および Microsoft IIS のプラグインと同じコンフィグレーション パラメータを使用します。

詳細については、以下のマニュアルを参照してください。

- 「別の HTTP サーバへのリクエストのプロキシ」
- 「WebLogic クラスタ へのリクエストのプロキシ」
- 「Web サーバ プラグインのパラメータ」

新しいセキュリティ クラス

WebLogic Server 6.1 SP02 には、セキュリティに関する 2 つの新しいクラスが含まれています。

- `weblogic.security.SSL.TrustManager` クラスを使用すると、ピアのデジタル証明書での検証エラーをオーバーライドし、SSL ハンドシェークを続けることができます。また、このクラスを使うと、サーバのデジタル証明書チェーンで付加的な検証を実行することで、SSL ハンドシェークを中止することもできます。詳細については、「WebLogic Security SPI を使用したプログラミング」の「トラスト マネージャの使用」を参照してください。
- `SSLContext` クラスを使用すると、特定の SSL 接続セットに対するホスト名検証やトラスト マネージャなどの情報を保持するセキュア ソケット プロト

コルを実装できます。詳細については、「WebLogic Security SPI を使用したプログラミング」の「SSL コンテキストの使用」を参照してください。

さらに、SP02 で追加された SSL セッション キャッシングに対するパラメータを使えば、接続において SSL ハンドシェークを再び行う必要がなくなります。詳細については、「セキュリティの管理」の「SSL セッション キャッシングのパラメータの変更」を参照してください。

サーブレットと JSP のエンコーディングに関する変更

WebLogic Server 6.1 SP02 から、サーブレットと JSP で使用するデフォルトのエンコーディングが ISO8859_1 に変更されました。以前は、JVM がデフォルトのエンコーディングとして使われていました。この変更は、最新のサーブレット仕様に従ったものです。サーブレット仕様 2.3 SRV 4.9 では、デフォルトのエンコーディングは ISO8859_1 でなければならないと規定されています。

エンコーディングに関する問題を防ぐため、リクエスト オブジェクトと応答オブジェクトのエンコーディングを明示的に設定することをお勧めします。エンコーディングを設定する方法の詳細については、『WebLogic Server 6.1J 日本語環境での使用にあたって』を参照してください。

WebLogic 6.1 SP02 と 5.1 の相互運用性

WebLogic Server 6.1 SP02 では、WebLogic 5.1 Server との双方向の相互運用が可能です。これにより、WebLogic 5.1 Server でホストされている EJB オブジェクトまたは RMI オブジェクトを、WebLogic 6.1 Server からリモートで呼び出すことができます。また、これとは逆に、WebLogic 6.1 Server でホストされている EJB オブジェクトまたは RMI オブジェクトを、WebLogic 5.1 Server からリモートで呼び出すこともできます。詳細については、『WebLogic Server 6.1 と 5.1 の相互運用性』を参照してください。

WebLogic Server 6.1 サービス パック 1

WebLogic Server 6.1 SP01 において解決された問題については、6-332 ページの「WebLogic Server 6.1 サービス パック 1 のソリューション」を参照してください。

WebLogic Server 6.1 (G.A. リリース)

以下の節では、WebLogic Server 6.1 の一般向け (G.A.) リリースにおける機能を説明し、動作確認と標準サポートについての情報を提供し、確認されている重要な問題を指摘します。

- 1-14 ページの「WebLogic API に関する注意事項」
- 1-29 ページの「WebLogic Server™ の現在のリリースには、WebLogic Web サービスという新機能が追加されています。Web サービスは、Web および XML の標準規格の新しいファミリであり、プログラミング モデル全体にわたる、柔軟に結合されたアプリケーション同士の相互運用を実現します。WebLogic Server 6.1 の Web サービス実装では、J2EE プログラミング モデルに対する簡単な拡張機能が提供されており、J2EE アプリケーションを Web サービスとして公開できます。WebLogic Web サービスは標準に準拠しており、WebLogic Server 以外のサーバでホストされる Web サービスと相互運用できます。Java クライアント アプリケーションからでも、非 Java クライアント アプリケーション (Microsoft SOAP Toolkit クライアントなど) からでも、WebLogic Web サービスを呼び出すことができます。WebLogic Web サービスを開発および呼び出す方法の詳細については、『WebLogic Server Web サービス プログラマーズ ガイド』を参照してください。非推奨となった機能と API」
- 1-30 ページの「マニュアルとサンプル」
- 1-31 ページの「J2EE と規格について」
- 「動作確認状況」

WebLogic API に関する注意事項

- 1-14 ページの「アプレット」
- 1-15 ページの「Beanshell コード」
- 1-15 ページの「クラスタ」
- 1-15 ページの「WebLogic Server 6.1 でのデプロイメント」
- 1-16 ページの「エンタープライズ JavaBean (EJB)」
- 1-17 ページの「Hypertext Transfer Protocol (HTTP) 1.1」
- 1-18 ページの「J2EE コネクタ」
- 1-19 ページの「Java Database Connectivity (JDBC)」
- 1-19 ページの「JavaMail」
- 1-20 ページの「Java Transaction API (JTA)」
- 1-20 ページの「JMS」
- 1-21 ページの「JSP」
- 1-22 ページの「セキュリティ」
- 1-23 ページの「サーブレットと Web アプリケーション」
- 1-24 ページの「SNMP」
- 1-24 ページの「システム管理」
- 1-27 ページの「WebLogic Tuxedo Connector」
- 1-28 ページの「XML」
- 1-29 ページの「Web サービス」

アプレット

JDK に付属する `tools.jar` の問題のため、アプレットがクライアントに正しく提供されません。`tools.jar` がサーバのクラスパスにある場合は、サーバの CLASSPATH において `weblogic.jar` を `tools.jar` より前で指定してください。

Beanshell コード

WebLogic Server では、Beanshell というオープン ソース製品が使用されています。Beanshell を最初に開発したのは、Pat Niemeyer です。Pat Niemeyer が作成した部分には、「Copyright (C) 2000. All Rights Reserved.」が設定されています。Beanshell は、Sun Public License および GNU Lesser General Public License で入手できます。Beanshell のソース コードは、http://www.beanshell.org にあります。

クラスタ

WebLogic Server のクラスタでは、共有ネットワーク ドライブは必要ありません。WebLogic Server とアプリケーションの両方を、ローカル ファイル システムにインストールできます。

クラスタ、DNS、NT のマルチホーム

クラスタ環境の Windows NT でマルチホームを使用する場合は、名前付けの問題に注意する必要があります。たとえば、マルチホームの Windows NT マシンでクラスタが動作しており、クラスタ内の 1 つのサーバがマシン名と同じ DNS 名にバインドされている場合は、名前の衝突が発生する場合があります。URL でその DNS 名を使用してサーバにアクセスしようとすると、Windows NT ではその DNS 名がマルチホーム Windows NT マシンのいずれかの IP アドレスに変換される場合があります。この場合、要求が間違ったアドレスに送られる可能性があります。DNS 名は、マシン名と同じにしないでください。

WebLogic Server 6.1 でのデプロイメント

WebLogic Server 6.1 では、アプリケーションの対象になっているサーバ インスタンスの 1 つでそのアプリケーションを更新すると、対象になっているすべてのサーバでアプリケーションが更新されます。たとえば、アプリケーションの対象がクラスタの場合、クラスタを構成するサーバ インスタンスの 1 つでアプリケーションを更新すると、アプリケーションはクラスタの全メンバで更新されます。同様に、クラスタとスタンドアロン サーバ インスタンスがアプリケーションの対象になっている場合は、スタンドアロン サーバのインスタンスでアプリケーションを更新すると、クラスタでもアプリケーションが更新されます。また、逆の場合も同様の結果になります。

アプリケーションまたはコンポーネントを対象のサーバ インスタンス群のサブセットで選択的に更新する必要がある場合は、アプリケーションのユニークなインスタンスを異なる対象にデプロイします。

新しく利用できるようになったデプロイメント チュートリアル

WebLogic Server 6.1 には、『展開形式 J2EE アプリケーションのデプロイメント』という名前の新しいデプロイメント チュートリアルが含まれています。「サンプルとチュートリアル」のページから利用できます。

エンタープライズ JavaBean (EJB)

WebLogic Server 6.1 では、WebLogic EJB が多くの点で拡張されています。詳細については、『WebLogic エンタープライズ JavaBeans プログラマーズ ガイド』を参照してください。

EJB 1.1 および 2.0 への準拠

WebLogic Server 6.1 は、JavaSoft EJB 1.1 仕様に準拠しています。また、WebLogic には仮の EJB 2.0 仕様のオプション実装も含まれます。WebLogic Server EJB のマニュアルでは、WebLogic Server を使用するために理解する必要のある EJB 2.0 仕様の主要な機能について説明されています。

このリリースでの EJB の新機能は以下のとおりです。

- トランザクション非対応のエンティティ Bean のキャッシュ
- 主キーの自動生成のサポート
- テーブルの自動作成
- Oracle の SELECT HINT
- EJB デプロイメント記述子エディタ
- EJB Client.jar のサポート
- BLOB と CLOB のサポート
- カスケード削除のサポート
- ローカル インタフェースのサポート

- CMP キャッシュのフラッシュのサポート
- チューニングされた CMP 1.1 のサポート

EJB の値渡しに関する J2EE への非準拠

WebLogic Server の EJB デプロイメント プロパティの多くには、パフォーマンスの向上を目的として最適化されたデフォルト値が用意されています。一部のデフォルト値は、EJB の仕様に準拠していません。WebLogic Server を EJB 仕様に準拠させるには、以下のようにプロパティを設定する必要があります。

`weblogic-ejb-jar.xml` で、enable-call-by-reference を False に設定します。

`weblogic-cmp-jar.xml` で、include-updates を True に設定します。

`weblogic-cmp-jar.xml` で、check-exists-on-method を True に設定します。

Hypertext Transfer Protocol (HTTP) 1.1

以下の節では、WebLogic Server 6.1 でサポートされる HTTP の機能について説明します。

Web サーバ

WebLogic Server 6.1 は、大容量の Web サイトを処理できる実践的な Web サーバであり、静的な HTML (テキスト) ファイルのほか、サーブレットや JavaServer Pages (JSP) のサービスにも対応してます。WebLogic Server 6.1 は、ハードウェアベースおよびソフトウェアベースの Web ロードバランシング ソリューションと完全に統合することもできます。WebLogic Server 6.1 では、HTTP 1.1 規格がサポートされています。

各 WebLogic Server 6.1 は、デフォルトの「Web サーバ」と、定義されている任意数の追加 Web サーバをホストします。各追加 Web サーバは、仮想ホスティングというプロセスで、異なる DNS 名に応答するようにコンフィグレーションされます。

Web サーバの属性は、新しい WebLogic Server Administration Console を使用してコンフィグレーションします。

詳細については、『WebLogic HTTP サーブレット プログラマーズ ガイド』を参照してください。

Apache、Netscape、IIS 用の HTTP プラグイン

WebLogic Server 6.1 では、プラグインと WebLogic Server の間のキープアライブ接続が自動的に実装されます。WebLogic Server 6.1 は、HTTP プラグイン使用時の SSL もサポートしています。

Apache プラグインは、Apache 2.0 で使用できます (非 SSL のみ)。Apache 2.0 でも、HTTP 1.1 キープアライブ接続がサポートされています。

SSL と各プラグインの詳細については、『WebLogic Server 管理者ガイド』を参照してください。

HTTP セッション永続性

クッキーを使用して HTTP セッションを永続化する新しいオプションが使用できます。詳細については、「セッションの永続性のコンフィグレーション」を参照してください。

HTML のページと Netscape

WebLogic Server の起動時に Netscape ブラウザが既に動作している場合は、一部の HTML ページが表示されません。たとえば、Netscape が動作しており、Microsoft Internet Explorer がない状態で、[スタート] メニューから [About WebLogic Server] ページを選択した場合、画面は点滅しますが、ページは表示されません。

J2EE コネクタ

BEA WebLogic Server は、J2EE コネクタ アーキテクチャをサポートすることで、Sun Microsystems の J2EE プラットフォーム仕様バージョン 1.3 の実装を引き続き強化していきます。BEA WebLogic J2EE コネクタ アーキテクチャを使用すると、従来のアプリケーションを J2EE プラットフォームに接続できるようになります。その目的は、コンポーネント モデル、トランザクションおよびセキュリティのインフラストラクチャなどの J2EE プラットフォームの長所を利用して、従来のアプリケーションの統合という難問を処理することです。

コネクタ アーキテクチャでは、アプリケーション サーバと従来のアプリケーションの共通インタフェースが定義されています。このインタフェースは、アプリケーション サーバに接続するアプリケーション固有のリソース アダプタに実

装されています。その結果、J2EE プラットフォームの長所を利用するスケーラブルな標準アーキテクチャを使用して、エンタープライズ アプリケーションを容易に統合できるようになります。

詳細については、『WebLogic J2EE コネクタ アーキテクチャ』を参照してください。

Java Database Connectivity (JDBC)

以下の項目は、JDBC の新機能または改良点です。

マルチプール

JDBC マルチプールは、WebLogic Server の 1 つのインスタンスが使用する接続プールのリストを作成します。どの接続を返すかは、コンフィグレーション可能なアルゴリズムが決定します。マルチプールでは、ロードバランシングと高可用性がサポートされます。アプリケーションは、マルチプールを使用することで、分散処理を目的として、またはフェイルオーバの状況において、簡単に別の RDBMS に切り替えることができます。

詳細については、『WebLogic JDBC プログラミング ガイド』を参照してください。

JDBC ベースの永続性

新しいカラム、`wl_max_inactive_interval` が、JDBC ベースの永続性を提供するための `wl_servlet_sessions` テーブルに追加されました。`wl_servlet_sessions` テーブルの詳細については、「データベースの永続ストレージとしての使い方 (JDBC 永続性)」を参照してください。

JavaMail

WebLogic Server 6.1 には、JavaMail 仕様の実装が含まれています。これは、JavaMail 仕様の標準の参照実装です。詳細については、『WebLogic Server アプリケーションの開発』を参照してください。

Java Transaction API (JTA)

WebLogic Server 6.1 では、仕様に準拠した JTA の実装により、分散トランザクションと 2 フェーズ コミット プロトコルがサポートされています。この実装は、WebLogic JMS や WebLogic jDriver for Oracle などの XA 準拠のリソースに対応しています。

詳細については、『WebLogic JTA プログラマーズ ガイド』を参照してください。

JMS

以下は、JMS の新機能または改良点です。

非同期 JMS

JMS ファイル ストアへの同期書き込みを無効にできるようになりました。その結果、信頼性は犠牲になるものの、パフォーマンスが向上します。アプリケーションで JMS の信頼性がそれほど重要でない場合は、この機能の利点を活かすことができます。詳細については、『WebLogic JMS プログラマーズ ガイド』を参照してください。

恒久サブスクリプションの管理

JMS 恒久サブスクリプションを管理するために RuntimeMBean が追加されました (JMSDurableSubscriberRuntimeMBean)。この MBean を使用すると、恒久サブスクリプションのモニタ、削除、修正といった簡単な管理タスクを、WebLogic Server Administration Console から実行できます。

再配信遅延

一時的または外部的な要因でアプリケーションがメッセージを正しく処理できない場合に、メッセージの再配信を遅延させることができます。再配信遅延は、メッセージがロールバックまたは回復されたときに、そのメッセージの再配信を遅延する時間のことです。

JMS が直ちにメッセージを再配信すると、状況がまだ解消されていず、アプリケーションがメッセージをまだ処理できない場合があります。再配信遅延に対応するようにアプリケーションをコンフィグレーションすることで、この問題を解決できます。

配信時間

アプリケーションへのメッセージ配信を、将来の指定した時間にスケジューリングできます。メッセージの配信は、短期間 (秒や分など) でも長期間 (バッチ処理のために数時間後にするなど) でも延期できます。また、相対的な配信時間をミリ秒単位で指定することもできます。この値から、WebLogic JMS はメッセージの絶対的な配信時間を計算します。配信時間に配信されるまで、メッセージは基本的に見えません。これにより、将来の特定の時間に処理をスケジューリングできます。

再配信の制限

WebLogic JMS がアプリケーションにメッセージを再配信する回数を制限できます。WebLogic JMS によるメッセージの再配信が指定した回数だけ失敗すると、そのメッセージは ErrorDestination キューにリダイレクトされます。ErrorDestination キューがコンフィグレーションされていない場合、メッセージは破棄されます。

JSP

WebLogic Server 6.1 では、以下の JSP 機能がサポートされています。

- WebLogic JSP コンパイラ (`weblogic.jspc`) に対するオプション。これにより、Web アプリケーションで EJB を使用する JSP をコンパイルできます。詳細については、「jsp-descriptor 要素」を参照してください。
- 新しいフォーム検証タグの使用。検証タグを使用することにより、HTML フォームから送信されたエントリを簡単に検証できます。詳細については、「WebLogic JSP フォーム検証タグの使い方」を参照してください。
- EJB jar ファイルから JSP タグ ライブラリを作成するツール。詳細については、「WebLogic EJB-to-JSP 統合ツールの使い方」を参照してください。
- JSP 1.2 のサポート

  Sun Microsystems の JSP 1.2 仕様の以下の機能のサポートが追加されています。ただし、JSP 仕様のバージョン 1.2 は提示された最終的な草案であり、変更される場合があります。アプリケーションで JSP 1.2 の機能を使用する予定の場合は、JSP 1.2 の最終仕様がまだ公開されておらず、将来変更される可能性があることに注意してください。

- `<jsp:include flush="false">`。この機能により、インクルードされた JSP のバッファをフラッシュするタイミングを制御できます。詳細については、「リクエストのインクルード」を参照してください。
- JSP タグに対する `TryCatchFinally` インタフェース。詳細については、「タグ本体内の例外処理」を参照してください。
- タグ ライブラリ バリデータ。この機能を使用すると、JSP を検証するクラスを記述できます。詳細については、「タグ ライブラリ バリデータの使用」を参照してください。
- `IterationTag` クラス。詳細については、「タグ本体の反復処理」を参照してください。
- タグ ライブラリ記述子内での変数の定義。詳細については、「新しいスクリプト変数の定義」を参照してください。

セキュリティ

WebLogic Server 6.1 のセキュリティ機能には以下のものがあります。

- JAAS ログインおよびレルムベースの認証のサポート
- 監査トレイル生成の拡張サポート
- 改善された ACL
- パスワード推測攻撃の防止
- サービス拒否攻撃の防止
- RDBMS レルムの改善された管理機能

詳細については、『WebLogic Security プログラマーズ ガイド』を参照してください。

デフォルトおよびカスタムの Login モジュール

WebLogic Server は、デフォルトの LoginModule (`weblogic.security.internal.ServerLoginModule`) を使用して、サーバの初期化時に認証情報を収集します。デフォルトの Login モジュールを変更するには、`server.policy` ファイルを編集し、デフォルト Login モジュール名をカスタム Login モジュール名に置き換えます。

server.policy ファイルで、カスタム Login モジュール名をデフォルト Login モジュールの前に記述して指定することもできます。WebLogic Server の JAAS 実装は、server.policy ファイルで定義されている順番に従って Login モジュールを使用します。デフォルト Login モジュールは、実行の前に既存のシステム ユーザ認証の定義をチェックし、既に定義されている場合は何の処理も行いません。

デフォルト Login モジュールでは、システム ユーザ名とパスワードの両方に対して JVM プロパティを定義する必要があります。それぞれ、weblogic.management.username と weblogic.management.password で指定します。カスタム Login モジュールを使用するには、各モジュールの仕様に従ってこのプロパティを設定する必要があります。

サーブレットと Web アプリケーション

サーブレットと Web アプリケーションには、以下の新機能と変更点があります。

- Web アプリケーション イベント (サーブレット 2.3 のみ)

  Web アプリケーション イベントを使用すると、サーブレット コンテキスト (各 Web アプリケーションは独自のサーブレット コンテキストを使用する)、または HTTP セッション オブジェクトの状態の変化を通知できます。これらの状態の変化に応答するイベント リスナ クラスを記述できます。詳細については、「アプリケーション イベントとリスナ」を参照してください。

- フィルタ (サーブレット 2.3 のみ)

  フィルタとは、Web アプリケーションのリソースに対するリクエストに応答して起動される Java クラスのことです。これらのリソースには、Java サーブレット、JavaServer Pages (JSP)、静的リソース (HTML ページや画像など) が含まれます。フィルタでは、リクエストを捕捉して、応答の検査と変更を行ったり、オブジェクトの要求や他のタスクを実行したりすることができます。詳細については、「フィルタ」を参照してください。

- Web アプリケーション間のシングル サインオン。サーバでは、サーブレット 2.2 仕様で指定されているとおりに、Web アプリケーション レベルではなく、Web サーバ レベルでユーザが追跡されます。

- クッキーのフォーマット

  - サービス パックが適用されていない WebLogic Server 6.0 によって作成されたクッキーは、WebLogic Server 6.1 では認識されません。

- クッキーには、Web アプリケーションのコンテキスト パスが含まれなくなりました。この変更により、ドメイン内のすべての Web アプリケーションに対する認証が 1 回で済みます。Web アプリケーションでユニークな認証を使用する場合、または Web アプリケーション固有のクッキーを生成する場合は、Web アプリケーションごとのクッキー名を `weblogic.xml` デプロイメント記述子で指定できます。詳細については、「複数の Web アプリケーション、クッキー、および認証」を参照してください。
- クッキー内の区切り記号が、「/」と「|」から「!」に変更されました。これまでは、「/」区切り記号は URL 書き換えエラーの原因に、また「|」区切り記号は Wireless Application Protocol (WAP) ゲートウェイのエラーの原因になっていました。現在では、URL 書き換えも WAP も WebLogic Server で正常に機能します。

Sun Microsystems のサーブレット仕様バージョン 2.3 は、提示された最終的な草案であり、変更される場合があります。アプリケーションでバージョン 2.3 で追加された機能を使用する予定の場合は、最終仕様がまだ公開されておらず、将来変更される可能性があることに注意してください。

SNMP

WebLogic Server 6.1 は、Simple Network Management Protocol (SNMP) エージェントとして機能できます。WebLogic SNMP エージェントは、SNMP マネージャからのリクエストに応答し、SNMP トラップ通知を SNMP マネージャに送信するサービスとして動作します。WebLogic SNMP エージェントは、標準の Java Management Extension (JMX) インタフェースを使用して、WebLogic リソースにアクセスします。詳細については、『WebLogic SNMP 管理ガイド』を参照してください。

システム管理

インストール

WebLogic Server 6.1 には、Windows および UNIX の両システムに WebLogic Server を簡単にインストールできるインストール プログラムがあります。このインストーラは、配布キットを復元し、基本的なコンフィグレーションを行い、

WebLogic Server を使用するためのショートカットをセットアップします。さらに、パッケージには JDK が同梱されているので、サーバをすぐに実行することができます。

詳細については、『WebLogic Server インストール ガイド』を参照してください。

インターナショナライゼーション

WebLogic Server 6.1 では、2 バイト文字セットが必要な言語を含むあらゆる言語のコンテンツを配信できます。新しいインターナショナライゼーション API の使い方については、『WebLogic Server インターナショナライゼーション ガイド』を参照してください。

さらに、WebLogic Server 6.1 の漢字バージョンが利用できるようになります。詳細については、BEA の販売担当者に問い合わせてください。

Administration Console

改良された管理アーキテクチャでは、WebLogic Server の動作中のインスタンスに対するコンフィグレーションを動的に変更できます。Web ベースの Administration Console は、Java Management Extension (JMX) 規格の実装である WebLogic Administration Service への入り口となるウィンドウです。Administration Console では、属性のコンフィグレーション、アプリケーションとコンポーネントのデプロイメント、リソース使用状況のモニタ、ログ メッセージの表示、デプロイメント記述子や `.xml` ファイルの編集などの管理作業を行うことができます。Administration Console の機能は以下のとおりです。

- ブラウザ インタフェース
- ユーザとセキュリティの一元的な管理
- 動的なコンフィグレーション管理
- J2EE アーカイブのデプロイメント記述子の編集ツール
- ドメインの管理
- クラスタの管理
- 一元的なアプリケーション リポジトリの提供
- サーバのステータスのメトリックの提供
- ドメイン内のすべてのサーバのログ メッセージに対する一元的なアクセス

詳細については、『WebLogic Server 管理者ガイド』を参照してください。

プロダクション モードと開発モード

プロダクション モードと開発モードの切り替えに使用できるフラグが用意されました。サーバが起動すると、プロダクション モードまたは開発モードのどちらを使用するのかに関係なく、`config.xml` ファイルにコンフィグレーション情報のあるすべてのアプリケーションがロードおよびデプロイされます。開発モードで実行する場合は、サーバの起動後に `..\applications` ディレクトリに配置されたすべてのアプリケーションもデプロイまたは再デプロイされます。これは動的デプロイメントと呼ばれ、アプリケーションを開発する際に便利です。サーバを停止すると、動的にデプロイされたアプリケーションのコンフィグレーション情報がそのアプリケーションのドメインの `config.xml` ファイルに書き込まれます。プロダクション / 開発フラグは、ドメインの起動スクリプトに追加できます。このフラグを追加しない場合、開発モードがアクティブになります。モードを変更するには、次の手順を行います。

1. テキスト エディタでドメインの起動スクリプトを開きます。

2. STARTMODE= で始まる行を編集して、値 true または false を追加します。true はプロダクション モード、false は開発モードです。

ツール

WebLogic Server 6.1 に付属する `weblogic.refresh` ツールを使うと、アプリケーションを再デプロイしないで、アプリケーションの静的コンポーネントを更新できます。`weblogic.refresh` を使えば、次のような静的ファイルの更新、追加、または削除を行うことができます。

- JSP
- XML ファイル
- HTML ファイル
- .gif や .jpg などの画像ファイル
- テキスト ファイル

WebLogic Tuxedo Connector

WebLogic Server 6.1 では、WebLogic Tuxedo Connector に以下のような制限があります。

- WebLogic Tuxedo Connector ゲートウェイに対するコンフィグレーションの動的な変更がサポートされていません。
- バッファ表示機能がサポートされていません。
- Tuxedo アプリケーションからの着信 RMI/IIOP トランザクションがサポートされていません。
- クラスタがサポートされていません。クラスタ環境で使用できる WebLogic Tuxedo Connector のインスタンスは 1 つだけです。
- VMS、AS/400、OS/390 の各プラットフォーム上で稼働する Tuxedo 6.5 は、サポートしていません。

WebLogic と Tuxedo の相互運用性

WebLogic Tuxedo Connector (WTC) は、WebLogic Server アプリケーションと、Tuxedo ATMI、CORBA Java、および CORBA C++ の各サーバ アプリケーションの間の相互運用を可能にします。WTC の tBridge 機能は、Tuxedo /Q および JMS の高度なメッセージング サービスを提供します。詳細については、「WebLogic Tuxedo Connector」を参照してください。

BEA WebLogic C++ クライアント

WebLogic Server 6.1 SP03 は、C++ クライアントとの相互運用に Tuxedo 8.0 C++ Client ORB を使用します。Tuxedo リリース 8.0 RP 56 以上が必要です。Tuxedo C++ Client ORB の入手方法については、BEA のサービス担当者にお問い合わせください。WebLogic Server と Tuxedo C++ Client ORB の相互運用方法の詳細については、『Tuxedo と Tuxedo クライアントを使用した RMI-IIOP』を参照してください。

XML

WebLogic Server 6.1 では、XML が必要不可欠なコンポーネントとしてサポートされています。サーバ間およびサーバとクライアントの間で XML を生成および利用するには、JSP を使用できます。WebLogic Server 6.1 は、JSP に対する XSL 処理タグをサポートしています。EJB は、XML を使用して、データのポータビリティを実現するデプロイメント プロパティを記述します。サーバでは、Administration Console で管理される DTD の XML スキーマ リポジトリが提供されます。

WebLogic Server 6.1 の XML サブシステムには、以下のような新機能と改良点が追加されています。

- JAXP API が 1.1 にアップグレードされました。
- 中小サイズの XML ドキュメントの解析に特化した、高性能な XML パーサが組み込まれました。この XML パーサは、非検証パーサであり、SAX モードでのみ使用できます。
- 組み込みの Apache Xerces パーサが 1.3.1 にアップグレードされました。
- 組み込みの Apache Xalan トランスフォーマが 2.0.1 にアップグレードされました。
- XML レジストリが、2 つの異なるセクション (パーサとトランスフォーマをコンフィグレーションするセクションと、外部エンティティの解決をコンフィグレーションするセクション) に分割されました。その結果、XML レジストリがより簡単で、直観的に使用できるようになりました。
- XML レジストリで、エンティティをローカルにコピーするだけでなく、URL を介してアクセスするローカル コンピュータ上の外部エンティティをキャッシュできるようになりました。外部エンティティのキャッシュは、XML レジストリを使用して、コンフィグレーションおよびモニタできます。
- 6.0 で提供されていた XML のカスタム パーサ生成機能は非推奨となりました。そのため、以前にカスタム生成のパーサを使用した箇所では、高性能の WebLogic XML パーサを使用する必要があります。

XML の詳細については、『WebLogic XML プログラミング ガイド』を参照してください。

XML エディタ

WebLogic Server に、XML ファイルの作成と編集のためのシンプルで使いやすいツールが用意されました。このツールを使うと、指定した DTD または XML スキーマに従って XML コードの有効性を検証できます。XML エディタは、Windows マシンまたは Solaris マシン上で使用でき、dev2dev Online からダウンロードできます。

Web サービス

WebLogic Server™ の現在のリリースには、WebLogic Web サービスという新機能が追加されています。Web サービスは、Web および XML の標準規格の新しいファミリであり、プログラミングモデル全体にわたる、柔軟に結合されたアプリケーション同士の相互運用を実現します。WebLogic Server 6.1 の Web サービス実装では、J2EE プログラミング モデルに対する簡単な拡張機能が提供されており、J2EE アプリケーションを Web サービスとして公開できます。WebLogic Web サービスは標準に準拠しており、WebLogic Server 以外のサーバでホストされる Web サービスと相互運用できます。Java クライアント アプリケーションからでも、非 Java クライアント アプリケーション (Microsoft SOAP Toolkit クライアントなど) からでも、WebLogic Web サービスを呼び出すことができます。WebLogic Web サービスを開発および呼び出す方法の詳細については、『WebLogic Server Web サービス プログラマーズ ガイド』を参照してください。

非推奨となった機能と API

以下の機能は WebLogic Server 6.1 では非推奨とされており、将来のバージョンでもサポートされません。

- `weblogic.db.jdbc.EventfulTableDataSet`
- WebLogic WorkSpace
- WebLogic Event
- WebLogic Time
- WebLogic JDBC t3 ドライバ
- ZAC

非推奨になった WebLogic Server のクラスのリストについては、http://edocs.beasys.co.jp/e-docs/wls61/javadocs/deprecated-list.html を参照してください。

マニュアルとサンプル

WebLogic Server 6.1 に付属するサンプルには、ant ビルド スクリプトが含まれています。これらのスクリプトは build.xml ファイルとしてサンプルに付属し、WebLogic Server 6.1 がサポートするすべてのプラットフォームで実行できます。ビルド スクリプトを実行するには、サンプルの適切なディレクトリを探し、コマンドラインで次のように入力してください。

```
ant
```

以下のサンプルとマニュアルは、WebLogic Server 6.1 で新しく導入されたものです。

- config.xml ファイルの新しいドキュメントについては、『WebLogic Server コンフィグレーション リファレンス』を参照してください。
- WebLogic Server 6.1 のパフォーマンスとチューニングの情報については、『WebLogic Server パフォーマンス チューニング ガイド』を参照してください。
- WebLogic Web サービスのドキュメントとサンプルについては、『WebLogic Web サービス プログラマーズ ガイド』を参照してください。
- WebLogic Server 6.1 は、Simple Network Management Protocol (SNMP) エージェントとして機能できます。新しい『SNMP 管理ガイド』を参照してください。
- JCA のサンプルとドキュメント「WebLogic J2EE コネクタ アーキテクチャの管理」は、『WebLogic Server 管理者ガイド』にあります。
- 『Web アプリケーションのアセンブルとコンフィグレーション』には、Web アプリケーションのすべてのコンフィグレーション、アセンブリ、およびデプロイメントの情報が記載されています。以前は、これらの情報の一部は、『管理者ガイド』および『WebLogic Server アプリケーションの開発』に記載されていました。

- WebLogic Server 6.1 には、無線接続について紹介するサンプルが用意されています。
- 詳細な移行チュートリアルでは、WebLogic Server の以前のリリースにデプロイされたアプリケーションを WebLogic Server 6.1 に適合させる方法が示されています。
- Oracle のカスケード削除を示す、新しいサンプル パッケージがあります。EJB 2.0 仕様で指定されている、2 つの EJB 間の 1 対多の関係を使用します。
- 主キー自動生成の EJB サンプルが追加されています。これらのサンプルでは、データベース テーブルへの挿入時に主キーが自動的に生成されます。
- マニュアルとサンプルについては、「WebLogic Tuxedo Connector」を参照してください。

WebLogic Server ツアー

WebLogic Server ツアーは、全面的に改訂されています。このツアーでは、機能を紹介する Pet Store アプリケーションを使用して、WebLogic Server の概要が説明されています。ツアーは、[スタート] メニューから利用できます。現在の Pet Store は、Sun の Java Pet Store バージョン 1.1.2 に基づいており、Ant 1.3 でビルド スクリプトを使用するようにアップグレードされています。

J2EE と規格について

Java Development Kit

JDK には、Java の実行時環境 (Java 仮想マシン : JVM) と、Java アプリケーションのコンパイルおよびデバッグ用のツールが用意されています。WebLogic Server 6.1 には、Sun Microsystems の JDK 1.3.1rc2 が付属しています。

BEA WebLogic Server 6.1 は、高度な J2EE 1.3 の機能を実装する最初の e コマース トランザクション プラットフォームです。J2EE のルールに準拠するために、2 種類のダウンロードが用意されています。1 つは J2EE 1.3 の機能が有効になっているもので、もう 1 つは J2EE 1.2 の機能に制限されているものです。いずれのダウンロードもコンテナは同じで、利用可能な API だけが異なります。

J2EE 1.2 の機能に加えて J2EE 1.3 の機能を備える WebLogic Server 6.1

このダウンロードでは、WebLogic Server は J2EE 1.3 の機能をデフォルトで使用して動作します。これらの機能には、EJB 2.0、JSP 1.2、サーブレット 2.3、および J2EE コネクタ アーキテクチャ 1.0 が含まれます。J2EE 1.3 の機能を有効にして WebLogic Server 6.1 を実行しても、J2EE 1.2 のアプリケーションは完全にサポートされます。J2EE 1.3 機能の実装では、適切な API 仕様の最終ではないバージョンが使用されます。したがって、J2EE 1.3 の新機能を使用する BEA WebLogic Server 6.1 用に開発されたアプリケーション コードは、BEA WebLogic Server の今後のリリースでサポートされる J2EE 1.3 プラットフォームとは互換性を持たない場合があります。

以下の節では、J2EE 1.3 の API クラスとデプロイメント記述子のうち、WebLogic Server 6.1 で実装されていないものについて説明します。

エンタープライズ JavaBeans のデプロイメント記述子

WebLogic Server 6.1 では、以下の例外を除き、Sun Microsystem の `ejb-jar.xml` 文書型定義 (DTD) ファイルで定義されているすべての EJB 2.0 デプロイメント記述子が完全に実装されています。

- `exclude_list`
- `unchecked`
- `run_as`

注意： `ejb-jar.xml` 文書型定義 (DTD) ファイルは、Sun Microsystem EJB 2.0 仕様の一部です。

JDBC API クラス

API	クラスまたは インタフェース	実装されていない メソッド	備考
java.sql	CallableStatement	getResultSet(int parameterIndex)	Oracle の拡張機能である OracleCallableStatement のメソッドとして実装されている。Oracle Thin Driver などの、OracleCallableStatement 拡張機能 をサポートする JDBC ドライバを使用する必要がある。
	Driver	getPropertyInfo(java.lang.String url, java.util.Properties info)	実装されていない。

JSP 1.2 API クラス

API	クラス	J2EE 1.3 との相違
javax.servlet.jsp	java.beans.Beans	jsp:id メカニズムは実装されていない。

注意： 次の機能は実装されていません。

パッケージ化されたタグ ライブラリを含む JAR を WEB-INF/lib ディレクトリに格納すれば、要求時にそのクラスを利用できるようになります。

JSP 1.2 のデプロイメント記述子

- Web アプリケーションでは、taglib.tld の <listener> 要素は登録されません。
- taglib.tld の <example> 要素は受け付けられません。
- WebLogic Server 6.1 は、TaglibraryValidator.validate() メソッドの文字列を返す古いシグネチャを使用しています。

サーブレット API クラス

API	クラス	J2EE 1.3 との相違点
javax.servlet	Filter	WLS がサポートする Filter インタフェースには、次のメソッドが含まれている。 doFilter(ServletRequest,ServletResponse,FilterChain) getFilterConfig(): この Filter に対する FilterConfig を返す。 setFilterConfig(Fil-terConfig): Filter をインスタンス化して Filter-Config に渡すとき、コンテナはこのメソッドを呼び出す。
	ServletContext	ServletContext.getResourcePaths() メソッドにディレクトリ引数 (String) が追加されている。
	ServletContextAttributesListener	最終仕様における名前が、次のように変更されている。 ServletContextAttributeListener
javax.servlet.http	HttpServletResponse	HttpServletResponse にステータスコード 307 (一時的リダイレクト) が追加されている。
	HttpSession	getServletContext() メソッドが HttpSession に追加されている。
	HttpSessionAttributesListener	最終仕様における名前が、次のように変更されている。 HttpSessionAttributeListener

サーブレットのデプロイメント記述子

WebLogic Server 6.1 では、以下の例外を除き、サーブレット 2.3 のすべてのデプロイメント記述子が完全に実装されています。

- ejb-local-ref

- run_as

J2EE 1.2 認定の WebLogic Server 6.1

このダウンロードでは、WebLogic Server は J2EE 1.3 機能が無効な状態をデフォルトとして動作し、J2EE 1.2 の仕様と規定に完全に準拠します。

J2EE 1.2 および 1.3 の製品 CD インストーラ

配布キットは両方とも、http://www.beasys.co.jp/evaluation/index.html からダウンロードできるほか、WebLogic Server 6.1 の製品 CD でも提供されます。Windows マシンでは、CD を挿入すると、J2EE 1.3 機能の有効な WebLogic Server のインストーラが自動的に開始します。

規格のサポート

規格	バージョン
HTTP	1.1
J2EE Connector Architecture	1.0
J2EE EJB	2.0
J2EE JDBC	2.0
J2EE JNDI	1.2
J2EE JSP	1.1
J2EE JTA	1.0.1
J2EE JMS	1.0.2
J2EE RMI	1.0
RMI/IIOP	1.0
J2EE Servlet	2.2

規格	バージョン
LDAP	2
SSL	3
X.509	3

2 WebLogic Server 5.1 からバージョン 6.1 へのアップグレード

以下の節では、WebLogic Server 5.1 から WebLogic Server 6.1 にアップグレードするために必要な情報が提供されています。

- WebLogic Server コンフィグレーションのアップグレード : 主な手順
- weblogic.properties ファイルの変換
- WebLogic Server 6.x でのクラスのロード
- 起動スクリプトの修正
- Oracle のアップグレード
- JMS のアップグレード
- JVM のアップグレード

WebLogic Server コンフィグレーションのアップグレード：主な手順

WebLogic Server 5.1 から WebLogic Server 6.1 にアップグレードするには、次の手順を行います。

1. WebLogic Server 6.1 をインストールします。`http://edocs.beasys.co.jp/e-docs/wls61/install/index.html` の『WebLogic Server インストール ガイド』を参照してください。
2. `weblogic.properties` ファイルを変換します。`weblogic.properties` ファイルを変換する方法については、2-2 ページの「weblogic.properties ファイルの変換」および Administration Console のヘルプを参照してください。
3. Java システム `CLASSPATH` にクラスを追加します。詳細については、2-3 ページの「WebLogic Server 6.x でのクラスのロード」を参照してください。
4. WebLogic 6.1 で利用できるように起動スクリプトを修正します。2-4 ページの「起動スクリプトの修正」を参照してください。
5. JMS をアップグレードします。WebLogic Server 5.1 から、多くの新しいコンフィグレーション属性が JMS に追加されています。詳細については、2-5 ページの「JMS のアップグレード」を参照してください。
6. JVM をアップグレードします。WebLogic Server 6.1 では JDK 1.3 を使用する必要があります。詳細については、「動作確認状況」ページを参照してください。

weblogic.properties ファイルの変換

以前のバージョンの WebLogic Server では、コンフィグレーション プロパティは `weblogic.properties` ファイルに格納されていました。WebLogic Server 6.0 以降のバージョンでは、サーバのコンフィグレーション属性は永続的な `.xml` ファイルの `config.xml` に格納されます。アプリケーションのコンフィグレーション属性は、アプリケーション固有の `.xml` ファイルに格納されます。アプリケー

ションによって、web.xml、weblogic.xml、または application.xml ファイルがアプリケーションに関連付けられます。以前のリリースの WebLogic でコンフィグレーションを扱っていた weblogic.properties ファイルは、WebLogic Server 6.0 以降のバージョンでは使用されなくなりました。

既存の weblogic.properties ファイルを適切な .xml に変換するには、Administration Console を使用します。weblogic.properties ファイルの変換手順については、Console のヘルプを参照してください。

- config.xml ファイルは直に編集しないでください。コンフィグレーションには、Administration Console、コマンドライン ユーティリティ、またはコンフィグレーション API を通じてアクセスします。WebLogic Server のコンフィグレーションの詳細については、『管理者ガイド』を参照してください。
- セキュリティ プロパティは fileRealm.properties ファイルに格納されます。
- weblogic.properties ファイルからプロパティを取得するメソッドを提供していた weblogic.common.ConfigServicesDef API はこのバージョンから削除されています。

WebLogic Server 6.x でのクラスのロード

WebLogic Server の以前のバージョンでは、クラスの動的なロードを容易にするために WebLogic クラスパス プロパティ（weblogic.class.path）が使用されていました。WebLogic 6.0 以降のバージョンでは、weblogic.class.path は必要ありません。Java システム クラスパスからクラスをロードできます。

これまで weblogic.class.path で指定していたクラスを標準の Java システム クラスパスに含めるには、コマンドラインで -classpath オプションを使用するか、または CLASSPATH 環境変数を設定します。

起動スクリプトの修正

以前のバージョンで WebLogic Server 起動スクリプトを使用していた場合は、バージョン 6.1 で利用できるようにスクリプトを変更する必要があります。

- 「クラスパス オプションの設定」で説明されているとおりに、起動スクリプトを変更します。WebLogic クラスパスは使われなくなりました。「WebLogic Server 6.x でのクラスのロード」で説明したように、Java システム クラスパスを使います。
- ライセンス ファイルをクラスパスで指定する必要はなくなりました。
- 新しい管理システムでは、管理サーバと管理対象サーバが明確に区別されます。したがって、サーバをどのように管理するのかに応じて、サーバを起動するスクリプトを記述し直す必要があります。新しいコマンドと必須の引数については、「WebLogic Server の起動と停止 」を参照してください。

Oracle のアップグレード

BEA では、Oracle のサポート ポリシーを反映して、『WebLogic jDriver for Oracle のインストールと使い方』で説明している Oracle のリリースをサポートしています。BEA では、現在 Oracle クライアント バージョンの 7.3.4、8.0.4、8.0.5、および 8.1.5 はサポートしていません。

Oracle クライアント バージョン 7.3.4 を使用する場合は、下位互換性を持つ oci816_7 共有ライブラリを使用します。上記の説明のとおり、現在、BEA ではこのコンフィグレーションをサポートしていません。

Oracle クライアント バージョン 8.1.7 をアップグレードする場合、または WebLogic jDriver と Oracle データベースの詳細を参照する場合は、『WebLogic jDriver for Oracle のインストールと使い方』を参照してください。

サポートされているプラットフォームと、DBMS およびクライアント ライブラリについては、「動作確認状況」ページを参照してください。最新の動作確認情報は、必ず「動作確認状況」ページで公開されます。

JMS のアップグレード

次の表は、WebLogic Server 5.1 以降に追加された WebLogic JMS コンフィグレーション属性を示しています。

既存のアプリケーションの移行については、「WebLogic Server アプリケーションの移行」を参照してください。

コンポーネント	JMS 属性	説明
JMS 接続ファクトリ	[最大メッセージ]	非同期セッションの間に存在し、メッセージリスナにまだ渡されていないメッセージの最大数。この属性のデフォルト値は 10。
	[超過時のポリシー]	マルチキャスト セッションでの超過時のポリシー。未処理のメッセージ数が [最大メッセージ] 属性の値に達すると、指定されたポリシーに基づいてメッセージが破棄される。 [新しいメッセージを保持] に設定されている場合は、最新のメッセージが最古のメッセージよりも優先され、必要に応じて最古のメッセージが破棄される。[古いメッセージを保持] に設定されている場合は、最古のメッセージが最新のメッセージよりも優先され、必要に応じて最新のメッセージが破棄される。 この属性のデフォルト値は [古いメッセージを保持]。
	[メッセージの短縮を許可]	`onMessage()` メソッドの呼び出しで `close()` メソッドの発行を可能にするメッセージ コンシューマが接続ファクトリで作成されるかどうかを指定するフラグ。この属性はデフォルトでは無効。
	[トランザクション タイムアウト]	処理されるセッションのタイムアウト値（ミリ秒単位）。この属性のデフォルト値は 3600。
	[ユーザ トランザクションを有効化]	JTA 対応のセッションが接続ファクトリで作成されるかどうかを指定するフラグ。この属性はデフォルトでは無効。

コンポーネント	JMS 属性	説明
	[デフォルト再送遅延]	ロールバックまたは回復されたメッセージが再配信されるまでの、デフォルトの遅延時間（ミリ秒単位）。この属性のデフォルト値は 0。
	[デフォルト送信時間]	メッセージが生成されたときからメッセージが送り先に表示可能になるときまでのデフォルトの遅延時間（ミリ秒単位）。この属性のデフォルト値は 0。
	[XA コネクション ファクトリを有効化]	キューまたはトピック接続ファクトリの代わりに、XA キューまたは XA トピック接続ファクトリが返されるのかどうかを指定するフラグ。`getXAResource` メソッドを持つ XA キュー セッションまたは XA トピック セッションを返すために使用できる。
	[確認応答ポリシー]	接続ファクトリのメッセージ確認応答ポリシー。この属性は、非トランザクション セッションで `CLIENT_ACKNOWLEDGE` 確認応答モードを使用する実装にのみ適用される。有効な値は次のとおり。 ■ [All] － どのメッセージで確認応答メソッドが呼び出されたのかに関係なく、特定のセッションで受信されたすべてのメッセージで確認応答が行われる。 ■ [Previous] － 確認応答メソッドを呼び出したメッセージまでの、特定のセッションで受信されたすべてのメッセージで確認応答が行われる。 この属性のデフォルト値は [All]。
JMS サーバ	[デフォルト JMS 接続ファクトリを有効化]	JMS のデフォルト接続ファクトリが JMS サーバでインスタンス化されるかどうかを指定するフラグ。この属性はデフォルトでは有効。

コンポーネント	JMS 属性	説明
	［Bytes Paging Enabled］	■ [Bytes Paging Enabled] チェック ボックスを選択しない場合は (False)、サーバのバイト ページングが明示的に無効になる。 ■ [Bytes Paging Enabled] チェック ボックスを選択した場合は (True)、ページング ストアがコンフィグレーションされていて、[最小バイトしきい値] 属性と [最大バイトしきい値] 属性が両方とも -1 より大きいと、バイト ページングが有効になる。 ■ [最小バイトしきい値] 属性または [最大バイトしきい値] 属性のいずれかが定義されていない場合、または -1 に設定されている場合は、[Bytes Paging Enabled] チェック ボックスが選択されていても (True)、サーバのバイト ページングは暗黙的に無効になる。
	［Messages Paging Enabled］	■ [Messages Paging Enabled] チェック ボックスを選択しない場合は (False)、サーバのメッセージ ページングが明示的に無効になる。 ■ [Messages Paging Enabled] チェック ボックスを選択した場合は (True)、ページング ストアがコンフィグレーションされていて、[最小メッセージしきい値] 属性と [最大メッセージしきい値] 属性が両方とも -1 より大きいと、サーバのメッセージ ページングが有効になる。 ■ [最小メッセージしきい値] 属性または [最大メッセージしきい値] 属性のいずれかが定義されていない場合、または -1 に設定されている場合は、[Messages Paging Enabled] チェック ボックスが選択されていても (True)、サーバのメッセージ ページングは暗黙的に無効になる。

コンポーネント	JMS 属性	説明
	［Paging Store］	非永続メッセージをページングする永続ストアの名前。ページング ストアは、永続メッセージ用または恒久サブスクライバ用と同じストアであってはならない。 異なる JMS サーバが同じページング ストアを使用することができないので、サーバごとに固有のページング ストアをコンフィグレーションする必要がある。
JMS の送り先	[ストアを有効化]	JMS サーバで指定されたバッキング ストアが送り先で使用されるかどうかを指定するフラグ。 このフラグが有効でも、バッキング ストアが定義されていない場合、コンフィグレーションは失敗し、WebLogic JMS は起動しない。このフラグが無効の場合、送り先では永続的なメッセージがサポートされない。このフラグが Default に設定されている場合、送り先では定義されているバッキング ストアが使用される。 この属性のデフォルト値は [デフォルト]。
	[マルチキャスト アドレス]	マルチキャストで使用する IP アドレス。このアドレスは、マルチキャスト コンシューマにメッセージを送信するために使用される。この属性にはデフォルト値がない。
	[マルチキャスト存続時間]	マルチキャストで使用する存続期間値。メッセージがコンシューマにたどり着くまでに通過できるルータの数を指定する。この属性のデフォルト値は 0。
	[マルチキャスト ポート]	マルチキャストで使用する IP ポート。このポートは、マルチキャスト コンシューマにメッセージを送信するために使用される。この属性のデフォルト値は 6001。

コンポーネント	JMS 属性	説明
	[送信時間オーバーライド]	プロデューサや接続ファクトリによって指定された配信時間とは無関係に、メッセージが生成されたときからメッセージが送り先に表示可能になるときまでのデフォルト遅延をミリ秒単位で定義する。デフォルト値 (-1) の場合は、送信時間の設定が送り先によってオーバーライドされることはない。
	[配信モードのオーバライド]	プロデューサによって指定された配信モードとは関係なく、送り先に到着するすべてのメッセージに割り当てられる配信モード。デフォルト値（[配信しない]）の場合、配信モードはオーバーライドされない。
	[再配信の制限]	メッセージがエラー送り先に配置されるまでのメッセージの再配信試行回数。再配信制限に達すると、エラー送り先がコンフィグレーションされているかどうかにより、次のどちらかが行われる。 ■ エラー送り先がコンフィグレーションされていない場合、またはエラー送り先の割り当て数を超過した場合には、永続的メッセージも非永続的メッセージも単に削除される。 ■ エラー送り先がコンフィグレーションされていて、エラー送り先が割り当て数に達している場合には、エラー メッセージがログに記録されて、メッセージは削除される。ただし、メッセージが永続的である場合は、永続ストアに残っている。これにより、永続的メッセージは WebLogic Server が再起動した時点で確実に再配信される。 デフォルト値 (-1) の場合は、送り先によって再配信制限の設定がオーバーライドされることはない。

コンポーネント	JMS 属性	説明
	[Error Destination]	再配信の制限に達したメッセージの送り先。エラー送り先が null の場合、メッセージは削除される。この属性は動的にコンフィグレーションできるが、受信するメッセージのみが影響を受け、格納されているメッセージは影響を受けない。この属性のデフォルト値は [なし]。
	［Bytes Paging Enabled］	■ [Bytes Paging Enabled] を False に設定すると、その送り先に対する送り先レベルのバイト ページングが無効になる。 ■ [Bytes Paging Enabled] を True に設定した場合は、JMS サーバに対してページング ストアがコンフィグレーションされていて、[最小バイトしきい値] 属性と [最大バイトしきい値] 属性が両方とも -1 より大きいと、その送り先に対する送り先レベルのバイト ページングが有効になる。 ■ [Bytes Paging Enabled] を Default に設定すると、テンプレートが指定されている場合は、テンプレートのこの値を継承する。送り先に対してテンプレートがコンフィグレーションされていない場合は、Default は False と同等である。

コンポーネント	JMS 属性	説明
	［Messages Paging Enabled］	■ [Messages Paging Enabled] を False に設定すると、その送り先に対する送り先レベルのメッセージ ページングが無効になる。 ■ [Messages Paging Enabled] を True に設定した場合は、JMS サーバに対してページング ストアがコンフィグレーションされていて、[最小メッセージしきい値] 属性と [最大メッセージしきい値] 属性が両方とも -1 より大きいと、その送り先に対する送り先レベルのメッセージ ページングが有効になる。 ■ [Messages Paging Enabled] を Default に設定すると、テンプレートが指定されている場合は、テンプレートのこの値を継承する。送り先に対してテンプレートがコンフィグレーションされていない場合は、Default は False と同等である。

コンポーネント	JMS 属性	説明
JMS テンプレート	［Bytes Paging Enabled］	■ [Bytes Paging Enabled] チェック ボックスを選択しない場合は (False)、送り先の設定によってテンプレートがオーバーライドされていない限り、JMS テンプレートの送り先に対する送り先レベルのバイト ページングが無効になる。 ■ [Bytes Paging Enabled] チェック ボックスを選択した場合は (True)、JMS サーバに対してページング ストアがコンフィグレーションされていて、[最小バイトしきい値] 属性と [最大バイトしきい値] 属性が両方とも -1 より大きいと、送り先の設定によってテンプレートがオーバーライドされていない限り、JMS テンプレートの送り先に対する送り先レベルのバイト ページングが有効になる。 ■ JMS テンプレート MBean で値が定義されていない場合は、デフォルト値として False が設定されて、JMS テンプレートの送り先に対するバイト ページングは無効になる。

コンポーネント	JMS 属性	説明
	［Messages Paging Enabled］	■ [Messages Paging Enabled] チェック ボックスを選択しない場合は (False)、送り先の設定によってテンプレートがオーバーライドされていない限り、JMS テンプレートの送り先に対する送り先レベルのメッセージ ページングが無効になる。 ■ [Messages Paging Enabled] チェック ボックスを選択した場合は (True)、JMS サーバに対してページング ストアがコンフィグレーションされていて、[最小メッセージしきい値] 属性と [最大メッセージしきい値] 属性が両方とも -1 より大きいと、送り先の設定によってテンプレートがオーバーライドされていない限り、JMS テンプレートの送り先に対する送り先レベルのメッセージ ページングが有効になる。 ■ JMS テンプレート MBean で値が定義されていない場合は、デフォルト値として False が設定されて、JMS テンプレートの送り先に対するメッセージ ページングは無効になる。

JSP

WebLogic Server 5.1 から現在のバージョンに至るまでの間に、JSP include ディレクティブの動作が変更されています。WebLogic Server 5.1 までのバージョンでは JSP include ディレクティブに存在しないページが含まれる場合、警告レベルのメッセージがログに記録されました。WebLogic Server 6.0 以降のバージョンでは、そうした場合に「500 Internal Server Error」が報告され、参照先となる場所に空のファイルを置くことでエラーを回避できます。

JVM のアップグレード

WebLogic Server 6.1 を実行するには、JDK 1.3 にアップグレードする必要があります。Java クライアント アプリケーションでのみ、JDK 1.2 以上のバージョンを使用できます。動作確認された JVM の最新情報については、「動作確認状況」ページを参照してください。

3 WebLogic Server 4.5 および 5.1 アプリケーションのバージョン 6.x への移行

以降の節では、WebLogic Server バージョン 4.5 または 5.1 から WebLogic Server 6.x にアプリケーションを移行する方法を説明するとともに、移行とデプロイメントに関する追加情報を提供します。説明は、WebLogic Server 4.5 および 5.1 から WebLogic Server 6.0 および 6.1 への移行に対するものです。詳細な手順については、*BEA dev2dev* の移行チュートリアルを参照してください。

- weblogic.properties ファイルと .xml ファイル
- WebLogic Server 6.x アプリケーションの種類
- 移行の主な手順
- weblogic.properties ファイルの .xml ファイルへの変換
- weblogic.properties のマッピング表
- Web アプリケーションの移行
- エンタープライズ JavaBean アプリケーションの移行と変換
- エンタープライズ アプリケーションの作成
- J2EE クライアント アプリケーションについて
- JMS アプリケーションの移行
- 移行およびデプロイメントの補足事項

weblogic.properties ファイルと .xml ファイル

WebLogic Server の以前のリリースでは、アプリケーションのコンフィグレーションに `weblogic.properties` ファイルを使用していました。WebLogic 6.0 および 6.1 では、アプリケーションのコンフィグレーションは XML 記述子ファイルと Administration Console を使用して処理します。WebLogic Server の以前のバージョンの `weblogic.properties` ファイルを変換すると、アプリケーションの新しいドメインが作成され、アプリケーションの設定方法を定義する `.xml` ファイルが追加されます。

`config.xml` ファイルは、WebLogic Server のドメイン全体のコンフィグレーションを記述した XML ドキュメントです。`config.xml` ファイルは一連の XML 要素で構成されます。Domain 要素は最上位の要素で、Domain 内のすべての要素は Domain 要素の子です。Domain 要素には、Server、Cluster、および Application 要素などの子要素があります。これらの子要素自身が子を持つ場合もあります。各要素には 1 つまたは複数のコンフィグレーション可能な属性があります。`config.dtd` で定義される属性には、コンフィグレーション API に対応する属性があります。

`weblogic.xml` ファイルには、Web アプリケーション用の WebLogic 固有の属性が入っています。このファイルでは、HTTP セッション パラメータ、HTTP クッキー パラメータ、JSP パラメータ、リソース参照、セキュリティ ロール割り当て、文字セット マッピング、およびコンテナ属性を定義します。

デプロイメント記述子 `web.xml` は、Sun Microsystems のサーブレット 2.3 仕様で定義されています。このデプロイメント記述子を使用して、J2EE 準拠のアプリケーション サーバに Web アプリケーションをデプロイできます。

WebLogic Server 6.x アプリケーションの種類

WebLogic Server 6.0 および 6.1 のような J2EE 準拠のサーバでは、アプリケーションは Web アプリケーション、エンタープライズ JavaBean、エンタープライズ アーカイブ、およびクライアント アプリケーションのいずれかのタイプで作成およびデプロイされます。既存のコンポーネントを WebLogic Server 6.1 に移行するには、適切な J2EE デプロイメント ユニットを作成します。Web アプリケーションは、一般にサーブレット、JSP、および HTML ファイルの集合であり、`.war` ファイルとしてパッケージ化されます。エンタープライズ JavaBean（`.jar` ファイルとしてパッケージ化）は、EJB 仕様に基づいて記述されたサーバサイド Java コンポーネントです。エンタープライズ アーカイブ（`.ear` ファイル）には、EJB コンポーネントと Web アプリケーション コンポーネントの組み合わせが格納されます。クライアント アプリケーションは、Remote Method Invocation（RMI）を使用して WebLogic Server に接続する Java クラスです。これらの各 J2EE デプロイメント ユニットについては、以降の節で詳しく説明します。

移行の主な手順

次に、移行の一般的な手順を簡単に示します。以降の節では、アプリケーションの種類ごとにこの手順をさらに詳しく説明します。*BEA dev2dev* の移行チュートリアルも参照してください。

1. `weblogic.properties` ファイルを .xml ファイルに変換します。
2. WebLogic Server 6.x で新しいドメインを指定します。アプリケーションはこのドメインに移行します。
3. 新しいドメインを示すように起動スクリプトと setenv スクリプトを修正します。
4. WebLogic Server 6.x を再起動します。
5. アプリケーションを移行します。

weblogic.properties ファイルの .xml ファイルへの変換

注意： 変換が正しく行われるようにするには、`weblogic.properties` ファイルで以下を指定する必要があります。

```
weblogic.password.system=gumby1234
```

`weblogic.properties` ファイルを適切な `.xml` に変換するには、次の手順に従います。

1. デフォルトの WebLogic Server 6.1 サーバおよびデフォルトの WebLogic Server 6.1 Administration Console を起動します。WebLogic Server 6.1 の起動の詳細については、「インストール後の作業の実行」を参照してください。
2. WebLogic Administration Console のホーム ページ（`http://localhost:7001/console/index.jsp` など）で、見出し [はじめに] の下にあるリンク [`weblogic.properties` のコンバート] をクリックします。
3. Administration Console のリンクを使用してサーバのファイル システムをたどり、以前のバージョンの WebLogic Server のルート ディレクトリ（`C:\weblogic` など）を見つけます。適切なディレクトリを見つけたら、そのアイコンをクリックして選択します。
4. サーバ別の `weblogic.properties` ファイルまたはクラスタ化 `weblogic.properties` ファイルがある場合は、表示されるウィンドウを使用してそれらのファイルを選択します。**これまで使用してきたバージョンの WebLogic Server の適切なルート ディレクトリが選択されていれば、選択した追加の properties ファイルに関係なくグローバルの `weblogic.properties` ファイルが変換されます。**
5. Console の表示されたウィンドウで新しいドメインの名前を入力します。[コンバート] をクリックします。

weblogic.properties ファイルを変換するプロセスによって、`wlserver6.1/config/`*`domainName`* ディレクトリに config.xml ファイルが作成されます。このファイルには、ドメインに固有のコンフィグレーション情報が収められています。Web アプリケーションは、

wlserver6.1/config/*domainName*/production_apps/DefaultWebApp_serverName/ ディレクトリに作成されます。Web アプリケーションに対する web.xml ファイルと weblogic.xml ファイルは、wlserver6.1/config/*domainName*/production_apps/DefaultWebApp_serverName/WEB-INF/ ディレクトリに格納されます。

注意： 前述の変換ユーティリティは、weblogic.xml ファイルで Java ホームの場所を指定します。変換ユーティリティは、System.getProperty(java.home) を使用してこの場所を読み取ります。つまり、WebLogic Server が変換を目的として起動された Java ホームの場所が指定されます。

このマニュアル全体を通して、作成された新しいドメインのディレクトリを *domainName* と呼びます。WebLogic Server 6.1 のオリジナルのインストール時のデフォルト ドメインは mydomain とします。これは wlserver6.1\config\ ディレクトリに配置されます。

weblogic.properties ファイルの変換時に生成される起動スクリプトは、start*domainName*.cmd （Windows ユーザ用）および start*domainName*.sh （UNIX ユーザ用）という名前で、WebLogic Server 6.x 配布キットの WL6x_HOME/config/*domainName* ディレクトリの下にあります。これらのスクリプトは、新しいドメインでサーバを起動します。

スクリプトおよびサーバの起動の詳細については、「WebLogic Server の起動と停止」を参照してください。

weblogic.properties のマッピング表

weblogic.properties のマッピング表は、どの config.xml、web.xml、または weblogic.xml 属性で weblogic.properties プロパティで以前に実行されていた機能が処理されるのかを示します。Administration Console で属性をコンフィグレーションできる場合は、その属性へのアクセス方法が示されます。

weblogic.properties ファイルのプロパティ	**.xml コンフィグレーション属性**	**Console でのアクセス**
weblogic.administrator.email	config.xml: EmailAddress（Administrator 要素）	
weblogic.administrator.location	config.xml: Notes（自由形式、省略可能）（Administrator 要素）	
weblogic.administrator.name	config.xml: Name（Administrator 要素）	
weblogic.administrator.phone	config.xml: PhoneNumber（Administrator 要素）	
weblogic.cluster.defaultLoadAlgorithm	config.xml: DefaultLoadAlgorithm	[クラスタ	*clustername*	コンフィグレーション	一般	デフォルトのロード バランス アルゴリズム]
weblogic.cluster.multicastAddress	config.xml: MulticastAddress	[クラスタ	*clustername*	コンフィグレーション	マルチキャスト	マルチキャスト アドレス]
weblogic.cluster.multicastTTL	config.xml: MulticastTTL	[クラスタ	*clustername*	コンフィグレーション	マルチキャスト	マルチキャスト生存時間]

weblogic.properties ファイルのプロパティ	**.xml コンフィグレーション属性**	**Console でのアクセス**
weblogic.cluster.name	config.xml ClusterAddress	[クラスタ	*clustername*	コンフィグレーション	一般	クラスタ アドレス]
weblogic.httpd.authRealmName	config.xml: AuthRealmName (WebAppComponent 要素)	[デプロイメント	Web アプリケーション	*applicationname*	コンフィグレーション	その他	認証レルム名]
weblogic.httpd.charsets	config.xml: Charsets (WebServer 要素)	
weblogic.httpd.clustering.enable	config.xml: ClusteringEnabled (WebServer 要素)	
weblogic.httpd.defaultServerName	config.xml DefaultServerName (WebServer 要素)	[サーバ	*servername*	コンフィグレーション	HTTP	デフォルト サーバ名]
weblogic.httpd.defaultServlet	web.xml: <servlet-mapping> 要素の URL パターンでサーブレットのマッピングを定義	
weblogic.httpd.defaultWebApp	config.xml: DefaultWebApp (WebServer 要素)	
weblogic.httpd.enable	config.xml: HttpdEnabled (Server 要素)	
weblogic.httpd.enableLogFile	config.xml: LoggingEnabled (WebServer 要素)	

weblogic.properties ファイルのプロパティ	**.xml コンフィグレーション属性**	**Console でのアクセス**
weblogic.httpd.http.keepAliveSecs	config.xml: KeepAliveSecs（WebServer 要素）	
weblogic.httpd.https.keepAliveSecs	config.xml: HttpsKeepAliveSecs（WebServer 要素）	
weblogic.httpd.indexDirectories	config.xml: IndexDirectoryEnabled（WebAppComponent 要素）	[デプロイメント	Web アプリケーション	*applicationname*	コンフィグレーション	ファイル	インデックス ディレクトリ]
weblogic.httpd.keepAlive.enable	config.xml: KeepAliveEnabled	[サーバ	*servername*	コンフィグレーション	HTTP	Keep Alive を有効化]
weblogic.httpd.logFileBufferKBytes	config.xml: LogFileBufferKBytes（WebServer 要素）	
weblogic.httpd.logFileFlushSecs	config.xml: LogFileFlushSecs（WebServer 要素）	
weblogic.httpd.logFileFormat	config.xml: LogFileFormat（WebServer 要素）	[サービス	仮想ホスト	ログファイル フォーマット]
weblogic.httpd.logFileName	config.xml: LogFileName	[サービス	仮想ホスト	ログファイル名]
weblogic.httpd.logRotationPeriodMins	config.xml: LogRotationTimeBegin（WebServer 要素）	

weblogic.properties ファイルのプロパティ	**.xml コンフィグレーション属性**	**Console でのアクセス**
weblogic.httpd.logRotationPeriodMins	config.xml: LogRotationPeriodMins (WebServer 要素)	
weblogic.httpd.logRotationType	config.xml: LogRotationType	[サーバ	*servername:*	ログ	HTTP	ローテーション タイプ]
weblogic.httpd.maxLogFileSizeKBytes	config.xml: MaxLogFileSizeKBytes	[サーバ	*servername*	ログ	HTTP	最大ログ ファイル サイズ]
weblogic.httpd.mimeType	web.xml: mime-type <mime-mapping> 要素	
weblogic.httpd.postTimeoutSecs	config.xml: PostTimeoutSecs	[サーバ	*servername*	コンフィグレーション	HTTP	POST タイムアウト秒]
weblogic.httpd.servlet.extensionCaseSensitive	config.xml: ServletExtensionCaseSensitive	[デプロイメント	Web アプリケーション	*applicationname*	コンフィグレーション	ファイル	大文字 / 小文字を区別する]
weblogic.httpd.servlet.reloadCheckSecs	config.xml: ServletReloadCheckSecs	[デプロイメント	Web アプリケーション	*applicationname*	コンフィグレーション	ファイル	再ロード間隔 (秒)]

weblogic.properties ファイルのプロパティ	**.xml コンフィグレーション属性**	**Console でのアクセス**
weblogic.httpd.servlet.SingleThreadedModelPoolSize	config.xml: SingleThreadedServletPoolSize (WebAppComponent 要素)	[デプロイメント | Web アプリケーション | *applicationname* | コンフィグレーション | ファイル | シングル スレッド サーブレット プール サイズ]
weblogic.httpd.session.cacheEntries	weblogic.xml: CacheSize <param-name> 要素と <param-value> 要素の組み合わせ	[サーバ | *servername* | コンフィグレーション | SSL | 認可キャッシュ サイズ]
weblogic.httpd.session.cookie.comment	weblogic.xml: CookieComment <param-name> 要素と <param-value> 要素の組み合わせ	
weblogic.httpd.session.cookie.domain	weblogic.xml: CookieDomain <param-name> 要素と <param-value> 要素の組み合わせ	
weblogic.httpd.session.cookie.maxAgeSecs	weblogic.xml: CookieMaxAgeSecs <param-name> 要素と <param-value> 要素の組み合わせ	
weblogic.httpd.session.cookie.name	weblogic.xml: CookieName <param-name> 要素と <param-value> 要素の組み合わせ	

weblogic.properties ファイルのプロパティ	.xml コンフィグレーション属性	Console でのアクセス
weblogic.httpd.session.cookie.path	weblogic.xml: CookiePath <param-name> 要素と <param-value> 要素の組み合わせ	
weblogic.httpd.session.cookies.enable	weblogic.xml: CookiesEnabled <param-name> 要素と <param-value> 要素の組み合わせ	
weblogic.httpd.session.debug	weblogic.xml: SessionDebuggable<param-name> 要素と <param-value> 要素の組み合わせ	
weblogic.httpd.session.enable	weblogic.xml: TrackingEnabled <param-name> 要素と <param-value> 要素の組み合わせ	
weblogic.httpd.session.invalidationintervalSecs	weblogic.xml: InvalidationIntervalSecs <param-name> 要素と <param-value> 要素の組み合わせ	
weblogic.httpd.session.jdbc.connTimeoutSecs	weblogic.xml: JDBCConnectionTimeoutSecs <param-name> 要素と <param-value> 要素の組み合わせ	

weblogic.properties ファイルのプロパティ	**.xml コンフィグレーション属性**	**Console でのアクセス**
weblogic.httpd.session.persistentStoreDir	weblogic.xml: PersistentStoreDir <param-name> 要素と <param-value> 要素の組み合わせ	
weblogic.httpd.session.persistentStorePool	weblogic.xml: PersistentStorePool <param-name> 要素と <param-value> 要素の組み合わせ	
weblogic.httpd.session.persistentStoreShared	weblogic.xml: SessionPersistentStoreShared <param-name> 要素と <param-value> 要素の組み合わせ	
weblogic.httpd.session.persistentStoreType	weblogic.xml: PersistentStoreType <param-name> 要素と <param-value> 要素の組み合わせ	
weblogic.httpd.session.sessionIDLength	weblogic.xml: IDLength <param-name> 要素と <param-value> 要素の組み合わせ	
weblogic.httpd.session.swapintervalSecs	weblogic.xml: SwapIntervalSecs <param-name> 要素と <param-value> 要素の組み合わせ	

weblogic.properties ファイルのプロパティ	**.xml コンフィグレーション属性**	**Console でのアクセス**
weblogic.httpd.session.timeoutSecs	weblogic.xml: TimeoutSecs <param-name> 要素と <param-value> 要素の組み合わせ	[サーバ	*servername*	コンフィグレーション	HTTP	POST タイムアウト 秒]
weblogic.httpd.session.URLRewriting.enable	weblogic.xml: URLRewritingEnabled <param-name> 要素と <param-value> 要素の組み合わせ	
weblogic.httpd.tunneling.clientPingSecs	config.xml: TunnelingClientPingSecs	[サーバ	*servername*	コンフィグレーション	チューニング	トンネリング クライアント Ping]
weblogic.httpd.tunneling.clientTimeoutSecs	config.xml: TunnelingClientTimeoutSecs （Server 要素）	[サーバ	*servername*	コンフィグレーション	チューニング	トンネリング クライアント タイムアウト]
weblogic.httpd.tunnelingenabled	config.xml TunnelingEnabled （Server 要素）	[サーバ	*servername*	コンフィグレーション	チューニング	トンネリングを有効化]
weblogic.httpd.URLResource	config.xml: URLResource （WebServer 要素）	
weblogic.iiop.password	config.xml: DefaultIIOPPassword （Server 要素）	[サーバ	*servername*	コンフィグレーション	プロトコル	デフォルト IIOP パスワード]

weblogic.properties ファイルのプロパティ	**.xml コンフィグレーション属性**	**Console でのアクセス**
weblogic.iiop.user	config.xml: DefaultIIOPUser (Server 要素)	[サーバ	*servername*	コンフィグレーション	プロトコル	デフォルト IIOP ユーザ]
weblogic.jdbc.connectionPool ■ url=JDBC ドライバの URL ■ driver=JDBC ドライバの完全パッケージ名 ■ loginDelaySecs= 接続間の秒数 ■ initialCapacity=*JDBC* 接続の初期数 ■ maxCapacity=JDBC 接続の最大数 ■ capacityIncrement= インクリメント間隔 ■ allowShrinking=true に設定すると縮小が許可される ■ shrinkPeriodMins= 縮小前の間隔 ■ testTable= 自動リフレッシュ テスト用のテーブルの名前 ■ refreshTestMinutes= 自動リフレッシュ テストの間隔 ■ testConnsOnReserve=true に設定すると予約時に接続をテストする ■ testConnsOnRelease=true に設定すると解放時に接続をテストする ■ props=JDBC 接続のプロパティ	config.xml JDBCConnectionPool 要素 ■ ConnLeakProfilingEnabled ■ ACLName ■ URL ■ DriverName ■ Properties ■ LoginDelaySeconds ■ InitialCapacity ■ MaxCapacity ■ CapacityIncrement ■ CapacityEnabled ■ ShrinkPeriodMinutes ■ RefreshMinutes ■ TestTableName ■ TestConnectionsOnRelease ■ SupportsLocalTransaction ■ KeepLogicalConnOpenOnRelease ■ Password	

weblogic.properties ファイルのプロパティ	**.xml コンフィグレーション属性**	**Console でのアクセス**
weblogic.jdbc.enableLogFile	config.xml: JDBCLoggingEnabled (Server 要素)	
weblogic.jdbc.logFileName	config.xml: JDBCLogFileName (Server 要素)	
weblogic.jms.ConnectionConsumer	config.xml JMSConnectionConsumer 要素 ■ MessagesMaximum ■ Selector ■ Destination	
weblogic.jms.connectionFactoryArgs.<<factoryName>> ■ ClientID ■ DeliveryMode ■ TransactionTimeout	config.xml: JMSConnectionFactory 要素 ■ ClientID ■ DefaultDeliveryMode ■ TransactionTimeout ■ UserTransactionsEnabled ■ AllowCloseInOnMessage	
weblogic.jms.connectionFactoryName	config.xml: JMSConnectionFactory 要素 ■ JNDIName	
weblogic.jms.connectionPool	ConnectionPool (JMSJDBCStore 要素)	
weblogic.jms.queue	config.xml: JNDIName StoreEnabled (JMSDestination 要素)	

weblogic.properties ファイルのプロパティ	**.xml コンフィグレーション属性**	**Console でのアクセス**
weblogic.jms.queueSessionPool	config.xml: ConnectionConsumer ConnectionFactory ListenerClass AcknowledgeMode SessionsMaximum Transacted （JMSSessionPool 要素）	
weblogic.jms.tableNamePrefix	config.xml: PrefixName	
weblogic.jms.topic	config.xml JNDIName StoreEnabled （JMSDestination 要素）	[サービス ｜ JMS ｜接続ファクトリ ｜ JNDI 名]
weblogic.jms.topicSessionPool	config.xml: ConnectionConsumer ConnectionFactory ListenerClass AcknowledgeMode SessionsMaximum Transacted （JMSSessionPool 要素）	
weblogic.jndi.transportableObjectFactories	config.xml: JNDITransportableObjectFactoryList （Server 要素）	[サーバ ｜ *servername:*]
weblogic.login.readTimeoutMillisSSL	config.xml LoginTimeoutMillis (SSL 要素)	[サーバ ｜ *servername:*]
weblogic.security.audit.provider	config.xml AuditProviderClassName (Security 要素)	[セキュリティ ｜一般｜監査プロバイダ クラス]

weblogic.properties ファイルのプロパティ	**.xml コンフィグレーション属性**	**Console でのアクセス**
weblogic.security.certificate.authority	config.xml ServerCertificateChainFileName (SSL 要素)	[サーバ	*servername*	コンフィグレーション	SSL	サーバ認証チェーン ファイル]
weblogic.security.certificate.server	config.xml: ServerCertificateFileName (SSL 要素)	[サーバ	*servername*	コンフィグレーション	SSL	サーバ認証ファイル名]
weblogic.security.certificateCacheSize	config.xml: CertificateCacheSize (SSL 要素)	[サーバ	*servername*	コンフィグレーション	SSL	認可キャッシュ サイズ]
weblogic.security.clientRootCA	config.xml: TrustedCAFileName (SSL 要素)	[サーバ	*servername*	コンフィグレーション	SSL	信頼性のある CA ファイル名]
weblogic.security.disableGuest	config.xml: GuestDisabled (Security 要素)	[セキュリティ	一般	ゲスト不可]
weblogic.security.enforceClientCert	config.xml: ClientCertificateEnforced (SSL 要素)	[サーバ	*servername*	コンフィグレーション	SSL	クライアント認証を強制する]
weblogic.security.key.export.lifespan	config.xml: ExportKeyLifespan (SSL 要素)	[サーバ	*servername*	コンフィグレーション	SSL	キーの有効期間をエクスポート]
weblogic.security.key.server	config.xml: ServerKeyFileName (SSL 要素)	[サーバ	*servername*	コンフィグレーション	SSL	サーバ キー ファイル名]

weblogic.properties ファイルのプロパティ	**.xml コンフィグレーション属性**	**Console でのアクセス**
weblogic.security.ldaprealm.authentication	config.xml: AuthProtocol（LDAPRealm 要素）	
weblogic.security.ldaprealm.credential	config.xml: Credential（LDAPRealm 要素）	
weblogic.security.ldaprealm.factory	config.xml LdapProvider（LDAPRealm 要素）	
weblogic.security.ldaprealm.groupDN	config.xml: GroupDN（LDAPRealm 要素）	
weblogic.security.ldaprealm.groupIsContext	config.xml: GroupIsContext（LDAPRealm 要素）	
weblogic.security.ldaprealm.groupNameAttribute	config.xml: GroupNameAttribute（LDAPRealm 要素）	
weblogic.security.ldaprealm.groupUsernameAttribute	config.xml: GroupUsernameAttribute（LDAPRealm 要素）	
weblogic.security.ldaprealm.principal	config.xml: Principal（LDAPRealm 要素）	
weblogic.security.ldaprealm.ssl	config.xml: SSLEnable（LDAPRealm 要素）	
weblogic.security.ldaprealm.url	config.xml: LDAPURL（LDAPRealm 要素）	

weblogic.properties ファイルのプロパティ	.xml コンフィグレーション属性	Console でのアクセス
weblogic.security.ldaprealm.userAuthentication	config.xml: UserAuthentication（LDAPRealm 要素）	
weblogic.security.ldaprealm.userDN	config.xml: UserDN（LDAPRealm 要素）	
weblogic.security.ldaprealm.userNameAttribute	config.xml: UserNameAttribute（LDAPRealm 要素）	
weblogic.security.ldaprealm.userPasswordAttribute	config.xml: UserPasswordAttribute（LDAPRealm 要素）	
weblogic.security.net.connectionFilter	config.xml: ConnectionFilter（Security 要素）	
weblogic.security.ntrealm.domain	config.xml: PrimaryDomain（NTRealm 要素）	
weblogic.security.realm.cache.acl.enable	config.xml: ACLCacheEnable（CachingRealm 要素）	
weblogic.security.realm.cache.acl.size	config.xml: ACLCacheSize（CachingRealm 要素）	
weblogic.security.realm.cache.acl.ttl.negative	config.xml: ACLCacheTTLNegative（CachingRealm 要素）	
weblogic.security.realm.cache.acl.ttl.positive	config.xml: ACLCacheTTLPositive（CachingRealm 要素）	

weblogic.properties ファイルのプロパティ	.xml コンフィグレーション属性	Console でのアクセス
weblogic.security.realm.cache.auth.enable	config.xml: AuthenticationCacheEnable (CachingRealm 要素)	
weblogic.security.realm.cache.auth.size	config.xml: AuthenticationCacheSize (CachingRealm 要素)	
weblogic.security.realm.cache.auth.ttl.negative	config.xml: AuthenticationCacheTTLNegative (CachingRealm 要素)	
weblogic.security.realm.cache.auth.ttl.positive	config.xml: AuthenticationCacheTTLPositive (CachingRealm 要素)	
weblogic.security.realm.cache.caseSensitive	config.xml: CacheCaseSensitive (CachingRealm 要素)	
weblogic.security.realm.cache.group.enable	config.xml: GroupCacheEnable (CachingRealm 要素)	
weblogic.security.realm.cache.group.size	config.xml: GroupCacheSize (CachingRealm 要素)	
weblogic.security.realm.cache.group.ttl.negative	config.xml: GroupCacheTTLNegative (CachingRealm 要素)	
weblogic.security.realm.cache.group.ttl.positive	config.xml: GroupCacheTTLPositive (CachingRealm 要素)	

weblogic.properties ファイルのプロパティ	**.xml コンフィグレーション属性**	**Console でのアクセス**
weblogic.security.realm.cache.perm.enable	config.xml: PermissionCacheEnable (CachingRealm 要素)	
weblogic.security.realm.cache.perm.size	config.xml: PermissionCacheSize (CachingRealm 要素)	
weblogic.security.realm.cache.perm.ttl.negative	config.xml: PermissionCacheTTLNegative (CachingRealm 要素)	
weblogic.security.realm.cache.perm.ttl.positive	config.xml: PermissionCacheTTLPositive (CachingRealm 要素)	
weblogic.security.realm.cache.user.enable	config.xml: UserCacheEnable (CachingRealm 要素)	
weblogic.security.realm.cache.user.size	config.xml: UserCacheSize (CachingRealm 要素)	
weblogic.security.realm.cache.user.ttl.negative	config.xml: UserCacheTTLNegative (CachingRealm 要素)	
weblogic.security.realm.cache.user.ttl.positive	config.xml: UserCacheTTLPositive (CachingRealm 要素)	
weblogic.security.realm.certAuthenticator	config.xml: CertAuthenticator (SSL 要素)	[サーバ | *servername* | コンフィグレーション | SSL | 認可済み認証機関]

weblogic.properties ファイルのプロパティ	**.xml コンフィグレーション属性**	**Console でのアクセス**
weblogic.security.SSL.ciphersuite	config.xml Ciphersuites（SSL 要素）	
weblogic.security.ssl.enable	config.xml: Enabled（SSL 要素）	[サーバ	*servername*	コンフィグレーション	SSL	有効化]
weblogic.security.SSL.hostnameVerifier	config.xml HostnameVerifier（SSL 要素）	[サーバ	*servername*	コンフィグレーション	SSL	ホスト名の検証]
weblogic.security.SSL.ignoreHostnameVerification	config.xml HostNameVerification Ignored（SSL 要素）	
weblogic.security.SSLHandler.enable	config.xml: HandlerEnabled（SSL 要素）	[サーバ	*servername*	コンフィグレーション	SSL	ハンドラを有効化]
weblogic.security.unixrealm.authProgram	config.xml: AuthProgram（UnixRealm 要素）	
weblogic.system.AdministrationPort	config.xml AdministrationPort（Server 要素）	[サーバ	*servername*	コンフィグレーション	一般	管理ポート]
weblogic.system.AdministrationPort	config.xml: AdministrationPort（Server 要素）	[サーバ	*servername*	コンフィグレーション	一般	管理ポート]
weblogic.system.bindAddr	config.xml: ListenAddress（Server 要素）	

weblogic.properties ファイルのプロパティ	**.xml コンフィグレーション属性**	**Console でのアクセス**
weblogic.system.defaultProtocol	config.xml: DefaultProtocol (Server 要素)	[サーバ | *servername* | コンフィグレーション | プロトコル | デフォルト プロトコル]
weblogic.system.defaultSecureProtocol	config.xml: DefaultSecureProtocol (Server 要素)	[サーバ | *servername* | コンフィグレーション | プロトコル | デフォルト セキュア プロトコル]
weblogic.system.enableConsole	config.xml: StdoutEnabled (Kernel 要素)	[サーバ | *servername* | ログ | 一般 | Stdout へログ出力]
weblogic.system.enableIIOP	config.xml: IIOPEnabled (Server 要素)	
weblogic.system.enableReverseDNSLookups	config.xml: ReverseDNSAllowed (Server 要素)	
weblogic.system.enableSetGID,	config.xml: PostBindGID	
weblogic.system.enableSetUID,	config.xml: PostBindUIDEnabled	
weblogic.system.enableTGIOP	config.xml TGIOPEnabled (Server 要素)	[サーバ | *servername:*]
weblogic.system.helpPageURL	config.xml HelpPageURL (Server 要素)	[サーバ | *servername:*]
weblogic.system.home	config.xml: RootDirectory (Server 要素)	

weblogic.properties ファイルのプロパティ	**.xml コンフィグレーション属性**	**Console でのアクセス**
weblogic.system.ListenPort	config.xml ListenPort (Server 要素)	[サーバ	*servername*	コンフィグレーション	SSL	リスン ポート]
weblogic.system.logFile	config.xml: FileName (Log 要素)	
weblogic.system.MagicThreadBackToSocket	config.xml: MagicThreadDumpBackToSocket (ServerDebug 要素)	
weblogic.system.MagicThreadDumpFile	config.xml: MagicThreadDumpFile (ServerDebug 要素)	
weblogic.system.MagicThreadDumpHost	config.xml: MagicThreadDumpHost (ServerDebug 要素)	
weblogic.system.magicThreadDumps	config.xml: MagicThreadDumpEnabled (ServerDebug 要素)	
weblogic.system.maxLogFileSize	config.xml: FileMinxSize (Log 要素)	
weblogic.system.nativeIO.enable	config.xml: NativeIOEnabled (Server 要素)	[サーバ	*servername*	コンフィグレーション	チューニング	ネイティブ IO を有効化]
weblogic.system.nonPrivGroup	config.xml PostBindGID (UnixMachine 要素)	

weblogic.properties ファイルのプロパティ	**.xml コンフィグレーション属性**	**Console でのアクセス**
weblogic.system.nonPrivUser	config.xml PostBindUID (UnixMachine 要素)	
weblogic.system.percentSocketReaders	config.xml: ThreadPoolPercentSocketReaders (Kernel 要素)	[サーバ	*servername*	コンフィグレーション	チューニング	ソケット リーダー]
weblogic.system.readTimeoutMillis	config.xml LoginTimeoutMillis (Server 要素)	[サーバ	*servername:*]
weblogic.system.SSL.useJava	config.xml: UseJava (SSL 要素)	[サーバ	*servername*	コンフィグレーション	SSL	Java を使用]
weblogic.system.SSLListenPort	config.xml: ListenPort (SSL 要素)	[サーバ	*servername*	コンフィグレーション	SSL	リスン ポート]
weblogic.system.startupFailureIsFatal	config.xml FailureIsFatal (StartupClass 要素)	
weblogic.system.user	config.xml: SystemUser (Security 要素)	
weblogic.system.weight	config.xml ClusterWeight (Server 要素)	[サーバ	*servername*	コンフィグレーション	クラスタ	クラスタの重み]
weblogic.workspace.showUserKeysOnly	config.xml: WorkspaceShowUserKeysOnly (Server 要素)	[サーバ	*servername*	コンフィグレーション	チューニング	ワークスペース ユーザのキーのみを表示]

weblogic.properties ファイルのプロパティ	.xml コンフィグレーション属性	Console でのアクセス
weblogic.zac.enable	config.xml: ZACEnabled (Server 要素)	[サーバ	*servername:*]
weblogic.zac.publishRootProp	config.xml: ZACPublishRoot (Server 要素)	[サーバ	*servername:*]

Web アプリケーションの移行

アプリケーションを WebLogic Server 6.0 または 6.1 上にデプロイされる Web アプリケーションに移行するには、アプリケーションのファイルを特定のパターンに従うディレクトリ構造に配置する必要があります。開発段階では、ディレクトリ フォーマットに注意する必要はありません。ただし、プロダクション段階では、アプリケーションを .war ファイルに 1 つの Web アプリケーションとしてまとめることをお勧めします。Web アプリケーションの詳細については、「WebLogic Server J2EE アプリケーションについて」と『Web アプリケーションのアセンブルとコンフィグレーション』を参照してください。

以下の節では、WebLogic Server 5.1 から WebLogic Server 6.1 への単純なサーブレットの移行手順など、Web アプリケーションの移行とデプロイについて知っておくべき情報を提供します。

- 3-27 ページの「Web アプリケーションのディレクトリ構造」
- 3-28 ページの「XML デプロイメント記述子」
- 3-28 ページの「WAR ファイル」
- 3-29 ページの「Web アプリケーションのデプロイメント」
- 3-30 ページの「セッションの移行」
- 3-30 ページの「JavaServer Pages（JSP）とサーブレット」
- 3-32 ページの「WebLogic Server 5.1 から WebLogic Server 6.1 への単純なサーブレットの移行」

Web アプリケーションのディレクトリ構造

Web アプリケーションは、WebLogic Server で保管およびデプロイできるように、指定されたディレクトリ構造に整理されます。Web アプリケーションに属するすべてのサーブレット、クラス、静的ファイル、およびその他のリソースは、ディレクトリ階層に基づいて配置されます。この階層構造のルートにより、Web アプリケーションのドキュメント ルートが定義されます。このルート ディレクトリの下に置かれたファイルは、`WEB-INF` および `META-INF` という特別なディレクトリの下にあるファイルを除き、すべてクライアントに提供されます。ルート ディレクトリは、Web アプリケーションと同じ名前にして、`wlserver6.1\config\`*`domainName`*`\applications` ディレクトリに配置する必要があります。

次の図は、Web アプリケーションのディレクトリ構造を示しています。

```
WebApplicationRoot\ (.jsp、.html、.jpg、.gif などの
                    |   公開されるファイル)
                    |
                    +WEB-INF\-+
                              |
                              + classes\ (Web アプリケーションで使用
                              |          されるサーブレットなどの
                              |          Java クラスを格納する
                              |          ディレクトリ)
                              |
                              + lib\ (Web アプリケーションで使用
                              |          される jar ファイルを格納する
                              |          ディレクトリ)
                              |
                              + web.xml
                              |
                              + weblogic.xml
```

`weblogic.properties` ファイルを変換すると、適切な `web.xml` ファイルおよび `weblogic.xml` ファイルが `wlserver6.1\config\`*`domainName`*`\applications\DefaultWebApp_myserver\WEB-INF` ディレクトリに自動的に作成されます。上記のディレクトリ構造に従って、`.xml` ファイルを `wlserver6.1\config\`*`domainName`*`\applications\`*`webAppName`*`\WEB-INF` ディレクトリに配置します。すべてのアプリケーションは、`wlserver6.1\config\`*`domainName`*`\applications` ディレクトリに配置する必要があります。このディレクトリに配置しないとデプロイできません。詳細については、『WebLogic Server アプリケーションの開発』を参照してください。

XML デプロイメント記述子

Web アプリケーションのデプロイメント記述子（web.xml）は標準の J2EE 記述子であり、サーブレットの登録、サーブレット初期化パラメータの定義、JSP タグ ライブラリの登録、セキュリティ制約の定義などに使用します。デプロイメント記述子を作成する詳しい手順については、「Web アプリケーションのデプロイメント記述子の記述」を参照してください。

WebLogic 固有のデプロイメント記述子（weblogic.xml）もあります。このファイルでは、JSP プロパティ、JNDI のマッピング、セキュリティ ロールのマッピング、および HTTP セッション パラメータを定義します。WebLogic 固有のデプロイメント記述子では、web.xml ファイルで指定されたリソースが WebLogic Server のどこかにあるリソースにどのようにマップされるのかも定義します。WebLogic 固有のデプロイメント記述子を作成する詳しい手順については、「WebLogic 固有のデプロイメント記述子の記述」を参照してください。前述のプロパティ、マッピング、またはパラメータが必要ない場合は、このファイルは必ずしも必要ではありません。

web.xml ファイルと weblogic.xml ファイルは、コンソールと一緒にアプリケーションをコンフィグレーションするために使用します。.xml ファイルは、テキスト エディタで表示できます。.xml ファイルを編集するには、必要な変更を行い、規定のディレクトリ構造で指定されている適切なパスを指定して web.xml または weblogic.xml として保存します。詳細については、『Web アプリケーションのアセンブルとコンフィグレーション』を参照してください。アプリケーションを 1 つの Web アプリケーションとしてデプロイしない場合は、自動的に作成された .xml ファイルを分割し、Web アプリケーションごとに適切な .xml ファイルを作成する必要があります。

WAR ファイル

.war ファイルは、Web アプリケーションのアーカイブです。Web アプリケーションの規定のディレクトリ構造に正確に従い、適切な web.xml ファイルと weblogic.xml ファイルを作成したら、プロダクション環境では、アプリケーションを .war ファイルとしてデプロイされる Web アプリケーションにまとめることをお勧めします。アプリケーションを .war ファイルにまとめた後は、WebLogic Server で各アプリケーションのインスタンスが 1 つだけになるように以前のディレクトリ構造を削除することが重要です。

Web アプリケーションのあるルート ディレクトリで次のコマンドを使用すると、.war ファイルを作成できます。「webAppName」は、Web アプリケーションに選択した特定の名前に置き換えます。

```
jar cvf webAppName.war *
```

この時点で、Web アプリケーションのすべてのファイルとコンフィグレーション情報を格納する .war ファイルが作成されています。

Web アプリケーションのデプロイメント

まとめた Web アプリケーションを適切にデプロイするには、適切な .war ファイルを c:\wlserver6.1\config*domainName*\applications ディレクトリに配置します。Administration Console を使用してアプリケーションをインストールすることもできます。そのためには、コンソールのホームに移動して、[はじめに] メニューの下にある [アプリケーションのインストール] をクリックします。正しい .war ファイルを選択すると、ファイルが自動的にインストールされます。アプリケーションを機能させるためには、アプリケーションが c:\wlserver6.1\config*domainName*\applications ディレクトリに配置されている必要があります。

Web アプリケーションは、インストールされた後に自動的にデプロイされます。Administration Console の左ペインの [デプロイメント] ノードの下に Web アプリケーションがデプロイされていることを確認してください。

Web アプリケーションの特定のデプロイメント属性は、Administration Console を使用してコンフィグレーションできます。[デプロイメント] という見出しの下の [Web アプリケーション] ノードを選択します。Web アプリケーションを選択します。コンフィグレーションする適切なタブをクリックします。Administration Console での属性設定の詳細については、Administration Console ヘルプの Web アプリケーションの節を参照してください。

セッションの移行

WebLogic Server 6.0 以降のバージョンでは、以前のバージョンからのクッキーが認識されません。バージョン 6.0 でクッキーのフォーマットが変更されたからです。WebLogic Server では、旧式のフォーマットのクッキーは無視され、新しいセッションが作成されます。

クッキーのデフォルト名はバージョン 5.1 から変更されています（以前の名前は `WebLogicSession`）。WebLogic 6.0 以降は、クッキーのデフォルト名は `JSESSIONID` です。

詳細については、`http://edocs.beasys.co.jp/e-docs/wls61/webapp/weblogic_xml.html` にある「web.xml デプロイメント記述子の要素」を参照してください。

JavaServer Pages（JSP）とサーブレット

この節では、アプリケーションに関連付けられる場合のある JSP およびサーブレットに特有の情報を提供します。

- Java および HTML の両方で、コードの URL を参照する箇所を変更する必要があります。デフォルトの Web アプリケーションではない Web アプリケーションのサーブレットと JSP がデプロイされるときに URL が異なる場合があります。詳細については、「Web アプリケーションでのサーブレットの参照」を参照してください。相対 URL が使用され、すべてのコンポーネントが同じ Web アプリケーションに格納される場合、そのような変更は不要です。
- 分散可能なアプリケーションの場合は、シリアライズ可能なオブジェクトだけをセッションに格納できます。
- weblogic.properties を、`web.xml` および `weblogic.xml` の `XML` 属性に変換する必要があります。このプロセスの詳細については、Administration Console ヘルプの変換のセクションを参照してください。
- ACL は `web.xml` でセキュリティ制約として定義されます。
- サーバサイドインクルードはサポートされていません。この機能は JSP を使用して実現する必要があります。

- このバージョンの WebLogic Server はサーブレット 2.3 仕様に完全に準拠しています。

WebLogic Server 5.1 から WebLogic Server 6.1 への単純なサーブレットの移行

次の手順では、WebLogic 5.1 Server で提供されていた単純な Hello World サーブレットを WebLogic Server 6.1 に移行します。

1. 『HTTP サーブレット プログラマーズ ガイド』で説明されているとおりに、適切なディレクトリ構造を作成します。つまり、ルート アプリケーション ディレクトリ（C:\hello など）と、C:\hello\WEB-INF ディレクトリおよび C:\hello\WEB-INF\classes ディレクトリを作成します。HelloWorld.Servlet.java ファイルを C:\hello\WEB-INF\classes ディレクトリに配置します。

2. このサーブレットの web.xml ファイルを作成します。weblogic.properties ファイルが変換されている場合、web.xml ファイルは既に自動的に作成されています。変換前に weblogic.properties ファイルで HelloWorldServlet が登録されている場合、サーブレットは新しい web.xml ファイルで適切にコンフィグレーションされます。.xml ファイルは、どのテキスト エディタでも作成できます。HelloWorldServlet で使用できる基本的な web.xml ファイルの例は次のとおりです。

```
<!DOCTYPE web-app (View Source for full doctype...)>
- <web-app>
- <servlet>
<servlet-name>HelloWorldServlet</servlet-name>
<servlet-class>examples.servlets.HelloWorldServlet</servlet-cla
ss>
</servlet>
- <servlet-mapping>
<servlet-name>HelloWorldServlet</servlet-name>
<url-pattern>/hello/*</url-pattern>
</servlet-mapping>
</web-app>
```

   web.xml ファイルの詳細については、「Web アプリケーションのデプロイメント記述子の記述」を参照してください。weblogic.xml ファイルは、HelloWorld のようなスタンドアロンの単純なサーブレットでは必要ありません。

   weblogic.xml ファイルの詳細については、「WebLogic 固有のデプロイメント記述子の記述」を参照してください。

3. web.xml ファイルを wlserver6.1\config*domainName*\applications\DefaultWebApp_myserver\WEB-INF から C:\hello\WEB-INF\ に移動します。
4. 開発環境をセットアップし（「開発環境の構築」を参照）、次のようなコマンドで HelloWorldServlet をコンパイルします。

   ```
   C:\hello\WEB-INF\classes>javac -d  . HelloWorldServlet.java
   ```

   このコマンドで、ファイルがコンパイルされ、適切なパッケージ構造が作成されます。
5. この時点で、サーブレットは次のコマンドを使用してアーカイブの .war ファイルにまとめることができます。

   ```
   jar cvf hello.war *
   ```

   このコマンドで、hello.war ファイルが作成され、そのファイルが C:\hello ディレクトリに配置されます。
6. この Web アプリケーションをインストールするには、サーバを起動し、Administration Console を開きます。[はじめに] の下にある [アプリケーションのインストール] を選択します。新しく作成した .war ファイルを参照し、[Upload] をクリックします。

   デプロイされたサーブレットが、コンソールの左ペインにある [デプロイメント] の下の [Web アプリケーション] ノードに表示されます。
7. このサーブレットを呼び出すには、ブラウザの URL ウィンドウに http://localhost:7001/hello/hello のように入力します。

   /hello/ は、サーブレットのコンテキスト パスです。コンテキスト パスは、.war ファイルの名前（この場合は hello.war）で決まります。2 番目の /hello は、web.xml ファイルのサーブレット マッピング タグでマップされています。

エンタープライズ JavaBean アプリケーションの移行と変換

以降の節では、エンタープライズ JavaBean の移行と変換の手順について説明します。

EJB の移行に関する考慮事項

エンタープライズ JavaBean を WebLogic Server 6.x に移行するときには以下の事項を考慮してください。

- WebLogic Server バージョン 6.0 および 6.1 ではエンタープライズ JavaBean 1.1 と 2.0 の仕様がサポートされています。
- WebLogic 6.0 および 6.1 では、WebLogic 5.1 よりも XML パーサが XML デプロイメント記述子に対してより厳密になっています。以前のバージョンでは容認された一部のエラーが現在は許されなくなっています。詳細については、『WebLogic エンタープライズ JavaBeans プログラマーズ ガイド』を参照してください。
- EJB 1.1 Bean は WebLogic Server 6.0 および 6.1 でデプロイできます。ただし、新しく Bean を開発する場合は、EJB 2.0 を使用することをお勧めします。EJB 1.1 Bean は、DDConverter ユーティリティを使用して 2.0 に変換できます。詳細については、「DDConverter」を参照してください。
- EJB 1.0 デプロイメント記述子は DDConverter ユーティリティを使用して EJB 2.0 にアップグレードできますが、その前にそれらの記述子を 1.1 にアップグレードする必要があります。WebLogic Server 5.1 のデプロイメント記述子を 6.0 または 6.1 にアップグレードすると、WebLogic Server 6.0 または 6.1 の新機能を利用できます。DDConverter ユーティリティの詳細については、『WebLogic エンタープライズ JavaBeans プログラマーズ ガイド』を参照してください。
- EJB 1.1 のファインダ式機能は現在はサポートされていません。EJB 1.1 の機能でサポートされていないのはこの機能だけです。
- Bean のデプロイの詳細については、『WebLogic エンタープライズ JavaBeans プログラマーズ ガイド』を参照してください。
- `ejbc` が実行されていない EJB 場合、WebLogic Server 6.1 では Bean がデプロイされるときに `ejbc` が自動的に実行されます。デプロイメントの前に `ejbc` で Bean をコンパイルする必要はありません。必要な場合は、起動時に `ejbc` を実行することもできます。詳細については、『WebLogic エンタープライズ JavaBeans プログラマーズ ガイド』を参照してください。

- EJB のデプロイメントでは、`ejb-jar.xml` ファイルに標準のデプロイメント記述子が含まれます。`ejb-jar.xml` は、EJB 1.1 DTD（文書型定義）または EJB 2.0 DTD に準拠している必要があります。
- EJB のデプロイメントでは、`weblogic-ejb-jar.xml` ファイルが必要です。このファイルは、WebLogic Server EJB コンテナのコンフィグレーション情報が含まれる WebLogic Server 固有のデプロイメント記述子です。このファイルは、WebLogic Server 5.1 DTD または WebLogic Server 6.0 DTD に準拠している必要があります。
- データベースに対するマッピングを指定するために、コンテナ管理の永続性エンティティ Bean では WebLogic Server 5.1 CMP DTD、WebLogic Server 6.0 EJB 1.1 DTD、または WebLogic Server 6.0 EJB 2.0 DTD に準拠した CMP デプロイメント記述子が必要です。

EJB の移行に関する推奨事項

- `TxDataSource` を使用する。

  EJB では常に、`TxDataSource` からデータベース接続を取得する必要があります。そうすることで、EJB コンテナのトランザクション管理は JDBC 接続と連携でき、XA トランザクションをサポートすることもできます。

  WebLogic Server 6.x CMP デプロイメント記述子は `TxDataSource` をサポートするので、この記述子を接続プールのみを指定する WebLogic Server 5.1 CMP デプロイメント記述子の代わりに使用する必要があります。

- `ejbc` とともに高速なコンパイラを使用する。

  WebLogic Server EJB コンパイラ（`weblogic.ejbc`）では、java コンパイラでコンパイルされる java コードが生成されます。デフォルトでは、WebLogic Server では同梱の JDK に含まれている `javac` コンパイラが使用されます。EJB コンパイラは、より高速な java コンパイラを使用すると非常に高速で動作します。`-compiler` オプションを使用すると、代わりのコンパイラを次のようにして指定できます。

  ```
  java weblogic.ejbc -compiler sj pre_AccountEJB.jar
  AccountEJB.jar
  ```

- EJB を WeblLogic Server 6.1 にデプロイする前にエラーを修正する。

  WebLogic Server 6.1 EJB コンパイラ（ejbc）には、以前のリリースの WebLogic Server にはなかった検証機能が追加されています。このため、以前のバージョンの WebLogic Server でエラーなくデプロイされた EJB でも、WebLogic Server 6.1 ではエラーが報告される可能性があります。それらのエラーは、EJB を WebLogic Server 6.1 にデプロイする前に修正する必要があります。

  たとえば、WebLogic Server 6.1 はトランザクション属性があるメソッド名に対して設定されている場合はそのメソッドが必ず存在するようにします。このことは、存在しないメソッドでトランザクション属性が誤って設定されているようなよくあるエラーを特定するのに役立ちます。

- WebLogic Server 6.1 のサンプルを参照する。

  WebLogic Server 配布キットには、デプロイメント記述子のあるいくつかの EJB サンプルが含まれています。それらの EJB サンプルは、WebLogic

Server 6.1 配布キットの `samples\examples\ejb` ディレクトリおよび `samples\examples\ejb20` ディレクトリにあります。

次の表は、WebLogic Server 6.1 でサポートされている記述子の組み合わせを示しています。

EJB のバージョン	WebLogic Server のバージョン	CMP のバージョン
既存の WebLogic Server 5.1 デプロイメントは、以下の組み合わせを使用し、記述子またはコードを変更することなく WebLogic Server 6.1 でデプロイできる。		
1.1	5.1	5.1
以下の組み合わせでは、新しい WebLogic Server 6.x の機能を `weblogic-ejb-jar.xml` ファイルで指定できるように WebLogic Server 6.0 デプロイメント記述子が使用される。		
1.1	6.x	5.1
以下の組み合わせでは、WebLogic Server 6.0 CMP デプロイメント記述子を利用する。WebLogic Server 6.x EJB 1.1 CMP デプロイメント記述子を利用すると、複数の EJB を 1 つの EJB JAR ファイルで指定でき、EJB が 2 フェーズ トランザクションまたは XA トランザクションに関与する場合に必要な `TxDataSource` を使用できる。		
1.1	5.1	6.x
1.1	6.x	6.x
EJB 2.0 Bean では常に WebLogic Server 6.x デプロイメント記述子を使用する。		
2.0	6.x	6.x

エンタープライズ JavaBean の詳細については、「エンタープライズ JavaBeans コンポーネント」および『WebLogic エンタープライズ JavaBeans プログラマーズ ガイド』を参照してください。

1.0 EJBean を WebLogic Server 4.5.x から WebLogic Server 6.1 に移行する手順

WebLogic Server 3.1.x、4.0.x、および 4.5.x では、EJB 1.0 仕様がサポートされていました。WebLogic Server 4.5 から WebLogic Server 6.1 へ 1.0 EJB を移行するには、次の手順を行います。

1. EJB 1.0 デプロイメント記述子を EJB 1.1 または EJB 2.0 XML デプロイメント記述子に変換します。この変換は、DDCreator ツールで自動的に行えます。
2. デプロイメントを JAR ファイルにパッケージ化します。
3. WebLogic Server EJB コンパイラ（ejbc）を実行して JAR ファイルをコンパイルします。`ejbc` ツールを使用すると、EJB がコンパイルされるときに、それが確実に EJB 1.1 仕様または EJB 2.0 仕様に準拠します。
4. EJB コンテナに EJB をデプロイする前に準拠エラーを修正します。

EJB 1.1 または 2.0 に確実に準拠するために、EJB 1.0 Bean で次のような変更を行います。

- EJB 1.0 Bean では、`SessionContext` または `EntityContext` が一時的に参照されていました。EJB 1.1 または 2.0 の Bean をデプロイする場合、参照は一時的にはなりません。次に例を示します。

  ```
  private transient SessionContext ctx;
  ```

  これは次のように修正する必要があります。

  ```
  private SessionContext ctx;
  ```

- EJB 1.0 CMP エンティティ Bean の `ejbCreate` メソッドには void 型の戻り値の型がありました。EJB 1.1 または 2.0 の Bean をデプロイする場合、戻り値の型は Bean 管理の永続性エンティティ Bean を記述し、CMP 実装でそれをサブクラス化することが可能な主キー クラスでなければなりません。次に例を示します。

  ```
  public void ejbCreate (String name) {
  firstName = name;
  }
  ```

  これは次のように修正する必要があります。

  ```
  public AccountPK ejbCreate (String name) {
  firstName = name;
  ```

```
return null; // EJB 仕様で必須
}
```

- EJB 1.1 または 2.0 の場合、エンティティ Bean では Bean 管理のトランザクションを使用できません。代わりに、コンテナ管理のトランザクション属性（Required、Mandatory など）を使用して実行する必要があります。

1.1 EJBean を WebLogic Server 5.1 から WebLogic Server 6.1 に移行する手順

WebLogic Server 5.1 デプロイメント記述子では、排他的または read-only の同時実行性オプションのみを使用できます。データベース同時実行性オプションは、WebLogic Server 6.x の `weblogic-ejb-jar.xml` ファイルにアップグレードすると利用可能になります。このオプションの詳細については、「weblogic-ejb-jar-xml デプロイメント記述子」のデータベース同時実行性に関する情報を参照してください。

WebLogic Server 6.x CMP デプロイメント記述子では、複数の EJB を指定でき、接続プールの代わりに `TxDataSource` を使用できます。EJB 1.1 CMP で XA を使用する場合は、`TxDataSource` を使用する必要があります。

WebLogic Server 5.1 から WebLogic Server 6.1 へ 1.1 EJB を移行するには、次の手順を行います。

1. Administration Console を起動します。ホーム ページの [はじめに] という見出しの下の [アプリケーションのインストール] をクリックします。
2. [参照] ボタンをクリックして移行する JAR ファイルを探し、[開く] および [Upload] の順にクリックします。Bean が自動的にデプロイされます。
3. クライアント ウィンドウで `setEnv` スクリプトを実行し、開発環境を設定します。詳細については、「開発環境の構築」を参照してください。
4. 必要なすべてのクライアント クラスをコンパイルします。たとえば、WebLogic Server 5.1 で提供されたサンプルのステートレス セッション Bean の場合は、次のコマンドを使用します。

   ```
   javac -d %CLIENTCLASSES% Trader.java TraderHome.java
   TradeResult.java Client.java
   ```

5. クライアントを実行するには、次のコマンドを入力します。

```
java examples.ejb.basic.statelessSession.Client
```

EJB 1.1 を EJB 2.0 に変換する手順

EJB 1.1 Bean を EJB 2.0 Bean に変換するには、WebLogic Server DDConverter ユーティリティを使用できます。

WebLogic Server 6.1 では、EJB 2.0 Bean を開発することをお勧めします。ただし、1.1 Bean が既にプロダクション環境で使用されている場合は、それらを 2.0 Bean に変換する必要はありません。EJB 1.1 Bean は、WebLogic Server 6.1 でデプロイできます。1.1 Bean を 2.0 Bean に変換する必要がある場合、その詳細については 「DDConverter」 を参照してください。

単純な CMP 1.1 Bean を 2.0 Bean に変換するために必要な基本手順を以下に示します。

1. Bean クラスを抽象クラスにします。EJB 1.1 Bean を Bean の CMP フィールドで宣言します。CMP 2.0 Bean は、各フィールドに対応する抽象 getXXX メソッドおよび setXXX メソッドを使用します。たとえば、1.1 Bean は `public String name` を使用します。2.0 Bean は、`public abstract String getName()` と `public abstract void setName(String n)` を使用します。この変更を行うと、Bean クラスが、`getName` メソッドでコンテナ管理フィールドを読み込んだり、`setName` メソッドで更新したりできるようになります。
2. `java.util.Enumeration` を使用していたすべての CMP 1.1 ファインダは、`java.util.Collection` を使用する必要があります。これを反映するように、コードを変更してください。CMP 2.0 ファインダは、`java.util.Enumeration` を返すことができません。

他の J2EE アプリケーション サーバからの EJB の移行

EJB 1.1 仕様または EJB 2.0 仕様に準拠したすべての EJB は、WebLogic Server 6.1 EJB コンテナにデプロイできます。各 EJB JAR ファイルでは、`ejb-jar.xml` ファイル、`weblogic-ejb-jar.xml` デプロイメント記述子、および CMP デプロ

イメント記述子（CMP エンティティ Bean を使用する場合）が必要です。WebLogic Server 配布キットの samples\examples\ejb および samples\examples\ejb20 にある WebLogic Server EJB サンプルには、サンプルの weblogic デプロイメント記述子が含まれています。

エンタープライズ アプリケーションの作成

エンタープライズ アプリケーションは、アセンブル済みのコンポーネントで構成された、拡張子が .ear の .jar ファイルです。.ear ファイルには、アプリケーションのすべての .jar および .war コンポーネント アーカイブ ファイルおよびコンポーネントを記述する XML 記述子が格納されます。META-INF\application.xml デプロイメント記述子には、各 Web モジュールおよび EJB モジュールのエントリのほか、セキュリティ ロールやアプリケーション リソース（データベースなど）を記述する追加エントリがあります。

```
EnterpriseApplicationStagingDirectory\
                                      |
                                      + .jar files
                                      |
                                      + .war files
                                      |
                                      +META-INF\-+
                                                 |
                                                 + application.xml
```

.ear ファイルを作成するには、次の手順を行います。

1. アプリケーションのすべての Web アーカイブと EJB アーカイブをアセンブルし、それらをステージング ディレクトリに配置します。
2. まず .war ファイルと EJB .jar ファイルをステージング ディレクトリにコピーし、次にアプリケーションの META-INF\application.xml デプロイメント記述子を作成します。上記のディレクトリ構造に従います。

   application.xml ファイルには、Sun Microsystems の DTD に基づいてアプリケーションの各コンポーネント用の記述子が定義されます。application.xml ファイルの詳細については、「クライアント アプリケーションのデプロイメント記述子の要素」を参照してください。JSP を使用する場合で、JSP を実行時にコンパイルするためには、.war ファイルのクラス

ディレクトリにある Bean のホーム インタフェースとリモート インタフェースが必要です。

3. ステージング ディレクトリで次のような jar コマンドを実行して、エンタープライズ アーカイブを作成します。

```
jar cvf myWebApp.war *
```

4. コンソールのホーム ページの [はじめに] という見出しの下にある [アプリケーションのインストール] リンクをクリックし、.ear ファイルを \wlserver6.1\config\domainName\applications ディレクトリに配置します。エンタープライズ アプリケーションの詳細については、「エンタープライズ アプリケーションのパッケージ化」を参照してください。

J2EE クライアント アプリケーションについて

WebLogic Server では、標準の XML デプロイメント記述子を使用して jar ファイルにパッケージ化された J2EE クライアント アプリケーションがサポートされています。このコンテキストのクライアント アプリケーションは、Web ブラウザではないクライアントです。それらのクライアント アプリケーションは、Remote Method Invocation（RMI）を使用して WebLogic Server に接続する Java クラスです。Java クライアントでは、エンタープライズ JavaBean、JDBC 接続、JMS メッセージなどのサービスに RMI を使用してアクセスできます。クライアント アプリケーションは、標準の I/O を使用する単純なコマンドライン ユーティリティから Java Swing/AWT クラスを使用して構築された高度に対話型の GUI アプリケーションまでさまざまです。

WebLogic Server Java クライアントを実行するには、クライアント コンピュータで、クライアント アプリケーション クラスだけでなく、weblogic_sp.jar ファイル、weblogic.jar ファイル、および WebLogic Server 上のすべての RMI クラスとエンタープライズ Bean のリモート インタフェースが必要になります。保守とデプロイメントを簡略化するために、クライアントサイド アプリケーションは、weblogic.jar ファイルおよび weblogic_sp.jar ファイルと一緒にクライアントのクラスパスに追加できる jar ファイルにパッケージ化したほうがよいでしょう。この仕様に基づいてクライアント アプリケーションをパッケージ化す

るには、クライアント コンピュータで weblogic.ClientDeployer コマンドライン ユーティリティを実行します。J2EE クライアント アプリケーションの詳細については、「クライアント アプリケーションのパッケージ化とデプロイ」を参照してください。

JMS アプリケーションの移行

WebLogic Server 6.1 では、JavaSoft JMS 仕様バージョン 1.0.2 がサポートされています。

既存の JMS アプリケーションを使用するには、最初に WebLogic Server のバージョンを確認した後、『WebLogic JMS プログラマーズ ガイド』の「WebLogic JMS アプリケーションの移行」で説明されている適切な移行手順を実行する必要があります。

移行手順を開始する前に、以下のリストをチェックして、使用している WebLogic JMS のバージョンで移行がサポートされていることを確認し、特別な移行ルールがそのリリースに適用されるかどうかを明らかにする必要があります。

- Weblogic Server 4.5.1 – 移行は、SP14 に対してのみサポートされています。すべてのサービス パックを実行している場合は、BEA カスタマ サポートまでお問い合わせください。
- Weblogic Server 5.1 – SP07 または SP08 の場合、既存の JDBC ストアをバージョン 6.0 または 6.1 に移行する前に BEA カスタマ サポートに連絡する必要があります。
  - オブジェクト メッセージを移行するには、オブジェクト クラスが、Weblogic Server 6.0 以降のサーバ クラスパス内になければなりません。
  - Weblogic Server 6.0 以降でコンフィグレーションされていない送り先では、移行されたメッセージは失敗し、イベントがログに記録されます。

注意： WebLogic Event は非推奨となり、NO_ACKNOWLEDGE または MULTICAST_NO_ACKNOWLEDGE 配信モードの JMS メッセージに置き換えられます。各配信モードの詳細については、『WebLogic JMS プログラマーズ ガイド』の「WebLogic JMS の基礎」を参照してください。

移行およびデプロイメントの補足事項

以下の節では、WebLogic Server 6.x でアプリケーションをデプロイする際に便利な補足情報を提供します。WebLogic Server 6.1 で非推奨となった機能、アップグレード、および重要な変更点を示してあります。

- 3-44 ページの「スタンドアロンの HTML および JSP」
- 3-45 ページの「アプリケーション サーバと管理対象サーバ」
- 3-45 ページの「Java Transaction API（JTA）」
- 3-45 ページの「Java Database Connectivity（JDBC）」
- 3-46 ページの「RMI」
- 3-47 ページの「インターナショナライゼーション（I18N）」
- 3-48 ページの「セキュリティ」
- 3-48 ページの「WAP アプリケーション」
- 3-49 ページの「セッションの移行」
- 3-50 ページの「Web コンポーネント」
- 3-51 ページの「XML 6.0->6.1 パーサ」
- 3-51 ページの「非推奨となった API および機能」
- 3-51 ページの「削除された API および機能」

スタンドアロンの HTML および JSP

WebLogic Server 6.1 で用意されているオリジナルのドメインと、`weblogic.properties` ファイル コンバータを使用して作成されたすべてのドメインでは、`wlserver6.1\config\`*`domainName`*`\applications\DefaultWebApp` というディレクトリが作成されます。このディレクトリには、Web サーバが公開するファイルが格納されます。HTML ファイルと JSP ファイルは、インス

トールしたアプリケーションとは別にこの場所に配置して公開できます。必要な場合は、画像ファイルなどの相対リンクを処理するために、DefaultWebApp ディレクトリ内にサブディレクトリを作成できます。

アプリケーション サーバと管理対象サーバ

デフォルトでは、アプリケーションは管理サーバの config.xml ブロックおよび JVM 上にデプロイされます。ただし、この方法はほとんどの場合に適切ではありません。管理サーバは管理上の目的のみに使用し、新しい管理対象サーバを定義して、それらのサーバを Administration Console でアプリケーションと関連付ける必要があります。詳細については、「WebLogic Server とクラスタのコンフィグレーション」と「ドメイン、管理サーバ、管理対象サーバ」を参照してください。

Java Transaction API（JTA）

JTA の変更点は以下のとおりです。

- WebLogic Server 6.1 では、JTA 1.0.1 仕様がサポートされています。JTA の最新マニュアルは、『WebLogic JTA プログラマーズ ガイド』で公開されています。
- JTA のサポートの追加のため、JTS JDBC ドライバ（プロパティは weblogic.jts.* 内、URL は jdbc:weblogic:jts:..）が JTA JDBC/XA ドライバに置き換えられています。既存のプロパティは下位互換性を目的として利用できますが、JTS から JTA への名前の変更に合わせてクラス名とプロパティを変更する必要があります。

Java Database Connectivity（JDBC）

JDBC の変更点は以下のとおりです。

- WebLogic Server 6.0 および 6.1 では、WebLogic T3 API は非推奨となり、代わりに RMI JDBC ドライバを使用します。この注意は、WebLogic Server 4.5.x からの移行の場合にも当てはまります。

- weblogic.jdbc20.* パッケージは、weblogic.jdbc.* パッケージに置き換えられます。すべての WebLogic JDBC ドライバが現在は JDBC 2.0 に準拠しています。
- 現在の接続があり、preparedStatement を使用中で、ストアド プロシージャが DBMS に保存されている場合、ストアド プロシージャを作成するには、新しい名前を使用します。同じ名前でストアド プロシージャを再作成する場合、preparedStatement では新しく作成されたストアド プロシージャへのアクセス方法がわかりません。新しく作成されたものは、基本的に、同じ名前を持つ別のオブジェクトであるためです。

RMI

以下のヒントは、以前のバージョンの WebLogic Server で RMI を使用していたユーザが WebLogic Server 6.1 に移行する際のものです。

- 既存のコードで WebLogic RMI コンパイラ（weblogic.rmic）を再実行して、WebLogic Server 6.1 と互換性を持つようにラッパー クラスを再生成してください。
- java.rmi.Remote を使用してインタフェースをリモートとしてタグ付けします。weblogic.rmi.Remote は使用しないでください。
- java.rmi.*Exception（import java.rmiRemoteException など）を使用します。weblogic.rmi.*Exception は使用しないでください。
- *.rmi.Naming の代わりに JNDI を使用します。
- weblogic.rmic を使用して動的プロキシおよびバイトコードを生成しますが、RMI IIOP の例外があります。スタブとスケルトンのクラスは、今後は生成されません。

注意： 詳細については、『WebLogic RMI プログラマーズ ガイド』を参照してください。

- weblogic.rmi.server.UnicastRemoteObject.exportObject() を使用してスタブのインスタンスを取得します。
- 現時点では、RMI のサンプルは、java.rmi.* と JNDI を使用するように更新されていません。将来のリリースで java.rmi.* と JNDI を反映するように見直される予定です。

FileServlet

WebLogic Server 6.1 サービス パック 2 以降では、Web アプリケーションのデフォルトのサーブレットである FileServlet の動作が変更されています。現在の FileServlet では、ソース ファイル名を指定する際に SERVLET_PATH も指定できます。この設定によって、FileServlet を `/dir/*` などにマップすることで、特定のディレクトリからのファイルのみを明示的に提供できるようになりました。

『デフォルト サーブレットの設定』を参照して下さい。

インターナショナライゼーション（I18N）

このバージョンでは、インターナショナライゼーションとローカライゼーションが一部変更されています。

- ログ ファイル フォーマットに、メッセージのローカライズ方法に影響する変更が行われています。新しいメッセージ フォーマットでは、最初の行に begin marker、machine name、server name、thread id、user id、tran id、および message id も追加されています。
- サーバおよびクライアントでメッセージをロギングするためにユーザが使用できるインターナショナライズされたロギング API も追加されました。
- クライアントは、それぞれのログファイルにロギングします。このログファイルは、servername および threadid フィールド以外はサーバのログファイルと同じフォーマットです。
- `LogServicesDef` は非推奨になりました。代わりに、インターナショナライズされた API または `weblogic.logging.NonCatalogLogger`（インターナショナライゼーションが不要な場合）を使用してください。

このバージョンのインターナショナライゼーションの詳細については、『BEA WebLogic Server インターナショナライゼーション ガイド』を参照してください。

セキュリティ

WebLogic Server 6.1 でのセキュリティの詳しい使い方については、『WebLogic Security プログラマーズ ガイド』を一読することをお勧めします。以前のリリースから移行する WebLogic Server ユーザは、以下のヒントと問題に目を通してください。

- WebLogic Server 環境でのセキュリティ実装の大部分はコンフィグレーションです。現在のセキュリティ コンフィグレーションを移行するには、Administration Console を使用して `weblogic.properties` ファイルを `config.xml` の XML 属性に変換します。`weblogic.properties` ファイルの変換の詳細については、Administration Console ヘルプを参照してください。セキュリティ プロパティと XML 属性のすべてのマッピングについては、『管理者ガイド』の「セキュリティの管理」を参照してください。
- デフォルト セキュリティ レルムの名前は、`WLPropertyRealm` からファイル レルムに変更されています。レルム属性は、現在は `weblogic.properties` ファイルではなく `fileRealm.properties` ファイルに格納されます。
- ある時点で、Administration Console を使用してレルム属性および認可属性を再定義する必要があります。再定義した情報は、`fileRealm.properties` ファイルに格納します。WebLogic Server 6.x でのカスタム レルムの作成については、『カスタム セキュリティレルムの作成』を参照してください。
- すべてのセキュリティ設定をチェックして、環境に適した設定になっていることをインストールの最後で確認することをお勧めします。
- スタンドアロン サーブレットは Web アプリケーションに完全に置き換えられたので、ACL を使ってスタンドアロン サーブレットに対するセキュリティを指定することはできなくなっています。サーブレット 2.3 仕様で定義されているように、Web アプリケーションのセキュリティは、Web アプリケーションのデプロイメント記述子を使ってだけ設定できます。

WAP アプリケーション

WebLogic Server 6.1 で WAP アプリケーションを実行するには、Web アプリケーションの `web.xml` ファイルで WAP と関連付けられている MIME タイプを指定する必要があります。WebLogic Server 5.1 では、デフォルトの mime-type を

weblogic.properties の weblogic.httpd.defaultMimeType を使用して設定できます。5.1 でのデフォルト値は「text/plain」です。WebLogic Server 6.0、WebLogic Server 6.1、および WebLogic Server 7.0 では、デフォルトの mime-type はありません。web.xml ファイルで、それぞれの拡張子に対して明示的に mime-type を指定する必要があります。必要な MIME タイプについては、『WebLogic Server Wireless Application 開発プログラマーズ ガイド』を参照してください。web.xml ファイルの作成と編集については、「Web アプリケーションのデプロイメント記述子の記述」を参照してください。

web.xml ファイルで mime-types をコンフィグレーションする例を次に示します。

```
<web-app>
  <mime-mapping>
    <extension>tiff</extension>
    <mime-type>image/tiff</extension>
  </mime-mapping>
  <mime-mapping>
    <extension>tiff</extension>
    <mime-type>image/tiff</extension>
  </mime-mapping>
</web-app>
```

セッションの移行

WebLogic Server 6.0 以降のバージョンでは、以前のバージョンからのクッキーが認識されません。バージョン 6.0 でクッキーのフォーマットが変更されたからです。WebLogic Server では、旧式のフォーマットのクッキーは無視され、新しいセッションが作成されます。

クッキーのデフォルト名はバージョン 5.1 から変更されています（以前の名前は WebLogicSession）。WebLogic 6.0 以降は、クッキーのデフォルト名は JSESSIONID です。

詳細については、「web.xml デプロイメント記述子の要素」を参照してください。

Web コンポーネント

以下のヒントは、以前のバージョンの WebLogic Server で Web コンポーネントを使用していたユーザが WebLogic Server 6.1 に移行する際のものです。

- WebLogic Server のすべての Web コンポーネントでは、WebLogic Server で JSP、サーブレット、および静的な HTML ページがどのように提供されるのかを定義するメカニズムとして Web アプリケーションが使用されます。WebLogic Server の新規インストールでは、サーバでデフォルト Web アプリケーションがコンフィグレーションされます。WebLogic Server 6.1 にアップグレードする場合は、以前のバージョンで `weblogic.properties` ファイルを使用して実行されたドキュメント ルート、JSPServlet、およびサーブレットの登録に近いことが行われるので、改めて登録を行う必要がありません。
- 既存の `weblogic.properties` ファイルを `.xml` に変換するには、Administration Console を使用します。詳細については、Administration Console ヘルプを参照してください。
- SSI は現在はサポートされていません。
- URL ACL は非推奨です。代わりにサーブレット 2.3 の機能を使用します。
- 一部の情報が `web.xml` から `weblogic.xml` に移動されました。この情報の再編成によって、サーブレット 2.2 に厳密に基づくサード パーティの Web アプリケーションが、J2EE 標準デプロイメント記述子（`web.xml`）を変更せずにデプロイできるようになりました。`<context-param>` 要素を使用して `web.xml` ファイルで行われる WebLogic Server 5.1 スタイルの設定が下位互換性のためにサポートされていますが、新しいデプロイメントの方法を採用することをお勧めします。以前は `web.xml` で定義されていた以下のパラメータは、現在は `weblogic.xml` で定義されます。

  JSP パラメータ（*keepgenerated*、*precompile compileCommand*、*verbose*、*packagePrefix*、*pageCheckSeconds*、*encoding*）

  HTTP セッション パラメータ（*CookieDomain*、*CookieComment*、*CookieMaxAgeSecs*、*CookieName*、*CookiePath*、*CookiesEnabled*、*InvalidationIntervalSecs*、*PersistenceStoreDir*、*PersistenceStorePool*、*PersistenceStoreType*、*SwapIntervalSecs*、*IDLength*、*CacheSize*、*TimeoutSecs*、*JDBConnectionTimeoutSecs*、*URLRewritingEnabled*）

- 詳細については、「Web アプリケーションのデプロイメント記述子の記述」を参照してください。

XML 6.0->6.1 パーサ

XML 6.0 -> 6.1 パーサが更新されており、現在は Apache Xerces パーサに基づいています。このパーサでは、SAX インタフェースおよび DOM インタフェースのバージョン 2 が実装されています。`weblogicaux.jar` に格納されていた古いパーサ（Sun の Project X パーサなど）を使用すると、旧式であることを示すメッセージが表示される場合があります。

非推奨となった API および機能

以下の API および機能は、将来的に削除される予定なので使用しないでください。

- WebLogic Event

  WebLogic Event は非推奨となり、NO_ACKNOWLEDGE または MULTICAST_NO_ACKNOWLEDGE 配信モードの JMS メッセージに置き換えられます。詳細については、『WebLogic JMS プログラマーズ ガイド』を参照してください。

- WebLogic HTMLKona
- T3 ドライバ
- `-Dweblogic.management.host`

削除された API および機能

以下の API および機能は削除されています。

- 旧式の管理コンソール GUI

  新しい Administration Console を使用してください。

- デプロイヤ ツール

- WebLogic Bean
- WebLogic jHTML
- WebLogic Remote
- ワーク スペース
- WebLogic Server Tour
- T3Client
- Jview サポート
- SSI
- Weblogic Bean バー
- RemoteT3
- Jview サポート
- Weblogic COM

  この機能は、現在はサポートされていない Microsoft JVM（Jview）に基づいてます。

4 WebLogic Server 6.0 アプリケーションの WebLogic Server 6.1 への移行

以下の節では、アプリケーションをバージョン 6.0 から 6.1 に移行する手順、および移行の際に注意する事項について説明します。

- バージョン 6.0 からバージョン 6.1 へのアプリケーションの移行
- JMS
- EJB 2.0
- JMX
- Apache XML パーサ
- Xalan XML パーサ
- Web アプリケーション
- セキュリティ
- config\applications ディレクトリ
- Solaris での WebLogic Server クラスタ
- スレッド プールのサイズ

バージョン 6.0 からバージョン 6.1 へのアプリケーションの移行

アプリケーションを WebLogic Server 6.0 から WebLogic Server 6.1 に移行するには、次の手順に従います。

1. WebLogic Server 6.1 をインストールします。

   注意： 古いバージョンの上に直接、新しいバージョンをインストールすることは、インストーラによって明示的に禁止されています。

2. WebLogic Server 6.1 に移行するコンフィグレーション ドメインごとに、ドメイン ディレクトリを WL_HOME\config ディレクトリにコピーします。このディレクトリは、WebLogic Server 6.1 のすべてのコンフィグレーション ドメインの格納に使用されます。バージョン 6.0 の config ディレクトリが WebLogic Server 6.0 配布キット内にない場合は、WebLogic Server 6.1 で WebLogic 6.0 コンフィグレーションを再利用してもかまいません。

   a. config.xml で参照されるすべての外部ファイルを確認します。
   WebLogic Server のコンフィグレーションは、ファイル システムに格納されるファイルに基づきます。通常、これらのファイルは、永続性リポジトリ（ログ ファイル、ファイルベースのリポジトリなど）またはユーティリティ（Java コンパイラ）であり、絶対パスまたは相対パスを使用してコンフィグレーションされます。

   すべての外部ファイルが相対パスで定義され、ドメイン ディレクトリ内またはその下位にある場合は、手順 3.b に進んでください。

   外部ファイルがドメイン ディレクトリの外部を示す相対パスで定義されている場合は、新しい config ディレクトリを基準にしたディレクトリ構造を再作成し、関連付けられるファイルを新しいディレクトリにコピーします。外部ファイルが絶対パスで定義されている場合は、これらのファイルが WebLogic Server 6.1 デプロイメントで適切に再利用できるかどうかを確認します。

   たとえば、ログ ファイルおよび永続性ストレージは再利用できますが、Java コンパイラなどのユーティリティは最新バージョンを使用するためには更新が必要になる場合もあります。更新が必要なファイルについては、新しいファイルまたはユーティリティを使用するよう、WebLogic

Server 6.0 Administration Console を使用して適切な属性を再コンフィグレーションしてから、次の手順に進みます。

デプロイメント記述子（`web.xml` および `weblogic.xml`）についても検討する必要があります。Java コンパイラなどの項目へのファイル パスが含まれている場合があるからです。

b. 適切なドメイン ディレクトリおよび `console.war` 以外のすべてのサブコンポーネント（ディレクトリとファイル）を、手順 2 で作成した `config` ディレクトリにコピーします。

すべてのトランザクション ログ（*.tlog）と数多くのメッセージ キューも必ず、ドメインと共に移動させてください。

3. WebLogic Server 6.1 ライブラリを使用するように、新しいドメインのサーバ起動スクリプトを編集します。

`CLASSPATH -`

WebLogic Server 6.1 配布キットの weblogic.jar を含める必要があります。

`java.security.policy property -`

WebLogic Server 6.1 配布キットの weblogic.policy ファイルを指す必要があります。

`bea.home property -`

WebLogic Server 6.1 の license.bea ファイルを含む BEA ホーム ディレクトリを指す必要があります。

JMS

WebLogic 6.0 から WebLogic Server 6.1 に移行する際には、以下の事項を考慮してください。

- JMS のマニュアルでは JMSDestinationMBean の StoreEnabled 属性について「default」、「true」、および「false」と正しく指定されているものの、WebLogic Server 6.0 では大文字 / 小文字を区別せずに処理できました。しかし、WebLogic Server 6.1 では、StoreEnabled 設定はすべて小文字で指定する必要があります。

- [JMS 接続ファクトリ] の [確認応答ポリシー] 属性では、[All] というデフォルト値を使用して、JMS 1.0.2 仕様の変更に対応しています。[確認応答ポリシー] 属性のデフォルト設定は、JMS の以前のバージョンからの変更を表します。以前は Administration Console にオプションとして表示されていなかったので、[確認応答ポリシー] 属性は内部的にデフォルトで [Previous] になっていました。
  - [All] ー どのメッセージで確認応答メソッドが呼び出されたのかに関係なく、特定のセッションで受信されたすべてのメッセージで確認応答が行われます。
  - [Previous] ー 確認応答メソッドを呼び出したメッセージまでの、特定のセッションで受信されたすべてのメッセージで確認応答が行われます。デフォルトのメッセージ駆動型 Bean（MDB）接続ファクトリでは既に設定されているが、外部の接続ファクトリではそうではありません。

WebLogic JMS アプリケーションの旧バージョンを移行する方法の詳細については、『WebLogic JMS プログラマーズ ガイド』の「WebLogic JMS アプリケーションの移行」を参照してください。

EJB 2.0

Sun Microsystem の EJB 2.0 仕様は、WebLogic Server のバージョン 6.0 と 6.1 の間で変更されています。そのため、デプロイメント記述子ディレクティブの中にはサポートされなくなったものがあります。これは、WebLogic Server 6.0 のアプリケーションは変更しなくても WebLogic Server 6.1 のサーバで機能するという、WebLogic Server 6.0 から 6.1 の移行ルールの例外になっています。WebLogic Server 6.0 で機能した EJB 1.1 Bean は、変更しなくても WebLogic Server 6.1 で機能します。

Weblogic 6.1 では、新しいインタフェース `EJBComponentMBean` によって `ComponentMBean` と `EJBContainerMBean` の両方が拡張されます。`EJBComponentMBean` を実装するすべてのカスタム Mbean（`getVerboseEJBDeploymentEnabled` など）は、WebLogic Server 6.0 から 6.1 に移行する際にこのインタフェースをサポートするように修正する必要があります。`EJBContainerMBean` が拡張されると、`EJBComponentMBean` はその `EJBComponentMBean` が EJB コンテナでなくても `EJBContainerMBean` と同じメソッドを提供します。

EJB 2.0 セッション Bean と BMP エンティティ Bean のアップグレード

EJB 2.0 セッション Bean または BMP エンティティ Bean モデルに大きな変更はありません。WebLogic Server 6.1 で WebLogic EJB コンパイラ（ejbc）を再実行すると、`ejb-jar.xml` ファイルが確実に最新の EJB 2.0 DTD に準拠します。

EJB 2.0 メッセージ駆動型 Bean（MDB）のアップグレード

`ejb-jar.xml` ファイルで、メッセージ駆動型 Bean の EJB 2.0 デプロイメント記述子要素が変更されています。`<jms-acknowledge-mode>` は `<acknowledge-mode>` に変更する必要があり、`<jms-destination-type>` は `<destination-type>` になりました。

WebLogic Server 6.0 のメッセージ駆動型 Bean ユーザは、デプロイメント記述子に `<run-as-specified-identity>` というタグがある場合があります。このタグは WebLogic Server 6.0 でサポートされていず、その名前は最新の EJB 2.0 の変更で `<run as>` に変更されています。WebLogic Server 6.0 ではこの機能がサポートされていなかったので、ほとんどの 6.0 メッセージ駆動型 Bean ユーザは run-as デプロイメントを必要とせず、この要素を `ejb-jar.xml` ファイルから削除できます。run-as ID が必要な場合は、`ejb-jar.xml` ファイルでその要素を `<run-as>` に変更する必要があります。

EJB 2.0 CMP エンティティ Bean のアップグレード

EJB 2.0 仕様では、CMP 2.0 モデルが多くの点で変更されています。

最初のステップは、ejb-jar.xml ファイルを確実に最新の EJB 2.0 DTD に準拠させることです。要素の名前がいくつか変更されています。<role-source> タグは、<relationship-role-source> という名前になりました。<run-as-specified-identity> タグは、<run-as> になっています。

最新の EJB 2.0 仕様では、ローカル インタフェースという概念が導入されています。EJB の関係は、ローカル インタフェースに基づくようになりました。関係に関与するすべての EJB では、ローカル インタフェースが必要です。

リモートの関係は、EJB 2.0 仕様から削除されました。WebLogic Server での関係は現在は EJB 2.0 CMP エンティティ Bean 間だけであり、関係する Bean は同じ EJB JAR ファイルにデプロイする必要があります。

JMX

WebLogic Server 6.0 のすべてのパブリックな MBean および属性は、WebLogic Server 6.1 でサポートされています。ただし、内部 MBean または属性を使用している場合には、移行の問題に直面するかもしれません。

Apache XML パーサ

XML WebLogic Server 6.0 -> 6.1 パーサが更新されており、現在は Apache Xerces 1.3.1 パーサに基づいています。このパーサでは、SAX インタフェースおよび DOM インタフェースのバージョン 2 が実装されています。以前のバージョンで提供されていた古いパーサを使用すると、旧式であることを示すメッセージが表示される場合があります。

xmlx.jar ファイルで提供されるパーサは非推奨になっていますが、下位互換性を目的として提供されます。外部パーサを使用する場合は、JAXP を通じて使用する必要があります。非 JAXP コードから JAXP コードへの移行では、コードの調整が必要になる場合があります。

Xalan XML パーサ

Xalan API は非推奨になりました。それらの API を依然として使用して問題が発生する場合は、JAXP API を使用して XSLT を使用することをお勧めします。

Apache の Xalan コードは、Xerces と Xalan を一緒に使用できるように変更が行われました。それらの変更が含まれないので、Apache から Xalan を使用すると問題が発生する場合があります。

通常は、JAXP を使用し、ベンダ固有のコードを SAX、DOM、および XSL 処理用に JAXP などのニュートラルな API に移行するのが最適です。

Web アプリケーション

`weblogic.management.descriptors.webapp.ServletMBean.getName() API`（WebLogic Server 6.0）は、WebLogic Server 6.1 で `weblogic.management.descriptors.webapp.ServletMBean.getServletName()` に変更されています。このインタフェースを使用する場合は、ソース コードをアップデートして再コンパイルする必要があります。

Java サーブレット仕様 2.3 では、転送時の認可がデフォルトでは行われなくなりました。セキュアなリソースへの転送時に認可を得るには、`<check-auth-on-forward>` を `weblogic.xml` に追加します。

サーブレットのリクエスト オブジェクトと応答オブジェクトには新しい API があります。それらのオブジェクトのシリアライズ可能で軽量な一部の実装は、新しい API を実装しないとコンパイルされない可能性があります。新しいサーブレット 2.3 モデルを使用し、サーブレットのリクエスト オブジェクトと応答オブジェクトの実装を置換することをお勧めします。WebLogic Server 6.0 でこれを行った場合は、それらのオブジェクトの文書化されていない内部実装に基づいているものと思われます。WebLogic Server 6.1 ではサーブレット 2.3 がサポートされているので、新しい ServletRequest/ResponseWrapper オブジェクトを利用できます。

セキュリティ

サーバのデフォルトの状態では、すべての MBean に対して最も厳しいアクセス制御リスト（ACL）が課されます。したがって、非システム ユーザとして MBean をプログラム的に変更すると、制限を緩めない限りアクセスが拒否されます。

config\applications ディレクトリ

WebLogic Server 6.1 では、実行時モードによって「開発」モードと「プロダクション」モードの 2 つのモードに区別されます。実行時モードは、Weblogic 管理サーバの起動時にコマンドライン パラメータを使用して選択します（`-Dweblogic.ProductionModeEnabled=true | false`）。このパラメータを設定しないと、サーバは開発モードで動作します。開発モードでは、サーバの動作は WebLogic Server 6.0 の場合と同じです。ただし、プロダクション モードの場合には自動デプロイメント機能が無効になります。コンフィグレーション リポジトリ（`config.xml`）で明示的にデプロイされない applications ディレクトリのデプロイメント ユニットは、自動的にはデプロイされません。WebLogic Server 6.1 では、デフォルト（mydomain）コンフィグレーションと Pet Store コンフィグレーションは開発モードで提供されます。サンプル コンフィグレーションは、開発モードで提供されます。applications ディレクトリで変更を調査する AppManager スレッドが管理サーバだけで作成されるので、この機能は管理サーバでのみ機能します。

Solaris での WebLogic Server クラスタ

Solaris 上の WebLogic Server クラスタにデプロイされた特定のアプリケーション（重い EJB アプリケーション）は、サーバ JVM ではなくクライアント JVM を使用することでパフォーマンスが向上します。このことは、負荷の大きな状況で特に顕著になります。

スレッド プールのサイズ

WebLogic Server 6.0 では、ワーカ スレッドの数はサーバ MBean の ThreadPoolSize パラメータで指定しました。WebLogic Server 6.1 からは、ワーカ スレッドの数はサーバ MBean の ExecuteQueue で定義します。

WebLogic Server 6.1 ではこのパラメータの移行パスが提供されるので、それが config.xml で指定される場合、またはコマンドラインでクライアントかサーバに渡される場合（-Dweblogic.ThreadPoolSize=<xx>）、ThreadPoolSize が自動的に ThreadCount 設定に移行されます。

WebLogic Server 6.1 サービス パック 1 では、コンソールのバグによって、ThreadCount に不正確な値（デフォルト値）が表示されていました。ただし、これはコンソールの中だけでのバグであり、クライアントおよびサーバでは正確な数のスレッドが使用されていました。

ワーカ スレッド数は次のように設定します。

WebLogic Server 6.0 の場合

```
<Server
Name="myserver"
ThreadPoolSize="23"
...
/Server>
```

WebLogic Server 6.1 以降

```
<Server
Name="myserver"
... >
<ExecuteQueue
Name="default"
ThreadCount="23" />
/Server>
```

コンソールでスレッド数の値を変更するには、次の手順を行います。

1. コンソールで、[**サーバ**｜ **myServer** ｜**モニタ**] を選択します。
2. [**すべてのアクティブなキューのモニタ**] をクリックします。

3. 「**default**」キューをクリックします（スレッドとそれらのスレッドの処理内容のリストが表示される）。
4. [**Execute Queues のコンフィグレーション**]（ページの最上部）をクリックします。
5. 「**default**」キューをクリックします。
6. このサーバと関連付けられているスレッド数を入力します。
7. サーバを再起動して変更を有効にします。

5　注意事項

以下の節では、このソフトウェア リリースにおいて確認されている問題について説明します。

- 5-3 ページの「コネクタに関する確認済みの問題」
- 5-3 ページの「Administration Console に関する確認済みの問題」
- 5-4 ページの「コアに関する確認済みの問題」
- 5-5 ページの「EJB に関する確認済みの問題」
- 5-9 ページの「サンプルと Pet Store Demo に関する確認済みの問題」
- 5-14 ページの「インストーラに関する確認済みの問題」
- 5-16 ページの「JCA に関する確認済みの問題」
- 5-16 ページの「JDBC と jDrivers に関する確認済みの問題」
- 5-18 ページの「JMS に関する確認済みの問題」
- 5-18 ページの「JSP に関する確認済みの問題」
- 5-20 ページの「JVM に関する確認済みの問題」
- 5-22 ページの「RMI に関する確認済みの問題」
- 5-22 ページの「RMI-IIOP に関する確認済みの問題」
- 5-23 ページの「セキュリティに関する確認済みの問題」
- 5-25 ページの「サーバに関する確認済みの問題」
- 5-28 ページの「サーブレットと Web アプリケーションに関する確認済みの問題」
- 5-31 ページの「システム管理に関する確認済みの問題」
- 5-35 ページの「ツールに関する確認済みの問題」
- 5-35 ページの「Web サービスに関する確認済みの問題」

- 5-38 ページの「WebLogic Tuxedo Connector に関する確認済みの問題」
- 5-40 ページの「XML に関する確認済みの問題」
- 5-41 ページの「ZAC に関する確認済みの問題」

コネクタに関する確認済みの問題

変更要求番号	説明
098342	WebLogic Server 6.1 の JCA の実装には、IBM CICS リソース アダプタとの互換性がない。BEA が実装する `javax.resource.spi.ConnectionManager` インタフェースは、リソース アダプタが実装する `javax.resource.spi.ManagedConnection` インタフェースの `getConnection()` メソッドによって返されるオブジェクトを直接返さない。代わりに、`java.lang.reflect.Proxy` クラスによって動的に生成されたクラスのインスタンスを返す。

Administration Console に関する確認済みの問題

変更要求番号	説明
053277	Netscape を使用して Administration Console を呼び出し、LDAP レルム コンフィグレーション用のテンプレートを編集する場合に、[コンフィグレーション情報] ウィンドウで矢印またはスクロール バー ボタンを使用して左右に移動すると、表示データが文字化けしたようになる。[適用] ボタンをクリックしても文字化けしたデータは `config.xml` に保存されないが、テンプレートの編集が困難になる。

コアに関する確認済みの問題

変更要求番号	説明

CR093104	SP04 および SP05 では、NT のパフォーマンス パックが原因で、スレッドのデッドロックが発生する場合がある。この問題は、データベースにアクセスする .jsp の負荷テストの際に、小さな負荷において発生した。スレッド ダンプは、「デフォルト」の実行スレッドの大部分が、「モニタ エントリ待ち」でスタックしていることを示していた。
	`"ExecuteThread: '10' for queue: 'default'" daemon prio=5 tid=0x273fb548 nid=0xadc waiting for monitor entry [0x2810f000..0x2810fdc4]`
	`at weblogic.socket.NTSocketMuxer.initiateIO(NTSocketMuxer.java:483)`
	`at weblogic.socket.NTSocketMuxer.read(NTSocketMuxer.java:474)`
	`at weblogic.servlet.internal.MuxableSocketHTTP.requeue(MuxableSocketHTTP.java:244)`
	`at weblogic.servlet.internal.ServletResponseImpl.send(ServletResponseImpl.java:1094)`
	`at weblogic.servlet.internal.ServletRequestImpl.execute(ServletRequestImpl.java:2364)`
	`at weblogic.kernel.ExecuteThread.execute(ExecuteThread.java:139)`
	`at weblogic.kernel.ExecuteThread.run(ExecuteThread.java:120)`
	このようなデッドロックが発生する場合は、`.hotspot_compiler` という名前で次のような内容のファイルを、JVM を呼び出すディレクトリに作成する必要がある。
	`exclude  weblogic/socket/NTSocketMuxer processSockets`

CR106616	Netscape Version 4.79 を Linux AS2.1 および HP-UX 上で実行する場合に問題がある。 WebLogic Server Administration Console でいくつかの操作を行った後、バス エラーによってブラウザが停止する (コア ダンプが生成されることもある)。

EJB に関する確認済みの問題

変更要求番号	説明
035884	WebLogic Server のインスタンスにデプロイ可能なエンタープライズ JavaBean (EJB) は最大で 390。これは Sun のバグで、1.3 JDK による制限。JDK 1.3.1 リリースで修正される可能性がある。Sun Bug ID は 4390238。
051543	まったく不適切な EJB をデプロイしようとすると、EJB コンテナによってそのデプロイ メソッドから例外が送出される。しかし、Administration Console でその EJB をクリックすると、[デプロイ] チェックボックスがチェックされたままになっている。

変更要求番号	説明

061938	特定の条件下で、EJB QL クエリが SQL クロス積の結果である擬似重複を返すことがある。発生する可能性のある条件は次のとおり。
	■ EJB-QL クエリに関係をナビゲートするパス表現が含まれている。この場合、生成された SQL SELECT 句で複数のテーブルが生成される。
	■ The WHERE 句に関係をナビゲートする OR オペランドが含まれており、OR オペランド中のすべてのパス表現が、生成された SQL SELECT 句のすべてのテーブルにマップされていない。この場合、その OR オペランドに対する結果にクロス積が現われる場合がある。
	次の例に、この問題を示す。
	EJB QL:
	`SELECT OBJECT(c)`
	`FROM CustomerBean AS c, IN(c.accounts)accts`
	`WHERE c.name = '100' OR c.accts.bal = 300`
	DATA: `customer '100' exists but has no accounts`
	EXPECTED RESULT: `customer '100' from clause #1`
	ACTUAL RESULT: `customer '100' X number of accts`
	SQL:
	`DROP TABLE thorick_customers;`
	
	`CREATE TABLE thorick_customers (`
	`cust_name VARCHAR(10),`
	`cust_interests VARCHAR(10),`
	`cust_rating INTEGER, acct_id INTEGER,`
	`PRIMARY KEY (cust_name));`
	
	`DROP TABLE thorick_accounts;`
	`CREATE TABLE thorick_accounts`
	`(acct_id INTEGER,`
	`bal FLOAT,`
	`PRIMARY KEY (acct_id));`
	
	`INSERT INTO thorick_accounts VALUES (100, 100.0);`
	`INSERT INTO thorick_accounts VALUES (200, 200.0);`
	`INSERT INTO thorick_accounts VALUES (300, 300.0);`
	`INSERT INTO thorick_accounts VALUES (400, 400.0);`
	`INSERT INTO thorick_accounts VALUES (500, 500.0);`
	`INSERT INTO thorick_customers VALUES('100', 'jazz', 2, null);`
	`INSERT INTO thorick_customers VALUES('900', 'punk', 3, 400);`
	
	`SELECT WL0.cust_name, WL0.cust_interests, WL0.acct_id`
	`FROM thorick_Customers WL0, thorick_Accounts WL1`
	`WHERE WL0.cust_name = '100'`
	`OR ( wl1.bal = 300 AND wl0.acct_id=wl1.acct_id );`
	
	`CUST_NAME  CUST_INTER  ACCT_ID`
	`----------  ----------  ----------`
	`100  jazz`
	`100  jazz`
	`100  jazz`
	`100  jazz`
	`100  jazz`

変更要求番号	説明
062927	WLS 6.1 によって提供される JSP から、WLE 5.1 によって提供される EJB を要求すると、次の例外が発生する可能性がある。 `java.lang.NoClassDefFoundError` WLE 5.1 の `wlej2eecl.jar` ファイルの場所を、WLS 6.1 の `startWeblogic.cmd` スクリプトまたは `startWeblogic.sh` スクリプトの CLASSPATH に追加する必要がある。
079539	WebLogic Server 6.1 sp3 は、WLI 2.1 sp1 の Application Integration と互換性がない。詳細についての問い合わせは、『BEA カスタマ サポート』まで。
088461	`ejbc` の動作が WebLogic Server 6.1 SP02 と SP03 で異なる。SP02 では、サーバにデプロイする前の Bean に対して ejbc を実行しても期待通りの結果が得られたが、SP03 でそのような Bean に対して同様の処理を行うと、`ejbc` でプリコンパイルせずにホット デプロイされていた場合に処理が失敗し、`OutOfMemory` 例外が送出されることがある。 これを回避するには、以下を使用する。 `java -verbosegc weblogic.ejbc -compiler javac -J-mx128m entityBeans.jar final.jar`
089847	WebLogic Server の EJB デプロイメント プロパティの多くには、パフォーマンスの向上を目的として最適化されたデフォルト値が用意されている。一部のデフォルト値は、EJB の仕様に準拠していない。WebLogic Server を EJB 仕様に準拠させるには、以下のようにプロパティを設定する必要がある。 weblogic-ejb-jar.xml で、「enable call by reference」を False に設定する。 weblogic-cmp-jar.xml で、「include-updates」を True に設定する。 weblogic-cmp-jar.xml で、「check exists on method」を True に設定する。

変更要求番号	説明
097323	`.jar` ファイルにコンパイルされていないクラスやインタフェースが含まれる場合、クラスタ内の単一のサーバインスタンスに EJB をデプロイまたは再デプロイしようとすると (固定されたデプロイメント)、問題が発生することが確認されている。 デプロイメントの際に、コンパイルされていない EJB はクラスタ内の各サーバインスタンスにコピーされるが、コンパイルが行われるのは、EJB がデプロイされたサーバインスタンスだけである。その結果、EJB のデプロイ対象になっていないサーバインスタンスには、EJB を呼び出すために必要なクラスでコンパイルの際に生成されるものが存在していない。別のサーバインスタンス上のクライアントが固定された EJB を呼び出そうとすると失敗し、RMI レイヤで `Assertion` エラーが送出される。 **回避策 :** クラスタ内の単一のサーバインスタンスに EJB をデプロイまたは再デプロイする場合は、デプロイする前に appc または ejbc を使って EJB をコンパイルし、生成されるクラスがすべてのサーバインスタンスにコピーされて、クラスタ内の全ノードで利用できるようにする。

サンプルと Pet Store Demo に関する確認済みの問題

変更要求番号	説明

043156	`ejbmanagedclient.jsp` は、`t3://localhost:7001` に設定した変数を使用するので、ポート番号を動的に取得しない。このため、ページが要求されたときにサーバを見つけることができない。スタック トレースは出力されないが、例外が送出される。ページは次のように表示される。
	`--------------UnexpectedError--------------`
	7001 以外のポートを使用する場合は、jsp ファイルを編集する必要がある。
	編集の必要がある変数は次のセクションにある。
	`<!-- Here, we declare a class method -->`
	`<%!`
	`String pagetitle = "JSP example using EJBean-managed persistence";`
	`String url = "t3://localhost:7001";`
	インストールされているサーバ / ポート番号を指すように URL を編集する。

044641	`examples/ejb/sequence/userDesignated/package-summary.html` で情報が欠落している。必ず、サンプルのパッケージに含まれている `table.ddl` ファイルをこのサンプルで作成するテーブルのソースとして使用する。また、テーブルの名前は「NAMED_SEQUENCE-TABLE」ではなく「NAMED_SEQUENCE_TABLE」にする。
047654	dbKona サンプルでは、プール接続ができず、Example サーバを起動しようとするとエラーが発生する。
047928	Web アプリケーション セキュリティ サンプルを実行すると、グループを作成したり、グループにメンバーを追加したりするためのリンクが Administration Console に表示されない。また、[グループ] ウィンドウで、[Name] または [メンバ] をクリックすると、「500 Internal Server Error」と表示される。
048358	`security/rdbmsrealm/rdbmsrealm.ddl` サンプルと `utils/ddl/demo.ddl` サンプルでコメント記号に誤りがある。

変更要求番号	説明
052556	WLEC サンプルの実行後に examples ドメインで WebLogic Tuxedo Connector CORBA の `simpappcns` サンプルを実行すると失敗する。この問題の原因は、WLEC サンプルで SERVER_CLASSES ディレクトリに残される Simpapp クライアント スタブとの衝突にある。`simpappcns` サンプルを実行する前に、SERVER_CLASSES 環境変数を確認する。WLEC サンプルが実行されたら、SERVER_CLASSES からすべての `Simpapp` クライアント スタブを削除する。
053356	`samples/examples/wtc/corba/simpappcns` サンプルに `run.sh` UNIX スクリプトが用意されておらず、サンプルが UNIX マシンで機能しない。 この問題は、WebLogic Server 6.1 SP04 で解決された。「053356」を参照。
062545	Solaris の `simpappcns` を実行している WebLogic Server ドメインは、NT で動作している Tuxedo Corba Server ドメインを呼び出すことができない。 この問題は、WebLogic Server 6.1 SP03 で解決された。「062545」を参照。

062799	WIRELESS サンプルのうち 3 つのドキュメントでは、手順説明の手順 7 に誤りがある。問題のあるサンプルは、helloWorld、stockDemo、および travelDemo である。
	手順 7 は、次の記述が正しい。
	`Path=./config/examples/helloWorld`
	次の記述は誤りである。
	`Path=./config/examples`
	このようにしないと、サンプルはデプロイされたアプリケーションを発見できない。

064001	`examples.webservices.message.package-summary.html` に、正しくない指示が含まれている。手順 13 に、「[送り先] ノードを右クリックし、ドロップダウン リストから [新しい <JMSTopic> のコンフィグレーション] を選択します。」と記述されている。 JMSTopic は JMSQueue でなければならない。
078441	一部のサンプルは、データ ソースを割り当て直すと動作しなくなる。データ ソースを割り当て直すと「Cannot obtain connection after X seconds」というエラー メッセージが表示されてサンプルが動作しない場合は、サンプル サーバを再起動し、変更を有効にする必要がある。

変更要求番号	説明

CR102138	WebLogic Server 6.1 SP05 では、`jssecacerts` ファイルに正しい `ca.pem` ファイルが含まれていないため、SSLClient サンプルのコード例の JSSE バージョンが動作しないことがある。
	
	この問題を回避するには、WebLogic Server の配布キットに含まれる `ca.pem` ファイルを `jssecacerts` ファイルに追加する。
	
	`ca.pem` を `jssecacerts` ファイルに追加するには：
	1. コマンド ウィンドウを開き、次のディレクトリに移動する。
	`*WL_HOME*\samples\examples\security\sslclient>`
	2. プロンプトで、次のように入力する。
	`keytool -v -keystore jssecacerts -trustcacerts -import -file %WL_HOME%\config\examples\ca.pem`
	3. パスワードを要求されたら、次のように入力する。
	`changeit`

CR103521	`dbkona` サンプルのドキュメントにおいて、サンプルのコンパイルと実行に不可欠な以下の手順が抜けている。 1. 使用している Oracle のバージョンに基づいて、適切な `weblogicoci37.dll` を選択する。PATH 環境変数に正しい DLL へのパスを追加する。 2. 以下のコマンドを実行してデータベースをロードする。 `java utils.Schema "jdbc:weblogic:oracle" weblogic.jdbc.oci.Driver -s oracle_server -u scott -p tiger -verbose utils\ddl\demo.ddl` 3. サンプルのコードをコンパイルする前に、サンプルの Java ファイルをすべて編集して、その `server` プロパティを前の手順で指定した *`oracle_server`* の値に変更する。

変更要求番号	説明
CR103547	HP および Solaris では、dbkona サンプルを起動できない。dbkona サンプルを起動すると、次のエラー メッセージが発生する。 `Starting Loading jDriver/Oracle ..... Trouble while executing example java.sql.SQLException: System.loadLibrary(weblogicoci37) threw java.lang.UnsatisfiedLinkError: /space/bt/weblogic/load4/bea/wlserver6.1/lib/hpux11/oci817_8/libweblogicoci37.sl: specified library, or one of its dependencies, does not exist.` `at weblogic.jdbc.oci.Driver.loadLibraryIfNeeded (Driver.java:243 at weblogic.jdbc.oci. Driver. connect (Driver.java:79)at examples.dbkona.query. main(query.java:118)` この問題は、jDriver が `ORACLE_HOME/lib` ディレクトリに存在しなければならない `libclntsh.so.8.0` ファイルを検索するが、`ORACLE_HOME` が定義されていないためにこのファイルを発見できないことが原因で発生する。サンプルの説明に、以下の記述が抜けている。 ■ `ORACLE_HOME` 環境変数が定義されていることを確認する。 ■ `ORACLE_HOME/lib` ディレクトリに `libclntsh.so.8.0` ファイルがあることを確認する。 ■ `LD_LIBRARY_PATH` に `ORACLE_HOME/lib` ディレクトリが含まれていることを確認する。 dbkona サンプルを実行するには、これらの手順が必要である。

変更要求番号	説明
CR127253	examples.security.acl サンプルで、ユーザが適切なコマンド ライン オプションを指定していない場合にエラーを捕捉できない。代わりに、スタック トレースが表示される。 examples.security.acl サンプルのパッケージの要約に、以下のように記述されている。 「ユーザおよびパスワードは joeuser のユーザ名およびパスワードです。-user および -pass コマンド ライン オプションが指定されていないと、JNDI の Initial_Context ではデフォルトによりユーザ「guest」、パスワード「guest」が指定されます。その場合、aclexample ACL がユーザ「guest」に対してパーミッションを割り当てないため、Altclient クライアントに障害が発生します」 これらの指示に従わずにこのサンプルを実行すると、以下のようなエラー スタック トレースが表示される。 C:\610sp6\wlserver6.1\samples\examples\security\acl>java -Dweblogic.security.SSL .ignoreHostnameVerification=true examples.security.acl.AltClient t3s://localhost :7002 -sslCert demokey.pem;democert.pem -servername weblogic.bea.com javax.naming.AuthenticationException. Root exception is java.lang.SecurityException: Authentication for user null denied in realm weblogic Start server side stack trace: java.lang.SecurityException: Authentication for user null denied in realm weblog ic at weblogic.security.acl.Realm.authenticate(Realm.java:212) at weblogic.security.acl.Realm.getAuthenticatedName(Realm.java:233) at weblogic.security.acl.internal.Security.authenticate(Security.java:17 1) at weblogic.common.internal.RMIBootServiceImpl.authenticate(RMIBootServi ceImpl.java:47) at weblogic.common.internal.RMIBootServiceImpl_WLSkel.invoke(Unknown Sou rce) at weblogic.rmi.internal.BasicServerRef.invoke(BasicServerRef.java:360) at....

インストーラに関する確認済みの問題

変更要求番号	説明

039773	インストーラを実行するのに十分なスペースが /temp にない場合に、次のような紛らわしいエラー メッセージが表示される。
	`Preparing to install...`
	`The included VM could not be extracted. Please try to download the installer again and make sure that you download using 'binary' mode. Please do not attempt to install this currently downloaded copy.`
	デフォルトでは、インストーラはすべてのインストーラ ファイルを /temp ディレクトリに復元しようとする。このディレクトリに十分な一時スペースがない場合は、インストールできないので、このメッセージが表示される。/temp にはインストーラ ファイルを格納できるスペースを確保する必要がある。

040353	共有ライブラリに実行パーミッションが与えられないため、HP-UX マシンと Solaris マシン上でインストールに関する問題が発生することがある。WebLogic Server のインストーラによるファイル パーミッションの変更は、標準の属性を強制されたインストーラによって元に戻される。この問題を避けるには、(lib/hpux11 と lib/solaris の下の) すべての共有ライブラリのオブジェクトに対して chmod +x コマンドを実行する。

043357	Solaris 2.6 または 2.7 UNIX システム上で次のエラー メッセージが表示された場合、
	`UnsatisfiedLinkError in AWT applications (libmawt.so: open failed: No such file for directory).`
	`The policytool will not open. The Java Plug-in Control Panel will not open.`
	通常は、LD_LIBRARY_PATH を編集することでこの問題を回避できる。次に例を示す。
	`setenv LD_LIBRARY_PATH`
	`/usr/j2se/jre/lib/i386/motif12:$LD_LIBRARY_PATH`
	ただし、libmawt.so ファイルに問題がある場合、この問題を修正できないことがあるので、Sun からパッチを入手する必要がある。Sun のパッチ情報については、http://java.sun.com//j2se/1.3/install-solaris-patches.html を参照。

変更要求番号	説明
053469、055598	Administration Console を使用する場合にユーザおよびグループにパーミッションを付与するには、`acl.modify.weblogic.admin=Administrators` という行を `filerealm.properties` に追加する。この ACL は、WebLogic Server 6.0、6.0 サービス パック 1、および 6.0 サービス パック 2 に存在した。
054058	サポートされていないブラウザでは、`beaexec.exe` を実行すると WebLogic Server がクラッシュする場合がある。

087287	Solaris 上にインストールする場合に `$DISPLAY` が設定されていないと、インストーラにより以下の例外が送出される。
	`Invocation of this Java Application has caused an InvocationTargetException.`
	`This application will now exit.`
	この問題を回避するには、`$DISPLAY` にホスト マシンの IP アドレスを設定する。

091420	WebLogic Server 6.1 SP04 では、Unix マシンに対して [Post-Bind UID を有効化] オプションが指定されていると、WLS が起動しなかった。(この属性は Administration Console の [マシン] ノードでコンフィグレーションできる。このオプションの目的は、特権付きのすべての起動アクションが行われた後で、Unix マシンを実行するサーバ インスタンスが稼働時に使用する Unix UID を返すことである。 この問題は、WLS 6.1 SP05 では解決されている。 6-129 ページの「CR091420」を参照。
102815	Install Anywhere インストーラを使用する場合、WebLogic Server 6.1 を Windows 2000 の互換性モードでインストールする必要がある。詳細については、『Installing on Windows XP』を参照。

JCA に関する確認済みの問題

変更要求番号	説明
049344	現時点の BEA WebLogic J2EE コネクタ アーキテクチャ実装では、J2EE コネクタ アーキテクチャ仕様バージョン 1.0 最終草案 2 のセクション 6.8 (「Scenarios: Local Transaction Management」) で規定されているエラーの検出機能を提供していない。

JDBC と jDrivers に関する確認済みの問題

変更要求番号	説明
033423	`QueryDataSet` と OCI805_8 ドライバで dbKona に関する問題が発生する。この問題は Windows NT 上では修正されたが、HPUX および Solaris では未解決。
035301	Sybase 接続プールで RMI ドライバを使用すると、問題が発生する。
035530	jDriver で OS 認証の有効化と Oracle 8.1.6/oci8 の使用に関して発生していた問題は NT と HP-UX では修正されたが、Solaris ではまだ解決されていない。この問題を報告するエラー メッセージは、「`OS level authentication is not currently supported due to a defect in OCI 8 libraries`」のように表示される。
040413	クラスタが Solaris マシン上でハングする場合がある。この原因は、`SIGALRM` と `SIGLWP` の間の `libthread` デッドロックと考えられる。デッドロックは、サーバが JDBC 接続をしようとしたときに発生する。この問題を避けるには、Administration Console を使って、`JDBCConnectionPool` の `InitialCapacity` 属性を `MaxCapacity` 属性と同じ値に設定する。 以前は同様の問題が、Solaris 2.6 の Sun Microsystems Bug (ID 4141709 SIGALRM と SIGLWP 間の libthread デッドロック) としてロギングされていた。現在、Solaris 2.7 に関してはこの問題が発生していない。Solaris 2.6 については、この問題のパッチ 105569-16 が用意されている。

変更要求番号	説明
043224	Oracle では、ThinDriver によって独自のクラス Array (`oracle.sql.Array`) をサポートしている。これらのクラスは、WebLogic Server JDBC で接続を取得するアプリケーションで使おうとした場合に機能しない。
048219	フラッシュしても RMI Clob 文字の書き込みが「コミット」されないように見える。フラッシュに問題が発生するのは、以下の書き込みメソッドを使用している場合のみと考えられる。 ■ `write(char[] cbuf)` ■ `write(char[] cbuf, int off, int len)` ■ `write(String str)` ■ `write(String str, int off, int len)` 次のメソッドでは正常に機能する。 ■ `write(int c)` Clob と共に Ascii ストリームを使用しているときにフラッシュしても問題はない。書き込みをコミットするには、ストリームを明示的にクローズする必要がある。書き込みストリームをクローズする準備が整う前に Clob から読み直そうとする場合にのみ、問題が発生することがある。
049519	Oracle の Thin ドライバのバグが原因で、次のメソッドは java.sql.PreparedStatement API に準拠していない。 `PreparedStatement.setTimeStamp(int param index, Timestamp x, Calendar cal)` タイムスタンプ値が、Calendar オブジェクトで指定された TimeZone に従って調整されない。デフォルトの TimeZone が常に使用される。 このバグは、Thin ドライバの今後のリリースで Oracle によって修正される可能性がある。
127348	WebLogic Server 6.1 SP6 に付属する Oracle 8.1.7 Thin ドライバを使用している場合に、以下の例外がときおり表示されることがある。 `java.sql.SQLException: Io exception: Protocol violation` これは、ドライバのエラーにより生じているおそれがある。この例外は Oracle から提供される以降のバージョンのドライバでは発生しない。WebLogic Server 6.1 SP6 に含まれるドライバのバージョンは 8.1.7.0.0。Oracle から現在提供されているバージョンは 8.1.7.1.0 である。 **回避策：**Oracle から新しいバージョンのドライバを入手して、そのドライバ (`classes12.zip`) を `weblogic.jar` の前のクラスパスに追加する。

JMS に関する確認済みの問題

変更要求番号	説明
037683	Administration Console の [JMS トピック] コンフィグレーション タブで、[マルチキャスト アドレス] の有効な値として任意のアルファベットを入力できる。インターネット アドレスのフォーマットに違反して入力される値の有効性を検証する必要がある。
049229	サポートされていない JDK 1.3.0 に対応したサーバを起動すると、JMS の問題が発生することがある。この問題は、JMS ストアに既存のメッセージがある場合にサーバを再起動すると発生し、それらのメッセージが無視される。サーバに付属している JDK 1.3.1 の使用を推奨する。
083538	マルチサーバのコンフィグレーションでは、あるサーバ上のアプリケーションまたは EJB は、別のサーバ上の `TxDataSource` をルックアップして接続を取得することはできない。アプリケーションまたは EJB は、同じサーバ上の `TxDataSource` を使用する必要がある。これはシステムでの制限事項である。

JSP に関する確認済みの問題

変更要求番号	説明
049854	ejb2 JSP ツールで EJB 2.0 ローカル インタフェースがサポートされない。
075671	JSP で `FileInputStream` または `ArrayList` などの Java クラスを使用する場合に、これらのクラスを JSP にインポートしていない場合にも `NoClassDefFoundError` が送出されない。

変更要求番号	説明
CR127701	JDK 1.3.1 および 1.4.1 に、確認済みのセキュリティの脆弱性がある。この脆弱性により、不正な XML または XSLT を解析しようとすると JVM がクラッシュし、Web サーバまたはアプリケーション サーバにおいてサービス拒否を招くおそれがあった。Sun Microsystems から提供されているパッチをインストールして、この確認済みのセキュリティの脆弱性を解決することを推奨する。 パッチを適用すると、さらに綿密な JSP 解析が実行される。その結果、パッチのインストール後には、JDK 1.3.1 および 1.4.1 ではコンパイルされていたコードがコンパイルされなくなる。Petstore サンプル アプリケーションにおいて、コンパイル エラーを回避するために JSP をアップグレードする方法については、http://edocs.beasys.co.jp/e-docs/wls/docs81/upgrade/upgrade6xto81.html#1062978 の『WebLogic Server 8.1 へのアップグレード』を参照。

JVM に関する確認済みの問題

変更要求番号	説明
CR099314	http://developer.java.sun.com/developer/bugParade/bugs/4755829.html で説明されているように、JDK 1.3.1_04 と nohup コマンドには問題がある。 nohup コマンドを実行すると失敗する。nohup の実行に関係なく、親のシェルが終了すると、WebLogic Server プロセスは終了する。この問題は ksh (Solaris 8 のデフォルトのシェル) で確認されており、他のシェルでも存在する可能性がある。bash シェルではこの問題は発生していない。 **回避策：** ■ 「(nohup startWeblogic &)」の代わりに「startWeblogic >output 2>&1 &」を使用する。

変更要求番号	説明
CR100564	WebLogic Server 6.1 SP04 と WebLogic Server 8.1 の間に相互運用性の問題があるため、WebLogic Server 6.1 SP04 上の RMI クライアントからの呼び出しでは、WebLogic Server 8.1 上の RMI オブジェクトによって送出された正しい例外を取得できない。このエラーは、WebLogic Server 6.1 SP04 では次のようになる。 `<Mar 8, 2003 12:36:35 PM PST> <Error> <NT Performance Pack> <failure in processSockets() - GetData: 'weblogic.socket.NTSocketMuxer$GetData - native pointer: '272329000', numBytes: '0'' java.lang.ClassCastException: weblogic.iiop. LocateRequestMessage at weblogic.iiop.EndPoint Impl.cleanupPendingResponses(EndPointImpl.java:659) at weblogic.iiop.ConnectionManager.got ExceptionReceiving(ConnectionManager.java:273) at weblogic.iiop.MuxableSocketIIOP.endOfStream (MuxableSocketIIOP.java:681) at weblogic.socket. NTSocketMuxer.processSockets(NTSocketMuxer.java:657) at weblogic.socket.SocketReaderRequest. execute(SocketReaderRequest.java:24) at weblogic. kernel.ExecuteThread.execute(ExecuteThread.java:139) at weblogic.kernel.ExecuteThread.run (ExecuteThread.java:120) > <Mar 8, 2003 12:41:35 PM PST> <Error> <NT Performance Pack> <failure in processSockets() - GetData: 'weblogic.socket. NTSocketMuxer$GetData - native pointer: '272089304', numBytes: '0'' java.lang.ClassCastException: weblogic.iiop. LocateRequestMessage at weblogic. iiop.EndPointImpl.cleanupPendingResponses(EndPointImpl.java:65 9) at weblogic.iiop.Connection Manager.gotExceptionReceiving(ConnectionManager.java:273) at weblogic.iiop.MuxableSocketIIOP. endOfStream(MuxableSocketIIOP.java:681) at weblogic.socket.NTSocketMuxer.processSockets(NTSocketMuxer.jav a:657) at weblogic.socket.Socket ReaderRequest.execute(SocketReaderRequest.java:24) at weblogic.kernel.ExecuteThread.execute (ExecuteThread.java:139) at weblogic.kernel. ExecuteThread.run(ExecuteThread.java:120)` `</failure></NT Performance Pack></Error></Mar 8, 2003 12:41:35 PM PST></failure></NT Performance Pack></Error></Mar 8, 2003 12:36:35 PM PST>`

RMI に関する確認済みの問題

変更要求番号	説明
CR100713	WebLogic Server 6.1 SP04 以前のバージョンは、WebLogic Server 8.1 と相互運用できない。6.1 と 8.1 のサーバインスタンス間のトランザクションは、6.1 のサーバインスタンスにおける `ArrayIndexOutOfBoundsException` でタイムアウトする。 6.1 のサーバインスタンスが 8.1 サーバインスタンスのコーディネータ リモート オブジェクトのルックアップを実行する際の RJVM/RMI に問題がある。6.1 サーバが `ack` を試みると、操作が失敗し、8.1 のコーディネータがトランザクションを解決できずに、コミット タイムアウト例外をクライアントに対して送出する。 たとえば、クライアントとリモート オブジェクト起動クラスが、8.1 と 6.1 のサーバにデプロイされているものとする。クライアントが 8.1 サーバ上のリモート オブジェクトをルックアップし、6.1 サーバの URL を指定してリモート メソッドを呼び出す。8.1 のメソッドは、6.1 サーバ上のリモート オブジェクトをルックアップし、8.1 サーバの URL でメソッドを呼び出す。6.1 のリモート メソッドは、8.1 サーバ上のリモート オブジェクトをルックアップして、メソッドを呼び出す。このとき、JVMID をブートストラップして送信すると、問題が発生する。リモート ピアがどれかを通知する手段がない。 この問題は、WebLogic Server 6.1 SP05 で解決される。

RMI-IIOP に関する確認済みの問題

変更要求番号	説明
CR126991	WebLogic Server 6.1 SP05 と WebLogic Server 6.1 SP06 の間に相互運用性の問題があることから、`ClassCastException` が送出されることがある。このエラーは `replaceObject` の実行中に `IIOPRemoteRef` を IOR にキャストしたために発生する。WebLogic Server 6.1 SP05 用のパッチを適用できる。

セキュリティに関する確認済みの問題

変更要求番号	説明
048159	ログ監査プロバイダの使用中は、監査が有効になっていることを通知するメッセージが WebLogic ログ ファイルに送信されない。
051215	WebLogic Server 起動プロセス中のシステム パスワードの入力時に、CTRL/C シーケンスを使用して WebLogic Server の起動を中断すると、ターミナル ウィンドウが非エコー モードになり、入力文字がエコーされない。 可能であれば、システム パスワードを入力し終えるまで、WebLogic Server の起動の中断を待機する。システム パスワードの入力時に CTRL/C シーケンスを使用する場合には、ターミナル ウィンドウをリセットするか、またはそのターミナル ウィンドウを閉じて別のターミナル ウィンドウを起動する。
051315	WebLogic Server のノード マネージャでは、Certificate Request Generator サーブレットによって作成されたプライベート キーを処理できない。この問題を解決するために、`wlkeytool` コンバータ ツールが提供されている。 `wlkeytool` コンバータ ツールは次のように使用する。 `wlkeytool` *`inputkey.pem outputkey.pem`* このコードの説明は以下のとおり。 *`inputkey.pem`* は、Certificate Request Generator サーブレットによって作成されたプライベート キーの名前。 *`outputkey.pem`* は、変換されたプライベート キーの名前。 プライベート キーがパスワードで保護されている場合 (つまり、Certificate Request Generator サーブレットの [Private Key Password] フィールドでパスワードを指定した場合) は、`wlkeytool` コンバータ ツールによってパスワードが要求される。プライベート キーがパスワードで保護されていない場合は、〔Enter〕を押す。 変換されたプライベート キーのパスワードが `wlkeytool` コンバータ ツールによって要求される。`wlkeytool` コンバータ ツールでは保護されていないプライベート キーは作成されないので、パスワードは必ず入力する。 デプロイ対象の WebLogic Server では、`wlkeytool` コンバータ ツールによって変換されたプライベート キーがサポートされていないので、Certificate Request Generator サーブレットによって作成されたプライベート キーを保持しておき、デプロイ対象の WebLogic Server ではそれを使用する必要がある。

変更要求番号	説明
053390	キャッシング レルムを有効にして RDBMSRealm を使用する場合で、DatabasePassword が config.xml ファイルの RDBMSRealm タグで指定されていない場合、サーバはエラー「<wlsstartuperror1>」で停止し、起動しない。 **回避策：**config.xml ファイルを開き、RDBMSRealm タグにクリア テキストで DatabasePassword を追加する。
053409	WebLogic Server で Java セキュリティ マネージャを使用するには、WebLogic Server の起動時に -Djava.security.manager property プロパティを指定する。 組み込みの起動スクリプトではポリシー ファイルが指定されるが、セキュリティ マネージャは指定されない。したがって、結局ポリシー ファイルは無視されてしまう。ポリシー ファイルが有効になるよう、セキュリティ マネージャを設定する必要がある。
078619	グループ メンバシップのキャッシングが、GroupMembershipCacheTTL で設定されている値を使用していない。代わりに、GroupCacheTTLPositive で設定されている値を使用している。
078894	FlatGroup.clearCache() を呼び出してキャッシュを更新すると、isMember の 2 回目の呼び出しで false が返る。isMember がまだ古いキャッシュ データを使用している。
087225	WebLogic Server 6.1 SP04 以降の LDAP レルム V2 では、多数のグループ (300 より多い) に対して getGroups() を実行したときに、Open LDAP サーバに関して問題がある。この問題の原因は、Open LDAP に存在するキャッシングのバグである。 **回避策：**getGroups() メソッドを使って Open LDAP サーバからグループを取得するときは、取得するグループが 300 を超えないようにする。

サーバに関する確認済みの問題

変更要求番号	説明
039575	サーブレットが同じサーバに対して HTTP 接続を行うと、ロード時にデッドロックが発生する場合がある。この問題を避けるには、DTD を返すサーブレットで別の実行キューを使用する。

042556	`weblogic.Admin` コマンドがユーザおよびパスワード オプションを要求する。次に例を示す。
	`java weblogic.Admin -username system -password gumby1234`
	ユーザ名「`system`」がほとんどの機能 (VERSION など) で要求される。`-username system` を指定しない場合、次のエラーが発生する。
	`Exception in thread "main" java.lang.SecurityException: Authentication for user system denied in realm weblogic <<no stack trace available>>`

051980	HotSpot VM 1.3 の使用中に、スレッド ダンプを行うと JVM が終了し、サーバがクラッシュする。
052300	アプリケーションを applications ディレクトリにない .ear ファイルとしてデプロイする場合、および ServletReloadCheckSecs が -1 に設定されている場合には、Call-by-Reference を使用できない。
052406	管理対象サーバを立て続けに起動しようとすると、問題が発生する。この問題は、起動間隔を空けることによって修正できる。
052543	RMI-IIOP を使ってサーバ間で複雑なオブジェクトを受け渡しする場合、初期コンテキストを取得するのに `com.sun.jndi.cosnaming.CNCtxFactory` を使用すると問題が多いので、代わりに `weblogic.jndi.WLInitialContextFactory` を使用することが推奨される。
053328	HP の使用中にメモリ不足エラーが発生した場合には、`max_thread_proc` というカーネル パラメータを 1024 に増やすようにする。
053330	SNMP エージェントでは、SNMP 管理システムとの通信に WebLogic のオブジェクト識別子 (OID) が使用される。バージョン 6.1 の WebLogic OID は バージョン 5.1 の WebLogic OID とは異なっている。バージョン 5.1 では .1.3.6.1.4.140.600 だったが、バージョン 6.1 では .1.3.6.1.4.140.625。

変更要求番号	説明
054504	JVM で -hotspot オプションを使用しているときに、Windows 2000 および NT で問題が発生する。この問題の原因は Sun のバグ (http://developer.java.sun.com/developer/bugParade/bugs/4479571.html) にあり、サーバの断続的なクラッシュが発生する。この問題を避けるには、次の設定を試してみる。 -XX:MaxPermSize=128m このオプションを使用すると、クラスの表現に HotSpot で使用される領域のサイズを増やすことができる。
057008	NTSocketMuxer getCPUCount() で不適当な値が返される。インストールされる Microsoft のサービス パックでは、すべての CPU が認識されない。
058868	HotSpot 仮想マシンのエラーにより、Solaris 上で動作する WebLogic Server は、信号 11 によって突然クラッシュするおそれがある。
059018、063937、063939、063965	デフォルトでは、WebLogic Server 6.1 サービス パック 2 のクライアントは、WebLogic Server 6.1 GA または 6.1 サービス パック 1 のサーバと相互運用**できない**。このことは、6.1 SP2 クライアントだけでなく、6.1 GA サーバまたは 6.1 SP1 サーバに対するクライアントとして機能する 6.1 SP2 サーバにも当てはまる。 SP2 クライアントが 6.1 GA サーバまたは 6.1 SP1 サーバと対話するためには、WebLogic Server 6.1 サービス パック 2 の起動スクリプトに次のスイッチを追加する必要がある。 -Dweblogic.61compat=true
065410	Sun のバグのため、ロードするクラスの数が多すぎると、WebLogic Server の起動時に java lang.OutOfMemory エラーが発生する場合がある。 回避策 : WebLogic Server を初期化する際に、次の構文を使って、JVM のオプション -XXMaxPermSize の値を大きくする。 java -XX:MaxPermSize=<value>K <value> はキロバイト単位の適当な値である。JVM が使用するデフォルトの最大値は 32 MB である。 この問題は Sun に報告済み。

変更要求番号	説明
078289	Java Plug-in 1.3.1_02 または 1.3.1_03 を使用するブラウザがアプレットを含むページを更新すると、JVM は、アプレットに対して `destroy()` と `init()` を同時に呼び出す。呼び出しが同時に行われるので、アプレットが初期化を試みる前に正しく閉じない場合がある。 回避策としては、アプレットの `init()` メソッドの中で `thread.sleep()` を使用して、1000 ミリ秒ほど初期化を延期する。これにより、アプレットが再初期化を試みる前に、`destroy()` の呼び出しが必要なクリーンアップを確実に終了できる。

サーブレットと Web アプリケーションに関する確認済みの問題

変更要求番号	説明
041528	サービス パックを適用していない WebLogic Server 6.0 の最初のリリースで生成されたクッキーが、WebLogic Server 6.1 で認識されない。
046536	HTTPS が仮想ホスティングでは機能しない。
046537	仮想ホストでは、独自のセキュリティ レルムを定義する必要がある。
048678	以前は、Web アプリケーションからユーザをログアウトさせる場合に、`session.invalidate()` を使用できた。現時点では、このメソッドは、ユーザが 1 つの Web アプリケーションにのみログインしている場合にしか機能しない。ユーザが複数の Web アプリケーションにログインすると、`session.invalidate()` ではユーザをログアウトできなくなる。これは、確認済みの問題というよりも、シングル サインオンを実装するための設計の変更である。サーバでは、サーブレット 2.2 仕様で指定されているとおりに Web アプリケーション レベルではなく、Web サーバ レベルでユーザが追跡される。複数の Web アプリケーションにログインしているユーザをログアウトさせる場合には、次のように処理する。`weblogic.servlet.security.ServletAuthentication.logout(request);`
050811	フィルタ クラスを修正する場合は、Web アプリケーションを再デプロイして変更を適用する必要がある。
051311	Web アプリケーションの名前 (コンテキスト パス) ではスペースを使用できない。

変更要求番号	説明
051329	次の問題は単なる情報の提供であり、修正または変更されることはない (これが目的の機能であるため) 。 WebLogic Server の config.xml ファイルでは、アーカイブ Web アプリケーションの Path 属性はアーカイブ ファイルの名前自体ではなくアーカイブ ファイルのパスのみに設定する必要がある。一方、Web アプリケーション コンポーネントの URI 属性は、アーカイブ ファイルの名前のみに設定する必要がある。次に例を示す。 <Application Name=foo, Path=/apps> <WebAppComponent Name=foo,URI=foo.war> </Application> 下位互換性を実現するため、WebLogic Server 6.1 では Path 属性でアプリケーションのパスとアーカイブ ファイル名の両方を指定する config.xml が許容される (たとえば Path=/apps/foo.war) 。ただし、XML ファイルはパス (/apps) のみを指定するために上書きされる。
052686	WebLogic Server 6.1 に付属の Web アプリケーション デプロイメント記述子エディタでは、サーブレット 2.3 仕様の最終草案バージョン 1 がサポートされる。このため、最終草案バージョン 2 で導入された web.xml の <ejb-local-ref> 要素をコンフィグレーションするための手段がエディタにはない。なおかつ、エディタでは既存の Web アプリケーション デプロイメント記述子に存在する可能性のある <ejb-local-ref> が削除される。
CR088485	WebLogic Server 6.1 SP03 に、正しくないバージョンの Jakarta 正規表現マッチング ライブラリが含まれていた。jakarta-oro-2.0.6.jar ではなく、jakarta-oro-2.jar が付属していた。この結果、顧客の予期しない結果が発生していた。 回避策 : WebLogic Server のクラスパスで、weblogic.jar の前に、正しいバージョンの Jakarta (jakarta-oro-2.0.6.jar) を指定する。

変更要求番号	説明
089868	WebLogic Server 6.1 サービス パック 3 で行われた重要なパフォーマンスの改善 (6-235 ページの「083654」を参照) により、マルチバイトのヘッダで問題が発生するようになった。以前は、未加工のバイトとして HTTP ヘッダを送信するのがデフォルトの動作であった。HTTP の仕様ではヘッダは ASCII のシングル バイト文字でなければならないと規定されているので、以前のこの動作は誤りであった。(この改善により WebLogic Servers 6.1 サービス パック 3 以下と WebLogic Servers 6.1 サービス パック 4 の間の相互運用性に問題が発生する可能性があることを、BEA は認識している)。 SP03 と SP04 の場合はヘッダが壊れる。回避策は以下のとおりである。 ■ BEA Customer Support から入手できる CR089868_610sp3.jar パッチまたは CR089868_610sp4.jar パッチをインストールする。 ■ そして、config.xml の <WebServer> 要素で、UseHeaderEncoding="true" と設定する。「true」に設定すると、サーバは、Content-Type の文字セットを使って、Content-Disposition ヘッダをエンコードする。「true」に設定しないと、デフォルトにより、サーバはプラットフォームのデフォルトのエンコーディングを使用する。結果としてエンコーディングが 8 ビット文字セットになる場合、サーバはどのエンコーディングも使用しないことに注意すること。 SP05 では、上記の 2 番目の項目で示したように UseHeaderEncoding="true" と設定するだけで、この問題に対処できる。

システム管理に関する確認済みの問題

変更要求番号	説明
034177	Administration Console では、アプリケーション ステータスの適切な属性値が設定されない。
034955	デプロイメント用にスキャンするアプリケーション ディレクトリ リストが永続的でない。複数のアプリケーション ディレクトリがサポートされていない。
035670	MBean 名を再設定すると、コンフィグレーションが適切に保存されない。この問題を避けるには、次の操作を行う必要がある。 1. Administration Console で MBean を複製する。 2. クローンの名前を変更する。 3. オリジナルの MBean を削除する。
036201	Administration Console で変更を行った後に、WebLogic Server を再起動できず、次のようなメッセージが表示されることがある。`Fatal initialization exception Throwable: weblogic.server.ServiceFailureException: Initializing JNDI: javax.naming.ServiceUnavailableException [Root exception is java.net.UnknownHostException: Unknown protocol: 'Protocol: 'unknown'']`

変更要求番号	説明

038589	管理対象サーバが HTTPS を使って管理サーバと通信している場合、管理対象サーバの検出が機能しない。たとえば、管理サーバが `-Dweblogic.management.discover=true` コマンド ライン引数を使用して再起動された場合、かつ実行中の管理対象サーバが `-Dweblogic.admin.host=https://<adminhost>:<adminport>` 引数を使用して起動されていた場合、管理対象サーバの検出が機能しない。
	回避策 : 管理サーバとの HTTPS 接続を使用して管理対象サーバを起動する場合に、管理サーバを `-Dweblogic.management.discover=true` 引数を使用して再起動しないようにする。
	代わりに、次の手順に従う。
	1. discover フラグを付けずに管理サーバを再起動する。
	2. 管理サーバが起動したら、次のコマンドを実行して管理サーバに強制的に検出させる。
	`java weblogic.Admin -url <ADMINHOST>:<ADMINPORT> -username system -password <SYSTEM-PASSWORD>`
	`INVOKE -mbean "<YOURDOMAIN>:Type=Domain,Name=<YOURDOMAIN>" -method discoverManagedServers`

039033	WebLogic Server で、存在しないアプリケーションをアンデプロイするエラーが発生する。
039311	複製されている管理対象サーバを起動すると、`ConfigurationException` が送出される。この問題を避けるには、複製するのではなくサーバを作成する。
039532	`.ear` ファイルをアプリケーション ディレクトリの下に展開した場合に、Administration Console を使用しないとこのファイルをアンデプロイできない。また、アプリケーションがアンデプロイされた場合に、Administration Console を使用しないとこのアプリケーションを再デプロイできない。

変更要求番号	説明
040034	すでに管理サーバとして実行中のサーバを管理対象サーバとして起動しようとすると、次のような紛らわしいエラーが発生する。 `<Dec 8, 2000 7:44:02 PM PST> <Error> <Configuration Management> <Error connecting to admin server and initializing admin home: admin URL: t3://localhost:7001 java.lang.NullPointerException` 回避策：この使い方は適切ではない。同じサーバを管理サーバとしても管理対象サーバとしても起動することはできない。この問題を避けるには、新しいサーバを管理対象サーバとして定義して、起動時にその名前を `-Dweblogic.Name` で指定する。
040124	管理サーバの実行中に、既存の `.jar` または `.ear` ファイルをアプリケーション ディレクトリにコピーしてもアプリケーションが再デプロイされない。次のエラーがサーバ ログに記録される。 `javax.naming.NameAlreadyBoundException: Can't rebind anything but a replica-aware stub to a name that is currently bound to a replica-aware stub; remaining name '' <Dec 12, 2000 12:09:37 PM PST> <Error> <J2EE> <Error deploying application <YOUR-EJB-NAME>: Could not deploy: '<YOUR-FILE-NAME>': JNDI name in use` `guest` ユーザまたは `everyone` グループに、`weblogic.jndi` に対するルックアップ パーミッションが付与されていることを確認する。 `fileRealm.properties` ファイルの記述に `acl.lookup.weblogic.jndi=everyone` が含まれている必要がある。
041290	ユーザ名プロパティを `weblogic.Admin` または `weblogic.Server` コマンド ラインから設定する場合、そのユーザ名は「`system`」とする必要がある。ただし、ユーザのスクリプトを壊さないために、ユーザ名の構文に従う。ユーザ名は、`config.xml` ファイルでは変更することができない。 `weblogic.management.password` プロパティの値はシステム パスワードでなければならない。
0422320、55217	管理サーバと同じ名前のドメインを使用することはできなくなった。ネームスペースの衝突によって非常に多くの問題が発生するため、この問題に対する対処は行われない。

変更要求番号	説明
048833	ServletRuntime MBean には、getName() メソッドがある。以前のリリースでは、このメソッドはサーブレットの名前を返した。これ以外のすべての MBean では、getName() メソッドは MBean の名前を返す。一貫性を保つため、WebLogic Server 6.1 では、ServletRuntime MBean の getName() メソッドも MBean の名前を返すように変更されている。サーブレットの名前を返す新しいメソッド getServletName() が ServletRuntime MBean に追加されている。
052657	アプリケーションがまだ実際に存在していないときに、config.xml でアプリケーションに対するコンフィグレーション属性を設定すると、Administration Console ではアプリケーションがデプロイされたものと誤って表示される。

077958	NodeManager を使用すると、次の例外が送出される場合がある。
	`<May 18, 2002 8:53:36 PM PDT> <Error>`
	`<NodeManager@localhost:5555>`
	`<SocketInputHandler: handshake failed 'FATAL`
	`Alert:HANDSHAKE_FAILURE - The`
	`handshake handler was unable to negotiate an acceptable set of security`
	`parameters.' on socket /172.17.24.148>`
	この例外の原因は、次のいずれかである。
	■ NodeManager を実行するために使われている JVM のバージョンを、WebLogic Server がサポートしていない。
	■ AIX プラットフォーム上で NodeManager が動作している場合、JSEE が衝突する。AIX プラットフォームで NodeManager が動作している場合は、『動作確認状況』の「サポート対象プラットフォームの一覧」で詳細を参照。

CR093687	WebLogic Server 6.1 SP03 以降では、起動クラスに対して LoadBeforeAppDeployments=true を指定すると、動的接続プールを作成できない。接続プールは作成されず、ハング状態になる。例外は報告されない。 LoadBeforeAppDeployments=false と指定すると、接続プールは正常に作成される。
CR097152	WebLogic Server 6.1 SP03 および SP4 では、管理サーバを再起動すると、次の例外が発生する場合がある。 `java.rmi.ConnectException: This RJVM has already been shutdown` WebLogic Server 6.1 SP05 では、この問題は再現しなかった。

ツールに関する確認済みの問題

変更要求番号	説明
063745	`weblogic.refresh` ツールのドキュメントには、ワイルドカード (`*.gif` など) を使ってファイルを指定できること、およびカンマ区切りのリストを使って複数のファイルを指定できることが記述されている。しかし、WebLogic Server サービス パック 2 の `weblogic.refresh` では、カンマ区切りのリストの中でワイルドカードを使うことはできない。

Web サービスに関する確認済みの問題

変更要求番号	説明
046807	Java クライアントの JAR ファイルで Java クライアント アプリケーションによって生成される例外メッセージがインターナショナライズされていない。
047435	RPC スタイルの Web サービスでは、EJB によって送出されるユーザ定義の例外が、Web サービス クライアント API を使用してサービスを呼び出すクライアント アプリケーションに伝播されない。それらの例外は、代わりに汎用 SOAPFault 例外として伝播される。
049966	EJB などの WebLogic Server 6.0 コンポーネントで WebLogic Server Web サービス クライアント API (バージョン 6.1) を使用すると、`java.rmi.ConnectException: No available router to destination` というエラーが発生する。この原因は、クライアント コードで指定している初期コンテキスト ファクトリではなく、6.0 JNDI URL ファクトリが呼び出されることにある。 この問題を避けるには、初期コンテキストの作成時に次のコードを追加することによって 6.0 JNDI URL ファクトリを除去する。 `h.put(Context.URL_PKG_PREFIXES, "");`
051917	メッセージ スタイルの Web サービスでは、対象となる JMS 送り先で発生する JMS 例外が、Web サービス クライアント API を使用してサービスを呼び出すクライアント アプリケーションに伝播されない。

変更要求番号	説明
055062	wsgen Ant タスクの clientjar 要素の path 属性は、path 属性が build.xml ファイルで指定されていない場合は、ドキュメントにあるとおりデフォルトで client.jar にならない。
055096	パッケージ名のない JavaBean は、パラメータとして WebLogic Web サービスに渡すことができない。
055596	相互運用性の問題。WebLogic Web サービスは、SOAP 1.1 の多重参照複合データ型をサポートしていない。
056287	相互運用性の問題。WebLogic Web サービスは、SOAP エンコーディングの xsd:timeInstant および xsd:dateTime をサポートしていない。
056452	相互運用性の問題。WebLogic Web サービスは、多次元配列データ型をサポートしていない。
056634	相互運用性の問題。WebLogic Web サービスは、SOAP-ENV:Header 属性をサポートしていない。
058175	相互運用性の問題。WebLogic Web サービスは、xsd:ur-type 配列データ型をサポートしていない。
058840	相互運用性の問題。WebLogic Web サービスのクライアント API は、戻り値の型を複数指定している WSDL にバインドできない。
059782	相互運用性の問題。WebLogic Web サービスのクライアント API は、<binding> セクションに <soap:header> 要素を含む WSDL を使用する Web サービスの呼び出しをサポートしていない。

059858	相互運用性の問題。WebLogic Web サービスは、次の SOAP リクエストの例で示すように、生成される SOAP リクエストにおいて 1999 XML スキーマの URL だけを使用する。
	`xmlns:xsi='http://www.w3.org/1999/XMLSchema-instance'`
	`xmlns:xsd='http://www.w3.org/1999/XMLSchema'`

061161	相互運用性の問題。WebLogic Web サービスのクライアント API は、<soap:body> 要素の use="literal" 属性をサポートしていない。
061335	相互運用性の問題。WebLogic Web サービスのクライアント API は、<message> 要素内の element 属性に <part> 要素を含む WSDL を使用する Web サービスの呼び出しをサポートしない。

変更要求番号	説明

061350	相互運用性の問題。WebLogic Web サービスのクライアント API は、デフォルトでは、修飾名を含む SOAP リクエストを生成する。このため、.NET のような他の一部の非 WLS Web サービスとの相互運用性が十分ではない。
	この問題を回避するには、クライアント アプリケーションに次のコードを追加する必要がある。
	`CodecFactory factory = CodecFactory.newInstance();`
	`SoapEncodingCodec codec = new SoapEncodingCodec();`
	`codec.writeQualifiedName( false );`
	`factory.register( codec );`

061572	相互運用性の問題。WebLogic Web サービスのクライアント API は、WSDL の `<soap:binding>` 要素の `style="document"` 属性をサポートしていない。
061606	`wsgen` Ant タスクを使ってメッセージスタイルの Web サービスをアセンブルすると、生成される WSDL にリテラル エンコーディングが追加される。そのため、必要がない場合でも、プログラムでは Web サービスを呼び出して CodecFactory に LiteralCodec を登録する必要がある。LiteralCodec を使用するには、クライアント コンピュータに `weblogic.jar` ファイルが存在していなければならない。
061929	相互運用性の問題。WebLogic Web サービスのクライアント API は、クライアント アプリケーションに次のコードが含まれていても、常に、生成する SOAP リクエストの中で修飾名を使用する。 `codec.writeQualifiedName( false );`
77911	WebLogic Server は、Java の組み込みデータ型 `byte[]` を `xsd:hexBinary` に正しくマップする。ただし、JAX-RPC の仕様では、`byte[]` は `xsd:base64Binary` にもマップされなければならないと規定されている。現在、WebLogic Server 6.1 SP3 の Web サービスには、このマッピングを指定する方法はない。
78062	WebLogic Web サービスは、パラメータまたは戻り値として配列および JavaBean を使用する .NET Web サービスと相互運用できない。
78530	Web サービスのサンプルが、Solaris オペレーティング システム上に構築されない。

WebLogic Tuxedo Connector に関する確認済みの問題

変更要求番号	説明
052737	WebLogic-Tuxedo Connector の XML コンフィグレーション ファイルでノード名を間違ってコンフィグレーションすると、システムがハングする。 次の手順に従って、システムを再起動する。 1. WebLogic Server を停止する。 2. WebLogic Tuxedo Connector の XML コンフィグレーション ファイルにあるノード名を確認する。 3. 有効なノード名で WebLogic Tuxedo Connector の XML コンフィグレーション ファイルを更新する。 4. WebLogic Server を再起動する。
077553	Tuxedo から WebLogic Server に整数配列オブジェクトを値で渡すと、`weblogic.utils.AssertionError` が送出される。 この問題は WebLogic Server 6.1 SP04 で解決された。「077553」を参照。
078774	tBridge セッションの接続が、Tuxedo /Q からのメッセージの取得を停止する場合がある。

変更要求番号	説明

079630	ユーザは普通、`setJMSCorrelationID(String)` メソッドを使って相関 ID を設定しようとする。このメソッドは、32 文字の文字列を受け取り、64 バイトの配列に変換する。JMS は、文字列を UTF-16BE として格納する。tBridge が Tuxedo から受け取る相関 ID は、32 バイトで表された 32 文字である。tBridge は、`setJMSCorrelationIDAsBytes(byte[])` メソッドを使って、JMS 受信キューに対するメッセージを設定する。このような 2 つの文字列は、ASCII では同じであっても、長さが異なるために比較しても一致しない。
	
	相関 ID を、Tuxedo から受信 JMS キューに返される ID と比較する必要がある場合は、相関 ID の 16 進数値をバイト配列に格納した後、`setJMSCorrelationIDAsBytes()` メソッドと `getJMSCorrelationIDAsBytes()` メソッドを使って ID を作成して、Tuxedo から受信 JMS キューに返される ID と比較する。
	
	例えば、相関 ID が「`1234567890ABCDEFGHIJKLMNOPQRSTUV`」という文字列の場合、次のようになる。
	
	`private byte[] coridbyte={0x31,0x32,0x33,0x34,0x35,0x36,0x37,`
	`0x38,0x39,0x30,0x41,0x42,0x43,0x44,`
	`0x45,0x46,0x47,0x48,0x49,0x4a,0x4b,`
	`0x4c,0x4d,0x4e,0x4f,0x50,0x51,0x52,`
	`0x53,0x54,0x55,0x56};`
	`msg.setJMSCorrelationIDAsBytes(coridbyte);`
	`corIDasBytes = msg.getJMSCorrelationIDAsBytes();`
	
	`corIDasBytes` には、Tuxedo から返される相関 ID と比較するための正しい値が設定される。

XML に関する確認済みの問題

変更要求番号	説明
050274	Administration Console を使用してパーサまたはエンティティの XML レジストリ エントリを追加するときに、パブリック ID を指定しないと、そのエントリにアクセスして編集できない。 この問題を避けるには、XML レジストリ エントリの作成時には常にパブリック ID のフィールドに値を入力する。
052985	組み込みの Xalan トランスフォーマ パッケージ `weblogic.apache.xalan` に、Xalan 互換性 API が含まれていない。これらのクラスは、Apache Web サイトから直接 Xalan 2.0.1 をダウンロードしていれば、`xalanj1compat.jar` ファイルに含まれている。 互換性 API の詳細については、`http://xml.apache.org/xalan-j/usagepatterns.html#compat` を参照。 これらのクラスが WebLogic Server に含まれていないので、Xalan 2.0.1 では非推奨となっている Xalan 1.X のコードがアプリケーションで使用されると、アプリケーションが正常に機能しなくなる。この問題を避けるには、非推奨の Xalan 1.x クラスを使用しないようにコードを変更するか、代わりに JAXP を使用するよう Xalan コードを書き換える。

ZAC に関する確認済みの問題

変更要求番号	説明
CR047534	ZAC Publish Wizard で設定される JRE Update Frequency プロパティは現在ではサポートされていないため、JRE のアップデートが必要な場合には、次のいずれかの方法で行う必要がある。 JRE がアプリケーションにパッケージ化されている場合、次の条件を満たすときにのみクライアントに対する再インストールが行われる。 ■ JRE 全体のインストール ディレクトリがクライアントから削除されていて、かつクライアント実行可能ファイルまたは ZAC のブートストラップ ルーチンが実行されている場合。または、 ■ 新しいバージョンの JRE のライブラリ パッケージが WebLogic Server にパブリッシュされていて、かつクライアントで (クライアント実行可能ファイルでなく) ブートストラップ ルーチンが実行されている場合。 サーバでは、パッケージのバージョニングは (バージョン番号がパッケージ名の一部を構成する場合を除き) 行われない。まだパブリッシュされていないパッケージを「前のバージョンに戻す」ことのみが可能である。

変更要求番号	説明

061699	WebLogic Server では、デフォルト Web アプリケーションの登録は自動的には行われない。ZAC はデフォルト Web アプリケーションを通してクラスを取得するので、ZAC を使用する場合は、デフォルト Web アプリケーションを明示的にコンフィグレーションする必要がある。
	デフォルト Web アプリケーションを明示的にコンフィグレーションしないと、ZAC は機能せず、サーバのログには次のようなメッセージが記録される。
	`<Nov 2, 2001 2:50:49 PM PST> <Error> <HTTP>`
	`<HttpServer(7908535,null default`
	`ctx,myserver) found no context for`
	`"/drp-exports/ZACHello/index.xml". This`
	`request does not match the context path for any installed web applications`
	`and there is no default web application configured.>`
	`<Nov 2, 2001 2:50:49 PM PST> <Error> <HTTP>`
	`<HttpServer(7908535,null default`
	`ctx,myserver) found no context for "/drp-publish/ZACHello/". This request`
	`does not match the context path for any installed web applications and there`
	`is no default web application configured.>`

6　解決済みの問題

以下の節では、WebLogic Server 6.1 のサービス パックで解決された問題について説明します。サービス パックは累積的であり、現行リリースのサービス パック 6 にはそれ以前にリリースされた WebLogic Server 6.1 用のサービス パックで行われたすべての修正が含まれます。

- WebLogic Server 6.1 サービス パック 6 のソリューション
- WebLogic Server 6.1 サービス パック 5 のソリューション
- WebLogic Server 6.1 サービス パック 4 のソリューション
- WebLogic Server 6.1 サービス パック 3 のソリューション
- WebLogic Server 6.1 サービス パック 2 のソリューション
- WebLogic Server 6.1 サービス パック 1 のソリューション
- WebLogic Server 6.1 リリースのソリューション

WebLogic Server 6.1 サービス パック 6 のソリューション

この節では、WebLogic Server 6.1 SP06 で解決された問題を示します。

- クラスローダ
- クラスタ
- コネクタ
- コンソール
- コア
- デプロイメント

- デプロイメント
- デプロイメント
- EJB
- JDBC
- jDriver
- JMS
- JNDI
- JSP
- JTA
- ノード マネージャ
- OA&M (操作と管理)
- プラグイン
- RMI
- RMI/IIOP
- セキュリティ
- サーブレット
- SNMP
- Web サービス
- WTC-ATMI
- XML

クラスローダ

変更要求番号	説明
CR101547	静的データの含まれるシングルトン クラスが格納されたエンタープライズ アプリケーションをデプロイするときにメモリ リークが発生していた。この問題は、シングルトン クラスを Web アプリケーションとしてデプロイしたときには発生しなかった。 この問題は、アプリケーション クラスローダの修正で解決した。
CR104513	WebLogic Server 6.1 サービス パック 2 で行われた CR069506 の修正により、1 つのマニフェスト ファインダが他のマニフェスト エントリを再帰的に参照する場合にクラスローダがファイルを検索できなかった。この問題は、このケースに対処するように `Classloader.getResources()` を修正することで解決した。
CR111924	リモート サーバの EJB からローカル サーバの EJB に送出されたユーザ定義例外を処理するときに、WebLogic Server から `ClassNotFoundException` が送出されていた。この問題は、ユーザ定義例外がローカル EJB のクラスローダに含まれている場合でも起こっていた。この問題は、例外を処理するソケット リーダーがアプリケーション クラスローダではなくシステム クラスローダを使用していたことが原因。 この問題は、コードを修正して解決された。
CR124348	プロキシ クラスがシステムのクラスパスに追加されていない限り、クライアント プログラムで `java.lang.reflect.Proxy` を使用して、WebLogic Server にデプロイされたプロキシ オブジェクトにアクセスすることができなかった。オブジェクトがシステムのクラスパスにない場合、クライアントは `ClassNotFoundException` を受け取っていた。 `resolveProxyClass()` メソッドが実装され、システム クラスローダだけでなくアプリケーション固有のクラスローダからもインタフェースがロードされるようになった。

クラスタ

変更要求番号	説明
CR103807	WebLogic Server 6.1 SP04 では、`weblogic.rmi.cluster.BasicReplicaList.add(BasicReplicaList.java:61)` で null ポインタ例外が送出されていた。この問題は、管理対象サーバが障害後に再起動されたときに発生していた。再起動時に、NPE が発生し、クラスタ内のすべてのサーバインスタンスを再起動しなければならなかった。このエラーはタイミングに関連しており、コードの修正で解決した。
CR104430	クライアントがクラスタ内の 1 番目と 2 番目のサーバ (クライアントのプライマリ サーバとセカンダリ サーバ) から 3 番目のサーバに切り替えるときに、クラスタ化されたサーバでセッションが失なわれる可能性があった。プロセスは最初の 2 つのサーバからセッションを削除する。クライアントがプライマリ サーバに戻ると、プライマリ サーバは 3 番目のサーバ上ではなく、セカンダリ サーバ上でセッションを探した。 3 番目のサーバ上でセッション情報からセッションを再作成し、プライマリおよびセカンダリ サーバからセッションを完全に削除するようにコードを修正して、問題を解決した。

変更要求番号	説明

CR107471	WebLogic Server 6.1 SP02 では、クラスタで HTTP トンネリングを使用した場合に、クライアントが次のメッセージを受け取っていた。
	`java weblogic.Admin -url http://colma:17683 -username system -password password PING 10`
	`<May 29, 2003 10:14:18 AM PDT> <Error> <RJVM> <000515> <execute failed java.net.ProtocolException: Tunneling result not OK, result: 'DEAD', id: '0'java.net.ProtocolException: Tunneling result not OK, result: 'DEAD', id: '0'`
	`at weblogic.rjvm.http.HTTPClientJVMConnection.receiveAndDispatch(HTTPClientJVMConnection.java:422)`
	`at weblogic.rjvm.http.HTTPClientJVMConnection.execute(HTTPClientJVMConnection.java:305)`
	`at weblogic.kernel.ExecuteThread.execute(ExecuteThread.java:213 at weblogic.kernel.ExecuteThread.run(ExecuteThread.java:189)> Failed to connect to <urldeleted> due to: <urldeleted>:Bootstrap to <urldeleted> failed. It is likely that the remote side declared peer gone on this JVM]`
	この問題は、プラグインが各トンネル リクエストを次のサーバにラウンドロビンしようとしたことが原因。リクエストが同じサーバに固定されていなかった。
	この問題は、jsessionid クッキーを設定し、それをプラグインに送り返すことで状態がクライアントで管理されるようにコードを修正して解決された。

変更要求番号	説明

CR108127	WebLogic Server 6.1 SP04 では、セカンダリ セッションを更新しようとしたときにエラーが発生していた。サーバ A、サーバ B、およびサーバ C という 3 つのクラスタ化されたサーバ インスタンスがあるとする。
	1) サーバ A でセッションを開始する (プライマリ A、セカンダリ C)。
	2) ロード バランシング ハードウェアがサーバ C にセッションを移行する (プライマリ C、セカンダリ B)。
	3) ロード バランシング ハードウェアがセッションをサーバ A に再び移行する。
	次のエラーがサーバ A で発生していた。
	`<Jun 4, 2003 4:24:56 PM PDT> <Debug> <Cluster> <Error updating secondary for 3535724191309537720 on 7312270160516507840S:<IPaddressdeleted>: [7005,7005,7002,7002,7005,7002,-1]:<IPaddressdeleted>:7005, <IPaddressdeleted>:7005,<IPaddressdeleted>:7005:mydomain:myserver3. Re-creating secondary.>`
	次のエラーがサーバ C で発生していた。
	`<Jun 4, 2003 4:24:56 PM PDT> <Info> <Cluster> <Lost 0 replication updates of object 3535724191309537720. Re-fetching secondary.>`
	この問題は、コードを変更して解決された。

CR112874	WebLogic Server 6.1 SP03 では、ステートフル セッション Bean のクライアントが、重みベースのロード バランシングでコンフィグレーションされたクラスタにデプロイされている Bean にアクセスし、1 番目と 2 番目の重みの管理対象サーバがその順序で強制終了された場合に、クライアントが次のメッセージを出していた。 `<Jul 22, 2003 1:32:56 AM IST> <Warning> <Kernel> <No replica in list has the expected weight. Reverting to round-robin>` 管理対象サーバが再起動すると、ロード バランシング アルゴリズムがラウンドロビンに切り替えられていた。 分析の結果、レプリカ リストは管理対象サーバがダウンしたときに更新されていたが、競合状態が原因で、`RichReplicaList` の最大の重みが適切にリセットされていなかったことが判明した。 レプリカ リストのサイズが変わったときに必ず `max weight` が再計算されるようにコード修正して、この問題は解決した。

変更要求番号	説明
CR116954	HTTPClusterServlet からの InitJVMID リクエストで web アプリケーション コンテナが 404 メッセージを返し、その結果としてログ ファイルがメッセージで一杯になっていた。 `####<Sep 6, 2002 4:56:43 PM EDT> <Debug> <HTTP> <petunia> <ms1petunia> <ExecuteThread: '13' for queue: 'default'> <kernel identity> <> <101147> <HttpServer(887891,null default ctx,ms1petunia) found no context for "/WLDummyInitJVMIDs".` この問題は、WebLogic Server に常にデプロイされている内部 Web アプリケーションに InitJVMID リクエストを送信することで解決した。

コネクタ

変更要求番号	説明
CR109463	トランザクションが 2 つのリソース アダプタへの接続を取得し、それらのアダプタが両方とも同じリソース マネージャを使用していた場合に、WebLogic Server が片方の接続だけクリーン アップしていた。接続は、トランザクションのコミット前に閉じられていた。この問題は、`javax.resource.spi.ResourceAllocationException` として表面化した。 この問題は、同じリソース マネージャを使用した 2 つめの接続をトランザクション マネージャが無視したことが原因だった。この問題は、コードを修正して解決された。
CR121418	新しい接続を割り当てるときに、コネクタ コンテナが記述子 MBean を使用してコンフィグレーション情報を取得していた。記述子 MBean は管理サーバ上にあるので、管理対象サーバは管理サーバにアクセスできないと新しい接続を割り当てることができない。管理サーバがダウンしている場合は、新しい接続を割り当てるプロセスから次の例外が送出されていた。 `weblogic.rmi.extensions.RemoteRuntimeException` コンテナは、アダプタのデプロイ後にコンフィグレーション情報をローカルに格納するように修正された。管理対象サーバは、新しい接続の割り当てで記述子 MBean ではなくこのローカルのコンフィグレーション情報を使用するようになった。
CR125555	リソース アダプタから MetaData 情報を取得するときに、WebLogic Server サービス パック 5 ではオブジェクトの使用前に戻り値で null がチェックされていなかった。これが原因で、`NullPointerException` と接続障害が発生する可能性があった。 `getMetaData()` の呼び出し後に戻り値で null がチェックされるようにコードが修正された。ManagedConnection.getMetaData() のアダプタ実装が null 値を返しても、WebLogic Server は `NullPointerException` を送出しなくなり、MetaData はログに記録されない。

コンソール

変更要求番号	説明

CR100006	ドメインの中に 2 つの管理サーバがあり、Administration Console を通じてそれらのサーバにアクセスしようとしたときに、WebLogic Server は次の NullPointerException を表示することがあった。
	`java.lang.NullPointerException`
	`at weblogic.management.console.helpers.UrlHelper.buildIconList(UrlHelper.java:149)`
	`at weblogic.management.console.helpers.UrlHelper.getIcon(UrlHelper.java:174)`
	`at weblogic.management.console.tags.SmartNavNodeTag.getIcon(SmartNavNodeTag.jacva:111)`
	`at weblogic.management.console.tags.SmartNavNodeTag.inferStuffFromContext(SmartNavNodeTag.java:96)`
	`at weblogic.management.console.tags.SmartNavNodeTag.doStartTag(SmartNavNodeTag.java:44)`
	`at weblogic.management.console.webapp._domain.__nav._jspService(__nav.java:301)`
	`at weblogic.servlet.jsp.JspBase.service(JspBase.java:27)`
	`at weblogic.servlet.internal.ServletStubImpl.invokeServlet(ServletStubImpl.java:262)`
	`at weblogic.servlet.internal.ServletStubImpl.invokeServlet(ServletStubImpl.java:198)`
	`at weblogic.servlet.internal.RequestDispatcherImpl.forward(RequestDispatcherImpl.java:250)`
	`[...]`
	この問題は、コードを修正して解決された。

変更要求番号	説明
CR102824	Solaris 8 で WebLogic Server SP02 から SP04 にアップグレードした後、顧客がコンソールにログインしたときに、ログ ファイルに次の例外が記録される。それでも、コンソールは機能を続ける。 `java.lang.NullPointerException at weblogic.management.console.utils.ConsoleComparator.compare(ConsoleComparator.java:81) at java.util.Arrays.mergeSort(Arrays.java:1176) at java.util.Arrays.sort(Arrays.java:1123) at java.util.Collections.sort(Collections.java:116) at weblogic.management.console.utils.MBeans.sort(MBeans.java:1103) at weblogic.management.console.tags.DeclareBeanSetTag.doStartTag(DeclareBeanSetTag.java:99) at weblogic.management.console.webapp._domain.__nav_services._jspService(__nav_services.java:982) at weblogic.servlet.jsp.JspBase.service(JspBase.java:27) at weblogic.servlet.internal.ServletStubImpl.invokeServlet(ServletStubImpl.java:262) at weblogic.servlet.internal.ServletStubImpl.invokeServlet(ServletStubImpl.java:198) at weblogic.servlet.internal.RequestDispatcherImpl.include(RequestDispatcherImpl.java:490) at weblogic.servlet.internal.RequestDispatcherImpl.include(RequestDispatcherImpl.java:316) at weblogic.servlet.jsp.PageContextImpl.include(PageContextImpl.java:116) at …` 分析の結果、MBean が null の `displayName` を返していたことが判明した。この問題は、コードを修正して解決された。
CR105598	Active Directory LDAP を使用した新しいテストで、メンバー名にカンマ (,) が含まれている場合に `group.isMember()` の呼び出しが失敗する問題が発見された。この問題は、コードを修正して解決された。

コア

変更要求番号	説明
CR071415	WebLogic Server 6.1 SP02 では、WebLogic Server のセキュリティ マネージャが起動して、ポリシー ファイルを設定する場合に、JSP のコンパイル時にパス `/usr/lib` が `javac` の先頭に付加され、`java.security.AccessControlException` が生じていた。 この問題は、コードを修正して解決された。
CR094410	`DebugHttp` が `ServerDebugMBean` でアクティブ化された場合に、WebLogic Server は `SocketResetException` を単純なデバッグ メッセージではなくエラーとして報告していた。この問題に対処するために、このデバッグ メッセージを報告するための `logMuxableSocketResetException()` がメッセージ カタログに追加された。
CR096091	WebLogic Server 6.1 SP04 と SP05 では、ファイルからクラスパスを使用して WebLogic Server を Windows サービスとして起動する場合にクラスパス ファイルの長さが約 2K を超えていると、 `“Thread created successfully! Exception in thread “main” java.lang.NoClassDefFoundError: weblogic/Server”` というエラーが送出される。 この問題は、ファイルからのクラスパスの読み出しに関連するエラーを修正して解決した。

変更要求番号	説明
CR102058	クライアントで URLClassLoader を使用する場合に、リモート メソッド呼び出しで NoClassDefFoundError が生じていた。 `c:\java\java131_07\bin\java -classpath ".;client.jar;C:\home\ravia\weblogic\dev\src700\3rdparty\weblogicaux.jar" URLTest qa146 9901 ejb20-statefulSession-TraderHome installadministrator installadministrator java.lang.NoClassDefFoundError: weblogic/rmi/extensions/server/Stub at java.lang.ClassLoader.defineClass0(Native Method) at java.lang.ClassLoader.defineClass(ClassLoader.java:488) at java.security.SecureClassLoader.defineClass(SecureClassLoader.java:106) at weblogic.utils.classloaders.GenericClassLoader.findLocalClass(GenericClassLoader.java:401) at weblogic.utils.classloaders.GenericClassLoader.findClass(GenericClassLoader.java:162) at java.lang.ClassLoader.loadClass(ClassLoader.java:294) at java.lang.ClassLoader.loadClass(ClassLoader.java:250) at java.lang.ClassLoader.loadClassInternal(ClassLoader.java:310) at java.lang.Class.forName0(Native Method) at java.lang.Class.forName(Class.java:190) at weblogic.utils.classfile.utils.CodeGenerator.generateClass(CodeGenerator.java:71) at ...` この問題は、ThreadContextClassLoader を AugmentableSystemClassLoader の親として設定することで解決した。
CR103525	WebLogic Server 6.1 SP04 では、次のエラーが発生していた。 `weblogic.utils.AssertionError: ***** ASSERTION FAILED *****[ Old socket closed and released FD before FDRecord still exists,...` この問題は、マルチプレクサの外側でのソケットのクローズが適切に処理されるようにコードを修正することで解決した。

変更要求番号	説明
CR103774	Solaris および Windows 2000 バージョンの WebLogic Server は、次のようなソケット例外をログに記録していた。 `<Apr 3, 2003 11:40:07 AM EST> <Error> <socket> <000424> <IOException on socket: weblogic.servlet.internal.MuxableSocketHTTP@1eca988 - idle timeout: '30000' ms, socket timeout: '0' ms, fd: 53 java.net.SocketException: Connection reset java.net.SocketException: Connection reset at java.net.SocketInputStream.read(SocketInputStream.java:168) at weblogic.socket.PosixSocketMuxer.readBytesProblem(PosixSocketMuxer.java:876) at weblogic.socket.PosixSocketMuxer.deliverGoodNews(PosixSocketMuxer.java:767) at weblogic.socket.PosixSocketMuxer.processSockets(PosixSocketMuxer.java:694) at weblogic.socket.SocketReaderRequest.execute(SocketReaderRequest.java:23) at weblogic.kernel.ExecuteThread.execute(ExecuteThread.java:213) ad.run(ExecuteThread.java:189) >` ただし、これらのメッセージはサーバまたはアプリケーションの欠陥を示していない。サーバがデバッグ モードの場合のみ上記の例外が表示されるようにコードが修正された。
CR103967	WebLogic Server 6.1 SP04 では、WebLogic Server 実行スレッドの子スレッドが適切なコンテキスト クラスローダを継承していなかった。この問題は、サーブレットまたは EJB がタイマー (`java.util.Timer`) を作成するときに共通に発生していた。 分析の結果、`java.lang.Thread` が独自のメンバー変数を子スレッドに直接割り当てるために、WebLogic Server 実行スレッドは独自のコンテキスト クラスローダを保持し、それらの実行スレッドの子スレッドはコンテキストを継承しないことが判明した。 この問題は、コードを修正して解決された。

変更要求番号	説明
CR104218	WebLogic Server 6.1 SP04 では、CR100665_61sp4.jar のパッチを当てた Solaris 8 サーバで次の例外が発生していた。 `<Apr 23, 2003 3:36:27 PM CDT> <Error> <Posix Performance Pack> <Uncaught Throwable in processSockets java.util.ConcurrentModificationException at java.util.HashMap$HashIterator.nextEntry(HashMap.java:750) at java.util.HashMap$EntryIterator.next(HashMap.java:792) at java.util.HashMap.putAllForCreate(HashMap.java:408) at java.util.HashMap.clone(HashMap.java:624) at java.util.HashSet.clone(HashSet.java:217) at weblogic.socket.PosixSocketMuxer.closeSockets(PosixSocketMuxer.java:486) at weblogic.socket.PosixSocketMuxer.processSockets(PosixSocketMuxer.java:589) at weblogic.socket.SocketReaderRequest.execute(SocketReaderRequest.java:24) at weblogic.kernel.ExecuteThread.execute(ExecuteThread.java:139) at weblogic.kernel.ExecuteThread.run(ExecuteThread.java:120)` この問題は、コードを修正して解決された。

変更要求番号	説明

CR105426	WebLogic Server には、wlntio_g.dll という名前のファイルが付属する。このファイルは、wlntio.dll のデバッグ バージョン (デバッグ目的にのみ必要)。WebLogic Server 6.0 サービス パック 2 には、wlntio_g.dll の間違ったバージョンが含まれていた。このファイルは、デバッグでの使用時に次のようなスタック トレースを生じさせる恐れがあった。
	`<NT Performance Pack> NATIVE.malloc(): allocated at 0x08DAD008 numAllocs 1`
	`numFrees 0`
	`NATIVE: start io on 1972 with addr 0x08dad008`
	`NATIVE: GetQueuedCompletionStatus on 1972 with addr 0x08dad008`
	`<May 6, 2003 11:17:19 AM EDT> <Error> <NT Performance Pack> <failure in processSockets() - GetData: 'weblogic.socket.NTSocketMuxer$GetData - native pointer: '0', numBytes: '0''`
	`java.lang.NoSuchFieldError: fd`
	`at weblogic.socket.NTSocketMuxer.getNextSocket(Native Method)`
	`at weblogic.socket.NTSocketMuxer.processSockets(NTSocketMuxer.java:589)`
	`at weblogic.socket.SocketReaderRequest.execute(SocketReaderRequest.java:24)`
	`at weblogic.kernel.ExecuteThread.execute(ExecuteThread.java:139)`
	`at weblogic.kernel.ExecuteThread.run(ExecuteThread.java:120)>`
	サービス パック 6 には、適切なバージョンの wlntio_g.dll が含まれている。

変更要求番号	説明
CR105516	WebLogic Server 6.1 SP04 では、複数のフェイルオーバが必要なときにステートフル セッション EJB のフェイルオーバが機能していなかった。 3 ノードのクラスタでは、JSP がレプリカ対応ステートフル セッション EJB の作成または呼び出しを行う。リモートの EJB スタブは、http セッションに格納される。ステートフル セッション EJB の各呼び出しの後に、更新された EJB スタブが EJB レプリカ リスト (プライマリ / セカンダリ) の変更を反映して http セッションでも更新される。 同じブラウザ ウィンドウで次のように呼び出しを行うと、クラスタの 1 つのノードだけが生き残る場合に `java.rmi.ConnectException` が送出される。 1. 3 つすべてのクラスタ ノードが動作している。 2. ノード 1 に対して呼び出しを行い、EJB を作成して、リモートを http セッションに格納する (HTTP セッションのレプリケーションが有効になっている)。 3. ノード 1 を強制停止する。 4. セカンダリ ノード 2 に対して呼び出しを行い、EJB のリモートがレプリケートされた http セッションから取得され、EJB の呼び出しが適切に機能する。この EJB の呼び出しの後、再びリモートが http セッションに格納される。 5. ノード 2 を強制停止する。 6. ノード 3 に対して呼び出しを行い、http セッションから EJB のリモートを取得する。 WebLogic Server はノード 2 で EJB をルックアップしようとし、ノード 3 (この時点で新しいセカンダリのはず) を使用しようとしない。ノード 3 で次の例外が送出される。 `java.rmi.ConnectException: Could not establish a connection with -3088833905169218734S:172.23.64.38:[7001,7001,7002,7002,7001,7002,-1]:mydomain:managed2, java.rmi.ConnectException: Destination unreachable; nested exception is: java.net.ConnectException: Connection refused: connect; No available router to destination` `at weblogic.rjvm.RJVMImpl.getOutputStream(RJVMImpl.java:275)` `at weblogic.rjvm.RJVMImpl.getRequestStream(RJVMImpl.java:408)` `at weblogic.rmi.internal.BasicRemoteRef.getOutboundRequest(BasicRemoteRef.java:97)` `at weblogic.rmi.cluster.ReplicaAwareRemoteRef.invoke(ReplicaAwareRemoteRef.java:255)` この問題は、コードを修正して解決された。

変更要求番号	説明
CR105814	WebLogic Server 6.1 SP04 では、AIX 上で `close_waits` が動作しているのが確認された。 分析の結果、WebLogic Serve 6.0 のクライアントが WebLogic Server 6.1 インスタンスに接続したときに、WebLogic Server がソケットをリークしていたことが判明した。この問題は、ソケットをマルチプレクサに登録することなくマルチプレクサを通じて WebLogic Server が MuxableSocket を閉じようとしたときに発生していた (ソケットは t3 によって要求された後、マルチプレクサに登録できる前に拒絶された)。この問題は、コードを修正して解決された。
CR106957	WebLogic Server 6.1 SP04 では、AIX 5.2 上で IBM の MQWorkflow と一緒に動作している場合に、WebLogic Server でサーブレットが MQWorkflow Java API を呼び出したときに、MQWorkflow でエラーが生じていた。この問題は、ネイティブ IO (パフォーマンス パック) が無効になっている場合、または libmuxer.so ライブラリがクラスパスから削除されている場合は Windows では起こらなかった。 分析の結果、WebLogic Server が「language code」(エンコーディング パラメータ) を当然そうであるべき「en-us」に設定せず、「c」に設定していたことが判明した。この問題は、コードを修正して解決された。

変更要求番号	説明

CR107598	WebLogic Server 6.1 SP03 と SP04 では、管理サーバの再起動で次のエラーが生じていた。
	`java.rmi.ConnectException: This RJVM has already been shutdown`
	この問題は、次の手順を行うと必ず発生していた。
	1. サーバインスタンスが少なくとも 2 つのマシン上にあるクラスタをコンフィグレーションする。
	2. サンプル EJB をクラスタにデプロイする。
	3. 管理サーバを再起動する。
	4. EJB の対象からクラスタを外す。
	5. クラスタを再び EJB の対象とする。
	次のエラーが生じていた。
	`<Feb 5, 2003 7:03:27 PM PST> <Info> <J2EE> <Undeployed: ejb_basic_statelessSession> java.rmi.ConnectException: This RJVM has already been shutdown -7152222714009328 37S:172.17.25.250:[7001,7001,7002,7002,7001,7002,-1]:mydomain: myserver`
	`at weblogic.rjvm.RJVMImpl.getOutputStream(RJVMImpl.java:280)`
	`at weblogic.rjvm.RJVMImpl.getRequestStream(RJVMImpl.java:408)`
	`at weblogic.rmi.internal.BasicRemoteRef.getOutboundRequest(BasicRemoteRef.java:97)`
	`at weblogic.rmi.internal.BasicRemoteRef.invoke(BasicRemoteRef.java:125)`
	`at weblogic.rmi.internal.ProxyStub.invoke(ProxyStub.java:35) at $Proxy7.getMBeanServer(Unknown Source)`
	`at weblogic.management.internal.MBeanProxy.getAttribute(MBeanProxy.java:253)`
	この問題は、古くなった MBean の参照を削除するようにコードを修正して解決した。

CR109339	WebLogic Server 6.1 SP05 では、アプレットでカーネルを初期化するときにセキュリティ例外が発生していた。この問題は、コードの修正で解決された。

変更要求番号	説明
CR109590	WebLogic Server 6.1 SP04 では、AIX と一緒に動作している場合に、IIOP 経由でリモート オブジェクトに対して行われるメソッド呼び出しがハングしていた。 このケースでは、非 WebLogic Server のサーバがリモート オブジェクトをホストしている。WebLogic Server 上で動作する EJB が、IIOP 経由で WebLogic Server ではないサーバ上のリモート オブジェクトを呼び出す。このメソッド呼び出しは、無限にハング状態を続ける。メソッド呼び出しは単純なものだった (ブール値を返すだけ)。スレッド ダンプは次のような内容だった。 “ExecuteThread: '11' for queue: 'default'" (TID:0x300C12A8, sys_thread_t:0x35649A58, state:CW, native ID:0x1011) prio=5 at java.lang.Object.wait(Native Method) at java.lang.Object.wait(Object.java:429) at weblogic.iiop.SequencedRequestMessage.waitForData(SequencedRequestMessage.java:24) at weblogic.iiop.EndPointImpl.sendReceive(EndPointImpl.java:641) at weblogic.iiop.OutboundRequestImpl.sendReceive(OutboundRequestImpl.java:43) at weblogic.iiop.IIOPRemoteRef.invokeInternal(IIOPRemoteRef.java:128) at weblogic.iiop.IIOPRemoteRef.invoke(IIOPRemoteRef.java:90) at weblogic.rmi.internal.ProxyStub.invoke(ProxyStub.java:35) at $Proxy70.isRunning(Unknown Source) at java.lang.reflect.Method.invoke(Native Method) at weblogic.iiop.IIOPInvocationHandlerImpl.invoke(IIOPInvocationHandlerImpl.java:285) at $Proxy71.isRunning(Unknown Source) at... 分析の結果、クラスローダに問題があることが判明した。コードの修正で、この問題は解決された。

変更要求番号	説明

CR110347	WebLogic Server 6.1 SP05 では、EJB スタブがクラスパスにない場合に、コンテキストのルックアップ (およびオブジェクトへの割り当て) で `IllegalArgumentException` が送出される。
	この問題は、ステートフル セッション EJB (WebLogic Server に付属する `examples.ejb20.basic.statefulSession`) が 1 つの WebLogic Server サーバにデプロイされている場合に発生していた。EJB ホーム オブジェクトで JDNI ルックアップが実行され、戻り値がクラス オブジェクトに割り当てられていた。クライアントのクラスパスには `weblogic.jar` は含まれていたが、クライアントのクラスは含まれていなかった。
	コンテキストのルックアップは、次のコードで実行されていた。
	`Context ctx = new InitialContext(ht); Object objref = ctx.lookup("ejb20-statefulSession-TraderHome");`
	ルックアップは WebLogic Server 6.1 SP03 では成功したが、WebLogic Server 6.1 SP05 では失敗していた。次のような例外が送出された。
	`java.lang.IllegalArgumentException: java.lang.IllegalArgumentException: interface examples.ejb20.basic.statefulSession.`
	TraderHome は、クラスローダから見えていなかった。JVM で verbose のクラスローディングが設定されている状況で、`examples.ejb20.basic.statefulSession.TraderHome` クラスが (クラスパスにはないが) WebLogic Server 6.1 SP03 では正常にロードされていることが判明した。WebLogic Server 6.1 SP05 では、そのクラスをロードしようとしたときに障害が発生していた。
	この問題は、`ClientRuntimeDescriptor` が現行のクラスローダを使用するようにコードを修正することで解決した (それが GenericClassLoader である場合)。

変更要求番号	説明
CR110892	異なる資格を使用して localhost で InitialContext() を呼び出すと、現在のユーザが変更されていた。 この問題は、WebLogic Integration のビジネス オペレーションが WebLogic Portal アプリケーションにデプロイされた EJB に JMS メッセージを送信するときに見受けられた。そのビジネス オペレーションは、ステートレス セッション Bean を使用して実装されていた。EJB を見つけるために、ビジネス オペレーションは定義済みのユーザ資格を使用して WebLogic Portal ホストとポートの InitialContext を取得していた。WebLogic Portal と WebLogic Integration が同じサーバ インスタンスにデプロイされている場合は、InitialContext() を呼び出すことによって、WebLogic Integration がワークフローを閉じるのを防止する WLI 環境に変化が生じる。WebLogic Integration の「完了した」タスクは、次の例外を送出していた。 `####<Jun 30, 2003 6:51:10 PM EDT> <Info> <B2B> <deletedstring> <deletedstring> <ExecuteThread: '14' for queue: 'default'> <kernel identity> <38:d06db1d7b915f0cd> <000000> <<B2B-BPM-Plugin> INFO: Log from Workflow: :::: Usecases BPM_AI_asyncWLP [12001] async ::: Done> ####<Jun 30, 2003 6:51:11 PM EDT> <Info> <EJB> <deletedstring> <deletedstring> <ExecuteThread: '14' for queue: 'default'> <kernel identity> <> <010099> <Message-Driven EJB: EventListener's transaction was rolledback. The transaction details are: Xid=38:d06db1d7b915f0cd(4745279),Status=Rolled back. [Reason=weblogic.transaction.internal.AppSetRollbackOnlyException],numRepliesOwedMe=0,numRepliesOwedOthers=0,seconds since begin=0,seconds left=60,ServerResourceInfo[JMS_JMSWLIStore]=(state=rolledback,assigned=si_server,xar=JMS_JMSWLIStore),ServerResourceInfo[weblogic.jdbc.jts.Connection]=(state=rolledback,assigned=si_server,xar=weblogic.jdbc.jts.Connection@3431bd),SCInfo[si_domain+si_server]=(state=rolledback),properties=({weblogic.jdbc=t3://10.61.7.124:8501}),OwnerTransactionManager=ServerTM[ServerCoordinatorDescriptor=(CoordinatorURL=si_server+10.61.7.124:8501+si_domain+t3+, Resources={WebLogic DBMS Adapter...` 分析の結果、現在のサーバの URL を使用して「new InitialContext(p)」を呼び出すと、呼び出しで使用されたユーザと現在のスレッドが関連付けられることが判明した。この問題は、コンテキスト オブジェクトを閉じるロジックをコード変更することで解決した。詳細については、http://dev2dev.bea.com/resourcelibrary/advisoriesnotifications/BEA03-35 を参照。

変更要求番号	説明
CR116712	WebLogic Server 6.1 SP04 では、INVOKE が管理ツールを使用して WLECConnectionPoolRuntime で発行された場合に WLECConnectionRuntime MBeans が更新されていなかった。 分析の結果、resetConnectionPool() が WLECConnectionPoolRuntime MBean で呼び出されたときに、プールがリセットされていないことが判明した。古い接続が削除されていなかった。 この問題は、コードを修正して解決された。現在、古い接続に対応する MBean は resetConnectionPool() の呼び出しの前に登録が解除されるようになっている。

変更要求番号	説明

CR117200	WebLogic Server 6.1 SP05 では、管理対象サーバが管理サーバに接続できないときに weblogic.socket.PosixSocketMuxer のデッドロックが検出されていた。次に示すのは、スレッド ダンプからの抜粋。
	`1wasLKDEADLOCK Deadlock detected!!!`
	`NULL`
	`NULL`
	`2LKDEADLOCKTHR Thread “ExecuteThread: '0' for queue: '__weblogic_admin_rmi_queue'” (0x8461B8A0) 3LKDEADLOCKWTR is waiting for:`
	`4LKDEADLOCKMON sys_mon_t:0x84D183A8 inflaming: 0x00000000: 4LKDEADLOCKOBJ weblogic.socket.PosixSocketMuxer$FDRecord@31FD4B10/31FD4B18:`
	`3LKDEADLOCKOWN which is owned by: 2LKDEADLOCKTHR Thread “ExecuteThread: '98' for queue: 'default'” (0x767D26E0)`
	`3LKDEADLOCKWTR which is waiting for:`
	`4LKDEADLOCKMON sys_mon_t:0x83AD8C18 inflaming: 0x00000000: 4LKDEADLOCKOBJ weblogic.rjvm.ConnectionManagerServer@31FA3430/31FA3438:`
	`3LKDEADLOCKOWN which is owned by:`
	`2LKDEADLOCKTHR Thread “ExecuteThread: '0' for queue: '__weblogic_admin_rmi_queue'” (0x8461B8A0)`
	次の例外も送出される。
	`java.rmi.ConnectException: The connection manager to ConnectionManager for: 'weblogic.rjvm.RJVMImpl@2a7512ad - id: '-8419466038107054512S:10.32.197.52:[7001,7001,-1,-1,7001,-1,-1]:bossapp:AdminServer' connect time: 'Fri. Aug 01 09:43:07 CST 2003'' has already been shut down at weblogic.rjvm.ConnectionManager.getOutputStream(ConnectionManager.java(Compiled Code)) at...`
	分析の結果、PosixSocketMuxer のデッドロックの原因は FDRecord ロックと ConnectionManager ロックの競合であることが判明した。1 つのスレッドが FDRecord ロックを保持しつつ、FDRecord によるクリーンアップの実行を待っている別のスレッドに保持されている ConnectionManager ロックを待っているのである。
	コードの修正でこの競合は解消された。現在、FDRecord ロックはディスパッチの間は保持されなくなっている。

変更要求番号	説明

CR122280	コンテキストワイドなセッションで、WebLogic Server はオブジェクトがシリアライズ可能かどうかセッションに追加する前にチェックしていなかった。この問題で、次のエラーが引き起こされる恐れがあった。
	`Could not deserialize context attribute java.io.NotSerializableException: com.app.name`
	`at java.io.ObjectOutputStream.outputObject(ObjectOutputStream.java(Compiled Code))`
	`at java.io.ObjectOutputStream.writeObject(ObjectOutputStream.java(Compiled Code))`
	`at weblogic.servlet.internal.AttributeWrapper.getObject(AttributeWrapper.java(CompiledCode))`
	`at weblogic.servlet.internal.WebAppServletContext.getAttribute(WebAppServletContext.java(Compiled Code))`
	`at weblogic.servlet.internal.WebAppServletContext.getAttribute(WebAppServletContext.java(Compiled Code))`
	`setAttribute()` がセッションへの追加前にシリアライズ可能オブジェクトをチェックするようにコードが修正された。

CR123571	同じマシン上で複数の T3 クライアントを同時に起動すると、それらの複数のクライアントが同じ JVMID を取得し、例外が生じるか、クライアントがハングする可能性があった。この問題は、同じマシン上で同時に複数の T3 クライアントを起動したときのみ発生していた。この問題は、JVMID を生成するコードを修正して解決した。

変更要求番号	説明
CR124763	WebLogic Server サービス パック 6 より前には、クラスタ内のすべてのサーバ インスタンスがサーバのリスン ポートをクラスタ通信のマルチキャスト ポートとして使用していた。サーバのリスン ポートと異なるマルチキャスト ポートをクラスタで使用することはできなかった。 このサービス パックでは、クラスタで使用するマルチキャスト ポートを指定するための新しい任意指定の Java プロパティ `-Dweblogic.cluster.multicastport=`*port_number* が導入される。この新しいプロパティを使用するには、すべてのクラスタ メンバーがその起動スクリプトで同じ `-Dweblogic.cluster.multicastport` 値を指定する必要がある。 起動時に `-Dweblogic.cluster.multicastport` が指定されないと、サーバは引き続きリスン ポートをクラスタ通信のマルチキャスト ポートとして使用する。

デプロイメント

変更要求番号	説明

CR102324	このサービス パックがリリースされる前は、WebLogic Server はデプロイメント エラーと関連するネストされた例外をログに記録していなかった。ログに記録されたテキストは、次のようにデプロイメント エラーのみを示していた。
	`<Mar 28, 2003 1:10:51 AM PST> <Error> <J2EE> <Error deploying application application_name: Caught IOException for path=path_to_application>`
	次のようにデプロイメント エラーだけでなくネストされた例外もログに記録されるようにコードが修正された。
	`<Mar 28, 2003 1:10:51 AM PST> <Error> <J2EE> <Error deploying application_name: with deployment error Caught IOException for path=path_to_application and nested error java.io.FileNotFoundException: File not found>`

EJB

変更要求番号	説明
CR056023	Oracle データベースにアクセスする CMP2.0 EJB は、データベースがロックされている場合にタイムアウトにならなかった。 この問題は、`trans-timeout-seconds` が「10」などのゼロ以外の値に設定されている状態で、ロックされたデータベース テーブルに対して ejb2.0 cmp の例が実行されたときに見受けられた (テーブルは SQLplus「lock table ejbaccounts in exclusive mode」を使用してロックされていた)。テーブルがロックされている状態でクライアントが実行されると、そのクライアントはアカウントを作成しようとして無限にハング状態を続けていた。SQLplus を使用してデータベース ロックが解除された後は、クライアントでロールバック例外が発生していた。トランザクションのタイムアウト時には、ロールバックが行われなければならない。 分析の結果、タイムアウト期間が切れて、タイマーが `TransactionRolledBack` メッセージをデータベースに送信したときに、データベースがロックを解除していないことが判明した。 この問題は、Oracle Thin ドライバの行レベルのロックについて以下の手段で解決した。 ■ 接続のロールバックの前に、`weblogic.jdbc.jts.Connection` クラスの `internalRollback` メソッドで `canceAllStatementsBeingUsed` を実行する ■ 開いているすべての文を反復処理してそれらをキャンセルし、接続がロールバックされ、SQL 例外を無視するようにする新しいメソッド `canceAllStatementsBeingUsed` を `weblogic.jdbc.common.internal.ConnectionEnv` クラスに導入する この修正は、Oracle Thin ドライバの行レベルのロックについてのみ有効。

変更要求番号	説明
CR098343	WebLogic Server 6.1 SP02 と SP04 では、トランザクションが 2 つのサーバ インスタンスに分散されているときに JDBC 接続がプールに戻らなかった。JDBC 属性 `KeepXAConnTillTxComplete` が `true` に設定されていた。このイベント シーケンスにより、次の問題が発生した。 1. サーバ インスタンス 1 でユーザ トランザクションを開始する。 2. JMS メッセージを永続キューに格納する。JMS サーバおよびキューはサーバ インスタンス 2 に存在する。 3. サーバ インスタンス 2 で EJB を更新する。 4. トランザクションをコミットする。 エラー メッセージは生成されなかった。更新が実行され、メッセージが JMS キューに格納された。しかし、ejb 接続プール内に「使用中」のままになっている接続が存在した。この手順を何度も繰り返すと、接続プール内の接続が使い果たされる。 この問題は、トランザクションが単一のサーバ インスタンスで行われるか、または JMS メッセージがトランザクション スコープの外のキューに格納される場合には発生しなかった。 この問題は、`aftercompletion` コールバックで接続が解放されるようにコードを修正したことで解決された。

変更要求番号	説明
CR099041	WebLogic Server 6.1 SP04 では、顧客が使用しているアプリケーションの特定の EJB でメソッドを呼び出している間に次の例外がランダムに生成されることが報告された。 `java.lang.ClassCastException: java.lang.String at weblogic.utils.classfile.DoubleKey.equals(DoubleKey.java:24) at java.util.Hashtable.get(Hashtable.java:318) at weblogic.utils.classfile.ConstantPool.find(ConstantPool.java:58) at weblogic.utils.classfile.ConstantPool.getUtf8(ConstantPool.java:251) at weblogic.utils.classfile.expr.ClassInfo.addField(ClassInfo.java:86) at weblogic.utils.classfile.expr.DotClassExpression.code(DotClassExpression.java:34) at weblogic.utils.classfile.expr.InvokeExpression.code(InvokeExpression.java:31) at weblogic.utils.classfile.expr.ExpressionStatement.code(ExpressionStatement.java:19) at weblogic.utils.classfile.expr.TryCatchStatement.code(TryCatchStatement` この問題は、`weblogic.utils.classfile.DoubleKey` の `equals()` メソッドをコード修正することで解決した。
CR099626	WebLogic Server 6.1 SP03 では、ejbc が `ejbSelect` クエリで無効な SQL を生成していた。このクエリは、テーブル エリアスが不正確に生成されて次のエラーが生じていたので失敗していた。 `ORA-00904: invalid column name` この問題は、コードを修正して解決された。
CR099760	EJB 1.1 CMP Bean でフィールドの最適化が実装されたので、更新があっても値は変更されなかったフィールドがデータベースに書き込まれなくなった。この最適化は、プリミティブおよび不変オブジェクトでのみ実行される。

変更要求番号	説明
CR100246	テストでステートフル EJB を削除しようとしたときに LockTimeoutException が発生していた。処理は次のように行われていた。 1. テスト時に、エンティティ Bean のリモート メソッドの中でステートフル セッション Bean を作成する (local/localhome インタフェース)。 2. ステートフル セッション Bean のビジネス メソッドを呼び出す (このメソッドが今度はさまざまなタイプの Bean を作成する)。 3. ステートフル セッション Bean の削除を試行する。 remove の呼び出しで、次の例外が送出される。 `Nov 18, 2002 12:51:33 AM PST> <Info> <EJB> <010049> <EJB Exception in method: remove: weblogic.ejb20.locks.LockTimedOutException: The lock request from EJB:LibraryInfoEJB with primary key:85335648642269185 timed-out after waiting 0 ms. The transaction or thread requesting the lock was:Thread[ExecuteThread: '6' for queue: 'default',5,Thread Group for Queue: 'default'].` `weblogic.ejb20.locks.LockTimedOutException: The lock request from EJB:LibraryInfoEJB with primary key:85335648642269185 timed-out after waiting 0 ms. The transaction or thread requesting the lock was:Thread[ExecuteThread: '6' for queue: 'default',5,Thread Group for Queue: 'default'].` `at weblogic.ejb20.locks.ExclusiveLockManager$LockBucket.lock(ExclusiveLockManager.java:449)` `at weblogic.ejb20.locks.ExclusiveLockManager.lock(ExclusiveLockManager.java:259)` `at weblogic.ejb20.manager.StatefulSessionManager.acquireLock(StatefulSessionManager.java:248)` `at weblogic.ejb20.manager.StatefulSessionManager.acquireLock(StatefulSessionManager.java` この問題は、weblogic-ejb-jar.xml デプロイメント記述子の新しい要素 <allow-remove-during-transaction> を追加することで解決した。この要素を true に設定すると、上の例外は発生しない。

変更要求番号	説明
CR101918	ホットスポットがあり、ネイティブ IO が無効化された NT 上で、ejb20 ホーム メソッドのテスト中に次のエラーが送出された。 `java.lang.IllegalStateException: zip file closed at java.util.zip.ZipFile.ensureOpen(ZipFile.java:377)atjava.util.zip.ZipFile.getEntry(ZipFile.java:138)at eblogic.utils.classloaders.ClasspathClassFinder.getSourcesInternal(ClasspathClassFinder.java:225)at weblogic.utils.classloaders.ClasspathClassFinder.getSource(ClasspathClassFinder.java:155)at weblogic.utils.classloaders.MultiClassFinder.getSource(MultiClassFinder.java:53)at weblogic.utils.classloaders.MultiClassFinder.getClassSource(MultiClassFinder.java:45)at weblogic.utils.classloaders.GenericClassLoader.findLocalClass(GenericClassLoader.java:303)at weblogic.utils.classloaders.GenericClassLoader.findClass(GenericClassLoader.java:161)weblogic.rmi.internal.BasicRuntimeDescriptor.getClientRuntimeDescriptor(BasicRuntimeDescriptor.java:526)...`  分析の結果、この問題はアンデプロイメント時の未キャンセルのトリガが関連していることが判明した。以下の修正で問題は解決された。  ■ ejbc の最後に Utilities クラスで静的クラスローダ エントリが削除されるように `rmic` のコードが修正された。  ■ スレッドを再利用する前にコンテキスト クラスローダがクリーニングされるように `ExecuteThread` が修正された。 ■ Exclusive 同時方式の cmp/bmp がアンデプロイされるときにトリガがキャンセルされるように ejb のコードが修正された。

変更要求番号	説明
CR102028	接続プールの作成で Oracle Thin ドライバを使用するときに、DB QueryString が不適切に生成されていた。 1. Oracle Thin ドライバを使用して接続プールを作成する。 2. テスト ケース ejb20/relations/FindersTest/testStringFunctionLOCATE() を実行する。 3. 上のテスト ケースは、ejb20/relations/finder.ts ファイルから実行できる。 4. EJB の「ComputerEJB」、「EmployeeEJB」、および「FinderEmployeeEJB」が必ず、この Thin ドライバを使用して作成された接続プールを使用するようにする。 5. これを実行すると、「Invalid Character」エラーが生じる。 スタック トレースは次のとおり。 `javax.ejb.FinderException: Problem in findByNameEquals while preparing or executing statement: 'weblogic.jdbc.jts.PreparedStatement@55c5eb': java.sql.SQLException: ORA-00911: invalid character java.sql.SQLException: ORA-00911: invalid character at oracle.jdbc.dbaccess.DBError.throwSqlException(DBError.java:134) at oracle.jdbc.ttc7.TTIoer.processError(TTIoer.java:289) at oracle.jdbc.ttc7.Oall7.receive(Oall7.java:573) at oracle.jdbc.ttc7.TTC7Protocol.doOall7(TTC7Protocol...` この問題は、適切な SQL クエリが生成されるようにコード修正することで解決した。

変更要求番号	説明
CR102180	WebLogic Server SP03 では、`100 EJBs` を使用して .jar のスタブとスケルトンをコンパイルするときに `ejbc` が失敗していた。次の例外が発生した。 `java.io.IOException: CreateProcess: c:\VisualCafe\bin\sj.exe -classpath c:\java\java130\jre\lib\rt.jar; c:\java\java130\jre\lib\i18n.jar;c:\java\java130\jre\lib\sunrsasign.jar;c:\java\java130\jre\classes; c:/java/java130/lib/tools.jar;D:/weblogic/dev/src/3rdparty/jconnect/jConnect.jar;D:/weblogic/src_130sj/license;D:/weblogic/dev/src/3rdparty/oracle/816/classes12.zip; D:/weblogic/src_130sj/classes;D:/weblogic/src_130sj/lib/xmlx.jar;D:/weblogic/src_130sj/lib/ejb20.jar;D:/weblogic /dev/src/3rdparty/weblogicaux.jar;D:/weblogic/dev/src/3rdparty /cloudscape/lib/cloudscape.jar;D:/weblogic/src_130sj/classes_ifmx4; D:/weblogic/src_130sj/classes_mssql4;D:/weblogic/src_130sj/tools;c:/java/java130/jre/lib/rt.jar;D:\raviWork\ejb...` この問題は、コンパイラのコマンドラインの長さ制限が原因。この問題は、`javac @tempfile` 機能を使用することで解決した。この機能を使用すると、一時ファイルを使用してファイル名をコンパイラに渡すことができる。
CR102308	WebLogic Server 6.1 SP04 では、Administration Console でエンティティ Bean の待機について不正確な値が報告されていた。 この問題は、`waiterCurrentCount` 属性を追加したことによって解決された。この属性は、クライアントがロック待機を開始するときに増加し、ロックを取得またはクライアントがタイムアウトしたときに減少する。
CR103047	WebLogic Server 6.1 SP03 では、MDB のトランザクション タイムアウトがログに記録されないという報告があった。JMS 送り先からのメッセージを処理するために MDB が要する時間がトランザクションのタイムアウト制限を超えると、トランザクションがロールバックされ、メッセージは再配信用に送り先に戻されるが、transaction-timeout または transaction-rollback メッセージがログに記録されていなかった。 この問題は、トランザクション タイムアウトを報告するように MDListener のコードを修正することで解決された。

変更要求番号	説明
CR103978	WebLogic Server 6.1 SP02 では、以下の状況下で Web アプリケーションがセッションからリモート オブジェクトを取得できていなかった。 ■ Web アプリケーションがドメイン 1 の管理サーバにデプロイされている ■ EJB がドメイン 2 のクラスタ内の 1 つの管理対象サーバにデプロイされている ■ Web アプリケーションが、EJB がデプロイされている管理対象サーバの URL を使用して EJB のルックアップを行った ■ Web アプリケーションがリモート オブジェクトを作成して、それを HttpSession に追加した ■ Web アプリケーションが、次の例外を発してセッションからリモート オブジェクトを取得するのに失敗した `<Apr 21, 2003 11:42:47 AM PDT> <Error> <HTTP Session> <Error reconstructing the EJBObject put into session for name: trader java.rmi.NoSuchObjectException: Unable to locate EJBHome: 'statelessSession.TraderHome' on server: 't3://acaoclust:7001 at weblogic.ejb20.internal.HomeHandleImpl.getEJBHome(HomeHandleImpl.java :80) at weblogic.ejb20.internal.HandleImpl.getEJBObject(HandleImpl.java:179) at weblogic.servlet.internal.session.SessionData.getAttribute(SessionDat a.java:390) at jsp_servlet.__remoteejb._jspService(__remoteejb.java:119) at weblogic.servlet.jsp.JspBase.service(JspBase.java:27) at` このエラーは、EJB がクラスタではなく 1 つの管理対象サーバにデプロイされているにもかかわらず、Web アプリケーションが HttpSession からリモート EJB オブジェクトを取得するのにクラスタの URL を使用したことが原因だった。 この問題は、コードの修正で解決された。現在、EJB ハンドル (HomeHandle) は、`weblogic-ejb-jar.xml` ファイルの `home-is-clusterable` デプロイメント要素の指定に従って、EJB ホームがクラスタ化可能でない場合は、EJB がデプロイされているサーバの URL を使用するようになっている。クラスタの URL は、`home-is-clusterable` が true の場合だけ使用される。

変更要求番号	説明
CR104414	WebLogic Server 6.1 SP04 では、クラスパスが長い場合にサーバ側で ejbc コンパイル エラーが発生していた。次のエラーが発生していた。 `[EJBCompiler] : Compiling EJB sources Warning: UNIXProcess.forkAndExec native error: The parameter or environment lists are too long. <Apr 24, 2003 2:55:27 PM GMT> <Error> <J2EE> <Error deploying application applicationname: Unable to deploy EJB: monitor/EMAServerManager.jar from monitor/EMAServerManager.jar: Compiler failed executable.exec(java.lang.String[javac, -nowarn, -classpath, ...long classpath which is not shown here.....)` `at weblogic.ejb20.ejbc.EJBCompiler.compileEJB(EJBCompiler.java(Compiled Code))` `at weblogic.ejb20.deployer.Deployer.runEJBC(Deployer.java(Inlined Compiled Code))` `at weblogic.ejb20.deployer.Deployer.compileEJB(Deployer.java(Compiled Code))` この問題は、-compilerclass オプションを指定してインライン コンパイルを使用し、compilerclass の compile メソッドが呼び出されるように callcompile フラグの代わりに noexit を使用することで解決した。

変更要求番号	説明
CR105609	WebLogic Server 6.1 SP04 のサーブレットまたはステートレス セッション Bean が WebLogic Server 6.1 SP02 にデプロイされたステートフル セッション Bean を呼び出したときに Null ポインタ例外が送出されていた。スタック トレースは次のとおり。 `java.lang.NullPointerException at weblogic.ejb20.internal.HandleImpl.readExternal(HandleImpl.java:101) at java.io.ObjectInputStream.inputObject(ObjectInputStream.java:1212) at java.io.ObjectInputStream.readObject(ObjectInputStream.java:386) at java.io.ObjectInputStream.inputClassFields(ObjectInputStream.java:2263) at java.io.ObjectInputStream.defaultReadObject(ObjectInputStream.java:519) at java.io.ObjectInputStream.inputObject(ObjectInputStream.java:1412) at java.io.ObjectInputStream.readObject(ObjectInputStream.java:386) at java.io.ObjectInputStream.readObject(ObjectInputStream.java:236) at java.util.ArrayList.readObject(ArrayList.java:531) at java.lang.reflect.Method.invoke(Native Method) at java.io.ObjectInputStream.invokeObjectReader(ObjectInputStream.java:2214) at java.io.ObjectInputStream.inputObject(ObjectInputStream.java:1411) at java.io.ObjectInputStream.readObject(ObjectInputStream.java:386) at java.io.ObjectInputStream.readObject(ObjectInputStream.java:236) at` 分析の結果、次のことが判明した。ステートフル セッション Bean のハンドルにはばらつきがあり、スタブがハンドルの使用において問題を抱えていた。そのことに関する問題を解決するため、ハンドルの通信フォーマットが WebLogic Server 6.1 SP02 後に拡張されていた。ステートレス セッション Bean の場合には、スタブにハンドルは追加されていない。 この問題は、ステートレス セッション Bean の「ハンドルなし」が許可されるようにコードを修正することで解決した。

変更要求番号	説明
CR106136	WebLogic Server 6.1 SP04 では、`delay-updates-until-end-of-tx` が false に設定されている場合に、EJB 1.1 CMP Bean のゲッター メソッドが `isModified()` を呼び出していなかった。 セッション EJB が、エンティティ EJB のゲッター メソッドを呼び出していた。両方の EJB に、トランザクション属性が `Required` に設定されたコンテナ管理のトランザクションがあった。ゲッター メソッドの各呼び出しの後には `ejbStore()` の呼び出しが続き、`delay-updates-until-end-of-tx` は `false` だった。しかし、Bean で `ejbStore` を呼び出す前に、コンテナが `isModified` メソッドを呼び出していなかった。`isModified()` メソッドは、トランザクションがコミットされたときにのみ呼び出されていた。 `ejbStore` は、Bean での `isModified` メソッドの結果に応じて `postInvoke()` から呼び出されるべき。この問題は、コードを修正して解決された。
CR111551	WebLogic Server 6.1 SP05 では、`java.lang.ClassCastException: oracle.sql.BLOB` エラーを生じさせる CMP コードを EJBC が生成していた。 分析の結果、クエリ用に生成された CMP RDBMS コードが不正確であることが判明した。WebLogic Server 6.1 SP05 より前は、サーバ側の JDBC は RMI ドライバ、プール ドライバを経由して DBMS ドライバに辿り着いていた。WebLogic Server 6.1 SP05 からは、RMI ドライバは外部クライアントでのみ呼び出される。サーバ側のクライアント コードは直にプール ドライバにアクセスし、そのプール ドライバが RMI ラッパーでラップされていない BLOB を直に DBMS ドライバに返す。この変更では、`weblogic.jdbc.common.OracleBlob` ではなく `oracle.sql.BLOB` に直接コードがキャストされる必要がある。 この問題は、SP05 以降の動作を反映するコードが生成されるように EJBC を修正することで解決した。

変更要求番号	説明
CR111771	WebLogic Server 6.1 SP02 では、Bean がアンデプロイされたときにデータベースが矛盾した状態のまま置かれていた。この問題は、次の呼び出しシナリオで起こっていた。 `webApp <--> OuterBean <--> InnerBean` Bean はステートレス セッション Bean であり、コンテナ管理のトランザクションを使用し、`trans-attribute=Required` が設定されている。`OuterBean.m()` はテーブルに行を挿入してから、別のテーブルに行を挿入する `InnerBean.m()` を呼び出す。このシナリオは単一トランザクションのコンテキスト内で起こるので、両方のテーブルに同じ数の行がある。Bean は、別々の jar にパッケージ化される。`OuterBean` は `InnerBean` のホームをキャッシュし、すべてが 1 つのサーバにデプロイされる。`InnerBean` は、テストの過程でサーバを対象から外す。この場合に、テーブルは矛盾した状態に置かれる (`OuterBean` はそれ以上テーブルに行を挿入しないが、`InnerBean` は引き続き行う)。 分析の結果、`InnerBean` がアンデプロイされた後に、`TxManager.undeploy` が処理中のトランザクションをロールバックし、それ以上トランザクションが登録されないようにそのインスタンスに応答なしのマーキングをすることが判明した。`OuterBean` と `InnerBean` は同じエンタープライズ アプリケーションにパッケージ化されており、呼び出しは参照によって行われ、RMI を経由していた。ホームと Bean は `OuterBean` にキャッシュされていたので、`InnerBean` がアンデプロイされた後でも、呼び出されたリモート メソッドではサービスが提供されていた。`InnerBean` がアンデプロイされた後、`OuterBean` が `InnerBean.insert()` を呼び出したときに、その呼び出しと関連するトランザクションがあった (`OuterBean` によって開始されたもの)。preInvoke では、TxListener が設定され、トランザクションが TxManager に登録される。TxManager への登録の間、インスタンスは応答なしと判断され、WebLogic Server はトランザクションをロールバックして通知なしに復帰する。トランザクションが登録されていないので、`InnerBean.insert()` 内で取得されたデータベース接続は呼び出されたトランザクション内にはなく、`OuterBean` で実行されたロールバックは `InnerBean` で挿入された行に影響を与えない。その結果として、`OuterBean` はそれ以上テーブルに行を挿入しないが、`InnerBean` は挿入を続ける、という状況が生まれる。(Bean がアンデプロイされたのでトランザクションをロールバックした後) 通知なく復帰することが問題の原因。 この問題は、`BaseEJBManager` の preInvoke でデプロイメントのステータスをチェックし、アンデプロイされた Bean に呼び出しが届かないようにするコード修正で解決した。

変更要求番号	説明
CR115026	WebLogic Server 6.1 SP02 では、MDB が MQSeries 上でリスンしている状態で、WebLogic Server 側の MQSeries の java スレッドが MQSeries の障害および MQSeries の再起動のときに適切に閉じられていなかった。障害または再起動のたびに、MQSeries の AsyncThread の数が 2 倍になっていた。 この問題は、`JMSConnectionPoller.createJMSConnection()` の障害とクリーン アップのコードを改良することで解決した。

JDBC

変更要求番号	説明
CR098343	トランザクションが 2 つのサーバ インスタンスに分散されているときに JDBC 接続がプールに戻らなかった。JDBC 属性 `KeepXAConnTillTxComplete` が true に設定されていた。このイベント シーケンスにより、次の問題が発生した。 1. サーバ インスタンス 1 でユーザ トランザクションを開始する。 2. JMS メッセージを永続キューに格納する。JMS サーバおよびキューはサーバ インスタンス 2 に存在する。 3. サーバ インスタンス 2 で EJB を更新する。 4. トランザクションをコミットする。 エラー メッセージは生成されなかった。更新が実行され、メッセージが JMS キューに格納された。しかし、EJB 接続プール内に「使用中」のままになっている接続が存在した。この手順を何度も繰り返すと、接続プール内の接続が使い果たされてしまった。 この問題は、トランザクションが単一のサーバ インスタンスで行われるか、または JMS メッセージがトランザクション スコープの外のキューに格納される場合には発生しなかった。 この問題は、`aftercompletion` コールバックで接続が解放されるようにコードを修正したことで解決された。
CR099872	`Exception during commit of transaction` スタック トレース例外のメッセージに接続プール名が含まれているが、データ ソース名が含まれていなかった。 `<Exception during commit of transaction Xid=39:74a54046e2c2bb30(5962554),Status=Rolled back. [Reason=javax.transaction.xa.XAException: JDBC driver does not support XA, hence cannot be a participant in two-phase commit. To force this participation, set the EnableTwoPhaseCommit property on the corresponding JDBCTxDataSource property, to true. Pool = CatPhase1]...` スタック トレースの内容を拡張して、データ ソースの名前が含まれるようにした。

変更要求番号	説明
CR103046	JDBC 接続プールの使用時に `Statement.close()` を発行しても、基底の `ResultSet` に関連付けられたすべてのリソースが解放されていなかった。それが原因で、`OutOfMemoryError` が生じる恐れがあった。この問題は、接続プールを通じて `Statement` を閉じるときにのみ発生していた。明示的に `ResultSet` 自体を閉じるとき、またはドライバを直接使用しながら文を閉じるときには発生していなかった。 この問題は、コードを修正して解決された。

変更要求番号	説明
CR103299	このサービス パックでは、次のスタック トレースを生じさせていたリーク検出コードのエラーが修正される。 `[SerialConnection] : Connection Leak detected!!!!!!java.lang.Throwable: StackTrace at creation of connection: Start server side stack trace: java.lang.Throwable: StackTrace at creation of connection: at weblogic.jdbc.rmi.SerialConnection.<init>(SerialConnection.java:47) at weblogic.jdbc.pool.Connection.writeReplace(Connection.java:1427) at java.lang.reflect.Method.invoke(Native Method)at java.io.ObjectStreamClass.invokeMethod(ObjectStreamClass.java:1615) at java.io.ObjectOutputStream.writeObject(ObjectOutputStream.java:303) at weblogic.common.internal.ChunkedObjectOutputStream.writeObject(ChunkedObjectOutputStream.java:115) at weblogic.rjvm.MsgAbbrevOutputStream.writeObject(MsgAbbrevOutputStream .java:82) at weblogic.jdbc.common.internal.RmiDataSource_WLSkel.invoke(Unknown Source) at weblogic.rmi.internal.BasicServerRef.invoke(BasicServerRef.java:398) at weblogic.rmi.cluster.ReplicaAwareServerRef.invoke(ReplicaAwareServerRef.java:114) at weblogic.rmi.internal.BasicServerRef$1.run(BasicServerRef.java:339) at weblogic.security.service.SecurityServiceManager.runAs(SecurityServiceManager.java:850) at weblogic.rmi.internal.BasicServerRef.handleRequest(BasicServerRef.java:334) at weblogic.rmi.internal.BasicExecuteRequest.execute(BasicExecuteRequest .java:30)at weblogic.kernel.ExecuteThread.execute(ExecuteThread.java:215) at weblogic.kernel.ExecuteThread.run(ExecuteThread.java:191) End server side stack trace` この問題は、メソッド コードの修正によって解決された。
CR103321	WebLogic Server 6.1 SP04 では、`JDBCConnectionPoolRuntimeMBean` の `getConnectionsTotalCount()` メソッドが期待通りに機能していなかった。インスタンス化からのプール内の JDBC 接続の総数を返す代わりに、インスタンス化からの最大接続数を返した。 この問題は、メソッド コードの修正によって解決された。

変更要求番号	説明

CR103421	WebLogic Server 6.1 のサービス パック 3 と 4 では、BigDecimal データ型が関わるトランザクションで、そのトランザクションがリモートの DataSource を取得する前にローカルの DataSource を取得した場合に SQLWarning 例外が送出されていた。スタック トレースのテキストは次のとおり。
	`java.sql.SQLException: java.sql.SQLWarning: Exhausted Resultset`
	`Start server side stack trace:`
	`weblogic.common.internal.WLSQLWarning: Exhausted Resultset`
	`at weblogic.jdbc.common.internal.DriverProxy.getWLSQLFromSQLException(DriverProxy.java:2`
	`at weblogic.jdbc.common.internal.ResultSetProxy.execute(ResultSetProxy.java:716)`
	`at weblogic.t3.srvr.ClientRequest.execute(ClientContext.java:769)`
	`at weblogic.kernel.ExecuteThread.execute(ExecuteThread.java:139)`
	`at weblogic.kernel.ExecuteThread.run(ExecuteThread.java:120)`
	`End server side stack trace`
	この問題は、T3 ドライバがキャッシュのデータではなく常にリモートの結果セットで getBigDecimal() を問い合わせていたことが原因。
	この問題は、コードを修正して解決された。

CR103619	このサービス パックでは、isClosed() メソッドと JTS 接続の問題が修正される。それらの問題は、Sun の JDBC 用 CTS を実行したときに検出された。
CR106767	アプリケーションが複数回 JDBC オブジェクトをクローズしようとすると、WebLogic Server は例外を送出するようになった。

変更要求番号	説明
CR108580	WebLogic Server は、ConnectionLeakProfile オブジェクトの作成前に null のプール名をチェックできず、次の例外を生じさせていた。 `java.lang.NullPointerExceptionat java.lang.String.<init>(String.java:193) at weblogic.jdbc.common.internal.ConnectionLeakProfile.<init>(ConnectionLeakProfile.java:21)at weblogic.jdbc.rmi.internal.ConnectionImpl.unreferenced(ConnectionImpl .java:144) at weblogic.rmi.internal.BasicServerRef$UnreferencedExecuteRequest.execute(BasicServerRef.java:702) at weblogic.kernel.ExecuteThread.execute(ExecuteThread.java:234 at weblogic.kernel.ExecuteThread.run(ExecuteThread.java:210)` この問題は、コードを修正して解決された。
CR112607	接続プールの URL またはパスワードが正確でない場合でも、WebLogic Server 6.1 サービス パック 4 では初期容量 0 の新しい JDBC 接続プールが作成されていた。URL またはパスワードが正しくない場合は JDBC 接続プールの作成が失敗するようにコードが修正された。
CR120455	WebLogic Server 6.1 サービス パック 2 でメモリ リークが検出された。このリークは、データベース上の BLOB カラムに TxDataSource を使用してアクセスしたときに発生していた (WebLogic XA ドライバを使用)。この問題は、コードを修正して解決された。

変更要求番号	説明
CR120531	WebLogic Server サービス パック 5 では、weblogic.Admin RESET_POOL (または同等の API) を使用して接続プールをリセットすると、接続が既に解放されている場合に次のような例外が生じていた。 `<Aug 13, 2003 1:33:11 PM EST> <Error> <HTTP> <[WebAppServletContext(7096795,jdbc_webapp,/jdbc_webapp)] Servlet failed with Exception java.lang.Error: 1 Was already released:weblogic.jdbc.common.internal.Connection` ガベージ コレクションの後、サーバは次のような接続リークの警告を表示していた。 `<Aug 13, 2003 1:33:19 PM EST> <Warning> <JDBC> <A JDBC pool connection leak was detected. A Connection leak occurs when a connection obtained from the pool was not closed explicitly by calling close() and then was disposed by the garbage collector and returned to the connection pool. The following stack trace at create shows where the leaked connection was created. Stack trace at connection create:` `at weblogic.jdbc.pool.Connection.<init>(Connection.java:55)` `at [...]` 接続リーク警告が表示されることのないようにコードが修正された。それでも、サーバはプールがリセットされる時点で接続が既に解放されている場合は初期例外を適切に表示する。

CR121671	WebLogic Server サービス パック 4 では、weblogic.jdbc.jta.ResultSet.getType() メソッドがそれ自体を再帰的に呼び出して、次のようなスタック オーバーフロー例外を引き起こす恐れがあった。
	`java.lang.StackOverflowError`
	`at weblogic.jdbc.jta.ResultSet.checkIfClosed(ResultSet.java:106)`
	`at weblogic.jdbc.jta.ResultSet.getType(ResultSet.java:507)`
	`at weblogic.jdbc.jta.ResultSet.getType(ResultSet.java:508)`
	この問題は、コードを修正して解決された。

変更要求番号	説明
CR125135	デフォルトでは、WebLogic jDriver for Oracle/XA のデータ ソースは `oracleXATrace` パラメータの値を false ではなく true に設定する。このことが原因で、`config.xml` で `oracleXATrace="false"` を設定してトレース ファイルを無効にしない限り、ドライバは時間が経つにつれて大きくなる `xa_poolname*.trc` という形式のトレース ファイルを作成していた。 値が指定されていない場合は `oracleXATrace` のデフォルト値が false に設定されるようにコードが修正された。
CR125705	JDBC マルチプールの使用時に、WebLogic Server は接続の試行時点で初期プールが完全に予約されている場合にクライアントにリソース例外を送出し、さらにバックアップ プールから接続を提供することに失敗していた。代わりに `ConnectDeadException` を送出するようにコードが修正された。マルチプールでは、この例外がリストの次のプールにフェイルオーバする根拠として解釈される。
CR127891	情報が読みやすくなるように、接続リーク ファイルの形式が修正された。

jDriver

変更要求番号	説明

CR104968	WebLogic jDriver for Oracle/XA は、Oracle RDBMS インスタンスで設定された NLS_NUMERIC_CHARACTERS パラメータを正しく処理していなかった。このことが原因で、データベースの小数点としてカンマ (‘,’) が使用されている場合、double 値または float 値を取得するときに java.sql.SQLException が生じていた。
	`java.sql.SQLException: java.lang.NumberFormatException - '1,2'`
	`at weblogic.jdbc.oci.ResultSet.getFloat(ResultSet.java:312)`
	`at weblogic.jdbc.jta.ResultSet.getFloat(ResultSet.java:190)`
	`at weblogic.jdbc.rmi.internal.ResultSetImpl.getFloat(ResultSetImpl.java:224)`
	`at weblogic.jdbc.rmi.internal.ResultSetStraightReader.getFloat(ResultSetStraightReader.java:67)`
	`at weblogic.jdbc.rmi.SerialResultSet.getFloat(SerialResultSet.java:219)`
	`at examples.servlets.DBAccess.service(DBAccess.java:54)`
	`at [...]`
	この問題は、非 XA バージョンの WebLogic jDriver for Oracle では起こらなかった。この問題は、コードを修正して解決された。

CR107474	WebLogic Server 6.1 サービス パック 3 では、close() が実行された後に OciObjectStream がバッファを破棄していなかった。この問題は、コードを修正して解決された。
CR113462	WebLogic Server サービス パック 5 は、Oracle の NLS_LANG 設定を使用した環境で BigDecimal 型を使用すると NumberFormatException を送出していた。この問題は、アメリカ式の数字の定義にあわせて Oracle の NLS_NUMERIC_CHARACTERS パラメータを使用したときには起こらなかった。この問題は、コードを修正して解決された。

JMS

変更要求番号	説明
CR103001	WebLogic Server は、メッセージング ブリッジ送り先のホスト サーバがアクセス不能な場合に過度に長い例外を表示していた。短い例外メッセージが表示されるようにコードが修正された。

変更要求番号	説明

CR104538	JMS のデッドロックは、BESession.java close(long lastSequenceNumber) メソッドを修正することで修正された。デッドロックは、次のような部分的なスレッド ダンプで調べることができた。
	`ExecuteThread: '9' for queue: 'queue_1'" daemon prio=5 tid=0x2757938 nid=0x5e waiting for monitor entry [0xce97f000..0xce97fc68]`
	`at weblogic.jms.backend.BETopic._addMessageToConsumers(BETopic.java:590)`
	`at weblogic.jms.backend.BETopic.addMessageToConsumers(BETopic.java:296)`
	`at weblogic.jms.backend.BEMessageWriteListener.storeIOComplete(BEMessageWriteListener.java:85)`
	`at weblogic.jms.store.StoreRequest.execute(StoreRequest.java:342)`
	`at weblogic.kernel.ExecuteThread.execute(ExecuteThread.java:139)`
	`at weblogic.kernel.ExecuteThread.run(ExecuteThread.java:120)`
	`"ExecuteThread: '14' for queue: 'queue_2'" daemon prio=5 tid=0x57b200 nid=0x54 waiting for monitor entry [0xd1b7f000..0xd1b7fc68]`
	`at weblogic.jms.backend.BESession.stop(BESession.java:386)`
	`at weblogic.jms.backend.BESession.close(BESession.java:434)`
	`at weblogic.jms.backend.BESession.close(BESession.java:1083)`
	`at weblogic.jms.backend.BESession.invoke(BESession.java:1024)`
	`at weblogic.jms.dispatcher.Request.wrappedFiniteStateMachine(Request.java:585)....`

変更要求番号	説明
CR112750	サーバが JMS メッセージの送信に失敗したときに、クライアントでエラー メッセージが表示されなかった。トランザクション マネージャがメッセージが異常であると宣言し、JMS はリソースの取得に失敗したときにメッセージをロールバックしていた。 トランザクションの取得の失敗はクライアントに通知されるようになった。コミットが失敗したことも報告される。
CR120619	JMS テンプレートのある JMS サーバからすべての対象を削除して、再びその JMS サーバで対象を設定しようとすると、WebLogic Server は次の例外を送出していた。 `<Aug 13, 2003 4:50:58 PM PDT> <Error> <JMS> <Failed to deploy JMS Server "server_name" due to weblogic.jms.common.JMSException: Error initializing JMSServer server_name. weblogic.jms.common.JMSException: Error initializing JMSServer server_name at weblogic.jms.backend.BackEnd.initialize(BackEnd.java:448) at weblogic.jms.JMSService.createBackEnd(JMSService.java:906) at weblogic.jms.JMSService.addJMSServer(JMSService.java:1273) at weblogic.jms.JMSService.addDeployment(JMSService.java:1169) at weblogic.management.mbeans.custom.DeploymentTarget.addDeployment(DeploymentTarget.java:364) at weblogic.management.mbeans.custom.DeploymentTarget.addDeployment(DeploymentTarget.java:150) at java.lang.reflect.Method.invoke(Native Method) [...]` この問題は、JMS サービスが一時的送り先ファクトリを RMI ランタイムにエクスポートし、そのファクトリの参照が、ファクトリが JNDI からアンバインドされたときに削除されなかったことが原因。一時的送り先ファクトリがアンバインドされたときにそのファクトリの参照が削除されるようにコードが修正された。JMS テンプレートのある JMS サーバの対象の解除と再設定で、例外が送出されることはなくなった。

変更要求番号	説明
CR122749	EJBComponentRuntimeMBean.getEJBRuntimes() を呼び出すと、ASSERTION FAILED エラーに続いて ArrayStoreException が生じる問題が修正された。

CR123675	非恒久メッセージの最適化コードの問題で、送り先が破棄されることがあった。その結果として、負荷が大きいときのメッセージのページアウトで次のような例外が送出されていた。
	`<Sep 12, 2003 9:54:12 PM EDT> <Error> <Kernel> <BEA-000802> <ExecuteRequest failed`
	`java.lang.NullPointerException.`
	`java.lang.NullPointerException`
	`at weblogic.jms.common.MessageImpl.writeExternal(MessageImpl.java:1622)`
	`at weblogic.jms.common.TextMessageImpl.writeExternal(TextMessageImpl.java:92)`
	`at weblogic.jms.store.ObjectIOBypassImpl.writeObject(ObjectIOBypassImpl.java:155)`
	`at weblogic.jms.store.BufferDataOutputStream.writeObject(BufferDataOutputStream.java:175)`
	`at weblogic.jms.store.FileIOStream.write(FileIOStream.java:506)`
	`at weblogic.jms.store.StoreRequest.doTheIO(StoreRequest.java:282)`
	`at weblogic.jms.store.JMSStore.execute(JMSStore.java:493)`
	`at weblogic.kernel.ExecuteThread.execute(ExecuteThread.java:197)`
	`at weblogic.kernel.ExecuteThread.run(ExecuteThread.java:170)`
	この問題は、コードを修正して解決された。

JNDI

変更要求番号	説明

CR105592	WebLogic Server 6.1 サービス パック 5 では、サーバが、環境をスレッドにプッシュしない ReadOnlyWrapper から環境をポップしようとしていた。このことが原因で、次の AssertionError と EmptyStackException が生じていた。
	`java.util.EmptyStackException`
	`at weblogic.utils.collections.Stack.pop(Stack.java:82)`
	`at weblogic.kernel.ResettableThreadLocalStack.pop(ResettableThreadLocalStack.java:79)`
	`at weblogic.jndi.internal.ThreadEnvironment.pop(ThreadEnvironment.java:18)`
	`at weblogic.jndi.internal.WLContextImpl.close(WLContextImpl.java:72)`
	`at weblogic.jndi.factories.java.ReadOnlyContextWrapper.close(ReadOnlyContextWrapper.java:30)`
	`[...]`
	`--------------- nested within: ------------------`
	`weblogic.utils.AssertionError: ***** ASSERTION FAILED *****[ attempt to pop from an empty stack ] - with nested exception: [java.util.EmptyStackException]`
	`at weblogic.kernel.ResettableThreadLocalStack.pop(ResettableThreadLocalStack.java:81)`
	`at weblogic.jndi.internal.ThreadEnvironment.pop(ThreadEnvironment.java:18)`
	`at weblogic.jndi.internal.WLContextImpl.close(WLContextImpl.java:72)`
	`[...]`
	ReadOnlyWrapper でポップが試行されないようにコードが修正された。

CR105761	このサービス パックには、サーバでの JNDI ルックアップのパフォーマンスを大幅に向上させる新しい JNDI キャッシング メカニズムが含まれている。

変更要求番号	説明
CR100726	WebLogic Server クラスタの単一メンバーをステートレス セッション EJB の対象に設定し、デプロイメント記述子で `home-is-clusterable` と `stateless-bean-is-clusterable` を false に設定しなかった場合に、WebLogic Server は適切に `ClassNotFoundException` をログに記録するようになった。

CR110916	コンテキストが `ejbCreate()` メソッドで作成され、`Context.close()` がそれに続いて `ejbRemove()` メソッドで呼び出された場合に WebLogic Server は `EmptyStackException` を送出していた。次に例外テキストの一部分を示す。
	`java.util.EmptyStackException`
	`at weblogic.utils.collections.Stack.pop(Stack.java:82)`
	`at weblogic.kernel.ResettableThreadLocalStack.pop(ResettableThreadLocalStack.java:79)`
	`at weblogic.jndi.internal.ThreadEnvironment.pop(ThreadEnvironment.java:18)`
	`at weblogic.jndi.internal.WLContextImpl.close(WLContextImpl.java:72)`
	`at javax.naming.InitialContext.close(InitialContext.java:476)`
	`[...]`
	この問題は、コードを修正して解決された。

JSP

変更要求番号	説明
CR088525	以前の WebLogic Server 6.1 サービス パックで、`getParameter()` によって、値が「/////」の JSP パラメータが、値が「/」であると解釈された。 この問題は、コードを修正して解決された。
CR092039	WebLogic Server は、JSP の charset 値に余計な引用符を付けた場合に `UnsupportedEncodingException` を送出していた。たとえば、次のタグは例外を送出する。 `<%@ page contentType="text/html; charset=\"Shift_JIS\"" %>` この問題は、コードを修正することで解決した。
CR093014	WebLogic Server 6.1 SP02 では、あるスクリプトレットで始まり、次のスクリプトレットで終わる Java コメントで次の例外が生じていた。 `<22.07.2002 14:12:24 CEST> <Error> <HTTP> <[WebAppServletContext(7422744,DefaultWebApp,/DefaultWebApp)] Servlet failed with Exception weblogic.servlet.jsp.JspException: (line 12): scriptlet close brace '}' unbalanced at line 12 which breaks scope '_base_service_scope_' at weblogic.servlet.jsp.ScriptletScopeLexer.mCLOSE_BRACE(ScriptletScopeLexer.java:527) at weblogic.servlet.jsp.ScriptletScopeLexer.mTOKEN(ScriptletScopeLexer.java:232) at weblogic.servlet.jsp.ScriptletScopeLexer.nextToken(ScriptletScopeLexer.java:159) at weblogic.servlet.jsp.ScriptletScopeLexer.parse(ScriptletScopeLexer.java:119) at weblogic.servlet.jsp.JspLexer.mSCRIPT(JspLexer.java:4367) at weblogic.servlet.jsp.JspLexer.mSTANDARD_THING(JspLexer.java:2173) at ....` この問題は、`ScriptletScopeLexer` が Java コメント全体をスキップするようにコードを修正して解決した。

変更要求番号	説明
CR093520	あるタイムゾーンにあるマシン上で JSP をプリコンパイルし、その同じ JSP を別のタイムゾーンにあるサーバにデプロイした場合には、WebLogic Server がその JSP を再コンパイルすることがあった。この問題は、WebLogic Server が JSP のローカル タイムスタンプ (JAR ユーティリティによって埋め込まれた) を、生成されたクラス ファイルのタイムスタンプと比較して JSP をチェックしていたことが原因。 この問題は、コンパイル時にタイム ゾーンを保存し、デプロイメント時にそのタイム ゾーンを使用して再コンパイルが必要かどうか判断することで解決した。
CR100823	JSP をコピーで更新し、そのコピーが正しく解析されない場合に、WebLogic Server はデッドロック状態に陥っていた。この問題は、2 つの別個のデッドロックが絡んでいた。それらのデッドロックは、JSP スタブ レベルで例外を送出および評価し、`getJarFiles()` で不要なスレッドの同期をなくすことで発生しなくなった。
CR102628	次の JSP コードでは、 `<jsp:plugin type="applet" code="examples.applets.PhoneBook1.class" codebase="/bea_WebLogic Server_internal/classes/DefaultWebApp@DefaultWebApp/" height="800" width="500" type="applet" jreversion="1.3.1_06" nspluginurl="http://java.sun.com/products/plugin/1.1.3/plugin-install.html"; iepluginurl="http://java.sun.com/products/plugin/1.1.3/jinstall-113-win32.cab#Version=1,1,3,0" >` 以下の HTML が生成される。 `<embed type="application/x-java-applet;version=1.3.1_06" pluginspage="http://java.sun.com/products/plugin/1.1.3/plugin-install.html"; height="800" width="500" code="examples.applets.PhoneBook1.class" codebase="/bea_WebLogic Server_internal/classes/DefaultWebApp@DefaultWebApp/">` 生成された HTML コードは Netscape で失敗した。Netscape は SUN のサイトからではなく WebLogic Server からプラグインをダウンロードしようとした。 pluginspage 行を codebase 行の前に移動すると、コードは Netscape 上で正しく動作した。 アプレット属性の順序が正しくなるようにコードを修正して、問題を解決した。

変更要求番号	説明
CR103214	compilerclass JSP パラメータを設定すると、WebLogic Server は次のように compilerclass=null メッセージを起動ログに記録していた。 JSPServlet with initArgs '[JspConfig: verbose=true,packagePrefix=jsp_servlet,-compiler= C:\61sp4\jdk131/bin/javac,compileFlags=,workingDir=C:\61sp4\wlserver6.1\config\examples\applications\examplesWebApp\WEB-INF_tmp_war_examples_examplesWebApp,pageCheckSeconds=1,superclass=weblogic.servlet.jsp.JspBase,keepgenerated=false,precompileContinue=false, compilerSupportsEncoding=true,encoding=null,defaultfilename=index.jsp,compilerclass= null,noTryBlocks=false]'> この問題は、compilerclass の設定が起動時に使用されなかったことが原因。JSP のコンパイルでは正しい設定が使われていた。サーバの起動時に適切なパラメータ値が取得されるようにコードが修正された。

変更要求番号	説明

CR105229	WebLogic Server は空の BodyTags を評価できず、次の例外が生じていた。
	`<ExecuteThread: '14' for queue: 'weblogic.kernel.Default'>`
	`<<WLS Kernel>> <>`
	`<BEA-101017>`
	`<[ServletContext(id=12216222,name=xmlxTestApp,context-path=/xmlxTestApp)]`
	`Root cause of ServletException.`
	`javax.servlet.jsp.JspException: Could not find file:`
	`java.io.FileNotFoundException: Could not find appropriate xml file`
	`at weblogicx.xml.tags.XsltTag.doEndTag(XsltTag.java:280)`
	`at jsp_servlet.__xsltprocessor._jspService(__xsltprocessor.java:195)`
	`at weblogic.servlet.jsp.JspBase.service(JspBase.java:33)`
	`at weblogic.servlet.internal.ServletStubImpl$ServletInvocationAction.run(ServletStubImpl.java:1053)`
	`at weblogic.servlet.internal.ServletStubImpl.invokeServlet(ServletStubImpl.java:387)`
	`at weblogic.servlet.internal.ServletStubImpl.invokeServlet(ServletStubImpl.java:305)`
	`at weblogic.servlet.internal.WebAppServletContext$ServletInvocationAction.run(WebAppServletContext.java:6304)`
	`at weblogic.security.acl.internal.AuthenticatedSubject.doAs(AuthenticatedSubject.java:317)`
	`at weblogic.security.service.SecurityManager.runAs(SecurityManager.java:118)`
	`at weblogic.servlet.internal.WebAppServletContext.invokeServlet(WebAppServletContext.java:361`
	`[...]`
	この問題は、コードを修正して解決された。

変更要求番号	説明

CR105772	WebLogic Server は、次のように空の要素タグを使用して空のボディ タグを指定したタグ ライブラリが JSP で使用されている場合に JSP 解析エラーを表示していた。
	`<%@taglib uri="/problem-taglib" prefix="c" %>`
	`<html>`
	`<head><title>Problem scenario</title></head>`
	`<body>`
	`<table border="1"><tr><th>Local file content</th>`
	`<th>Included file content</th></tr>`
	`<tr>`
	`<td>Cell containing data from the original JSP</td>`
	`<td><c:import url="IncludedResource.jsp" /></td>`
	`</tr>`
	`</table>`
	`</body>`
	`</html>`
	この問題は、次のように空のボディ コンテンツに終了要素タグが追加された場合には起こらなかった。
	`<td>Cell containing data from the original JSP</td>`
	`<td><c:import url="IncludedResource.jsp" ></c:import></td>`
	この問題は、コードを修正して解決された。

CR106072	`pageCheckSeconds` 属性が、JSP が最初に修正されたときに機能していなかった。この属性には、JSP ファイルに変更があって再コンパイルが必要かどうかについて WebLogic Server が確認する間隔が秒単位で設定される。 この問題は、最初に JSP が呼び出されたときに WebLogic Server が `lastStaleCheck` を設定しなかったことが原因。 `lastStaleCheck` が設定されていない場合は現在の時刻に設定されるようにコードが修正された。この修正で、JSP は不必要に再コンパイルされなくなる。
CR107042	JSP `compilerclass` パラメータの値として `com.sun.tools.javac.Main` を指定した場合、JSP へのアクセスがあると WebLogic Server は突然に終了していた。 この問題は、コードを修正して解決された。

変更要求番号	説明
CR111423	WAR ファイルにパッケージ化されている場合、アプリケーションのサブコンテキストに格納されたコンパイル済み JSP は WebLogic Server へのデプロイメントのたびに再コンパイルされていた。この問題は、JSP が展開されたアーカイブ ディレクトリからデプロイされた場合、または WAR ファイルのルートコンテキストに JSP がある場合には起こらなかった。 この問題は、`jar` と `zip` で使用されるタイムスタンプの丸め方法が異なることが原因だった。丸めの矛盾により、古い方のタイムスタンプ (1 秒差) が WAR ファイル内のクラス ファイルに記録され、サーバがクラスを再コンパイルすることになっていた。 コンパイル済み JSP クラスのタイムスタンプを 1 秒進めて JSP の再コンパイルを防止するようにコードが修正された。
CR120914	WebLogic Server のフォーム検証タグ ライブラリを使用すると、リクエスト パラメータが次以降の JSP から利用できなかった。この問題は、コードを修正して解決された。
CR127515	WebLogic Server 6.1 SP02 では、`weblogic.jspc` による `javac` エラー メッセージが日本語インストールでは壊れていた。この問題は、`DataInputStream` ではなく `InputStreamReader` を使用するようにコードを修正して解決された。

JTA

変更要求番号	説明

CR082504	Sybase jConnect/XA ドライバの使用時には、接続プールに戻された後に接続が再利用できなかった。これが原因で次の例外が生じ、結局はプールの利用可能な接続が使用されていた。
	`XA error: XAER_RMERR : A resource manager error has occured in the`
	`transaction branch start() failed on resource 'ZeusPool': XAER_RMERR : A`
	`resource manager error has occured in the transaction branch`
	`javax.transaction.xa.XAException`
	この問題は、Sybase が DDL の呼び出しのローカル トランザクションを開始し、接続がプールに戻されたときにそのトランザクションがクリーンアップされなかったことが原因。
	DDL の呼び出しで生成されたローカルの Sybase トランザクションが終了するように、接続のクリーンアップ コードが修正された。

CR091192	このサービス パックには、回復まで 5 分待つのではなく、トランザクション リソース マネージャを直ちに回復する拡張機能が含まれている。

変更要求番号	説明

CR091974	XA の回復で XID が返されない場合に、WebLogic Server 6.1 サービス パック 3 は NullPointerException を送出する恐れがあった。
	`java.lang.NullPointerException`
	`at weblogic.transaction.internal.ServerResourceInfo.rollback(ServerResourceInfo.java:721)`
	`at weblogic.transaction.internal.ServerSCInfo.rollback(ServerSCInfo.java:318)`
	`at weblogic.transaction.internal.ResourceDescriptor.rollbackXids(ResourceDescriptor.java:1160)`
	`at weblogic.transaction.internal.ResourceDescriptor.recover(ResourceDescriptor.java:1060)`
	`at weblogic.transaction.internal.ResourceDescriptor.access$9(ResourceDescriptor.java:1029)`
	`at weblogic.transaction.internal.ResourceDescriptor$1.execute(ResourceDescriptor.java:770)`
	`at weblogic.kernel.ExecuteThread.execute(ExecuteThread.java:139)`
	`at weblogic.kernel.ExecuteThread.run(ExecuteThread.java:120)`
	XID の有無をチェックして例外が送出されないようにコードが修正された。

CR098273	トランザクション ログの古いエントリが原因で、サーバの再起動時に回復のための不要なオーバーヘッドが生じることがあった。この問題は、コードを修正して解決された。

変更要求番号	説明

CR103327	トランザクションがタイムアウトになった後、2 つのスレッドが同じトランザクション ID をロールバックしようとして、次のようなエラーになる場合があった。
	`XA.end FAILED (rm=pool_name, xar=pool_name, error code: XAER_RMERR : A resource manager error has occured in the transaction branch, message: null>`
	`oracle.jdbc.xa.OracleXAException`
	`at oracle.jdbc.xa.OracleXAResource.checkError(OracleXAResource.java:659)`
	`at oracle.jdbc.xa.client.OracleXAResource.end(OracleXAResource.java:305)`
	`at weblogic.jdbc.jta.VendorXAResource.end(VendorXAResource.java:47)`
	この問題は、コードを修正して解決された。

CR103601	WebLogic Server 6.1 SP03 と SP04 では、`weblogic.transaction.internal.XidImpl` の「equals」メソッドに渡されたオブジェクトが `XidImpl` のインスタンスでない場合に (たとえば `String` 型など)、次の例外が送出される。 `java.lang.ClassCastException: java.lang.String at weblogic.transaction.internal.XidImpl.equals(XidImpl.java:114) at test.main(test.java:9)` この問題は、型の不一致をチェックして報告するようにコードを修正して解決した。

変更要求番号	説明
CR113226	リソース名が 64 文字を超えている場合、WebLogic Server 6.1 サービス パック 4 は接続プールのテスト時に次の例外を送出することができた。 `java.sql.SQLException: XA error: XAER_RMERR : A resource manager error has occured in the transaction branch start() failed on resource 'weblogic.jdbc.jta.DataSource' null at weblogic.jdbc.jta.DataSource.enlist(DataSource.java:1167) at weblogic.jdbc.jta.DataSource.refreshXAConnAndEnlist(DataSource.java:1133) at weblogic.jdbc.jta.Connection.getXAConn(Connection.java:153) at weblogic.jdbc.jta.Connection.prepareStatement(Connection.java:241) [...]` この問題は、ユニークかどうかのテストが最初の 64 文字のみに対して行われたことが原因。64 文字を超えるリソース名が正しく処理されるようにコードが修正された。
CR126201	サーバが複数のドメインでは、管理対象サーバが別のアドレスまたはポート番号を使用するために再起動された場合に、JTA サブシステムでそのアドレス情報を更新することができなかった。これが原因で、変更されたサーバが再起動されたときに次の例外が発生していた。 `javax.naming.CommunicationException.  Root exception is java.net.ConnectException: t3://ip_address:port_number: Destination unreachable; nested exception is: java.net.ConnectException: Connection refused; No available router to destination [...]` アドレスまたはポートの変更に応じて管理サーバから新しいアドレス情報が取得されるようにコードが修正された。

ノード マネージャ

変更要求番号	説明
CR104285	ノード マネージャの共有オブジェクト コードは、サーバ インスタンスの起動時に特定のコード パスがとられたときにセグメント違反を引き起こすことがあった。IBM の zLinux JDK を使用したときにも同じコード パスで問題が発生していた。これらの問題は、ノード マネージャのコード修正により解決した。
CR125829	ノード マネージャのリスン ポートに誤りのあるデータが送信されると、ノード マネージャがクラッシュして回復しなかった。これはノード マネージャの通常の処理中には発生せず、誤りのある特定のデータがポートに送信される場合にのみ発生する。この問題は、コードを修正することで解決した。 詳細については、 http://dev2dev.bea.com/resourcelibrary/advisoriesnotifications/BEA03_42.00.jsp を参照。

OA&M (操作と管理)

変更要求番号	説明
CR090118	WebLogic Server 6.1 SP03 以降では、Administration Console デプロイメント記述子エディタによって `bea\wlserver6.1` ディレクトリの下に `jarxxxx` という名前で作成される一時ディレクトリ (`bea\wlserver6.1\jar7018` など) が削除されなかった。 分析の結果、内部ファイルのオープン ファイル ストリームが原因で、一時ディレクトリが削除されていないことがわかった。 この問題は、コードを修正することで解決した。
CR093653	WebLogic Server 6.1 SP04 では、管理対象サーバにデプロイされている展開された Web アプリケーションを更新すると、管理対象サーバにステージングされている展開されたアプリケーションには更新が反映されるが、`.wlnotdelete` ディレクトリの .war ファイルには反映されなかった。サーバの再起動後に旧バージョンのアプリケーションがデプロイされた。 `weblogic.refresh` を使用してアプリケーションを更新すると、更新された .jsp は、管理対象サーバにステージングされている展開形式のアプリケーションにはコピーされるが、元のバージョンの .war にはコピーされなかった。 この問題は、ローカルのデプロイメント ファイルからアプリケーション エントリを削除するようにコードを修正することで解決した。 この変更の結果、アプリケーションの最新のバージョンは、更新の後で管理対象サーバが再起動されるときに、ソースから管理対象サーバのステージング領域にコピーされる。
CR093687	WebLogic Server SP03 で、起動クラスに対して `LoadBeforeAppDeployments=true` が設定されている場合、起動クラスが動的接続プールを作成してサーバ インスタンスに割り当てると、サーバがハングした。例外は報告されなかった。 分析の結果、同期化の問題が判明したが、コードの修正により解決された。
CR095967	WebLogic Server SP03 および SP04 で、管理対象サーバと管理サーバを同時に停止したときに、管理対象サーバが管理サーバに再接続しようとした。 この問題は、停止モードまたはサスペンド モードにあるときに、管理対象サーバが管理サーバに再接続しないようにコードを変更することで解決した。

変更要求番号	説明
CR097152	WebLogic Server SP03 および SP04 で、管理サーバを再起動すると次のようなエラーが発生した。 `Admin restart causes java.rmi.ConnectException: This RJVM has already been shutdown.` この問題は、次の手順を行うと常に発生する。 1. 2 つ以上のマシン上にクラスタ インスタンスが配置されたクラスタをコンフィグレーションする。 2. サンプルの EJB をデプロイしてクラスタに割り当てる。 3. 管理サーバを再起動する。 4. コンソールを表示して、クラスタから EJB の割り当てを解除し、適用する。次にその EJB をクラスタに再び割り当てて、変更を適用する。 この問題は、`weblogic.management.internal.MBeanProxy` が `AdminMBeanHome` の古いスタブを使用していたことと、管理サーバが再起動されたときに `AdminMBeanHome` オブジェクト インスタンス (動的プロキシ スタブ) が有効ではなくなったために発生した。 この問題は、コードを修正することで解決した。

CR099973	WebLogic Server 6.1 サービス パック 3 で、`license_update_file` を指定して `UpdateLicense` ユーティリティを使用しても、ライセンス ファイルが正しく更新されなかった。このため、サーバの起動時に次のような例外が発生した。
	`$$$$$$$$$$$$$$$$  License Exception $$$$$$$$$$$$$$$$`
	`Unable to start WebLogic Server !!`
	`WebLogic: license signature validation error!`
	この問題は、コードを修正することで解決した。

変更要求番号	説明
CR102593	WebLogic Server 6.1 SP04 で、ServletSessionRuntimeMBean を取得するときに ClassCastException が発生した。 この問題は 2 ノード クラスタで見受けられた。特定のユーザが 2 つのノードのいずれかに既にログオンしたかどうかを判別するために、アプリケーションは各クラスタ ノードの WebAppComponentRuntime から ServletSessionRuntimeMBean の配列を取得する。 呼び出し手順は次のとおり。 1. 呼び出し：ノード 1 ---> test.jsp (HTTP セッションが作成される) 2. 呼び出し：ノード 2 ---> test.jsp から (GET を介して) /ctrl/* サーブレットにデータを送信する この呼び出し手順を同じ HTTP セッションで何度か繰り返すと、WebAppComponentRuntime.getServletSessions() メソッドで ClassCastException が発生する。 `ClassCastException: java.lang.ClassCastException: $Proxy79 at weblogic.servlet.internal.session.SessionData.getServletSessionRuntimeMBean(SessionData.java:271) at weblogic.servlet.internal.session.SessionContext.getServletSessionRuntimeMBean(SessionContext.java:199) at weblogic.servlet.internal.session.SessionContext.getServletSessionRuntimeMBeans(SessionContext.java:216) at weblogic.servlet.internal.WebAppServletContext.getServletSessionRuntimeMBeans(WebAppServletContext.java:2442) at weblogic.servlet.internal.WebAppRuntimeMBeanImpl.getServletSessions(WebAppRuntimeMBeanImpl.java:126) at java.lang.reflect.Method.invoke(Native Method) at...` この問題は、ServletSessionRuntimeMBean が実装 (SessionData.java の ServletSessionRuntimeMBeanImpl) に不適切にキャストされたために発生した。 この問題は、コードを修正することで解決した。
CR102860	このリリースでは JMX のメモリ リークを解決した。このリークは WebLogic Server 6.1 SP02 で負荷テストの実行中に検出された。JMX を数日間実行する場合、ガベージ コレクション後のヒープの消費量が 17MB から 40MB に増加した。

変更要求番号	説明
CR103735	WebLogic Server SP03 で、ドメインとドメイン内のクラスタが同じ名前である場合、ドメインの config.xml ファイルにある Targets 属性のエントリが重複する。次のような状況が観察された。 アプリケーションをコンソールからクラスタに最初に割り当てたとき、問題は発生せず、対象は config.xml に正しく反映された。 管理サーバを再起動した後、Administration Console はアプリケーションがクラスタに割り当てられたことを表示せず、クラスタ内の管理対象サーバでは、アプリケーションが JNDI ツリーにバインドされたものとして示されなかった。対象はそれでも config.xml に正しく反映されていた。 ユーザが Administration Console を使用してアプリケーションをクラスタに再び割り当てると、config.xml の Targets 属性にはクラスタ名が 2 回出現した。 `<Application Deployed="true" Name="seb" Path=".\config\rom\applications"> <WebAppComponent Name="seb" Targets="rom,rom" URI="seb.war"/> </Application>` どのバージョンの WebLogic Server でも、すべての対象ドメイン、クラスタ、サーバ、および仮想ホストはユニークな名前を持つ必要があるため、この問題が発生した。 この問題は、属性が対象である場合、解決する際に weblogic.managment.internal.UnresolvedMBean.java が DomainMBean を無視するようにコードを修正することで解決された。

変更要求番号	説明
CR105338	WebLogic Server SP03 では、最大ファイル数に達すると、管理サーバと管理対象サーバの両方のロギングが停止される。 次のようなエラー メッセージがドメイン ログ ファイルに出力された後で、サーバのロギング サービスが停止する。 `####<Apr 21, 2003 3:17:39 PM GMT+09:00> <Alert> <Log Management> <KESATO01> <myserver> <ExecuteThread: '10' for queue: 'weblogic.kernel.Default'> <<anonymous>> <> <BEA-170017> <The log file .\myserver\myserver.log will be rotated. Reopen the log file if tailing has stopped. This can happen on some platforms like Windows.> ####<Apr 21, 2003 3:17:40 PM GMT+09:00> <Critical> <Logging> <KESATO01> <myserver> <ExecuteThread: '10' for queue: 'weblogic.kernel.Default'> <<anonymous>> <> <000000> <Handler: 'C:\bea81ga\mydomain\myserver\myserver.log' raised several exceptions. Shutting it down> ####<Apr 21, 2003 3:17:40 PM GMT+09:00> <Error> <Logging> <KESATO01> <myserver> <ExecuteThread: '10' for queue: 'weblogic.kernel.Default'> <<anonymous>> <> <000000> <Handler: 'C:\bea81ga\mydomain\myserver\myserver.log' raised exception when opening. Exception weblogic.logging.LogRotationException null> ####<Apr 21, 2003 3:17:40 PM GMT+09:00> <Error> <Logging> <KESATO01> <myserver> <ExecuteThread: '10' for queue: 'weblogic.kernel.Default'> <<anonymous>> <> <000000> <Handler: 'C:\bea81ga\mydomain\myserver\myserver.log' raised exception when opening. Exception weblogic.logging.LogRotationException null>` この問題が発生すると、すべてのログ メッセージがサーバ ログ ファイルに出力されなくなる。この問題は管理サーバと管理対象サーバの両方で発生する。 分析の結果、WebLogic Server はローテーション中にログ ファイルを閉じており、ログ ローテーションが失敗すると、そのログ ファイルは閉じたままになることがわかった。 WebLogic Server が新しいログ ファイルの誤ったインデックスを生成したときに、ログ ローテーションが失敗した。最新のタイム スタンプを持つファイルが最新のインデックスを持つと見なされたためである。そのインデックスを持つファイルが既に存在しているかどうかはチェックされていなかった。WebLogic Server は、ログ ファイルが既に存在する場合はログ ファイルを削除しようとせず、`rename()` が成功したかどうかを確認しなかった。 また、時間に基づいたログ ローテーションの場合、マルチ CPU マシンでは、複数のローテーションが同じミリ秒内に発生した。この問題はコードを修正することで変更された。

変更要求番号	説明
CR108448	WebLogic Server 6.1 SP05 で、管理対象サーバを再起動すると警告が発生した。管理サーバと管理対象サーバが動作していて、管理対象サーバが停止した。管理サーバは管理対象サーバが動作していないことを認識した。管理対象サーバを再起動すると、次のような警告が発行された。 <Warning> <Management> <c4wlg001> <node001> <ExecuteThread: '1' for queue: '__weblogic_admin_rmi_queue'> <system> <> <141029> <Unable to reconnect to Admin Server running at http://c4wlg001:7001 from Managed Server - node001.> weblogic.management.configuration.ConfigurationException: Server: node001 is already running. at weblogic.management.Admin.registerManagedHome2AdminHome(Admin.java:1679) at weblogic.management.Admin.reconnectToAdminServer(Admin.java:1637) at weblogic.t3.srvr.ServerRuntime.reconnectToAdminServer(ServerRuntime.java:413) at java.lang.reflect.Method.invoke(Native Method) at weblogic.management.internal.DynamicMBeanImpl.invokeLocally(DynamicMBeanImpl.java:636) at weblogic.management.internal.DynamicMBeanImpl.invoke(DynamicMBeanImpl.java:621) at... 管理対象サーバは正常に起動している。 分析の結果、不適切な時点でエラー チェックが実行されたために JNDI ツリーが適切に更新されないことがわかった。この問題は、コードを修正することで解決した。

変更要求番号	説明

CR108727	WebLogic Server SP04 で、CR071109_610sp2.jar を実行し、次のコマンドライン オプションを指定して管理サーバを起動した。
	`weblogic.management.discover.interval = 60`
	`weblogic.management.discover.retries = 6`
	`weblogic.management.internal.debug = true`
	障害の発生後 (java.rmi.ConnectException が送出される) に、管理対象サーバが検出されて正常に再接続したが、`OutOfMemoryError` が発生した。
	その後、管理対象サーバでの MBean の呼び出しは次の例外により失敗した。
	`java.rmi.NoSuchObjectException: RemoteInvokable - id: '267'`
	分析の結果、`ManagedServerReDiscoveryChecker` スレッドが実行を停止したために、管理対象サーバが検出されなかったことがわかった。この問題は、検出スレッドが必要なときに常に動作するように再試行ロジックを変更することで解決した。

変更要求番号	説明
CR127860	WebLogic Server 6.1 SP05 で、管理サーバが動作していないときに管理対象サーバで停止クラスが呼び出された。次のような例外が送出された。 `<29.10.2003 10:48:16 CET> <Notice> <WebLogicServer> <Started WebLogic Managed Server "managed1" for domain "mydomain" running in Production Mode>` `<29.10.2003 10:49:54 CET> <Alert> <WebLogicServer> <The disabling of server logins has been requested by system>` `<29.10.2003 10:49:54 CET> <Alert> <WebLogicServer> <Server logins have been disabled.>` `<29.10.2003 10:49:54 CET> <Alert> <WebLogicServer> <Server shutdown has been requested by system>` `<29.10.2003 10:49:54 CET> <Alert> <WebLogicServer> <The shutdown sequence has been initiated.> <29.10.2003 10:49:56 CET> <Emergency> <Server> <Unable to initialize the server: 'Fatal initialization exception Throwable: weblogic.rmi.extensions.RemoteRuntimeException - with nested exception: [java.rmi.ConnectException: Could not establish a connection with 462251061190312726OS:172.23.64.209:[7001,7001,7002,7002,7001,7002,-1]:mydomain:myserver, java.rmi.ConnectException: Destination unreachable; nested exception is: java.net.ConnectException: Connection refused: connect; No available router to destination] java.rmi.ConnectException: Could not establish a connection with 462251061190312726OS:172.23.64.209:[7001,7001,7002,7002,7001,7002,-1]:mydomain:myserver, java.rmi.ConnectException: Destination unreachable; nested exception is:` `java.net.ConnectException: Connection refused: connect; No available router to destination` `at weblogic.rjvm.RJVMImpl.getOutputStream(RJVMImpl.java:276)` `at weblogic.rjvm.RJVMImpl.getRequestStream(RJVMImpl.java:428)..` この問題は、管理レイヤを停止する前に停止クラスを実行するようにコードを変更することで修正された。

変更要求番号	説明
CR122538	WebLogic Server で JMX クライアント アプリケーションが `getAllMBeans()` を使用した場合、次のような例外が送出された。 `Exception in thread "main" java.lang.ClassNotFoundException: weblogic.management.descriptors.ejb11.MailServiceMBean at java.net.URLClassLoader$1.run(URLClassLoader.java:195) at java.security.AccessController.doPrivileged(Native Method) at java.net.URLClassLoader.findClass(URLClassLoader.java:183) at java.lang.ClassLoader.loadClass(ClassLoader.java:294) at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:281) at java.lang.ClassLoader.loadClass(ClassLoader.java:250) at java.lang.ClassLoader.loadClassInternal(ClassLoader.java:310) at java.lang.Class.forName0(Native Method) at java.lang.Class.forName(Class.java:190) at weblogic.management.internal.Helper.findClass(Helper.java:831) [...] --------------- nested within: ------------------ weblogic.management.configuration.ConfigurationError - with nested exception: [java.lang.ClassNotFoundException: weblogic.management.descriptors.ejb11.MailServiceMBean] at weblogic.management.internal.Helper.getAdminOrConfigMBeanInfo(Helper.java:118) [...]` この問題は、コードを修正することで解決した。

プラグイン

変更要求番号	説明
CR087204	[Apache] WebLogic Server 6.1 SP03 で、PathTrim と PathPrepend が URL の末尾に予期しない「/」を追加した。元の URL に PathTrim が追加されて「/」になった後で、PathPrepend 変数に余分な「/」が付加された。 この問題は、コードを修正することで解決した。
CR091910	[Apache] WebLogic Server 6.1 SP03 で、Apache プラグインは、<IfModule mod_weblogic.c> の使用時に PathPrepend を読み込んでいなかった。この問題は、Apache 1.3.x および Apache 2.0.43 用のプラグインで発生していた。 この問題は、コードを修正することで解決した。
CR092970	[Apache] WebLogic Server 6.1 SP04 で、Apache プラグインはソケットを CLOSE_WAIT 状態のままにした。分析の結果、WebLogic Server は sendRedirect を返すときにソケットを閉じていないことがわかった。この問題は、コードを修正することで解決した。
CR092327	[Apache] WebLogic Server 6.1 SP04 で、VirtualHosts が使用されるときに WLLogFile の設定が機能しなかった。ログ ファイルのデフォルトは /tmp/wlproxy.log。
CR100070、CR107200	[Apache] 2 つの仮想ホストが Apache サーバで使用される場合は、各仮想ホストが独自の WLLogFile を定義している。このパラメータの値は、2 つの仮想ホストに定義された WLLogFile 値の間を切り替わり続ける。 2 つの問題があった。1 つは、WebLogic Server がサーバの起動時に 1 度だけ WLLogFile を設定していたことで、もう 1 つは WebLogic Server がそれを VirtualHost から取得する前に設定していたことである。 Apache プラグインで各仮想ホストの WLLogFile を設定することで問題は修正された。

変更要求番号	説明
CR100601、CR105658、CR108747	(NSAPI) WebLogic Server 6.1 サービス パック 3 に付属のプラグインは、WL-Proxy-Client-IP ヘッダを使用してクライアントの IP アドレスを送信していた。このことは、さまざまな環境で問題を引き起こしていた。以前のサーバは、プラグインが X-Forwarded-For および Proxy-Client-IP ヘッダを使用してクライアントの IP アドレスを送信するものと想定していたからである。 WL-Proxy-Client-IP だけでなく X-Forwarded-For および Proxy-Client-IP の両ヘッダが含まれるようにコードが修正されたので、さまざまな環境で修正なしにクライアントのアドレスを取得できるようになった。
CR101937	(NSAPI) HTTP リクエストに expect-continue ヘッダが含まれている場合、プラグインのクライアントが HTTP/1.0 を使用した場合でも、プラグインは 100 (Continue) 応答を送信していた。HTTP/1.1 仕様に準拠するようにコードを修正して、クライアントが HTTP/1.0 以前を使用している場合 (または、リクエストに expect-continue ヘッダがない場合)、プラグインが 100 (Continue) ステータスで応答しないようにした。
CR102616	(NSAPI、Apache) NSAPI プラグインで、DNS サーバから IP アドレスのリストが返され、プラグインが WebLogicHost = 'DNS name' でコンフィグレーションされている場合に、DNS 名の動的な DNS ルックアップがサポートされるようになった。
CR105123	(ISAPI) すべてのバージョンの WebLogic Server で、iisforward.dll が使用されている場合に DefaultFileName が機能しない。問題は次の場合に発生した。 ■ 仮想ディレクトリがコンフィグレーションされている。 ■ MIME タイプが * (すべてをプロキシする) にコンフィグレーションされている。 ■ iisproxy.ini に DefaultFileName が追加されている。 ファイル名のない、ディレクトリに対するリクエストで、DefaultFileName が使用されない。この問題は、コードを修正することで修正された。

変更要求番号	説明
CR105173	(WLS-NSAPI) WebLogic Server 6.1 SP04 で、クライアントへの応答の送信をクライアントが止めると (応答を完全に受信する前にブラウザを閉じるなど)、Web サーバのログに 500 [WRITE TOCLIENT ERROR] が記録される。 この動作は、Web サーバの状態モニタ ツールを使用してサーバの状態を判断するクライアントには受け入れることができない。この問題は通常、リクエストから応答までにかなり時間がかかる場合に発生する。 Web サーバの状態モニタ ツールは 500 エラーを使用して、サーバの状態に何らかの問題があることを示しているが、これはサーバの状態の問題ではなく、応答を終了したクライアントの問題であるため、500 エラーではないはずである。 リクエストと応答のパスは次のとおり。 クライアント -> proxyWebserver -> プラグイン -> wls 予期される応答のパスは次のとおり。 クライアント <- proxyWebserver <- プラグイン <- wls ただし、WebLogic Server が応答を正常に送信したものの、Web サーバがその応答をクライアントに完全に送信しなかった場合に、クライアントが通信を中断すると、Web サーバの access.log には、通常はサーバに問題があることを示す 500 エラーが記録される。 そのような状況では、別のエラーが記録されるか、または何も記録されないというのがユーザの考えである。 クライアントが接続を中断した場合に 500 エラーが生成されないように、コードの変更を実装した。
CR106764	(NSAPI) プラグインのスレッドが長い期間クリティカル ロックを取得できたことで (デフォルトで 5 分、あるいは `WLIOTimeoutSecs` でコンフィグレーション)、他のすべてのスレッドがブロックされ、WebLogic Server がハングしているように見えていた。この問題は、プラグインをコード修正して解決した。
CR107254	(Apache) Hewlett-Packard では HP Apache-based Web Server バージョン 1.3.x のサポートを停止したので、WebLogic Server は HP 用の Apache 1.3 プラグインを削除した。
CR108092	(ISAPI) WebLogic Server 6.1 サービス パック 4 で、ISAPI プラグインは、修正されたクッキーに遭遇したときに未処理の例外エラーを Windows のイベント ログに記録していた。そのイベント テキストは次の行で始まっていた。 `The HTTP Server encountered an unhandled exception while processing the ISAPI Application.` この問題は、ISAPI プラグインのコード修正で解決した。

変更要求番号	説明
CR109755	[Apache] ワイルドカード文字 (*/?) 以外の正規表現が含まれるコンフィグレーション パラメータをプラグインが無視していた。このことが原因で、次のようなパラメータを使用した場合に 404 エラーが生じることがあった。 `LocationMatch "/weblogic/(abc|def)/ghi"` この問題は、コードを修正することで解決した。
CR110664	(NSAPI) プラグイン コードが例外の捕捉に失敗し、代わりに、Windows プラットフォーム上の iPlanet サーバが `sendResponse` フェーズでクラッシュした。例外を捕捉するようにプラグイン コードを変更した。
CR111167	(ISAPI) WebLogic Server 6.1 サービス パック 2 で、ISAPI プラグインを使用すると、HTTP 応答に 2 つの日付ヘッダ (WebLogic Server が挿入するヘッダと IIS が挿入するヘッダ) が含まれる。日付ヘッダの重複が原因で、単一の日付ヘッダを想定している特定のキャッシング サービスで問題が発生した。 この問題は、WebLogic Server によって挿入される日付ヘッダを除外するように ISAPI プラグインを更新することで解決された。
CR113033	(ISAPI) WebLogic Server 6.1 サービス パック 4 で、プラグインは `_wl_proxy` フォルダの `WLTempDir` フラグを認識しなかった。フラグを使用するようにコードを修正した。

CR113093	[Apache] 次のように、`httpd.conf` で複数の `MatchExpression` パラメータを使用して、さまざまな場所にリクエストをルーティングする場合、
	`MatchExpression *.jsp WebLogicHost=localhost|WebLogicPort=8001`
	`MatchExpression *.html`
	`WebLogicCluster=localhost:8001,localhost:8003`
	各リクエストが同じグローバル パラメータ情報を上書きしたために、リクエストが誤った場所に送信された。上の例では、この問題の結果 `*.jsp` リクエストがポート 8003 のサーバに送信された。
	各リクエストがパラメータ情報の独自のコピーを使用するようにコードを修正した。

CR121341	(Apache) このサービス パックでは、URL 書き換えを行うプラグインを使用する場合にサービス拒否攻撃が起こるおそれのある問題が解決されている。 http://dev2dev.bea.com/resourcelibrary/advisoriesnotifications/BEA03_39.00.jsp のセキュリティ勧告 BEA03-39.00 を参照。

変更要求番号	説明
CR121688	(Apache) URL 内の感嘆符「!」が「%21」で置換されると、プラグインがクッキーの解析に失敗した。この置換は一般に、URL 書き換えを使用するときに WAP ゲートウェイによって行われる。URL 内の文字を正しく解析するようにコードを修正した。
CR121943	[Apache] URL 内の「%1」が「%21」で置換された場合に、プラグインはクッキーを解析しなかった。この置換は、WAP ゲートウェイと URL 書き換えを使用する場合に発生した。この問題は、コードを修正することで解決した。
CR122207	(NSAPI) `KeepAliveEnabled` と `DynamicServerList` の両方が有効である場合、プラグインはソケットを `CLOSE_WAIT` 状態のままにする可能性があった。この問題は、コードを修正することで解決した。
CR122754	(ISAPI) WebLogic Server 6.1 サービス パック 4 で、MIME タイプで転送するときに、プラグイン パラメータ `WLExludeByPathOrMimeType` が機能しなかった。この問題は、コードを修正することで解決した。
CR122755	(ISAPI) WebLogic Server 6.1 サービス パック 4 で、URL に「`.wlforward`」が手動で追加されると、プラグイン フィルタが無視された。初期リクエストに `.wlforward` の MIME タイプがある場合は 404 エラーを送出するようにコードを変更した。
CR123120、CR123775	(Apache、NSAPI) プラグインを介して POST メソッドが使用され、`Content-Length` が定義されていなかった場合、プロキシ ログ ファイルには次のようなメッセージが含まれる。 `POST and PUT requests *must* contain a Content-Length` `Content-Length` が定義されていない場合はコンテンツ長をゼロ (0) に設定するようにコードを変更した。

変更要求番号	説明

CR123925	(ISAPI) プラグインがブラウザに対して 500 エラー メッセージで応答する場合があった。この問題にはさらに、以下の 3 つの兆候があった。
	
	1. IIS アクセス ログに次のメッセージが表示された。
	
	`Out-of-process+ISAPI+extension+request+failed. 500 1726 99122 2003 84078`
	
	2. Windows イベント ログはイベント ID 37 を記録した。
	
	`Event Type: Warning`
	`Event Source: W3SVC`
	`Event Category: None`
	`Event ID: 37`
	`Date:  8/26/2003`
	`Time:  6:45:03 PM`
	`User:  N/A`
	`Computer: name`
	`Description:`
	`Out of process application '/LM/W3SVC/2/Root/caf' terminated unexpectedly.`
	`For additional information specific to this message please visit the Microsoft Online Support site located at: http://www.microsoft.com/contentredirect.asp.`
	
	3. `wlproxy.log` エントリは次のとおり。
	
	`Fri Nov 21 19:06:31 2003 Write to the browser failed: calling URL::close at line 1270 of .\iisproxy.cpp`
	`Fri Nov 21 19:06:31 2003 *******Exception type [WRITE_ERROR_TO_CLIENT] raised at line 1271 of .\iisproxy.cpp`
	
	この問題は、コードを修正することで解決した。

CR124433	(ISAPI) IIS が `WlForwardPath=/` でコンフィグレーションされている場合、プラグインはサーバが停止している場合でも転送しようとした。クライアントにエラー ページは表示されなかった。この状況でパスを適切に除外するようにプラグインを修正した。
CR124464	(NSAPI) プラグインのクラッシュを引き起こす可能性のあるメモリ リークが検出された。この問題は、オブジェクトが削除された後でプラグインが例外オブジェクトにアクセスしたために発生した。例外オブジェクトから例外コードを取得した後でオブジェクトを削除し、メモリ リークが発生しなくなるようにコードを修正した。

変更要求番号	説明
CR125545	(NSAPI) クライアントへの応答の送信をクライアントが止めた場合 (応答が完了する前にブラウザを閉じるなど)、プラグインは Web サーバのログに 500 [WRITE TOCLIENT ERROR] を書き込んだ。500 エラーを Web サーバの問題として解釈する状態モニタ ツールで問題が発生することがあった。 この問題は、コードを修正することで解決した。
CR125690	(ISAPI) 9 個の IIS サーバと 9 個のクラスタ化された WebLogic Server インスタンスを含むコンフィグレーションで、IIS が数時間おきにクラッシュし、イベント ログにイベント 37 が書き込まれた。`wlproxy` ログにには次のメッセージが含まれていた。 `Thu Oct 09 13:01:46 2003 *******Exception type [WRITE_ERROR_TO_CLIENT] raised at line 1269 of .\iisproxy.cpp` 診断の結果、初期バッファの増加中に `Reader::fill()` メソッドが十分なメモリを割り当てていなかったことがわかった。バッファの最後をマークするための 4 バイトが失われており、そのためにコア ダンプが発生した。この問題は、コードを修正することで解決した。
CR126103	(NSAPI) 負荷テスト中、HP11.00 上で動作する NSAPI が 2 つの Solaris マシン (それぞれに WebLogic Server インスタンスが 3 つずつ) 上の 6 ノード クラスタにプロキシする場合、メモリ消費量が徐々に増加して、約 50 分後に `ns-httpd` プロセスがクラッシュした。 HP11.00 または Solaris では同じ負荷テストはクラッシュしなかった。 分析の結果、`proxy.cpp` のコードがネイティブ システム呼び出しの `strdup()` を使用していることがわかった。 `strdup()` はシステム メモリをプログラムのヒープ領域に割り当てる。WebLogic Server は Iplanet の FREE マクロを使用して、以前に割り当てられた領域を必要なくなった時点で解放する。FREE は `strdup()` 呼び出しで割り当てられた領域を解放しないため、メモリ リークが発生した。 FREE マクロが解放する対象を認識するように、`proxy.cpp` 内のすべてのネイティブ `strdup()` システム呼び出しを Iplanet の STRDUP マクロで置き換えることで、この問題を解決した。

変更要求番号	説明
CR126568	(NSAPI) プラグインは POST リクエスト内の %0A を正常に処理しなかった。 末尾に %0A があり、NSAPI プラグインを介して WebLogic Server に送信される POST リクエストが正常に処理されなかった。リクエストは本文のストリームに関係のないデータを追加し、ヘッダは本文の最後に出現した。WebLogic Server に直接送信されたリクエストは正しく処理されて適切に機能する。末尾の %0A を正常に処理できるプラグインが必要である。 HTTP/0.9 応答を正しく検出して処理するようにプラグインのコードを変更することで、この問題を解決した。

変更要求番号	説明

CR126982	(NSAPI) WLExcludePathOrMimeType が設定されている場合、そのファイル タイプは WebLogic Server へのリクエストから除外されるが、iplanet はそれらのファイルを代わりに処理できなかった。
	たとえば、.jpg を含む .jsp に対して次のようなリクエストがあった。
	`<Object name="test5" ppath="*/weblogic/*"> Service fn="wl_proxy" WebLogicHost="lorna" WebLogicPort="7001" PathTrim="/weblogic" Debug="ALL" DebugConfigInfo="ON" WLExcludePathOrMimeType="*.jpg" </Object>`
	.jsp に対するリクエストは WebLogic Server にプロキシされて、.jsp は .jpg なしで表示された。iplanet は jpg を処理できなかった。iplanet のアクセス ログには次のメッセージが含まれていた。
	`10.40.4.117 - - [28/Oct/2003:11:45:34 -0500] "GET /weblogic/images/logo_tm_onwt.jpg HTTP/1.1" 500 305 I get the following in wlproxy.log: Tue Oct 28 11:45:35 2003 ============================ new request ============================== Tue Oct 28 11:45:35 2003 INFO: SSL is not configured Tue Oct 28 11:45:35 2003 URI=[/hello.jsp] Tue Oct 28 11:45:35 2003 attempt #0 out of a max of 5 Tue Oct 28 11:45:35 2003 general list: trying connect to '10.40.4.117'/7001/7001 at line 1224 for '/hello.jsp' Tue Oct 28 11:45:35 2003 INFO: New NON-SSL URL Tue Oct 28 11:45:35 2003 Going to check the general server list Tue Oct 28 11:45:35 2003 WLS info : 10.40.4.117:7001 recycled? 0 Tue Oct 28 11:45:35 2003 Hdrs from Client:[accept]=[*/*] Tue Oct 28 11:45:35 2003 Hdrs from Client:[accept-language]=[en-us] Tue Oct 28 11:45:35 2003 Hdrs from Client:[accept-encoding]=[gzip, deflate] Tue Oct 28 11:45:35 2003 Hdrs from Client:[user-agent]=[Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)]`
	WLExcludePathOrMimeType を指定した場合、WebLogic Server はリクエストを処理せず、Web サーバがリクエストの処理を続行できるように Web サーバに制御を渡すはずである。
	この問題は、コードを修正することで解決した。

RMI

変更要求番号	説明
CR106281	WebLogic Server 6.1 SP04 で、大きな負荷がかかっているときにセッション Bean をレプリケートすると、ヒープ使用率が上がり、`OutOfmemoryErrors` が生じていた。 分析の結果、ガベージ コレクションが強制されても examples.ejb20.basic.statefulSession.TraderBean_5ysgq2_EOImpl オブジェクトがガベージ コレクションされないという事実が判明した。クラスタ化されたステートフル セッション Bean では、プライマリ オブジェクト `EOImpl` の強い参照が `weblogic.rmi.cluster.PrimarySecondaryRemoteObject` に維持されていた。Bean で remove は絶対呼び出されないので、EOImpl の参照が `eoMap` から削除されることはなかった。 `session-timeout-seconds` の後、パッシベーションされた Bean が削除されたときに EO をアンエクスポートするようにコードが修正された。

RMI/IIOP

変更要求番号	説明
CR112991	WebLogic Server 6.1 SP04 インスタンスから WebLogic Server 7.0 SP02 インスタンス上で動作する EJB へ IIOP 呼び出しを行うと、IIOP 接続がアイドル状態でタイムアウトしたときに例外が発生した。例外は WebLogic Server 7.0 側で発生した。WebLogic Server 7.0 が応答を返送しようとすると、接続が閉じられた。例外は 6.1 サーバには送信されず、クライアント スレッドがハングする。 WebLogic Server 7.0 SP02 の例外は次のとおり。 `<Jul 29, 2003 5:21:55 PM PDT> <Error> <IIOP> <002013> <Complete failure to send exception class java.rmi.MarshalException: java.rmi.MarshalException: IOException while sending; nested exception is: java.io.IOException: Attempt to send message on closed socket. java.rmi.MarshalException: IOException while sending; nested exception is: java.io.IOException: Attempt to send message on closed socket java.io.IOException: Attempt to send message on closed socket at weblogic.iiop.MuxableSocketIIOP.send(MuxableSocketIIOP.java:362) at weblogic.iiop.EndPointImpl.send(EndPointImpl.java:1078) at weblogic.iiop.OutboundResponseImpl.sendThrowable(OutboundResponseImpl.java:194) at weblogic.rmi.internal.BasicServerRef.handleThrowable(BasicServerRef.java:473) at weblogic.rmi.internal.BasicServerRef.postInvoke(BasicServerRef.java:440) at weblogic.rmi.internal.BasicServerRef.handleRequest(BasicServerRef.java:328) at weblogic.rmi.internal.BasicExecuteRequest.execute(BasicExecuteRequest.java:30) at weblogic.kernel.ExecuteThread.execute(ExecuteThread.java:213) at weblogic.kernel.ExecuteThread.run(ExecuteThread.java:189) >` 分析の結果、接続を閉じたというメッセージが正しく処理されていないことがわかった。このエラーは、コードを修正することで解決した。

変更要求番号	説明

CR124377	シン クライアント .jar (wlclient.jar) を使用するクライアント アプリケーションが EJB にアクセスすると、WebLogic Server が java.rmi.UnmarshalException を送出することがあった。サーバ側の例外の一部は次のとおり。
	`java.rmi.UnmarshalException: error unmarshalling arguments; nested exception is:`
	`java.io.IOException: Serializable readObject method failed internally.`
	`java.rmi.UnmarshalException: error unmarshalling arguments; nested exception is:`
	`java.io.IOException: Serializable readObject method failed internally at com.ejb_cvps36_EOImpl_WLSkel.invoke(Unknown Source)`
	`[...]`
	クライアント側の例外の一部は次のとおり。
	`java.rmi.MarshalException: CORBA MARSHAL 0 No; nested exception is:`
	`org.omg.CORBA.MARSHAL: vmcid: 0x0 minor code: 0 completed: No at com.sun.corba.se.internal.iiop.ShutdownUtilDelegate.mapSystemException(ShutdownUtilDelegate.java:97)`
	`at javax.rmi.CORBA.Util.mapSystemException(Util.java:65)`
	`[...]`
	この問題は、クライアントで weblogic.jar を使用する場合は発生しなかった。コードを修正してこの問題に対応した。

CR128594	このサービス パックより前では、WebLogic Server は AIX 上で Java オブジェクトを読み込むときに typecode を処理できなかった。エイリアス ラッパーを破棄するようにコードを修正した。
CR128660	WebLogic Server は解釈できない着信コンテキストに対してリフレクションを使用した。その結果、AIX 上の IIOP クライアントは失敗した。IBM ORB はサービス コンテキストに基づいてストリームの形式を想定するためである。IIOP クライアントを AIX 上で使用できるように、サービス コンテキストを正しく処理するようにコードを修正した。

セキュリティ

変更要求番号	説明
CR067670	カスタム セキュリティ レルムを使用するように WebLogic Server をコンフィグレーションした場合、レルムが使用できないと、WebLogic Server は起動しなかった。このサービス パックでは新しい起動コマンド `-Dweblogic.security.RealmFailureOk=true` を導入した。このコマンドを使用すると、レルムが使用できない場合にもサーバが起動できる。このコマンドの場合、サーバはコンフィグレーション済みのカスタム レルムではなくファイル レルムを使用して起動する。 このコマンドは WebLogic Server 6.x でのみ使用できるもので、他のバージョンのサーバには実装されていない。 **警告：** このコマンドを使用する場合、管理者は注意する必要がある。カスタム レルムが (認証ではなく) 認可に対してのみコンフィグレーションされていて、ユーザ ストアがファイル レルムである場合、`-Dweblogic.security.RealmFailureOk=true` を指定してサーバを起動するとセキュリティで保護されない可能性がある。 このオプションは Administration Console にアクセスするときにのみ使用し、サーバが正常に起動できるようにカスタム レルムを再コンフィグレーションする。 `-Dweblogic.security.RealmFailureOk=true` を指定して起動したサーバにアプリケーションがアクセスしないようにすること。

CR093292	WebLogic Server サービス パック 5 に CR093292 パッチを適用すると、サーバは次のようなメッセージをログに記録し始めた。
	`java.io.IOException: Length is Zero`
	`at weblogic.security.SSL.GenericCipher.input(GenericCipher.java:196)`
	`at weblogic.security.SSL.SSLCiphertext.input(SSLCiphertext.java:68)`
	`at weblogic.security.SSL.SSLSocket.getRecord(SSLSocket.java:1147)`
	`[...]`
	これらのメッセージはデバッグに利用することのみを目的としていた。メッセージがログに記録されないようにコードを修正した。

変更要求番号	説明
CR093813	ノード マネージャ キー ファイル内の暗号化されたプライベート キーへアクセスするためのパスワード `weblogic.nodemanager.keyPassword` が、プレーン テキストでアクセス可能になっていた。 この問題は、ノード マネージャ プロパティ ファイル `nodemanager.properties` を作成することで解決した。

CR100703	別のユーザがロック アウトされた後に、認証済みユーザが WebLogic Server 6.1 SP04 にアクセスできなくなる可能性があった。この問題は Novell eDirectory で発生する可能性があった。あるユーザが誤ったパスワードを何度も入力してロック アウトされた後に、他の認証済みユーザがサーバにアクセスできなくなり、次のような 500 エラーを受け取った。
	`<[WebAppServletContext(7814505,security,/security)] Servlet failed with Exception`
	`netscape.ldap.LDAPException: error result (53); NDS error: login lockout`
	`(-197);`
	`DSA is unwilling to perform`
	`[...]`
	この問題は、このような場合に送出される `LDAPException` を捕捉して、LDAP サーバへの接続を破棄することで修正された。

CR102452	クロスサイト スクリプティングの問題が解決された。http://dev2dev.bea.com/resourcelibrary/advisoriesnotifications/SA_BEA03_36.00.jsp のセキュリティ勧告 BEA03-36.01 を参照。
CR102712	HTTP トンネリングを使用したクライアントと比較すると、HTTPS トンネリングを使用して EJB にアクセスした Java クライアントはパフォーマンスが低下し、ソケットの使用率が増加した。WebLogic Server はソケットを再利用しないでリクエストごとに新しいソケットを作成した。この問題は、コードを修正することで解決した。
CR105443	クロスサイト スクリプティングの問題が解決された。Security Advisory BEA03-36.00 を参照。

変更要求番号	説明
CR105513	このサービス パックには新しいプロパティ `weblogic.security.SSL.trustManager` がある。このプロパティを使用すると、`HttpsURLConnection` を使用して別のサーバに接続している場合に、カスタム トラスト マネージャを指定できる。このプロパティを使用すると、ビジネス ロジックを目的として、SSL ハンドシェーク中に CA ルートを取得できる。 **警告：** この機能は WebLogic Server 6.1 のみで実装されており、WebLogic Server 7. x 以降のリリースでは使用できない。
CR108265	WebLogic Server 6.1 SP04 で、`CachingRealm.refresh()` が呼び出されると HTTP リクエストが失敗する場合があった。このサーブレットはキャッシュとデータベースを同期させる。クラスタは基本レルムとして RDBMS レルムを使用した。 Web アプリケーションの `index.html` を呼び出す HTTP リクエストの簡単な状態チェックが、次の例外により失敗する場合があった。 `<Error> <HTTP> <hanptl02> <managedServer2> <ExecuteThread: '9' for queue: '__weblogic_admin_rmi_queue'> <> <> <101020> <[WebAppServletContext(1014083,healthcheck02,/healthcheck02)] ƒT_[ƒuƒŒƒbƒg,Í Exception ,É,æ,è_„”s,µ,Ü,µ,½_B> java.lang.NullPointerException: Start server side stack trace: java.lang.NullPointerException at weblogic.security.acl.GroupImpl.addMember(GroupImpl.java:46) at weblogic.security.acl.OwnerImpl.<init>(OwnerImpl.java:32) at weblogic.security.acl.AclImpl.<init>(AclImpl.java:164) at weblogic.servlet.security.internal.SecurityModule.auditPerm(SecurityModule.java:358)`  分析の結果、2 つのスレッドが `CachingRealm` に同時にアクセスしていたことがわかった。たとえば、更新サーブレットがキャッシュをクリアする一方で、`index.jsp` へのリクエストはそのキャッシュからクリアされたユーザを見つけようとした。 この問題は、コードを修正することで解決した。
CR111041	Web アプリケーションが HTTPS 接続を確立しようとして、リモート サーバとのハンドシェークに時間がかかった場合に、WebLogic Server は関連付けられた SSL ソケットをタイム アウトしなかった。実行スレッドは無期限に「スタック」状態になり、結局サーバがロックされた。HTTPS 接続でタイムアウト値が強制されるようにコードを修正した。

変更要求番号	説明
CR112563	WebLogic Server はパスワードをファイルに格納するときにプライベート キー パスワードを暗号化しなかった。パスワードをファイルに書き戻すときにパスワードが自動的に暗号化されるようにコードを修正した。最初の起動時に、WebLogic Server はパスワードが暗号化されているかどうかを検証し、必要な場合はパスワードをファイル内で暗号化する。 WebLogic Server はドメインに関連付けられた暗号化サービスを使用する。つまり、パスワード ファイルは、そのファイルが作成されたドメインでのみ使用できる。 **注意：** WebLogic Server がパスワードをファイルに書き込む前に、プライベート キー パスワードのプレーン テキストのコピーを保護すること。このサービス パック レベルでは、サーバの起動後にファイルからプレーン テキストの パスワードを取得することはできない。
CR113459	WebLogic Server 6.1 SP05 で、`CR093813_61sp5.jar` において、ノード マネージャ プロパティ ファイルを削除すると問題が発生した。 ノード マネージャ プロパティ ファイルが、常に `<saved logs dir>/NodeManagerInternal` ディレクトリに作成されていた。このユーザは `NodeManagerInternal` の内容を定期的にアーカイブして削除していた。ノード マネージャ プロパティ ファイルが削除されるため、ノード マネージャ プロパティ ファイルに格納されている証明書パスワードが解読できず、ノード マネージャが正しく動作しなかった。 この問題は、ノード マネージャ プロパティ ファイルが `NodeManagerInternal` または `user.dir` に見つからない場合には、`user.dir` で指定したディレクトリに作成されるようにコードを修正することで解決した。

変更要求番号	説明
CR120850	WebLogic Server 6.1 SP05 で、`weblogic.net.http.HttpsURLConnection` は `https.nonProxyHosts` 環境変数を受け付けなかった。問題は次のシナリオで発生した。 クライアント <--> プロキシ <---> サーバ クライアントは WebLogic Server の内部で動作している場合も、スタンドアロンの Java プログラムの場合もある。リクエストは、対象のホストが `https.nonProxyHosts` で指定されている場合でも、常にプロキシ経由になった。代わりに、ホストが `http.nonProxyHosts` で指定されている場合には、問題は発生しない。`proxyHost` ではなくホストへの直接接続が確立される。 分析の結果、`proxyHost` が定義されている場合でも `https.nonProxyHosts` で指定されたホストへ直接接続するロジックがあることが明らかになった。この問題は https.nonProxyHosts でのみ発生し、http.nonProxyHosts の場合は発生しなかった。 `proxyHost` が定義されている場合でも、`https.nonProxyHosts` で指定されたホストへ直接接続するために、適切なロジックを開発した。
CR121206	WebLogic Server 6.1 SP05 で、外部 LDAP サーバを使用するときに、開いている LDAP 接続の数が増加し続けた。 この問題は、コードを修正することで解決した。

CR121920	WebLogic Server 6.1 サービス パック 3 で、WebLogic Server JSSE を使用して `Socket.getSendBufferSize()` を呼び出すと次のようなエラーが発生した。
	`java.net.SocketException: Socket closed`
	`at java.net.PlainSocketImpl.socketGetOption(Native Method)`
	`at java.net.PlainSocketImpl.getOption(PlainSocketImpl.java:190)`
	`at java.net.Socket.getSendBufferSize(Socket.java:527)`
	`[...]`
	`getSendBufferSize()` を適切にオーバーライドして委託するようにコードを修正した。

CR121921	`LDAPDelegate.groupContainsInternal()` メソッドは、グループのリストに目的のグループが含まれているかどうかを最初にチェックしないで、リストの再帰的検索を実行した。このパフォーマンスの問題は、`groupContainsInternal()` がまずグループ リストを検索して、目的のグループが含まれているかどうかを判別するように更新することで解決された。

サーブレット

変更要求番号	説明
CR080998	SP06 より前のバージョンの WebLogic Server 6.1 は、HTTP/1.1 仕様のセクション 14.35 に記載されている Range ヘッダ フィールドの定義をサポートしていなかった。この問題は、コードを修正することで解決した。
CR084455	WebLogic Server 6.1 SP03 が、すべての HTTP リクエストを拡張フォーマットでログに記録するようにコンフィグレーションされている場合、ローテーション時に、ローテーションがスケジュールされてから約 1 分間隔 (ログのフラッシュ間隔の設定) で、次のようなエラーが送出される。 `####<Aug 20, 2002 9:00:00 PM PDT> <Error> <HTTP myapp> <host2> <myserver> <ExecuteThread: '6' for queue: 'default'> <system> <> <000000> <Exception flushing HTTP log file> java.io.IOException: Failed to rename log file on attempt to rotate logs at weblogic.servlet.logging.LogManagerHttp.rotateLog(LogManagerHttp.java:168) at weblogic.servlet.logging.LogManagerHttp.access$2(LogManagerHttp.java:148) at weblogic.servlet.logging.LogManagerHttp$RotateLogTrigger.trigger(LogManagerHttp.java:432) at weblogic.time.common.internal.ScheduledTrigger.executeLocally(ScheduledTrigger.java:238) at weblogic.time.common.internal.ScheduledTrigger.execute(ScheduledTrigger.java:229) at weblogic.time.server.ScheduledTrigger.execute(ScheduledTrigger.java:69) at weblogic.kernel.ExecuteThread.execute(ExecuteThread.java:139) at weblogic.kernel.ExecuteThread.run(ExecuteThread.java:120) ####<Aug 20, 2002 9:00:05 PM PDT> <Error> <HTTP myapp> <host2> <myserver> <ExecuteThread: '7' for queue: 'default'> <system> <> <000000> <Exception flushing HTTP log file>...` 分析の結果、ログが不適切にフラッシュされていたことがわかった。この問題は、ログがローテーション時にフラッシュされないようにし、ログ ファイル名に null 値がないかどうかチェックするようにコードを変更することで、解決された。
CR086234	`FileServlet` の `indexDirectories` 機能はインターナショナライズされていなかったため、マルチバイトのファイル名を表示または使用できなかった。この問題は、コードを修正することで解決した。

変更要求番号	説明
CR088785	HTTP アクセス ログで、`cs-uri` が指定されている場合、WebLogic Server は URI の基本部分とクエリ部分の両方ではなく、基本部分のみを記録していた。この問題は、コードを修正することで解決した。

CR092625	HTTP ロギングが無効な状態で起動されたためにログ ファイルが存在しない管理対象サーバで HTTP ロギングを有効にすると、WebLogic Server によって `NullPointerException` が送出された。管理対象サーバへの HTTP アクセスのたびに、次の例外が送出された。
	`java.lang.NullPointerException`
	`at weblogic.servlet.logging.LogManagerHttp.log(LogManagerHttp.java:292)`
	`at weblogic.servlet.internal.HttpServer.log(HttpServer.java:865)`
	`at weblogic.servlet.internal.ServletResponseImpl.send(ServletResponseImpl.java:1044)`
	`at weblogic.servlet.internal.ServletRequestImpl.execute(ServletRequestImpl.java:2265)`
	`at weblogic.kernel.ExecuteThread.execute(ExecuteThread.java:139)`
	`at weblogic.kernel.ExecuteThread.run(ExecuteThread.java:120)`
	この問題は、コードを修正することで解決した。

CR095945	Unix 上で動作する WebLogic Server 6.1 SP05 で、WAR 内の CGI スクリプトを実行するために抽出する `CGIServlet` が、現在の作業ディレクトリを設定しないでスクリプトを呼び出していた。 CGI スクリプトが WAR Web アプリケーションに含まれていてもサブスクリプトを呼び出せるように、現在の作業ディレクトリを設定するような修正を実装した。

変更要求番号	説明
CR095981	WebLogic Server 6.1 SP01、SP02、SP03、および SP04 で、`precompile=true` を指定して JSP ファイルをコンパイルする場合、`weblogic.xml` の `<charset-mapping>` が無視されていた。 結果として、一部の 2 バイト文字セットが、`<charset-mapping>` で指定された文字のエンコーディングとコンパイル済みクラスで指定された文字のエンコーディングの不一致のために取り違えられていた。 分析の結果、プリコンパイラにエラーがあることが判明した。この問題は、コードを修正することで解決した。
CR096459	HTTP 1.0 でサーブレットまたは JSP 内のフラッシュを呼び出すとソケットがうまく閉じない場合があり、ソケットがタイムアウトになるまでクライアントが待機する原因となっていた。この問題は、コードを修正することで解決した。
CR100590	`HttpClusterServlet` 実装が更新されて、`WebLogicCluster` パラメータのデフォルトのポート番号が 443 に変更された。更新後の実装は、WebLogicCluster パラメータが正しく指定されていない場合にエラーを報告しなかった。 `HttpClusterServlet` 実装が以前の以下の動作と一致するように、コードを修正した。 ■ WebLogicCluster は SSL ポート番号の指定に *`host`*`:`*port*`:`*sslport* という形式を使用する。 ■ WebLogic Server は、SSL ポートが指定されていない場合にエラーをログに記録し、デフォルト SSL ポートの 7002 を使用する。
CR100645	`attributeReplaced()` を介して受け取ったイベントに対して `HttpSessionBindingEvent.getValue()` を呼び出す場合、`HttpSessionAttributeListener` インタフェースを実装するオブジェクトが正しい値を受け取らなかった。サーブレット 2.3 仕様に記載されているように古い属性値を返すのではなく、`getValue()` は新しい方の値を返した。この問題は、コードを修正することで解決した。

変更要求番号	説明

CR101061	このサービス パックより前は、WebLogic Server インスタンスにパッチが適用されている場合、`weblogic.version` を呼び出すと、正しいバージョン情報が表示されなかった。`weblogic.version` が正しいバージョンおよびパッチ情報を次の形式で表示するようにコードを変更した。
	`WebLogic Server version date build with patch [, patch] [...]`
	次に例を示す。
	`WebLogic Server 8.1 03/20/2003 246620 with CRXXXX, CRXXXX, CRXXXXX`

CR101838	WebLogic Server SP04 で、`CGIServlet` の動作が変更されていたため、ファイル名に拡張子がない CGI スクリプトの実行に影響するおそれがあった。たとえば、ファイル `web.xml` で CGI ディレクトリを定義し、サーブレット マッピングを次のようにしたとする。
	`<param-name>cgiDir</param-name>`
	`<param-value>/home/user/cgi-bin</param-value>`
	`</init-param>`
	`</servlet>`
	`<servlet-mapping>`
	`<servlet-name>CGIServlet</servlet-name>`
	`<url-pattern>/cgi-bin/*</url-pattern>`
	`</servlet-mapping>`
	この場合、`myscript` というスクリプトを `/home/user/cgi-bin` ディレクトリに格納すると、`http://localhost/mywebapp/cgi-bin/myscript` のような URL は、スクリプト名を指定しない場合 `/home/user/cgi-bin` にマップされた (この問題は、`myscript.ksh` のように拡張子を付けたスクリプトでは発生しなかった)。
	`CGIServlet` の動作が以前のリリースと一致するようにコードを修正した。上記の例であれば、URL `http://localhost/mywebapp/cgi-bin/myscript` は `/home/user/cgi-bin/myscript` にマップされるようになった。

変更要求番号	説明

CR102262	null の HTTP-Version フィールドを含む HTTP リクエストを解析した後で、WebLogic Server は次のような例外を送出した。
	`java.lang.NullPointerException`
	`at weblogic.servlet.internal.ServletRequestImpl.initInputEncoding(ServletRequestImpl.java:727)`
	`at weblogic.servlet.internal.ServletRequestImpl.mergePostParams(ServletRequestImpl.java:565)`
	`at weblogic.servlet.internal.ServletRequestImpl.parseQueryParams(ServletRequestImpl.java:489)`
	`at weblogic.servlet.internal.ServletRequestImpl.getParameter(ServletRequestImpl.java:686)`
	`at weblogic.servlet.internal.ServletRequestImpl.initSessionInfo(ServletRequestImpl.java:1481)`
	`at weblogic.servlet.internal.ServletRequestImpl.getSession(ServletRequestImpl.java:1345)`
	`at weblogic.servlet.internal.ServletContextImpl.invokeServlet(ServletContextImpl.java:1010)`
	`at weblogic.servlet.internal.ServletContextImpl.invokeServlet(ServletContextImpl.java:910)`
	`at weblogic.servlet.internal.ServletContextManager.invokeServlet(ServletContextManager.java:279)`
	`at weblogic.socket.MuxableSocketHTTP.invokeServlet(MuxableSocketHTTP.java:403)`
	`at weblogic.socket.MuxableSocketHTTP.execute(MuxableSocketHTTP.java:285)`
	`at weblogic.kernel.ExecuteThread.run(ExecuteThread.java:130)`
	リクエストで指定されていない場合はデフォルト バージョンとして HTTP/0.9 を使用するようにコードを修正した。

変更要求番号	説明
CR102574	サーブレットのクライアントを強制的に停止すると、WebLogic Server でメモリ使用率が上昇した。これは `OutOfMemoryError` になる可能性がある。この問題は、コードを修正することで解決した。
CR102689	クロスサイト スクリプティングの潜在的な脆弱性に対応するため、生成後のエラー ページを調べて修正した。クロスサイト スクリプティングの問題の詳細については、http://dev2dev.bea.com/resourcelibrary/advisoriesnotifications/SA_BEA03_36.00.jsp のセキュリティ勧告 BEA03-36.01 を参照。
CR102769	拡張ログ フォーマットを使用して HTTP トランザクションをログに記録する場合、WebLogic Server は、あるサーブレットから別のサーブレットまたは JSP から別の JSP へ送信されたリクエストのクエリ パラメータ (`cs-uri-query`) を記録しなかった。また、WebLogic Server はクライアントの元のリクエスト URI ではなく、転送されたリクエストの URI (`cs-uri-stem`) を記録した。 転送される HTTP トランザクションをログに記録する場合、元のリクエストの URI とクエリ文字列を `access.log` に記録するようにコードを修正した。
CR103059	`<servlet-mapping>` 要素の 1 文字の `<url-pattern>` 値は、`web.xml` デプロイメント記述子で適切に処理されていなかった。たとえば、`<url-pattern>*.f</url-pattern>` では `welcome.f` にアクセスしようとしたときに 404 File Not Found エラーが生成されたが、`<url-pattern>*.oop</url-pattern>` は `welcome.oop` にアクセスしようとしたときに適切に機能していた。 この問題は既に解決されており、1 文字の `<url-patterns>` は適切に機能するようになっている。
CR103256	コードを変更したことで、JDBC の `JDBCSessionContext` および `JDBCSessionData` 内のバブル キャッシュに関係するパフォーマンスが向上した。
CR103289	`HttpClusterServlet` は POST データのセッション ID を正しく解析していなかった。 `HttpClusterServlet` を介してリクエストをクラスタに送信し、セッションを確立してから、クッキーがなく、POST パラメータにセッション ID のある 2 番目のリクエストを送信すると、サーブレットはセッションを認識しなかった。 サーブレットは POST データからセッション ID 抽出できるようになった。

変更要求番号	説明
CR103339	サーブレットの出力において、Transfer-Encode の値に「Chunked」と大文字が使用されていた。HTTP/1.1 仕様では小文字で「chunked」とする必要がある。コードを修正したことで、Transfer-Encode タイプが小文字で「chunked」と出力されるようになった。
CR103925	setAttribute メソッドは、hashCode の同一性のみをチェックしていた。現在は、equals メソッドの結果で古いオブジェクトと新しいオブジェクトをチェックするようにもなった。
CR104975	ログ ローテーションが失敗すると、WebLogic Server はログ ファイルを再び開けなかった。この場合、ログ ファイルをフラッシュするトリガが java.io.IOException: Bad file descriptor を送出する可能性があった。この問題は、コードを修正することで解決した。
CR105016	HttpClusterServlet はクラスタ内の WebLogic Server がハングした場合に障害の回数を増やすことができなかった。そのため、クラスタ内の各サーバが HungServerRecoverSeconds よりも長い時間スリープするか、クラスタ内の 3 つのサーバのうち 2 つが HungServerRecoverSeconds より長くスリープした場合、HttpClusterServlet は無限ループに陥った。障害の回数が適切にインクリメントされるようにコードを修正した。
CR105339	サービス パック 5 で、WebLogic Server は TagExtraInfo クラスを使用したタグ呼び出しの変数定義を作成しなかった。(サービス パック 4 以前と同様に) 変数定義が作成されるようにコードを修正した。
CR106186	サーブレットでバッファ サイズをゼロに設定すると、サーブレットは応答に失敗し、サーバで無限ループを開始した。WebLogic Server は最後に次のような警告を表示した。 "Suspend Checker Thread" prio=10 tid=0x23eb90 nid=0xfec runnable <May 14, 2003 10:53:34 AM PDT> <Warning> <WebLogicServer> <000337> <ExecuteThread: '10' for queue: 'default' has been busy for "1,181" seconds working on the request "Http Request: servlet_uri", which is more than the configured time (StuckThreadMaxTime) of "600" seconds.> この問題は、コードを修正することで解決した。

変更要求番号	説明

CR107419	JSP コードでパラメータの値として 2 バイト文字を設定すると問題が発生していた。たとえば、次のようなコードでは、
	`<jsp:include page="included.jsp">`
	`<jsp:param name="title"`
	`value="<%= java.net.URLEncoder.encode(\"[double byte characters]\")`
	`%>"/>`
	`</jsp:include>`
	2 バイト文字が URL エンコーディングされたコードに変換されていた。この問題は修正された。

CR108607	アプリケーションが XSL で FOP 出力メソッドを使用してアーカイブ ファイルから画像を取得すると、WebLogic Server がエラーを生成していた。この問題は、展開形式でデプロイされたアプリケーションでは発生していなかった。この問題は、コードを修正することで解決した。
CR108350	Administration Console は、フォーマットを共通ログ フォーマット (CLF) と拡張ログ フォーマット (ELF) の間で動的に変更できると不適切に表示した。実際には、そのような変更の場合にはサーバの再起動が必要である。このコンフィグレーション パラメータを変更するときはサーバの再起動が必要であることを正しく表示するように、コードを変更した。
CR109958	プロキシ サーブレットを使用してリクエストを処理する場合に、WebLogic Server は HTTPS を使用する受信リクエストでのみ `SecureProxy` 設定を尊重していた。受信リクエストが HTTP を使用する場合、プロキシ サーブレットで `SecureProxy` が有効になっていても WebLogic Server は HTTPS 接続を使用しなかった。この問題は、コードを修正することで解決した。
CR110798	JSP で `weblogic.servlet.security.ServletAuthentication.killCookie(req)` を呼び出すと、セッションはクリーンアップされずに WebLogic Server に保持された。`killCookie(req)` が完了する直前にセッションが無効になるようにコードを修正した。
CR110914	`TrackingEnabled` が false に設定されていると要求ごとに新しいセッションが作成されるが、それらのセッションが無効にされていなかった。 コードを修正したことで、`TrackingEnabled` が false に設定されているとセッションが即座に無効になるようになった。これが正しい動作である。

変更要求番号	説明

CR111752	特定の状況では、useByteStream パラメータが true に設定されている状態で CGIServlet を使用するときに WebLogic Server は NullPointerException を送出していた。この問題は、1 つのフレームに静的 URL リンクが含まれ、別のフレームで CGIServlet が使用されているフレームセットの使用時に発生していた。もう一方のフレームがページのダウンロードを完了する前に静的リンクの含まれているフレームをユーザが選択すると、IO 例外が捕捉され、次のように表示されていた。
	`java.lang.NullPointerException`
	`at`
	`weblogic.utils.Executable$Drainer.run(Executable.java:366)`
	この問題は、コードを修正することで解決した。

CR112799	IOException が発生した後にも、WebLogic Server が出力ストリームに書き込みをしようとしていた。このため、IOException を処理しない Web アプリケーションで予期せぬソケットの切断が発生すると、CPU 使用率が 100% になるおそれがあった。 org.apache.xml.serialize.XMLSerializer は、そのプロセスが終了するまで IOException を無視する。このため、HTTP 応答の一部として XML ドキュメントを返している最中に IOException が送出されると、この問題が発生していた。 コードを修正したことで、IOException が発生した後は WebLogic Server が出力ストリームへの書き込みを停止するようになった。
CR120440	シングル サインオン コンフィグレーションに複数の Web アプリケーションがデプロイされていて、あるアプリケーションが weblogic.servlet.security.ServletAuthentication.invalidateAll(request) を呼び出した場合、他のアプリケーションの HttpSessionListeners は、セッション タイムアウトが発生するまで呼び出されなかった。これは、最初の Web アプリケーションに関連付けられたセッションのみが無効化の対象として登録されていたために発生した。ユーザが認証された後、以降のセッションは登録されなかった。 すべての Web アプリケーションのセッション ID とコンテキスト パスの両方を、invalidateAll(request) の必要に応じて無効化の対象として登録するように、コードを修正した。
CR122177	サービス パック 3 の CR060023 の修正が原因で、FileServlet は、ファイルが見つからない場合に応答コード 404 ではなく 200 を返した。ファイルが見つからない場合に 404 を返すようにコードを修正した。

変更要求番号	説明
CR121846	WebLogic Server サービス パック 3 で、サーバは、拡張ログ フォーマット ヘッダを書き込む前に、標準のログ エントリをログ ファイルに書き込むことはできなかった。この状況は、ログ ローテーション中に、複数のスレッドが同時に新しいログ ファイルに書き込もうとする場合に発生する可能性があった。 ログ ローテーションを処理するスレッドが、ログ ヘッダが書き込まれるまで、新しいログ ファイルへ排他的にアクセスするようにコードを修正した。

変更要求番号	説明

CR125718	WebLogic Server サービス パック 5 は、Apache および Netegrity の SiteMinder と一緒に使用すると、NullPointerException を送出する可能性があった。Apache から SiteMinder に転送されて、SiteMinder によって認証された最初のリクエストは、WebLogic Server でも正常に認証される。ただし、ユーザがブラウザの [戻る] ボタンを使用してログイン ページに戻り、別のユーザ名とパスワードを使用して再度認証を受けると、WebLogic Server は NullPointerException を送出する。
	`java.lang.NullPointerException`
	`at weblogic.servlet.security.internal.SecurityModule.logoutSession(SecurityModule.java:386)`
	`at weblogic.servlet.security.internal.SecurityModule.checkAuthenticate(SecurityModule.java:292)`
	`at weblogic.servlet.security.ServletAuthentication.weak(ServletAuthentication.java:353)`
	`at com.uprr.security.weblogic.SiteMinderAuthFilter.doPreAuth(Unknown Source)`
	`at weblogic.servlet.security.AuthFilter.service(AuthFilter.java:51)`
	`at weblogic.servlet.internal.ServletStubImpl.invokeServlet(ServletStubImpl.java:262)`
	`at weblogic.servlet.internal.ServletStubImpl.invokeServlet(ServletStubImpl.java:198)`
	`at weblogic.servlet.internal.RequestDispatcherImpl.include(RequestDispatcherImpl.java:530)`
	`at weblogic.servlet.internal.RequestDispatcherImpl.include(RequestDispatcherImpl.java:350)`
	`at weblogic.servlet.security.internal.ServletSecurityManager.checkAccess (ServletSecurityManager.java:144)`
	`[...]`
	この問題は、コードを修正することで解決した。

SNMP

変更要求番号	説明
CR088000	WebLogic Server 6.1 SP04 で、SNMP 停止トラップが受信されなかった。 `8-Oct-02 15:57:24 FAILURE mytest!0[41]!oam/systest/snmp/oam_snmp!2[69]!if!3[86]!weblogic.qa.tests.oam.systest.snmp.trap.SNMPTrapTest.testServerShutdownTrap() wlsServerShutdownTrap wasn't received. PARAMETERS: managedServerName = oamserver1 adminServerName = adminServer` 分析の結果、リスン アドレスまたはリスン ポートがコマンドライン オプションでオーバーライドされている場合、SNMP トラップが管理対象サーバに対して生成されないことがわかった。 この問題は、URL のリスン ポート値ではなく ServerMBean のリスン ポート値を使用するようにロジックを変更することで解決された。

Web サービス

変更要求番号	説明
CR099255	双方向 SSL を使用して Web サービスを呼び出し、クライアント コードが 呼び出しで Web サービスの WSDL を使用する際に、クライアント API は、WSDL のルックアップを行うときにも操作を呼び出すときにも、証明書を正しく送信するようになった。以前は、証明書は WSDL ルックアップでのみ送信され、操作を実際に呼び出すときには送信されなかった。 この問題は、コードを修正することで解決した。
CR099534	`wsgen` Ant タスクを最適化して、1 つの `build.xml` ファイルからアセンブルする Web サービスごとに、新しいクラス ローダを使用するようにした。 以前は、`build.xml` ファイルに複数の Web サービスが示されている場合でも、`wsgen` は実行全体で 1 つのクラスローダを使用していた。また、以降の各 Web サービスのすべての補助クラスは、1 つのクラスローダから取得した後にコピーされており、それが必要なすべてのクラス ファイルを生成する次善の方法だった。`build.xml` ファイルに示された、アセンブルの必要がある Web サービスの数によっては、`wsgen` を完全に実行するのに 30 分以上かかることがあった。 この問題は、コードを修正することで解決した。
CR101383	WebLogic Web サービスは、SOAP 応答で null の `xsd:dateTime` 値を `xsi:nil='true'` と正しく表現するようになった。 以前は、WebLogic Web サービスが null の `xsd:dateTime` 値を返す場合、SOAP 応答では `xsi:null='1'` と不適切に表現されていた。これは 1999 年に策定された XML スキーマでの null 値の定義である。WebLogic Web サービスは 2001 年に勧告された XML スキーマをサポートしており、null 値は `xsi:nil='true'` と定義されている。 この問題は、コードを修正することで解決した。
CR102677	外部エンティティの解決に XML レジストリを使用する場合、WebLogic Server はパブリック ID を使用して DTD を解決するときにエラーを返さなくなった。 この問題は、コードを修正することで解決した。

変更要求番号	説明
CR106892	生成された WebLogic Web サービス クライアント スタブは、XML ドキュメントを `org.w3c.dom.Element` オブジェクトとして Web サービスに送信するときに、XML ファイルの要素の本文にある改行文字を正しく保持するようになった。以前、クライアント スタブはこれらの改行文字を誤ってスペースに変換していた。 この問題は、コードを修正することで解決した。
CR111151	WebLogic Web サービスは、Web サービスの呼び出しで例外の結果として生成される SOAP Fault 応答で、utf-8 エンコーディング ヘッダを正しく送信するようになった。完全なヘッダは正しくは次のようになる。 `<?xml version="1.0" encoding="utf-8"?>` 以前、WebLogic Web サービスはこのヘッダを SOAP Fault 応答に含めていなかった。 この問題は、コードを修正することで解決した。

WTC-ATMI

変更要求番号	説明

CR107323	Tuxedo 8.1 を使用するようにコンフィグレーションされた WebLogic Server 6.1 SP03 では、コミットされていない Tuxedo トランザクションがあるときに WebLogic Server が異常終了 (〔CTRL〕+〔C〕) すると、再起動時に WebLogic Server インスタンスがハングした。問題は次のシナリオで発生した。
	1. Tuxedo でまだコミットされていないトランザクションがあるときに WebLogic が強制終了した。
	2. WTC は「ON_DEMAND」で起動するようにコンフィグレーションされている。
	3. WebLogic を起動する。
	4. WebLogic は報告された問題によりハングする。
	問題は次の場合には発生しなかった。
	1. Tuxedo でまだコミットされていないトランザクションがあるときに WebLogic が強制終了した。
	2. WTC は「ON_DEMAND」で起動するようにコンフィグレーションされている。
	3. WebLogic の tlog ファイル「*.tlog」を削除する。
	4. WebLogic を起動する。
	5. WebLogic は問題なく起動する。
	または
	1. Tuxedo でまだコミットされていないトランザクションがあるときに WebLogic が強制終了した。
	2. WTC は「ON_STARTUP」で起動するようにコンフィグレーションされている。
	3. WebLogic を起動する。
	4. WebLogic は問題なく起動する。
	再起動後にハングが起きる場合、デフォルト グループ内のすべてのスレッドは以下のスタック トレースによりハングする。
	`at java.lang.Object.wait(Native Method)`
	`at java.lang.Object.wait(Object.java:415)`
	`at weblogic.wtc.jatmi.TuxXidRply.get_specific_reply (TuxXidRply.java:203)at weblogic.wtc.gwt.TuxedoXA.internalCommit(TuxedoXA.java:259)`
	`at weblogic.wtc.gwt.TuxedoXA.commit(TuxedoXA.java:328)`
	`at weblogic.wtc.gwt.OatmialCommitter.execute(WTCStartup.java:2767)at...`
	この問題は、`weblogic.wtc.gwt.WTCStartup` メソッドが各 `Kernel.execute(myCommitter)` の後に 1 秒間スリープするように修正することで解決された。

変更要求番号	説明
CR120681	WebLogic Server 6.1 SP02 で、WebLogic Tuxedo Connector はローカル リソース名をリモート サービス名に正しく変換していなかった。 トランザクションの修正で問題は解決された。

XML

変更要求番号	説明

CR102397	Xerces パーサが常に System.getProperty() を呼び出そうとしていた。このため、プロパティを取得するパーミッションのないアプレット内で例外が送出されていた。JMS を呼び出したアプレット内で、次のような例外が送出された。
	`----------- Linked Exception -----------`
	`java.rmi.MarshalException: failed to marshal connectionCreate(Lweblogic.jms.dispatcher.DispatcherWrapper;); nested exception is:`
	`java.rmi.UnexpectedException: Failed to parse descriptor file; nested exception is:`
	`java.security.AccessControlException: access denied (java.util.PropertyPermission weblogic.apache.xerces.maxentityrefs read)`
	`[...]`
	コードを修正したことで、パーサがアプレット クライアントを使用してシステム プロパティを読み込むことはなくなった。

WebLogic Server 6.1 サービス パック 5 のソリューション

以下の節では、WebLogic Server 6.1 サービス パック 5 で解決された問題について説明します。

- クラスタ
- コネクタ
- コンソール
- コア
- デプロイメント
- EJB
- インストーラ
- JDBC
- jDriver
- JMS
- JNDI
- JSP
- JTA
- その他
- プラグイン
- RMI
- RMI-IIOP
- セキュリティ
- サーブレット
- WLS ツアー / サンプル
- Web サービス
- WebLogic Tuxedo

- XML

クラスタ

変更要求番号	説明
CR082443	同じマルチキャスト アドレスを使用する複数のクラスタが存在する環境では、大きなログ ファイルが作成され、ファイルのロールオーバが過度に頻繁に発生していた。 デバッグ メッセージは次のようなものであった。 `####<Jul 23, 2002 4:06:26 PM EDT> <Debug> <Cluster> <ustrntda01> <wts-cluster_test> <ExecuteThread: '9' for queue: 'default'> <> <> <000000> <dropped fragment from foreign domain/cluster domainhash=95475927 clusterhash=-1674453961>` この問題は、`weblogic.cluster.ClusterDebug.java` において、デバッグ メッセージをログに記録するか否かを制御するフラグが true に設定されていたために発生していた。コードを修正し、この問題を解決した。
CR087240	セッションをホストする管理対象サーバをシャットダウンすると、HTTP セッションのフェイルオーバで `ClassCastException` が発生していた。エラー メッセージを無視すると、フェイルオーバは成功した。 この問題は、次のコンフィグレーションにおいて確実に再現した。 1) 6.1 SP02 を実行するクラスタ化された 2 つのサーバ インスタンス 2) WLP 4.0 SP002 ポータル アプリケーションをホストするクラスタ 3) 2 ノードのクラスタを指す WebLogic Server プラグインを含む IPlanet Web サーバ 他のコンフィグレーションでは、この問題が再現しない場合があった。 この問題は、WebLogic Server のシャットダウン シーケンスの問題と関係がある。 `RMIServerService` をシャットダウンするための新しいサービスにより、この問題を解決した。

CR092146	クラスタのステートフル セッション Bean に対してフェイルオーバが機能しなかった。リクエスト URL に、クラスタの DNS 名ではなくローカル サーバの名前が含まれていた。 次の例外が発生した。 `java.rmi.NoSuchObjectException: Unable to locate EJBHome: 'statefullCluster' on server: 't3://172.23.135.83:7600 at weblogic.ejb20.internal.HomeHandleImpl.getEJBHome(HomeHandleImpl.java:80) at weblogic.ejb20.internal.HandleImpl.getEJBObject(HandleImpl.java:204) at com.bea.cluster.testClusterEJBServlet.doGet(testClusterEJBServlet.java:137) at javax.servlet.http.HttpServlet.service(HttpServlet.java:740) at javax.servlet.http.HttpServlet.service(HttpServlet.java:853) at weblogic.servlet.internal.ServletStubImpl.invokeServlet(ServletStubImpl.java:262) at weblogic.servlet.internal.ServletStubImpl.invokeServlet(ServletStubImpl.java:198) at weblogic.servlet.internal.WebAppServletContext.invokeServlet(WebAppServletContext.java:2637) at weblogic.servlet.internal.ServletRequestImpl.execute(ServletRequestImpl.java:2359) at weblogic.kernel.ExecuteThread.execute(ExecuteThread.java:139) at weblogic.kernel.ExecuteThread.run(ExecuteThread.java:120)` `weblogic.ejb20.internal.BaseEJBHome.java` の `getURL()` メソッドを修正し、URL のホスト部分としてクラスタ アドレスを渡すことで、この問題を解決した。

CR093809	セッションのルックアップに関するエラーが原因で、スタック トレースが発生した。セカンダリ サーバ インスタンスが、シャットダウンしていた。処理しているリクエストに対するログ エントリを作成する際に、HTTP サーバはセッションからユーザの情報を取得しようとする。そのため、シャットダウン中であってもセカンダリ サーバに対するルックアップを試み、結果として次のスタック トレースが発生した。 `<Dec 26, 2002 7:17:55 PM EST> <Error> <HTTP Session> <Error looking up session for id:2LcR6Ool2IC5qNtFZhThJClYVYEXt9KWG2ciXmAXZ4XiBAHPiHx4!-1379505194!stcasfi01b.usa-ed.net!8001!7002!-1127797657!stcasfi02b.usa-ed.net!8001!7002 java.rmi.ConnectIOException: Server is being shut down Start server side stack trace: java.rmi.ConnectIOException: Server is being shut down at weblogic.rjvm.RJVMImpl.dispatchRequest(RJVMImpl.java:692) at weblogic.rjvm.RJVMImpl.dispatch(RJVMImpl.java:666) at...` プライマリ サーバは利用可能なサーバのリストから新しいセカンダリ サーバを正しく選択するので、セッション データが失われることはなかった。 コードを修正し、この問題を解決した。サーバは、セッションではなく ThreadLocal からユーザ情報を取得するようになった。
CR94561	WebLogic Server 6.1 SP02 では、アプリケーションがインメモリ セッション レプリケーションまたはレプリケートされたステートフル セッション Bean を使うと、2 人のユーザの間でセッション データを共有できてしまうという、潜在的なセキュリティの弱点が発見された。この障害の原因は、競合状況の結果として 2 つのセッションに同じ ROID (レプリカ可能オブジェクト ID と呼ばれる内部 ID) が割り当てられるためであった。たとえば、セッション A に ROID が割り当てられた後、セッション B に同じ ROID が割り当てられる。すると、それ以降のセッション B のリクエストはすべて、A の HTTP セッション /SFSB にマップされる。HTTP セッションが混じり合うと、セッション B のユーザがセッション A のコンテンツを見ることができた。この問題が発生する可能性は非常に低く、意図的に利用することはできなかった。 `ROID.create()` の同期を取るようにコードを修正し、重複する ROID が複数のセッションに割り当てられないようにすることで、この問題を解決した。 http://dev2dev.bea.com/resourcelibrary/advisoriesnotifications/BEA03-26.01.jsp の「SECURITY ADVISORY (BEA03-26.01)」を参照すること。

CR101609	WebLogic Server 6.1 SP04 では、シリアライズされていないオブジェクトのレプリケートに失敗すると、その後、セッション レプリケーションが機能しなくなった。シリアライズされていないデータをセッション オブジェクトに設定しようとする JSP を呼び出すと、予想される例外が発生した。
	`<Mar 19, 2003 2:58:23 PM PST> <Error> <Cluster> <All session objects should be serializable to replicate. Please check the objects in your session. Failed to replicate non serializable object>`
	その後、それ以降のすべてのリクエストに対してセッションがレプリケートされず、次の例外が発生した。
	`<Mar 19, 2003 2:58:48 PM PST> <Debug> <Cluster> <Unable to create secondary for -8956818414963087828>`
	コードを修正し、この問題を解決した。シリアライズ不可能なオブジェクトに遭遇すると、メソッドは他のセカンダリを試みずに返る。

CR102655	CR101609 と同じ。前記を参照。

コネクタ

変更要求番号	説明
CR086251	WebLogic Server 6.1 SP03 では、シャットダウンの間に unregisterResource においてメッセージング ブリッジ アダプタが `NullPointerException` を送出した。この問題は、メッセージが送信されたが、消費される前にサーバのシャットダウンが開始された場合に発生した。メッセージング ブリッジがデプロイされていず、アダプタがデプロイされている場合には、問題は発生しなかった。 コンフィグレーションは、`jms-xa-adp.rar` メッセージング ブリッジ アダプタと、MQSeries 5.2 と JMS の間のメッセージング ブリッジであった。 登録解除を試みる前に wrappedXARes をチェックするようにコードを修正することで、この問題を解決した。
CR090002	WebLogic Server 6.1 SP03 では、`JCA-Connection.close` が `ResourceException` ではなく `java.lang.reflect.UndeclaredThrowableException` を送出した。 リソース アダプタの本来の例外を渡すように、コードを修正した。

CR090792	WebLogic Server 6.1 SP03 と SP04 では、Console で RAR の記述子パラメータのいずれかを変更すると、RAR のデプロイメントが失敗した。
	分析の結果、プール パラメータがデフォルト値から変更されていない場合、Console で記述子を変更すると、プール パラメータがリセットされることがわかった。プール パラメータの定義が不適切であった。
	`<pool-params>`
	`<initial-capacity>0</initial-capacity>`
	`<max-capacity>0</max-capacity>`
	`<capacity-increment>0</capacity-increment>`
	`<shrinking-enabled>false</shrinking-enabled>`
	`<shrink-period-minutes>0</shrink-period-minutes>`
	`</pool-params>`
	この結果、最大容量が 0 という許されない値になるためデプロイメントが失敗した。その後は、RA に対していかなる処理も実行できなかった。
	6.1 では、記述子の値を取得するときは `ResourceAdapterComponentMBean` が使用される。しかし、`ResourceAdapterComponentMBean` のコンストラクタはデフォルトを自動的には設定せず (`ConnectorComponentMBean` が行っている方法)、デフォルトを手動でも設定していなかった。手動で設定するようにして、問題を解決した。

コンソール

変更要求番号	説明
CR062102	6.1 SP01 から大きなアプリケーション (40 MB) を移行すると、起動時にアプリケーションを管理対象サーバにコピーするため、ネットワークが過負荷になり、WebLogic Server の起動が遅くなった。 コードを修正し、この問題を解決した。変更されていないアプリケーションは、起動時に管理対象サーバにコピーされなくなった。 変更の有無にかかわらず起動時にアプリケーションを管理対象サーバにコピーするには、`forceApplicationCopy` パラメータを使用する。1-3 ページの「アプリケーションを強制的に更新するための新しいパラメータ」を参照すること。
CR069284	ドメインの [コンフィグレーション | アプリケーション] ページの [自動デプロイを有効化] 属性がオフになっていても、アプリケーション ファイルがアプリケーション ディレクトリにコピーされると、アプリケーションのコンポーネントがまだ自動的にデプロイされた。 調査の結果、WebLogic Server が [自動デプロイを有効化] 属性を使用していないことがわかった。代わりに、自動デプロイメントは `DomainMBean.setProductionModeEnabled` プロパティによって制御されている。このプロパティは、起動時にコマンドラインで設定する。 Administration Console から `AutoDeployedEnabled` 属性を削除した。
CR084607	WebLogic Server 6.1 SP03 では、実行時にクラスタで JMS トピックを作成すると、`InstanceNotFoundException` が発生した。トピックは、クラスタ内の管理対象サーバを対象として、起動クラスの `JMSHelper.createPermanentTopicAsync` で作成されていた。トピックが作成された後、Administration Console で JMS サーバに対する送り先リンクをクリックすると、次のようなメッセージが表示された。 `java.lang.reflect.UndeclaredThrowableException: javax.management.InstanceNotFoundException: mydomain:Name=testTopic12,Type=JMSTopic at com.sun.management.jmx.MBeanServerImpl.getMBean(MBeanServerImpl.java:1680) at com.sun.management.jmx.MBeanServerImpl.getAttribute(MBeanServerImpl.java:1152) at weblogic.management.internal.MBeanProxy.getAttribute(MBeanProxy.java:254) at weblogic.management.internal.MBeanProxy.invoke(MBeanProxy.java:187)`

CR088842	WebLogic Server 6.1 SP02 では、1 つの管理サーバと 3 つの管理対象サーバというコンフィグレーションにおいて、管理対象サーバの 1 つを再起動するとデッドロックが発生した。 分析の結果、管理対象サーバの `lookupServerRuntime()` が、管理サーバに対する不要な `getMBean()` の呼び出しを行っていた。 コードを修正し、この問題を解決した。
CR089747	Solaris Sparc 2.8 上の WebLogic Server 6.1 SP03 において、ドメイン ログ ファイルが時間でローテーションされなかった。Windows 2000 では、ログ ファイルが 3 つ作成された後、ローテーションが停止していた。古いログ ファイルを削除すると、ローテーションが再開した。Windows 2000 の場合、SP01 から SP03 へのアップグレードの後ではログ ローテーションが機能したが、SP03 を新規インストールすると機能しなかった。ドメイン ログのコンフィグレーションは、次のとおりであった。 `<Log FileName="config/mydomain/logs/wl-domain.log" FileTimeSpan="1" Name="mydomain" RotationType="byTime" RotationTime=10:00/>` コードを修正し、この問題を解決した。
CR091359	WebLogic Server 6.1 SP03 および SP04 では、`.ear` ファイル内の Web アプリケーションに対して、Administration Console のオプション [Web アプリケーション｜*webapp1*｜モニタ｜すべてのアクティブな Web アプリケーションのモニタ ...] が機能しなかった。 コードを修正し、この問題を解決した。
CR093409	WebLogic Server 6.1 SP02、Solaris 2.8、JDK1.3.1_04 という環境では、1 つの管理サーバと 5 つの管理対象サーバというコンフィグレーションにした場合、異なるマシンから `weblogic.deploy` ユーティリティを使って、3 つの EJB コンポーネントを含む EAR をデプロイしようとすると、管理サーバがハングした。スレッド ダンプはデッドロックを示していた。スタック トレースは次のような内容であった。 `Execute thread 4(admin_rmi queue) holds lock of the com.sun.management.jmx.MBeanServerImpl and waiting to acquire lock on weblogic.logging.LogManager. Execute thread 8(default queue) holds lock of the weblogic.logging.LogManager and waiting to acquire lock on com.sun.management.jmx.MBeanServerImpl.` 分析の結果、`LogManager` と `FileStreamLogger` でデッドロックが発生していた。 コードを修正し、この問題を解決した。

CR096726	WebLogic Server 6.1 SP02 では、`MBeanHome.deleteMBean()` を呼び出すと Null ポインタ例外が発生した。 コンフィグレーションは、4 つの管理対象サーバがそれぞれ異なる物理マシン上で稼働し、それぞれに対して JMX クライアントが動作する (1 対 1 で) というものであった。JMX クライアントが `MBeanHome.deleteMBean` を行うと、`NullPointerException` が発生した。 分析の結果、複数の管理対象サーバの非同期削除に関する問題であった。ある管理対象サーバが、MBean 自体を削除する前に他の MBean における参照を削除し、別の管理対象サーバが同時に同じことを行うと、Null ポインタ例外 (NPE) が発生した。 MBean に対するメタデータを取得するコード (つまり `mbean.getMBeanInfo()`) に対して `try/except` ブロックを実装することで、この問題を解決した。MBean を削除するときは、最初に MBeanServer にある現在の MBean のリストを取得した後、削除する Bean に対する参照を削除する。問題は、他の削除スレッドが同じことを行おうとして (静的な) リストから Bean を削除することがあり、そのような場合は、更新するために Bean のメタデータを取得しようとすると、NPE が発生する。このパッチでは、NPE を無視し、リストにある次の Bean を移るようにした。
CR096950	`FileDistributionServlet` の `doGet()` メソッドを使うと、管理サーバ上のファイルをダウンロードできた。コードを修正し、この問題を解決した。 http://dev2dev.bea.com/resourcelibrary/advisoriesnotifications/BEA03-28.jsp の「SECURITY ADVISORY (BEA03-28.00)」を参照すること。
CR098623	`<beahome>\wlportal4.0\applications\portal` などのポータル アプリケーションを EAR ファイルとしてデプロイすると、Administration Console で `application-config.xml` のパラメータを変更できなかった。コードを修正し、この問題を解決した。

CR098739

WebLogic Server 6.1 SP02 では、管理サーバにおいて Java のデッドロックが検出された。スレッド ダンプには、次のような Java レベルのデッドロックが示されていた。つまり、スレッド 8 が RemoteMBeanServerImpl をロックし、さらに LogManager をロックしようとしたが、LogManager はスレッド 6 がロックしており、スレッド 6 は RemoteMBeanServerImpl を待っていた。

「ExecuteThread: '8' for queue: '__weblogic_admin_rmi_queue'」に対する Java スタック：

```
at weblogic.logging.LogManager.addSubsystem(LogManager.java:
58) - waiting to lock <f55b6358> (a weblogic.logging.Log
Manager) at weblogic.logging.LogOutputStream.getLogManager
(LogOutputStream.java:32) at weblogic.logging.LogOutput
Stream.error(LogOutputStream.java:63) at weblogic.management.
internal.Helper.error(Helper.java:1391) at weblogic.
management.internal.Helper.error(Helper.java:1404) at
weblogic.management.internal.ConfigurationMBeanImpl.
postDeregister(ConfigurationMBeanImpl.java:900) at com.sun.
management.jmx.MBeanServerImpl.postDeregisterInvoker
(MBeanServerImpl.java:2314) at com.sun.management.jmx.MBean
ServerImpl.unregisterMBean(MBeanServerImpl.java:967) - locked
<f550eec8> (a weblogic.management.internal.RemoteMBean
ServerImpl) at weblogic.management.internal.RemoteMBeanServer
Impl.unregisterMBean(RemoteMBeanServerImpl.java:213) at ..
```

「ExecuteThread: '6' for queue: 'default'」に対する Java スタック：

```
at com.sun.management.jmx.MBeanServerImpl.getMBean(MBean
ServerImpl.java:1672) at com.sun.management.jmx.MBeanServer
Impl.getAttribute(MBeanServerImpl.java:1150) at weblogic.
management.internal.MBeanProxy.getAttribute(MBeanProxy.java:25
4) at weblogic.management.internal.MBeanProxy.invoke
(MBeanProxy.java:187) at $Proxy7.isStdoutEnabled(Unknown
Source) at weblogic.logging.ConsoleLogger.isLog(Console
Logger.java:81) at weblogic.logging.ConsoleLogger.log
(ConsoleLogger.java:70) at weblogic.logging.LogManager.log(Log
Manager.java:260) - locked <f55b6358> (a
weblogic.logging.LogManager) at
weblogic.logging.MessageLogger.log(MessageLogger.java:17) at
weblogic.t3.srvr.T3SrvrLogger.logRemovingClientContextSoftDisc
onnect(T3SrvrLogger.java:1204) at
weblogic.t3.srvr.ClientContext.dieIfTimedOut(ClientContext.jav
a:512) at ...
```

LogManager における不必要なロックを削除するようコードを修正することで、この問題を解決した。

コア

変更要求番号	説明
CR036602	`ClusterMBean.setClusterAddress()` メソッドが不正な値を受け入れて、`IllegalArgumentException` を送出していなかった。 サーバ起動時にクラスタ アドレスが正しいことをチェックするようにして、この問題を解決した。
CR036603	`ClusterMBean.setMultiCustAddress()` メソッドが不正な値を受け入れて、`IllegalArgumentException` を送出していなかった。 コードを修正し、この問題を解決した。
CR077170	WebLogic Server 6.1 SP03 では、管理サーバをシャットダウンした後で管理対象サーバを停止すると、例外が発生した。この問題は、以下のアクションのシーケンスで発生した。 1. 管理サーバと管理対象サーバを起動する。 2. 管理サーバをシャットダウンする。 3. 管理対象サーバの `ServerConfigMBean` で「停止」メソッドを呼び出す。 例外は、次のようなものであった。 `Unexpected Exception Start server side stack trace: java.rmi.ConnectException: Unable to get direct or routed connection to: '-19546889165621794S:dwarkamai:[7001,7001,-1,-1,7001,-1,-1]:OAMdomain:adminServer' at weblogic.rmi.internal.BasicOutboundRequest.sendReceive(BasicOutboundRequest.java:109) at weblogic.rmi.internal.BasicRemoteRef.invoke(BasicRemoteRef.java:127) at...` `weblogic.Admin` からのシャットダウン コマンドは成功していた。 管理サーバが停止しているときに管理対象サーバをシャットダウンすると発生する例外を捕捉することで、この問題を解決した。

CR079354	WebLogic Server 6.1 SP02 では、Apache プラグインを通して DataSource にアクセスする Java クライアントが、`TunnelDeadResponse` を受け取っていた。 WebLogic Server は、Windows 2000 の下で稼働し、パッチ CR061847 を適用していた。プラグインは、Solaris の下で動作していた。 例外は、次のようなものであった。 `<May 31, 2002 4:16:35 PM PDT> <Error> <HTTPClientJVMConnection> <java.net.ProtocolException: Tunneling result not OK, result: 'DEAD', id: '2' at weblogic.rjvm.http.HTTPClientJVMConnection.receiveAndDispatch (HTTPClientJVMConnection.java:413) at weblogic.rjvm.http.HTTPClientJVMConnection.execute (HTTPClientJVMConnection.java:296) at weblogic.kernel.ExecuteThread.execute(ExecuteThread.java:139) at weblogic.kernel.ExecuteThread.run(ExecuteThread.java:120)>` プラグインを削除すると、例外は発生しなかった。 分析の結果、ホスト / ポートの組み合わせに対する RJVM が存在しているにもかかわらず、WebLogic Server がそれを発見できなかった。rjvm マネージャのシノニム キャッシュが、ポートの情報を保持していなかった。WebLogic Server は、ブートストラップのときキャッシュ内に一致するものが見つからないと、新しい RJVM を作成した。 ブートストラップの後、WebLogic Server は新しい RJVM が重複していることを識別し、それを閉じようとした。メッセージがキューに格納されている場合、サーバにメッセージが送付されないことがあった。その結果、サーバは RJVM でタイムアウトし、DEAD 応答を送信していた。 シノニム キャッシュがポート情報を保持するようコードを修正することで、この問題を解決した。

CR089470	WebLogic Server 6.1 SP03 では、クライアント ソケットが接続を開いて閉じると、ノード マネージャが NumberFormatException を送出した。
	`<Oct 28, 2002 3:38:18 PM CST> <Info> <NodeManager@localhost:5555>`
	`<SecureSocketListener: listening on localhost:5555>`
	`java.lang.NumberFormatException: null`
	`at java.lang.Integer.parseInt(Integer.java:382)`
	`at java.lang.Integer.parseInt(Integer.java:463)`
	`at weblogic.nodemanager.SocketInputHandler.run(SocketInputHandler.java:109)`
	認識できる副次的影響はなかった。エラーが発生した後でも、ノード マネージャは管理対象サーバを起動および停止できた。
	データ転送を行わないで接続を開いて閉じるソケットにおいて例外を処理するよう、コードの修正を実装した。

CR070887	WebLogic Server 6.1 に SP02 を適用してもしなくても、`setInstanceFollowRedirects()` が機能しなかった。 JDK1.3.1 では、`java.net.HttpURLConnection` にメソッド `getInstanceFollowRedirects()` と `setInstanceFollowRedirects()` が追加された。 これらのメソッドを使うと、特定の `HttpURLConnection` に対する URL のリダイレクトを無効にすることができる。`HttpURLConnection` が、`setInstanceFollowRedirects()` で設定されたフラグを無視していた。 `InstanceFollowRedirects` の設定に従ってリダイレクトを処理するように `HttpURLConnection` のリダイレクト ロジックのコードを修正することで、この問題を解決した。

CR077108	35 以上のノードで構成されるクラスタの負荷テストの際に、`weblogic.utils.collections.NumericValueHashtable.containsKey(NumericValueHashtable.java:138)` で null ポインタ例外が発生した。 `<May 9, 2002 5:08:22 PM BST> <Error> <Management> <InvocationTargetException getting attribute SecondaryDistributionNames on MBean bluemartini:Location =WebConnectaw01,Name=WebConnectaw01,ServerRuntime=WebConnectaw 01,Type=ClusterRuntime. Method: public java.lang.String[] weblogic.cluster.ClusterRuntime.getSecondaryDistributionNames( ) java.lang.NullPointerException at weblogic.utils. collections.NumericValueHashtable.containsKey(NumericValueHash table.java:138) at weblogic.cluster.replication. ReplicationManager.getSecondaryDistributionNames(ReplicationMa nager.java:1245) at weblogic.cluster.ClusterRuntime.` `getSecondaryDistributionNames(ClusterRuntime.java:100) at java.lang.reflect.Method.invoke(Native Method) at weblogic.management.internal.DynamicMBeanImpl.getAttribute(Dyn amicMBeanImpl.java:511) at weblogic.management.internal. DynamicMBeanImpl.getAttribute(DynamicMBeanImpl.java:477) at com.sun.management.jmx.MBeanServerImpl.getAttribute(MBeanServe rImpl.java:1181) at com.sun.management.jmx.MBeanServer Impl.getAttribute(MBeanServerImpl.java:1151) at weblogic.management.internal.RemoteMBeanServerImpl_WLSkel.invo ke(Unknown Source) at weblogic.rmi.internal.BasicServer Ref.invoke(BasicServerRef.java:298) at weblogic.rmi. internal.BasicServerRef.handleRequest(BasicServerRef.java:267) at weblogic.rmi.internal.BasicExecuteRequest.execute (BasicExecuteRequest.java:22) at weblogic.kernel.Execute Thread.execute(ExecuteThread.java:139) at weblogic.kernel. ExecuteThread.run(ExecuteThread.java:120)` 分析の結果、`getSecondaryDistributedNames()` メソッドは `getOtherHost()` から返る null 値に備えなければならないことがわかった。この問題を解決した。

CR079354

SP02 とパッチ CR061847 を適用した WebLogic Server 6.1 では、Java Swing クライアントがコンテキストを取得して JDBC データ ソースにアクセスすると、TunnelDeadResponse を受け取った。クライアントは、初期コンテキストを取得し、データソースを通して接続を取得し、初期コンテキストを閉じて、接続を閉じる。その後は、この処理シーケンスを繰り返す。このエラーは、以下のような結果になる。

サーバでは、次のような例外が送出された。

```
####<Nov 21, 2002 12:24:20 PM CST> <Debug> <HTTPServerJVMConnection>
<hpwlinc03> <admin> <ExecuteThread: '14' for queue: 'default'> <> <>
<000000> <Closing JVM socket:
'weblogic.rjvm.http.HTTPServerJVMConnection@4cfbd8 - id: '0',
closed: 'true', lastRecv: '1037903025816''> java.lang.
Throwable: Stack trace at weblogic.rjvm.http.HTTPServer
JVMConnection.close(HTTPServerJVMConnection.java:383) at
weblogic.rjvm.ConnectionManager.removeConnection(ConnectionManager.
java:879) at weblogic.rjvm.ConnectionManager.
shutdown(ConnectionManager.java:508) at weblogic.rjvm.
ConnectionManagerServer.shutdown(ConnectionManagerServer.java:443)
at weblogic.rjvm.RJVMImpl.peerGone(RJVMImpl.java:804) at
weblogic.rjvm.RJVMImpl.gotExceptionReceiving
(RJVMImpl.java:533) at weblogic.rjvm.ConnectionManager.got
ExceptionReceiving(ConnectionManager.java:736) at
weblogic.rjvm.http.HTTPServerJVMConnection.checkIsDead(HTTPServerJV
MConnection.java:238) at weblogic.rjvm.http.HTTPServer
JVMConnection$TunnelScavenger.trigger(HTTPServerJVMConnection.java:
405) at weblogic.time.common.internal.
ScheduledTrigger.executeLocally(ScheduledTrigger.java:238) at
weblogic.time.common.internal.ScheduledTrigger.
execute(ScheduledTrigger.java:229) at weblogic.kernel.Execute
Thread.execute(ExecuteThread.java:139) at weblogic.kernel.
ExecuteThread.run(ExecuteThread.java:120)
```

クライアントでは、次のような例外が送出された。

```
<Nov 21, 2002 12:24:18 PM CST> <Error> <HTTPClientJVM
Connection> < java.net.Protocol Exception: Tunneling
result not OK, result: 'DEAD', id: '0' at weblogic.rjvm.
http.HTTPClientJVMConnection.receiveAndDispatch(HTTPCli
entJVMConnection.java:413) at weblogic.rjvm.http.HTTPClient
JVMConnection.execute(HTTPClientJVMConne ction.java:296) at
weblogic.kernel.ExecuteThread.execute(ExecuteThread.java:139) at
weblogic.kernel.ExecuteThread.run(ExecuteThread.
java:120)
```

分析の結果、ホスト / ポートの組み合わせに対する RJVM が存在するにもかかわらず、WLS が RJVM の検索に失敗してタイムアウトが発生していることがわかった。RJVM の同義語キャッシュが、ポートの情報を保持していなかった。RJVM の同義語キャッシュにポート情報を保持するようにコードを修正することで、問題を解決した。

CR080822	パフォーマンス パックを適用した Solaris 配布キットでは、ログ メッセージで表示されるファイル記述子ソフト リミットの値が正しくなかった。ファイル記述子ハード リミットの値が、ソフト リミットとハード リミットの両方に示されていた。
	次に示すのは、weblogic.log に記録されるエラーである。
	`####<Mar 5, 2002 5:07:27 PM PST> <Info> <Posix Performance Pack> <bolinas> <myserver> <ListenThread> <system> <> <000000> <System has file descriptor limits of- soft: '1024', hard: '1024'>`
	`####<Mar 5, 2002 5:07:27 PM PST> <Info> <Posix Performance Pack> <bolinas> <myserver> <ListenThread> <system> <> <000000> <Using effective file descriptor limit of: '1024' open sockets/files.>`
	このエラーは、Solaris 2.6 および Solaris 2.8 で発生した。
	関連するスクリプト テンプレートの構文を修正することで、この問題を解決した。
	正しくない構文：
	`[ !$? -a "$maxfiles" != 1024 ];`
	修正後の正しい構文：
	`[ ! $? -a "$maxfiles" != 1024 ];`

CR085259	同じドメインでホストされる 2 つの独立した WebLogic クラスタがセッション クッキーに対して同じ名前を使用すると、クラスタ間でセッション データが複製されていた。 セカンダリ ホスト / ポートがセッションのプライマリ ホストと同じクラスタ内にあるかどうかの検査を実装することで、この問題を解決した。
CR086425	CR085259 と同じ。前記参照。

CR086758	多層型の実装では、負荷テストを 10 時間続けると、Web 層がハングした。Web 層 (サーブレット、JSP、およびカスタム キャッシュを実装するカスタム クラスで構成される) が EJB 層 (すべてステートレス セッション Bean) と通信している最中に、RJVM ハートビートが消えたために EJB 層が Connection Manager をクローズすると、この問題が発生した。 Web 層がハングする前に、EJB 層で次の例外が送出された。 <Sep 24, 2002 8:43:58 AM PDT> <Info> <RJVM> <Failure in heartbeat trigger for RJVM: '7831636024374910916S:10.10.10.187:[8001,8001,8002,8002,8001,8002,-1]:webserver:sourcingWebserver' java.rmi.ConnectException: The connection manager to ConnectionManager for: 'weblogic.rjvm.RJVMImpl@3bedf2 - id: '7831636024374910916S:10.10.10.187:[8001,8001,8002,8002,8001,8002,-1]:webserver:sourcingWebserver' connect time: 'Tue Sep 24 06:15:16 PDT 2002'' has already been shut down at weblogic.rjvm.ConnectionManager.getOutputStream(ConnectionManager.java:1348) at weblogic.rjvm.ConnectionManager.createHeartbeatMsg(ConnectionManager.java:1306) at weblogic.rjvm.ConnectionManager.sendHeartbeatMsg(ConnectionManager.java:497) at weblogic.rjvm.RJVMImpl$HeartbeatChecker.trigger(RJVMImpl.java:1032) at weblogic.time.common.internal.ScheduledTrigger.executeLocally(ScheduledTrigger.java:238) at weblogic.time.common.internal.ScheduledTrigger.execute(ScheduledTrigger.java:229) at weblogic.kernel.ExecuteThread.execute(ExecuteThread.java:139) at weblogic.kernel.ExecuteThread.run(ExecuteThread.java:120) 保留中の応答にピア消失イベントを通知するロジックを追加することで、この問題を解決した。

CR087808

Sunblade 100 シングル CUP Solaris マシンで稼働する WebLogic Server 6.1 SP03 では、2 つの独立したサーバインスタンスが、相互に InitialContext をルックアップできなかった。一方のサーバインスタンスが InitialContext をルックアップした後、同じマシン上の 2 番目のサーバインスタンスが同じマシン上の InitialContext をルックアップしようとすると、次の例外が発生した。

```
<2002/10/09 4:23:56:JST> <Debug> <ConnectionManager> <Attempt
to sendMsg using a closed connection>
javax.naming.CommunicationException. Root exception is
java.rmi.ConnectException: Attempt to sendMsg using a closed
connection at
weblogic.rmi.internal.BasicOutboundRequest.sendReceive(BasicOu
tboundRequest.java:85) at
weblogic.rmi.cluster.ReplicaAwareRemoteRef.invoke(ReplicaAware
RemoteRef.java:262) at
weblogic.rmi.cluster.ReplicaAwareRemoteRef.invoke(ReplicaAware
RemoteRef.java:229) at
weblogic.rmi.internal.ProxyStub.invoke(ProxyStub.java:35) at
$Proxy45.lookup(Unknown Source) at
weblogic.jndi.internal.WLContextImpl.lookup(WLContextImpl.java
:341) at
javax.naming.InitialContext.lookup(InitialContext.java:345) at
com.bea.samples.servlet.TestServlet.service(TestServlet.java:4
0) at
javax.servlet.http.HttpServlet.service(HttpServlet.java:853)
at
weblogic.servlet.internal.ServletStubImpl.invokeServlet(Servle
tStubImpl.java:265) at
weblogic.servlet.internal.ServletStubImpl.invokeServlet(Servle
tStubImpl.java:200) at
weblogic.servlet.internal.WebAppServletContext.invokeServlet(W
ebAppServletContext.java:2546) at
weblogic.servlet.internal.ServletRequestImpl.execute(ServletRe
questImpl.java:2260) at
weblogic.kernel.ExecuteThread.execute(ExecuteThread.java:139)
at weblogic.kernel.ExecuteThread.run(ExecuteThread.java:120)
```

問題は、Sun Enterprise 上では再現しなかった。localhost ではなく IP アドレスを使用してルックアップを行った場合は、Sun Blade 上でも発生しなかった。

分析の結果、クライアントの RJVM が重複する T3JVMConnections を閉じようとしていた。クライアントはサーバに CMD_REQUEST_CLOSE を発行し、RJVM を閉じた。しかし、サーバがこのメッセージをキューに入れると、RJVM は閉じたものとしてマークされていた。

重複する接続が検出され、CMD_REQUEST_CLOSE がサーバに到着していない場合は、RJVM をシャットダウンしないようにコードを修正し、この問題を解決した。

CR087944	Windows XP で稼働すると、「`java.lang.UnsatisfiedLinkError: no muxer in java.library.path`」のエラーが発生した。 原因は、WebLogic Server がホスト オペレーティング システムとして Windows XP を正しく通知しなかったためである。JDK 1.3.1_03 では、`os.name` は「Windows 2000」を返す。 `SocketMuxer` のメソッドを修正することで、この問題を解決した。
CR088022	ノード マネージャで管理対象サーバを起動した後、Administration Console の左ペインでサーバ名を右クリックして [接続を見る] を選択しても、管理サーバと管理対象サーバの間の通信に関する情報 (使用されているネットワーク チャンネルや JVMID など) が表示されなかった。 分析の結果、`ConnectionManagerServer` のバグが元で、`ServerRuntimeMBean` の `getConnections()` メソッドが接続オブジェクトのインスタンスを返していないことがわかった。 RJVM が null である接続についても接続マネージャが報告するようコードを修正することで、問題を解決した。
CR088056	WebLogic Server の muxer (マルチプレクサ) ライブラリにビルド日時の情報がないため、使用されているライブラリのバージョンを特定するのが困難であった。`NTSocketMuxer` にはビルド日時が含まれていたが、`PosixSocketMuxer` には含まれていなかった。 `PosixSocketMuxer` にビルド日時を追加することで、この問題を解決した。muxer の初期化時に、次のようなメッセージが表示されるようになった。 `<Oct 15, 2002 3:44:04 PM PDT> <Info> <socket> <000406> <PosixSocketMuxer was built on Oct 15 2002 15:05:11>`
CR089060	OS/390 Linux (SuSE Linux Kernel 2.2) では、`libmuxer.so` が次のエラーを送出した。 `java.lang.UnsatisfiedLinkError: /opt/weblogic6/lib/linux/s390/libmuxer.so: /opt/weblogic6/lib/linux/s390/libmuxer.so: ELF file machine architecture not s390 at java.lang.ClassLoader$NativeLibrary.load(Native Method) at java.lang.ClassLoader.loadLibrary0 (ClassLoader.java:1799)` Linux Kernel 2.2 用のバイナリを提供することで、この問題を解決した。

CR089144	jar にカスタム呼び出しルータが含まれている EJB のデプロイメントが、IllegalArgumentException で失敗した。ReplicaAwareInfo クラスの classForName() メソッドが現在のスレッドの ContextClassLoader ではなくシステム クラスローダでクラスを探していたため、クラスがロードされなかった。 スタック トレース： java.lang.reflect.InvocationTargetException: java.lang.IllegalArgumentException: Class weblogic.qa.tests.cluster.ejb.callrouter.BaseCallRouterImpl not found at weblogic.rmi.cluster.ReplicaAwareInfo.classForName(ReplicaAwareInfo.java:172) at weblogic.rmi.cluster.ReplicaAwareInfo.getCallRouter(ReplicaAwareInfo.java:132) at weblogic.rmi.cluster.BasicReplicaHandler.newReplicaList(BasicReplicaHandler.java:78) at weblogic.rmi.cluster.BasicReplicaHandler.<init>(BasicReplicaHandler.java:72) at java.lang.reflect.Constructor.newInstance(Native Method) at weblogic.rmi.cluster.ReplicaAwareInfo.instantiate(ReplicaAwareInfo.java:186) at weblogic.rmi.cluster.ReplicaAwareInfo.getReplicaHandler(ReplicaAwareInfo.java:117) at weblogic.rmi.cluster.ReplicaAwareRemoteRef.initialize(ReplicaAwareRemoteRef.java:79) at weblogic.rmi.cluster.ClusterableRemoteRef.initialize(ClusterableRemoteRef.java:28) at weblogic.rmi.cluster.ClusterableRemoteObject.initializeRef(ClusterableRemoteObject.java:275) at weblogic.rmi.cluster.ClusterableRemoteObject.onBind(ClusterableRemoteObject.java:150) at weblogic.jndi.internal.BasicNamingNode.bindHere(BasicNamingNode.java:346) at weblogic.jndi.internal.ServerNamingNode.bindHere(ServerNamingNode.java:105) at weblogic.jndi.internal.BasicNamingNode.bind(BasicNamingNode.java:281) at ... ClientDrivenBeanInfoImpl で deploy() を呼び出すときはコンテキスト クラスローダをアプリケーション クラスローダに設定し、呼び出しルータ クラスをロードすることをコンテキスト クラスローダでチェックすることで、この問題を解決した。

CR089454	実行キューの最大長をコンフィグレーションすることで受信するリクエストのトラフィックを抑制する新しい機能が追加された。実行キューの最大長は、`weblogic.kernel.allowQueueThrottling` の新しい `-D` フラグで設定できる。 この機能は、カスタム キューにおいて「低速で移動し、リソースを大量に使用する」リクエストを抑制できるように設けられた。このキュー スロットリング機能は、カスタム アプリケーション キューに対してだけサポートされており、デフォルト キューまたは WebLogic Server の内部キューに対してはサポートされていない。 コンフィグレーションされているキューの長さを超えると、`Kernel.execute()` の呼び出しの結果として、クライアントは回復可能なリモート例外と、503 番の応答を受け取る。
CR090071	`PosixSocketMuxer.cleanup()` でアサーション失敗エラーが大量に発生すると共に、CLOSE_WAIT 状態のソケットの数が増え続けた。最終的に `ulimit` に達し、サーバ インスタンスの再起動が必要であった。 クリーンアップにおいてアサーション エラーを送出する前に `fdr.sock` が null かどうかをチェックするロジックを実装することで、この問題を解決した。
CR090341	`logListenFailed()` を呼び出した `t3.srvr.ListenThread` で、誤った障害時間が表示されていた。IOException が原因でサーバがリスンに失敗したときに生成されるメッセージの値に誤りがあった。メッセージの例を次に示す。 `<Jun 11, 2002 2:37:33 PM PDT> <Critical> <WebLogicServer> <Failed to listen on port 7772, failure count: 3, failing for 1,023,831,450 seconds, java.net.SocketException: File table overflow>` `t3.srvr.ListenThread` にあった綴りの間違いを修正することで、この問題を解決した。
CR090823	e2e サンプルで、JSP の不要な再コンパイルが行われていた。b2c および b2b のサンプルを実行すると、e2e ドメインの JSP が必ず再コンパイルされた。ただし、これらの JSP のタイム スタンプは、正しく元に戻されていた。 原因は `JspStub.getClassLoader` メソッドのバグであった。バグを修正して問題を解決した。
CR091420	WebLogic Server 6.1 SP04 は、Unix Machine に対して [Post-Bind UID を有効化] オプションが指定されていると起動に失敗した。この属性は、Administration Console の [マシン] ノードでコンフィグレーションできる。この属性の目的は、UNIX マシンを実行するサーバ インスタンスが特権付きのすべての起動アクションを実行した後にその下で稼働する、Unix UID を返すことである。 コードを修正し、この問題を解決した。

CR092704	WebLogic Server 6.1 SP02 では、スーパー パッチを適用して実行すると、ソケット リーダー スレッドがブロックされるために、ハングが頻繁に発生した。POLL ロックを所有するスレッドが SSL ソケットを閉じようとしたが、`sendRecord()` メソッドにおいて必要な出力ストリームのロックを取得できないために、処理を進めることができなかった。 スタック トレースは、次のようなものであった。 `“ExecuteThread: '23' for queue: 'default'” daemon prio=5 tid=0x6d6c40 nid=0x24 waiting for monitor entry [0xe4e81000..0xe4e81a28] at weblogic.security.SSL.SSLSocket.sendRecord(SSLSocket.java:1049) at weblogic.security.SSL.SSLSocket.sendAlert(SSLSocket.java:1007) at weblogic.security.SSL.SSLSocket.close(SSLSocket.java:1153) at weblogic.security.SSL.SSLSocket.close(SSLSocket.java:1141) at weblogic.socket.SocketMuxer.closeSocket(SocketMuxer.java:236` アボートによるシャットダウンに対するクローズでは SSL ソケットがソケットにデータを書き込まないよう、コードを修正した。
CR092933	次のコンフィグレーションで、スレッド ダンプ エラーが発生した。 ■ VM: java.version='1.3.1_02' ■ os.name='windows 2000' ■ java.vendor.url='http://java.sun.com/' Hotspot クライアントおよび Version 1.3.1_x 仮想マシンに対して JVM_DumpAllStacks シンボルがグローバルに宣言されていなかったため、スレッド ダンプが失敗した。 1.3.1_X Hotspot クライアント / サーバ JVM において、weblogic.Admin を使用するスレッド ダンプが可能になった。

CR093416	Controller サーブレットがデプロイされていると、管理対象サーバが `EmptyStackException` を送出した。 Controller サーブレットのデプロイ中に、管理対象サーバが次のエラーを送出した。 `[WebAppServletContext(35527,btn,/btn)] Servlet failed with Exception> java.util.EmptyStackException at weblogic.utils. collections.Stack.pop()Ljava.lang.Object;(Unknown Source) at weblogic.kernel.ResettableThreadLocalStack.pop()Ljava.lang.Obj ect;(Unknown Source) at weblogic.jndi.internal. ThreadEnvironment.pop()Lweblogic.jndi.Environment;(Unknown Source) at weblogic.jndi.internal.WLContextImpl.close()V (Unknown Source) at weblogic.jndi.factories.java.ReadOnly ContextWrapper.close()V(Unknown Source) at be.btn.util. JndiUtil.getEnv(Ljava.lang.String;)Ljava.lang.Object;(Unknown Source) at be.btn.web.HttpControllerServlet.init()V(Unknown Source) at javax.servlet.GenericServlet.init(Ljavax.servlet. ServletConfig;)V(Unknown Source) at weblogic.servlet. internal.ServletStubImpl.createServlet()Ljavax.servlet.Serv let;(Unknown Source) at weblogic.servlet.internal. ServletStubImpl.createInstances()V(Unknown Source) at...` 問題は `btn.utils.JndiUtils.getEnv()` メソッドに関係しており、ルックアップで取得したコンテキスト `java:com/env` を閉じていたため、`javaURLContextFactory` が環境をプッシュしていなかった。 この問題を解決した。
CR094101	WebLogic Server 6.1 SP02 と WebLogic Server 7.0 SP01 の相互運用に問題があった。WebLogic Server 6.1 の MDB が、WebLogic Server 7.0 の EJB を呼び出していた。EJB はタイムアウトすると `weblogic.transaction.TimedOutException` を送出していたが、この例外は WebLogic Server 7.0 で新しく追加されたものであり、WebLogic Server 6.1 では利用できなかった。WebLogic Server 6.1 の MDB の `onMessage()` メソッドが例外をキャッチし、`Exception.printStackTrace()` のコードを実行するときに `ClassNotFoundException` を送出していた。 `weblogic.transaction.TimedOutException` クラスを追加し、WebLogic Server 7.0 サーバと相互運用する際に `ClassNotFoundException` が発生しないようにすることで、この問題を解決した。

CR094724	WebLogic Server 6.1 SP03 以降では、HP-UX での WebLogic Server ネイティブ ライブラリが、aCC ではなく、cfront を使ってコンパイルされていた。aCC は、1998 年から cfront の代わりに使われるようになった ANSI C コンパイラである。 その結果、互換性のないランタイム ライブラリがロードされていた (cfront のコンパイルでは libC.2、また aCC でコンパイルされた SUN の Java では libCsup.2)。互換性のないランタイム ライブラリを使用するとクラッシュする場合がある。 HP-UX 11 に対しては aCC を使用するように WebLogic Server のビルドを変更することで、この問題を解決した。
CR095267	「CR094561」と同じ。
CR095487	「CR087808」と同じ。
CR095949	WebLogic Server 6.1 SP04 では、起動クラスの FailureIsFatal オプションを true に設定してあっても、起動クラスで障害が発生したときに、意図したように WebLogic Server がシャットダウンしなかった。 StartupClassRunner は、アプリケーションがデプロイされる前と後に呼び出される。分析の結果、アプリケーションがデプロイされる前に起動クラスで障害が発生し、起動クラスに failureIsFatal が設定された場合、fatalException が失われていた。 fatalException が失われず、サーバ インスタンスが正しくシャットダウンされるようにコードを修正することで、この問題を解決した。
CR096114	「CR092933」に関する変更 (Sun JDK の 1.3.1_x バージョンでスレッド ダンプのリダイレクションを有効にするというもの) が、stdio のハンドルが無効であったために、NT サービスでは機能しなかった。 beasvc で stdio のハンドルを作成するようにコードを変更することで、この問題を解決した。
CR101322	クライアントが、RMI の発信呼び出しを行うカスタム レルムを使用していた。これは、リーダー スレッドの RJVM レイヤ内の BootServicesImpl.invoke() から発生した。このような呼び出しが多く行われると、リーダー スレッドは発信呼び出しをブロックされて、応答を読み取るためのスレッドが残らなかった。 BootServices を RMI レイヤに移動し、デフォルトの実行キューにディスパッチできるようにすることで、この問題を解決した。

CR102021	WebLogic Server 6.1 SP04 では、負荷テストにおいて、PosixMuxer が以下の例外を送出した。 `<Nov 21, 2002 9:38:29 AM EST> <Error> <socket> <000421> <Uncaught Throwable in processSockets java.io.IOException: unexpected exception in poll: (4) Interrupted system call java.io.IOException: unexpected exception in poll: (4) Interrupted system call at weblogic.socket.PosixSocketMuxer.poll(Native Method) at ...` 中断した場合はポーリングを再び開始するようコードを修正し、この問題を解決した。
CR102058	クライアントでのリモート メソッド呼び出しで `URLClassLoader` を使用すると、`NoClassDefFoundError` が発生した。 `c:\java\java131_07\bin\java -classpath ".;client.jar;C:\home\ravia\weblogic\dev\src700\3rdparty\weblogicaux.jar" URLTest qa146 9901 ejb20-statefulSession-TraderHome installadministrator installadministrator java.lang.NoClassDefFoundError: weblogic/rmi/extensions/server/Stub at java.lang.ClassLoader.defineClass0(Native Method) at java.lang.ClassLoader.defineClass(ClassLoader.java:488) at...` 分析の結果、`AugmentableSystemClassLoader` を作成するときに、それを常に `SystemClassLoader` に設定していることがわかった。このため、クライアントがシステム クラスローダを使用し、`NoClassDefFoundError` が発生した。 `AugmentableSystemClassLoader` に対する親として `ThreadContextClassLoader` を設定することで、この問題を解決した。

デプロイメント

変更要求番号	説明
CR089031	cr058358_61sp2.jar を適用した 6.1 SP02 を SP03 にアップグレードすると、管理対象サーバにおけるデプロイメントのパフォーマンスが低下していた。管理サーバでは低下しなかった。 CR058358 に対して実装された `weblogic.j2ee.Component.getModuleMBean()` メソッドに関して問題があった。`ApplicationDescriptorMBean` は管理サーバだけに登録されているので、管理対象サーバがこの MBean を要求すると、必ず管理サーバに対する RMI 呼び出しが行われる。 管理対象サーバにローカル キャッシュの MBean (`ApplicaitonDescriptorMBeanImpl_Cached`) を作成することで、この問題を解決した。メソッドに対する最初のリクエストは管理サーバに送られて、取得された情報はローカルに保存される。それ以降のリクエストはローカルに処理され、アプリケーションがアンデプロイされるとキャッシュは無効になる。
CR089765	Windows 2000 で `weblogic.refresh` を実行して、Linux 上で稼働する WebLogic Server の管理サーバおよび管理対象サーバに対するファイル (JSP または HTML) を更新すると、`java.util.zip.ZipException` が発生した。 この ZIP 例外が発生するのは、WebLogic Server が管理サーバまたは管理対象サーバとしてコンフィグレーションされていて、`weblogic.refresh` を実行する Web アプリケーションの対象が管理対象サーバになっている場合である。 管理対象サーバが更新されたファイルを管理サーバから取得しようとすると、両方のサーバ インスタンスが例外を送出する。 次のコンフィグレーションで発生した。 ■ Linux (Redhat、JDK1.3.1) 上の WebLogic Server 6.1 SP03 (管理サーバが 1 つ、管理対象サーバが 1 つ) ■ Windows 2000 で `weblogic.refresh` を呼び出す (WebLogic Server 6.1 SP3、JDK1.3.1) 問題を診断した結果、管理対象サーバが管理サーバに対して示す更新されたファイルの位置のパスにエラーがあることがわかった。`FileDistributionServlet.doGetMultipleJspRefreshRequest` を修正することで問題を解決した。

CR091897	WebLogic Server 6.1 SP04 では、ab.war や i.war のような短い名前の Web アプリケーションのデプロイメントが失敗した。次のエラーが発生した。 java.lang.IllegalArgumentException: Prefix string too short at java.io.File.createTempFile(File.java:1232 at weblogic.j2ee.Component.retrieveComponent(Component.java:296 at weblogic.j2ee.Component.<init>(Component.java:188)at weblogic.j2ee.WebAppComponent.<init>(WebAppComponent.java:45at weblogic.j2ee.Application.addComponent(Application.java:149)at weblogic.j2ee.J2EEService.addDeployment(J2EEService.java:117)... このバグを修正した。
CR097560	WebLogic Server 6.1 SP04 では、カスタム InitialContextFactory を weblogic.jndi.Environment に設定しても、getInitialContext() は代わりに DEFAULT_INITIAL_CONTEXT_FACTORY を使用した。 InitialContextFactory が指定された場合は weblogic.jndi.Environment オブジェクトがそれを考慮するようコードを修正することで、この問題を解決した。

EJB

変更要求番号	説明

CR063275	WebLogic Server の EJB では、byte [] フィールドにファインダ メソッドを書き込むことができなかった。これは、「EJB QL で許される型は、エンティティ Bean と依存オブジェクトの抽象スキーマ型、cmp-field の定義済みの型、およびリモート エンティティ Bean のエンティティ オブジェクト型である」という J2EE EJB-QL の定義に準拠していない。この問題は WebLogic Server 6.1 SP01 で発見された。このエラーはビルド時に発生した。
	`ejbc:`
	`  [java]`
	`  [java] ERROR: Error from ejbc: Error while reading 'META-INF/weblogic-cmp-rdbms-jar.xml'. The error was:`
	`  [java]`
	` [java] invalid query: In EJB containerManaged, for a query defined in the ejb-jar.xml file with a method signature, findFk(byte[]), we failed to find a corresponding method in the remote home interface, local home interface, or bean class that matches this signature. Note that class parameters such as java.lang.String must be fully qualified, thus 'String' would not match 'java.lang.String'.[java]`
	この問題を解決した。WebLogic Server EJB は、byte [] フィールドに対するファインダ メソッドをサポートするようになった。

CR063837	WebLogic Server 6.1 SP02 では、EJB コンテナが存在していないデータベース テーブルとカラムの存在を検査しようとすると、次のエラーが送出された。 `weblogic.utils.AssertionError: ***** ASSERTION FAILED *****[Table: Cmp_Birthday Full Table Check failed, but table all columns were found!] at weblogic.ejb20.utils.TableVerifier.verifyTableAndColumnsExist(TableVe rifier.java:371) at weblogic.ejb20.utils.TableVerifier.verifyTableExistsAndCreateM aybe(Ta bleVerifier.java:391)` コードを修正し、この問題を解決した。

CR076272	EJB をホット デプロイするときのコンパイル エラー情報が不親切だった。たとえば、次のようなものであった。
	`<30 avr. 02 11:03:14 BST> <Error> <Management> <Error deploying application.\config\mydomain\applications\ldapprofile.jar:java.lang.reflect.UndeclaredThrowableException>`
	ホット デプロイされる EJB に対するエラー レポートを改善した。実行時の例外とエラーを、スタンドアロンの ejbc で使われているのと同様の方法で報告するようにした。サーバのログと標準出力で、障害の理解に役立つ情報が提供される。次に例を示す。
	`<Oct 1, 2002 4:37:18 PM PDT> <Error> <J2EE> <Error deploying application ldapprofile: Unable to deploy EJB: ldapprofile.jar from ldapprofile.jar: java.lang.NoClassDefFoundError: com/bea/p13n/property/EntityPropertyManager at java.lang.ClassLoader.defineClass0(Native Method) at java.lang.ClassLoader.defineClass(ClassLoader.java:486) at java.security.SecureClassLoader.defineClass(SecureClassLoader.java:111) at weblogic.utils.classloaders.GenericClassLoader.findLocalClass(GenericClassLoader.java:394) at weblogic.utils.classloaders.GenericClassLoader.findClass(GenericClassLoader.java:157) at java.lang.ClassLoader.loadClass(ClassLoader.java:297) at java.lang.ClassLoader.loadClass(ClassLoader.java:253) at.....`

CR076386	6.1 SP03 では、`cmr-field` のみと共に `cmp-ejb` を使用すると、正しくない SQL `insert` 文が生成された。この問題は、MS-SQL Server 2000 で `Automatic-Primary-Key-Generation-TPYE : SQL_SERVER` を指定すると発生した。コンテナによって生成される SQL 文は次のようなものであった。
	`Insert into tblLicence(, 5 , 7) values (,licCompID,licOrgID)`
	`compID` と `orgID` は、このエンティティ Bean に対応するテーブルの外部キーである。`cmp-field` を追加した後では、変更されていない (空の) `ejbCreate-Method` は正しい SQL を生成する。
	`Insert into tblLicence(licArchivNR,licCompID,licOrgID)values (null , 5 , 7)`
	この問題を解決した。Bean に (pk) フィールドと cmr フィールドだけがある場合でも、コンテナは不正な SQL `Insert` 文を生成しなくなった。

CR080331	6.1 SP03 では、Administration Console を使ってバージョン 5.1 の EJB jar をデプロイすると、ProcessorFactoryException が発生した。
	`Unable to deploy EJB: TxRolledbackExceptionBean.jar from TxRolledbackExceptionBean.jar:`
	`The XML parser encountered an error in your deployment descriptor. Please ensure that your DOCTYPE is correct. You may wish to compare your deployment descriptors with the WebLogic Server examples to ensure the format is correct. The error was:`
	`weblogic.xml.process.ProcessorFactoryException: The public id, "-//BEA Systems, Inc.//DTD WebLogic 5.1.0 EJB//EN ", specified in the XML document is invalid.  Use one of the following valid public ids:`
	`"-//BEA Systems, Inc.//DTD WebLogic 5.1.0 EJB//EN"`
	`"-//BEA Systems, Inc.//DTD WebLogic 6.0.0 EJB//EN"`
	この問題は、デプロイメント記述子に対する DTD 宣言中のホワイト スペースを WebLogic Server 5.1 がサポートしているために発生していた。
	キーを使ってハッシュ テーブル中のパーサを検索する前にこのキー文字列から余分なスペースを取り除くようにコードを変更することで、この問題を解決した。

CR083239	EJB QL NOT MEMBER 句で無限ループが発生しなくなった。
CR083240	EJB QL NOT MEMBER 句と WHERE を併用しても、不正なクエリが作成されなくなった。
CR084978	Administration Console に [EJBC オプション] フィールドを追加し、Console を使って ejbc にヒープ サイズや他のオプションを渡すことができるようにした。

CR087151	ステートレス セッション Bean に関して、以下の問題があった。
	`<stateless-bean-is-clusterable>False</stateless-bean-is-clusterable>`
	weblogic-ejb-jar.xml で上のように指定されていて、Bean をクラスタにデプロイすると、次のエラーが発生した。
	`<Mar 14, 2003 3:35:00 PM PST> <Error> <Cluster> <Conflict start: You tried to bind an object under the name ejb20-statelessSession-TraderHome_EO in the JNDI tree. The object you have bound from 172.17.24.112 is non clusterable and you have tried to bind more than once from two or more servers. Such objects can only deployed from one server.>`
	クラスタ化されていないスタブに対する JNDI バインドはレプリケートしないようコードを修正することで、この問題を解決した。

CR088526	WebLogic Server 6.1 SP03 では、EJB コンテナが EJB 2.0 の仕様「BMT 使用中に例外が送出されたときに、ロールバックするよう設定されていないトランザクション」に違反していた。 BMT を使用する MDB で `onMessage` メソッドが例外を送出したときに問題があった。コンテナが、EJB 2.0 仕様 (18.3.2 節の表 18) に従って例外を処理していなかった。コンテナは、例外を記録し (行われていた)、Bean インスタンスを削除し (行われていた)、トランザクションをロールバックするようにマークしなければならなかった (行われていなかった)。結果として、トランザクションがスレッドと関連付けられたままになっていた。 コードを修正して、この問題を解決した。

CR089759	WebLogic Server 6.1:SP03: [EJB] : `UserTransaction` コンテキストを使用して `ejbStore()` から `context.getCallerPrincipal()` を呼び出すと、次の例外が送出された。 `<Oct 24, 2002 5:52:24 AM PDT> <Info> <EJB> <Exception from ejbStore:javax.ejb.EJBException: ejbStore: nulljavax.ejb. EJBException: ejbStore: null at AccountBean.ejbStore(Account Bean.java:99)atAccountBean_t4qrab_Impl.ejbStore (AccountBean_t4qrab_Impl.java:131)at weblogic.ejb20.manager. DBManager.storeBean(DBManager.java:266) at weblogic.ejb20. manager.DBManager.beforeCompletion(DBManager.java:397) at weblogic.ejb20.internal.TxManager$TxListener.beforeCompletion(TxManager.java:494) at weblogic.transaction.internal. ServerSCInfo.callBeforeCompletions(ServerSCInfo.java:551) at weblogic.transaction.internal.ServerSCInfo.startPrePrepareAndC hain(ServerSCInfo.java:88)at weblogic.transaction. internal.ServerTransactionImpl.localPrePrepareAndChain(ServerT ransactionImpl.java:979)at weblogic.transaction. internal.ServerTransactionImpl.globalPrePrepare(ServerTransact ionImpl.java:1503) at weblogic.transaction.internal. ServerTransactionImpl.internalCommit(ServerTransactionImpl.jav a:215) at weblogic.transaction.internal.Server TransactionImpl.commit(ServerTransactionImpl.java:189)at weblogic.transaction.internal.CoordinatorImpl.commit(Coordinat orImpl.java:68) at weblogic.transaction.internal. CoordinatorImpl_WLSkel.invoke(Unknown Source)at weblogic.rmi. internal.BasicServerRef.invoke(BasicServerRef.java:305)at weblogic.rmi.internal.BasicServerRef.handleRequest(BasicServer Ref.java:274) at weblogic.rmi.internal.BasicExecute Request.execute(BasicExecuteRequest.java:22) atweblogic.kernel.ExecuteThread.execute(ExecuteThread.java:139) at weblogic.kernel.ExecuteThread.run(Execute Thread.java:120)><Oct 24, 2002 5:54:53 AM PDT> <Info> <Management> <Configuration changes for domain saved to the repository.>` この問題は、EJB 仕様の 21.2.5.1 節に準拠するために SP03 で行われた変更が原因であった。この問題を解決した。

CR089953	SP03 では、ejb20.locks.ExclusiveLockManager のテストでメモリ リークが発見されていた。メモリ リークは、デプロイとアンデプロイの結果として発生した。分析の結果、EJB がアンデプロイされたときに DiskSwap のタイマ / トリガがキャンセルされず、TimeGenerator に参照が残り、ExclusiveLockManager バケットのガベージ コレクションが行われていなかった。 タイマ / トリガを正しくキャンセルし、適切にガベージ コレクションが行われるようにコードを修正することで、この問題を解決した。
CR090143	MDB に対するデフォルトのプール サイズがスレッドのプール サイズのデフォルトより大きいため、MDB インスタンスがデフォルトのスレッド プールと、非 MDB 作業の完了を待つブロッキングをすべて使用してしまい、非ブロッキング作業を行うためのスレッドが残らなかった。 メイン スレッド プールに MDB を割り当てるときは MDB プール サイズを ((スレッド プール サイズ) - (ソケット リーダ スレッド)) の割合に制限することで、この問題を解決した。
CR090515	Administration Console で、エンティティ Bean に対する待機数の値が正しく表示されなかった。クライアントがタイムアウトした後でも、待機している EJB があるように Console では表示されていた。 待機が終わったときには ExclusiveLockManager が waiterTotalCount をデクリメントするように修正することで、この問題を解決した。
CR091436	SP03 では、JMSConnectionPoller が初期コンテキストを閉じていなかった。MDB がルックアップに外部の InitialContextFactory を使用し、JMS プロバイダがアクセス不能であると、接続および他のリソースが急速に蓄積していた。 アクティブなメモリのクリーンアップが済んだときにはコンテキストを閉じるようにコードを修正することで、この問題を解決した。

CR091722	SP03 では、大きな負荷がかかっている場合、ビジネス メソッドを呼び出した後でトランザクションを再開すると、MDB が `IllegalStateException` を送出し、トランザクションの再開中に JVM がクラッシュした。 この問題は、次のようなアプリケーションにおいて発生した。クライアントがキューにメッセージ送信し、MDB がキューをリスンする。この MDB に対する `tx.attribute` は「`Required`」である。MDB の作成メソッドが BMT SLSB を作成するために `home.create` を行っていて、`onMessage` メソッドがこの SLSB に対するビジネス メソッドを呼び出している。そして、この SLSB のビジネス メソッドが、2 つの異なるキューにメッセージを送信している。 この場合、次のようなエラーが発生した。 `<EJB Exception during invocation from home: com.gfs.corp.batch.cost.costupdate.ejb.NextOrderCostUpdateProcessorEJB_xtvwzj_LocalHomeImpl@54b120 threw exception: java.lang.StackOverflowError> java.lang.StackOverflowError <<no stack trace available>>` ローカル インタフェースが、例外を正しく処理していなかった。 例外が発生したときはコンテナがトランザクションを正しくロールバックするようエラー処理のコードを修正することで、この問題を解決した。
CR093850	SP04 では、エンティティ Bean の実行時モニタが、現在キャッシュされている Bean のカウントの値を誤って報告していた。値が、デプロイメント記述子で指定されているキャッシュ内の最大 Bean 数だけでなく、現在キャッシュされている Bean のカウントを超えていた。 報告される値が正しくなく、新しい Bean が不適切に作成されていた。 Bean をキャッシュから削除した後で、Bean を解放してプールに戻すよう、コードを修正した。

CR095173	`weblogic-ejb-jar.xml` の `idle-timeout-seconds` 要素は、ステートフル セッション Bean をパッシベーションするまでに EJB コンテナが待機する時間の長さを決定する。つまり、この時間が過ぎると、EJB はキャッシュから削除されて、ディスクに書き込まれる。また、EJB コンテナは、パッシベーションされた EJB をディスクから削除するまでの待機時間の長さを決めるためにも、この要素を使用していた。しかし、場合によっては、`idle-timeout-seconds` より長くステートフル セッション Bean をディスクに残しておきたいことがあった。つまり、ステートフル セッション Bean がキャッシュ内でアイドル状態になっている長さと、ディスク上でアイドル状態になっている長さを、異なる 2 つの要素で指定したいことがあった。 新しい要素 `session-timeout-seconds` を追加した。この要素は、アイドル状態のステートフル セッション Bean をディスクから削除するまでに EJB コンテナが待機する長さを指定する。

CR094524

WebLogic Server 6.1 SP03 および SP04 では、多対一のコンテナ管理関係 (CMR) の読み取り専用 EJB で LockTimedOutException が発生した。

```
avant getCustomerVO.getCountryVO() [ExclusiveLockManager$Lock
Bucket] : ** LOCK ACQUIRE --> WAITING -- ejb-name: Country
primary key: 002 lockClient: Name=[EJB charu.ejbrelations
final.CountryBean.getCountryVO()],Xid=11:1d0d46b2(3053175),Sta
tus=Active,numRepliesOwedMe=0,numRepliesOwedOthers=0,seconds
since begin=0,seconds left=30,activeThread=Thread
[ExecuteThread: '2' for queue: 'default',5,Thread Group for
Queue: 'default'],SCInfo[mydomain+myserver]=(state=
active),properties=({weblogic.transaction.name=[EJB
charu.ejbrelationsfinal.CountryBean.getCountryVO()],
LOCAL_ENTITY_TX=true}),OwnerTransactionManager=ServerTM[Server
CoordinatorDescriptor=(CoordinatorURL=myserver+172.23.135.56:7
011+mydomain+, Resources={})]) wait (MS): 30000
[ExclusiveLockManager$LockBucket] : ** LOCK TIME OUT AFTER
WAITING -- ejb-name: Country primary key: 002 lockClient:
Name=[EJB charu.ejbrelationsfinal.CountryBean.get
CountryVO()],Xid=11:1d0d46b2(3053175),Status=Marked rollback.
[Reason=weblogic.transaction.internal.TimedOut
Exception: Transaction timed out after 29 seconds Name=[EJB
charu.ejbrelationsfinal.CountryBean.getCountryVO()],Xid=11:1d0
d46b2(3053175),Status=Active,numRepliesOwedMe=0,numRepliesOwed
Others=0,seconds since begin=29,seconds left=30,active
Thread=Thread[ExecuteThread: '2' for queue:'default',5,Thread
Group for Queue: 'default'],SCInfo[mydomain+myserver]=
(state=active),properties=({weblogic.transaction.name=[EJB
charu.ejbrelationsfinal.CountryBean.getCountryVO()],
LOCAL_ENTITY_TX=true}),OwnerTransactionManager=ServerTM[Server
CoordinatorDescriptor=(CoordinatorURL=myserver+172.23.135.56:7
011+mydomain+, Resources={})])],numRepliesOwedMe=0,
numRepliesOwedOthers=0,seconds since begin=30,seconds
left=10,activeThread=Thread[ExecuteThread: '2' for queue:
'default',5,Thread Group for Queue:'default'],SCInfo [mydomain+
myserver]=(state=active),properties=({weblogic.
transaction.name=[EJB charu.ejbrelationsfinal.CountryBean.
getCountryVO()], LOCAL_ENTITY_TX=true}),OwnerTransaction
Manager=ServerTM[ServerCoordinatorDescriptor=(CoordinatorURL=m
yserver+172.23.135.56:7011+mydomain+, Resources={})]) wait
(MS): 30000
```

if condition チェックを rbd.isReadOnly() から READONLY_EXCLUSIVE_CONCURRENCY のチェックに変更することで、この問題を解決した。

CR095545	同じ Weblogic Server インスタンス上に、ejb-name は同じだが JNDI マッピングが異なる 2 つのメッセージ駆動型 Bean をデプロイしようとすると、最初の Bean は正常にデプロイされたが、2 番目の Bean のデプロイメントは次の例外で失敗した。 weblogic.management.ManagementException: - with nested exception: [javax.management.InstanceAlreadyExistsException: jpdomain:Location=jpserver,Name=MessageQueueHandlerBean,ServerRuntime=jpserver,Type=EJBMessageDrivenRuntime] at weblogic.management.runtime.RuntimeMBeanDelegate.register(RuntimeMBeanDelegate.java:96) at weblogic.management.runtime.RuntimeMBeanDelegate.<init>(RuntimeMBeanDelegate.java:83) at weblogic.management.runtime.RuntimeMBeanDelegate.<init>(RuntimeMBeanDelegate.java:53) at weblogic.management.runtime.RuntimeMBeanDelegate.<init>(RuntimeMBeanDelegate.java:63) at weblogic.ejb20.deployer.MessageDrivenRuntimeMBean.<init>(MessageDrivenRuntimeMBean.java:18) at weblogic.ejb20.deployer.MessageDrivenBeanInfoImpl.deploy(MessageDrivenBeanInfoImpl.java:450) at weblogic.ejb20.deployer.Deployer.deployDescriptor(Deployer.java:1299) at weblogic.ejb20.deployer.Deployer.deploy(Deployer.java:1005) この問題は、MessageDrivenRuntimeMBean 名がユニークでないことが原因であった。ユニークな名前を作成するようコードを修正し、この問題を解決した。
CR096848	6.1 SP04 では、CMP 2.0 の Bean に対して <is-modified-method-name> が呼び出されていなかった。Bean のメソッドを呼び出すたびに ejbStore を呼び出し、後でストアを回避するかどうかを判定していた。そのため、<is-modified-method-name> を頻繁に使用するアプリケーションでは、パフォーマンスの問題が発生していた。 CMP EJB 2.0 に対する <is-modified-method-name> 機能を実装することで、この問題を解決した。

CR099420	引数として 2 次元配列を指定すると、ejbc コンパイラが次のエラーで失敗した。 `Unable to set the method permission for method "isAuthorized(java.lang.String,[[java.lang.String)". No matching method could be found. Please verify the method signature specified in the ejb-jar.xml file matches that of your EJB. ERROR:ejbc found errors.` 渡される配列の次元に基づいて適切なメソッド パラメータ シグネチャを生成するように、`eblogic.ejb20.deployer.mbimpl.MethodDescriptorImpl` クラスの `makeMethodParameter()` メソッドを修正することで、この問題を解決した。
CR101384	`MDListener` で `UserTransaction.commit` が失敗すると、それ以降に JMS から呼び出される MDB は、スレッドに既にトランザクションが存在するため、`UserTransaction.begin` で失敗した。次の一連のイベントが発生した。 1. MDB が別のサーバにある JMS キューからメッセージを取得している。 2. JMS サーバが停止する。 3. `weblogic.ejb20.internal.MDListener` の内部で、`UserTransaction.commit` が例外を送出する。 4. `MDListener` は存続時間の最後まで存在する。 5. それ以降に JMS から呼び出された MDB は、`UserTransaction.begin` で失敗する。 コードを修正し、この問題を解決した。予期しない例外によってトランザクションを完了できない場合、`MDListener` は、新しいトランザクションを開始する前に現在のトランザクションをサスペンドする。

CR101315	jsp:setProperty パラメータの変換の失敗が報告された。 jsp:setProperty のアクションが Bean 内のプロパティの値を設定するときに、ターゲットの型を決定するためのターゲット プロパティを使用し、JSP.2.13.2.1 に従って、パラメータが変換されていた。jsp:setProperty アクションを含む JSP ファイルは、次のエラー メッセージでコンパイルに失敗していた。 `... setCount(int) in weblogic.qa.xmlbasedtests.webapp_jsp_tests.FooBean cannot be applied to (java.lang.String) ... setBool(boolean) in weblogic.qa.xmlbasedtests.webapp_jsp_tests.FooBean cannot be applied to (java.lang.String) ... setmyDouble(double) in weblogic.qa.xmlbasedtests.webapp_jsp_tests.FooBean cannot be applied to (java.lang.String) ... setFloat(float) in weblogic.qa.xmlbasedtests.webapp_jsp_tests.FooBean cannot be applied to (java.lang.String) ... setmyLong(long) in weblogic.qa.xmlbasedtests.webapp_jsp_tests.FooBean cannot be applied to (java.lang.String)` この動作は、「CR098402」に関連する変更の結果であった。この CR では jsp:setProperty に関する問題が解決され、プロパティ値の中にある要求時の式が String に変換された。要求時属性として渡される値から割り当てるときは、型の変換は適用されない。 調査の結果、仕様では次のよう規定されていることがわかった。 「要求時属性として渡された値から代入するときは、JSP.2.13.2.3 節で示されているように、型の変換を適用しない。」 コードは仕様に従って機能しているものと判断され、この CR を解決済みにするためにコードの修正は不要であった。

インストーラ

変更要求番号	説明
CR092156	WebLogic Server 6.1 SP04 アップグレード インストーラのビルドに、最新の beasvc.exe が含まれていなかった。関連する CR については、6-129 ページの「CR091420」および 5-15 ページの「091420」を参照すること。 SP05 のアップグレード インストーラでは、この問題は発生していない。

JDBC

変更要求番号	説明
CR085559	Oracle データベースがダウンしているためにユーザ トランザクションがロールバックに失敗すると、トランザクションで使用されている JDBC 接続が解放されず、接続プールに空きがなくなっていた。 この問題は、WebLogic Server 6.1 SP02、Oracle 8.1.6 または 8.1.7 の jDriver for Oracle、および Oracle シン ドライバにおいて発生した。 分析の結果、JTS 接続の `internalrollback` メソッドが `internalclose()` メソッドを呼び出していないため、ロールバックが失敗すると接続がリークすることがわかった。 コードを修正してこの問題を解決した。
CR089713	WebLogic Server 6.1 SP03 では、`weblogic.Admin RESET_POOL` が接続を非予約状態に再初期化していなかった。 `weblogic.Admin RESET_POOL` を使用した後、ログはすべての接続が更新されたことを示していたが、リセットの前に予約されていた接続は、リセットの後でもまだ予約されていた。 予約されたリソースを `resourcesFree` リストに戻して関連するフラグをクリアするようにコードを修正し、この問題を解決した。
CR090379	WebLogic Server の以前のリリースでは、アプリケーション コードが、閉じられないまま破棄される JDBC オブジェクトを作成した場合、オブジェクトは失われるが、ガベージ コレクションされた後でもメモリまたはオープン カーソルが保持されていた。そのようなオブジェクトが多数作成されると、最終的にサーバはメモリ不足になり、データベースではカーソルが不足する。 WebLogic Server 6.1SP5 では、破棄された JDBC オブジェクトがガベージ コレクションの前に閉じられるようにコードを変更した。 **注意：** 破棄されたオブジェクトと同一の JDBC オブジェクトを直ちに作成すると、WebLogic Server は元の JDBC オブジェクトの複製を作成する。ただし、リークされた元のオブジェクトを閉じると、複製も影響を受ける。 『WebLogic JDBC プログラミング ガイド』の「JDBC オブジェクトを閉じる」で説明されたベスト プラクティスに従うと、JDBC オブジェクトの破棄を回避できる。

CR090761	OCI ドライバで StringIndexOutOfBoundsException が発生した。この問題は、CMP エンティティ Bean を実行するときには、複数のクエリにおいてときどき発生した。また、weblogic.jdbc.pool.ResultSet.getLong(ResultSet.java:107) の箇所またはその近くの積分 (データベース内の数値の精度によって異なる) を扱うときは、常に発生した。 例外の全文は次のとおりである。 java.sql.SQLException: java.lang. StringIndexOutOfBoundsException - String index out of range: 0 分析の結果、weblogic.db.oci.OciCurser.getValue (OciColumn,int,int,boolean,boolean) において、適切なチェックを行わないで new Long(string) メソッドと new BigInteger(string) メソッドを呼び出していたため、StringIndexOutOfBoundsException が発生していた。 コードを修正してこの問題を解決した。
CR091577	WebLogic Server 6.1 SP03 では、接続プール リセット メソッドについて不要な同期を取っていた。ResourceAllocator の resetThisOne() メソッドの同期が取られていた。その結果、すべての実行スレッドが testConnOnReserve を実行して応答のない接続を置き換える際に、並行して接続を確立するのではなく、キューに入れていた。 コードを修正してこの問題を解決した。

CR092453	WebLogic Server 6.1 SP04 では、Oracle 9.2 に付属する CLASSES12.zip の Oracle Thin XA により、EJB を呼び出すステートレス セッション Bean において、「コンフィグレーションの変更がリポジトリに保存された」というメッセージの後で、XAER_PROTO が発生していた。 START SLEEP 2: After updating thevalue to 1... DONE SLEEP 2: After updating thevalue to 1... START SLEEP 2: After updating thevalue to 2... DONE SLEEP 2: After updating thevalue to 2... START SLEEP 2: After updating thevalue to 3... DONE SLEEP 2: After updating thevalue to 3... START SLEEP 2: After updating thevalue to 4... DONE SLEEP 2: After updating thevalue to 4... START SLEEP 2: After updating thevalue to 5... DONE SLEEP 2: After updating thevalue to 5... Current value is 5 < Dec 6, 2002 10:26:59 PM MST> <Info> <Management> <Configuration changes for domain saved to the repository.> SQLException -- XA error: XAER_PROTO : Routine was invoked in an improper context start() failed on resource 'OracleXA' null Current value is 0 この問題は、Oracle クライアント 9.2.0.[01] の既知の問題が原因で発生しており、9.2.0.2 で解決される。WebLogic Server では、9.2.0.[01] の問題を回避するためのコード修正を行った。
CR092791	WebLogic Server 6.1 (すべての SP) では、Oracle Thin Driver に基づく接続プールを通して、Oracle の特定のオブジェクト (Array、Struct など) を使うことができなかった。Oracle Thin Driver が返すオブジェクトは シリアライズ可能でもリモートでもなく、そのため RMI を介して渡すことができなかった。 新しいパッケージ weblogic.jdbc.vendor.oracle を導入した。パッケージにはこのようなオブジェクトのための「プロキシ」が含まれており、接続プールを通して渡すことができる。
CR093245	JDBCConnectionPoolRuntimeMBean.resetStatementProfile を呼び出した後、SQL トレースを表示できなかった。resetStatementProfile() を呼び出すとトレース ファイルはクリアされたが、JdbcSqlTraceAdmin ツールでトレースを有効にしても、トレースを行うことができなかった。トレースを再度有効にするには、サーバ インスタンスを再起動する必要があった。 コードを修正してこの問題を解決した。

CR093563	「ファイナライザ」と `weblogic/jdbc/jta/Statement.java` の間で、デッドロックが発生した。 このエラーは、Solaris 5.8 - WebLogic Server 6.1 SP3 - java v1.3.1_03 (ビルド 1.3.1_03-b03)、Java HotSpot(TM) Client VM (ビルド 1.3.1_03-b03、混合モード) において明らかになった。次のようなエラーが発生した。 `"ExecuteThread: '5' for queue: 'default'": waiting to lock monitor 0xe1ba8 (object 0xf1327fb0, a java.util.HashSet), which is locked by "Finalizer" "Finalizer": waiting to lock monitor 0xe1b70 (object 0xf1327fe8, a weblogic.jdbc.jta. ResultSet), which is locked by "ExecuteThread: '5' for queue: 'default'"` この問題は、Oracle データベースに対するクエリと更新を行う Web アプリケーションを実行するサーバ インスタンスにおいて発生した。デッドロックが発生するコードは、ユーザのログイン時にユーザ情報を収集する共通のユーザ管理インタフェースの一部であった。このコードでは、WebLogic Server でコンフィグレーションされている接続プールの `oracle.jdbc.xa.client.OracleXADataSource` を使用していた。 解析の結果、接続テストが同期を取っているため、利用できるロックがない場合でも、ロックを取得するために待っていた。 コードを修正してこの問題を解決した。
CR093734	WebLogic Server 6.1 では、クライアントが Oracle シン ドライバを使って利用できないデータベースに接続しようとしたときに、正しいエラー コードが返されなかった。 ■ `Driver.connect()` を使って接続を直接取得すると、`SQLExcetpion.getErrorCode()` は 17002 を返した。 ■ Oracle シン ドライバがコンフィグレーションされていて、`TestConnectionsOnReserve` の設定が true である接続プールから、データソースを使って接続を取得すると、`SQLExcetpion.getErrorCode()` は 0 を返した。 分析の結果、`weblogic.jdbc.common.internal.ConnectionEnvFactory` がエラー コードを取り込んでいた。 コードを修正してこの問題を解決した。
CR094645	0 以外の開始インデックスを `getStatementProfiles()` メソッドに渡すと、トレースは、指定した開始インデックスではなく、`.tsf` ファイルの先頭からデータを返した。 コードを修正してこの問題を解決した。

CR095059	WebLogic Server 6.1 SP04 では、`JDBCStatementProfile` インタフェースの `JDBCStatementProfile.getTimeTaken()` メソッドが常に 0 を返した。 `weblogic/jdbc/common/internal/ProfileStorage.java` のコードを修正して、この問題を解決した。
CR095994	`weblogic.jdbc.rmi.internal.ConnectionImpl` に、接続リーク プロファイリングと接続リーク検出のロジックがなかった。 接続が解放されてプールに戻される可能性があるのは、次の 3 つのイベントの場合である。 1. クライアントからの peerGone 2. DGC の間に `BasicServerRef` から参照されない呼び出し 3. リモート クライアントのコードによる接続の適切なクローズ 最初の 2 つのイベントでは、接続リークの可能性があることの警告が、ユーザに対して示されなければならない。 `SerialConnection.java` と `ConnectionImpl.java` にこのロジックを追加した。
CR096710	Weblogic Server 6.1 SP04 では、データベースに十分な数のプロセス (`max_processes`) がないために、指定された初期容量で接続プールを作成できない場合、ログに対して ORA-00020 メッセージが送出されていなかった。 メッセージをログに記録するようコードを修正し、この問題を解決した。
CR096922	負荷のある状態で、WebLogic Server が `ResourceAllocator.markBorrowed()` を呼び出し、JMX が `ConnectionPoolRuntimeMBeanImpl.getConnectionDelayTime()` を呼び出すと、デッドロック状態が発生した。 `ResourceAllocator.java` のコードを修正し、この問題を解決した。

CR097832	Weblogic Server 6.1 SP04 シン ドライバ、Solaris 8、JDK 1.3.1_04、および Oracle 8.1.7.4. というコンフィグレーションでは、接続の更新が有効になっていると、デッドロックが発生した。 スタック トレースは次のようなものであった。 `“ExecuteThread: '35' for queue: 'default'” daemon prio=5 tid=0x45ac68 nid=0x30 waiting for monitor entry [0xce901000..0xce9019d8] at weblogic.jdbc.common.internal.ConnectionEnv.destroy(ConnectionEnv.java:571) at weblogic.common.internal.ResourceAllocator` `weblogic.jdbc.common.internal.ConnectionEnv.destroy` と `weblogic.common.internal.ResourceAllocator.release` はどちらも、同期が取られたメソッドである。 `weblogic/jdbc/common/internal/ConnectionEnv.java` のコードを修正し、この問題を解決した。
CR099363	WebLogic Server 6.1 SP04 では、負荷テストにおいて、接続プールの更新間隔を 15 分に設定し、`TestConnsOnReserve` と `TestConnsOnRelease` を false に設定すると、次の例外が送出された。 `weblogic.common.ResourceException: No available connections in pool ODSConnectionPool` この問題は、プール内の接続の一部だけが使用されていると、その他の接続はすべて、更新テスト間隔が経過した時点でのテスト用に予約されてしまうために発生した。このような状態で、クライアントが接続を要求すると、例外が送出された。 次の 2 つの動作を実装するようにコードを修正し、この問題を解決した。 ■ 他に何も変更せず、まったく同じ条件で使用すると、更新テストは一度にただ 1 つの接続を予約してテストし、直ちに接続を解放する。これにより、すべての接続がロックされる問題は解決される。 ■ プールの定義に対してドライバ プロパティ `secondsToTrustAnIdlePoolConnection` を追加し、このプロパティに 1、2、30 のような正の整数を設定した場合には、プールは、その更新間隔中に DBMS に対して正常に接続されたことがわかっているプール接続については、テストを行わない。これにより、更新と `testConnsOnReserve` の処理が速くなる。

CR101093	WebLogic Server 6.1 SP04 では、接続プールのプロパティに誤ったパスワードが設定されていると、以下の例外が送出された。 `<Mar 13, 2003 10:35:45 AM IST> <Info> <JDBC> <Sleeping in createResource()> <Mar 13, 2003 10:35:46 AM IST> <Info> <JDBC Pool DemPool> <Pool DemPool is created with initial capacity 0> In the earlier versions of WLS. The exception was more descriptive. The following exception is thrown when the incorrect password is specified in sp 3. <Mar 12, 2003 9:41:38 AM IST> <Error> <JDBC> <Cannot startup connection pool "Dpool" weblogic.common.ResourceException: Could not create pool connection. The DBMS driver exception was: java.sql.SQLException: ORA-01017: invalid username/password; logon denied` `weblogic/common/internal/ResourceAllocator.java` を修正し、この問題を解決した。

jDriver

変更要求番号	説明
CR083694	`weblogic.db.oci.OciCursor` の `getTimeStamp()`、`getTime()`、`getDate()`、`getJavaDate()` のいずれかのメソッドを Calendar オブジェクトを null にして呼び出すと、新しいカレンダが作成されていた。 パフォーマンス向上のため、カレンダを 1 つだけ作成し、これらのメソッドのいずれかが初めて呼び出された時点で、カーソルに格納するようにした。
CR088387	jDriver 上で XADataSource を使用すると、ヒープ サイズが減少して `OutOfMemoryError` が発生した。 `weblogic.jdbc.oci.Connection` クラスでは、トランザクションの結果セットの LOB フィールドは、`Connection.addLob()` メソッドで接続に登録される。登録された LOB は、以下のいずれかの条件が発生すると、ネイティブ jDriver ライブラリ内の対応するオブジェクトと共に解放される。 ■ 接続レベル (この Connection は `weblogic.jdbc.oci.Connection` である) で `Connection.commit()/Connection.rollback()/Connection.close()` が呼び出される。 ■ SQL 文が実行されて、接続が `autoCommit` モードに設定される (つまり、非 TX データ ソースまたはプールからの直接接続)。 ■ OCI SQL 呼び出しからの戻りコードが、データベース レベルでコミット / ロールバックが発生したことを示している。 XADataSource が使用されているときは、上記の条件はいずれも当てはまらない。結果として、結果セット内の jDriver の LOB フィールドは、JVM ヒープ内では解放されなかった。 `closelob()` を呼び出すよう `weblogic.jdbc.oci.xa.DataSrcThreadInfo.java` のコードを修正することで、この問題を解決した。

CR090025	WebLogic Server 6.1 SP04 では、jDriver for Oracle 9.2 が AL32UTF8 文字セット (Unicode バージョン 3.1) をサポートしていない。`NLS_LANG` を `AMERICAN_AMERICA.AL32UTF8` に設定すると、`weblogic.jdbc.oci.Connection.<init>(Connection.java:246)` で次のエラーが発生する。 `java.sql.SQLException: Unsupported Oracle codeset: al32utf8. Set weblogic.codeset in your connection Properties to a valid JDK codeset which is compatible with the Oracle codeset defined in your NLS_LANG setting.` `NLS_LANG=AMERICAN_AMERICA.UTF8` のときは、「`ORA-01461: can bind a LONG value only for insert into a LONG column`」というエラーが発生する。これは、クライアントとデータベースの文字セットが一致していないことを示している。 `weblogic/db/oci/OciConnection` のコードを修正することで、この問題を解決した。
CR091151	WebLogic Server 6.1 SP03 および SP04 では、jDriver for Oracle の `ResultSet#getBigDecimal()` メソッドが正しい値を返さなかった。たとえば、9999999999.999999 という値は 9999999999.999998 に丸められていた。 コードを修正してこの問題を解決した。
CR093795	WebLogic jDriver for MSSQL Server 2000 で `statement.getString()` を使用すると、ユーロ通貨記号として「?」が取得された。 新しい String コンストラクタを使用して文字セットを正しく処理するようコードを修正し、この問題を解決した。
CR098071	WebLogic Server の旧リリースにバンドルされているバージョンの Oracle Thin driver 9.2.0 には、データ エラーが発生することのあるエラーが含まれていた。ドライバを、Oracle 製の新しいバージョンに置き換えた。
CR101517	Oracle の SQL DATE 型は 4712 B.C. から 4712 A.D. の間の日付をサポートするが、jDriver for Oracle は「1899-12-31」以前の日付を処理できなかった。jDriver によって格納された日付を、Oracle Thin Driver や SQL*Plus のような他のツールでクエリできなかった。この問題は、prepared statement を使用し、`setDate()` メソッドを使ってこのような日付を DATE カラムにバインドした場合にだけ発生した。 `weblogic/db/oci/OciColumn.java` のコードを修正し、この問題を解決した。

CR102060	WebLogic Server 6.1 SP02 では、jDriver for Oracle と Oracle 8.1.7 を使うと、`CallableStatement` を使用した後で、同じ接続に対して SQL の `UPDATE` に対する `PreparedStatement` が機能しなかった。この問題は、「CR080931」の再発であった。 `getUpdateCount()` からは 0 が返り、データベースでは何も変更されなかった。この問題は、JDBC 接続に対して `weblogic.oci.min_bind_size` プロパティを設定すると発生した。 `props.put("weblogic.oci.min_bind_size", "660")` 分析の結果、jDriver が文に対して `min_bind_size=33000` を使用していることがわかった。 `weblogic.jdbc.oci.Connection#prepareCall` は、内部的に `min_bind_size` に 33000 を設定していた。 `db\oci\OciConnection.java` のコードを修正し、この問題を解決した。

JMS

変更要求番号	説明
CR080296	WebLogic Server 6.1 SP02 では、`UserPassword` が null のときにメッセージング ブリッジが正しく機能しなかった。null ではない `UserName` と null の `UserPassword` でブリッジをコンフィグレーションすると、明確なエラー メッセージが表示されずにブリッジの起動が失敗した。 コードを修正してこの問題を解決した。

CR083933	WebLogic Server 6.1 SP03 では、大きな負荷がかかると JMS サーバが NullPointerException を送出した。 `<Aug 2, 2002 65438 PM EDT> <Warning> <JTA> <XA resource [JMS_JMSServer1JDBCStore] has not responded in the last 120 second(s).>` `<Aug 2, 2002 65438 PM EDT> <Warning> <JTA> <Resource JMS_JMSServer1JDBCStore was not assigned to any of these servers JMSServer1 >` `<Aug 2, 2002 65438 PM EDT> <Warning> <JTA> <Resource JMS_JMSServer1JDBCStore was not assigned to any of these servers JMSServer1 >` `<Aug 2, 2002 65438 PM EDT> <Warning> <JTA> <Resource JMS_JMSServer1JDBCStore was not assigned to any of these servers JMSServer1 >` `<Aug 2, 2002 65438 PM EDT> <Warning> <JTA> <Resource JMS_JMSServer1JDBCStore was not assigned to any of these servers JMSServer1 >` `<Aug 2, 2002 65438 PM EDT> <Alert> <JMS> <JMSServer "JMSServer1",unhandled exception during rollback, java.lang.NullPointerException.java.lang.NullPointerException at weblogic.jms.backend.BEDurableTopicMessageInfo.rollbackReceiveTran(BEDurableTopicMessageInfo.java352) atweblogic.jms.backend.BEXATranEntrySubscribe.startRollback(BEXATranEntrySubscribe.java145)at weblogic.jms.backend.BEXATranEntry.execute(BEXATranEntry.java127) at weblogic.kernel.ExecuteThread.execute(ExecuteThread.java139) at weblogic.kernel.ExecuteThread.run(ExecuteThread.java120)` この問題は、400 の実行スレッド、200 の JMS スレッド、および接続プールに対する 100 の接続というコンフィグレーションで発見された。例外は、負荷テストにおいて、恒久サブスクライバに対する毎秒約 10 個のメッセージ ( 約 2K) を JMS サーバに対して発行すると発生した。分散トランザクションはなかった。JDBCStore に対する接続プールには Sybase ドライバが使用されていた。 `rollbackReceiveTran()` を呼び出す前に `prepareStoreRequest` を検査するようコードを修正することで、この問題を解決した。

CR086976	WebLogic Server 6.1 SP02 では、DB2 を使用する AS400 マシンに JMS JDBS ストアを作成できなかった。 コードを修正することでこの問題を解決した。
CR089114	JMS セレクタのパフォーマンス強化が実装された。対象となるのは、それぞれがユニークな識別子を持つ 20,000 件のサブスクライバの小さなサブセットにメッセージをパブリッシュするような場合である。すべてのサブスクライバのセレクタを評価して一致するサブセットを決定する必要があるため、これは CPU に大きな負荷のかかる処理であった。 このような状況でのパフォーマンスを向上させるため、限定的なサブスクライバ インデックス機能を JMS に追加した。この拡張機能では、各サブスクライバは、サブスクライバにインデックスを付けることができることを JMS がわかるような方法で、それぞれの「ユニークさ」を JMS に通知する。このユニークさは、次の形式の JMS セレクタを使って指定する。 `xxxxx IS NOT NULL` xxxxx は JMS で既に使われているメッセージ属性とは異なる任意の識別子であり、複数のサブスクライバが同じ「xxxxx」を指定してもよい。これは、JMS の標準構文である。WebLogic JMS は、このセレクタを、サブスクライバにインデックスを付けることで最適化できることを示しているものと見なす。 インデックス付けされたサブスクライバを含むトピックに対してメッセージがパブリッシュされると、JMS は、メッセージの各ユーザ プロパティに対して反復処理を行い、ユーザ プロパティの名前と一致するインデックス付きサブスクライバにメッセージを渡す。JMS は、メッセージがパブリッシュされたときの従来の動作も引き続きサポートしており、インデックス付けされていないサブスクライバに対する反復処理を行う (上記の特殊なパターンと一致しないセレクタが指定されたサブスクライバ)。

CR089583	WebLogic Server SP02 では、恒久サブスクライバのサブスクライブを取り消すと JMSException が発生した。 `weblogic.jms.common.JMSException: Subscription clientID.vIJAY in use, uncommitted/unacknowledged messages at weblogic.jms.backend.BEConsumer.delete(BEConsumer.java:1784) at weblogic.jms.backend.BEManager.removeSubscription(BEManager.java:204) at weblogic.jms.backend.BEManager.invoke(BEManager.java:216) at weblogic.jms.dispatcher.Request.wrappedFiniteStateMachine(Request.java:510) at weblogic.jms.dispatcher.DispatcherImpl.dispatchAsync (DispatcherImpl.java:149) at weblogic.jms.dispatcher.Request.dispatchAsync(Request.java:734) at weblogic.jms.frontend.FEManager.removeSubscription(FEManager.java:288) at weblogic.jms.frontend.FEManager.invoke(FEManager.java:320) at weblogic.jms.dispatcher.Request.wrappedFiniteStateMachine(Request.java:510) at weblogic.jms.dispatcher.DispatcherImpl.dispatchAsync (DispatcherImpl.java:149) at weblogic.jms.dispatcher.DispatcherImpl.dispatchSyncFuture (DispatcherImpl.java:300) at weblogic.jms.dispatcher.DispatcherImpl_WLSkel.invoke(Unknown Source) at weblogic.rmi.internal.BasicServerRef.invoke(BasicServerRef.java:298) at weblogic.rmi.internal.BasicServerRef.handleRequest(BasicServerRef.java:267) at weblogic.rmi.internal.BasicExecuteRequest.execute (BasicExecuteRequest.java:22) at weblogic.kernel.ExecuteThread.execute(ExecuteThread.java:139) at weblogic.kernel.ExecuteThread.run(ExecuteThread.java:120)` 分析の結果、確認応答未送信メッセージのカウントをメッセージごとに 2 回インクリメントし、1 回だけデクリメントしていることがわかった。そのため、確認応答未送信メッセージ カウントは常に 0 より大きくなり、恒久サブスクライバのサブスクライブを取り消そうとすると例外が送出された。 コードを修正することでこの問題を解決した。

CR089682	WebLogic Server SP03 では、再試行の値 (下限、上限、増分) が MessagingBridge に対して正しく機能しなかった。この問題は、WebLogic Server の JMS サーバからそれ以外に対してメッセージを送信するようにコンフィグレーションされた MessagingBridge で発生した。すべて同じ WebLogic Server インスタンスで動作していた。JMS の 1 つを対象から外すと、次のエラーが発生した。
	`<30-Oct-02 15:13:34 GMT> <Error> <Connector> <Error granting connection request.>`
	再試行の試みは、接続再試行パラメータによって制御された間隔で行われなければならない。しかし、起動時のデフォルト値 (下限 = 15、上限 = 60、増分 = 5) では、次のような動作になった。
	1. 接続を試みて失敗する - OK
	2. 15 秒後に 2 回目の接続を試みる - OK
	3. それ以降は、5 秒間隔で接続を試みる - NOT OK
	`<30-Oct-02 15:13:09 GMT> <Error> <Connector> <Error granting connection request.>`
	`<30-Oct-02 15:13:14 GMT> <Error> <Connector> <Error granting connection request.>`
	`<30-Oct-02 15:13:19 GMT> <Error> <Connector> <Error granting connection request.>`
	`<30-Oct-02 15:13:24 GMT> <Error> <Connector> <Error granting connection request.>`
	`<30-Oct-02 15:13:29 GMT> <Error> <Connector> <Error granting connection request.>`
	デフォルト パラメータのいずれかを変更すると、増分は正しくなる。しかし、上限値には達しなかった。デフォルト値の場合、55 までしか達しなかった。
	再試行の値に対するロジックを修正することで、この問題を解決した。

CR091195	WebLogic Server SP04 では、MDB が受信した永続キュー メッセージに対して適切に確認応答が行われず、Administration Console のモニタ機能では「BYTES PENDING」と表示された。MDB が受信したメッセージは、WebLogic Server をシャットダウンして再起動すると解放された。 EJB のトランザクション記述子が NotSupported の時の MDB の確認応答に関するコードを修正することで、この問題を解決した。

CR091827	WebLogic Server SP02 では、OracleThinDriver for XA、Windows NT、JDK 1.3.1、および Oracle 9011 という環境で、JDBC のログ機能が「`java.sql.SQLException: Connection has already been closed`」を送出した。このエラー メッセージの後、JDBC のログ ファイルには何も記録されなかった。 JDBC ログ機能のエラーが発生しないようにコードを修正することで、この問題を解決した。
CR092464	WebLogic Server SP02 および SP04 では、恒久トピック サブスクライバが JMS サーバに接続し、トピックに対して送信されたメッセージがあると、メッセージが受信されて、受信側によってコミットされた場合でも、WebLogic Server は「現在のバイト数」の値にメッセージ サイズを加えていた。対応する「現在のメッセージ数」の値は、変更されなかった。恒久トピック サブスクライバが JMS サーバに接続していず、トピックに対して送信されたメッセージがある場合は、「現在のメッセージ数」と「現在のバイト数」が両方とも加算されていた。恒久トピック サブスクライバが再び JMS サーバに接続すると、「現在のメッセージ数」と対応する「現在のバイト数」が両方とも減算されていた。このような不整合の結果として、恒久トピック サブスクライバに対して未処理のメッセージがなくなっても、「現在のバイト数」の値が 0 にならなかった。 分析の結果、`backend/BEConsumer.java` の `addMessages()` において、`bytesCurrentCount` が不適切に更新されており、結果として Administration Console における統計情報の表示が正しくなかった。 コードを修正し、この問題を解決した。

CR093712	WebLogic Server SP04 では、メッセージング ブリッジに対する最大アイドル タイム アウトが過ぎると、以下に示すメッセージがサーバのログ ファイルに書き込まれる。多数のメッセージング ブリッジがコンフィグレーションされている場合、ログ ファイのサイズが急速に大きくなる。 `####<Dec 19, 2002 12:44:42 PM EST> <Info> <MessagingBridge> <dws-stage> <DwsStageServer> <ExecuteThread: '3' for queue: 'MessagingBridge'> <> <> <200027> <Bridge "DWSBackend Inbound Bridge" works in asynchronous mode and has not received messages for the predefined maximum idle time. The connections to the adapters will be interrupted and re-established.>` `####<Dec 19, 2002 12:44:42 PM EST> <Info> <MessaginBgridge> <dws-stage> <DwsStageServer> <ExecuteThread: '3' for queue: 'MessagingBridge'> <> <> <200020> <Bridge "DWSBackend Inbound Bridge" is stopped.> ####<Dec 19, 2002 12:44:42 PM EST> <Info> <MessagingBridge> <dws-stage> <DwsStageServer> <ExecuteThread: '3' for queue: 'MessagingBridge'> <> <> <200033> <Bridge "DWSBackend Inbound Bridge" is getting the connections to the two adapters.> ####<Dec 19, 2002 12:44:42 PM EST> <Info> <MessagingBridge> <dws-stage> <DwsStageServer> <ExecuteThread: '3' for queue: 'MessagingBridge'> <> <> <200032> <Bridge "DWSBackend Inbound Bridge" is configured not to allow degradation of its quality of service in cases where the configured quality of service is not reachable.>` `####<Dec 19, 2002 12:44:42 PM EST> <Info> <MessagingBridge> <dws-stage> <DwsStageServer> <ExecuteThread: '3' for queue: 'MessagingBridge'> <> <> <200030> <Bridge "DWSBackend Inbound Bridge" is configured to work in "Exactly-once" mode and it is actually working in "Exactly-once" mode.> ####<Dec 19, 2002 12:44:42 PM EST> <Info> <MessagingBridge> <dws-stage> <DwsStageServer> <ExecuteThread: '3' for queue: 'MessagingBridge'> <> <> <200028> <Bridge "DWSBackend Inbound Bridge" starts transferring messages.>` `DebugMessagingBridgeruntime` デバッグ フラグを使ってログ メッセージの書き込みを禁止できるようにすることで、この問題を解決した。
CR099455	競合状態が原因で、JMS メッセージが失われていた。 コードを修正し、この問題を解決した。

CR101374	JMS メッセージが失われていた。JMS サーバには着信キューと発信キューがある。別のサーバにある MDB は、着信キューからメッセージを取り出し、1 対 1 の関係で発信キューにメッセージを格納する。この処理は、MDB が開始するトランザクションの中で行われる。トランザクションの終わりは、MDB が発信キューにメッセージをエンキューしたときであった。JMS サーバを停止して再起動すると、メッセージが失われた。10,000 メッセージの負荷に対して、失われるメッセージは少数であった。 トランザクション処理される `XAConnection` を強制的に `CLIENT_ACK` にするようコードを修正し、この問題を解決した。
CR101670	`weblogic.jms.client.JMSSession$UnackedMessage` テストの際に、コミットまたは確認応答を行わない JMS クライアント セッションが、メモリをリークしていた。テストでは、キューとトピックの両方に対して非同期レシーバとして機能する起動クラスを使用していた。レシーバは、トランザクション処理を行うセッションと `AUTO-ACKNOWLEDGE` モードを使って作成された。メッセージを受信する、レシーバはセッションに対して `Rollback` を行った。診断の結果、`weblogic.jms.client.JMSSession$UnackedMessage` でメモリリークが発生していることがわかった。 `weblogic/jms/client/JMSSession.java` のコードを修正し、この問題を解決した。

JNDI

変更要求番号	説明

CR087237	WebLogic Server 6.1 SP03 では、Solaris で稼働する 2 ノードのクラスタにおいて、ステートレス セッション Bean のフェイルオーバが 5 分程度かかっていた。サーバはフリーズし、クライアントはハングした。 複数のコードとコンフィグレーションを変更することで、フェイルオーバを 45 秒に短縮した。 ■ `RJVMFinder` を `PeerGoneListener` とすることにより、利用できないサーバを憶えておき、レプリカ リストが空になるまではサーバを試さないようにした。 ■ `BasicReplicaHandler` を修正し、レプリカ リストにまだサーバがあるときには、フェイルオーバの間に `refreshReplicalist` を呼び出さないようにした。 ■ `T3JVMConnection` に、ワーカ スレッドにソケットを作成するための新しい機能を追加した。さらに、`so_timeout` に対して同じタイムアウトを使うようにし、Solaris 8 クライアントにおいて読み取りが 1 分間ブロックしないようにした。新しいフラグは `-Dweblogic.client.SocketConnectTimeoutInSecs=15` である。 ■ サーバとクライアントでのハートビート間隔を、既存の `-D` の値を使って制御するようにした。サーバにおけるデフォルトの実行キューを増やし、増大したハートビートの負荷に対応するようにした。クライアントのハートビート時間は、`-Dweblogic.PeriodLength=10000` フラグによって制御される。サーバのハートビートは、`config.xml` の `ServerMBean` の `PeriodLength="10000"` フラグによって制御される。
CR093700	JNDI ツリーに対するクラスタ対応 RMI オブジェクトのバインド / リバインドおよびアンバインドの際に、アンバインド操作が、オブジェクトを null にしてガベージ コレクション用にセットアップしたにもかかわらず、オブジェクトがサーバ上でローカルにホストされているときは、null オブジェクトに対する参照を保持したままになることがあった。 クラスタ内に他に利用できるこのようなオブジェクトのレプリカがないときにだけガベージ コレクション用にオブジェクトを設定するよう、コードを修正した。

CR093799	複合名に対して Context.lookup を行うとき、名前の全体または一部が解決できないと、NameNotFoundException が発生した。 名前 remainingName = nnfe.getRemainingName(); は、名前から解決されないコンポーネントを取得しなければならない。このようなコンポーネントは、JNDI ツリー内に作成する必要のあるサブコンテキストである。 そのため、列挙 e = remainingName.getAll() を呼び出したなら、取得される列挙には未解決のコンポーネントが文字列として含まれていなければならない (「Unresolved1」、「Unresolved2」、「Unresolved3」など)。しかし、列挙にはただ 1 つの文字列「Unresolved1.Unresolved2.Unresolved3」が含まれているだけであった。 区切り文字「/」を使って CompositeName を作成するよう BasicNamingNode.java を変更することで、この問題を解決した。
CR096838	WebLogic Server 6.1 SP03 および SP04 では、weblogic.jndi.Environment オブジェクトが適切なタイミングで解放されず、InitialContext で ejb-ref-name をルックアップする際に OutOfMemoryError エラーが発生していた。 コードを修正し、コンテキストを正しく閉じるようにした。

JSP

変更要求番号	説明

CR092039	WebLogic Server 6.1 SP03 では、次のように JSP タグの charset の値に余分な引用符があると、JspParser が UnsupportedEncodingException を送出した。
	`<%@ page contentType="text/html; charset=\"Shift_JIS\"" %>`
	その結果、次のようなエラーが発生した。
	`java.lang.RuntimeException: Unknown/unsupported charset: "Shift_JIS" - java.io.UnsupportedEncodingException: Charset: '"Shift_JIS"' not recognized, and there is no alias for it in the WebServerMBean`
	WebLogic Server は HTTP 仕様を正しくサポートしていなかった。
	Content-Type ヘッダーの charset 属性値に対する引用符の付いた文字列やコメントのサポートを追加することで、この問題を解決した。

CR088019	WebLogic Server 6.1 SP03 では、仮想ホストを使用する HP-UX 11 の場合、生成される Java ファイルをコンパイルできなかった。仮想ホストが対象でないときは、.ear ファイルは正しくデプロイされた。アクセスすると、login.jsp が Java ファイルを生成してコンパイルした。 仮想ホストをコンフィグレーションし、.ear の Web アプリケーション部分の対象を仮想ホストにした後、サーバを再起動すると、アクセスしたときに、login.jsp は Java ファイルを生成するが、クラス ファイルにコンパイルすることができなかった。 コードを修正してこの問題を解決した。WebServer コンポーネントを識別する要素を一時ディレクトリに格納し、仮想ホストとデフォルトの Web サーバを区別するようにした。これにより、Web アプリケーションがデプロイされて一時ディレクトリが空になっても、デプロイメント ファイルがクラッシュしなくなった。

CR092089	名前にマルチバイト文字を含む JSP ファイルがコンパイルされず、ディレクトリがその内容を返していた。 分析の結果、ファイル名にマルチバイト文字が含まれると、JSP ファイルのパスが正しく検出されなかった。WebLogic Server が VM のデフォルトでパスをデコードした後、パスは「input-charset」を使って再びデコードされていた。 ブラウザのデコード方法の実装の問題が関係していた。Internet Explorer は、UTF-8 に基づいて `URLencode` を行っていた。Netscape は、プラットフォームのデフォルトに基づいて行っていた。 UTF-8 に基づいて `ContextPath`、`ServletPath`、および `PathInfo` をデコードするようにコードを修正することで、この問題を解決した。 デコード方式は、`weblogic.http.URIDecodeEncoding` でコンフィグレーションできる。
CR092366	`javax.servlet.jsp.PageContext.out` 属性が、本体コンテンツ タグに対して正しくクリーンアップされていなかった。 コードを修正してこの問題を解決した。

CR093014	あるスクリプトレットで開始した Java コメントが次のスクリプトレットで終了すると、エラーが発生していた。
	`<22.07.2002 14:12:24 CEST> <Error> <HTTP>`
	`<[WebAppServletContext(7422744,DefaultWebApp,/DefaultWebApp)] Servlet failed with Exception weblogic.servlet.jsp.JspException: (line 12): scriptlet close brace '}' unbalanced at line 12 which breaks scope '_base_service_scope_'  at ....`
	WebLogic Server 7.0 から文法を逆移植することで問題を解決した。ScriptletScopeLexer が Java コメント全体をスキップするようになった。

CR093291	サーバによる JSP のコンパイルに、システム クラスパスが含まれていなかった。サーバ起動時に、CLASSPATH の jar を参照する JSP がコンパイルされなかった。 次のようなメッセージが発生した。 `package com.bea.p13n.util.debug does not exist probably occurred due to an error in /devtools/debug.jsp line 1:<%@ page import="com.bea.p13n.util.debug.Debug" %>` 分析の結果、システム クラスパスの jar が「../..」で始まっていると問題が発生することがわかった。 クラスパスを整理するときに ./ と ../ の場合も考慮するようコードを修正することで、この問題を解決した。

CR093625	WebLogic Server 6.1 SP03 および SP04 では、JSP ページにおける JSP include に対する出力をサーバが誤って示していた。include、forward、wrapped-response、および JSP BodyTag の処理を統一するようサーブレット エンジンを変更することで、この問題を解決した。
CR094190	JVM の仕様によると、メソッドのサイズに対する制限は 64K である。WebLogic Server 6.1 SP03 では、生成される Java コードが `__jspService` メソッドに対する 64K の制限を超えるような本体付きタグを多く含む JSP があった。 コードを修正し、この問題を解決した。`<my:tag ... />` の形式の空の BodyTag があると、普通であれば本体に対してスコープを設定するように生成されるコードが、`StandardTagLib.java` の静的メソッドに対する単一の呼び出しに置き換えられる。これにより、BodyTag は、通常どおり JSP ソースから呼び出されているものと考える。
CR098402	WebLogic Server 6.1 SP04 では、`jsp:setProperty` が、プロパティ値に含まれる要求時の式を、String に変換していた。コードを修正し、この問題を解決した。要求時属性として指定される値から割り当てるときは、タイプ変換が適用されなくなった。

JTA

変更要求番号	説明
CR088171	WebLogic Server 6.1 SP02 では、トランザクションが unknown 状態であると、トランザクションが失われた。 次のようなクラスタ コンフィグレーションについて考える。 MS-1 ■ Queue1 - MDB1 TX Required ■ Queue2 - MDB2 TX Required ■ SLSB MS-2 ■ Queue3 - MDB3 TX required ■ SLSB - `send()` - TX required ■ `sendImmediate()` - TX RequiresNew キューは DB に保持され、MDB はトランザクションとしてメッセージをキューから取り出す。以下の操作を行う。 1. MDB3 の `SLSB.send()` を呼び出した後、`SLSB.sendImmediate()` を呼び出す。 2. SLSB.send は、`SLSB.send()` ...>MDB2-> MDB1 -> `SLSB.sendImmediate()` を 2 回呼び出す。 3. Queue3 に 100 メッセージを送信し、処理の開始を許可する。すべてのメッセージがエンキューされた後、Queue3 からすべてのメッセージが取り出される前に、MS-1 を停止する。 4. MS-1 を再起動する。 MS-2 を停止すると、MS-1 は peergone 例外を受け取り、ステータスをコミットする準備ができているトランザクションは、prepared から unknown になる。2 番目のサーバを再起動すると、トランザクションが UNKNOWN 状態であるために、コミットとロールバックのどちらを行えばよいのかわからないので、トランザクションは失われる。 コミット中に unknown 例外を受け取ったときはトランザクションをロールバックするようコードを修正することで、この問題を解決した。

CR088844	WebLogic Server 6.1 SP02 では、サーバの起動時に LogManager と JTA の間でデッドロックが発生した。スレッド ダンプを分析した結果、次のことがわかった。 実行スレッド 7: このスレッドはリソースを取得しようとする。`ResourceDescriptor.startResourceUse` を呼び出す。このメソッドの同期ブロックの内部は、トレース文である。 実行スレッド 6: Audit Bean が `LogBroadcaster` を呼び出そうとする。これによって行われる RMI 呼び出しにより、トランザクションはサスペンドしてリソースを解放する。解放では `ResourceDescriptor.startResourceUse` の呼び出しも行われる。 スレッド 7 の `startResourceUse` 内のトレース文はスレッド 6 での `LogBroadcast` を待ってブロックしており、スレッド 7 の取得はスレッド 6 での解放を待っている。 `startResourceUse` のトレース文を同期ブロックの外に移動することで、この問題を解決した。
CR091628	`javax.transaction.xa.XAResource` インタフェースで定義されている `setTransactionTimeout()` メソッドを使うと、トランザクション マネージャは、`XAResource` インスタンスに関する操作のために、リソース マネージャにトランザクション タイムアウトを設定できる。アプリケーションがグローバル トランザクションでリソース マネージャにアクセスするとき、WebLogic Server のトランザクション マネージャはこのメソッドを呼び出していなかった。 `XASetTransactionTimeout` が「true」のときに使用される `XATransactionTimeout` プロパティを追加することで、この問題を解決した。これは、JTA トランザクションのタイムアウト値が使用されていてその値が大きい場合、トランザクションの途中で WebLogic Server がクラッシュすると、Oracle が行をロックしてしまい、Oracle が再起動するまで行にアクセスできなくなるためである。
CR091882	WebLogic Server 6.1 SP03 では、6.1 SP03 のリモート クラスタでステートレス EJB を呼び出すと、トランザクション タイムアウトが発生した。 `transaction/internal/TransactionImpl.java` のコードを修正することで、この問題を解決した。

CR092301	WebLogic Server 6.1 SP04 では、`weblogic.transaction.internal.ServerResourceInfo.isAccessibleAtAndAssignableTo` が Null ポインタ例外を送出した。この問題は、サーバがアイドル状態の時にも散発的に発生し、確実に再現させることはできなかった。 `weblogic/transaction/internal/ServerResourceInfo.java` のコードを修正することで、この問題を解決した。
CR093045	WebLogic Server 6.1 SP02 以降では、WebLogic Server に負荷がかかっている状態でデータベースを停止して再起動した後、Oracle 8.1.7.4 Thin Driver XA 接続プールが使用できなくなった。 テスト アプリケーションは、XA 接続プールを使用し、MDB を通して、Oracle 8.1.7.4 のカラムを更新していた。WebLogic Server が MDB メッセージを処理しているときに、Oracle データベースを停止して再起動すると、XA 接続が更新されず、Oracle の分散ロックが回復されなかった (ORA-01591)。WebLogic Server は周期的な「ハング」状態になり、XA 接続を更新しようとした。XA 接続の更新と使用を「循環している」とき、WebLogic Server のログ ファイルには次の情報が記録された。 `... ####<13.12.2002 12:09:54 CET> <Info> <JDBC Pool FoundationThinXAPool> <A4PMA9ALX> <myserver> <ExecuteThread: '9' for queue: 'default'> <guest> <484:6c709abb7f28edb9> <000000> <javax.transaction.SystemException: start() failed on resource 'weblogic.jdbc.jta.VendorXAResource' null at weblogic.transaction.internal.ServerResourceInfo.xaStart(ServerResourceInfo.java:1010) at ...` WebLogic Server は、この状態 ( サイクル ) から回復しなかった。サーバのすべてのスレッドはブロックされているように見え、サーバを停止しなければならなかった (`weblogic.Admin SHUTDOWN` は機能しなかった )。WebLogic Server を再起動した場合にだけ、JTA の回復が行われた直後に、Oracle の分散ロックが解決された。回復が行われる前は、MDB が未処理のメッセージを再送信しようとして、ORA-01591 例外が大量に送出された。状態不明の分散トランザクションを JTA が回復した後は、未処理の JMS メッセージの処理は正常に行われた。 `weblogic\jdbc\oci\xa\XADataSource.java` のコードを修正し、この問題を解決した。

CR100357	トランザクションに参加するサーバを停止して再起動すると、回復が遅かったり、デッドロックが発生したりした。コンフィグレーションには、ピン固定された接続ファクトリおよび分散された着信キューと発信キューを備えた 4 つの JMS Server と、4 つの MDB Server (それぞれに 4 つの MDB がデプロイされ、プール内の 5 つの Bean が着信キューの各分散メンバをリスンしている) が含まれ、着信キューにはサイズが 2K のメッセージが 40,000 件あった。 JMS サーバと MDB サーバを再起動すると、JMS サーバがすべてのデフォルト キュー スレッドを使ってしまい、実行キューが増え続けた。 `ServerTransactionImpl.java` を修正し、この問題を解決した。
CR100898	トランザクションがロールバックした場合、トランザクションに参加するリソースを持たないサーバがトランザクションに参加していると、トランザクションが完了しない場合があった。トランザクションは調整を行っているサーバ上に存在し続け、ロールバックするリソースがないことをサーバに通知する再試行を続けていた。最終的に、トランザクションは破棄された。すべての参加リソースはロールバックの通知を受けたが、完了した後で、調整を行っているサーバ上でコールバックが処理されない場合があった。 リソースを持たないサーバにも対応し、すべてのサーバ / リソースが通知を受けたら直ちに状態をロールバック済みに設定するよう、ロールバック非同期応答処理のロジックを変更することで、この問題を解決した。

その他

変更要求番号	説明

CR094764	WebLogic Server 6.1 SP04 では、Windows NT を再起動したとき、beasvc.exe が正常にシャットダウンしなかった。この問題は、以下の手順で発生した。
	1. NT サービスとして動作するように WebLogic Server をコンフィグレーションする。
	2. Windows のコントロール パネルから WebLogic Server を開始する。
	3. Windows の [スタート] メニューから、Windows NT を再起動する。
	サーバ ログは空で、イベント ビューアには次のメッセージが記録されていた。
	`The myserver service terminated unexpectedly.  It has done this 1 time(s).`
	`The following corrective action will be taken in 0 milliseconds: No action.`
	分析の結果、Windows Service Control Manager (SCM) は、システム シャットダウンの際に beasvc.exe に SERVICE_CONTROL_SHUTDOWN を通知できるよう、SERVICE_ACCEPT_SHUTDOWN の通知を受けなければならないことがわかった。
	必要な処理を行うようコードを修正し、問題を解決した。

CR099590	WebLogic Server 6.1 SP02 では、JMX クライアントを使用して管理対象サーバを削除すると、`weblogic.management.internal.MBeanHomeImpl.getInterface(MBeanHomeImpl.java:431)` で Null ポインタ例外が発生した。 MBean の削除に関する修正により、この問題を解決した。

プラグイン

問題が解決されたプラグインは、「説明」カラムにおいてかっこで囲んで示してあります。

変更要求番号	説明
CR091740	(HttpClusterServlet) ベンチマーク テストの際に、HttpClusterServlet を実行するプロキシ サーバが 503 番と 500 番の HTTP エラーを返した。このエラーは、ときどき不規則に発生した。コンフィグレーションは、JRockit 7.0 SP2 Load 2 と Sun JDK 1.3.1_06 であった。`AcceptBacklog` とスレッド カウントは、それぞれ 1000 および 50 まで増加した。ベンチマークは、100 ～ 200 のクライアント スレッドで実行された。 この問題は、サーブレットの動的サーバ リストにエントリが含まれない場合に発生した。 動的サーバ リストの配列を作成する前にリストのサイズをチェックするようコードを修正することで、この問題を解決した。

CR084625	(NSAPI) iPlanet プラグインのすべてのバージョンで、/_wl_proxy/_post がハードコーディングされていた。
	`#else /* UNIX */`
	`sprintf(name,"/tmp/_wl_proxy/_post__%d_%d", pid, postCounter);`
	`fsep = '/';`
	`#endif`
	このため、複数のサーバが異なるユーザ ID とグループを使用している場合に問題が発生した。
	たとえば、クライアントからアップロードされたファイルは、WebLogic Server に対してプロキシを行う Web サーバの /tmp/_wl_proxy に一時的に保存される。_wl_proxy ディレクトリは、Web サーバが再起動されるたびに削除される。ユーザ プロセスはデフォルトの umask を使ってパーミッションを作成するので、環境が共有されている場合 (同じ物理サーバに複数のインスタンスがある場合) には、このディレクトリを作成したユーザ プロセスが、他のユーザの書き込みをブロックする。つまり、アプリケーション A (A というユーザ ID で動作) が /tmp/_wl_proxy/< 何らかの一時ファイル > を作成した後、アプリケーション B がファイルをアップロードしようとすると、エラー ファイルに次のエラーが記録される。
	`[27/Aug/2002082140] failure (14367) for host 172.18.5.122 trying to POST /filetransfer/TransferUtility.do, wl-proxy reports exception occurred 'READ_ERROR [os error=13, line 501 of proxy.cpp] Cannot open TEMP file '/tmp/_wl_proxy/_post__14367_5' in $TMP/_wl_proxy for POST of 2867 bytes`
	コンフィグレーション可能な引数 WLTempDir を追加することで、この問題を解決した。この引数は、wlproxy.log と _wl_proxy の両方に対して適用される。下位互換性のため、ログ ファイル作成時に WLLogFile は WLTempDir をオーバライドする。

CR085192	(NSAPI、ISAPI) 接続プールをオフにすると、パフォーマンスが低下した。 URL を作成するために多くのリソースが必要であった。 プラグインが接続を閉じる前に WebLogic が接続を閉じたため、接続が「半分開いた状態」になり、プラグインで PROTOCOL_EXCEPTIONS が発生した。 複数のコードを変更することで、この問題を解決した。 ■ ISAPI と NSAPI に対して Keep-Alive をデフォルトで有効にした。 ■ WebLogic Server がプラグインに対して特殊なヘッダーで keepAliveSecs の値を送信するようにした。

CR085285	(NSAPI) プラグインからの InitJVMID リクエストに対して Web アプリケーション コンテナが 404 番のメッセージを返し、ログ ファイルが次のメッセージで一杯になっていた。 ####<Sep 6, 2002 4:56:43 PM EDT> <Debug> <HTTP> <petunia> <ms1petunia> <ExecuteThread: '13' for queue: 'default'> <kernel identity> <> <101147> <HttpServer(887891,null default ctx,ms1petunia) found no context for "/WLDummyInitJVMIDs". WebLogic Server 上に常にデプロイされている内部 Web アプリケーションに対して InitJVMID リクエストを送信することで、この問題を解決した。
CR085541	(すべてのプラグイン) __WebLogicBridgeConfig が、デバッグの内部表現 (紛らわしい 10 文字のコード) を返した。 コードによって表される値を返すようコードを修正した。
CR085695	(ISAPI) KeepAlive が有効になっているときに、HEAD HTTP リクエストについて接続を閉じるまでの時間が約 30 秒であった。 この問題は、6.1 SP01、Windows 2000、IIS 5.0 という環境で発生した。 HEAD リクエストに対して接続を直ちに閉じるようコードを修正することで、この問題を解決した。
CR085921	(ISAPI) すべてのプラグインにおいて、WebLogic Server へのリクエスト / ポストデータを書き込む際に発生した WRITE_ERRORS と、__WebLogicBridgeConfig 統計でブラウザに応答ヘッダ / データを書き込む際に発生した WRITE_ERRORS を区別することができなかった。この問題により、プラグイン / クラスタの状態をモニタするために __WebLogicBridgeConfig によって表示される統計情報の使用がいっそう困難になっていた。 2 つの新しいエラー タイプ WRITE_ERROR_TO_CLIENT と WRITE_ERROR_TO_SERVER を実装することで、この問題を解決した。
CR085923	(ISAPI) すべてのプラグインで、Debug=ERR と設定すると、INF メッセージもログに記録されていた。 コードを修正してこの問題を解決した。Debug=ERR と設定したときは、ERR メッセージだけがログに記録されるようになった。
CR086224	(NSAPI) プラグインが、POST データから SESSIONID を読み取らなかった。 分析の結果、getPreferredServersFromCookie() に問題があることがわかった。session.getId() が POST データとして保持されているとき、getId() に余分な文字列「|time」が含まれていた。 コードを修正してこの問題を解決した。

CR086490	(ISAPI) すべてのプラグインと `HttpClusterServlet` について、`MaxSkips` 機能が無効になっていた。これは、この機能のデフォルト値 10 では、この機能が意図したほど有効でないことが判明したためであった。この機能がないため、クラスタ内の 1 つ以上のサーバが応答できないときに、パフォーマンスが低下する可能性があった。 新しいパラメータ `MaxSkipTime` を設けることで、この問題を解決した。 `MaxSkipTime` を使用すれば、サーバを不良状態にしてから新しいサーバ リストを強制するまでの待機時間を秒単位で指定できる。

CR086518	(NSAPI) クエリ文字列とアンカーの両方を含む URL を使ってリダイレクトを行う場合、アンカーが、最後のクエリ文字列メンバの値部分の一部と誤って見なされていた。この問題は、3 層環境で Internet Explorer を使用し、HTTP サーバに対して iPlanet 6.0 を使用した場合にだけ発生した。Netscape や Mozilla を使用した場合、または 2 層環境でサーバを実行した場合は、問題は発生しなかった。
	複製するためには、foo.jsp を呼び出す。
	`foo.jsp:`
	`<%`
	`String redirectURL = "http://hostname/T/foo2.jsp?a=b#foo";;`
	`response.sendRedirect(redirectURL);`
	`%>`
	`foo2.jsp:`
	`<%= request.getParameter("a") %>`
	Administration Console では、「b」ではなく「a」に対する値が「b#foo」として、表示された。
	クエリ文字列からアンカーを除去するようにコードを修正することで、この問題を解決した。Internet Explorer では、sendRedirected URL に対してこの処理が行われない。

CR088914	(NSAPI) 6-181 ページの「CR088915」を参照。

CR088915
CR088914

(ISAPI、NSAPI) SSL 証明書チェーン攻撃への弱さを緩和するための変更を行った。CA 証明書に対する `BasicConstraints` をチェックするコードを追加した。制約の検査は、プラグインのパラメータ `EnforceBasicConstraints` で制御される。パラメータの設定は以下のとおりである。

- OFF — 本当に他の選択肢がないのでない限り、強制を完全に無効にすることは推奨されない。たとえば、顧客が商用 CA から証明書を購入しても、チェーンは新しい検査を渡さない。現在の商用 CA 証明書のほとんどは、デフォルトの STRONG 設定の下で動作しなければならないことに注意すること。
- STRONG — V3 CA 証明書に対する `BasicConstraints` が検査されて、証明書が CA 証明書であることが検証される。デフォルトの設定である。
- STRICT — STRONG レベルと同じ検査を行うが、さらに、IETF RFC 2459 も厳密に強制する。この RFC では、CA 証明書に対する `BasicConstraints` が「critical」としてマークされている必要もあると指定されている。RFC 2459 に厳密に従う場合は、このレベルに設定する。

この変更により、WebLogic Server 6.1 SP04 では発生しなかったセキュリティ例外が発生する可能性がある。たとえば、発信 SSL 呼び出しを行おうとした場合、リモート サーバ インスタンスによって提示される証明書に基本制約がないと、次のエラーが発生する。

```
<May 8, 2003 4:36:18 PM MDT> <Notice> <WebLogicServer>
<SSLListenThread listening on port 7002> <May 8, 2003 4:36:19 PM
MDT> <Notice> <Management> <Starting discovery of Managed
Server... This feature is on by default, you may turn this off
by passing -Dweblogic. <May 8, 2003 4:36:19 PM MDT> <Notice>
<WebLogicServer> <Started WebLogic Admin Server "myserver" for
domain "412253" running in Production Mode> EXCEPTION: Socket
Closed java.io.IOException: Socket Closed at
weblogic.security.SSL.SSLSocket.sendAlert(SSLSocket.java:1086)
at...
```

このようなエラーは、プラグイン パラメータ `EnforceBasicConstraints` を false に設定することで避けることができる。コマンドラインで基本制約のチェックを無効にするには、次のコマンドを使用する。

```
-Dweblogic.security.SSL.enforceConstraints=false
```

CR089746	(NSAPI) WebLogic Server は次のようなシナリオをサポートしていた。 ■ T3 を使用してアプリケーション サーバに接続するプラグイン (SSL セッション キャッシングなし、キープアライブなし)。 ■ HTTPS を使用してアプリケーション サーバに接続するプラグイン (SSL セッション キャッシングなし、キープアライブなし)。ユーザは、プラグインと Weblogic Server の間の SSL セッションの再開を望んでいる。 ユーザは、T3 と HTTPS の両方について、プラグインと WebLogic Server の間の SSL セッション キャッシングとキープアライブのサポートを要求していた。 `KeepAliveEnabled` フラグと SSL セッション キャッシュを追加することで、この要求に応えた。
CR091635	(NSAPI) SP03 バージョンのプラグインは、次のようなリクエストの JVMID を解析できなかった。 `2002 Found Encoded Session=[JSESSIONID=9dq5Z081!805492834!174332983!7001!7002?_JASPER=1000/r/1037920979629&PAGE=HOME_TEMPLATE&INTEGRATION=YES] Thu Nov 21 15:16:09 2002 Parsing cookie JSESSIONID=9dq5Z081!805492834!174332983!7001!7002?_JASPER=1000/r/1037920979629&PAGE=HOME_TEMPLATE&INTEGRATION=YES Thu Nov 21 15:16:09 2002 getpreferredServersFromCookie: 805492834!174332983!7001!7002?_JASPER=1000/r/1037920979629&PAGE=HOME_TEMPLATE&INTEGRATION=YES Thu Nov 21 15:16:15 2002 parseJVMID: could not resolve hostname 'r', treating it as invalid` `sessionid` 全体ではなく「/」含むセッション クッキーの JVMID 部分をチェックするようコードを修正することで、この問題を解決した。
CR092756	(ISAPI) バックエンドから HTTP 503 の応答を受け取っても、ISAPI プラグインがフェイルオーバしていなかった。コードを修正し、この問題を解決した。
CR093196	(NSAPI) Web アプリケーションがクラスタにデプロイされていて、iPlanet プロキシ プラグインを通して JSP にアクセスすると、JSP の `sendRedirect()` が失敗した。 IE ブラウザでは「`The page cannot be displayed...`」というメッセージが返り、Netscape では「`The document contained no data...`」というメッセージが返った。 セカンダリ セッションの Null 値をチェックするようにコードを修正することで、この問題を解決した。

CR093530	(ISAPI) PathTrim の値で大文字と小文字が区別されていたため、大文字と小文字が区別されない URL を使ってクライアントがアプリケーションにアクセスできることが望ましい環境で問題が発生していた。 PathTrim の値について大文字と小文字を区別しないように変更することで、この問題を解決した。
CR093662	(NSAPI) 次のすべてに当てはまると、encodeRedirectURL() の呼び出しが失敗した。 (a) Web アプリケーションがクラスタに .war ファイルとしてデプロイされている。 (b) Web アプリケーションの JSP に、iPlanet プロキシ プラグインを通してアクセスする。 (c) JSP が、HTTP GET メソッドを使用する FORM で encodeRedirectURL() を呼び出す。 iPlanet Web サーバでコア ダンプが発生した。Internet Explorer ブラウザは「The page cannot be displayed...Cannot find server or DNS Error」を返し、Netscape は「The document contained no data... 1」を返した。 分析の結果、問題の原因は、クエリ文字列を切り捨てないで固定長バッファにプライマリまたはセカンダリの jvmid を格納していたことであった。 セッション クッキーを解析する前にクエリ文字列を切り捨てるようにコードを変更することで、この問題を解決した。
CR094109	(Apache) 応答時間が HungServerRecoverSecs で設定されている時間制限を超え、Indempotent プリミティブが オンになっていると、プライマリ サーバからのフェイルオーバが発生しなかった。 分析の結果、ap_proxy.cpp の例外処理ソース コードにおけるエラーに関する問題であることがわかった。 フェイルオーバのコードを書き直して、この問題を解決した。
CR094282	(NSAPI) WebLogicHost と WebLogicPort、または WebLogicCluster にリストされているサーバが稼働していない場合、またはたとえば Solaris がシングル ユーザ モードで起動している場合、オペレーティング システムが接続をタイムアウトするまでプラグインがハングしていた。 デフォルトでは、このタイムアウトに長い時間がかかった。 新しいプラグイン パラメータ WLSocketTimeoutSecs を追加した。このパラメータを使用すると、マシンがダウンしていても長い遅延を避けることができる。デフォルト値は 2 秒である。この値は 0 より大きくなければならない。
CR094663	6-184 ページの「CR095166」と同じ問題および解決。

CR094768	(ISAPI) クッキーに対するポスト データの解析が、プラグインで正しく実装されていなかった。次のエラーが発生した。 `"Bad request in the post data [NameField=aaaaa&ValueField=111111111111&AddValue=+Add+]"` `UrlUnescape()` メソッドが、正しくない戻り値の型を返し、`application/x-www-form-urlencoded` 以外の content-type に対するポスト データを解析していた。 コードを修正し、この問題を解決した。
CR095009	(ISAPI) プラグインが、適切なサーバ インスタンスの解決に成功していなかった。ログ ファイルに、「`Primary JVMID not found in the server list, ignore preferred servers`」というエラーが記録された。 分析の結果、`getPreferredServersFromCookie()` にエラーがあることがわかった。コードを修正し、この問題を解決した。
CR095558	(ISAPI) プラグインが、ブラウザからの読み取りの障害と、サーバに対する書き込みの障害を、正しく区別していなかった。クライアントとプラグインの間の障害の時にだけ接続アボート メッセージ (10053/10054) を処理するよう、コードを修正した。
CR095559	(Apache) プライマリ サーバがフェイルオーバした後、プラグインが、セカンダリ サーバのインスタンスを探して、プライマリ サーバで障害が発生したときに行っていたリクエストをセカンダリ サーバに送信するのではなく、引き続きプライマリ サーバのインスタンスを探していた。 フェイルオーバのコードを書き直して、この問題を解決した。
CR095166	(ISAPI) クライアントが複数のクッキーを送信すると IIS プラグインがクッキーを誤って解釈し、プラグインとプライマリ サーバ インスタンスの間の結び付きが確立されなかった。 分析の結果、文字列検査のロジックに誤りがあり、`strstr()` がセッション クッキー名を正しく解析していないことがわかった。
CR096625	(Apache) `X_WEBLOGIC_FORCE_JVMID` ヘッダーが、クラスタのサーバ インスタンスに送信されていた。 `X_WEBLOGIC_FORCE_JVMID` は、サーバ リストがクラスタではなく、現在のサーバがまだ jvmid を持っていないときにだけ、送信されなければならない。 コードを修正し、この問題を解決した。

CR097202	(すべてのプラグイン) WebLogic Server とブラウザの間で 128 ビットのステップアップ証明書を使用するプラグインを含むコンフィグレーションでは、WebLogic 側でブラウザの暗号強度を定義する必要がある。ブラウザには、40 ビットと 128 ビットの強度があった。ブラウザの暗号強度を取得するには、`(Integer)httpRequest.getAttribute("javax.servlet.request.key-size")` というコードが使用されていた。 ブラウザは 128 ビットより小さい暗号サイズを使うように設定されていたが、返される値は常に 128 であった。`https_keysize` によって、作成される接続の強度が決まる。プラグインでは 128 ビットの証明書が使用されていたため、ブラウザで使用されている強度にかかわらず 128 が返された。WebLogic Server に対してリクエストが送られるときは、ブラウザの強度に基づいて、40 または 128 が返された。 プラグインは、`HTTPS_KeySIZE` ヘッダーまたは `HTTPS_SECRETKEYSIZE` ヘッダーを変更しない。 このニーズを満たすため、`WL-Proxy-Client-Keysize` と `WL-Proxy-Client-Secretkeysize` という 2 つのヘッダーを新しく実装した。どちらのヘッダーの値も、`request.getHeader()` で取得できる。
CR096850	(NSAPI) 負荷テストアプリケーションがサーバに対して複数の接続を開こうとすると、Windows 2000 の下で稼働する iPlanet サーバがクラッシュした。iPlanet サーバは、サーバに対する接続を何回か試みた後でクラッシュした。 分析の結果、iPlanet サーバはアクセス違反のためにクラッシュし、`MSVCRT!CxxThrowException` を示していた。 例外を送出するのではなく例外を返して処理するよう `sendResponse` を修正することで、この問題を解決した。
CR100361	次の新しい例外タイプが追加された。 `READ_ERROR_FROM_CLIENT` — クライアントからのデータ読み取り時のエラー `READ_ERROR_FROM_SERVER` — バックエンド サーバからのデータ読み取り時のエラー `READ_ERROR_FROM_FILE` — POST データが格納されている一時ファイルからの読み取り時のエラー `WRITE_ERROR_TO_FILE` — 一時ファイルへの POST データ書き込み時のエラー

CR101222	(NSAPI および Apache) プライマリ サーバ インスタンスがハングしたときのセカンダリ サーバ インスタンスへのフェイルオーバが、確実に行われなかった。プライマリ セッションとセカンダリ セッションを確立した後、`request.getParameter("primary")` がサーバ名と同じ場合、クライアントは、`HungServerRecoverySecs` の値より長くスリープしているリカバリ リクエストを送信していた。Web サーバは、プライマリ サーバがハングし、リクエストをセカンダリ サーバにフェイルオーバしていることを検出しなかった。 分析の結果、セカンダリ サーバが異常 (bad) とマークされていたため、プラグインがセカンダリ サーバをスキップし、一般サーバ リストを使用していたことがわかった。 `MaxSkipTime` の後でサーバを正常 (good) にリセットするようコードを修正し、この問題を解決した。
CR101428	(ISAPI) リクエストに複数のクッキーが含まれていて、名前が JSESSIONID ではないクッキーが JSESSIONID クッキーの後にあると、そのクッキーは取り除かれて、WebLogic Server に送付されなかった。 コードを修正し、この問題を解決した。
CR101596	(NSAPI) WebLogic Server 6.1 SP04 では、特定のエラー状態に対し、`ssl_certchain_verify_callback()` が誤って `SSLNoErr` を返した。 エラー状態が発生したときは `X509CertChainInvalidErr` を返すようにコードを修正し、この問題を解決した。
CR103126	(Apache) Linux 上で ssl128 用のプラグインを使って Apache を起動すると、次のエラーが発生した。 `Cannot load /usr/local/apache/libexec/mod_wl_ssl128.so into server: /usr/local/apache/libexec/mod_wl_ssl128.so: undefined symbol: pthread_mutexattr_init` 分析の結果、ビルドの問題であることがわかり、プラグインを再ビルドすることで解決した。
CR103161	「CR097202」と同じ。

RMI-IIOP

変更要求番号	説明
CR090010	WebLogic Server 6.1 SP03 では、`weblogic.ejbc` で生成される多次元配列に対する `BoxedRMI` IDL ファイル `seq2_seq1_octet.idl` に、モジュール / パスの宣言 `org.omg.boxedRMI` が 2 回出現していた。 コードを修正して多次元 `boxedRMI` シーケンスに対する IDL 宣言を正しくすることで、この問題を解決した。
CR093988	WebLogic Server 6.1 SP04 では、ActiveX コンポーネントから IIOP を通して EJB にアクセスすると、WebLogic Server がハングした。スレッド ダンプは、Java 層におけるデッドロックを示していた。 この問題は、Red Hat Linux 7.2 および Windows 2000 での SUN JDK 1.3.1 で発生した。 分析の結果、ソケットを閉じる IIOP ソケット タイムアウト トリガがデッドロックの原因であることがわかった。 コードを修正して、このような状況の発生を防いだ。
CR100334	WebLogic Server 6.1 の旧リリースでは、引数を指定しないでメソッドをディスパッチすると、または戻り値の型が void のメソッドを呼び出すと、WebLogic Server が存在しない可能性のあるデータを読み取ろうとした。 7.0 SP2 から提供されるときはデータは存在せず、他のリリースについては、余分な (不必要な) データを提供することで 6.1 への対応を試みる。
CR100430	6.1 SP04 以前の WebLogic Server と WebLogic Server 8.1 を相互運用できなかった。6.1 と 8.1 のサーバインスタンス間のトランザクションが、6.1 サーバインスタンス上の `ArrayIndexOutOfBoundsException` でタイムアウトした。 この問題は、WebLogic Server 6.1 SP05 において解決した。

CR100334	このリリースでは、WebLogic Server 6.1 SP04 と WebLogic Server 7.0 SP02 の間の相互運用性の問題を解決した。 WebLogic Server 6.1 SP04 では、返された void のパラメータを誤って読み込もうとして `UnmarshalException` が発生した。これは次の場合に発生した。 ■ WebLogic Server 6.1 SP04 が WebLogic Server 7.0 SP02 サーバインスタンスへの呼び出しの結果として、void の戻り値を受け取った。 ■ 6.1 サーバインスタンスにある引数のないメソッドが、WebLogic Server 7.0 SP02 サーバインスタンスによって呼び出された。 この問題は他の ORB との相互運用性を妨げていた。

RMI

変更要求番号	説明
CR096511	WebLogic Server 6.1 SP03 および SP04 では、InitialContext の作成とクローズを繰り返すと、メモリ不足のメッセージが発生した。 分析の結果、`DGCServerHelper` クラスでメモリ リークが発生していることがわかった。 記述子の `dgcpolicy` が「Managed」ではない場合にだけ `LocalServerRef` を登録するようコードを修正することで、この問題を解決した。
CR089481	WebLogic Server 6.1 SP03 では、クライアントが適切に閉じなかったクライアント JDBC 接続が、WebLogic Server によって閉じられず、ガベージ コレクションされなかった。 リモート クライアントが接続を適切に閉じなくても接続が正しく閉じられるようコードを修正することで、この問題を解決した。 **注意：** クライアントサイドのコードが接続を適切に閉じなければならない。WebLogic Server は閉じられていない接続を閉じるが、使用状況と分散ガベージ コレクションの動作によっては、処理に長い時間を要する場合がある。
CR100430	WebLogic Server 6.1 SP04 では、WebLogic Server 8.1 のサーバ インスタンスが WebLogic Server 6.1 SP04 サーバ インスタンスのクライアントになっていると、トランザクションがタイムアウトし、次の例外が発生した。 `java.rmi.RemoteException: Exception while committing Tx: Name=[EJB weblogic.qa.tests.interop.ejb11.txcrossdomain.TravelPlannerBean.planItinerary_Required(java.lang.String,java.util.Properties)],Xid=BEA1-000147799CCFA039D1D7(2653792),Status=Committing,numRepliesOwedMe=0,numRepliesOwedOthers=0,seconds since begin=120,seconds` コードを修正し、この相互運用性に関する問題を解決した。

セキュリティ

変更要求番号	説明
CR051018	WebLogic Server の以前のリリースでは、システム パスワードが暗号化されないで保管されていた。パスワードを暗号化して `password.ini` に格納するようにした。 **注意：** `password.ini` は、WebLogic Server バージョン 7.0 で使用されている `boot.properties` ファイルにはアップグレードできない。
CR100592	IPlanet LDAP レルムがユーザを無効にした後、WebLogic Server 6.1 SP04 が使用できなくなった。この問題は、次のイベントのシーケンスで発生した。 サーバで LDAP レルムを起動する。フォーム認証を使用する Web アプリケーションをデプロイする。LDAP レルムの任意のユーザについて、ユーザ ロックアウトしきい値に達するまで不正なパスワードを使用する。クライアントは 500 番のエラーを受け取り、サーバでは次のエラーが発生する。 `<Mar 9, 2003 7:17:22 PM PST> <Error> <HTTP> <[WebAppServletContext(3017390,security,/security)] Servlet failed with Exception netscape.ldap.LDAPException: error result (19); Exceed password retry limit. Please try later.; Constraint violation` このエラーの後、サーバは認証に関して使用できない状態になった。たとえば、ほかのユーザは Web アプリケーションにログインできず、Console にもアクセスできなかった。 ユーザが無効なパスワードを入力したときの LDAP 例外を検査するようコードを修正し、この問題を解決した。

サーブレット

変更要求番号	説明
CR077922	テストにおいて、セッション バインディング リスナ (`HttpSessionBindingListener`) の `valueBound()` メソッドと `valueUnBound()` メソッドを 1 回だけ呼び出しても、実際にはメソッドが 2 回以上呼び出されていた。 メソッドを重複して呼び出さないようコードを修正した。
CR083624	`config.xml` の `WebServer` 要素に、新しいコンフィグレーション パラメータ `UseHighestCompatibleHTTPVersion` を追加した。このパラメータを指定すると、WebLogic Server は、リクエストで使用されているバージョンと互換性があって最も高いバージョンの HTTP プロトコルを使って、応答を送信する。たとえば、このパラメータを true に設定すると、HTTP/1.0 のリクエストに対して HTTP/1.1 で応答する。このパラメータは、応答のバージョン文字列の部分に対してだけ影響を与える。WebLogic Server 6.1 では、デフォルトの設定は false である。 このパラメータの値は、`config.xml` を編集することで変更できる。Administration Console には、このパラメータの値を変更するためのインタフェースはない。
CR086579	ポート 80 でリスンする iPlanet を通して、フォーム ベースの認証で NSAPI プロキシ プラグインを使用すると、`j_security_check` がリダイレクトを正しく行わず、認証が失敗した。 分析の結果、デフォルトのポートを使用した場合、フォーム ベース認証モジュールが `processRedirectedURL` を呼び出していなかった。
CR086587	WebLogic Server 6.1 SP03 では、Web サーバ ログ (`access.log`) と JDBC サブシステム ログ (`jdbc.log`) が、RootDirectory ではなく、製品のインストール ディレクトリを基準にして作成されていた。 コードを修正し、この問題を解決した。
CR087573	WebLogic Server 6.1 SP01 では、セッションに対して JDBC 永続性を使用し、ネストされた `hashMaps` と `TreeMaps` を含む複雑なオブジェクトを格納した場合、セッションが永続化される前に後続のリクエストが届くと、セッション データが失われた。 `removeSession(id)` を `FileSessionContext.java` の `method sync()` の最後に移動することで、この問題を解決した。

CR087647	「CR055987」に関する修正で、接続が解放されない問題を解決するためのタイムアウトを導入した。この修正で追加された `InterruptedIOException` が処理されていなかった。例外が発生すると、不要な再試行が行われていた。 `InterruptedIOException` をキャッチし、例外を送出し直して再試行を防ぐようにコードを修正することで、この問題を解決した。
CR087984	WebLogic Server 6.1 SP02 では、`HttpSessionListener` の前に `ServletContextListener` が呼び出されていた。2.3 サーブレット仕様に従えば、コンテキスト リスナがアプリケーションのシャットダウンの通知を受ける前に、セッション リスナはセッション無効化の通知を受けなければならない。 仕様に準拠するようコードを修正した。
CR088166	WebLogic Server 6.1 SP03 では、Web アプリケーションが、見つからないファイルの名前を示さずに `java.lang.ClassNotFoundException` を生成していた。 `<[WebAppServletContext(4730538,olam,/olam)] Error loading servlet: ""> java.lang.ClassNotFoundException:` 分析の結果、WebLogic Server が `auth-filter` タグに対する Null または空の文字列値を正しく処理していないことがわかった。コードを修正し、この問題を解決した。

CR088350	HttpProxyServlet を使って KnowNow からサードパーティのイベント サーバに対するプロキシ処理を行うと、Internet Explorer 6 が応答しないまま無限に待っていた。 分析の結果、KnowNow サーバについて次のことがわかった。 ■ HTTP 1.0 のリクエストに対してのみ応答する。 ■ ContentLength を送信しない。 コンテンツ長がなく、データがチャンク転送されていない場合、WebLogic Server はストリームの最後まで読み取る。KnowNow サーバがデータを送信し続けると、HttpProxyServlet はデータを読み続ける。 新しいコンフィグレーション パラメータ UseKnowNow を導入することで、この問題を解決した。このパラメータを指定すると、HttpProxyServlet の動作は次のようになる。 ■ バイト単位で読み書きする。 ■ HTTP/1.0 を使用して KnowNow にリクエストを送信する。 このパラメータを使用するには、Web アプリケーション HttpClusterServlet に対する web.xml に次の構文を追加する。 `<init-param> <param-name>UseKnowNow</param-name> <param-value>true</param-value> </init-param>`

CR088384	WebLogic Server 6.1 SP02 では、HttpServletRequest からの getContextPath が不正な値を返した。例を示す。
	`http://localhost//webapp/foo`
	上のようなリクエストに対し、getContextPath は次の値を返した。
	`/webap`
	コードを修正し、この問題を解決した。

CR088478	HttpClusterServlet およびすべてのプラグインに対する MaxSkips パラメータが使われなくなり、MaxSkipTime パラメータに置き換えられた。 このパラメータを指定すると、次のような処理が行われる。 ■ プラグインは、サーバを「不良 (bad)」とマークすると、その時刻を記録する。 ■ 不良サーバは、MaxSkipTime に達するまでロード バランシング サイクルにおいてスキップされる。 ■ その後、新しいサーバ リストが強制される。

CR088747	weblogic.xml で CookiePath を明示的に設定していない Web アプリケーションに対するリクエストの後で、WebLogic Server セッションを確立し直そうとすると、セッションが失われた。 分析の結果、ブラウザ (IE6) が、次のような異なるパスを含む 2 つのセッション クッキーを返していた。 ■ パスとして /webapp の設定された JSESSIONID ■ パスとして / の設定された JSESSIONID J2EE によれば、ヘッダー内の複数のクッキーは、Path 属性が限定的なものから一般的なものの順に並んでいなければならない。WebLogic Server は、パフォーマンスの理由で、JSESSIONID に対するクッキーの並びを逆に検索していた。そのため、より限定的ではないパスの方を検出していた。 ヘッダー内の最初のクッキーを使うようにコードを修正することで、この問題を解決した。
CR088759	WebLogic Server 6.1 SP03 では、redirect-with-absolute-url = false が j_security_check に対して機能しなかった。 これは、j_security_check が絶対パスを格納し、sendRedirect() メソッドにそれを渡していたためであった。 j_security_check も redirect-with-absolute-url を認識しなければならなかった。 redirect-with-absolute-url の設定に従って FormSecurityModule のリダイレクト リクエストを行うよう、コードを修正した。
CR089582	Solaris 8 の WebLogic Server 6.1 SP03 では、Host ヘッダー フィールドのない HTTP/1.1 リクエストを受け取ると、WebLogic Server は「Date: Thu, 01 Jan 1970 00:00:00 GMT」という値の Date ヘッダー フィールドを返していた。そのため、不正なリクエストがいつ生成されたのかを特定できなかった。 不正リクエスト エラーを送信する前にリクエスト生成時刻を設定するようにコードを修正して、この問題を解決した。
CR089868	WebLogic Server 6.1 サービス パック 3 で行われたパフォーマンスの改善 (6-235 ページの「083654」を参照) により、マルチバイトのヘッダが壊れるようになった。対応策はあるが、SP03 と SP04 では対応策を適用するためにはパッチが必要である (詳細については 5-30 ページの「089868」を参照)。 SP05 では、CR089868 のパッチを適用する必要はなくなった。

CR090033	WebLogic Server 6.1 SP03 では、WL-Proxy-SSL ヘッダーが Host ヘッダーより前にあると、リクエストの解析において Null ポインタ例外が送出された。この現象は、クライアントが HTTPS で動作し、SSL アクセラレータ、IPlanet、および WebLogic Server が HTTP で動作すると発生した。この問題は、コンフィグレーションに SSL アクセラレータが含まれているときにだけ再現した。 `java.lang.NullPointerException at weblogic.servlet.internal.ServletRequestImpl.setField(ServletRequestImpl.java:1903) at weblogic.servlet.internal.RequestParser.parse(RequestParser.java:254) at weblogic.servlet.internal.MuxableSocketHTTP.dispatch(MuxableSocketHTTP.java:390) at weblogic.socket.MuxableSocketDiscriminator.dispatch(MuxableSocketDiscriminator.java:275) at weblogic.socket.NTSocketMuxer.processSockets(NTSocketMuxer.java:633) at weblogic.socket.SocketReaderRequest.execute(SocketReaderRequest.java:23) at weblogic.kernel.ExecuteThread.execute(ExecuteThread.java:152) at weblogic.kernel.ExecuteThread.run(ExecuteThread.java:133)` このエラーは、リクエストが解析中でコンテキストがまだわからず、コンテキストが Null であるために発生した。WL-Proxy-SSL が、HOST ヘッダーより前にあった。 コードを修正し、この問題を解決した。
CR090188	WebLogic Server 6.1 SP03 では、チャンク データ転送の場合、転送されるデータの前にチャンク サイズが付加され、転送されるデータに余分な文字が追加されていた。 `setChunking` を、`ServletRequestImpl` の `mergePostParams` から `request.setInputStream` の後の `MuxableSocketHttp` に移動することで、この問題を解決した。 この変更により、サーブレットと JSP はチャンク データをデコードする必要がなくなった。WebLogic Server のサーブレット エンジンが、チャンク データを自動的にマージする。

CR090225	テストの際、デバイス ドライバが HTTP を使ってリクエストを行うようにすると (`request.aux.%2a.jsp` または `aux.*.jsp`)、リクエストを処理するスレッドがハングした。 `JSPServlet` のコードを修正して、この問題を解決した。上で名前を挙げたデバイス ドライバはすべて、長さ 0 のファイルとして扱われるようになった。WebLogic Server は、この種のファイルを読み込まずに提供する。 http://dev2dev.bea.com/resourcelibrary/advisoriesnotifications/BEA03-14.jsp の「SECURITY ADVISORY (BEA03-14.05)」を参照すること。
CR090465	ユーザ定義のエラー ページがある Web アプリケーションをホストすると、WebLogic Server が、400 番のエラーに対して `Connection: close` ヘッダーを設定しないで次の応答を返し、直ちに接続を閉じていた。 `HTTP/1.1 400 Bad Request Date: Thu, 01 Jan 1970 00:00:00 GMT Content-Length: 463 Content-Type: text/html Last-Modified: Thu, 07 Nov 2002 21:56:52 GMT <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN"> <html>` コードを修正し、この問題を解決した。
CR090555	X_WEBLOGIC_CLUSTER_HASH をチェックした WebLogic Server が、`HttpClusterServlet` に対して Null ポインタ例外を送出していた。スタック トレースは、以下のような内容であった。 `<Tue Nov 05 15:58:00 PST 2002>: Start connection timeout scheduler <Tue Nov 05 15:58:00 PST 2002>: GenericProxyServelt: init() <Tue Nov 05 15:58:00 PST 2002>: HttpClusterServlet:init() <Tue Nov 05 15:58:00 PST 2002>: ===New Request===GET /base;JSESSIONID=9G8ZGUTNwnNTXm8B5V5z41e hG0T3oChTPvhnXe7EHCZ1IQI5lgF4!822557425 HTTP/1.1 <Tue Nov 05 15:58:00 PST 2002>: Found cookie: 822557425 <Tue Nov 05 15:58:00 PST 2002>: Trying to get JVMID id from server: coolant!7003!443 <Tue Nov 05 15:58:00 PST 2002>: Update dynmaic hash: WakHYk8LlxQ7XrnhkSXEohfZMAw <Tue Nov 05 15:58:00 PST 2002>: java.lang.NullPointerException at weblogic.servlet.proxy.HttpClusterServlet.initDynamicServerList(HttpClusterServlet .java:761) at weblogic.servlet.proxy.HttpClusterServlet.getPreferred(HttpClusterServlet.ja va:622) at weblogic.servlet.proxy.HttpClusterServlet.service(HttpClusterServlet.java:21` 分析の結果、応答のヘッダーがランダムな順序で HashMap に格納されていることがわかった。ハッシュを設定する前にサーバ リストを設定するよう、コードを修正した。

CR090665	WebLogic Server 6.1 SP03 では、サーブレット フィルタは、chain.doFilter() を呼び出してクライアントに応答を送信した後、応答ラッパーに対して flushBuffer() を明示的に呼び出さなければならなかった。そのため、アプリケーション サーバ別に異なるフィルタを作成する必要があった。そうしないと、応答にデータが含まれなかった。この問題は、printWriter の OutputStreamWriter オブジェクトにデータがバッファリングされていたために発生した。 ユーザ コードが flushBuffer() を明示的に呼び出す必要がないように、コードを修正した。

CR091410	HTTP ログ オプションで「拡張フォーマット」を選択し、サーバを再起動すると、次のような Null ポインタ例外が送出された。
	`<Oct 9, 2002 6:45:48 PM PDT> <Error> <kernel> <000802> <ExecuteRequest failed`
	`java.lang.NullPointerException`
	`java.lang.NullPointerException`
	ファイル名が Null の場合は access.log ファイルを開くようにコードを修正して、この問題を解決した。

CR091759	RequestDispatcher.forward が、URL ヘッダーの最後にあるバージョン番号を除去していた。転送されるリクエストでは、次のように記述されていなければならない。
	`Request URI: /RequestDispatcher/SnoopServlet/myTest.txt;37`
	実際には、次のような記述が転送されていた。
	`Request URI: /RequestDispatcher/ProxySnoopServlet/myTest.txt`
	forward() を行うときは setRequestURI() で processPathParameters を false に設定するようにコードを修正して、この問題を解決した。

CR091878	負荷テストの間に HttpClusterServlet をアンデプロイすると、.IndexOutOfBoundsException が発生した。
	`<Nov 27, 2002 4:05:11 PM PST> <Error> <HTTP> <101268>`
	`<ServletContext(id=5220998,name=proxywebapp,context-path=): Failed while destroying servlet:`
	`java.lang.IndexOutOfBoundsException: Index: 2, Size: 2`
	`IndexOutOfBoundsException while destroying HttpClusterServlet`
	分析の結果、pool.remove(i) で接続プールの最後の要素を削除するときにエラーが発生していた。
	コードを修正して、この問題を解決した。

CR092255	WebLogic Server 6.1 SP04 では、クッキーの値に引用符 (") が含まれている場合、引用符が無視されていた。次に例を示す。 `Cookie cookieWithQuotedValue = new Cookie("TestQuotedCookie", "\"TestValue\"");` このような構文では、「"TestValue"」ではなく「TestValue」がクッキー値として追加されていた。 クッキー値内の引用符を残すようにコードを修正した。
CR092428	WebLogic Server 6.1 SP04 では、Netscape のクッキーにカンマ (,) が含まれていると、「`Got bad cookie header`」(クッキー ヘッダー不正) というエラー メッセージが表示された。 コードを修正して、この問題を解決した。
CR092778	WebLogic Server 6.1 SP04 では、`HttpRequest` に対する `JISAutoDetect` のエンコーディングで、次の `UnsupportedEncodingException` が発生した。 `java.io.UnsupportedEncodingException: JISAutoDetect at sun.io.Converters.getConverterClass(Converters.java:102) at sun.io.Converters.newConverter(Converters.java:133) at sun.io.CharToByteConverter.getConverter(CharToByteConverter.java:62) at weblogic.servlet.internal.ServletRequestImpl.setCharacterEncoding(ServletRequestImpl.java:344) ...` この問題は、WebLogic Server 6.1 SP03 では発生しなかった。 この問題は、`ServletRequestImpl.java` が、`sun.io.ByteToCharConverter` ではなく `CharToByteConverter` を使用したために発生した。`CharToByteConverter` は、入力コンバータ用である。`CharToByte` は、出力コンバータ用である。`JISAutoDetect` は、入力ストリームに対してだけ使用できる。 コードを修正し、この問題を解決した。

CR093167	WebLogic Server 6.1 SP03 では、`HTTPClusterServlet` を使ってバックエンド クラスタのセキュアな Web アプリケーションにアクセスすると、ユーザ名とパスワードを入力した後で、Internet Explorer 5.5 が 30 秒間ハングした。この問題は他のブラウザでは再現せず、プラグインを介さないでバックエンド サーバに直接アクセスすると、Internet Explorer でも発生しなかった。 この問題は、サーバから `Connection: Close` ヘッダーを受け取ったときに Internet Explorer が接続を閉じないために発生していた。WebLogic Server に直接アクセスしたときは、401 エラーを送出した時点で WebLogic Server が接続を閉じていた。しかし、`HttpProxyServlet` を介したときは、バックエンド サーバが接続を閉じても、ブラウザと ProxyServlet の接続が維持されていた。 バックエンド サーバから `Connection: Close` ヘッダーが返されたときはフロントエンドの接続がキープ アライブを無効にするように `GenericProxyServlet` のコードを修正して、この問題を解決した。これにより、Internet Explorer は、WebLogic Server で接続がタイムアウトになるまで待つのではなく、認証の詳細を直ちに返送するようになった。

CR093209

新規にインストールした WebLogic Server 6.1 SP04 では、Java プロトコルに設定されたプロパティを持つ正常にデプロイされた Web アプリケーションにアクセスすると、Null ポインタ例外が送出された。ブラウザは、500 番のサーバ内部エラーを返した。アップグレードインストールでは、この問題は発生しなかった。

```
<Dec 16, 2002 11:59:05 AM PST> <Error> <HTTP>
<[WebAppServletContext(2092664,sampleApp,/sampleApp)] Servlet
failed with
Exception
java.lang.NullPointerException
        at
weblogic.servlet.JSPServlet.service(JSPServlet.java:132)
        at
javax.servlet.http.HttpServlet.service(HttpServlet.java:853)
        at
weblogic.servlet.internal.ServletStubImpl.invokeServlet(Servle
tStubImpl.java:262)
        at
weblogic.servlet.internal.ServletStubImpl.invokeServlet(Servle
tStubImpl.java:198)
        at
weblogic.servlet.internal.WebAppServletContext.invokeServlet(W
ebAppServletContext.java:2637)
        at
weblogic.servlet.internal.ServletRequestImpl.execute(ServletRe
questImpl.java:2359)
        at
weblogic.kernel.ExecuteThread.execute(ExecuteThread.java:139)
        at
weblogic.kernel.ExecuteThread.run(ExecuteThread.java:120)
>
```

分析の結果、`ZipSource.getURL()` のエラーであることがわかった。`source.getURL()` の Null 値をチェックするロジックを追加して、この問題を解決した。

CR093634	WebLogic Server 6.1 SP03 では、`HttpRequest.getRemoteAddr()` メソッドが、クライアント ブラウザの IP アドレスではなく、`HttpClusterServlet` の IP アドレスを返した。 分析の結果、`HttpClusterServlet` が固有のヘッダー「WL-Proxy-Client-IP」を設定せず、代わりにソケットがリスンしている IP アドレスを返していることがわかった。 `WebLogic Plugin Enabled` パラメータが有効になっている場合は「WL-Proxy-Client-IP」ヘッダーを設定するように `HttpClusterServlet` を修正して、この問題を解決した。
CR093755	次のエラー メッセージが誤って生成されていた。 `<Warning> <HTTP> <WebAppServletContext(5294080,blue,/blue) One of the getParameter family of methods called after reading from the ServletInputStream, not merging post parameters>` メッセージが誤って生成されないようコードを修正した。
CR094488	HTTP を使って開始されたセッションにおいてセッション データを失うことなく HTTPS リソースに安全にアクセスできるようにする、新しい機能を追加した。この新機能を有効にするには、`config.xml` の `WebServer` 要素に `AuthCookieEnabled="true"` を追加する。 `<WebServer Name="myserver" AuthCookieEnabled="true"/>` この設定を追加すると、HTTPS 接続を通して認証を行った時点で、新しいセキュアなクッキーがブラウザに送信される。また、このクッキーがブラウザから送信された場合にだけ、セッションは、セキュリティ制約のある他の HTTPS リソースにアクセスできる。 **注意：** 普通の HTTP を通して認証が行われた場合は、セキュアなクッキーは設定されず、HTTPS リソースに対してセキュアなクッキーが要求されることはない。保護されていない HTTPS リソースへのアクセスでは、クッキーは検査されない (ブラウザから送信されていないため)。これにより、ブラウザは、ユーザがログインしていなくても、保護されていない HTTPS リソースにアクセスできる。

CR094773	「CR094561」と同じ。 WebLogic Server 6.1 SP02 では、アプリケーションがインメモリ セッション レプリケーションまたはレプリケートされたステートフル セッション Bean を使うと、2 人のユーザの間でセッション データを共有できてしまうという、潜在的なセキュリティの弱点が発見された。この障害の原因は、競合状況の結果として 2 つのセッションに同じ ROID (レプリカ可能オブジェクト ID と呼ばれる内部 ID) が割り当てられるためであった。たとえば、セッション A に ROID が割り当てられた後、セッション B に同じ ROID が割り当てられる。すると、それ以降のセッション B のリクエストはすべて、A の HTTP セッション /SFSB にマップされる。HTTP セッションが混じり合うと、セッション B のユーザがセッション A のコンテンツを見ることができた。この問題が発生する可能性は非常に低く、意図的に利用することはできなかった。 `ROID.create()` の同期を取るようにコードを修正し、重複する ROID が複数のセッションに割り当てられないようにすることで、この問題を解決した。 http://dev2dev.bea.com/resourcelibrary/advisoriesnotifications/BEA03-26.01.jsp の「SECURITY ADVISORY (BEA03-26.01)」を参照すること。
CR094997	WebLogic Server SP02 では、対話する 2 つの Web アプリケーションが異なるセッション クッキー名を使用すると、セッション ID が失われた。 分析の結果、Web アプリケーションで `cookieName` が指定されていて、別の Web アプリケーションからのリソースがインクルードされており、リクエスト中で外部リソースのインクルードより前に `RemoteUser` が存在していると、セッション属性が正しく更新されないことがわかった。 コードを修正して、この問題を解決した。

CR095548	テストにおいて、永続性タイプが JDBC のとき、セッション データがときどき失われた。次の例外が発生した。
	`weblogic.qa.tests.cluster.webapp.plugins.RequestSessionBeforeInvalidateTest.testRequestSessionBeforeInvalidationTest()`
	`Session Was Invalidated`
	`Sent: http://qa47-1:7771/pluginsjdbc/SessionBeforeInvalidate.jsp?count=1`
	`Received: <html><body>PROBLEM: session is not the same as the first`
	`request!FirstRequest sid: 2tjgF02nPIGgHGyojFubyOIjqt0ZAVK4no1Ny0afTxGKMrOj25z5!2356550501043194848402Sid`
	`isRequestedSessionIdValid()` メソッドのコードを修正し、この問題を解決した。

CR095570	WebLogic Server 6.1 SP04 では、サーブレットの HTTP ヘッダーに Null 値が設定されていると、NullPointerException が送出された。 <Jan 22, 2003 11:11:53 AM KST> <Error> <Kernel> <Execute Request failed java.lang.NullPointerException: Start server side stack trace: java.lang.NullPointerException at weblogic.servlet.internal.ResponseHeaders.writeHeaders(ResponseHeaders.java:360) at weblogic.servlet.internal.ServletResponseImpl.writeHeaders(ServletResponseImpl.java:902) at weblogic.servlet.internal.ServletOutputStreamImpl.sendHeaders(ServletOutputStreamImpl.java:239) at weblogic.servlet.internal.ServletOutputStreamImpl.flush(ServletOutputStreamImpl.java:121) at weblogic.servlet.internal.ServletOutputStreamImpl.commit(ServletOutputStreamImpl.java:484) at weblogic.servlet.internal.ServletOutputStreamImpl.finish(ServletOutputStreamImpl.java:531) at weblogic.servlet.internal.ServletResponseImpl.send(ServletResponseImpl.java:1091) at weblogic.servlet.internal.ServletRequestImpl.execute(ServletRequestImpl.java:2364) at weblogic.kernel.ExecuteThread.execute(ExecuteThread.java:139) at weblogic.kernel.ExecuteThread.run(ExecuteThread. java:120) End server side stack trace at weblogic.servlet.internal.ResponseHeaders.writeHeaders(ResponseHeaders.java:360) at weblogic.servlet.internal.ServletResponseImpl.writeHeaders(ServletResponseImpl.java:902) at weblogic.servlet.internal.ServletOutputStreamImpl.sendHeaders(ServletOutputStreamImpl.java:239) at weblogic.servlet.internal.ServletOutputStreamImpl.flush(ServletOutputStreamImpl.java:121) at weblogic.servlet.internal.ServletOutputStreamImpl.commit(ServletOutputStreamImpl.java:484) at weblogic.servlet.internal.ServletOutputStreamImpl.finish(ServletOutputStreamImpl.java:531) at weblogic.servlet.internal.ServletResponseImpl.send(ServletResponseImpl.java:1091) at weblogic.servlet.internal.ServletRequestImpl.execute(ServletRequestImpl.java:2364) at weblogic.kernel.ExecuteThread.execute(ExecuteThread.java:139) at weblogic.kernel.ExecuteThread.run(ExecuteThread.java:120) ResponseHeaders.java の Null 値をチェックするロジックを追加することで、この問題を解決した。

CR096459	WebLogic Server 6.1 SP02 にデプロイされた Web アプリケーションに対し、`flushRes=true` が設定された HTTP Version 1.0 を使用するブラウザからアクセスしたときの応答時間に問題があった。この問題は、`flushRes=false` の場合には発生しなかった。 分析の結果、`useKeepAlive` が誤って再初期化されていることがわかった。閉じられなければならないソケット接続が閉じられず、クライアントはソケットのタイムアウトを待っていた。 リクエストと応答を設定するときに `useKeepAlive` を初期化するよう、コードを修正した。
CR097719	WebLogic Server SP02、SP03、および SP04 では、WAP デバイスから特定の POST リクエストを取得すると、Web アプリケーションで `java.util.ConcurrentModificationException` が発生した。 `<29.jan.03 13:03:05 WET> <Error> <HTTP> <[WebAppServletContext(2810713,TnMFF,/TnMFF)] Servlet failed with Exception java.util.ConcurrentModificationException at java.util.HashMap$HashIterator.next(HashMap.java:736) at weblogic.utils.enumerations.IteratorEnumerator.nextElement(IteratorEnumerator.java:25) at com.colibria.core.xmlswitch.XMLSwitch.getQueryStringXML(XMLSwitch.java:347) at ...` 分析の結果、文字セットがリクエストの `content-type` と関連付けられていると、アプリケーションにおいて、コードがエンコーディングを使ってデータを再び読み取り、クエリのパラメータをリセットしていることがわかった。アプリケーションは既に列挙オブジェクトを持っており、それ (`request.getParameterNames`) に対して反復を行って、パラメータの値 (`request.getParameterValues(k)`) を取得しようとした。その時点で、クエリのパラメータは消去されており、`getParameterValues` メソッドがそれを設定しようとした結果、例外が発生した。 文字セットがリクエストの `content-type` と関連付けられていてもデータを再び読み取ることができるよう、消去されたクエリ パラメータを設定するようにコードを修正して、この問題を解決した。
CR098955	`weblogic.httpd.logRotationBeginTime` に過去の日付が設定されていると、WebLogic Server HTTP サーバ ログに対して設定されているログ ローテーション時間が正しく計算されなかった。 `INT` から `LONG` への計算で使用されるオペランドのデータ型を変更することで、この問題を解決した。

CR099984	サーブレットのコードが入力ストリームにアクセスできるようになる前に、サーブレット エンジンが入力ストリームを使用していた。内部読み取りが入力ストリームを使用しないようコードを修正し、この問題を解決した。
CR100265	受信したリクエストにクッキーがなくても、有効な資格が含まれている場合には、WebLogic Server は保護されたリソースへのアクセスを許可しなければならない。しかし、アクセスは許可されず、次の例外が発生していた。 `java.lang.NullPointerException at weblogic.servlet.security.internal.SecurityModule.checkAuthCookie(SecurityModule.java:579) at weblogic.servlet.security.internal.BasicSecurityModule.checkUserPerm(BasicSecurityModule.java:157) at ...` コードを修正し、この問題を解決した。

CR100572	正しくない URI を含むリクエストをプラグインから受信すると、WebLogic Server は次のスタック トレースを送出していた。 `<Mar 8, 2003 3:27:20 PM PST> <Error> <Socket> <BEA-000421> <Uncaught Throwable i n processSockets java.lang.NullPointer Exception. java.lang.NullPointerException at weblogic. servlet.internal.ServletResponseImpl.writeHeaders(ServletRespo nseImpl.java:968) at weblogic.servlet.internal.Servlet OutputStreamImpl.sendHeaders(ServletOutputStreamImpl.java:244) at weblogic.servlet.internal.ServletOutputStreamImpl. flush(ServletOutputStreamImpl.java:121) at weblogic.servlet. internal.ServletOutputStreamImpl.commit(ServletOutputStreamImp l.java:486) at weblogic.servlet.internal.ChunkOutput. commit(ChunkOutput.java:269) at weblogic.servlet.internal. ChunkOutputWrapper.write(ChunkOutputWrapper.java:91) at weblogic.servlet.internal.ChunkWriter.write(ChunkWriter.java:3 7) at java.io.Writer.write(Writer.java:150) at java.io.PrintWriter.write(PrintWriter.java:230) at java.io.PrintWriter.write(PrintWriter.java:247) at java.io.PrintWriter.print(PrintWriter.java:378) at weblogic.servlet.internal.ServletResponseImpl.sendError(Servle tResponseImpl.java:420) at weblogic.servlet.internal. ServletResponseImpl.sendError(ServletResponseImpl.java:373) at weblogic.servlet.internal.MuxableSocketHTTP.dispatch (MuxableSocketHTTP.java:512) at weblogic.socket.MuxableSocket Discriminator.dispatch(MuxableSocketDiscriminator.java:281) at weblogic.socket.NTSocketMuxer.processSockets(NTSocket Muxer.java:102) at weblogic.socket.SocketReaderRequest. execute(SocketReaderRequest.java:32) at weblogic.kernel. ExecuteThread.execute(ExecuteThread.java:178) at weblogic.kernel.ExecuteThread.run(ExecuteThread.java:151)` コードを修正し、この問題を解決した。

CR100172	複数のチャンクになったリクエストがサーブレットにポストされると、複数のリクエストが NumberFormationException で失敗した。ほとんどの場合、代替リクエストは正常に処理され、次のリクエストが失敗した。同じ接続で複数のリクエストが行われているように見えた。例外のスタック トレースは、次のようなものであった。 <ExecuteThread: '11' for queue: 'default'> <kernel identity> <> <101017> <[ServletContext(id=4864139,name=387551,context-path=)] Root cause of ServletException> java.lang.NumberFormatException: at java.lang.Integer.parseInt(Integer.java:430) at weblogic.utils.http.HttpChunkInputStream.readChunkSize(HttpChunkInputStream.java:118) at weblogic.utils.http.HttpChunkInputStream.initChunk(HttpChunkInputStream.java:72) 分析の結果、PostInputStream に 2 つの問題があることがわかった。 ■ PostInputStream を再利用するときに、制御変数 readAllChunks をリセットしていなかった。 ■ isChunkComplete がチャンクの終了を正しく検出していなかった。 PostInputStream でストリームを最後まで読み取り、HttpChunkInputStream でチャンクの終わりを正しく検出するようコードを修正して、この問題を解決した。
CR100837	WebLogic Server 6.1 SP04 が、JSP ページの JSP includes に対して出力を誤って送っていた。 コードを修正し、この問題を解決した。

WLS ツアー / サンプル

変更要求番号	説明

CR087117	WebLogic Server 6.1 SP04 では、ユーザがパスワードを入力する前に管理サーバ、Pet Store サーバ、および Example サーバを起動すると、パスワード エラーが発生した。
	`Starting WebLogic Server ....`
	`<Oct 1, 2002 9:22:10 AM EDT> <Notice> <Management> <Loading configuration file .\config\petstore\config.xml ...>`
	`<Oct 1, 2002 9:22:13 AM EDT> <Emergency> <Security> <Authentication failure - reenter password to boot weblogic server:>`
	コードを修正し、この問題を解決した。

CR087366	WebLogic Server 6.1 SP04 では、サンプル サーバのインデックス ページ (`index.jsp`) の [WebLogic Server サンプル索引] リンクが機能しなかった。リンクをクリックすると、次のメッセージが表示された。 `“Netscape is unable to find the file or directory named /D:/bea61. Check filename and try again.` 分析の結果、WebLogic Server のインストールで使用された BEAHOME に空白文字が含まれていた。 `index.jsp` の関係する `href` に二重引用符を追加して BEAHOME の空白文字を許容することで、この問題を解決した。
CR090776	WebLogic Server 6.1 SP04 から WebLogic Server 8.1 へのアップグレードの際に、Pet Store アプリケーションをデプロイするとエラーが発生した。 分析の結果、デプロイメント記述子に EJB の参照がないことがわかった。この参照がないことは、WebLogic Server 8.1 では捕捉されたが、WebLogic Server 6.1 では捕捉されなかった。 Pet Store のデプロイメント記述子を修正して、この問題を解決した。
CR092881	Unix AIX 上の WebLogic Server 6.1 SP04 では、ユーザが `StartPetStore.sh` を使って PetStore を開始しようとすると、次のエラーが発生した。 `java.lang.SecurityException: Unable to locate a login configuration.` `StartPetStore.sh` を修正して、この問題を解決した。

CR096147	WebLogic Server のサンプル simpFML32 では、dom1config に対して dmloadcf が失敗した。 分析の結果、*DM_LOCAL_SERVICES に次の記述が含まれていた。 *DM_LOCAL_SERVICES REVERSE_STRING RDOM=”TDOM1” これは誤りで、次が正しい記述である。 *DM_LOCAL_SERVICES REVERSE_STRING LDOM="TDOM1" Tuxedo のコンフィグレーション ファイル dom1config のエラーを修正し、ローカル ドメイン TDOM1 で提供されるローカル サービス REVERSE_STRING を Tuxedo がエクスポートできるようにした。
CR102138	SSL クライアント サンプルのドキュメントが正しくなかった。WL_HOME\lib\cacerts を JAVA_HOME\jre\lib\security\jssecacerts にコピーするように指示されていた。 指示を修正し、sslclient サンプル ディレクトリに jssecacerts ファイルを追加し、JSSE サンプルに対する指示にサンプル出力を追加することで、この問題を解決した。

Web サービス

変更要求番号	説明
CR087529	WebLogic Server 6.1 SP03 では、`samples\examples\webservices\rpc\weatherEJB` のサンプル Web サービスに対するクライアントは、`wsgen` でビルドされ、`weblogic.jar` ではなく `client.jar` だけを使ってデプロイされていた。Java クライアントが非 SSL の Web サービスを呼び出すと、次の例外が送出された。 `<< Missing class files in servicegen client.jar. Getting Exception in thread "main" java.lang.NoClassDefFoundError: weblogic/net/http/HttpsURLConnection at weblogic.soap.http.SoapContext.lookup(SoapContext.java:87) at javax.naming.InitialContext.lookup(InitialContext.java:350) at com.verizon.client.client.main(client.java:72) >>` 分析の結果、何も手を加えていない最低限の機能には `weblogic.net.http.HttpsURLConnection` クラスが必要であり、`client.jar` にこのクラスを含める必要があることがわかった。 `client.jar` に `weblogic.net.http.HttpsURLConnection` を追加して、この問題を解決した。

CR091862

WebLogic 6.1 SP04 では、XML ドキュメントの DTD 部分を使うと、XML パーサに CPU やメモリを大量に使用させることができた。

XML ドキュメントの DTD 部分を使うと、ドキュメントで名前付きのエンティティを定義できる (定義済みの <、> などは除く)。エンティティは、他のエンティティを使って定義できる (再帰指定は XML 1.0 では禁止されている)。エンティティは、参照されると、XML ドキュメント内で展開される。攻撃は、あるエンティティを定義して参照するという方法で行われる。このエンティティは、別のエンティティの 2 つのインスタンスを使って定義されていて、それがさらに別のエンティティの 2 つのインスタンスによって定義される、ということが繰り返される。この定義プロセスは必要な長さだけ繰り返すことができ、通常は、100 レベルのネストで十分である。100 番目のエンティティは、単に文字列として定義しなければならない。このような定義は、理論的には、最初のエンティティに 100 番目のエンティティの値を 2^99 (2 の 99 乗) 個連結したものが含まれているのと同じ影響を及ぼす。

次に例を示す (DTD は、XML 宣言の後、XML ドキュメントの root 要素の前に置く)。

```
<!DOCTYPE root [
<!ENTITY x100 "foobar">
<!ENTITY x99 "&x100;&x100;">
<!ENTITY x98 "&x99;&x99;">
<!ENTITY x97 "&x98;&x98;">
 ...
<!ENTITY x3 "&x4;&x4;">
 <!ENTITY x2 "&x3;&x3;">
 <!ENTITY x1 "&x2;&x2;"> ]>
```

アプリケーションがドキュメント内にあるこの最初のエンティティを参照すると (&x1; という構文を使って)、ドキュメントのそれ以外の部分には問題がなくても、このエンティティを展開するために XML パーサが非常に大きな CPU 負荷またはメモリ負荷を必要とするため、結果として DoS 状態が発生する。

コンフィグレーション可能なシステム プロパティ `weblogic.apache.xerces.maxentityrefs` を新しく追加することで、この問題に対処した。

このプロパティの使用方法は、次のとおりである。

```
java -Dweblogic.apache.xerces.maxentityrefs=10000
```

http://dev2dev.bea.com/resourcelibrary/advisoriesnotifications/BEA02-23.jsp の「SECURITY ADVISORY (BEA02-23.01)」を参照すること。

CR095044	Web サービスの動作が例外を送出した場合、<SOAP-ENV:Fault> 要素の内容とセマンティクスを示すために org.apache.soap.Fault が使用されるとき、Fault.getDetailEntries() メソッドの呼び出しは例外の Vector を返さなければならない。
	WebLogic 6.1 SP03 では、これとは異なり、このメソッド呼び出しは Null を返し、クライアントは例外のスタック トレースを見ることができなかった。SP02 では、この問題は発生しなかった。
	クライアントは、WebLogic 6.1 SP02 または WebLogic 6.1 SP03 のサーバ インスタンスにアクセスする際、同じ CLASSPATH で動作する。問題は、次のようなクライアント CLASSPATH で再現した。
	`CLASSPATH=.;.\soap.jar;.\xerces.jar;.\j2ee.jar`
	動作の違いは、SP03 では、障害の詳細の内容が <![CDATA[ ]]> の中にラップされていることである。これは、スタック トレースに次のような内容が含まれる場合のパーサでの障害を防ぐために行われている。
	`<<no stacktrace available>>`
	新しいサーブレット初期化パラメータ cdata-fault-detail を追加することで、この問題を解決した。このパラメータを false に設定すると、障害の詳細は <![CDATA[ ]]> にラップされなくなる。
	`<servlet> ...`
	`<servlet-class>weblogic.soap.server.servlet.StatelessBeanAdapter</servlet-class> <init-param>`
	`<param-name>cdata-fault-detail</param-name>`
	`<param-value>true</param-value> </init-param> ...`
	cdata-fault-detail を false に設定した場合は、スタック トレースに次のような記述が含まれると、パーサ エラーが発生する。
	`<<no stacktrace available>>`

CR096906	WebLogic 6.1 SP04 では、WebLogic Server によって生成された Web サービスが呼び出されたとき、dateTime エントリが正しく返されなかった。タイム ゾーン オフセットが正の時 (たとえば、GMT+1 = CET (デンマークのタイム ゾーン))、dateTime エントリのタイム ゾーン オフセット部分の「+」文字が省略されていた。次に例を示す。 CET の 2003 年 2 月 3 日 9 時 12 分 4 秒は、dateTime では次のように表記されなければならない。 <date xsi:type="xsd:dateTime">2003-02-03T09:12:04.000+01:00 </date> Weblogic では、次のように表記されていた。 <date xsi:type="xsd:dateTime">2003-02-03T09:12:04.00001:00 </date> コードを修正し、この問題を解決した。
CR098364	WebLogic 6.1 SP03 では、Ant を使用して、次のような WebLogic タスクを使う Web サービスの .war ファイルを生成した場合、Transactions.jar ファイルに含まれる EJB が 155 個より多いと、wsgen が失敗していた。 <taskdef name="wsgen" classname="weblogic.ant.taskdefs.ejb.WSGen"/> CompilerInvoker のコードを修正することで、この問題を解決した。
CR099255	WebLogic 6.1 SP04 では、weblogic.soap.WebServiceProxy クラスの invoke() メソッドが、クラスに証明書が設定されているかどうかをチェックしていなかった。 そのため、双方向の認証が失敗した。または、証明書が InitialContext に渡された場合は、HTTP プロトコルが機能しなかった。 コードを修正し、この問題を解決した。

WebLogic Tuxedo

変更要求番号	説明
CR084435	WebLogic 6.1 SP02 では、WTC の接続ポリシーが ON_DEMAND と ON_STARTUP に設定されていると、フェイルオーバおよびフェイルバックの状況において、接続ポリシーが正しく機能しなかった。接続ポリシーを ON_STARTUP に設定すると、フェイルオーバが機能しなかった。問題の再現を試みた結果、関連するエラーが明らかになった。 1. WTC サービス リクエストが、bdmconfig ファイルのサービス セクションでリストされているセカンダリ ドメインにフェイルオーバしていた。 2. WTC サービス リクエストは、bdmconfig ファイルのサービス セクションでリストされているプライマリ ドメインをチェックしていなかった。リクエストは、セカンダリ ドメインに移っていた。 プライマリ ノードが利用できなくなると、bdmconfig.xml のコンフィグレーションに従って、サービス リクエストは、インポートされたサービス セクションのセカンダリ サーバにフェイルオーバした。 <T_DM_IMPORT ResourceName="TOUPPER" LocalAccessPoint="WLSDOM" RemoteAccessPointList="SIMPDOM,TESTDOM"> <TranTime>600</TranTime> < /T_DM_IMPORT> SIMPDOM が利用できなくなると、サービス リクエストは TESTDOM によって処理された。接続ポリシーが ON_DEMAND のときは、このような処理が行われてはならない。 分析の結果、接続ポリシーがチェックされていないことがわかった。 weblogic/wtc/gwt/TuxedoConnection.java のコードを修正することで、この問題を解決した。
CR095588	simpconv サンプルにおいて、Tuxedo クライアントが WebLogic Server に接続できなかった。クライアントは、WebLogic Server ではなく Tuxedo のローカル サーバにアクセスしていた。 パディングを探さないよう、Tuxedo クライアントのコードを修正した。さらに、dubb にリモート サービス セクションを追加し、bmdconfig.xml ファイルにエクスポート セクションを追加した。

CR092860	WTC を使って Tuxedo 8 Corba オブジェクトにアクセスすると、`JavaIDL Reader` スレッドのリークが発生した。簡単なサンプル アプリケーションを実行すると、WebLogic Server を実行する JVM で `JavaIDL Reader` スレッドが増加した。サーバをシャットダウンするまで、スレッドが破棄されなかった。 分析の結果、`finalize()` が正しいタイミングで呼び出されていなかった。`ORB.destroy()` の呼び出しを追加することで、この問題を解決した。
CR090137	WebLogic Server 6.1 SP02 では、WebLogic Server の Xalan エンジンが、一致しない `<META>` タグを含む無効な XML ファイルを生成していた。 apache/xalan のコードを修正し、この問題を解決した。
CR074963	FML テーブルを作成するとき、`mkfldclass[32]` は、各エントリをハッシュ テーブルにインスタンス化するためのメソッドを生成していた。大きな FML テーブル (4000 エントリ以上) の場合、このメソッド 1 つで JVM のサイズ制限を超えていた。 フィールド テーブルの動的ロード メカニズムを実装することで、この問題を解決した。

XML

変更要求番号	説明

CR091862	WebLogic 6.1 SP04 では、XML ドキュメントの DTD 部分を使うと、XML パーサに CPU やメモリを大量に使用させることができた。
	XML ドキュメントの DTD 部分を使うと、ドキュメントで名前付きのエンティティを定義できる (定義済みの <、> などは除く)。エンティティは、他のエンティティを使って定義できる (再帰指定は XML 1.0 では禁止されている)。エンティティは、参照されると、XML ドキュメント内で展開される。攻撃は、あるエンティティを定義して参照するという方法で行われる。このエンティティは、別のエンティティの 2 つのインスタンスを使って定義されていて、それがさらに別のエンティティの 2 つのインスタンスによって定義される、ということが繰り返される。この定義プロセスは必要な長さだけ繰り返すことができ、通常は、100 レベルのネストで十分である。100 番目のエンティティは、単に文字列として定義しなければならない。このような定義は、理論的には、最初のエンティティに 100 番目のエンティティの値を 2^99 (2 の 99 乗) 個連結したものが含まれているのと同じ影響を及ぼす。
	次に例を示す (DTD は、XML 宣言の後、XML ドキュメントの root 要素の前に置く)。
	`<!DOCTYPE root [`
	`<!ENTITY x100 "foobar">`
	`<!ENTITY x99 "&x100;&x100;">`
	`<!ENTITY x98 "&x99;&x99;">`
	`<!ENTITY x97 "&x98;&x98;">`
	`...`
	`<!ENTITY x3 "&x4;&x4;">`
	`<!ENTITY x2 "&x3;&x3;">`
	`<!ENTITY x1 "&x2;&x2;"> ]>`
	アプリケーションがドキュメント内にあるこの最初のエンティティを参照すると (&x1; という構文を使って)、ドキュメントのそれ以外の部分には問題がなくても、このエンティティを展開するために XML パーサが非常に大きな CPU 負荷またはメモリ負荷を必要とするため、結果として DoS 状態が発生する。
	コンフィグレーション可能なシステム プロパティ `weblogic.apache.xerces.maxentityrefs` を新しく追加することで、この問題に対処した。
	このプロパティの使用方法は、次のとおりである。
	`java -Dweblogic.apache.xerces.maxentityrefs=10000`
	http://dev2dev.bea.com/resourcelibrary/advisoriesnotifications/BEA02-23.jsp の「SECURITY ADVISORY (BEA02-23.01)」を参照すること。

WebLogic Server 6.1 サービス パック 4 のソリューション

以下の節では、WebLogic Server 6.1 サービス パック 4 で解決された問題について説明します。

- クラスローダ
- クラスタ
- コンソール
- EJB
- サンプル
- JDBC および jDriver
- JMS
- その他
- プラグイン
- サーブレット、JSP、Web アプリケーション
- セキュリティ
- システム管理

クラスローダ

変更要求番号	説明
077067	config.xml で PreferWebInfClasses プロパティを設定すると、リソース ファイルに対して正しく作用するようになった。このプロパティを設定すると、ファイルはユーザの Web アプリケーションから取得される。
077344	クラス ファイルが WEB-INF\lib フォルダの JAR ファイルに入っていると、java.lang.ClassNotFoundException: org.apache.commons.logging.impl.LogFactoryImpl が送出されていた。この問題を解決した。
079738	大きなメモリ構造への静的な参照を含む Web アプリケーションをデプロイまたはアンデプロイするときに発生する可能性のあるメモリ リークを解決した。
081377	デプロイメント時にコンテキスト クラス ローダを介してクラスをロードすると発生していた問題を解決した。
083752	クラスが InitialContext の作成を試み、java コマンド ラインが -Xbootclasspath 引数を使って weblogic.jar を指定すると、weblogic.i18ntools.L10nLookup.loadProps から NullPointerException が送出されていた。この問題を解決した。
084236	サンプル samples.examples.iiop.wls2wls.package-summary.html に、正しい URL 情報が含まれるようになった。

クラスタ

変更要求番号	説明
CR043366	サーバが開発モードで開始したとき、クラスタ アドレスに対してカンマ区切りの IP アドレスを指定できるようになった。
CR078454、CR081750	WebLogic Server 6.1 SP02 および SP03 では、WebLogic Server クラスタがセッション レプリケーション中にハングした。コンフィグレーションには、F5 ロード バランサ、Internet Information Server、および WebLogic Server ISAPI プラグインが含まれていた。 スレッド ダンプでは、RJVM サブシステムにおける今までに報告されたことのないパスが示された。 `ExecuteThread: '33' for queue: 'default'" daemon prio=5 tid=0x49edf0 nid=0x2b waiting for monitor entry [0xd527f000..0xd527fc68]^M at weblogic.rjvm.RJVMManager. findOrCreateRemoteInternal(RJVMManager.java:213)^M at weblogic.rjvm.RJVMManager.findOrCreate(RJVMManager.java:188)^M at weblogic.rjvm.RJVMFinder.findOrCreateRemoteServer (RJVMFinder.java:178)^M at weblogic.rjvm.RJVMFinder. findOrCreate(RJVMFinder.java:149)^M at weblogic.rjvm. ServerURL.findOrCreateRJVM(ServerURL.java:207)^M at weblogic.jndi.WLInitialContextFactoryDelegate.getInitialContex t(WLInitialContextFactoryDelegate.java:311)^M at weblogic.jndi.Environment.getContext(Environment.java:156)^M at weblogic.jndi.Environment.getInitialContext(Environment.java:139)^M at weblogic.servlet.internal. HttpServer.lookupROIDS(HttpServer.java:751)^M at...` 分析の結果、ISAPI プラグインは、リクエストをプライマリまたはセカンダリ以外のサーバに転送していたことがわかった。その後、リモートの `ROIDImpl` に対する呼び出しが行われた。 この問題は、WebLogic Server 6.1 SP04 でコードの修正により解決された。
CR078677	管理対象サーバの正常なシャットダウンの際に、`HttpClusterServlet` がフェイルオーバを実行するようになった。
CR081908	TCP レベルでセカンダリ サーバを利用できないときの HTTP セッションのレプリケーションに関する問題を解決した。これにより、明らかに TCP リクエストを受け取ることができず、クラスタ ビューから既に削除されているセカンダリ サーバにサーバが接続を試みても、長い遅延が発生しなくなった。

CR083532	サーバのシャットダウン シーケンスが途中でアボートする原因となっていた `weblogic.rmi.cluster.ReplicaAwareRemoteRef.getCurrentReplica` での Null ポインタ例外を解決した。
CR087870	高可用性マルチプールを使用した場合、初期容量を 0 にしてプールを作成し、ハードウェア障害が発生すると、Weblogic は正しくフェイルオーバするようになった。送出されていた `ResourceException` が、マルチプールがフェイルオーバする理由として解釈される種類の例外ではなかった。それが問題であった。

コンソール

変更要求番号	説明
057307	RDBMS レルムでユーザ パスワードを変更しようとすると送出されていた `NullPointerException` が送出されなくなった。
061975	自動キー生成を使用したときの `createColumnsSql` における問題を解決した。
073536	チェック ボックスをチェックすることにより、コンソールでオプションの相互 SSL を選択できるようになった。
074998	フォームベースの認証と複数のロケールを使用したときに認証エラーを受け取る問題を解決した。英語以外の言語を使用する資格は、`fileRealm.prorperties` ファイルには正常に追加できたが、サーブレットのフォームベースの認証を利用した認証には使用できなかった。サーブレットのフォームベースの認証を利用した認証にも、資格を使用できるようになった。
079130	管理対象サーバにデプロイされている EJB に対する JNDI ツリーをブラウズできるようになった。
079805	ソース サーバを対象とする Web アプリケーションが、複製されたサーバを対象として示されるようになった。
082578	[サーバ	コンフィグレーション	HTTP] タブに WAP 対応フラグが追加された。
083107	コンソールが、Mozilla 1.0 ブラウザをサポートするようになった。
083330	コンソールでクラスタをマルチプールの対象として設定できるようになった。
083377	`WebLogicCluster` で各クラスタ ノードに DNS 名を指定した場合、`HttpClusterServlet` は、静的なサーバ リストへの参照を使って HTTP リクエストの固定した関係を維持できるようになった。
083578	サーバのプロトコル タブの最大メッセージ サイズのテキスト フィールド サイズが増やされた。
084114	コンソールの [`mydomain`	サーバ	`myserver`	デプロイメント	Web アプリケーション] タブの [すべてのアクティブな Web アプリケーションのモニタ] リンクを使って Web アプリケーションをモニタしようとすると、例外が送出されていた。この問題を解決した。

086652	コンソールで新しいドメインを作成すると、コンソールの左ペインで [ファイル レルム] タブが機能するようになった。

EJB

変更要求番号	説明
069899	両方のテーブルに外部キーと同じカラムがある場合でも、多対多 CMR が SQL 例外を送出しなくなった。
076112	`weblogic-ejb-jar.xml` ファイルで `<max-beans-in-free-pool>` パラメータを指定してステートレス セッション Bean をデプロイした場合、ステートレス セッション Bean に対して同時呼び出しを行っても、この値が考慮されなかった。WebLogic Server は、ステートレス セッション Bean のインスタンスを、このパラメータで指定されている値より多く新たに作成していた。この問題を解決した。
076597	`<max-beans-in-free-pool>` を 0 にしてステートレス EJB をアンデプロイしても、`ejbRemove()` メソッドが呼び出されていなかった。この問題を解決した。
077049	アプリケーションが EJB メソッドのパラメータとして動的プロキシを使用した場合、パラメータ インタフェースがサーバ クラスパスに存在していても、リモート クライアントがインタフェースを使用できるようになった。
078456	一部の EJB サンプルのドキュメントに記述されているクライアントとサーバの出力が、テキストが折り返されていなかったため読みづらかった。テキストを折り返して読みやすくした。
079745	`weblogic-ejb-jar.xml` ファイルに別の XML ファイルが含まれていても、EJB のデプロイが失敗しなくなった。
080569	ECperf を実行した後、`EJBCacheRuntime` 統計情報で、1 つのステートフル セッション Bean (CartSes) の `CachedBeansCurrentCount` に不正な（負の）値が表示されていた。この問題を解決した。
081355	EJB 2.0 の JAR ファイルで関係を移動しても、EJB が `NullPointerException` を送出しなくなった。
081807	ホームを JNDI ツリーにバインドする前に、EJB デプロイヤがコンテキスト オブジェクトで `replicateBindings` を「True」に固定的に設定しているので、ユーザが `weblogic-ejb-jar.xml` ファイルで `home-is-clusterable` を「False」に設定していても、クラスタ内の残りのノードに対する通知が試みられて、`NameAlreadyBoundException` が送出されていた。この問題を解決した。

081817	`weblogic-ejb-jar.xml` ファイルで Bean レベルの `dispatchPolicy` を設定するためのオプションが提供された。
081991	キューからメッセージを取得するために「inhibited」を選択してから「allowed」を選択した後で、メッセージ駆動型 Bean が MQSeries への再接続に失敗しなくなった。
082029	接続プールに対する `openString` プロパティの使用時に、XA ドライバを使用するデータベースに EJB を使って接続するときの問題を解決した。
082451	WebLogic Server ではない JMS プロバイダを使用するメッセージ駆動型 Bean に対するサードパーティの JAR ファイルに対し、`ClassNotFoundException` が送出されなくなった。
083050	`ejbRemove()` から `context.getCallerPrincipal()` を呼び出しても、例外が送出されなくなった。
083098	キューを閉じると、MQSeries に対する接続数が増える。
083689	SQLServer を使用し自動キー生成を行う CMP エンティティ Bean で `javax.ejb.NoSuchEntityException` が発生しなくなった。
083896	`max-beans-in-cache` を 10 に設定した後、クライアントを実行し、同じ主キーで Bean に 200 回アクセスして、キャッシュに 200 個の Bean を取得した場合、`concurrency-strategy` が排他に設定されていると、データベース キャッシュが無視されていた。この問題を解決した。
083896	プロパティ `max-beans-in-cache` が無視されていた問題を解決した。
084224	Read-mostly パターンが使用されていると、コンテナが `invalidate()` メソッドを呼び出していなかった。
084978	Administration Console から EJBC パラメータを追加および変更できるようになった。これにより、たとえば、スタブのコンパイルに割り当てられるメモリの量を変更できる。
085320	ステートレス セッション Bean とサーバに関する問題を解決した。Administration Console に表示されるアイドル Bean カウントが誤ってインクリメントされ続けていた。
085659	マシンがネットワークから切断されたときの EJB に関する問題と遅延を解決した。

サンプル

変更要求番号	説明
053356	サンプル samples/examples/wtc/corba/simpappcns に UNIX スクリプト run.sh がなく、UNIX マシンで動作しなかった。 ubbdomain、dom1config、run.sh の各ファイルを用意し、Tuxedo 側をビルドする方法についての説明を javadoc に加えることで、この問題を解決した。この解決策により、マルチプラットフォーム / マシンのサポートが提供される。古いビルド スクリプトは、単一マシン環境を想定していた。
079543	BEAHOME の名前にスペースが含まれていると、サンプル サーバが起動しなかった。この問題を解決した。
084236	サンプルの samples.examples.iiop.wls2wls.package-summary.html に正しくない URL 情報が含まれている。
084903	WTC simpappcns サンプルが、単独のマシンだけではなく、ネットワーク経由でも動作するように修正された。

JDBC および jDriver

変更要求番号	説明
077974	Oracle 8.1.7 OCI における OCI のバグが解決されたため、Solaris に対する OS 認証が追加された。
080487	DBMS のロールバックが失敗した場合でも、JTS ドライバでプール接続のリークが発生しなくなった。
CR080931	Oracle jDriver に関する問題を解決した。JDBC 接続に対して `weblogic.oci.min_bind_size` プロパティを設定すると、`CallableStatement` を使用した後に同じ接続で実行した `SQL UPDATE` prepared statement が失敗した。
082438	静的に定義されたプールには prepared statement のキャッシュ サイズを定義するためのプロパティが反映されているが、Console、`weblogic.Admin`、または API を使って動的に作成されたプールには反映できなかった。この問題を解決した。
082484	`PreparedStatement` で 511 より多くのパラメータが設定された場合でも、JDriver for Oracle で `ArrayIndexOutOfBoundsException` が発生しなくなった。
086557	Windows、HP、および Solaris 用の jDrivers for Oracle 9.2 を追加し、それに合わせて 8.1.7 と 9.0.1 のドライバを更新した。
087005	Oracle 9.2.0 用の Thin ドライバが追加された。
087803	Oracle 9.0.1 を使用する JMSServer 用にコンフィグレーションされた JDBC ストアを作成する際の問題を解決した。更新された 9.0.1 用 Thin ドライバを提供することで、この問題を解決した。
088156	EJB 永続性マネージャが原因で発生する可能性があった JDBC のデッドロック状態を解決した。

JMS

変更要求番号	説明
063743	メッセージ駆動型 Bean がオブジェクトのメッセージに応答しなくなる問題を解決した。
079547	クラスタ環境にデプロイされた JMS サーバに対する恒久サブスクライバを作成する際の問題の原因となるデッドロックを解決した。
081525	クラスタで `TopicSubscriber.close()` を呼び出しても、`weblogic.management.NoAccessRuntimeException` が送出されなくなった。
080668	JMS サーバ用のリスナを設定するクラスタ環境では、サーバのいずれかのノードが、直ちにではなく、しばらくしてから起動すると、`NoSuchObjectException` が送出されていた。この問題を解決した。
082298	恒久 JMS サブスクライバの数が正しくなかった。コンシューマが恒久サブスクライバに接続するたびに、余分なインクリメントが行われていた。サブスクライバが作成された時点で数がインクリメントされ、削除された時点でデクリメントされるようになった。
082438	サービス パック 4 より前は、動的に作成された JDBC 接続プールに対して `STATEMENT_CACHE_SIZE` パラメータが実装されていなかった。SP04 ではこの問題が解決されたので、動的接続プールは、Statement Cache を使って prepared statement と callable statement をキャッシュし、再利用できるようになった。
083290	プライマリ サーバが停止してから再起動したときの、管理対象サーバにおける JMS の回復に関する問題を解決した。
083503	管理対象サーバを起動および停止したときの、JMS とメッセージ駆動型 Bean の回復に関する問題を解決した。
084175	`MessagingBridge` を通してメッセージを送信し、サーバの再起動によって中断されると、メッセージが失われていた。`MessagingBridge` によってメッセージが処理されている最中に（MQ Queue から WebLogic JMS Queue に処理されているメッセージ）、WebLogic Server が停止して再起動すると、場合によっては、一部のメッセージが失われることがあった。失われるメッセージの数は、`MessagingBridge` の `BatchSize` に対応していた。この問題を解決した。

084182	`exceptionListener` を Null に設定しても、`setExceptionListener` メソッドは `NullPointerException` を送出しなくなった。
CR084374	メッセージング ブリッジは、いずれかのサーバに対する初期コンテキストの取得に失敗すると、パスワードを含むエラー メッセージを表示していた。 パスワードをクリア テキストとして表示しないよう `JMSBaseConnection.java` を修正し、この問題を解決した。 http://dev2dev.bea.com/resourcelibrary/advisoriesnotifications/BEA03-24.jsp の「SECURITY ADVISORY (BEA03-24.00)」を参照すること。
086487	83503 を参照。
CR093060	前記の「CR084374」と同じ。

その他

変更要求番号	説明
049340	`ComponentMBeans` に対する `DeploymentOrder` を `application.xml` ファイルでの順序に基づいて初期設定し、初期デプロイメントと同じ順序で再デプロイメントできるようになった。
053054	サーバがトランザクション コンテキストをコピーしなかった問題を解決した。実装クラスを JNDI ツリーにバインドした後でクライアントが新しい JVM を生成すると rjvm エラーが発生し、新しい JVM はメソッドで RMI 呼び出しを行うことができなかった。
053957、071626、075394、076265、078527、079652、079655、082693、082830	このサービス パックでは、WebLogic による RJVM の処理について、以下に示すようなさまざまな改善が行われた。 WebLogic Server は、RJVM へのメッセージ送信を試みる前に、ローカルとリモートの Java 仮想マシン上で稼働するサーバ間に有効な接続が完全に確立されていることを確認するようになった。 ローカルとリモートの Java 仮想マシン上で稼働するサーバ間に有効な接続を確立するときに発生する可能性のあったデッドロックを解決した。 これまでは、RJVM が待機状態に入る前に接続成功の通知が発行されて、RJVM が接続の確立を待ちながら無限に待機状態になる場合があった。この問題を修正し、待機状態に入る前に接続成功の通知があっても、通知を検出できるようになった。 複数のスレッドが同時に RJVM への接続を確立しようとした場合、その結果は、接続の確立を実際に管理する最初のスレッドの成功または失敗に依存するようになった。 これまでは、接続に成功した最初のスレッドによって確立された接続を使うのではなく、各スレッドが、常に、個別に接続の確立を試みていた。 異なるコンピュータ上で稼働しピアを構成する複数の WebLogic Server が、それまで接続を確立していたピア サーバの停止を正しく検出し、壊れた接続を正しくクリーンアップするようになった。 OutputStream の取得に失敗すると、Remote Method Invocation (RMI) レイヤがフェイルオーバするための適切な例外を受け取るようになった。 これまでは、2 つのピア サーバが接続を確立しようとして失敗すると、RMI レイヤに対して例外が送出されず、不適切なフェイルオーバ、または適切なフェイルオーバの遅延が発生していた。 T3JVMConnection から RJVM に対する ConnectionManager への相互呼び出しの間で発生する可能性のあったデッドロックを解決した。

056403、063910、076409、076903、078288、078918、079599、080292、080744、081010、081116、081856、082700、083623、086362	以下に示すように、muxer に関するいくつかの問題を解決し、改善を行った。 ソケットにマップするすべての muxer （Java とネイティブ コード）の集合が、固有のキーを持つようになった。これにより、データの破損、不正なポインタ参照、WebLogic Server のハングとクラッシュなどの、さまざまな問題が解決された。 muxer のデータ構造と状態マシンが正しく同期するようになった。これにより、ロックの順序の衝突によって発生していた一部のデッドロックが解決された。 muxer とその上のプロトコル層の間の呼び出しスタックが双方向になり、ソケットまたは接続のクローズをスタックのどのポイントからでも実行できるようになった。デッドロックを防ぐための手段が追加されている。 muxer における状態マシンの遷移が修正され、ソケットが CLOSE_WAIT 状態のままになることがなくなった。
064301	アプリケーション サーバが突然終了されると、高い確率で、未解決の分散トランザクションが発生する。WebLogic Server 6.1 SP01 では、未解決の分散トランザクションは存在し続け、DBA に手動で介入することで解決する必要があった。WebLogic Server 6.1 サービス パック 4 では、アプリケーション サーバを再起動すると、トランザクション ログ ファイルが再スキャンされて、回復処理が開始される。
071510	`NodeManagerMBean` から JMX を使ってサーバを起動できなかった。WebLogic Server 6.1 サービス パック 4 では、JMX クライアントを介して `ManagedServers` を起動および停止するための新しいメソッドが、`ServerMBean` に追加された。
072188	管理サーバが、`NodeManager` を指定して確立されたネットワーク接続を、タスクの完了後に閉じていなかった。この問題を解決した。
073023	ノード マネージャは、リモート サーバの起動を試みたときに、ランダムに `OutputHandler` エラーを送出しなくなった。
074844	動的プロキシを使用したときのアサーション エラーの原因となっていた問題を解決した。
075406、076002	大きな負荷がかかっている状態で `NullPointer` 例外と `ClassCast` 例外が発生する問題を解決した。また、NT muxer に対するいくつかの改善も行った。
076605	「`<ExecuteThread: '12' for queue: 'default'> <> <> <000000> <No IORecord present for fd>`」というメッセージが表示される原因となっていた問題を解決した。

076705	送り先がリモートの場合に、メッセージ駆動型 Bean のデプロイメントが失敗し、ゲストに対して `NoAccessRuntimeException` が送信されることがなくなった。
076997	MBean を使ってクラスタに EAR コンポーネントをデプロイしたとき、サーバを再起動しなくてもすべての管理対象サーバで JNDI のバインドが表示されるようになった。
077248	ネイティブ IO がオフになっている場合に IIOP を介して EJB に接続するときの問題を解決した。
077831	`AppletArchiver` ユーティリティを使ってクライアント JAR ファイルを作成し、問題となっているアプレットが `javax.swing.JApplet` クラスをコードの中で使用した場合、`AppletViewer` でエラーが発生していた。この問題を解決した。
077919	`EJBHandle` で `ObjectOutputStream.writeObject()` を呼び出した後、パッチ CR064447_61sp2.jar を使うとログ ファイルでエラーが発生していた。ログでの `EJBObject` の出力が、`LocalServerRef` から `LeasedRemoteRef` に変わっていた。この問題を解決した。
078431	MuxableSocketHTTP と PosixSocketMuxer の間のデッドロックの原因になっていた問題を解決した。
078952	EJB のリモート メソッドからのシリアライズ可能なオブジェクトで null 値が返されても、`org.omg.CORBA.MARSHAL` 例外が送出されなくなった。
079220	クラスタ環境では、JMS を実行すると、クライアントが「`Could not find RJVM for $Proxy68.. (IllegalArgumentException)`」という例外を受け取っていた。クライアントは、管理対象サーバの 1 つにアクセスして利用していた。この問題を解決した。
079672	`WarClassFinder.getSource` を呼び出すと VM がクラッシュする可能性がある。Sun は、この問題への対処を JDK の 1.4.1_02 および 1.3.1_07 で行っている。Sun のバグについては、http://developer.java.sun.com/developer/bugParade/bugs/4369396.html を参照すること。
080177	同じ EJB を繰り返しデプロイおよび削除しても、`weblogic.deploy` でメモリリークが発生しなくなった。

080740	Windows サービスとして動作しているサーバがシャットダウンすると削除されるファイルを Web アプリケーションが作成した場合、コマンドラインから (`java weblogic.Admin SHUTDOWN` コマンドを使って)、または Administration Console からサーバをシャットダウンすると、ファイルは意図したとおりに削除された。しかし、Windows の [スタート] メニューから [設定	コントロール パネル	サービス] を開き、サービスの [停止] ボタンをクリックしてサーバをシャットダウンすると、ファイルが削除されなかった。 この機能を実装した。
080895	スタンドアロンの java クライアントがステートレス セッション EJB を呼び出したとき、約 500,000 回繰り返すと、クライアント サイドで次の例外が送出されていた。 `Exception in thread "main" java.lang.ClassCastException: Assigning instance of class java.io.ObjectStreamClass to field weblogic.rmi.internal.ClientMethodDescriptor#signature at java.io.ObjectInputStream.inputClassFields(ObjectInputStream.java:2271)` この問題を解決した。
080901	Windows NT におけるソケット マルチプレクサの問題のため、DoS 攻撃によるクラッシュに対して弱かった。 http://dev2dev.bea.com/resourcelibrary/advisoriesnotifications/BEA02-19.jsp の「SECURITY ADVISORY (BEA02-19.00)」を参照すること。
080929	ワイルドカードを使用した場合でも、`weblogic.refresh` がファイルを更新できるようになった。
081311	DTD の `SAXParseException` の重要度が、`<Info>` ではなく `<Warn>` になった。
081404	WebLogic Server 6.1 サービス パック 4 には、メッセージング ブリッジ用の JMS ブリッジ リソース コネクタ (`jms-xa-adp.rar`、`jms-notran-adp.rar`、`jms-notan-adp51.jar`) が収められた。
081568	SNMP エージェントが、サーブレットの `ServletRuntime` をモニタできるようになった。
081732	WebLogic Server 6.1 SP03 では、Sybase データベースを使用した場合、サーバを起動しようとすると例外が送出されていた。この問題を解決した。
081765	SSL に、途中でソケットを閉じて muxer のエラーが発生する問題があった。

081853	JDK が混在する環境で特に顕著であった、RMI-IIOP および複雑なオブジェクトのマーシャリングに関するいくつかの複雑な問題を解決した。
081868	WebLogic Server 6.1 サービス パック 2 では、`java weblogic.Admin SHUTDOWN` を使って管理サーバをシャットダウンすると、SNMP エージェントが `serverShutDown` トラップを生成していなかった。サーバの起動時には、`serverStart` トラップを生成できた。SNMP エージェントは、管理サーバがシャットダウンすると自己シャットダウン トラップを発行するようになった。

CR081870	WebLogic Server による Xalan の配布キットでは、次のように XSLT の出力方法を HTML (必須) に設定すると、アンカー タグの HREF 属性の中のスペースが、「%20」にエンコードされていた。
	`<xsl:output method="html"/>`
	例 :
	`<a href="javascript: var win = openWindow()">Link</a>`
	上のような記述に対して、下のような誤った出力が生成されていた。
	`<ahref="javascript:%20var%20win%20=%20openWindow()">Link</a>`
	Netscape は「%20」をスペースと解釈できず、リンクが壊れていた。出力タイプを「XML」に変更すると (この場合の対応策としては適切ではない)、属性は正しい結果を出力した。
	vanilla Xalan の配布キット (つまり BEA 拡張バージョンではないもの) は、同じ入力に対して正しい変換を行う。
	Xalan の WebLogic Server による配布キットは、HREF 属性に対して不適切な URL エンコーディングを行った。スペースが許されない通常の URL (つまり http://somehost/...) に対しては問題なく動作するが、JavaScript の URL (つまり javascript: doSomething()) も有効であり、これはエンコーディングされてはならない。XSLT トランスフォーマの場合、URL を明示的にエンコードする方法はスタイルシートに既にあるので、URL に対して何も行ってはならない。
	この問題は、WebLogic Server 6.1 SP2、SP3、および SP4 で発生した。
	URL をエンコードしないよう XSLT トランスフォーマを修正することで、この問題を解決した。

082004	WebLogic Server 6.1 サービス パック 2 では、新しい `InitialContext()` を取得する際に、`CR073910_61sp2.jar` パッチによってメモリ リークが発生していた。
082263	`weblogic.deploy` ツールが、Windows 2000 から Solaris 上で稼働するサーバに対して、Web アプリケーションの `.WAR` ファイルを更新できるようになった。

082533	Xerces Parser を使用したとき、デバッグ メッセージが `std.out` に書き込まれていた。このメッセージは削除された。
082693	WebLogic Server 6.1 サービス パック 1 および 2 では、旧世代のスタブからの呼び出しを受け取ったサーバは、クライアントに `peerGone` を正しく送信し、エラーをログに記録していた。サービス パック 3 では、サーバは `NullPointeException` を出力し、クライアントはタイムアウトするまで待機していた。サービス パック 4 では、これを以前の正しい動作に戻した。
083102	`J2EE12OnlyModeEnabled="true"` を指定して 2 つの EJB（EJB 2.0 仕様に準拠したもの）をデプロイすると、サーバは Bean に対して警告メッセージを 2 つ送出し、`DistributedManagementException` でクラッシュしていた。この問題を解決した。
083485	t3 クライアントを呼びだしても、クライアントはデフォルト プロトコルとして t3 を使用する `JavaSocketMuxer` でハングしなくなった。
083652	WebLogic Server 6.1 サービス パック 3 では、アプリケーションが起動時にアンデプロイされた後、2 回再デプロイされていた。この問題を解決した。
083654	Microsoft WAS ツールからリクエストが連続して発行されたときのパフォーマンスが、WebLogic Server 6.1 サービス パック 3 より改善された。英語圏以外のユーザは、5-30 ページの「089868」の関連情報を参照すること。
083721	スタートアップ クラスの `FailureIsFatal` オプションが、Solaris 環境での実行時例外を抑止しなくなった。
083920	再起動した管理サーバが、管理対象サーバが稼働していることを発見したときの起動トラップが抑止されて、管理サーバの再起動時に冗長なトラップが送信されなくなった。
084127	プロダクション モードでは、自動デプロイメントがオフになっていても、WebLogic Server は `config.xml` ファイルでコンフィグレーションされていない一部のアプリケーション ファイルの検証を行っていた。この問題を解決した。
084622	ZAC を使用しても、`getInitialContext()` を呼び出したときにアサーション エラーが発生しなくなった。
085463	ネットワーク プラグを抜いたときの JNDI ルックアップのフェイルオーバに、長い時間がかからなくなった。

085914	管理対象サーバが停止した後で、SNMP リクエストがタイムアウトしていた。管理対象サーバが停止しても、SNMP エージェントが `getnext` リクエストにしばらく応答できない状態になることはなくなった。
085979	カスタム Mbean が定義されているときに `MBeanHome.getAllMBeans()` が不必要な例外を送出する問題を解決した。
086108	パフォーマンスが改善された。ListenThread が消費するメモリが、サービス パック 3 より少なくなった。
086248	ボックス化された RMI シーケンスに対するコンポーネント タイプが、先行して 2 回宣言されていたが、1 回は正しく、もう 1 回は valuetype のシーケンスに対しては正しくないタイプであるインタレースであった。
086350	例外のクラス名の最後が大文字の「I」である場合に、その名前から生成される IDL が壊れる問題を解決した。

CR086362	大きな負荷がかかると SocketException: Connection timed out エラーが発生していた。スタック トレースは以下のとおりである。 `####<Sep 18, 2002 9:37:07 AM EDT> <Error> <Posix Performance Pack> <lpsapp01> <LPSAPP01_C> <ExecuteThread: '34' for queue: 'default'> <> <> <000000> <IOException on socket: 'weblogic.rjvm.t3.T3JVMConnection@67cf01', fd: '1046'> java.net.SocketException: Connection timed out: Connection timed out Start server side stack trace: java.net.SocketException: Connection timed out: Connection timed out at java.net.SocketInputStream.socketRead(Native Method) at java.net.SocketInputStream.read(SocketInputStream.java:86) at weblogic.socket.PosixSocketMuxer.readBytesProblem(PosixSocketMuxer.java:824) at weblogic.socket.PosixSocketMuxer.deliverGoodNews(PosixSocketMuxer.java:712) at weblogic.socket.PosixSocketMuxer.processSockets(PosixSocketMuxer.java:637) at weblogic.socket.SocketReaderRequest.execute(SocketReaderRequest.java:24) at weblogic.kernel.ExecuteThread.execute(ExecuteThread.java:139) at weblogic.kernel.ExecuteThread.run(ExecuteThread.java:120) End server side stack trace`  また、サーバ インスタンスが、CLOSE_WAIT 状態のソケットでハングしていた。スレッド ダンプは以下のとおりである。  `"ExecuteThread: '3' for queue: 'default'" daemon prio=5 tid=0x7f79a8 nid=0x10 waiting for monitor entry [0xa6101000..0xa6101a40] at weblogic.socket.PosixSocketMuxer.read(PosixSocketMuxer.java:542) at weblogic.socket.JVMSocketManager.accept(JVMSocketManager.java:185) at weblogic.t3.srvr.ListenThread$RJVMListenRequest.execute(ListenThread.java:594) at weblogic.kernel.ExecuteThread.execute(ExecuteThread.java:139) at weblogic.kernel.ExecuteThread.run(ExecuteThread.java:120)` スレッド ダンプを分析した結果、ネイティブな AddFdToPollSet のスレッドが PollSet Loopback FD への書き込みを終了せず、POLLSET_LOCK を正しく解放していないことがわかった。 パイプに書き込んだバイト数を追跡し、パイプに書き込んでも大丈夫であることをスレッド リーダーが示すまでパイプへの以降の書き込みを待つことで (代わりに配列を使って)、この問題を解決した。
086761	Weblogic 管理ツールが正しく実行キューを作成し、サーバ起動時に機能するようになった。
087265	WebLogic のバージョン間の相互運用性を保つため、互換性のないいくつかの serialVersionUID を修正した。

088372	`weblogic.security.acl.TTLCache.cleanup` での `ArrayIndexOutOfBoundsException` などの複数の例外が発生していた負荷状況下での問題を解決した。
089044	Administration Console をロードしても、ブラウザ /OS の互換性チェックが行われなくなった。

プラグイン

変更要求番号	説明
074128	クラスタにおいて、Internet Explorer で 10053 番のソケット エラー メッセージが表示された後、プロキシ プラグインが誤ってフェイルオーバしていた。プライマリ サーバは正常に動作していた。問題は次のコンフィグレーションで発生した。 ■ WebLogic Server 6.1.0 と SP02 ■ IIS プラグイン ■ Windows 2000 SP2 ■ IIS 5.0 10053 番のエラーは WinSock エラーであり、winsock.h では WSAECONNABORTED と定義されている。このエラーは、ISAPI 拡張機能がデータの返送を終了する前にページを再ロードすると発生する。ページを再ロードすると、ブラウザは現在の接続を閉じた後、新しい接続を作成してリクエストを再送信する。拡張機能がデータ送信を終了する前にブラウザの [中止] ボタンをクリックしたときも、同様の状況が発生する場合がある。 これは正常なイベントであり、ブラウザはそれ以上リスンしない。エラーの後、プラグインは、単にクリーンアップを行って HttpExtensionProc から戻らなければならない。 IIS プラグインを変更することで、不適切なフェイルオーバの発生を解決した。POST データの読み取りでタイムアウトが発生した場合、sendRequst フェーズの間に処理が障害になってもフェイルオーバが発生することはなくなった。

076936	ISAPI 環境下で、次のように `WlForwardPath` と共に `PathTrim` で複数レベルのパスを指定すると、ISAPI プラグインが指定されたパスをトリムできなかった。
	`WlForwardPath=/`
	`PathTrim=/path/to/weblogic`
	この問題を解決した。

078713	`iisforward.ini` ファイルに新しいパラメータ セット `FilterPriorityLevel` が追加された。このパラメータの値は 0、1、または 2 で、それぞれ、フィルタの優先度が低、中、高であることを意味する。デフォルト値は 2 である。仮想ホストを使ってこのプロパティを設定できない場合は、`iisforward.ini` ファイルを使用する。

079186	HTTP 1.1 仕様に従って、すべてのプラグインが複数行にわたるヘッダ プロパティを解析するようになった。
079683	`mod_wl_ssl.so` を使うと、Chunked Transfer Encoding HEX の値がブラウザに表示されていた。これは、Weblogic で `jsp:include` を使用したときにだけ発生した。`mod_wl.so` は正しく機能したが、ブラウザがページを完全に表示するまでに長い遅延があった。この問題を解決した。
079973	WebLogic Server 6.1 サービス パック 4 では、`Idempotent` が OFF の場合はフェイルオーバが行われず、`ConnectTimeoutSecs=0` の場合はコア ダンプが行われない。iPlanet のコンフィグレーションで `ConnecyTimeoutSecs` が 0 に設定されていると、リトライは試みられない。以前は、デフォルトの設定 (10) または 0 であるとリトライが試みられていた。さらに、`Idempotent` が OFF に設定されているとリトライが 2 回行われるのに対し、ON に設定されていると 5 回行われていた。
080219	すべての WebLogic Server インスタンスをシャットダウンしても、iPlanet のコア ダンプが発生しなくなった。
080382	NSAPI プラグインを使用しても、WebLogic Server のエラー ログに予期しないエラー メッセージが記録されなくなった。
080746	ネイティブ Web サーバ プラグインでの 64 ビット プラットフォームがサポートされるようになった。
081185	ファイルのアップロードをキャンセルしても、CPU の負荷が増えなくなった。
081493	HPUX-11 の iPLanet がクラッシュする原因となっていた問題を解決した。
082093	WebLogic Server が NSAPI プラグインを使用する iPlanet プロキシを介してアクセスされていて、QALoad を使って同時に多数のクライアント接続が生成されても、高負荷下の Web サーバに対し、プロキシ プラグインによって余分な呼び出しが行われることがなくなった。
082096 および 083386	`Idempotent=OFF` であっても、IIS プロキシ プラグインは、`HungServerRecoverSecs` の後ですべての POST リクエストに対して 1 回のリトライ（2 リクエスト = 1 オリジナル + 1 リトライ）を試みなくなった。
082113	プラグインは、クライアントからのデータの読み取り中にエラーを受け取っても、バックエンドの WebLogic Server に不完全なデータ（リクエスト）を送信しなくなった。

082206	特定のページを WebLogic への転送から除外し、IPlanet のフロントエンドで処理させることができるようになった。定義した式に一致する特定のリクエストを除外するには、`obj.conf` に `WLExcludePathOrMimeType` を追加する（`WLExcludePathOrMimeType="/test,/*.html"` など）。
083174	最新の IIS プラグインを使用すると、セッションに新しい値を追加したときにフェイルオーバが機能しなかった。
083643	NSAPI のパフォーマンスの問題を解決した。`netbuf_getc` 関数は非常に遅く、POST データ中の「\0」と「0」を区別できなかった。
083558	最新の Apache である Apache 2.0.42 の動作が確認された。

サーブレット、JSP、Web アプリケーション

変更要求番号	説明
042655	拡張ログ フォーマットで access.log を使用した場合、HTTP リクエストの実行時刻が GMT 時刻ではなくローカル時刻で記録されることがなくなった。
053974	access.log の名前をコンフィグレーションできるだけでなく、管理しやすいようにログのローテーション規則も選択できるようになった。
058389	javax.servlet.http.HttpServletRequestWrapper に対して呼び出された getInputStream() が、null を返さなくなった。
063630	Web アプリケーションのデプロイが失敗した場合、クラスタに他のアプリケーションをデプロイできなかった。この問題を解決した。
065967	<type> が指定されていて、タイプが配列の場合でも、taglibs が失敗しなくなった。Weblogic Server は、タイプが実際には配列の時には、タイプを文字列として扱わなくなった。
068577	JSP 仕様 1.2 によれば、すべてのタグ ハンドラは DO_END_TAG によって解放されなければならないが、生成されるコードでは、<jsp:forward> に対して _releaseTags() メソッドが呼び出されていなかった。この問題を解決した。
069731	Web アプリケーションで JSP を使用すると、JSP から作成される .java ファイルは _tmp_war （Web アプリケーションのルートの下）に置かれる。_tmp_war ディレクトリがまだ存在していない場合は、Web アプリケーションのデプロイメント時に（サーバを起動した後の初期化の中で）作成される。サーバがルートとして実行されている場合は、_tmp_war ディレクトリはルートによって作成されて所有される。以前は、初期化の最後に nonPrivUser を使って特権のないユーザに切り替えたため、JSP サーブレットと WebLogic は _tmp_war ディレクトリにアクセスできなくなっていた。この問題を解決した。
071513	WebLogic Server は、Web アプリケーションの中で weblogic.xml ファイルを検証するようになった。
073780	WebLogic Server 6.1 サービス パック 2 および 3 つのパッチ CR065452A_610sp2.jar、CR063457A_610sp2.jar、および CR063829_610sp2.jar を使用したときよりも、JSP タグのパフォーマンスが改善された。

073791	サーブレットのコードから `ServletOutputStreamImpl.write` を呼び出した後に `ChunkUtils.getEnd` メソッドで `NullPoinerException` が発生していた問題を解決した。
074265	ServerMBean に新しい属性 `MaxBackoffBetweenFailures` を追加した。デフォルトは 10 秒である。サーバのリスン スレッドがソケット接続の受け付けをまだ試みている場合に、「Failed to listen on port (ポートでのリスンに失敗)」を送信する間隔を変更するには、この属性を設定する。
074940	`weblogic-tags.jar` の `cache` タグの `size` 属性を使用しても、`NullPointerException` が送出されなくなった。
075471	WebLogic Server サービス パック 4 では、`JRun` タグ ライブラリが機能するようになった。
075721	`TimeExpression` の `beanName` を使用する `jsp:useBean` タグが、正しく解釈されるようになった。

076375	`weblogic.log` および `access.log` (HTTP ログ) は、ログ ファイル名のコンフィグレーションに基づいて名前が変更されるようになった。ログ ファイル名で「%」記号を使用すると、ログ ファイルがローテーションされる時の日付 / 時刻に基づいてログ ファイルの名前を変更するロジックが実行される。現在ログが記録されているファイルは、「%」記号（ある場合）が取り除かれたファイル名になる。次は、有効な形式の例である。
	`weblogic_%yyyyMMdd%_%HH%_%mm%.log`
	`foobar_%yyyy%-%MM%-%dd%_%HH%_%mm%.log`
	注意： SimpleDateFormat では、大文字の「Y」は何も表さない。年を表すには、小文字の「y」を使用する。無効な形式を指定すると、WebLogic Server のデフォルトのログ ファイル名が使用される。パーセント記号 (%) で囲まれた文字列は、SimpleDateFormat に基づいてデコードされる。この形式はファイル名に対して適用され、ディレクトリ名には適用されない。

077018	リクエスト / 応答ラッパを使用したときの `ServletAuthentication` の `ClassCastException` が送出されなくなった。
077306	一時的なパッチを使用しても、インクルードされている（コンパイル済みの）JSP クラスが再コンパイルされなくなった。

077422	CookieParser.cookieToString が、カンマ (,) を含む SimpleDateFormat の expires クッキーを作成していた。expires の日付の中にカンマがあるため、リクエストに入れて返されたクッキーが誤って解釈されていた。カンマは、クッキーの区切り文字として解釈され、expires の日付の解釈が中断していた。この問題を解決した。
077888	setCharacterEncoding は、UnsupportedEncodingException を送出するようになった。
078090	JSP 仕様によれば、リクエスト パラメータで jsp_precompile が使用されている場合には、コンテナは、JSP をコンパイルしなければならないが（コンパイルされていない場合）、JPS を表示してはならない。WebLogic Server は、クエリ文字列を解析して jsp_precompile を検出するので、ブラウザにページを表示していなかった。
078262	ServletContext に getResourcePaths(String) が追加された。
078325	文字セット値の前後に余分な引用符がある場合に UnsupportedEncodingException が送出される原因となっていた問題を解決した。
078698	access.log ファイルは、ログのサイズが 2MB を超えるとローテーションするようになった。
079582	WebLogic Server 6.1 サービス パック 3 では、データ同期の最中に SocketException が送出されていた。この問題を解決した。
079767	アプリケーションまたはコンポーネントの再デプロイメントのたびに、EAR ファイル、WAR ファイル、または JAR ファイルを WebLogic Server に再デプロイしても、管理対象サーバの .wlnotdelete ディレクトリのサイズが増えなくなった。
079892	CGI スクリプトの実行中に停止ボタンをクリックしても、CPU の使用率が 100% のままになることがなくなった。
080090	リダイレクトの際に、ロケーション ヘッダは、次のパラメータが指定されている場合は常にそれを使用するようになった。 FrontendHTTPPort FrontEndHTTPSPort FrontEndHost

080384	ただ 1 つの応答でステータス コード ヘッダにスクランブルがかけられている場合であっても、HttpClusterServlet はサーバを不良 (bad) とマークしていた。この問題を解決した。
080751	サーブレットの例外を拡張するカスタム例外がコンフィグレーションされている場合は、エラー ページにリダイレクトされるようになった。
080791	クラスタ / プロキシ サーブレットにおいて、複数行のヘッダが考慮されるようになった。
080837	weblogic.xml ファイルに新しい要素が追加された。 <!-- wl-dispatch-policy を使用すれば、コンフィグレーション済みの実行キューの名前を指定して、Web アプリケーションに実行キューを割り当てることができる。この Web アプリケーション レベルのパラメータは、個別のサーブレット /JSP レベルでオーバーライドできる。次に例を示す。 <servlet> ... <init-param> <param-name>wl-dispatch-policy</param-name> <param-value>CriticalAppQueue</param-value> </init-param> </servlet> --> <!ELEMENT wl-dispatch-policy (#PCDATA)>
081071	weblogic.servlet.internal.ServletContextManager.trimAbsUrl で NullPointerException が送出されなくなった。
081484	複製時には PersistentStoreType と共に HttpSessionListener を使用するとセッション属性が失われなくなったが、非複製時にはセッション属性を取得できる。
081521	JSP から CGI スクリプトを呼び出すことができるようになった。
082156	常に True を返すことがないように、isRequestedSessionIdValid の実装が変更された。
082238	CreateSessionServlet は、Cache-Control ヘッダが存在するかどうかをチェックし、存在する場合は有効な値をヘッダに設定し、存在しない場合はヘッダ リストに追加していた。どちらの場合も、別に不正なヘッダが追加されていた。weblogic.xml で CacheSessionCookie を True に設定することで、この問題が解決された。

082310	ページの `contentType` ディレクティブが指定されている JSP を、展開形式の WAR ファイルで削除または上書きできなくなった。
082580	WebLogic Server 6.1 サービス パック 3 では、URL にアクセスするときにフォームベースの認証が実行されて認証に成功すると、HTTP POST パラメータが保持されなかった。
082671	内部的な実装において `HttpSession.getServletContext()` がパブリックになった。
083191	Console で Servlet Average Execution Time （サーブレット平均実行時間）に対して誤った値が表示されなくなった。
083200	`nonProxyHosts` でフィルタ処理が正しく行われるようになった。
083487	`CGIServlet` が正確に一致する `<url-pattern>` を処理するようになった。
083517	`SecureProxy` が ON であっても、`HttpClusterServlet` が SSL に対するリクエストをプロキシ処理していなかった。この問題を解決した。
083597	WebLogic Server 6.1 サービス パック 3 では、ページ ディレクティブを使用するコンテンツ タイプが「`<%@ page contentType="text/html;charset=UTF-8" %>`」と設定されていると、コンテンツ タイプが正しく設定されず、意図した出力が得られなかった。この問題を解決した。
083912	Oracle Thin ドライバの使用時に、サーバがセッション データを JDBC 管理永続性に書き込もうとしても、ORA-01461 エラーが発生しなくなった。
084002	WebLogic Server は、HTTP コードの 204 を返すときに、HTML エラー ページで Content-length に 886 を設定していた。したがって、負荷をかけているクライアントでは障害が発生し、「応答コード 204 には本体があってはならないが、Content-length が 886 に設定されていた（HTML エラー ページで）」ことを示すエラーが生成されていた。この問題を解決した。
084030	展開された Web アプリケーションを 2 つのサーバで共有すると、2 番目のサーバの起動時に、生成された一時ライブラリ (tmp lib) ディレクトリが削除されていた。このディレクトリが削除されなくなった。
084058	`ServletAuthentication.logout()` の後で `request.getUserPrincipal()` メソッドがゲストを返していた。SP4 では、`getRemoteuser()` は null を返すようになった。

084536	CR083377 に対するパッチが原因で発生するセキュリティと SSL 接続に関する問題を解決した。
084649	CGI サーバが内部で実行されているスクリプトに対して提供すると想定されている特定の環境変数がある。Netscape の CGI サーバで使用される SERVER_URL、HTTP_COOKIE、および QUERY_STRING は、同じ CGI プログラムを WebLogic の CGI サーブレットの内部で実行したときには利用できない。これらの変数のいずれかに対してリクエストが行われると、値の代わりに NULL が返されていた。この問題を解決した。
084785	GenericProxyServlet は、ラップされた応答を処理できるようになった。
084847	チャンクされた転送に対し、WebLogic Server は、他のサーブレット エンジンでは無視される 16 進数値を含めていた。この問題を解決した。
085754	リクエスト ディスパッチャと共に使用するときに、中間の JSP である BEATestCase2.jsp に改行 / コメントを入れると、クライアントは空白のページを受け取っていた。サーバ サイドでは例外は送出されなかった。この問題を解決した。
086026	パラメータ CacheSessionCookie のデフォルトとして False が使用されていた。このデフォルトが True に変更された。
086052	WebLogic Server から外向きの HTTPS 接続が、URL の最後にスラッシュ「/」がないと動作しなかった。この問題を解決した。
086280	キープ アライブ接続を通してサーブレットにアクセスしても、NullPointerException が送出されなくなった。
086481	Web アプリケーションが比較的大きなファイルをダウンロードし、マルチパートのフォーム データの読み取り中にバッファ オーバーフローが発生したときの、いくつかの問題を解決した。
088301	JSP の BodyTag タグの内部で Forward ディスパッチされた別のリソースからディスパッチされたとき、インクルードされているリソースがサーブレットの出力ストリームに送信されない問題を解決した。 注意：リクエストが Forward ディスパッチされるとタグが終了するため、ネストされた BodyTag タグの出力ストリームはフラッシュされていなかった。しかし、WebLogic は、まだ出力をタグのネストされた出力ストリームに送る必要があるものと考えていた。Forward リクエストがディスパッチされたときに BodyTag タグのネストされた出力ストリームに対する参照をクリアすることで、この問題を解決した。

CR089803	「CR090225」と同じ。 テストの際、デバイス ドライバが HTTP を使ってリクエストを行うようにすると (`request.aux.%2a.jsp` または `aux.*.jsp`)、リクエストを処理するスレッドがハングした。 `JSPServlet` のコードを修正して、この問題を解決した。上で名前を挙げたデバイス ドライバはすべて、長さ 0 のファイルとして扱われるようになった。WebLogic Server は、この種のファイルを読み込まずに提供する。 http://dev2dev.bea.com/resourcelibrary/advisoriesnotifications/BEA03-14.jsp の「SECURITY ADVISORY (BEA03-14.05)」を参照すること。

セキュリティ

変更要求番号	説明
058355	ユーザが `LoginContext` で `login()` を呼び出すと、この呼び出しは LoginModule の `login()` に直接マップされたが、`logout()` メソッドが実装されていなかった。したがって、このメソッドに対する呼び出しを、LoginModule の `logout()` にマップできなかった。WebLogic Server 6.1 サービス パック 4 では、`LoginModule.logout()` が実装された。
057950	セキュリティのため、Certificate Request Generator サーブレットから [Random String] フィールドを削除した。
068729	`weblogic.security.SSL.SSLCertificate` クラスのドキュメントが作成された。
075109	LDAPRealV2 がメンバシップをキャッシュするようになった。
075451	WebLogic Server の中からリモート WebLogic Server に対する `InitialContext` を取得しようとすると、`weblogic.net.http.HttpsURLConnection` で SSL 接続を行った後で、例外が送出されていた。この問題を解決した。
076783	RDBMS レルムを使用すると、余分な呼び出しが行われていた。キャッシング レルムが修正されて、余分な呼び出しが行われなくなった。
077233	SSL 証明書の絶対パスが提供されるようになった。
077288	WebLogic Server 6.1 SP02 では、グループ内のユーザに対する CN を解析するときに、ユーザ名に「=」が含められていた。 LDAPRealmv2 の字句解析コードを修正することで、この問題を解決した。
077425	キャッシング レルムとともにカスタム レルムが使用されていて、大文字と小文字が区別されないキャッシュのデフォルトが true に設定されていると、キャッシング レルムはエラー メッセージ「Server cannot check in a case-insensitive way」を送出していた。この問題を解決した。
078893	`weblogic.management.password` が空白でない場合は、カスタム ログイン モジュールが呼び出されなかった。`j_security_check` が JAAS ログイン モジュールを使用するには、JAAS ログイン モジュールと共にプログラムによる認証 (`servletauthentication.weak()`) を使用する必要がある。

079183	`weblogic.security.acl.ManageableRealm.setName` を呼び出した後で、`weblogic.security.acl.ManageableRealm.getAcl` が null に戻らなくなった。
079364	deleteAcl は、getACLOwner プリンシパルと共に使用したときにだけ動作した。この問題を解決した。
079637	プライベート キーのパスワードを含むファイルの名前を指定するための `pkpassfiile` プロパティが追加された。
082003、087419	Microsoft Active Directory Server 用にコンフィグレーションされた LDAP レルムを使うと、フォームベースの認証は正しく機能したが、基本認証は失敗していた。この問題を解決した。
082098	資格がリモート サーバ レルムに対して使用および検証されるようになった。
083478	証明書フォームでは電子メールは必要ないが、証明書に `EMAIL=` が含まれていた。この問題を解決した。
084166	ネストされたグループに対するサーバのパフォーマンスが向上するよう、委託グループのキャッシングが修正された。
087485	WebLogic Server 6.1 SP03 では、LDAP はグループのすべてのメンバに対して `Group.isMember()` を呼び出していた。このサービス パックでは、LDAP レルムの本来の動作が回復されている。`groupmembershipcache` の値に 0 を設定すると、WebLogic はグループ メンバシップに対して LDAP レルムをチェックするので、グループに複数のメンバが含まれている場合でも、すべての `members()` がロードされることはない。

システム管理

変更要求番号	説明
059779	データ ソースを、異なるサーバの同じ JNDI 名にバインドできるようになった。
062102	起動時に管理対象サーバにコピーする必要のないアプリケーションがコピーされなくなり、ディスクの空き領域が増えた。
063630	デプロイメントが失敗したときに、残っているアプリケーションを管理対象サーバにデプロイできるようになった。
072833	Administration Console で listenAddress フィールドを空白にして管理対象サーバをコンフィグレーションした場合、異なる複数のマシンからこの管理対象サーバの複数のインスタンスを同じ名前で実行することができた。このような動作は不可能になった。
074370	WebLogic Server 6.1 サービス パック 2 では、Administration Console で EJB 記述子エディタを使用すると、`javax.management.AttributeNotFoundException` が送出された。この問題を解決した。
074653	Administration Console のデプロイメント記述子エディタは、メモリまたはディスクに書き込むときに、「&」を「&」に置き換えるようになった。これは、XML ファイルでは「&」は特殊文字であり、「&」としてエスケープする必要があるためである。
075949	`weblogic.deploy` が JAR ファイルをデプロイメントの直後にアンデプロイする問題を解決した。
076539	`load-on-startup` を指定された Web アプリケーションが、`DeploymentOrder` に従って正しい順序でロードされるようになった。
078257	管理サーバの JNDI が管理対象サーバの MBeanHome を示さず、`NameNotFoundExceptions` が送出されていた。この問題を解決した。
079270	対象となる管理対象サーバの中に停止しているものがあると、アプリケーションの再デプロイメント（削除とデプロイ）操作が成功しないことがあった。たとえば、EJB に対する JNDI エントリが稼働しているサーバに登録されなかったり、警告メッセージのために `config.xml` ファイルの「Targets」属性が失われたりした。この問題を解決した。
079455	デプロイメントが失敗しても `ConfigurationError` が送出されなくなった。

079819	多数のアプリケーションがデプロイされている管理対象サーバの起動に要する時間が短くなった。
080016	アプリケーションが仮想ホスト（管理対象サーバが対象になっている）および管理サーバを対象にした場合、管理サーバの起動の際にアプリケーションのデプロイでエラーが発生しなくなった。
080324	パブリック API に `setParent` メソッドが公開された。
080690	管理サーバは、バウンスして再起動した場合には、管理対象サーバへの接続を再確立するが、JNDI ツリーに管理対象サーバが示されなかった。その結果、アプリケーションは `NameNotFoundExceptions` を受け取っていた。管理対象サーバは、再起動した後に管理 JNDI ツリーに表示された。この問題を解決した。
080778	セッションのモニタリングをオンにしても、新しいセッションが作成されるたびに管理サーバへのトラフィックが発生することがなくなった。
081736	`weblogic.Admin -help` がゲストに対してパスワードを要求した後、パスワードを無視しなくなった。
082765	シャットダウンを呼び出して管理対象サーバをシャットダウンすると、`removeManagedHome` も呼び出されてキャッシュから管理対象サーバが削除される。そうしないと、他の管理対象サーバが起動したときに不要な RMI 呼び出しが行われて、起動に時間がかかる。
082910	EAR ファイル内からカスタム MBean を登録するときに、MBean を使用する JDBC 接続プールを削除できるようになった。
083400	管理対象サーバがデフォルト プロトコルとして T3s を使用して起動する際に、`StackOverflow` が発生しなくなった。
084484	ドメイン ディレクトリ以外のディレクトリからサーバを起動するときには、`-Dweblogic.RootDirectory=<root directory>` オプションを指定できる。起動スクリプトがルート ディレクトリに移動するように変更された場合、起動スクリプトはアプリケーションを検索したが、`config.xml` のパスは、`RootDirectory` を付加されて絶対パスに変更されていた。`config.xml` のパスは絶対パスに変更されなくなった。
084740	`weblogic.deploy` を使ってクラスタにアプリケーションをデプロイすると、サーバを再起動するまでデプロイメントは有効にならなかった。

086235	`ServerRuntimeMBean.getWeblogicVersion` メソッドからサーバのバージョンを取得しようとすると、クラスパスで最初にあるパッチのバージョン情報だけが返されていた。WebLogic Server 6.1 サービス パック 4 では、すべてのバージョン情報が `weblogic.Admin` コマンドから返される。

Web サービス

変更要求番号	説明
076510	Web サービスのメソッドが、実際には CDATA セクションを含む XML ドキュメントである String をパラメータとして受け取ると、メソッドを呼び出した後で、次の応答が送信されていた。 `weblogic.soap.SoapFault: Connector - Bad request to the server` この問題を解決した。
082661	`wsgen` が余分なクラスパスを処理するようになった。
084960	`URLConnection` メソッドを使って Web サービスへの HTTP 接続を確立するときの問題を解決した。Weblogic は、`java.net.HttpURLConnection` を独自のバージョンでオーバーライドした。このバージョンは、ステータス コードが 400 より大きい場合は、`getInputStream` メソッドから例外を送出する。例外が送出されたとき、Web サービスのクライアントが入力ストリームを取得する方法はなく、HTTP SOAP バインドは壊れている。この問題が発生する場合は、新しいコマンドライン オプション `-Dweblogic.net.http.ignore500status` に true を設定する必要がある。
085214	Web サービスがパラメータとして日付を渡すときに発生する問題を解決した。`java.lang.IllegalArgumentException` は送出されなくなった。

WebLogic Tuxedo

変更要求番号	説明
077553	Tuxedo から WebLogic Server に整数配列オブジェクトを値で渡すと、`weblogic.utils.AssertionError` が送出された。 コードを修正し、この問題を解決した。

XML

変更要求番号	説明
080756	組み込みの XSLT プロセッサを使って日本語の文字を変換すると、意図しないコードに変更されていた。この問題を解決した。
080961	`org.xml.sax.helpers.AttributesImpl.removeAttribute()` にあった Xerces 1.3.1 のバグを解決した。
081372	入力エンコードが「Unicode」のときの `DocumentBuilder.parse()` がマルチスレッドセーフになった。
081870	XSLT 出力方法を HTML （必須）に設定すると (`<xsl:output method="html"/>`)、アンカータグの HREF 属性内のスペースは、「%20」にエンコードされた。Netscape は「%20」をスペースとして解釈できず、リンクが壊れていた。この問題を解決した。

WebLogic Server 6.1 サービス パック 3 のソリューション

以降の節では、WebLogic Server 6.1 サービス パック 3 のリリースで解決された問題について説明します。

- クラスローダ
- クラスタ
- コンソール
- EJB
- サンプル
- JDBC
- jDriver
- JMS
- JTA
- その他
- プラグイン
- RMI over IIOP
- セキュリティ
- サーブレット、JSP、および Web アプリケーション
- システム管理
- Web サービス
- WebLogic Tuxedo Connector
- XML

クラスローダ

変更要求番号	説明
063742	エンティティ Bean のリモート インタフェースに対するスケルトン生成の際に、`java.lang.VerifyError` が送出されていた。この問題を解決した。
064273	クラスローダ `weblogic.utils.classloaders.GenericClassLoader` で、`definePackage()` メソッドがマニフェストに従ってパッケージ情報を設定するようになった。
069506	EJB のマニフェスト ファイルの Class-Path: エントリ内にある XML ファイルにアクセスを試みると、`ClassLoader.getResources` が動作するようになった。
075553	不必要なコア クラスローダが保存されていたために発生していたメモリ リークが解決された。

クラスタ

変更要求番号	説明
072281	WAPEnabled=true の場合に、HttpClusterServlet が正しく動作するようになった。
072464	サーバ内で作成された SFSB ハンドラを使用してフェイルオーバを行っても、NullPointerException が発生しなくなった。
073917	クラスタ内の SFSB のハンドラから、フェイルオーバ時に NoSuchObjectException が送出されていた。この問題を解決した。
073920	実行キューの長さはコンソールに表示されない。この機能は、WebLogic Server 6.1 では実装されていない。
074058	WebLogicCluster パラメータに対して DNS の代わりに IP 名を使用すると、一次サーバがダウンした場合に二次サーバが NONE に設定されていた。この問題を解決した。
074236	ClusterMBean に属性 MemberWarmupTimeoutSeconds を追加した。デフォルト値の 0 は、クラスタのウォームアップを行わないことを意味する。デフォルトでは、クラスタの起動シーケンスは変更されない。0 より大きい値を設定すると、クラスタのメンバは、起動の際に、指定された時間だけ待ってから他のメンバとの同期を取る。すべての MessageDriven Beans は、ウォームアップ時間が経過した後でデプロイされる。その後ではじめて、クライアント リクエストの受け付けが開始する。延期された MessageDriven Bean のデプロイメントに対する特別なプロパティはない。

コンソール

変更要求番号	説明
051136	コンソールから壊れたモニタ機能のリンクが取り除かされた。
055821	編集時にパブリック ID に対しては非 null が強制される。
056161	コンソールの EJB 実行時モニタのカスタマイズを使うと、EJB がデプロイされているマシンを見ることができるようになった。
058271	コンソールを使ってデプロイメント記述子を変更しても、Bean のマニフェストファイルが変更されなくなった。
059784	`DefaultWebApp` として `wls_management_internal{1,2}` を選択できなくなった。
059963	コンソールの [サーバ	コンフィグレーション] 画面に [最大オープン ソケット数] が追加された。
060116	WebLogic Server をプロダクション モードで起動すると、管理コンソールに [自動デプロイを有効化] チェックボックスが表示されなくなった。
061664	Netscape ブラウザを使用しても、サーバのシャットダウン メッセージが正しく表示されるようになった。
061686	デプロイメント記述子のヘルプの壊れたリンクが修正された。
061838	管理サーバを再起動しなくても、管理対象サーバを起動できるようになった。
062020	コンソールの XML エディタが `<validate-db-schema-with>` を認識するようになった。
062788	更新ボタンに対する Netscape のデフォルト値が修正されて、[OK] ボタンをクリックするたびにセキュリティ コンソールのページが再表示されなくなった。
062991	ノード マネージャを使って管理対象サーバをリモートで起動するための 5 つのフィールドをセットアップする方法を説明した情報が、コンソールのヘルプファイルに追加された。
063543	コンソールでのサーブレットのモニタに対する `ConsoleMainAttribute` が有効になった。

064849	`weblogic.xml` の記述子エディタが `PersistentStoreTable` パラメータを処理するように修正された。
067101 & 073625	コンソールを通してローカル JNDI ツリーを表示するリモート コンテキストが可能になった。
067362	コンソールが誤ったログイン タイムアウト属性を参照しなくなった。
070018	デフォルト ブラウザが Netscape 6.2 であっても、[Start Default Console] で異常が発生しなくなった。
072909	正しい JMS プール名が実行時テーブルに表示されるようになった。
073536	オプションの相互認証に対するチェック ボックスが Console に追加された。
074693	レルムの実装の変更を保存するために使われるリンクが、いっそう明確になった。
075338	Web アプリケーションのモニタ中に Console に表示される情報が正しくなった。

コア

変更要求番号	説明
CR073529	クラスタを構成する 2 サーバのコンフィグレーションの負荷 (Web ベース) テストにおいて、サーバのロックが発生した。 チャンク処理に対するコードを修正し、この問題を解決した。 http://dev2dev.bea.com/resourcelibrary/advisoriesnotifications/BEA02-20.jsp の「SECURITY ADVISORY (BEA02-20.00)」を参照すること。

EJB

変更要求番号	説明
047305	Console における TxCommits の値の誤りが訂正された。
054789	メッセージ駆動型 Bean が NoSuchElementException を送出しなくなり、メッセージ リスナが処理を終了する前にリスナがリストから削除されなくなった。
056081	EAR のデプロイメントは、コンポーネント ターゲット設定の前で停止する。
057608	4K より多くのデータを CMP Bean に保存できるようになった。
059045	ejbCreate() で run-as が動作するようになった。
059250	WLDD に永続性要素がなくても、ejbc で NullPointerException が発生しなくなった。
060720	6.1 サービス パック 2 では、Blob/Clob を使用する場合は、<isolation-level> を TRANSACTION_READ_COMMITTED_FOR_UPDATE に設定する必要がある。設定しないと、次の例外が発生する。 java.io.IOException: ORA-22920: row containing the LOB value is not locked この場合、Blob/Clob を使用すると、コンテナがトランザクション内のすべてのテーブルをロックする。6.1 サービス パック 3 では、<isolation-level> を TRANSACTION_READ_COMMITTED_FOR_UPDATE に設定する必要はない。コンテナは、Blob/Clob を含むテーブルだけをロックし、トランザクション内の他のテーブルはロックしない。これにより、Blob/Clob が格段に使いやすくなる。
060738	再起動した JMS サーバに JMS クライアントが再接続した後で対応する新しいセッションがリストに追加されず、実体のない複数のメッセージ駆動型 Bean 接続が作成されなくなった。
060867	RemoteMethod の名前が「find」で始まっていても、EJB2.0 のデプロイヤは AssertionError を送出しなくなった。
061700	生成される SQL クエリの一部の識別子に引用される識別子のエスケープがないため、生成される EJB CMP SQL で引用符で囲まれた識別子がエスケープされなかった。この問題を解決した。

062256	DDConverter で -c フラグを指定すると、入力の JAR ファイルが ejb20 に変換されて、複数の JAR ファイルが結合されるようになった。「`java weblogic.ejb20.utils.DDConverter *****-c new.jar***** -d . convert.jar`」のように指定された送り先 JAR ファイルでは、DDConverter が動作しなかった。
062481	EJB を Web 層と同じ場所に配置するかどうかを指定する属性が、`weblogic-ejb-jar.xml` に追加された。
062518	DDConverter は、`findAll` ファインダ クエリに遭遇しても例外を送出しなくなった。
062626	空の `ejb-ql` タグが許されるようになった。
062676	クラスタにおけるステートレス EJB のロード バランシングが、ランダムなロード アルゴリズムでも失敗しなくなった。`weblogic-ejb-jar.xml` で、デプロイメント要素 `home-is-clusterable` をステートレス セッション Bean に対して定義できる。
062916	6.1 サービス パック 2 では、サーバで開始したトランザクションが完了した後で、呼び出し側のトランザクションが再開していなかった。この状態になると、呼び出し側のトランザクションが永久に完了しなかった。この問題は、クライアントのトランザクションを一時停止させるトランザクション属性を持つ削除メソッドがトランザクション内の `EntityBean` で呼び出されて、呼び出しの処理を行うためにサーバで新しいトランザクションが開始されたときにだけ発生した。このような動作の原因となるトランザクション属性は、`NotSupported` と `RequiresNew` である。6.1 サービス パック 3 では、サーバで開始したトランザクションが完了した後で、呼び出し側のトランザクションが再開するようになった。
062974	1 桁のファインダが CMP 1.1 の Bean で定義されていると、クエリから複数の結果が返されても `FinderException` が送出されていなかった。この問題を解決した。
063077	`weblogic-ejb-jar.xml` のトランザクション タイムアウト 属性が機能するようになった。
063146	BMP EJB の同時方式が Exclusive に設定されていて、トランザクション属性が NonSupported に設定されている場合、トランザクション タイムアウトのしきい値に達すると、異なるインスタンスが割り当てられていた。この問題を解決した。さらに、ローカルなトランザクションがロールバックしても、リモート例外が送出されなくなった。

063387	デッドロック状況の後で、WebLogic Server がエンティティ Bean に対するロックのクリアに失敗しなくなった。`LockManager` の待機者がクリアされるようになった。
063469	CMP EJB の関係を作成している際に、`NullPointerException` が送出されなくなった。
064293	待機者が `TimedOut` になった場合でも、`ExclusiveLockManager` はリストから待機者を削除するようになった。
064425	`ejbHomeMethod` の個々の呼び出しでは、プールされている Bean インスタンスを利用するのではなく、新しい Bean インスタンスを作成していた。この問題を解決した。
064447	クッキーから EJB オブジェクト ハンドルをデコードしても、`InvalidClassException` が送出されなくなった。
064561	`<finders-load-bean>=false` の場合、プールされているインスタンスを無視して、新しい Bean インスタンスが作成されていた。この問題を解決した。
064625	受け取るべきではないときに、JDBC から「local interface must not include `java.rmi.RemoteException` in its throws clause」を受け取ることがなくなった。
064967	環境変数が空の文字列であっても、ejbc で異常が発生しなくなった。
064969	DDConverter が正しいアセンブリ記述子を生成するようになった。
065092	6.1 サービス パック 2 における EJB1.1 Bean では、単一のトランザクションにおいて、Bean の永続フィールドを変更した後で同じ Bean を返すファインダを呼び出した場合、永続フィールドの新しい値は、データベースから取り出された値で上書きされる。6.1 サービス パック 3 では、`EntityBean` に対する変更がトランザクションの途中で失われることはなくなった。
065290	クラスタリング可能な Bean に対しても、EJBC/RMIC は `clusterable=true` にした XML 記述子を生成する。
065321	エンティティ Bean に関するロックの問題を解決した。
065571	エンティティ Bean に create メソッドがないと、DDinit が Descriptor ファイルを生成しなかった。この問題を解決した。
066245	`ctx.getCallerPrincipal` が run-as プリンシパルを返さなくなった。

066644	永続的ファイル ストアを備えた JMS のメッセージ駆動型 Bean と Queue が、`onMessage` のトランザクション コンテキストを持ち越していた。この問題を解決した。
067018	ホーム メソッドから `EntityContext.getEJBLocalHome` を呼び出すと、`IllegalStateException` が送出されていた。この問題を解決した。
067515	5.1/6.1 の相互運用性テストの際に JSP を呼び出すと、最初は成功するが、2 回目にはエラーが発生していた。この問題を解決した。
068262	EJB1.1 Bean では、`setBytes` を使って LONG カラムに値を挿入していた。データが 4K より大きいと、この処理は失敗した。`setBinaryStream` を使うようにしたので、次の例外は発生しなくなった。 `ejbCreate(): java.sql.SQLException: ORA-01461`
068569	サーバサイド ejbc を使っていなくても、`tmp_ejb` ディレクトリが作成されていた。この問題を解決した。
068971	ejbc は、CMP EJB に対して非推奨のメソッドを含む JDBC コードを生成しなくなった。
069280	エンティティ Bean に対するクライアントの並行呼び出しの際に、Illegal Reentrant Error が誤って送出されなくなった。
069391	CMP パートナに対して壊れていた CMP マネージャ `cascadeDeleteRemove` を解決した。
070099	method-permission タグを生成するように DDConverter が修正された。
070565	EJB が Web 層と同じ場所に存在することを示すために、clients-on-same-server という名前のプロパティが `weblogic-ejb-jar.xml` に追加された。これは、マルチキャスト トラフィックを減らすために行われた、パフォーマンスに関する修正である。プロパティのデフォルト値は false である。
070754	6.1 サービス パック 2 では、ステートレス セッション Bean のメソッドがまだ処理を行っている間に、Bean で `afterCompletion` が呼び出されていた。この動作は、EJB 2.0 の仕様に違反している。6.1 サービス パック 3 では、ビジネス メソッドと EJB コールバックのシリアライズが行われており、ステートフル セッション Bean のメソッドが処理を完了するまで、Bean で `afterCompletion` が呼び出されることはない。

071027	これまで、コンテナは、SQL クエリ (select * from tableName where 1=0) を発行して CMP テーブルをチェックしていた。DB2 の場合、クエリによりテーブル全体のスキャンが行われて、テーブルのデータが多いと非常に時間がかかるため、このチェック方法では効率が悪い。そのため、`DatabaseMetadata` を使ってテーブルをチェックするオプションが、コンテナに用意された。この方法は、通常は SQL クエリを使うチェックより処理が遅いが、DB2 に関しては SQL クエリより効率がよい。詳細については、`weblogic-rdbms-jar.xml` ファイルの `<validate-db-schema-with>` タグを参照。
071296	エンティティ Bean (EJB 2.0) で `ejbCreate()` を実行すると、`java.lang.IllegalStateException` が送出されていた。この問題を解決した。
071377	EJB QL 拡張機能「`? IS [NOT] NULL`」を生成する機能がサポートされるようになった。
074710	クラスタでステートフル セッション Bean を実行しても、エラーが発生しなくなった。ステートフル セッション Bean に対してパッシベーションを行うとリモート エントリが削除され、Bean がアクティブ化されるとリモート エントリが返らなかったため、`NullPointerException` が発生していた。
075867	WebLogic Sever 上にデプロイされたメッセージ駆動型 Bean には、MQSeries JMS サーバに対する接続の再確立に関して問題があった。スケジューリングされるトリガの数のため、余分な接続が作成されていた。この問題を解決した。
076167	6.1 サービス パック 2 では、読み取り専用 Bean の `ejbLoad` で `SQLException` が送出されると、呼び出し側の（グローバル）トランザクションが再開しなかった。6.1 サービス パック 3 では、再開されるようになった。
078147	EJB11/ CMP11 Bean の使用時に発生していた次の例外の原因である問題を解決した。 Exception in `ejbCreate(): java.sql.SQLException: ORA-01461:` can bind a LONG value only for insert into a LONG column

サンプル

変更要求番号	説明
048185	ACL サンプルが、クライアント サンプルでドキュメント化されるようになった。
053467	`i18n.logging.message` ディレクトリの 2 回目のコンパイルでエラーが発生しなくなった。
056884	サンプルに対する説明の誤りを訂正した。
057904	接続プールの変更が、`petstore.html` に記述されるようになった。
062545	Solaris `simpappcns` を実行する WebLogic Server ドメインが、NT で動作する Tuxedo Corba Server ドメインを呼び出すことができなかった。 `simpappcns` は、Solaris で稼働する Tuxedo と Windows NT で稼働する WebLogic Server のようなマルチプラットフォーム環境に対して動作するようになった。
062962	JAAS サンプルのユーザ名 / パスワード認証が、`CertLoginModule` を使って UNIX 環境でも動作するようになった。
062999	WebLogic Server と Tuxedo の間の WLEC 接続を確立するときに発生していた問題を解決した。WLEC の `simpapp` サンプルが、Tuxedo サービスに接続できるようになった。
063229	`examples/iiop/ejb/stateless/server/tux` を改善した。
071478	固有のウィンドウによって JMS Draw デモ サンプルが壊れなくなった。
077055	Petstore サーバでエラーが発生して Petstore をデプロイできなくなる原因となっていた問題を解決した。

JDBC

変更要求番号	説明
052393	StringBuffer オブジェクトを配置する際に JDBC セッションの永続性でエラーが発生しないよう、シン ドライバを更新した。
054864	6.1 サービス パック 2 では、高負荷の状況において複数のデータベースで MultiPool を使用すると、サーバがハングしていた。データベースが応答しないために MultiPool リクエストがハングし、それによって、weblogic.jdbc.common.internal.MultiPool.searchHighAvail(MultiPool.java:200) においてサーバ全体がロックしていた。原因はコードにおいて過剰な同期を行っていたためであり、この問題は 6.1 サービス パック 3 で解決された。
057340	接続プールが空の場合は、「No available connections in pool」というメッセージが表示されるようになった。
057977	JTS 接続のクリーンアップにおいて、接続プールがロールバックしなくなった。
059020	接続プールが別のユーザに使われているときに weblogic.jdbc.common.pool.shutdownSoft() を呼び出しても、デッドロックが発生しなくなった。
061786	DESTROY_POOL の直後に CREATE_POOL が接続プールを起動するようになった。
064198	一部のクラスは、サーバ サイド コードの中で、オプションの Oracle クラスを直接参照していた。このようなことは行われなくなった。
066001	データベースのパスワードを暗号化するためのプロパティが、JDBCConnectionPool XAPassword に追加された。
066964	JDBC 接続プール用に WebLogic jDriver for Oracle/XA を使用している場合、DBMS の障害と復元の後で接続プールをリフレッシュすると、コア ダンプが発生する場合があった。この問題を解決した。
066966	テスト テーブルに対する JDBC 呼び出しを行っている間、ResourceAllocator モニタが保持されなくなった。
068490	Java レベルで xa_open に対する呼び出しの同期が取られた。
068952	jDrivers に対する自動化されたビルド スクリプトが提供された。

070209	Oracle 9.0.1 シン ドライバを使って Oracle データベースに接続できるようになった。ドライバを使用するには、CLASSPATH において、ドライバのクラス ファイル (classes12.zip) に対するパスを、weblogic.jar に対するエントリの前に追加する。
071974	PreparedStatement キャッシュ サイズがデフォルトで 0 に設定されるようになった。
073640	JDBC 接続プールに対する新しい属性 Open String Password（config.xml ファイルの XAPassword）が追加された。この新しい属性を使うと、オープン文字列を必要とする XA JDBC ドライバを使用する接続プールに対するオープン文字列中のデータベース パスワードを暗号化できる。詳細については、『管理者ガイド』の「接続プールのコンフィグレーションにおけるデータベース パスワード」を参照。
076090	JDBC 接続プールの Properties 属性に含まれるデータベース パスワードの動作が改善された。接続プロパティまたはオープン文字列の一部としてデータベースのパスワードを指定した場合、WebLogic Server はその値を解析して、それぞれ Password 属性と Open String Password 属性に移す。これらの値は、config.xml ファイルに格納される際に暗号化される。詳細については、『管理者ガイド』の「接続プールのコンフィグレーションにおけるデータベース パスワード」を参照。

jDriver

変更要求番号	説明
056531	jDriver におけるパフォーマンスの低下がなくなった。
061643	データベースの回復の後で、`weblogic.Admin` の `RESET_POOL` で障害が発生しなくなった。
063872	ストアド プロシージャの出力パラメータから文字列値が返る場合、末尾のスペースが取り除かれなくなった。
063874	JDBC の `getDate()` メソッド使用時に、SQL DATE の時間コンポーネントが除去されるようになった。この動作は、java.sql.Date に対する JDBC 2.0 の仕様に準拠していなかった。「SQL DATE の定義に準拠するには、java.sql.Date インスタンスによってラップされるミリ秒の値は、そのインスタンスが関係する特定のタイム ゾーンにおいて時、分、秒、およびミリ秒をゼロに設定することにより、「正規化」されなければならない」。 WebLogic jDriver for Oracle を修正した。JDBC getDate() メソッドの戻り値から、時、分、秒、およびミリ秒を除去するようにした。
069109	JDBC のテスト スクリプトが変更されて、Oracle 8.1.7 がデフォルトになった。
069676	WebLogic jDriver for Oracle が、`we8iso8859p15` 文字セットをサポートするようになった。
070878	WebLogic jDriver for Oracle を使用して作成された DBMS 接続で、不正な位置インデックス「0」を指定して `PreparedStatement.setString` を使用すると、JVM がクラッシュしていた。jDriver は、コーディング エラーを適切に処理し、例外を送出するようになった。
078936	WebLogic Server 6.1 Service Pack 2 では、HP-UX UNIX パッチ (PHSS_24627) のアプリケーションが原因で、WebLogic Server 6 が起動時に Oracle にアクセスすると、HP サーバ バス エラーが発生した。より新しいバージョン A.03.33 の `aCC` ライブラリでは、`-AA CXXFlag` を使用すると、WebLogic Server 起動時の Oracle へのアクセスにおいて、`java.lang.UnsatisfiedLinkError` が発生した。 この問題を解決した。

JMS

変更要求番号	説明
044976	MapMessage の get メソッドからの戻り値と例外が、JMS の仕様に準拠するようになった。
058876	Administration Console から JMS サーバを無効または有効にすると、登録されている恒久サブスクライバに対して例外が送出されていた。この問題を解決した。
061094	クラスタ環境において、クラスタの WebLogic Server ノードの 1 つがシャットダウン後に直ちに再起動しない場合でも、JMS サーバでリスンしている JMS クライアントに対して NoSuchObjectException が送出されなくなった。
061552	サーバが 3 時間アイドル状態になっていても、JDBC 接続がリセットされなくなった。
062669	JMS が Unicode 文字列のセレクタをサポートするようになった。特に、表現パーサが StringReader を使うようになり、charVocabulary の範囲が増やされた。
062744	JMS の仕様に準拠するよう、JMS の MapMessage プロパティに対して NULL 値が受け付けられるようになった。
064727	アプレット内の JMS に対する ObjectMessage の作成に関する問題を解決した。
067134	weblogic/management/configuration/JMSConstants.STORE_ENABLED_XXX が追加された。
067286	JMS ページングが、サーバの再起動中にページングを失敗しなくなった。
068667	起動時に、JMS の非永続的メッセージ ページング ファイル ストアが、すべてのディスク ブロックを再利用可能にできなかった。この問題を解決した。
069757	SSL を使用する JMS が weblogic.jms.common.LostServerException を送出しなくなった。
070454	恒久サブスクリプションと共にメッセージ駆動型 Bean を使用している場合、サーバのクラッシュに対してメッセージの回復が失敗または遅延しなくなった。
072556	addReader() で JMS キューがデッドロックしなくなった。
073403	異なる種類で同じ JNDI 名を持つ JNDI オブジェクトによる問題を解決した。

JTA

変更要求番号	説明
062681	JDBC コードにおいて接続オブジェクトがクローズされない場合でも、プールに対する JDBC 接続リクエストで障害が発生することがなくなった。
063053	XA 接続で `openString` プロパティを指定できるようになった。
064232	`javax.transaction.HeuristicMixedException` がコミット時に送出されなくなった。
064403	XA ドライバとステートレス セッション Bean を使用するトランザクションがタイムアウトした後は、WebLogic Server を再起動するまで XAConnection が利用できなかった。この問題を解決した。
064825	`SubCoordinatorImpl` に対するメソッド記述子が、実行時に解析されるようになった。
065495	コーディネータの参照を取得するまでの待機時間を設定できるシステム プロパティが追加された。待機時間を設定するには、クライアントまたはサーバの起動時に `-Dweblogic.JTA.ContactCoordinatorWaitSeconds=<seconds>` を指定する。
066420	メンバ サーバの `peerGone` イベントに対して JTA のサーバ リソース キャッシュが正しく更新されていなかった。この問題を解決した。
068447	`TestConnectionsOnReserve="true"` を指定された XA トランザクションが失敗しなくなった。
070855	負荷のある状態で延期された JTA サーバ リソース キャッシュの更新によってトランザクションが中止されなくなった。
076439	「`Illegal State (Expected: preparing)`」メッセージと共にランダムな XA トランザクションがロールバックする原因となっていた問題を解決した。

その他

変更要求番号	説明
042372	WebLogic で提供されている API `URL()` が正しく機能するようになった。以前は、一部の URL に対して例外エラーが発生していた。
050388	Jolt 接続プールに対する統計情報が Console に表示されるようになった。
052478	Solaris にはタイミングに関する JVM のバグがあり、サーバ起動時の管理オブジェクトのデシリアライズ中に VM が終了する原因となっている。この JVM のバグが発生した場合は、次の方法で対処する。 ■ サーバを再起動する。 ■ 仮想マシンを JDK バージョン 1.3.1_04 にアップグレードする。
053029	RA コンポーネントが削除中に除去されるようになった。
054080	Solaris マシンの 6.1 サービス パック 2 では、SSL を使って（クライアントおよびサーバとして）通信している WebLogic Server が `NativeIOEnabled="true"` を使用すると、JVM インスタンスが消費するメモリが増え続け、最終的にヒープ サイズの限度に達していた。このバグは、取引パートナ間の SSL 通信に依存する WebLogic Integration のインストール環境で発生していた。この問題は、6.1 サービス パック 3 で解決された。
055007	許される接続の数を決定するコンフィグレーション可能なパラメータ `weblogic.system.openSockCount` が作成された。
056622	新しいシステム プロパティ `weblogic.net.http.URLStreamHandlerFactory` が追加された。
056698	顧客プロトコル ハンドラをインストールするときの問題を解決した。
057313	`AbbrevMap` を反復処理する際の `ConcurrentModificationException` を解決した。
057357	エラー ページ 401 を定義するときに発生していた問題を解決した。

057743 & 060981	WebLogic Server の `verboseToZip` ユーティリティと `logToZip` ユーティリティによって、バージョン情報に関する問題を修正した。WebLogic のクラスがバージョン情報に関して依存するマニフェストのエントリが、`weblogic.jar` から抽出されるようになった。`verboseToZip` の出力ファイルには、動的なプロキシ クラスのエントリが含まれなくなった。メッセージ ログに i18n メッセージのインスタンスが含まれている場合、`logToZip` は、i18n 関連のすべてのファイルを `weblogic.jar` からクライアントの jar に抽出する。
058327	アプレットから `DataSource` へのルックアップを試みるとき、`ClassCastException` を取得するようにした。
058358	`application.xml` ファイル中の `<alt-dd>` タグが認識されるようになった。
059054	クッキーの区切り文字として「,」が追加された。
059104	管理対象サーバの起動に使用される JVM を指定するプロパティ `javaHome` がノード マネージャに追加された。
060062	接続プール モニタ画面の [最大接続数] カラムに、WebLogic Server によって更新される接続が含まれなくなった。
060079	t3s 経由で JMS を使用すると、メッセージ サイズが 50K を超えていなくても、`MaxMessageSizeExceededException` が送出されることがあった。
060211	Console の [リモート スタート] タブからノード マネージャに渡される起動引数の配置と引用が修正された。
060739	負荷があっても `UserTransaction` が正しく動作するようになった。ユーザがトランザクションを開始した場合と同様に、EJB がトランザクションを開始した場合も、トランザクション管理による EJB がアイソレーション レベルを自動的に制御する。
061059	データベースから削除されてしまったレコードは発見できなくなった。
061458	WebLogic Server は、ネイティブ ライブラリを使わなくても Solaris および HP-UX で NodeManager の起動をサポートするようになった。
061512	Windows NT および Windows 2000 のサービスとして WebLogic Server を起動するときの問題を解決した。
061847	`examples.io.FileBrowser` で HTTP を使用しても、エラー メッセージが発生しなくなった。

062086	DDConverter を使って CMP 1.1 から CMP 2.0 に変換しても、複数の EJB を含む JAR ファイルに対する例外が送出されなくなった。
062177	クラスタ化されたノード間でピア ツー ピア接続が使われなくなった。
062375	クラスタ環境においてトランザクションの伝播コンテキストをマーシャリングしても、`PeerInfoable.getPeerInfo` が null を返さなくなった。
062439	Windows のネイティブ レイヤにエラー処理コードが追加された。
062565	1.2 JVM クライアントによってソケット接続リークが発生しなくなった。
062752	100 行以上が選択される外部結合操作において、jDriver から `ORA-24347` エラーが発生していた。この問題を解決した。
062926	Solaris または HP において、JSP ページのタグが MS932 固有の文字を処理できるようになった。
062989	低速のモデム接続を使用する JMS が、2 つの WebLogic Server で構成されるクラスタと JMS の `topicConnection` を確立しようとしても、`ClassCastException` が発生しなくなった。
062997	ソース ファイルが削除された場合、サーブレット エンジンが生成される JSP ファイルのサーブレット スタブを削除するようになった。
063103	生成されたスレッド内でコンテキスト ルックアップが失敗していた。この問題を解決した。
063147	JDK 1.3.1_01 プラグインでホーム インタフェース スタブをダウンロードしている最中にアプレットで異常が発生しなくなった。
063310	負荷のある状況でファイル記述子のリークのために `CGIServlet` で異常が発生することがなくなった。
063354	サーブレットからセッション Bean をルックアップする際に、`rjvm.MsgAbbrevInputStream.readClassDescriptor()` から `ClassCastException` を受け取らなくなった。
063671	HTTPS トンネリングを使用するクライアント サイドで、get の `java.net.ProtocolException` が発生していた。トンネリングの結果は OK ではなく、「DEAD（応答なし）」であった。

063830	キャッシュが一杯でも `javax.transaction.TransactionRolledbackException` が送出されなくなった。
064117	サーバが接続を中止して `java.net.SocketException` を送出する原因となっていた問題を解決した。
064125	ノード マネージャでサーバを起動すると、起動コマンドにパスワードが表示されていた。コードを修正し、この問題を解決した。 http://dev2dev.bea.com/resourcelibrary/advisoriesnotifications/BEA02-15.jsp の「SECURITY ADVISORY (BEA02-15.00)」を参照すること。
064130	IIOP 下で複数のルックアップを実行できるよう、`ReplicaLists` がシリアライズ可能になった。
064391	不正な `MANIFEST.MF` エントリを含むアプリケーションをデプロイしても、例外が送出されなくなった。
064404	IE の同じインスタンスを使って、Console を使用する webApp をコンフィグレーションした後、同じ webApp（たとえばフランス語の文字を含む）にアクセスすると、アクセント記号の付いた文字が正しく表示されないことがあった。この問題を解決した。
064434	サーブレットが `RequestDispatcher.include()` を使って JSP の出力を取り込んでいると、後処理フィルタが応答にヘッダを追加できなかった。この問題を解決した。
064860	ページ バッファリングを使用する JSP と共に CGI スクリプトを実行しても、異常が発生しなくなった。
065697	SetUID が有効になっていると、サーバが起動しなかった。この問題を解決した。
066130	準拠チェックの際に ejbc が送出していたエラー メッセージを修正した。
066158	ログ フィルタが定義されていて、管理対象サーバが対象になっている場合、`NonCatalogLogger` を使用するとトラップが生成されるようになった。
066252	特定の Factory/Trader パターンを使用すると、クライアント サイドで `ReplicaAwareRemoteRef` の警告ログが表示されていた。このログは表示されなくなった。
066504	Solaris および Windows 用の JDK 131_02 の SP インストーラが更新された。

067508	AppletArchiver でシン クライアントを生成すると、一般的なクラスローディング例外が送出されていた。この問題を解決した。
067516	appletArchiver が、i18n とマニフェストのファイルをシン クライアントの jar にスプールするようになった。
067517	クライアントの jar を作成するときに、AppletArchiver が L10n タイプの例外を生成しなくなった。
067648	PosixSocketMuxer のコードのために WebLogic Server が ハングする原因となっていた問題を解決した。
068570	エージェントが DoS 攻撃の影響を受けないよう SNMP キットがアップグレードされた。 http://dev2dev.bea.com/resourcelibrary/advisoriesnotifications/BEA02-16.jsp の「SECURITY ADVISORY (BEA02-16.01)」を参照すること。
069043	フィルタと共に HttpProxyServlet を使用しても、NTSocketMuxer で SocketResetException が発生しなくなった。
069675	BEASVC のコマンド ラインで @file オプションを指定してファイルから読み込む際の 4K の制限がなくなった。
069991	サーブレットからセッション Bean をルックアップしても、rjvm.MsgAbbrevInputStream.readClassDescriptor() から ClassCastException を受け取らなくなった。
070575	サーバが約 2 日間稼働すると、最終的に JNDI ツリーと JNDI オブジェクトが同期しなくなり、別の JVM で動作するクライアントが Bean をルックアップできなくなっていた。この問題を解決した。
071580	WebLogic の Time サービスが、Context Class Loader を設定するようになった。
071822	jCOM が WebLogic のライセンス ファイルを正しく発見できるためには、beahomelist ファイルが C:\bea に存在していなければならない。
072010	WLEC 接続プールのコンフィグレーションが、コマンド ラインのプロパティと競合していた。この問題を解決した。
072061	ノード マネージャの -Dweblogic.nodemanager.sslHostNameVerificationEnabled=true オプションでホスト名の検証が実行されるようになった。

072211	エラー状態の処理についてノード マネージャの Solaris ネイティブの実装が改善された。
072228	管理サーバで JDBCConnectionPoolRuntime に対するトラップが生成されていなかった。この問題を解決した。
072278	エラー状態の処理についてノード マネージャの HP-UX ネイティブの実装が改善された。
072327	LONG 型の属性を SNMP でモニタできるようになった。
072495	Timer サービスが原因で発生していたメモリ リークを解決した。
072904	CallRouter を使用すると、ReplicaList が壊れていた。この問題を解決した。
073116	weblogic.Admin の SERVERLOG コマンド ライン ユーティリティを修正した。
073201	モデム接続を通して複数の HTTP トンネリング クライアントを実行すると、サーバが最終的にハングしていた。この問題を解決した。
073400	出力ファイルから javax クラスを除外してクライアント JAR のサイズを小さくするよう、verboseToZip を変更した。
073529	クラスタ化されたサーバが厳しい負荷テストにおいてロックする原因となっていた問題を解決した。
073569	ファイルの領域を節約するため、weblogic.log ファイルから <DGCserver> 情報メッセージが除かれた。
073578	JAVA_HOME として JRE ディレクトリを指定して Windows NT サービスをインストールすると、サービスの起動時にエラーが発生していた。この問題を解決した。
073788	HP のインストーラがビルドに失敗していた。この問題を解決した。
074991	高負荷においてパフォーマンス パックで WebLogic Server がクラッシュする原因となっていた問題を解決した。
075271	weblogic.utils.BubblingAbbrever.getValue (BubblingAbbrever.jaava:163) で NullPointerException が送出されなくなった。Abbrev ツリーが壊れなくなった。

075700	WebLogic Server のインスタンスが Windows サービスとして実行される場合、サービスを実行するために Windows の Administrator 特権が必要なくなった。
076395	WebLogic Server のインスタンスが Windows サービスとして実行される場合、Administration Console または `weblogic.Admin` ユーティリティを使ってサーバをシャットダウンしても、Windows サービスでエラー状態が報告されなくなった。
076705	送り先がリモートの場合でも、guest に対する `NoAccessRuntimeException` によって `MessageDriven Bean` のデプロイが失敗することがなくなった。
077238	起動スクリプトが、正しい値の Oracle jDriver パスを指定するようになった。このサービス パックより前の起動スクリプトでは `LD_LIBRARY_PATH=oci816_8` と指定されていたが、このパスはサポートされなくなっている。現在は、`oci817_8` が指定されている。
077278	WebLogic Server 6.1 サービス パック 3 のインストールから、Oracle 816/806 ドライバが削除された。
077919	`EJBHandle` で `ObjectOutputStream.writeObject()` を呼び出すと「`Closing:'weblogic.rjvm.t3.T3JVMConnection@49c89f' because of: 'Server received a message over an uninitialized connection`」というエラーが送出される原因になっていた、以前のバグ修正に関する問題を解決した。
078523	WebLogic Server 6.1 サービス パック 3 のインストーラには、次のバージョンの Java がバンドルされている。 ■ Windows 2000 および Solaris のインストーラには、Java バージョン 1.3.1-03 がバンドルされている。 ■ HP のインストーラには、Java バージョン 1.3.1.05 がバンドルされている。

プラグイン

変更要求番号	説明
054344	WebLogic Server は、一部の MIME タイプを除くすべてのリクエストをプロキシできるようになった。
057448、059142	高負荷の下で SSL を使用すると発生していたメモリの問題を解決した。
058886	SSL を使用する IIS Proxy でのメモリ リークを解決した。
060325	`GET` リクエストと `POST` リクエストが同時に発生すると、Apache プラグイン内でハングしていた。
060863	NSAPI と ISAPI のプラグインの問題を解決した。2 回目のリトライが成功し、この HTML の内容がブラウザに出力された後であっても、POST タイムアウト エラーがブラウザに出力されていた。
061229	ハンドラが明示的に `weblogic_handler` に設定されていない場合、リクエストは拒否されるようになった。
061379	`RequireSSLHostMatch=true` が NSAPI で機能するようになった。
061386	新しいコンフィグレーション属性 `WeblogicPluginEnabled` が追加された。この属性はサーバ レベルまたはクラスタ レベルでコンフィグレーションできる。 プロキシ プラグイン (または `HttpClusterServlet`) からリクエストを受信するサーバ (またはクラスタ) でこの属性が true になっている場合、`getRemoteAddr` を呼び出すと、Web サーバの代わりに、独自の `WL-Proxy-Client-IP` ヘッダからブラウザ クライアントのアドレスが返される。 プロキシされるリクエストを受信する、クラスタ化されていないサーバの場合は、[サーバ	コンフィグレーション	一般] タブで、この属性をサーバ レベルで設定できる。 `WeblogicPluginEnabled` は `ClusterMBean` と `ServerMBean` で重複する。`ClusterMBean` は `ServerMBean` をオーバーライドする。デフォルト値は false。
061561	ハングしたサーバからのフェイルオーバにおいて、リクエストがタイムアウトしなくなった。
061881、065980、076503	NSAPI プロキシ プラグインが 2k より大きい POST リクエストを処理しても、問題が発生しなくなった。

062607	プラグインの IIS で KeepAlive が有効になっている場合、HEAD HTTP リクエストで接続が直ちに閉じるようになった。
062641	IPlanet プラグインで、JSSE を使用する Java クライアントからのサーブレット アクセスが機能するようになった。
062778	プラグインは、一次 / 二次サーバが現在のサーバ リストにあるかどうかをチェックしないで、これらのサーバへのリクエストをプロキシできるようになった。
062808	Apache 2.0 の最新のベータ版に対するサポートが追加された。
063004	Apache プロキシ プラグインが、高負荷の下で不正な URL を生成しなくなった。
064153	Solaris 用 (libproxy.so) および Win32 用 (proxy36.dll) の NSAPI プラグインが、プロキシ中に、URL の %20 をスペースに置き換えなくなった。
064268	応答ヘッダのステータス「HTTP/1.1 200 OK」で、文字列「OK」が iisproxy.dll によって取り除かれなくなった。
064472	mod_wl_ssl.so を使用する Solaris 2.8 の Apache 1.3.12 EAPI で、ポート ベースの仮想ホストを使うと「Segmentation Fault - core dumped」が生成されていた。この問題を解決した。
064890	Apache で PathTrim を使用しても、クッキーのパスが壊れなくなった。
065188	静的ページに対して 404 エラーの WRITE_ERROR を生成しないよう、iPlanet プラグインが修正された。
065443	NSAPI プラグインで 5.1 と 6.1 のバックエンドに対してプロキシを実行できるようになった。
065660	最後の ISAPI プラグインで HTTP ヘッダが壊れる。
067100	プラグインのビルド情報にバージョン番号が含まれるようになった。
068243	Web アプリケーションに対してフォーム認証を使用しても、ISAPI プラグインの PathPrepend パスが 2 回付加されることがなくなった。
068517	プロキシ プラグインが、ECC 暗号モジュールをロードしなくなった。
068562	複数のサイトに対してプロキシを行うように IIS をセットアップするとき、IIS サーバの前面にファイアウォールまたはロード バランサを置くと、プラグインが動作しなかった。この問題を解決した。

068634	Content Length がプラグインのパラメータ `MaxPostSize` を超えると、413 エラーが返るようになった。
068790	Apache2.0.32 をサポートするようになった。
068985	WebLogic Server で特定のリクエストをプロキシから除外する機能が追加された。
070358	サーバが利用できないとき、Apache プラグインは応答コード 500 を返さなくなった。
072895	二次サーバが削除されているとき、リクエストが同じサーバ（一次）に送信されていなかった。現在は送信されるようになった。
073838	iPlanet を使用している場合、`CookiesEnabled` が false に設定されていても、HTTP セッション ID が失われなくなった。
073936	Apache プラグインが、OpenVMS 7.3 プラットフォームに移植された。
074137	`x-forwarded-for` ヘッダと `client-proxy-ip` ヘッダが送信または設定されていない場合、NSAPI プラグインが「unknown」を返していた。この問題を解決した。
074286	Apache プラグインで 5.1 と 6.1 のバックエンドに対してプロキシを実行できるようになった。
074393	最新の iPlanet プラグインでは URL パラメータが削除されたので、正しい `requesturi` 変数に URL パラメータが追加された。
074656	WebLogic Server を起動するパスワードがコマンド ラインで提供された場合、プロパティ `weblogic.security.jaas.Configuration` が初期化されなかった。この問題を解決した。
076457	高負荷（800 ユーザ以上）の下でも、NSAPI プラグインが安定するようになった。
076578	NSAPI プラグインが、応答不能のサーバへの接続の試みを続けなくなった。
076701	Content-Length が 0 の場合はサーバがポスト データを読み取らないよう要求することで、POST リクエストの Content-Length が 0 の場合でも、Apache サーバがクラッシュしなくなった。
076722	POST リクエストの Content-Length が 0 の場合でも、Apache サーバがクラッシュしなくなった。

076731	特に指定されている場合は、一次サーバと二次サーバがロックされていると、NSAPI プラグインはアクティブなサーバにフェイルオーバするようになった。
076996	SSL のメモリ リークの問題を解決した。
077440	プロキシの IP アドレスが HTTP リクエスト ヘッダに設定されるようになった。
077794	このリリースでは、Apache 2.0 に対するキープアライブ機能が無効になった。
078265	`DynamicServerList=OFF` の場合は `MaxSkips` が使われるようになった。
078883	長い URL で URL の書き換えを使用しても、NSAPI プラグインでコア ダンプが発生しなくなった。

RMI over IIOP

変更要求番号	説明
047057	IIOP の実装において、複雑なオブジェクトに対するサポートが修正された。
062082	IIOP URL の解析が改善された。
067013	デバッグが有効になっている場合にだけ、リースの更新が失敗したときに常に `onMessages` が出力されるようになった。
072179	プール ドライバまたは `DataSource` から取得された JDBC 接続に関するクローズされないときの動作の違いを解決した。
073028	カプセル化本体の前のチャンクの継続が許されるようになった。
073889	`MuxableSocketIIOP` がチャンクで複数のメッセージを受信し、超過量がメッセージ ヘッダの長さより短い場合、メッセージ長を読み取るための以降の試みが誤っていた。この問題を解決した。
074100	Muxer の境界条件が原因でメッセージが壊れることがなくなった。

セキュリティ

変更要求番号	説明
061659	WebLogic Server が、コンフィグレーションされている `NTrealms` で起動するようになった。
061694	SSLClient サンプルを実行すると、`weblogic.security.provider.MD2.<init>(MD2.java:24)` で `NullPointerException` が送出されていた。この問題を解決した。
062172	ブラウザでクッキーが無効になっている場合、フォーム ベースの認証がセキュアな JSP/ サーブレットにリダイレクトされるようになった。
062261	EJB の `SessionContext.getCallerPrincipal()` におけるセキュリティの問題を解決した。
062988	NT レルムと複数の PDC に関する問題を解決した。
063349	`AuthFilter` クラスを実装すると、`doFailAuth()` メソッドと `doSuccessAuth()` メソッドが呼び出されるようになった。
063763	双方向 SSL の Java クライアント接続が、暗号化されたプライベート キーで動作するようになった。
064250	Web サーバが、セッション ID の強制に対して無防備ではなくなった。これは、フォーム ベースの認証で保護されている Web アプリケーションに適用される。
064263	322122 より大きいロックアウト遅延で、ユーザがロックアウトされるようになった。
064285	`ssl.proxyHost/ssl.proxyPort` で SSLClient を実行すると、`SSLHostnameVerifier` が意図したとおりに動作するようになった。
064342	`password.ini` を使用しないで、または `-Dweblogic.management.password=$password` プロパティを指定しないで、echo $password コマンドの出力を Java インタープリタにリダイレクトして WebLogic Server を起動できるようになった。
065046	システム パスワードが変更されたとき、`ManageableRealm.newAcl()` が NULL を返さなくなった。

065314	LDAP Version 2 では自然言語を含むユーザ名に関する問題が発生しなくなった。
066232	RMI/IIOP の `PingClient` サンプルにおける双方向 SSL が動作するようになった。
066264	日本語版 Windows 2000 を使用するロックアウト メカニズムに関する問題を解決した。500 エラーが発生していた。
066491	ブラウザがドメイン ディレクトリのファイルにアクセスできた。 コードを修正し、この問題を解決した。 http://dev2dev.bea.com/resourcelibrary/advisoriesnotifications/BEA02-17.jsp の「SECURITY ADVISORY (BEA02-17.00)」を参照すること。
067726	フォーム ベースの認証に関する問題を解決した。
068524	RMI/IIOP での双方向 SSL および EJB アクセスと認証が、`SimpleCertAuthenticator` サンプルで動作するようになった。
069083	周辺認証に関する問題を解決した。
069160	クラスタ化されていないサーバに対して `ClientCertProxyEnabled` を設定する手段が提供された。
070870	LDAPRealmv2 が完全に実装されるようになった。
071167	iPlanet プロキシ サーバを経由して HTTPS をトンネリングしたときに、`ProtocolException` が発生しなくなった。
072096	SSL と `DebugRC4="true"` を使って管理対象サーバを起動できるようになった。
072916	Serializable を実装するように `weblogic.security.X500Name` クラスが変更された。
075109	LDAPRealmv2 がメンバーシップをキャッシュするようになった。

サーブレット、JSP、および Web アプリケーション

変更要求番号	説明
044528	特定の状況では、`isUserInRole` が、true を返すべきときに false を返していた。この問題を解決した。
051065	JSP ページの XML ビューの XML フラグメントが、出力に渡されていなかった。標準タグでもカスタム タグ（タグ ライブラリ）でもない非標準フラグメントが、出力の現在値にそのまま送信されるようになった。
054694	`RequestDispatcher` がインクルード / 転送に対して適切にフィルタを呼び出すようになった。
055333	JSP の途中で `contentType` ディレクティブを検出すると、JSP コンパイラで異常が発生していた。
056153	ファイル サーブレットが、コンテキスト パラメータだけでなく初期化パラメータをサポートするようになった。
058352	「i.war」や「j.war」のように名前が 1 文字の Web アプリケーションにアクセスできるようになった。
059026	サーブレットを含む Web アプリケーションにおいて、このサーブレットに対して定義されたマッピングが「*.aBc」のような形式であっても、サーブレットへのアクセスが失敗しなくなった。
059102	`RequestDispatcher` で圧縮フィルタが正しく動作するようになった。転送によってフィルタが呼び出されていたが、サーブレット エンジンが空の応答をクライアントに返送していた。この問題を解決した。
059219	Host ヘッダがない場合、WebLogic Server はデフォルトのポート番号を返さなくなった。
060023	ソース ファイル名を算出するときに、`FileServlet` が `SERVLET_PATH` を含むようになった。つまり、`FileServlet` を「/dir/*」などにマッピングすることで、特定のディレクトリからファイルを明示的に提供することだけができる。
060184	`PathInfo` がデコードされて、元の URI の文字を保持するようになった。

060351	必要な URL が見つからないときに、`PageContext.include(String relativeUrlPath)` メソッドが例外を送出していなかった。コンテキスト ログ ファイルに記録され、親リソースおよび取り込むことのできないリソースを示すログ メッセージが追加された。ログ メッセージは mycontext.log ファイルに記録される。mycontext は Web アプリケーションのコンテキスト パスである。
060825	`BodyTagSupport` の `getBodyContent()` メソッドが、本体付きタグが空のときに誤って null を返していた。
061377	jspc は、`EVAL_BODY_AGAIN` の代わりに `EVAL_BODY_TAG` を生成しなくなった。
061782	`ServletResponseImpl.setContentType()` は、マルチパート タイプのプロパティを処理できるようになった。
061784	管理対象サーバからログを取得するとき、管理サーバの Java プロセスが 90% に固定されていた。この問題を解決した。
061864	セキュアなクッキーが表示されなくなった。
061948	タグと、「tagdependent」または「jsp」に設定されている本体の内容を区別するために、新しい意味論的な属性が追加された。lexer の既存の開始タグ ルールの前に新しいルールが追加された。tagdependent ルールが呼び出されると、普通の開始タグルールと同じように動作するが、対応する終了タグが見つかるまで先読みして、JSP の内容を突き合わせる処理も行う。見つかったものはすべて、\n、\r、および \ に対してエスケープされ、「出力」に書き出される。その後、親の lexer ルールに戻り、終了タグを直ちに発見しなければならない。
061968	HEAD メソッドが、`jsp:include` を使用してリクエスト本体を返さなくなった。
062109	JSP スクリプト変数の同期に関する問題を解決した。
062160	プリコンパイルにおいて、`<compileFlags>`、`<encoding>`、および `<compilerSupportsEncoding>` の各パラメータが無視されなくなった。
062289	URL 内のスペースがデコードされるようになった。
062395	`web.xml` の `load-on-startup` が省略可能になった。
062536	`HttpServletResponseWrapper` でフィルタ処理を行ったときに、日本語を含む JSP 応答が壊れなくなった。
062538	`contentType` の場所に関係なく、ページが正しく解析されるようになった。

062542	Windows で稼働する WebLogic Server の `aux.jsp / AUX.jsp / PRN.jsp / prn.jsp` に対するリクエストが、不明の状態のままになるのではなく、成功または失敗するようになった。
062579	CGIServlet がデフォルト サーブレットとして登録されていても、CGI スクリプトが正しく解決されるようになった。
062896 & 063009	Web アプリケーション (WAR) の `weblogic.xml` で JSP パラメータ precompile が true に設定されていると、サーバが再起動するたびに、ドキュメント ルート レベルの JSP だけが再コンパイルされていた。サブディレクトリの JSP は再コンパイルされていなかった。プリコンパイルの際にも、JSP が変更されている場合にだけ再コンパイルされるようになった。
062920	Web アプリケーションをデプロイするときに Web アプリケーションの `web.xml` で `<exception-type>` を指定する（例外クラスは WEB-INF/classes にある）、WebLogic Server は例外クラスを発見できず、`ClassNotFoundException` を送出していた。この問題を解決した。
063007	JDBC 永続性に対するセッション テーブル名がコンフィグレーション可能になった。
063127	セッションが新しい場合、`ServletInputStream` がデータを返すようになった。
063169	アクセント付き文字と `Charset=UTF-8` がある場合、`weblogic.utils.ParsingException` およびネストされた `TokenStreamException` が生成されなくなった。
063288	`CookiesEnabled` が true の場合は、クッキーの中で `SessionCookie` だけが検索されるようになった。
063414	アプリケーションを WebLogic Server 5.1 から 6.1 SP1 に移行する際に、CPU の使用量が大きくならなくなった。
063524	nativeIO をオンにして FileServlet を使用しても、`NullPointerException` が送出されなくなった。
063563	`weblogic.jspc` が、ページ タグから javac に `-encoding JAVAENCODING` の値を渡していなかった。この問題を解決した。
063904	JSP に各ページ ディレクティブを含めると、生成される HTML 出力に余分な改行文字が挿入されていた。HTML が画像で構成されていると、画像の書式設定が失われていた。この問題を解決した。

063925	Web アプリケーションに対して言語固有の文字エンコーディングが指定されている場合でも、応答の出力ストリームに書き込まれるバイナリ データが壊れなくなった。
064294	仮想ホストを使用する WebLogic Server の SessionID の再利用に関する問題を解決した。
064383	アプレットからの HEAD リクエストによって、`ClasspathServlet` から `ClassCastException` が発行されなくなった。
064446	2 レベルのプロキシを使用すると、`getRemoteUser` の呼び出しが常に NULL を返していた。この問題を解決した。
064449	Web アプリケーションのデプロイメントが失敗して `StringIndexOutOfBoundsException` が発生していた。この問題を解決した。
064650	名前の中にピリオドのあるクラスを JAR ファイルからロードできるようになった。
064880	フォーム ベースの認証で、ユーザがログイン ページに直接移動すると、WebLogic Server はユーザをウェルカム ページに転送していた。この問題を解決した。
064988	-k フラグが設定されていても、コンパイル エラーの後で JSP のプリコンパイルがアボートしなくなった。
065104	064650 および 065213 により解決された。
065194	SSL プロセスにおけるマルチパート リクエストが、遅い接続のために失敗していた。この問題を解決した。
065213	`xalan.jar` と `xerxes.jar` の両方が lib ディレクトリにある場合、`NoClassDefFound` が発生していた。この問題を解決した。
065924	新しいオプションの `replicated_if_clustered` を使うことで、クラスタ固有のプロパティを含む Web アプリケーションを、非クラスタ環境にデプロイできるようになった。
066395	誤って挿入された null 文字で、`response.addHeader` が失敗していた。この問題を解決した。
066500	転送された JSP ページにより、現在のページの実行が中止されるようになった。

066569	`request.getParameter()` が、Javaclient によるポスト リクエストのパラメータを返すようになった。
066708	`request.setCharacterEncoding` が、JSP のインクルードに対して動作しなかった。この問題を解決した。
067001	Http Proxy/Cluster サーブレットを使うと、リクエストがステータス 100 で戻ってきた場合、リクエストはいずれかのバックエンド サーバに送信されて、応答の本体がクライアントに返されなかった。この問題を解決した。
067072	JSP に `contentType` 文字セットの指定なしで `pageEncoding` があると、生成される .java ファイルは VM デフォルトのエンコーディングになり、コンパイルが失敗していた。この問題を解決した。
067073	`contentType` と `pageEncoding` の両方が存在すると、生成される .java と javac に対する -encoding が一致していなかった。この問題を解決した。
067077	`pageEncoding` のある JSP ファイルで `weblogic.xml` の `jsp-param` にエンコーディングの設定があると、`pageEncoding` が使用されていなかった。この問題を解決した。
067292	ある JSP から別の JSP へのリダイレクトの際に、リクエスト パラメータの Latin 1 文字が失われていた。この問題を解決した。
067505	HTTP リダイレクトに対する Location ヘッダ内のポートに関する問題を解決した。
067748	サーブレットの出力ストリームを圧縮できるようになった。
067948	インクルードされている JSP ページ内にあるページ バッファ ディレクティブが無視されなくなった。
068024	`HttpServer` または `HttpClusterServlet` で sync ブロックを使用する必要がなくなった。変わらないことが保証されている Set の参照が必要なたびに、新しい列挙値が作成される。
068560	特定の JSP ページのソース コードがブラウザに表示されていた。この問題を解決した。
068648	最大 POST サイズの制限が正しく機能するようになった。
068674	フィルタと共に `HttpProxyServlet` を使用しても、`ClassCastException` が送出されなくなった。

068809	デフォルトとは異なるエンコーディングを使用すると、書き込みシグネチャが優先されるようになった。
068821	Request オブジェクトからの `getRequestURI` が、URI の全体を返すようになった。
069304	`HttpClusterServlet` が、宣言されているコンテンツ長まで、入力ストリームからポスト データを読み込み直すようになった。
069511	エラー ページの属性が、JSP エラー ディレクティブに対するリクエストで設定されるようになった。
069630	JSP 内のカスタム タグのプロパティにエスケープ シーケンス (`\"`) が含まれていると、JSP がコンパイルを行わなかった。この問題を解決した。
069652	クライアントにページを提供するとき、応答ヘッダ (`ContentType`) と JSPWriter が同じ文字セットを使用するようになった。
069809	パブリック ルートにある jsp ファイル、jws (cajun) ファイル、その他のサーバが解析するファイルを見ることができた。コードを修正し、この問題を解決した。 http://dev2dev.bea.com/resourcelibrary/advisoriesnotifications/BEA02-03.jsp の「SECURITY ADVISORY (BEA02-03.03)」を参照すること。
069956	複製されたセッションに対し、オープン セッションのカウントが 1 回だけインクリメントされるようになった。
070090	クライアントに対する応答でセッション クッキーを送信するとき、キャッシング プロキシがクッキーをキャッシングしないようキャッシュ制御の指定が正しく行われていなかった。この問題を解決した。
070132	中国語のファイル名を要求できるようになった。
070151	`<load-on-startup>` に空の値が含まれていても、`weblogic.jspc` が黙って終了しなくなった。
070470	コンテナにまだデプロイされている古いコードによって、以降の再ロードや更新が実行されることがあった。JSP が解析に失敗した場合は古いクラスを再ロードすることで、この種の問題を防止するようになった。
070755	リクエストが wlsproxy 経由でない場合は、wlsproxy 固有のヘッダを送信しないようになった。

070823	WAR コンポーネントと EAR コンポーネントの間にあったインデックス ディレクトリ パラメータの動作の不整合が解決された。
071076	child1 と child2 が同じクラスローダを使用しているにもかかわらず別のクラスローダが作成されて破棄される原因となっていた、クラスローディングの動作に関する問題を解決した。
071082	壊れていた `URLEncoding` を修正した。
071634	WebLogic Server 6.1 が WebLogic Server 5.1 と通信すると発生していたチャンク転送のエンコーディングに関する問題を解決した。
072557	`response.addCookie(name)` が、ブラウザにクッキーが送信された順序を考慮するようになった。
073203	`<url-pattern>` で「+」の文字を使用できるようになった。
073380	`CGIServlet` に対するエラー メッセージが改善された。
073516	CGI の `NonParsedHeader` に関する問題を解決した。
073792	アクティブなセッションが無効にならなくなった。
073797	JSP の `useBean` タグが正しく解析されるようになった。
074015	リクエストの 1 行目に PROTOCOL がないと、`NegativeArraySizeException` が発生していた。この問題を解決した。
075201	URL パターンが 2 文字のディレクトリを参照していても、セキュリティ制約が機能するようになった。
075607	PrintWriter で破棄されたソケット例外が、サーブレットおよび JSP における実行スレッドを拘束することがあった。この問題を解決した。
075628	文字セットを含むように設定された Content-Type を使用してプログラムがサーブレットにデータを送信すると、`getParameterValues("...")` が送信されたすべての値を 2 回返していた。この問題を解決した。
076056	JAR ファイルの `MANIFEST.MF` が別のディレクトリにある JAR ファイルを含んでいると、`StringIndexOutOfBoundsException` が送出されて、Web アプリケーションがデプロイされなかった。この問題を解決した。
076476	複数のフレームを使用すると、1 つのフレームによって設定された最後の `accesstime` を、他のフレームが考慮しなかった。この問題を解決した。

076742	<env-entry-value> 要素に対するチェックが追加された。
077329	セッションが複製に設定されていると、HttpSessionListener が正しく動作しなかった。この問題を解決した。
078930	http://host:port/webApp/aux%20.jsp を要求しても、リクエストを処理するスレッドがハングしなくなった。

システム管理

変更要求番号	説明
050136	コマンド ラインから MBean に対するプロパティ値の設定を解除できるようになった。
052774	実際には存在していない `config.xml` ファイル内のアプリケーションが、デプロイされなくなった。
054836	サーバのクラスタを WebLogic Server 4.5 または 5.1 から WebLogic Server 6.0 または 6.1 に移行すると、`InstanceNotFoundException` が送出されていた。この問題を解決した。
056158	デフォルト Web アプリケーションを含むエンタープライズ アプリケーションをアンデプロイすると、アンデプロイメント時に例外が発生していた。この問題を解決した。
056832	クラスタでは、WLEC 接続プールが起動を試みていなかった。この問題を解決した。
058946	管理サーバでプリコンパイルされた JSP は、管理対象サーバでコンパイルされなくなった。
061426	起動クラスの実行前ではなく後に、サーブレットの起動時ロード (load-on-startup) が行われるようになった。
061676	コンバータ ユーティリティが、すべてのログ ファイルを `SERVER/DOMAIN/logs` ディレクトリに格納するようになった。
061703	アプリケーションが `.wlstaging` 以外のパスを使って（つまり、`config.xml` を手動で編集して）もともとデプロイされていた場合でも、`weblogic.deploy` は常にデフォルトの `.wlstaging` ディレクトリを使用していた。そのため、`weblogic.deploy` 以外の方法でインストールされたアプリケーションに対して更新が行われなかった。この問題を解決した。
062135	展開された Web アプリケーションに Web アプリケーションと同じ名前のサブディレクトリがあると、再デプロイの際にエラーが発生していた。この問題を解決した。
062151	リモート マシン上の管理対象サーバが、Web アプリケーションの名前とタイムスタンプを保持するようになった。

062749	Web アプリケーションを削除し、デプロイし、更新し、再び更新した場合、Web アプリケーションを更新できるようになった。
064101	管理サーバの JNDI が、管理対象サーバの MBeanHome を表示するようになった。
064280	`weblogic.refresh` ユーティリティを使用して管理対象サーバの JSP を更新すると発生していたパフォーマンスの低下を解決した。
064575	`weblogic.refresh` が、Solaris で稼働しているサーバに対し、Windows 2000 から JSP および静的ファイルを更新しなかった。この問題を解決した。
064722	デプロイされるコンポーネントが起動時に存在していない場合でも、`weblogic.deploy` を使って EJB をデプロイできるようになった。
066480	名前が null のオブジェクトを含むカスタム MBean を登録すると、サーバが例外を送出していた。この問題を解決した。
066704	WebLogic Server 5.x から WebLogic Server 6.x への移行において、`FailuresFatal` プロパティが自動的に生成されていなかった。この問題を解決した。
067917	Console アプリケーションの [< ドメイン > | コンフィグレーション | 一般] ページにある Console コンテキスト パス プロパティを使えば、Console 以外のコンテキスト パスでドメインにアクセスできるようになった。
068231	context-root を変更した後で `weblogic.deploy` を使って管理対象サーバにエンタープライズ アプリケーションを再デプロイすると、サーバが再起動するまでの間も、新しい context-root が有効になるようになった。
068579	仮想ホストに対して再デプロイメントを行うと、`javax.management.InstanceNotFoundException` が返っていた。この問題を解決した。
068745	100 ノードのクラスタにおける Console の動作速度が向上した。
069145	アプリケーション ディレクトリにおける自動デプロイでは、管理サーバが空ではない場合には、管理サーバがターゲットに追加されていた。
069407	複数のセンダを `MBeanServerNotificationListener` のリストに保存できるようになった。

069686	Web アプリケーションをデプロイした後、MBeans を使って `weblogic.xml` のクッキー名を変更できるようになった。
071633	MBean インタフェースに対する svuid を修正した。
072676	起動クラス用に Console に `LoadBeforeAppDeployments` が追加された。
072964	Console で EJB 記述子エディタを使用すると `javax.management.AttributeNotFoundException` が発生する原因となっていた問題を解決した。
073674	Web アプリケーションをデプロイした後でも、`weblogic.xml` のセッション永続性タイプを変更できるようになった。
075139	`weblogic.deploy` を使ってクラスタにアプリケーションをデプロイできない原因であった問題を解決した。
075667	管理サーバを再起動すると、管理サーバが管理対象サーバを発見できなかった。この問題を解決した。
077750	JNDI 名が既に使用されている状態で EJB をデプロイするときの問題を解決した。

Web サービス

変更要求番号	説明

046215	wsgen Ant タスクのプロトコル属性を使って、WebLogic Web サービスの起動時に、デフォルトの HTTP ではなく HTTPS を使用するよう指定できるようになった。
	ただし、HTTPS を使用する場合は、以下の追加手順を実行する必要がある。
	■ クライアント アプリケーションにおける URL 作成の直前に、次の Java コードを追加する。
	`System.setProperty( "java.protocol.handler.pkgs", "weblogic.net" );`
	■ WebLogic Server に付属する SSL 証明書を使ってサンプル サーバを実行している場合は、クライアント アプリケーションを実行するときに、コマンドラインで以下を指定する必要がある。
	`-Dweblogic.security.SSL.ignoreHostnameVerification=true`

062976	`AuthUser` および `AuthPassword` に対する `MSSoapClient.ConnectorProperty` が、EJB の `InitialContext` に渡されるようになった。
065205	Web サービスの FaultHandler が呼び出されていなかった。この問題を解決した。
065669	WSDL URL でセキュリティ ACL を使用できるようになった。

066226	WebLogic Web サービスは、クライアント アプリケーションと RPC スタイルの Web サービスの間の双方向 SSL 認証をサポートするようになった。
	双方向 SSL 認証を使用するには、クライアント アプリケーションで Web サービスのルックアップを使用するコンテキストを取得する前に、次の Java コードを追加する必要がある。
	`InputStream certs[] = new InputStream[3];`
	`certs[0]=new PEMInputStream(new FileInputStream("sample_key.pem"));`
	`certs[1]=new PEMInputStream(new FileInputStream("sample_cert.pem"));`
	`certs[2]=new PEMInputStream(new FileInputStream("sample_ca.pem"));`
	`h.put(SoapContext.SSL_CLIENT_CERTIFICATE, certs);`

066703	SOAP クライアントで過剰なエラー メッセージが生成されていた。この問題を解決した。
072788	複合型に対する属性の代わりに要素を使って WSDL が生成されるようになった。
073897	SOAP 障害が CDATA を使用するようになった。

WebLogic Tuxedo Connector

変更要求番号	説明
052022	WebLogic Server の `weblogic.wtc.encoding property` を使って、WebLogic Tuxedo ドメインと Tuxedo ドメインの間のコードセット変換が提供されるようになった。詳細については、「非 ASCII コードセットに対する WebLogic Tuxedo Connector のコンフィグレーション」を参照。
055170	WTC パッケージの `TypedFML(32)` クラスの `Fchg` メソッドに関する問題を解決した。
056230	`tbridge` が、`wlsErrorDestination` JMS キューの初期化に失敗しなくなった。
062843	WTC パッケージの `TypedFML(32)` クラスの `Fdel` メソッドに関する問題を解決した。
063290	Tuxedo から WebLogic Server に大きなコード テーブルをロードするときのサービス呼び出し (`tpcall`) の時間が、Jolt より WTC の方が極端に長いということがなくなった。
066489	FML32 バッファを使用している場合、Tuxedo からクライアントに EJB を呼び出すことができるようになった。

XML

変更要求番号	説明
061660	JAXP API を使って XML ドキュメントを解析しても、エラー メッセージ「Failed to open XML document」が発生しなくなった。
062297	Administration Console を使って、外部エンティティ キャッシュに対するキャッシュ メモリ サイズ、キャッシュ ディスク サイズ、およびキャッシュ タイムアウト間隔を設定できるようになった。以前は、これらのフィールドはいかなる値も受け付けていなかった。
063433	WebLogic Server 6.0 に埋め込まれた XSLT プロセッサに関する問題を解決した。String と、Literal または WebLogic Server 6.0 Filler フィールドに埋め込まれた XSLT プロセッサを含むグループが MFL ファイルに含まれていると、変換がハングしていた。
067966	WebLogic Server クラスローダの改善により、任意のバージョンの Xerces XML パーサをプラグインできるようになった。
065345	`CharReader.fillCurrentChunk` が `java.lang.ArrayIndexOutOfBoundsException` を送出する原因となっていた問題を解決した。
067022	WebLogic Server 6.1 に付属するトランスフォーマが、XML ドキュメントを意図したとおりに変換するようになった（指定された XSL を使って）。
068656	`weblogic.xml.jaxp.ChainingEntityResolver.pushEntityResolver` における `NullPointerException` の原因となっていた問題を解決した。
073311	`weblogic.apache.xalan.xslt` を使って大量のデータを変換すると、`javax.xml.transform.TransformerException`（文字列インデックスが範囲外）が発生していた。この問題を解決した。
075800	外部エンティティの参照を使って元の XML ファイルに外部の XML を取り込んでいる XML ファイルを、組み込みの XML パーサが正しく解析するようになった。

WebLogic Server 6.1 サービス パック 3 に付属するサードパーティの JAR ファイル

JAR ファイル	説明
J2EE、javax とサブディレクトリ	JTA 1.0.1
	JCA 1.0
	JMX 1.0
	EJB 2.0
	Servlet 2.3
	JSP 1.2
	JMS 1.0.2
XML	org.apache.xalan 2.0.1
	org.apache.xerces 1.3.1
	Xerces 1.2.0 - DOM と SAX のパーサ
	org.w3c.dom のインタフェース
	org.xml.sax のインタフェース
LIBRARIES	antlr 2.7.0 - antlr コンパイラ構築ツールキットの実行時クラス
	bsh 1.2b3 - Bean シェル
	certicom - SSLPlus/Java 3.1.12B （ノード マネージャが使用）
	netscape - Netscape Directory SDK 4.1 for Java
	Oracle 8.1.6
	Oracle 8.1.7
	Apache Jakart Ant - 1.3 Apache ビルド システム

Apache Jakart Oro 2.0 - 正規表現マッチング ライブラリ

WebLogic Server 6.1 サービス パック 2 のソリューション

以降の節では、サービス パック 2 を適用した WebLogic Server 6.1 で解決された問題について説明します。

- EJB
- サンプル
- JDBC
- jDriver
- JMS
- その他
- プラグイン
- RMI over IIOP
- セキュリティ
- サーブレット、JSP、および Web アプリケーション
- システム管理
- Web サービス
- WebLogic Tuxedo Connector
- XML

EJB

変更要求番号	説明
044294	複数のエンティティ Bean を作成するときに発生する可能性があったメモリ リークが修正された。ガベージ コレクタが、使用したすべてのメモリを再使用していなかった。
049590	`ejbc` コンパイラは、プリミティブ値型が先で非プリミティブ値型が後という、正しい順序で IDL ファイルを生成するようになる。
049709、 055644、 059540	これまでは、クラスタに EJB をデプロイした後で再デプロイまたはアンデプロイを試みると、タイミングの問題によって例外 / エラーが送出されていた。 ローカルなアンデプロイが行われた後で JNDI アンデプロイのメッセージが到着する可能性がある。EJB のローカルなアンデプロイが行われると、EJB に対するクラスローダは破棄されるので、これによりアンマーシャリング エラーが発生する。このアンマーシャルの問題によりログ ファイルには `DistributedManagement` 例外が記録され、EJB はクラスタからアンデプロイされない。 同様に、ローカルなデプロイメントが成功する前に JNDI バインディングの通知が到着し、同じアンマーシャリング / デシリアライゼーション エラーが発生する可能性がある（アンマーシャルする必要のある EJB クラスがまだ存在しないため）。 6.1 サービス パック 2 での解決策はこの両方の問題に対処し、ローカルなデプロイ / アンデプロイと JNDI 通知におけるタイミングの依存性を取り除いている。クラスタの EJB のデプロイ / アンデプロイ / 再デプロイを正常に実行でき、競合状態による例外と問題は発生しなくなっている。
051454	`idle-timeout-secs` の値より長く EJB がアイドルになっている場合、EJB はキャッシュから削除されて、`ejbremove()` が呼び出される。
055113	`examples\ejb\basic\statelessSession\build.xml` ファイルで、EJB jar ファイルの名前に誤りがあった。これを修正した。
055163	クラスタ内でインメモリ レプリケーションを行うようにコンフィグレーションされたステートフル セッション Bean の問題を解決した。`EJBException` が送出されると Bean はフェイルオーバしていたが、Bean が削除されるとフェイルオーバしていなかった。

055502	アプリケーションが例外を送出した後、メッセージ駆動型 Bean がメッセージを回復するようになった。
055510、056543	コンテナ管理による永続性 1.1 EJB が再デプロイされた後でデプロイメント記述子を読み込み直さない問題を解決した。
057119、054806	EJB を何回も再デプロイしたときに発生するメモリ不足の問題を解決した。`<max-beans-in-free-pool>` 記述子が無視されて、この値を超過していた。
058210	`ejbRemove()` メソッドは、ステートフルとステートレスの両方のセッション Bean の、ホーム インタフェースとリモート インタフェースの両方について、アンデプロイメント時に呼び出されるようになっている。
058857	ホームに `create()` メソッドがない場合にエンティティ Bean が認識されない問題を解決した。
059043	`RemoteException` を送出しない例外句を `EJBLocalObject()` メソッドが送出しても、`ejbc` はそれを報告しなくなった。
059046	クラスタ内でステートレス セッション Bean からステートフル セッション Bean にアクセスしても失敗しなくなった。
060018	コンテナ管理による関係で ReadOnly EJB にアクセスしても、`LockTimedOut` 例外が発生しなくなった。
055842	クライアントが認証を受けられないためにメッセージ駆動型 Bean が別の物理マシン上にあるリモート キューにアクセスできない問題を解決した。
058074	ステートフル セッション Bean のインスタンス フィールドに EJB ローカル インタフェースへの参照を格納しても、パッシベーションの際にアサーション エラーが発生しなくなった。
053523、060032	プロトコルが t3s および HTTPS の場合に `weblogic.ejb20.internal.BaseEjbHome` の `getURL()` メソッドが非セキュア ポートを使用する問題を解決した。
054335	ReadWrite エンティティ Bean を削除すると対応する ReadOnly エンティティが無効になるようになった。これまでは、ReadOnly エンティティがキャッシュに残っていた。
059674	Bean 管理による永続性を使用する読み取り専用エンティティ Bean を使用しても、ファインダを介して Bean を読み込んだときに `ejbLoad()` が 2 回呼び出されることがなくなった。

060142	`getEJBHandle()` メソッドの問題を解決し、2 ノードのクラスタでもフェイルオーバが動作するようにした。
060416	階層環境において EJB ルックアップを実行した場合に EJB Home オブジェクトを固有の型にキャストすると `ClassCastException` が送出される問題を解決した。
061883	読み取り専用 Bean に対する再入可能呼び出しがロック タイムアウトのために失敗する問題を解決した。
062515	コンテナ管理による永続性の EJB ホームに対して `findByChar(char c)` メソッドが失敗しなくなった。

サンプル

変更要求番号	説明
055155	次のサンプル ファイルにおける問題を解決した。 `ejb20\cascadeDelete\one2many\table.ddl` この問題により、「SQL not properly ended」というエラー メッセージ（エラー番号 933）が表示されていた。

055370	`wlserver6.1\samples\examples\jsp\` の次のサンプルにあった切れたリンクが解決された。
	`SimpleSession.jsp`
	`URLEncode.jsp`
	`SimpleSession` では「セッション タイムアウト」のリンクが切れていた。
	`URLEncode` では「Web アプリケーションのデプロイメント記述子の記述」のリンクが切れていた。

056711	`wtc/simpFML32` サンプルのビルドにおいてコンテナ トランザクションに関する警告が生成されなくなった。

057237、056063	RMI-IIOP C++ のサンプルが UNIX プラットフォームで動作するように修正された。これらのサンプルは、以下のサブディレクトリにある。
	`/samples/examples/iiop/ejb/entity/tuxclient`
	`/samples/examples/iiop/ejb/stateless/server/tux`
	`/samples/examples/iiop/ejb/stateless/tuxclient`
	`/samples/examples/iiop/rmi/server/tux`
	`/samples/examples/iiop/rmi/tuxclient`

JDBC

変更要求番号	説明
055044	動的に作成された接続プールが、`config.xml` の中で永続的として扱われなくなった。
055435	JDriver for Oracle は、データベースをクエリしてデータベースが 500 行より多い場合、正しい数の null 値を取得するようになった。
056268	言語をロシア語に設定して SQL クエリの実行を試みたとき、タイムスタンプで使用されるハイフンにエラーの原因となるハイフンが含まれなくなっている。

jDriver

変更要求番号	説明
046149	CLOB または NCLOB のカラムに WebLogic Server がマルチバイト文字列を書き込む際に不正な長さの文字列が書き込まれる問題を解決した。

JMS

変更要求番号	変更要求番号
055293	JMS のデフォルト接続ファクトリが、`config.xml` ファイルの `JMSDefaultConnectionFactoriesEnabled` プロパティの機能として正しくバインドされるようになった。これまでは、他の JMS オブジェクト (`JMSConnectionFactory`、`JMSServer`) が特定の WebLogic Server で定義されていないと、デフォルトの接続ファクトリが JNDI にバインドされなかった。
056320	`Topic` を使って `ObjectMessage` として文字列を送信した後で別の `ObjectMessage` として `RemoteObject stub` を送信した場合の、オブジェクトのデシリアライズのエラーに関する JMS の問題を解決した。
061783	同じ JMS トピックをサブスクライブする異なるメッセージ駆動型 Bean のインスタンスを同時にデプロイしようとしたときに発生する問題を解決した。 これまでは、一方または両方の Bean の動作が停止していた。

その他

変更要求番号	説明
036679	Solaris で Netscape ブラウザのフレームのサイズを変更した場合でも、Administration Console が正しく機能するようになった。
041873	`-Djava.security.manager` でサーバを起動したときの `weblogic.policy` ファイルに関する問題を解決した。ユーザが正しいパーミッションを持っていても、`java.security.AccessControlException: access denied` 例外が発生していた。
041991	`RemoteMBeanServer` から `MBeanHome` を取得しようとしても、アプリケーションが null を受け取ることはなくなった。
043010	WebLogic を Windows サービスとして実行している場合、サーバを実行している Windows セッションからユーザがログアウトしても、WebLogic が終了することはなくなった。
050437	WebLogic Express の下で動作していても、WebLogic は `<Error> <JMS> <JMSServer "null", JMS license validation failed, com.bea.utils.misc.NoSuchProcessException: No such license: WebLogic Server 6.1, JMS.>` というメッセージを出力しなくなった。このメッセージは、誤解を招く無害なものであった。
053732、054781	`beasvc.exe` ユーティリティに `-delay` オプションが導入されて、管理サーバが既に動作している場合にだけ、管理対象サーバが起動することが保証される。新しい `-delay` オプションを使えば、管理対象サーバの JVM スレッドの実行を、指定したミリ秒数だけ遅らせることができる。 これまでは、管理対象サーバは、稼働している管理サーバに依存するために起動に失敗していた。 **注意：** WebLogic Server 6.1 SP04 での関連する変更については、1-3 ページの「beasvc.exe に対する -delay 引数の動作の変更」を参照すること。
053990	ノード マネージャは、名前にホワイトスペースを含むサーバをロードするようになった。
055007	WebLogic Server にスロットリング メカニズムが実装されて、一定の限度を超えた未処理のリクエストは拒否されるようになった。これまでは、サーバの能力を超える可能性のある新しい接続の数を制限する上で `acceptBacklog` 値は役に立たなかった。

055442	HTTP 経由での XML の送信に関係する問題を解決した。WebLogic Server が `getInputStream()` メソッドを誤って呼び出していたのが問題であった。
056332	RPC スタイルの Web サービスを呼び出すと `IllegalStateException` が発生する問題を解決した。
057091	Internet Explorer でのアプレットに関する問題を解決した。誤った時点でコンテンツの長さのチェックが実行され、「`didn't meet stated Content-Length`」エラーが送出されていた。
058832	プライマリ サーバが停止した直後にクライアントがセッションにアクセスすると `HttpSession` フェイルオーバ サービスが失敗してセッションが失われる問題を解決した。この問題は、2 サーバのクラスタでのみ発生していた。
059119	Weblogic クラスタと対話する Java クライアントの実行が 3 サーバ クラスタでは 2 サーバ クラスタより遅くなる問題を解決した。さらに、この問題では、クラスタのサーバの数と少なくとも同じ数のソケット リーダー スレッドをクライアントが確保するまで、サーバの数を増やしてもクライアントのパフォーマンスの低下が続いていた。これには、実行スレッド数 (Execute Thread count) とソケット リーダーの割合 (Percent Socket readers) というクライアント側プロパティのコンフィグレーションが関係していた。 この解決策では、サーバの数を動的に増やしたり減らしたりできるよう、このようなコンフィグレーションの依存性を取り除いた。修正後の WebLogic Server では、動作中にソケット読み取り用のスレッドが新しく作成されて、クライアントからの送信ソケットごとに 1 つのスレッドが存在することが保証される。つまり、WebLogic クライアントが WebLogic クラスタと対話する場合に、ユーザはパフォーマンスを最適化するためにクライアントに対するこれらのプロパティを調節する必要はない。
059219	WebLogic が正しくない情報が存在する可能性のある場所からポート番号を取得するという、ワイヤレス アプリケーションで発生していた問題を解決した。
060557	FileT3 属性でファイルを作成する場合に、パスが管理対象サーバに対するものであっても正しいパスでファイルが作成されるようになった。
061060	`getTimeout()` メソッドと `setTimeout()` メソッドが、`HttpURLConnection` クラスに存在するようになった。

056911	これまでは、`.war` ファイルに対するマニフェスト クラスパスにエントリを追加した場合、Web アプリケーションのクラスローダがそれをロードしていた。今回の修正で、アプリケーションのクラスローダがそれをロードするようになった。実質的な影響として、すべての Web アプリケーションは、クラスの独自のコピーを取得する前であっても、マニフェスト クラスパスからロードされたクラスのインスタンスを共有するようになった。
051220	WebLogic Server のログまたはコンソールでマルチバイトのシステム メッセージが壊れて読めなくなる問題を解決した。
054871	Administration Console で JNDI ツリーが正しく表示されるようになった。以前は、デプロイメントが誤ったコンテキストの下に表示されていた。
057466	日本語ロケール (`LANG=ja_JP.PCK`) の Solaris で動作するサーバに対して `weblogic.Admin PING` コマンドを実行するとサーバで `java.net.SocketException` が表示される問題を解決した。
058786	Netscape Enterprise Server プラグイン（NSAPI プラグイン）を使うと、Netscape Enterprise Server（NES、iPlanet とも呼ばれる）から WebLogic Server へのリクエストをプロキシできるようになった。プラグインは、WebLogic Server の動的な機能を必要とするリクエストを WebLogic Server が処理できるようにすることで、NES のインストールを拡張する。 以前は、NSAPI プラグインで HTTP ヘッダを読み取ると、予期しない EOF (PROTOCOL_ERROR [line *linenumber* of *filename*]: unexpected EOF reading HTTP headers at line *linenumber*) が発生していた。 大きなクッキーの解析に関する問題であることが判明した。 この問題を解決した。

061460

クライアント接続の数を制限するよう Weblogic 6.1 をコンフィグレーションできるようになった。2 つの方法がある。

1. `config.xml` ファイルを編集して `weblogic.system.openSockCount` パラメータを設定する。このパラメータは、許される接続の数を制限する。

```
<Server ListenPort="7001" Name="myserver"
NativeIOEnabled="true"
TransactionLogFilePrefix="config/mydomain/logs/"
MaxOpenSockCount=1000>
```

2. `weblogic.Admin` を使ってソケット数を設定するか、または JMX Java クライアントを記述してソケット数を設定し、次のコードで示すように `weblogic.management.configuration.ServerMBean` の API を利用する。

```
/**
* ある時点においてサーバで開くことのできるソケットの最大数を返す。
* 最大しきい値に達すると、サーバは、ソケットの数がしきい値より少なくなるまで、
* それ以上新しいリクエストを受け付けることを停止する。
*
* @default java.lang.Integer.MAX_VALUE
* @legalMin 0
* @legalMax java.lang.Integer.MAX_VALUE
*
* @configurable
*
* @oldprop weblogic.system.openSockCount
*/

int getMaxOpenSockCount();

/**
* ある時点においてサーバで開くことのできるソケットの最大数を設定する。
* 最大しきい値に達すると、サーバは、ソケットの数がしきい値より少なくなるまで、
* それ以上新しいリクエストを受け付けることを停止する。
*
* @default java.lang.Integer.MAX_VALUE
*
* @legalMin 0
* @legalMax java.lang.Integer.MAX_VALUE
*
* @configurable
*
* @oldprop weblogic.system.openSockCount
*
*/
void setMaxOpenSockCount (int sockCount);
```

051882	Web アプリケーションに web.xml がない場合にコンソールでデプロイメント記述子を編集または入力しようとしても、空白の画面が表示されなくなった。

プラグイン

変更要求番号	説明
053641	`KeepAliveEnabled` パラメータの設定にプラグインが正しく応答するようになった。HTTP 1.1 を使用すると、`Connection: Keep-Alive` ヘッダが WebLogic に返送されていなかった。
055276	ISAPI プラグインでキャッシングを無効にする際の問題を解決した。
055750	Apache の Stronghold バージョンで `MatchExpression` を使用する際の問題を解決した。
056628	Apache HTTP Server プラグインの `PathTrim` パラメータを設定すると発生していた問題を解決した。`PathTrim` パラメータに “/” を設定しても、セッション情報が失われることがなくなった。
060064	Solaris プラットフォームにおいて Apache HTTP Server プラグインを静的にリンクされたモジュールとしてインストールすると致命的エラーが発生した問題を解決した。

RMI over IIOP

変更要求番号	説明
056233	`ejb11.security.HomeMethodPermissionTest.testHandleRemoveWithoutPermission()` がメモリ不足エラーで失敗する問題を解決した。

セキュリティ

変更要求番号	説明
060491	Web アプリケーションの初期リソースが保護されたリソースでない場合にシングル サインオン (SSO) が機能しない問題を解決した。
CR061313	別のサーバインスタンスのコンテキストを取得するサーバインスタンスに関して、セキュリティ上の弱点があった。 `Security.getCurrentUser()` で認可されたユーザであることを確認するようにコードを修正し、この問題を解決した。 http://dev2dev.bea.com/resourcelibrary/advisoriesnotifications/BEA02-13.jsp の「SECURITY ADVISORY (BEA02-13.00)」を参照すること。

サーブレット、JSP、および Web アプリケーション

変更要求番号	説明
046879、057371	cgi 実行ファイルがブラウザに画像（.gif、.jpg、.png の各フォーマット）を正しく提供するようになった。以前は、この処理を試みると WebLogic Server ログに例外が送出されていた。
047698	IndexDirectoryEnabled プロパティが管理対象サーバに対して正しく機能するようになった。ディレクトリとファイルの両方がリストされる。 以前は、特定の Web アプリケーションに対して IndexDirectoryEnabled を true に設定した場合、管理対象サーバでその Web アプリケーションが要求されると、ファイルだけリストされて、ディレクトリはリストされなかった。また、その Web アプリケーションでファイルを追加または削除した場合、ブラウザのページを更新してもその変更が反映されなかった。
050402	Web アプリケーションを異なる名前で再デプロイすると発生していた問題を解決した。以前は、Administration Console を使って Web アプリケーションをデプロイした後、MBean を使って同じ .war アプリケーションを異なる名前でプログラム的にデプロイすると、WebLogic Server で無限ループが発生していた。
053513	HttpClusterServlet を使ってリクエストをプロキシするとブラウザでのページ表示が正しくなくなる問題を解決した。
053706	Web アプリケーションで AuthFilter を登録して ServletAuthentication.weak を呼び出すと発生する問題を解決した。このような場合は、AuthFilter で doSuccessAuth が正しく呼び出されるようになった。
053965	response.sendRedirect の対象となる URL パラメータの一部に ASCII で 128 ～ 156 の間の文字がある場合、これらの文字は第 2 のサーブレットに対して正しく送信されるようになり、第 2 のサーブレットがこれらの文字を ASCII 63 として受け取ることはなくなった。
054206	HttpServletRequest の実装クラスである weblogic.servlet.internal.ServletRequestImpl において、複数行にまたがる HTTP ヘッダ行が正しく処理されるようになった。以前は、コードが正しく解析されなかった。

054211、055987	新しいクライアントサイド プロパティ `http.keepAliveCache.lifeTime` と `http.keepAliveCache.proxyLifeTime` を使って、HTTP の永続的接続の有効期間をコンフィグレーションできるようになった。以前は、有効期間をクライアントサイドでハードコーディングしていた。この値は、サーバサイドの `keepAlive` の値をオーバーライドし、スケーラビリティの問題となっていた。
054852	エラー ハンドラがコンフィグレーションされた Web アプリケーションでは、JSP1 が JSP2 に転送して JSP2 が例外を送出すると、このエラー ハンドラにリダイレクトするようになった。 以前は、サーバは空白ページを返し、200 というステータス コードを表示していた。サーバは例外を記録していたが、クライアントはそれを見ることができなかった。
054898	`Authfilter` と `doSuccesAuth` を使用する JSP ページに関する問題を解決した。最初の JSP ページは、異なる Web アプリケーションに含まれる一連の JSP ページ内にある。
055333	JSP の先頭ではなく途中で `contentType` タグを使用した場合の JSP のコンパイルに関する問題を解決した。
055466	EJB を呼び出すことができずに `ClassCastExceptions` が発生するサーブレット フィルタに関する問題を解決した。
055535	JSP 更新ツールを使用したときにファイル配布サーブレットがファイルを間違った場所にコピーする問題を解決した。
055583	JSP カスタム タグで `TryCatchFinally` を実装すると、コンパイル時に正しい Java コードが作成されるようになった。
055595	WebLogic Server は JSP のタグ ライブラリ URI を解決できるようになった。
055636、56265	JSP で `getParameter()` メソッドを使用する際に、入力された値が空であっても、null が返ることはなくなった。空の文字列が返る。これにより、クエリ文字列で等号を忘れた場合の問題が解決された。
056092	JSP コンパイラで変数宣言の重複が誤って示されることがなくなった。
056189	WebLogic Server は「Transfer-Encoding: Chunk」を使用するクライアントをサポートするようになった。つまり、クライアントは、POST リクエストの中で「Transfer-Encoding: Chunk」を使用する場合、「Content-Length」ヘッダを送信する必要はない。

056322	異なるアプリケーション内の 2 つの異なるサーブレットが同じ EJB にアクセスを試みても、断続的な RMI エラーは送出されなくなった。
056668	パラメータが空であっても、`getParameter()` メソッドは null ではなく空を返すようになった。
056955	Web アプリケーションの `weblogic.xml` で `<charset-params>` が定義されている場合、または `web.xml` で `<context-param>` が定義されている場合に、リクエストからの POST データが `forward()` の後で失われる（あるサーブレットが POST データを使ってリクエストを別のサーブレットに転送した場合）という問題を解決した。
057101	JSP が含まれる場合に `getParameter()` メソッドが複数の値の最初の値を返さないという問題を解決した。 以前は、最初ではなく 2 番目の値が返されていた。
057279	展開されたディレクトリとパッケージ化された Web アプリケーションの間にあった動作の不一致を解決した。展開ディレクトリでは、クラスがロードされる順序が正しくなく、`WEB-INF/classes` のクラスより前に `WEB-INF/lib` のクラスがロードされていた。
057620	コードセットのエンコーディングが「UTF8」の場合に、「+」や「&」などの特殊文字が正しくデコードされるようになった。
057684、057559	アプレットが EJB に対するクライアントの場合に `PortableRemoteObject.narrow()` を呼び出しても `ClassCastException` が発生しなくなった。
058125	Web サービスに対して生成される `client.jar` が JDK 1.2 Java クライアントと互換性を持つようになった。
058187	JSP の更新が失敗する `weblogic.deploy` に関する問題を解決した。展開ディレクトリ形式のアプリケーションに追加された新しいディレクトリが管理対象サーバに伝播されず、そのために更新が失敗していた。
058220	SOAP リクエストの送信を試みても、`servlet.internal.ChunkOutput.clearBuffer` でサーブレット エラー `java.lang.NullPointerException` が発生しなくなった。
058352	「i.war」のように名前が 1 文字の Web アプリケーションにアクセスできるようになった。以前は、この種のアプリケーションをデプロイすることはできても、アクセスすると 404 エラーで失敗していた。

058714	Web コンテナを介して直接ストリーミングするのではなく cgi-bin 実行ファイルを通してストリーミングすると `.gif` 画像が正しく表示されないという問題を解決した。
058738	HTTP 1.0 でリクエストを送信した場合に、応答ヘッダにおいて HTTP のバージョンが正しく保持されるようになった。
058931	JSP を削除すると、500 エラーではなく 404 エラーが発生するようになった。
059054	WebLogic Server がクッキーの区切り文字「,」の解析に失敗し、新しいセッションが誤って作成される問題を解決した。
059180	WebLogic Server が独自のクッキーを単純に追加するのではなく、別のエンティティによって作成された既存のクッキーを上書きしてしまう問題を解決した。
059412	`session.removeAttribute()` を 2 回呼び出してから `session.setAttribute()` を 1 回呼び出すと、両方の削除がオーバーライドされるのではなく 1 つの `removeAttribute` が残ってしまうという、サーブレットの問題を解決した。
060192	URL の書き換えを使ってセッションを追跡すると、リクエストがプライマリサーバにピン固定されないでセッションが失われる問題を解決した（ブラウザでクッキーが無効になっていた）。
060536	`jspc` が SUCCESSFUL の終了コードで終了しない問題を解決した。
060595	`HTTPServletRequest` をラップするフィルタを使用するとネストされた JSP で `ClassCastException` が送出される問題を解決した。
060615	Web アプリケーション間でセッション ID が一致していない問題を解決した。現在では、各 Web アプリケーションは、特定のユーザのセッションに対して単一の同じ ID を認識するようになっている。
060963	Web アプリケーションがシングル サインオンに参加していない場合、WebLogic Server はセッション ID を再利用しないようになった。
061277	(JSP 1.1) 日本語プラットフォームにおいて `javac` コンパイラのエラー メッセージが壊れなくなった。
061387	JSESSIONID クッキーが 2 つのユーザ定義クッキーに挟まれている場合に、ページが要求されるたびに WebLogic Server が新しいセッションを作成する問題を解決した。

061488	値で「=」が使用されている場合にリクエスト パラメータが正しく解析されない問題を解決した。
066184	6.1 SP1 では、デフォルトのエンコーディングは JVM であった。6.1 SP2 からは、デフォルトのエンコーディングが ISO8859_1 に変更された。この変更は、最新のサーブレット仕様に従ったものである。サーブレット仕様 2.3 SRV 4.9 では、デフォルトのエンコーディングは ISO8859_1 でなければならないと規定されている。

システム管理

変更要求番号	説明
045780	同じ名前で複数のサーバを定義することができなくなった。サーバを起動した後、2 番目のサーバを同じ名前で起動しようとすると、サーバの起動は中止されて、次のエラー メッセージが表示される。 "Server: <*yourservername*> already exists, make sure that you do not have duplicate copies of this Server name in your XML config file"
055190	アプリケーションがクラスタを対象としている場合は JSP の更新が失敗する問題を解決した。以前は、稼働している Web アプリケーションから新しい JSP を追加したり既存の JSP を変更または削除したりするには、アプリケーションのアンデプロイと再デプロイを行う必要があった。
056091	管理対象サーバが動作している状態で別のマシン上の管理サーバを再起動できる新しいプロシージャが利用できるようになった。詳細については、「管理サーバのフェイルオーバに関する考慮事項」を参照。
058946	管理対象サーバ上のプリコンパイル済み JSP が、ページが変更されていない場合でも再コンパイルされる問題を解決した。この問題により、管理対象サーバの起動に時間がかかっていた。 この解決策により、管理サーバでクラスが既にコンパイルされている場合は、管理対象サーバを起動すると、そのクラスが移動されてタイムスタンプが維持されるので、管理対象サーバでは再コンパイルが行われない。
054624	`-component` オプションの使用時に `weblogic.deploy` が意図したとおりに動作するようになった。 以前は、`-component` オプションを使用すると `weblogic.deploy` が意図したとおり動作せず、コンポーネントのサーバ リストにないサーバにアプリケーションが誤ってデプロイされていた。
054573	JDBCConnectionPool、MultiPool、または (Tx)DataSource に関係するコンフィグレーション エラーが、コンソールのエラー ダイアログで報告されない問題を解決した。正しくコンフィグレーションされたように表示されていた。
058183	WebLogic Server が `.jar` ファイルより前に `.war` ファイルをデプロイする問題を解決した。サーバ起動時に EJB より前に Web アプリケーションがデプロイされたため、`NameNoteFoundException` が発生していた。

059702	ネットワーク カードが無効になっていると Web アプリケーションがデプロイされない問題を解決した。 以前は、WebLogic Server は http://www.bea.com/servers/wls610/dtd/weblogic-web-jar.dtd の DTD で検証を試みていた。www.bea.com にアクセスできないと失敗していた。現在は、ネットワーク アクセスがない場合は、`weblogic.jar` で `weblogic-web-jar.dtd` を見つけて処理を続ける。
060244	Administration Console を使ってクラスタの対象をただ 1 つのアプリケーションにすると個別のサーバもターゲット リストに取り込まれる問題を解決した。この解決策により、クラスタはサーバのリストに展開されなくなった。
061642	5.1 の `weblogic.properties` ファイルから 6.1 の `config.xml` ファイルへの変換に関する問題を解決した。変換ユーティリティが、JDBCConnectionPool に対するパスワードを正しく変換していなかった。その結果、新しいドメインでサーバを起動しようとすると失敗した。
055067	プリコンパイルがオンになっていると EJB の `.jar` ファイルのクラスと依存関係のあるアプリケーションがデプロイされない原因となる問題を解決した。
060411	(EJB 1.1) 必須の EJB トランザクション メソッドに対する 2 番目の呼び出しによってトランザクションがロールバックする問題を解決した。最初の呼び出しによってエンティティがいくつか削除されてからいくつか作成されていた。2 番目の呼び出しでは、前に作成されたエンティティを削除してから新しいエンティティを作成していた。
042052	あるリリースから別のリリースへの移行で発生していた問題を解決した。`weblogic.properties` ファイルが正しく変換されず、Web アプリケーションの `web.xml` ファイルに含まれていないプロパティがあった。
061300	WebLogic Server コンソールが __weblogic_admin_html_queue（優先度の高いキュー）ではなくデフォルトのキュー（優先度の低いキュー）で実行される問題を解決した。そのために、サーバに大きな負荷がかかっているとコンソールの応答能力が低下していた。

Web サービス

変更要求番号	説明
055415	SoapWare テスト仕様と UserLand クライアントを使用するときの相互運用性の問題を解決した（参照実装）。
054749	`wsgen` Ant タスクを使ってステートレス セッション EJB を RPC スタイルの Web サービスにアセンブルするとき、次に示すように EJB のパッケージ レベルに 1 つのレベルしか含まれていないと `wsgen` が解析エラーで失敗する問題を解決した。 `package mytest;`
055645	WebLogic Web サービスを手動でアセンブルするために使用する `Remote2WSDL` クラスに、HTTPS のような HTTP 以外のプロトコルを指定するための `protocol` オプションが正しく含まれるようになった。
056332	以前は、WebLogic Server が SOAP リクエストを処理して WebLogic Web サービスを呼び出す間に例外が発生すると、`ServletResponse.getWriter()` メソッドが 2 回実行されていた。そのために別の例外が発生していた。この問題を解決した。
056382	WebLogic Web サービスが `xsd:decimal` データ型をサポートするようになった。
056631	WebLogic Web サービスが `xsd:hexBinary` データ型をサポートするようになった。
056639	WebLogic Web サービスのクライアント API を使って、`ejb-jar.xml` ファイルおよび `weblogic-ejb-jar.xml` ファイルでセキュリティ制約が定義されたステートレス セッション EJB を持つ RPC スタイルの WebLogic Web サービスを呼び出しても、エラーが発生しなくなった。
058230	WebLogic Web サービス クライアント API において、SOAP リクエストの内部でオブジェクトを受け取った後、クライアント側で複雑な Java オブジェクトを正しく作成できるようになった。この複雑なオブジェクトとは、JavaBean の配列で構成されていて、さらにその属性の 1 つ自体が別の種類の JavaBean の配列になっているようなものである。
058431	WebLogic Web サービスは、ネストされた配列データ型の値を正しく返すようになった。

059653	Remote2Wsdl を使って Web サービスをアセンブルするとき、WSDL の <soap:address> 要素の location 属性に相当する、URL の context 部と jndi_name 部を指定できるようになった。つまり、WSDL から取得した Web サービス プロキシ上でメソッドを正常に呼び出すことができる。

WebLogic Tuxedo Connector

変更要求番号	説明
055889	WebLogic Tuxedo Connector から TPESVCFAIL と共に送出された `TPReplyException` を、Tuxedo クライアントが受信できるようになった。
057150	`FldTblClass` 要素を持つコンフィグレーション ファイルの有効性を検証するときに、`WTCValidateCF` ユーティリティがエラーを送出しなくなった。

XML

変更要求番号	説明
044211	xmlx.jar ファイルに、HTMLEntities.res リソース ファイルが正しく含まれるようになった。
046934	組み込み SAX パーサが、任意の文字セットを使って記述されているデプロイメント記述子ファイルを正しく解析するようになった。 **警告：** 以前は、英語のみのアプリケーションの .xml ファイルにおけるエンコーディングの問題を検出できなかった。したがって、以前は問題なく動作していた英語専用アプリケーションが、サービス パック 2 におけるこの対処によって異常を示すようになる可能性がある。非サポート エンコーディング エラーが報告される。 このエラーが通知された場合は、指定された .xml ファイルでエンコーディング名と構文が正しいことをチェックする必要がある。
055082	weblogic.jar アーカイブの XSLTInfo.properties ファイルについて、正しくない /org/apache/xalan/res/XSLTInfo.properties ではなく、/weblogic/apache/xalan/res/XSLTInfo.properties が正しく呼び出されるようになった。その結果、変換が正しく動作し、SAXException が送出されなくなった。
056839	weblogic.apache.xalan.serializer.Encodings クラスに、MS932 と Shift_JIS の間の正しいマッピングが含まれるようになった。
062320	WebLogic FastParser が、非常に大きな文字列（1MB より大）を分単位ではなく秒単位で解析するようになった。

WebLogic Server 6.1 サービス パック 1 のソリューション

以降の節では、サービス パック 1 を適用した WebLogic Server 6.1 で解決された問題について説明します。

- デプロイメント記述子エディタ
- EJB
- サンプル
- JDBC
- JMS
- JTA
- その他
- プラグイン
- RMI over IIOP
- サーブレットと JSP
- システム管理
- Web サービス
- XML

デプロイメント記述子エディタ

変更要求番号	説明
041479	J2EE アプリケーションの更新（再デプロイメント）で発生していたメモリ リークが修正された。
048757	コンソールのデプロイメント記述子エディタの隣にあった黄色の警告アイコンがすべて削除された。
048804	コンソールの EJB デプロイメント記述子エディタで、Container Transaction Trans 属性に「(none)」オプションがなくなった。
049465	コンソールの EJB デプロイメント記述子エディタで、Weblogic RDBMS Jar の表示名が記述子の読み込まれたファイル名から派生する。
051858	`weblogic-ejb-jar.xml` ファイルで参照されるすべての RDBMS 記述子がデプロイメント記述子エディタでアクセス可能になった。
051939	`Auto-acknowledge` と `Dups-ok-acknowledge` をメッセージ駆動型 Bean の有効な確認応答モード要素として受け入れるように `ejb-jar.xml` 解析コードが修正された。
052613	デプロイメント記述子を編集するためのリンクが、アプリケーションのコンフィグレーション時には表示されないようになった。
052665	`web.xml` ファイルに `<run-as>` という有効な要素がある場合に、それがデプロイメント記述子エディタの使用時に失われることがなくなった。
052804	`<charset-params>` エンティティとすべての子エンティティが、デプロイメント記述子エディタによって `weblogic.xml` ファイルで維持されるようになった。
052963	`.war`、`.jar`、または `.rar` ファイルに関して、コンソールのコンポーネント テーブルでデプロイメント記述子エディタにアクセスできなくなった。
053089	デプロイメント記述子エディタにおいて、自動キー生成機能で使用する主キーが Java の型（`java.lang.Integer`）でなければならないことが示されるようになった。
053098	`MessageDrivenDestinationMBean` の `SubscriptionDurability` 属性で有効な値のリストが追加された。`MessageDrivenMBean` から `SubscriptionDurability` 属性が削除され、`MessageDriven.jsp` から `SubscriptionDurability` プロパティが削除された。

053282	EJB 2.0 デプロイメント記述子エディタに exclude-list タグが追加された。
053375	デプロイメント記述子エディタが、コンソールのチェック ボックスを使用して修正された Bean が再デプロイされた後に空白にならなくなった。
054682	デプロイメント記述子エディタから EJB デプロイメント記述子の設定を維持するときに、すべての description 要素の内容が解析エラーを防ぐために CDATA セクションで囲まれるようになった。

EJB

変更要求番号	説明
031350	EJB コンパイラで、エンティティ Bean の主キー クラスにすべてのパブリック フィールドがあることが正確に検証されるようになった。以前も検証機能はあったが正確に機能しなかった。
044294	エンティティ Bean を繰り返し作成するときに発生していたメモリ リークが修正された。
047050	メッセージ駆動型 Bean の `onMessage` で非検査例外が送出された場合に、`setRollbackOnly` の代わりにトランザクションのロールバックが呼び出されるようになった。
048506	複数の Bean が作成されて、削除されなかった場合のメモリ リークが修正された。
048716	EJB 準拠エラーのメッセージで、どの Bean に問題があるのかが示されるようになった。
049180	Bean の解放を要求されたときに、`EntityPool` で `unsetEntityContext()` が呼び出されなかった。この問題は修正済み。
049658	`ejb-ql` の入力パラメータで算術演算子を使用する場合の問題が修正された。+、-、/ という 3 つの記号のための機能が追加された。
050134	`weblogic.ejbc` デバッグ フラグが機能するようになった。デバッグ機能を利用してコンパイルするには、`-debug` の代わりに `-g` を使用する。次に例を示す。 `java weblogic.ejbc -g foo.jar bar.jar`
050860	DDConverter で、cmp20 Bean の更新された構文で `ejb-ql` が生成されるようになった。
051528	`weblogic-rdbms20-persistence-600.dtd` の `<weblogic-query>` タグでオプションの `<description>` タグ要素が追加されたので、クエリを記述できるようになった。
051653	主キー フィールドが外部キーの一部である多対 1 の関係における SQL クエリ生成の問題が修正された。
051670	別の EJB のハンドルを参照する EJB でパッシベーションが行われたときに、`ClassCastException` が送出された。この問題は修正済み。

051735	EJB コンパイラは、sj として指定されたときに -j オプションを認識せず、したがってコマンドを実行できなかった。この問題は修正済み。
051803	準拠チェッカーで、local または local home をリモート インタフェースのパラメータの型または戻り値の型として渡すことが許可されなくなった。詳細については、EJB 仕様のセクション 10.3.10.1 を参照。
051807	準拠チェッカーにおいて、リモート コンポーネントとローカル コンポーネントが同じ CMR マッピングを共有することが認められなくなった。
051875	XML ジェネレータで、明示的に設定された値が無視されなくなった。値がデフォルト値と同じでも生成される。
051941	remove というビジネス メソッドの不正な処理が発生する EJB コンテナの問題が修正された。
052075	ejbSelect でフィールドの集合が返される場合に SELECT DISTINCT が機能するようになった。
052502	コンソールの [デプロイ] チェックボックスで EJB をデプロイすると、EJB のデプロイメント記述子に対する変更が取得されるようになった。
053093	テーブルの自動作成機能が有効な場合に、シーケンス テーブルが生成されるようになった。
053096	ejbSelect メソッドで、Bean インスタンスのセットではなくローカル Bean のセットが返されるようになった。
053101	準拠チェッカーで、<acknowledge-mode> の値が不正ではなくなった。

053287	ejb-jar.xml ファイルの次のエントリで、ファイル全体がロードされないということがなくなった。
	
	`<message-driven-destination>`
	`<destination-type>javax.jms.Topic</destination-type>`
	`<subscription-durability>Durable</subscription-durability>`
	`</message-driven-destination>`

053596	アクティブな StatelessSessionBeans の数が、max-beans-in-free-pool 設定を超えられなくなった。
053599	Bean クラスを判別する EJB DDInit ツールの機能が改良された。

053643	ローカル インタフェースのない EJB から `SessionContext.getEJBLocalObject()` が呼び出された場合に、`IllegalStateException` が送出されるようになった。
053646	`ejb-jar.xml` ファイルをバリデートするときに、準拠チェッカーと XML バリデータで「role-source」ではなく「relationship-role-source」を参照して EJB 仕様の変更に準拠するようになった。
053735	ローカル メソッドの呼び出し時に、Null ポインタ例外が `StatefulSessionManager` で送出されなくなった。
053768	デプロイヤがアップデートされて、EJB 2.0 を有効にすることを目的として `weblogic.EJB20Enabler` クラスが検索されなくなった。また、別のダウンロードを参照するライセンス エラー メッセージが不要になったので削除された。
053801	メッセージ駆動型 Bean のクラスに `ejbCreate()` メソッドが含まれていないが、そのサブクラスには含まれている場合に、準拠チェッカー エラーが送出されなくなった。
053827	DDConverter ユーティリティに、java `weblogic.ejb20.utils.DDConverter` を説明どおりに使用してアクセスできるようになった。
053884	`ejbPostCreate` メソッドが throws `CreateException` を使用して定義されており、SQLServer 自動キー生成機能が有効な場合に、不正確なコードが生成された。この問題は修正済み。
053905	UPDATE で EJB2.0 BLOB/CLOB マッピングを使用する場合で、更新されたフィールドが元のフィールドより短い場合に、新しく更新された結果に元のフィールドのデータが残されなくなった。
053944	ステートレス セッション EJB のコンストラクタで `RuntimeException` または `RuntimeError` が送出される場合に、EJB にアクセスした後で呼び出し側の `java:comp/env JNDI` コンテキストが不正確にならないように例外処理コードが修正された。
053970	デフォルトのロール名と関係名が生成されるようになったので、`ejb-jar.xml` 記述子の関係宣言で `ejb-relation-name` 要素を設定することなく EJB 2.0 関係を設定できるようになった。
053986	記述子 MBean が、XML の大文字 / 小文字の区別に関して一貫性を持つようになった。

054021	クラスタ対応エンティティ Bean がクラスタ非対応の EJB から呼び出されるときにアサーション エラーが送出されなくなった。
054225	jar ファイルの `Manifest.mf` ファイルの Class-Path エントリに循環的な参照が含まれている場合の `ear` ファイルのデプロイメントの問題が修正された。
054449	コンテナ管理の永続性を使用する EJB 2.0 エンティティ Bean 向けに生成されるコードが、Oracle の制限を克服するように修正された。Oracle LONGRAW カラムにマップされる cmp-field が、4096 バイトを超える長さを持つことができるようになった。

サンプル

変更要求番号	説明
045300	Oracle Thin ドライバと JDriver の両方を使用する新しい JTA サンプルが追加された。このサンプルでは、2 つのデータベース インスタンスを使用する。各データベースでは、1 つのテーブルを設定し、新しいトランザクション接続プールとトランザクション データソースを作成する必要がある。このサンプルでは、最終ではない API 仕様を使用する J2EE 1.3 機能が使用される。

048031	クラスタのアドレスおよびサンプル クライアントを呼び出すときのリスン ポートを指定するための手順、構文、および例が明確になった。以下の 2 ファイルが変更されている。
	`...\samples\examples\cluster\ejb\package.html`
	`...\samples\examples\cluster\rmi\package.html`

050429	テーブル作成 sql スクリプトがベンダに依存しないように標準化された。
051476	`examples/ejb20/bands` の問題を解決した。`demoPool` 接続プールは、`oraclePool` 接続プールを指し示すようになった。
051491	`examples\jta\jmsjdbc\build.xml` ファイルの参照とサンプル コードが修正された。
051714	`examples/ejb/sequence/mssqlserver /package-summary.html` の問題を解決した。`demoPool` 接続プールは、`mssqlserver` 接続プールを指し示すようになっている。
052595	`i18n.logging.message` ディレクトリの 2 回目のコンパイルでエラーが出なくなった。
052689	Petstore を使用して複数のクライアントを実行したときの問題が修正された。
053008	SSLClient サンプルのマニュアルの情報が修正された。アプリケーション ディレクトリとしての `production_apps` ディレクトリの参照がなくなり、`SnoopServet` は `\WLS_HOME\config\examples\jsp` ディレクトリではなく `\WLS_HOME\samples\examples\jsp` ディレクトリに配置されている。

053463、053466	インターナショナライゼーションのロギング サンプルにあった壊れたリンクを解決した。 `WL_HOME/samples/examples/i18n/logging/package-summary.html` には `WL_HOME/samples/examples/i18n/logging/message/UserSMessagesServlet.html` へのリンクが含まれるようになっている。 `WL_HOME\samples\examples\i18n\logging\message\UserSMessagesServlet.html` における「Setting up your environment」、「examplesweb application」、および「Start the WebLogic Server with the examples configuration」の各リンクは、`WL_HOME\samples\examples\examples.html` を正しく指している。
053383	JSP フォーム バリデータ サンプルになかった設定ステップが追加された。
053396	無線 stockDemo サンプルで、リアルタイムの株価情報が返されるようになった。
053525	開発モードで mydomain のデフォルト サーバを起動すると、アプリケーション ディレクトリの `DefaultWebApp` と `certificate.war` ファイルが 2 回デプロイされていた。
053852	DTD の最新バージョンを登録するように DOCTYPE が更新された。
053871	すべてのプラットフォームで jsp が一貫性を持ってコンパイルされるように構築スクリプトが修正された。
054963	移行されたドメイン用に生成された起動スクリプトが他のスクリプトと一貫性を持つようになった。
056080	`examples\i18n\logging\message` 構築スクリプトの失敗の原因になっていた問題が修正された。
056114	メッセージ Web サービス サンプルの問題を修正するリテラル エンコーディングが追加された。
056347	Petstore のショッピング カート EJB でクラスタ化が使用されなくなった。
056495	Time サンプルが削除された。

JDBC

変更要求番号	説明
034041	日付に基づいて JDBC ログをローテーションする機能が追加された。ローテーションの頻度と開始のタイミングを指定することができる。現在の `jdbc.log` ファイルには、ファイル名の最後に最近の日付タイムスタンプが付けられる。
034046	トランザクションの動作が不正確になるため、XA 対応接続プールと非対応接続プールの混在するマルチプールに基づいてデータ ソースを作成することはできなくなった。
035380	properties ファイルだけでなく JMX API を使用して JDBC DataSource を動的に作成できるようになった。
042924	マルチプールを指し示す `TxDataSource` の作成が機能するようになった。以前は、`ResourceException` が発生していた。
045991	コンソールで、プロパティとしてではなく URL の一部としてログインとパスワードを指定して接続プールを定義しても、例外が送出されなくなった。
049963	接続プール実行時 MBean を提供する `JDBCConnectionLeakProfile` インタフェースと `JDBCStatementProfile` インタフェースに関する記述が javadoc のコメントに追加された。
051058	`weblogic.jdbc.common.internal.ConnectionPool.startup(Deployment MBean)` で「reserve」ACL から「admin」ACL にチェックが変更された。
051165	対応するプロファイルにリセット メソッドが追加されて、プロファイルのストレージを 0 の初期サイズにプログラムでリセットできるようになった。
051371	`weblogic.Admin` JDBC の使い方メッセージに、接続プールのコマンドが含まれるようになった。
052540	`PStatement` キャッシュのサイズが、Administration Console および `JDBCConnectionPoolMBean` を通じて明示的に処理されるようになった。`PreparedStatement` キャッシュのサイズは、データベース プロパティで設定することができなくなった。
052960	`JDBCMultiPoolMBean.java` の `HighAvailability` と `LoadBalance` のゲッター メソッドとセッター メソッドが非推奨になった。代わりとして、`AlgorithmType` のゲッター メソッドとセッター メソッドが導入された。

053085	JDBC プール ドライバの Connection クラスの `close( )` メソッドと `isClosed( )` メソッドに同期化が追加された。
053721	`getPoolList( )` メソッドが 6.1 でもまだ `JDBCMultiPoolMBean` に存在し、javadoc で文書化されている。
054187	複数の呼び出し時に、`PreparedStatement` クエリで不完全な `ResultSet` が返されなくなった。
055655	Oracle 8.1.6 の JDBC ドライバ DLL が正常になった。

JMS

変更要求番号	説明
051938 & 053464	送り先にアクセスできなくなった場合に、プロデューサではそのことを正確に検出できるようになり（peerGone）、送り先がアクセス不能のままの間は例外を送受信するようになった。送り先にアクセスできるようになると、プロデューサは回復して機能を正常に再開する。
053635 & 054655	適切なトランザクション セマンティクスを取得するために、アプリケーションは EJB 内で常に XA 接続ファクトリを使用する必要がある。さらに、メッセージ駆動型 Bean（MDB）で使用されるデフォルト接続ファクトリが XA 接続ファクトリになった。EJB 内で XA として使用されるユーザ定義の接続ファクトリをコンフィグレーションするには、Administration Console を使用して、[接続ファクトリ] ノードの [コンフィグレーション	トランザクション] タブの [XA コネクション ファクトリの有効化] チェック ボックスを選択する（XAConnectionFactoryEnabled="true"）。
053649	WebLogic JMS では、確認応答されるメッセージまでのメッセージだけが確認応答されていた。現在は、[確認応答ポリシー] 属性が [Previous] に設定されていない限りはすべてのメッセージが確認応答される。
054154	メッセージが JMS の外でシリアライズされる場合に（EJB または RMI 呼び出しへの値渡し）、送り先が削除されなくなった。
054365	セッション プール接続コンシューマによって実行されるメッセージの事前割り当ての動作が修正された。接続ファクトリの messagesMaximum 設定に関係なく、接続コンシューマは connectionConsumer.messagesMaximum の設定に従って消費する。
055364	読み込み専用モードで受信されるメッセージのプロパティを設定する場合に、clearProperties() メソッドを呼び出してプロパティを書き込み可能にできるようになった。以前、受信メッセージにクリアするプロパティがない場合に、clearProperties() ではプロパティが書き込み可能にならなかった。
056556	同じセッションで複数のサブスクライバが作成される場合に、最初のメッセージの受信後に非同期サブスクライバがハングしなくなった。

JTA

変更要求番号	説明
024199	JTA ドライバが、最上位のワークスペースで読み込みパーミッションを必要としなくなった。
048979	複数のクライアントで同じリソースが同時に使用された場合に次のエラーを生じさせた JTA の競合状況が修正された。 `Unable to create resource runtime mbean`
050789	ドメインにデプロイされたアプリケーションをリフレッシュするための自動更新間隔が、[JTA Subsystem] ではなく [Domain options] で指定されるようになった。
050849	JTA リソース状態モニタ機能が改良された。不健全な（ブロックされた）トランザクションと不健全なリソースははっきりと区別されるようになった。リソースが応答なしとしてマークされると、リフレッシュが可能なようにそのことが一般的なメカニズムによってリソース プロバイダに通知される。2 分間隔（これを過ぎると応答のないリソースが不健全を宣言される）などのハードコード化された定数はコンフィグレーション可能になった。
051631	prepared statement を `testConnectionOnRelease` と使用すると、Weblogic OCI XA ドライバで SQLException が送出される。
051673	ヒューリスティックな例外が、サーバが 1 つのコンフィグレーションと複数のコンフィグレーションで一貫性なく報告されることがなくなった。
053638	複数のサーバから同じデータベース インスタンスにアクセスすることで生じる Oracle XA ドライバの制限に解決策が用意された。
054513	トランザクション動作の管理が修正された。

その他

変更要求番号	説明
032447 および 054281	サーバが管理者によって意図的に停止されたときに停止に直接関連する「Emergency」、「Critical」、および「Alert」メッセージがサーバ ログに出力されなくなった。
047596	SHLIB 環境変数が定義されていない場合に、WebLogic Server が起動時にパフォーマンス パックを見つけられるようになった。
048222	コンソールのモニタ グラフが Netscape 4.75 で機能するようになった。
048524	基のカスタム LoginModule 実装で送出される例外に関係なく、LoginContext クラスで常に「Authentication Failed」というメッセージを伴って LoginException が送出されていた。LoginContext クラスでは、LoginModule 実装で送出される例外が伝播されるようになった。
049668	サービス パックのインストーラで空のディレクトリを削除できるようになった。
050345	i18n_user.properties ファイルまたは weblogic\msgcat\i18n.properties ファイルのいずれかの複数のバージョンが共存できるようになった。この機能により、メッセージ情報をより柔軟に配置したり、必要に応じてシステム メッセージを更新したりできるようになった。
050385	Jolt 接続プールに関して、接続プールのリセット オプションがコンソールに追加された。このオプションは、WebLogic Server 6.0 SP1 および SP2 でも利用できる。
050489	値を 2 つの引数に分割するパーサなしで、コメントの含まれる引数をスタートアップ クラスに渡せるようになった。
050517	Solaris 上の CORBA サーバに valuetype を渡すときに、OBJ_ADAPTER が送出されなくなった。
050888	サーバのクラスパスに格納されたクラスのネットワーク クラスロードが修正された。以前は、クライアントがそれらのクラスのオブジェクトをアンマーシャリングしている間にクライアントで ClassNotFoundException が送出された。
051297	アプリケーション パスが間違ってコンフィグレーションされている場合の ZipException メッセージが改善された。

052146	重みベースのロード バランシングに関する問題が修正された。
052349	WebLogic Server 5.1 SP7 以降から WebLogic Server 6.1 に移行すると発生していたパスワードに関する問題を解決した。以前は、`ldaprealm.properties` で LDAP をコンフィグレーションした場合は、`weblogic.properties` ファイルの変換を行った後で、LDAP サーバに対するパスワードを `config.xml` ファイルに手作業で追加する必要があった。この作業は必要なくなった。
052567	パフォーマンス パックが見つからない場合に使用される詳細なエラー メッセージが追加された。
052666	`WL_HOME\config` ディレクトリから起動される `startNodeManager.cmd` スクリプトに関連するエラー メッセージが修正された。エラー メッセージで正しいディレクトリが示されるようになった。
053308	`simpappcns` サンプルの Tuxedo サーバが起動するようになった。

053320	以下の 2 つのサンプルが削除された。
	`time.ServerTimer`
	`time.ClientTimer`
	WebLogic Time が非推奨になっている。代わりに `java.util.Timer` を使用する。

053422	コンソールの [デプロイ] チェックボックスに関する問題が修正された。このチェックボックスの選択を解除した場合にアプリケーションはデプロイされなかったが、次にサーバが起動したときにアプリケーションがデプロイされていた。
053426	変換ユーティリティで `production_apps` ディレクトリが作成および使用されなくなった。
053559	`weblogic.ant.taskdefs.ejb[20].DDInit` ツールによる Bean クラスの判別が向上した。
053584	スタンドアロン コンポーネントをコンフィグレーションするときにコンソールで使用する [Path] フィールドと [URI] フィールドが結合された。
053617	クライアントサイドのロールバックで、トランザクションのデータ構造が直ちに解放されるようになった。
053693	`weblogic.net` API が `javadoc.packages` でエクスポーズされるようになった。

053725	コンソールの SNMP エージェント画面で SNMPTrapDestinationMBean 型の [Targetted Trap Destinations] というフィールドが追加された。
053952 および 054777	コマンドラインから管理対象サーバを起動する際の問題が修正された。また、エラー ログ メッセージでエラーの発生元が現在の起動インスタンスなのかそれより前なのかが示されなかった。この問題は修正済み。
053955	%JAVA_HOME\bin が、startNodeManager.cmd スクリプトと startNodeManager.sh スクリプトの PATH 環境変数に追加された。これらの追加により、NodeManager がその PATH 変数に「java」のない状態で起動されている場合に、NodeManager を通じた管理対象サーバの起動が失敗しなくなった。
054131	ProxyServer（HttpClusterServlet）のあるクラスタ環境で、ProxyServer を通じてクライアントから呼び出しが行われた場合に、クラスタのノードが ProxyServer の詳細情報ではなくクライアントの詳細情報を受信するようになった。
054281	WebLogic Server の停止時に ListenThread が失敗しなくなった。
054517	サーバでユーザが開始したスレッドを通じて RMI 呼び出しを行う場合の HTTP トンネリングの問題が修正された。
054608	プロパティ SNMPAtributeChange および SNMPLogFilter に関して、コンソールのダイアログとテーブルに属性が追加された。この属性では、各プロパティでサーバを有効にするオプションが提供される。
054673	コマンドライン ユーティリティの snmpwalk コマンドが正しく機能するようになった。
054734	リソース アダプタで CONNECTION_ERROR_OCCURRED 例外が送出された後に接続プールで発生する問題が修正された。
054738	ear で、rar を格納するが、その rar の Java クラスは rar ではなく ear に存在する jar にパッケージ化できるようになった。classpath エントリが rar の Manifest.mf ファイルでサポートされるようになった。

054771	接続リークの検出が有効な場合に、weblogic-ra.xml デプロイメント記述子ファイルの connection-cleanup-frequency 要素と connection-duration-time 要素で指定された時間で ManagedConnections が自動的に破棄されていた。しかし、リソース アダプタそれ自体で ManagedConnection が現在アクティブであることを認識した場合に、ManagedConnection の破棄を中止する手段がなかった。 現在は、ManagedConnection.cleanup() メソッドの呼び出しから javax.resource.spi.IllegalStateException を送出することで、アクティブであることが認識された ManagedConnection の自動破棄をリソース アダプタから中止できるようになった。
054777 および 053952	コマンドラインから管理対象サーバを起動する際の問題が修正された。また、エラー ログ メッセージでエラーの発生元が現在の起動インスタンスなのかそれより前なのかが示されなかった。この問題は修正済み。
054795	サービス パックを含めた特定の製品またはリリースで、インストーラが 30 日の試用期間のみを許可するようになった。期間を延長するためのライセンスは、BEA の Web サイトから入手できる。
055090	WebLogic Server との様々な対話の中で、リソース アダプタが NoAccessRuntimeExceptions 例外を受信した。この例外により、リソース アダプタは様々な処理を実行することができなかった。これらのアクセスの問題は修正済み。
055121	コンバータ ツールで生成される起動スクリプトで、%CLASSPATH% が利用されるようになった。
055681	すべての nav.jsp ファイルで、ナビゲーション ツリーが UTF-8 エンコーディングで送信されるようになった。そうすることで、Solaris 上の JNDI ツリーを正しく表示することができる。
056481	コネクタの config.xml ファイルで Bean デプロイメント パスが修正された。connector.ConnectorEJBTest.testOpenConnection が失敗しなくなった。

プラグイン

変更要求番号	説明
047784	`iisforward.dll` ファイルが、同じ IIS インスタンスで複数の Web サイトをサポートするようになった。
050080	Apache プラグインの `PathPrepend` パラメータと `PathTrim` パラメータが、SSL 上で HTTPS を使用するときに同時に機能するようになった。
050253	ISAPI プラグインが、POST と PUT でコンテンツ長 0 を許可するようになった。
051912	Windows NT 上の Netscape Enterprise Server 3.5.1 の NSAPI プラグイン（`proxy35.dll`）で、`ns-httpd36.dll could not be found in the specified path` というエラー メッセージが出されていた。NES Server 3.5.1 にそのような DLL はないので、このエラー メッセージは削除された。
052444	`HttpClusterServlet` プラグインの使用時に、プライマリ サーバが停止した場合に、これまでより便利なエラー メッセージが表示されるようになった。
052480	ISAPI プラグインで、リクエストが完全に処理されていなくてもバッファがフラッシュされるようになった。
053773	WebLogic Server への接続時にプラグインがバインドする IP アドレスを指定するためのプラグイン オプションが追加された。
054140	すべてのプラグインで、デフォルト値が OFF の WLProxySSL という新しいパラメータが追加された。このパラメータは、プロキシが基づくユーザのブラウザで HTTPS が使用されるのかどうかを示す。プロキシは、`response.sendRedirect` の場合にユーザをリダイレクトする URL を判断する。
054526	キープアライブ接続で同じソケットを使用する場合に発生する問題が修正された。
054998	プロキシとしての IIS Web Server 5.0 を通じて WebLogic Server 6.1 からファイルをダウンロードする場合に発生する問題が修正された。`inetinfo.exe` がクラッシュしたことを通知するエラーが発生しなくなった。
055093	IIS プラグインで SSL を使用することから生じる問題が修正された。

RMI over IIOP

変更要求番号	説明
052086	EJB を再デプロイした後の IIOP を使用した外側への呼び出しで生じる問題が修正された。
052300	参照による呼び出しを妨げていた ear クラスローダの問題が修正された。
054182	6.1 リリースで IIOP/SSL の停止があった。この問題は修正済み。クライアントで SSL ソケット ファクトリにアクセスするためのコンビニエンス クラスが追加された。
054376	セキュリティ資格の伝播が IIOP でサポートされるようになった。
055330	リモート JVM で生成される IIOP JVMID 識別子が修正された。
055394	`weblogic.iiop.IIOPRemoteRef` の SVUID が修正された。
056505	AIX 上の IBM JDK で Any の valuetype をエンコードする際の問題が修正された。

サーブレットと JSP

変更要求番号	説明
C043505	Unicode 文字を含む URL が、誤って解釈されていた。コードを修正し、この問題を解決した。 http://dev2dev.bea.com/resourcelibrary/advisoriesnotifications/BEA00-09.jsp の「SECURITY ADVISORY (BEA00-09.00)」を参照すること。
043659	クラスタの WAP のセッション ID が短くなった。

049139	次の JSP でコンパイルが失敗しなくなった。
	`<%@ taglib uri="accents.tld" prefix="accents" %>`
	`<%@ page contentType="text/html; charset=ISO-8859-1" %>`
	`<accents:test>`
	`fail`
	`</accents:test>`

049147	デフォルト サーバを起動するときに、サーブレット 2.2 DTD を参照する警告メッセージが発生しなくなった。
050343	`ResponseHeaders.writeHeaders()` で、単一バイト文字列の場合のみ HTTP ヘッダが生成される。
050370	`PrefetchTag` クラスが `weblogic-tags.jar` ファイルに追加された。
050398	`weblogicx.jsp.tags.PrefetchTag` クラスが `weblogic-tags.jar` に追加された。
051396	起動時のロードで、無効な値が入力された場合に警告が生成されるようになった。
052109	`ServletSessionRuntimemBeanImpl.getMainAttribute()` を呼び出すたびに `NullPointerException` が送出されることがなくなった。
052134	異なる `ear` ファイルで同じ名前の 2 つの Web アプリケーションまたは `ejb-jars` を持てるようになった。以前は、Web アプリケーション `foo.war` ファイルの含まれている 2 つの `ear` ファイルをデプロイすると、`InstanceAlreadyExists` エラーが発生していた。
052665	Web アプリケーション デプロイメント記述子エディタからサーブレットの `<run-as>` 要素をコンフィグレーションする機能が追加された。

052786	例外型マッピングの実装で、例外型の完全一致が検索されなくなった。複数のクラスを階層で登録できるようになった。例外型マッピングでは最近の一致が選択される。
053314	インメモリ レプリケーションおよびメモリ セッション永続性のパフォーマンス最適化機能が追加された。
053333	Web アプリケーションで例外が送出される場合に JSP ページがエラー ページとして使用されるようになった。
053344	URL で複数のスラッシュを使用することで生じる問題が修正された。
053377	以下の場合に、例外 `weblogic.servlet.internal.session.MemorySessionDataMissing no-arg constructor for class` が送出されなくなった。 ● サーブレットから EJB メソッドが呼び出され、`HTTPSession` オブジェクトの参照が渡された。 ● サーブレットと EJB がデフォルトの Web アプリケーションにデプロイされた。 ● メモリ ベースのセッション永続性がサーブレットで使用された。
053573	`RequestDispatcher` の `Forward()` に対する複数の呼び出しがサーブレットのサービス メソッド内で行われた場合に、コンテナで無内容の応答が返される代わりに例外が送出されるようになった。
053634	forward または include で追加されたクエリ パラメータが、サーブレット仕様に従って既存のパラメータより優先されるようになった。

053722	絶対 URL を使用してクライアントをリダイレクトするスイッチが `weblogic.xml` ファイルで導入された。次に例を示す。
	`<container-descriptor>`
	`<redirect-with-absolute-url>false</redirect-with-absolute-url>`
	`</container-descriptor>`
	false に設定すると、絶対 URL が使用される。

053743	サーブレット仕様に従って、ServletA が ServletB にリクエストを転送した後に同じリクエストを JSP に転送しようとすると `IllegalStateException` が送出されるようになった。
053932	`secureProxy` が有効な場合に、`HttpClusterServlet` で Null ポインタ例外および `weblogic.utils.AssertionError` が送出されなくなった。

053959	BodyTags で、doStartTag メソッドから EVAL_BODY_INCLUDE を返すことができるようになった。
054016	Content-Disposition ヘッダが存在する場合の応答ヘッダ形式の破損の問題が修正された。
054194	応答バッファの内容が、setContentlength で指定された量が満たされた時点ですぐにフラッシュされるようになった。
054427	jnlp ファイルのマッピングが weblogic\servlet\internal\webapplicationServletContext.java ファイルに追加された。
054694	RequestDispatcher でサーブレット 2.3 フィルタが呼び出されるようになった。
055002	web.xml ファイルで MIME タイプを設定した後に、ファイルをブラウザで表示するだけでなくダウンロードできるようになった。
055106	POST リクエストの HttpServletRequest.getParameter (param) メソッドがサーブレットの doPost() メソッドで実行された後に、リクエストのパラメータが返されるようになった。
055241	Content-Type=text/xml で、メソッドが POST の場合に、request.getAttribute と request.setAttribute が無効にならなくなった。
055265	getServletContext().getRequestDispatcher と request.getParameter が引数で使用されるときに、新しいパラメータが失われたように見えなくなった。
055591	request.getParameterMap() メソッドが getParameter() メソッドで呼び出された場合に Null ポインタ例外が送出されなくなった。
055630 および 055962	ServletResponse の内部実装に関する問題が修正された。
056188	解析エラーが修正された。pageEncoding が省略された場合に JSP コンテナが解析に失敗していた。
056277	Netscape ブラウザを使用する場合に、POST で ArrayIndexOutOfBoundsException が送出されなくなった。
057022	Class-Path が使用される場合に、Win32 コマンドラインには長すぎるクラスパスが JSP コンパイラで生成されなくなった。

システム管理

変更要求番号	説明
045203	weblogic.Admin の使用時に不正なパスワードが原因で生成されるメッセージが、使用コマンドに関係なく統一された。
049177	クラスタでの Web アプリケーションのデプロイメントに関するエラーを解決した。
049178	デフォルト Web アプリケーションをデプロイするときに発生 するるエラーを解決した。
049919	管理サーバと管理対象サーバで、ServerConfigMBeans の余分なコピーが作成されなくなった。
050374	DeliveryMode=persistent の場合に、weblogic.properties ファイルの変換が失敗しなくなった。
050853	コンフィグレーションとデプロイメントが、間違ったパスと URI で行われることがなくなった。
050874	複製サーバを管理対象サーバとして起動できるようになった。
051618	weblogic.Admin が、-URL オプションで指定されたアドレスで動作しているサーバの MBeanHome から情報を取得するようになった。-URL オプションが指定されていない場合、weblogic.Admin はドメインの AdminHome を使用して情報を取得する。
052046	weblogic.Admin に対するコマンドラインで -password オプションを使用してパスワードが指定されていない場合、ユーザはパスワードの入力を求められる。
052095	管理サーバの起動スクリプトを実行すると、管理サーバは起動したが、動作中の管理対象サーバが検出されなかった。現在は、管理サーバの起動時には常に、properties ディレクトリからルート ディレクトリ (-Dweblogic.RootDirectory) をロードするか、またはデフォルト ディレクトリ（カレント ディレクトリ）を使用する。
052198	複製サーバが起動時に失敗しなくなった。
052261	weblogic.deploy ユーティリティを使用して複数の EJB ターゲットをデプロイする場合の問題を解決した。

052326	config.xml ファイルで間違ったルート ディレクトリが指定された場合、WebLogic Server は警告メッセージを発し、コンフィグレーションではサーバが起動したディレクトリが使用される。
052349	デプロイメント記述子コンバータを実行して、weblogic.security.realmClass=weblogic.security.LDAPRealm プロパティの設定されている weblogic.properties ファイルから config.xml ファイルを生成する際に、CustomRealm に対する ConfigurationData に加えて Password 属性が作成されるようになった。ConfigurationData と Password の値は両方とも、ldaprealm.properties ファイルの情報に基づく。config.xml が保存されると、Password は暗号化される。
052656	自動デプロイ ディレクトリのアプリケーションが 2 回デプロイされることがなくなった。
052753	管理サーバを起動してから、2 つの管理対象サーバを同時に起動しても、サーバの障害が発生しなくなった。
052804	weblogic.xml に <charset-params> エントリがある場合、デプロイメント記述子エディタにそれが表示されるようになった。以前は、表示できず、ClassCastException が発生していた。
053053	アンデプロイした後でメッセージ駆動型 Bean を再デプロイする際の問題を解決した。
053485	これは、WebLogic Server の旧リリースからの移行を行うユーザ、および移行チュートリアルに従うユーザに関係のある、変換ユーティリティについての既知の問題である。 変換プロセスによって weblogic.xml の compileCommand タグで自動的に指定される JDK は、JAVA_HOME¥javac に設定される。この設定から bin サブディレクトリが漏れることがなくなり、JAVA_HOME\bin\javac と設定されるようになった。変換ユーティリティの修正バージョンは、http://developer.bea.com/tools/techguides.jsp で公開されている。
053926	Administration Console を使用して src510sp10_117sj の weblogic.properties を src_ag_131sj の config.xml ファイルに変換するときに、ルート ディレクトリが変換されなかった。この問題を解決した。
053980	MBean API を使用する Web アプリケーションをクラスタにデプロイするときの問題を解決した。

053991	JMX で指定される MBean 名のパターンで、MBean サーバのクエリを実行し、MBean を検索できるようになった。
054173	クラスタ環境で複数の管理対象サーバを同時に起動する場合の問題を解決した。
054485	管理対象サーバ起動 URL のエラー メッセージの正確さを改善した。
055150	Sun の推奨に従って外部および内部クラスの serialVersionUID を修正した。
055945	MBean の参照を削除することで `ClassCastException` が発生していた。この問題を解決した。

Web サービス

変更要求番号	説明
048800	配列要素に関連する問題が修正された。配列要素が SOAP-ENV ネームスペースに存在しなくなった。.NET 配列は常にゼロ要素の非 NULL 配列になる。

51222	以下の行を Java クライアント プログラムに追加すれば、WebLogic Web サービスのクライアント API を使用して Microsoft がホストの Web サービスを呼び出せるようになった。
	`CodecFactory factory = CodecFactory.newInstance();`
	`SoapEncodingCodec codec = new SoapEncodingCodec();`
	`codec.writeQualifiedName( false );`
	`factory.register( codec );`

053249、053254	XML ドキュメントを送受信することによって静的な Java クライアントがメッセージスタイル Web サービスを呼び出した場合に、リテラル エンコーディング スタイルをサポートしていない Web サービスの WSDL が原因でエラーが発生することがなくなった。
058431、054687	プリミティブ整数で配列を使用することによって静的な Java クライアントからリモート プロシージャ呼び出し Web サービスを呼び出そうとしたときに Null 例外が送出されなくなった。

XML

変更要求番号	説明
046955	xsl:inlcude のある XML ドキュメントを Web アプリケーションからロードするときに、SAX 例外が生じていた。この問題は修正済み。xsl:include のパスはページ相対的でなければならない。
048165	全般的におよび特定のエンティティでキャッシュを回避できるようになった。
049063	Administration Console でパーサおよびエンティティの XML レジストリ エントリの作成後、パブリック ID、システム ID、ルート タグの各フィールドが読み込み専用になり、それらの値を変更することができない。これは正常な動作である。
049066	コンソールでの変更がエンティティ キャッシュに伝播されるようになった。
049082	ServerDebugMBean の属性を通じてデバッグがサポートされるようになった。
049855	XSLT タグ ライブラリを使用する JSP によってロードされるスタイルシートで使用される場合に、xsl:inlcude がインクルード スタイルシートの間違ったパスになっていた。この問題は修正済み。xsl:include のパスはページ相対的でなければならない。
050469	XML レジストリが管理サーバに割り当てられていない場合でも、管理サーバによって XMLRegistryEntryMBeans が新しい MBean 形式に正しく変換されるようになった。
050620	エンティティが解決されるときに、レジストリ エントリが解決のために 1 回、そしてキャッシュに入れるためのタイムアウト間隔を取得するために 1 回呼び出されていた。これらの機能は 1 回の呼び出しで実行されるようになった。
052258	スタンドアロン Java アプリケーション用のデフォルトの JAXP ファクトリが用意された。システム プロパティがコマンドラインで設定されていない場合は、weblogic ファクトリが Apache ファクトリの代わりにインスタンス化される。
054332	整形式の XML ドキュメントをデフォルト パーサで解析する場合に発生するエラーの原因だった問題が修正された。
054670	XML ファイルの解析で、ユーザが宣言されていないタグを指定した場合にパーサによって生成されるエラー メッセージが理解しやすくなった。

055499	weblogic.apache.xerces.dom.Node の cloneNode(true) メソッドがネームスペース属性（xmlns="http://www.abc.com/xml";）のあるルート要素で呼び出された場合に、間違った DOMException が送出されなくなった。

WebLogic Server 6.1 リリースのソリューション

以下の節では、WebLogic Server 6.1 で解決された問題について説明します。

- デプロイメント記述子エディタ
- EJB
- サンプル
- インターナショナライゼーション
- JCA
- JDBC
- JMS
- その他
- プラグイン
- サーブレットと JSP
- Web サービス
- XML

デプロイメント記述子エディタ

変更要求番号	説明
048382 および 049320	Administration Console のコネクタ デプロイメント記述子エディタの表示に関して、メニュー項目 [コネクタ記述子の編集] をクリックしたときの問題を修正した。
048965	Administration Console のデプロイメント記述子エディタで、[EJB]、[WebApp]、[WebServices] の各ノードを表示する際の問題を修正した。これらのノードをクリックすると、エラー メッセージが表示された。
048966	デプロイメント記述子エディタの [WebApp] ノードで [Security Role Assignments] を表示する際の問題を修正した。[Security Role Assignments] をクリックすると、`IllegalArgument` 例外が発生し、Web アプリケーションのエラー メッセージが表示された。
048971	既存のリソース参照インスタンスをクリックするときの、EJB 2.0 デプロイメント記述子エディタの問題を修正した。
048973	永続 EJB エンティティでの EJB デプロイメント記述子の問題を修正した。[Persistence] タブをクリックしたときに例外の発生することがあった。
049458	WebLogic RDBMS Bean、WebLogic RDBMS Jar、トランザクション アイソレーション、セッション Bean、およびエンティティ Bean の複製に関する問題を修正した。
049460	EJB デプロイメント記述子エディタには、新しい WebLogic EJB の説明フォルダの作成に関する問題があった。また、新しくコンフィグレーションされたり、複製されたりした WebLogic EJB に対して、新しいエンティティ記述子も、ステートレスセッション記述子も、ステートフルセッション記述子もコンフィグレーションできなかった。これらの問題は修正済み。
049413	各デプロイメント記述子エディタにおける永続性に関する問題を修正した。

EJB

変更要求番号	説明
042794	Administration Console の [クラスタ] パネルを使ってメッセージ駆動型 Bean をクラスタ全体にデプロイできなかった。この問題は修正済み。
046960	メッセージ駆動型 Bean（MDB）が関与する分散トランザクションが 2 つのサーバにまたがると、両方のサーバがハングする、という確認済みの問題を修正した。

サンプル

変更要求番号	説明
041463	UNIX の場合、`config/petstore/StartPetStore.sh` スクリプトが HP 上で Hotspot 仮想マシンを使用するように変更された。この変更は、HP クライアント VM のバグを回避するため。
047509	RMI サンプルを実行すると、デプロイ後にサーバを再起動する必要があった。再起動前に、サーバで `javax.naming.NameNotFoundException` が送出された。この問題は修正済み。
049011	`WEBLOGIC_HOME\samples\examples\xml\sax` にある `build.xml` ファイルの問題を修正した。
049264、048678	「サイン アウト」を行うと、正常に PetStore Demo からログアウトされるようになった。

インターナショナライゼーション

変更要求番号	説明
035036	WebLogic Server が起動し、起動アカウントのパスワードが要求される場合に、指示がインターナショナライズされた。
041021	l10ngen の実行後に一部のメッセージ カタログのプロパティ ファイルを生成できない問題を修正した。
041078	`english.xml` ファイルの一部の複製キーを修正した。
041264	特定のタグが `english.xml` に入っていないインターナショナライゼーションの問題を修正した。
041266	タイム スタンプがインターナショナライズされた。

JCA

変更要求番号	説明
049346	WebLogic Server は、javax.resource.spi.security.GenericCredential 資格インタフェースおよび Kerbv5 認証メカニズム タイプをサポートするようになった。以前は、デプロイされているリソース アダプタの ra.xml ファイルで <authentication-mechanism> に対してどちらの値を指定しても、デプロイメントが失敗していた。

JDBC

変更要求番号	説明
033423	`QueryDataSet` と OCI805_8 ドライバで dbKona 問題が発生していた。この問題は Windows NT 上では修正されたが、HP-UX および Solaris では未解決。
033920	接続プールにアクセスするテストを実行する場合に、JDBC の問題があった。この問題は修正済み。
035530	jDriver で OS 認証の有効化と Oracle 8.1.6/oci8 の使用に関して問題が発生していた。この問題は NT および HP-UX 上では修正されたが、Solaris では未解決。この問題を示す次のエラー メッセージが追加された。OS レベルの認証は OCI 8 ライブラリの欠陥が原因で現在はサポートされていない。
037589	Oracle Thin ドライバのサポートを追加し、コマンド ライン ヘルプを改訂した。
037591	ORACLE_THIN データベース タイプのサポートを追加した。使われなくなったデータベース タイプとコマンド ライン オプションを削除した。
037596	`PreparedStatement.setBlob` と `PreparedStatement.setClob` を修正した。
037693	`examples.dbkona.rowid` を実行する場合に Oracle ドライバと ROWID の選択に関する問題がなくなった。
037893	`getBytes()` が、Oracle LONG RAW カラムで長さゼロのバイト配列の代わりに、null を返すように修正した。
038143	`JDBConnectionPoolRuntimeMBean` が機能しなかった。`RuntimeMBean` 実装を `ConnectionPools` に追加した。
038229	jDriver for Oracle 8.1.6 の使用時に、管理対象サーバでメモリ不足が生じなくなった。
038275	JDBC エラー メッセージで `weblogic.classpath` を参照しなくなった。WebLogic 6.1 には `weblogic.classpath` がない。

040379	JDBC 接続プールの server プロパティと serverName プロパティに同じ値を指定すると、次のエラーが発生した。 <Cannot startup connection pool "jtaXAPool" server and serverName properties must have the same value> この問題は、XA ドライバに server または serverName プロパティを設定する必要がある場合に発生した。たとえば、WebLogic jDriver for Oracle を使用し、Oracle Thin ドライバまたは Cloudscape ドライバを使用しない場合に発生した。この問題は修正済み。
040509	Solaris 2.6 および jdriver に関する問題を修正した。
041286	接続プールのコンフィグレーションが正しくない場合のエラー メッセージを改訂した。
041302	MS SQLServer ドライバを更新した。
041514	コンソールで JDBC および接続プールをモニタできるようにした。
048764	Administration Console を使用した接続プールの終了が修正された。

JMS

変更要求番号	説明
039726	バージョン 6.0 より前の WebLogic Server がデータベースの処理中に停止された場合、バージョン 6.1 へのアップグレード後、一部の恒久コンシューマが、恒久サブスクリプションの作成後にトピックに送信されたメッセージを受信することがあった。データベースを定期的に更新するスカベンジャと、〔Ctrl〕+〔C〕を押してバージョン 6.0 より前のサーバを停止する処理の競合状況が発生していた。この問題は修正済み。
039809	clearBody() を呼び出す前にレシーバがメッセージを変更できないように、TextMessage、ObjectMesssage、MapMessage、および XMLMessage の JMS メッセージ タイプが読み込み専用モードで実装されるようになった。
041031	JMS フロー制御で、コンシューマからの非同期メッセージが滞留しなくなった。
041165	ユーザが JMS ObjectMessage を受信し、getObject を呼び出す前に toString メソッドを呼び出すと、メッセージが自動的にデシリアライズされるようになった。
041821	クライアントが正常に終了しないで切断した場合、JMS 接続がクリーン アップされるようになった。
042096	MDB で setRollbackOnly を呼び出しても、冗長なログ メッセージが生成されなくなった。
042458	JMS で外部プロバイダから受信するメッセージの送信を処理できるようになった。
042792	JMS で JMSServerSessionPool を使用して、複数のメッセージを同時に処理できるようになった。
042997	マルチ CPU マシンで、サーバ セッション プールのデッドロックが発生しなくなった。
043077	メッセージ リスナによる、サーバ セッション プールでのメッセージの自動確認応答が失敗しなくなった。
043155	JMS トピック メッセージの有効期限が切れるようになった。メッセージがコンシューマに渡されるときにチェックされるようになったため。
043223	JMS で 32,000 文字より多い TextMessage を送信できるようになった。

043225	setObject を実行すると、JMS ObjectMessage が不必要にアンシリアライズされなくなった。
043445	Fiorano 5.0 を使用して WebLogic Server JMS メッセージを送信しても、例外が発生しなくなった。
043447	JMS サーバ セッション プールが終了すると、そのサーバ セッションがクリーンアップされクローズされるようになった。
043519	セッションのロールバックおよび回復による、非同期コンシューマのメッセージ受信に対する妨害がなくなった。
043752	期限の切れたメッセージを回復しても、NullPointerException が発生しなくなった。

043884	ファイル ストアへの書き込みを同期させないことによって、JMS のパフォーマンスが向上した。同期書き込みは、プロパティを使用するコマンドラインを介してコンフィグレーションでき、デフォルトでは有効になっている。
	`Dweblogic.JMSFileStore.SynchronousWritesEnabled=(false|true)`
	`Dweblogic.JMSFileStore.<store name>.SynchronousWritesEnabled=(false|true)`
	最初のプロパティは、サーバで動作しているすべての JMS ファイル ストアに適用される。2 番目のより詳細なプロパティは、最初のプロパティに優先する。プロパティの指定順は重要ではない。

043973	メッセージがトランザクションの一部として受信されたり、セッションでまだ確認応答されていなかったりする場合に、`TopicSesssion.unsubscribe()` が拒否されるようになった。
044200	JMSConnection Mbean destroy() 関数をサーバ側で実行すると、接続を破棄できるようになった。接続の破棄後は、コンシューマでメッセージを受信でき、プロデューサでメッセージを送信できる。
044285	十分なサーバ スレッドがなくても、JMS 永続性メカニズムがロックしなくなった。
044450	JDBC ストアのコードで、不必要に「COMMIT WORK」が生成されなくなった。
044605	未送信 JMS メッセージのシリアライゼーションで `NullPointerException` が生成されなくなった。

044891	JMS トランザクション セッションで `Session.close()` が、進行中のトランザクションをコミットせず、ロールバックするようになった。
044961	名前の値が見つからない場合、`getBooleanProperty()` が正しく false を返すようになった。その他すべての getProperty メソッドは、値が見つからない場合、NumberFormatException を返す。
044976	`MapMessage` の `getXXX()` からの戻り値および例外が、以下のように正しい値を返すようになった。 フィールドの値が存在しない場合、`getByte()` は NumberFormatException を返す。 フィールドの値が変換できない場合、`getByte()` は MessageFormatException を返す。 フィールドの値が存在しない場合、`getBoolean()` は false を返す。 フィールドの値が変換できない場合、`getBoolean()` は `MessageFormatException` を返す。
045531	JMS 恒久サブスクリプションがコンフィグレーションされるときにストアがコンフィグレーションされていなくても、NullPointerException が発生しなくなった。
045956	デフォルトのファクトリを使用するサーバ セッション プールの作成時に、接続コンシューマが適切に動作するようになった。
045963	接続コンシューマで、すべての選択されたメッセージを無条件で受信できるようにする `maxMessages` フィールドが無視されないようになった。
045980	最初の SQL クエリの失敗に対して、詳細なログ メッセージが追加された。ログ メッセージには、発生が予想される問題と回避策が記載されている。回避策を使用することで、使用中の接続プールで指定されたテーブルにアクセスできるようになる。また、JMS JDBC ストアで、スキーマ名およびカタログ名を含むテーブル名のプレフィックスをコンフィグレーションすることによって（[schema.[catalog.]]prefix）、テーブル名が完全に修飾されるようになった。
046705	サブスクライバでのメッセージ リスナの設定時に、JMS で設定時のコンテキスト クラスローダを記憶できるようになった。その結果、そのメッセージ リスナで onMessage を呼び出すときに、適切にコンテキスト クラスローダを設定できるようになった。これにより、EJB ではその EJB にのみ存在しているクラスを含む Web アプリケーション サブスクライバにメッセージを送信できるようになった。

046726	新しい動作または古い動作を指定できる「確認応答」用の新しいコンフィグレーション オプションが追加された。このオプションは、JavaSoft によって「確認応答」の意味の変更が提示されたために追加された。変更は、次のような、JMS 1.0.2 仕様での記載に基づいており、Javadoc で指定されている意味とは異なっている。 「セクション 4.4.13: 消費されたメッセージの確認応答を行うと、そのセッションによって配信されたすべてのメッセージ受信の確認応答が自動的に行われる。」
047424	JMS では、メッセージ サイズが割り当てサイズを超過しても NullPointerException が送出されなくなった。
048181	一時的なトピックの作成後に接続をクローズしても、トピック名のルックアップ時に次のような例外が送出されなくなった。 `weblogic.jms.common.JMSException: Error unregistering RuntimeMBean.`
049518	WebLogic Server 6.0 で生成されたテキスト メッセージを WebLogic Server 6.1 に移行する場合の問題を修正した。メッセージが受信されず、次のメッセージが表示されていた。 `The corresponding file block will be zeroed out on disk, and this corruption will no longer be reported in the future. java.io.StreamCorruptedException: Unknown object stream version.`

その他

変更要求番号	説明
035039	WebLogic Server で起動パスワードを要求する場合に、そのことがログ（`security.xml`）に書き込まれるようになった。これにより、マシンが「ハング」している理由がリモート マシンの管理者にわかる。
036141	ACL で JNDI ルックアップを実行するために `guest` が無効になっている場合は、管理対象サーバが起動しなかった。この問題は修正済み。
036604	メソッド `ClusterMBean.setDefaultLoadAlgorithm()` は不正な値を受け付けず、適切な `IllegalArgumentException` を送出するようになった。
036986	WebLogic Server の XA ドライバを CMP で使用する場合に、デプロイ時に障害が発生しなくなった。
038883	ルート ディレクトリが `\config` ディレクトリの下にない場合、サーバの起動時にルート ディレクトリを指定できなかった。この問題は修正済み。
039066	システムの起動時に要求されたパスワードの入力を間違えた場合でも、サーバが起動しようとした。最終的にサーバは起動しない。現在は、パスワードが間違っている場合には再入力が要求されるようになった。
039182	同じインタフェースを実装する 2 つのクラスタ対応オブジェクトを 1 つのサーバ上で同じ JNDI 名にバインドした場合、例外が送出されず、最初のクラスタ対応オブジェクトがバインドされた。この問題は修正済み。
039249	Administration Console を使用して JMSServer をモニタするときに、管理対象サーバでメモリ リークが発生した。この問題は修正済み。
039361	RDBMS レルムでグループを作成した場合に、そのグループを削除できるようになった。
039891	サーブレットの初期化時にサーバでコンテキスト属性が正しく設定されるようになった。
039960	ログ ビューアが US 以外のロケールで正しく動作するようにした。
039995	Administration Console を使用して WLEC 接続プールを作成した後に、サーバの起動エラーが発生しなくなった。

040168	インターナショナライゼーションのロギング サンプルは、パス内でドット（.）が見つかると失敗した。この問題は修正済み。
040177	WebLogic Server 6.0 で使用するために `weblogic.properties` ファイルを変換する場合には、さらにユーザ情報が必要とされた。この点は製品のマニュアルに追加された。
040178	変換ユーティリティを使用するときに、`weblogic.properties` ファイルにプロパティ `weblogic.httpd.register.`*`Name`*`=weblogic.servlet.FileServlet` が入っておらず、サーブレットに初期引数が関連付けられていない（同じ `weblogic.properties` ファイルで定義されたプロパティ `weblogic.httpd.initArgs.file=`*`arg1=value`* を持っていない）場合は、変換ユーティリティが Null ポインタ例外を送出した。この問題は修正済み。
040213	管理対象サーバのリスンポートを管理サーバのリスンポートと同じポートにできない。たとえば、管理対象サーバと管理サーバのどちらもホスト番号が 7001 という状態でローカルホストまたは 127.0.0.1 を使用することはできない。より説明的で理解のしやすいエラー メッセージが追加された。
040297	`application.xml` で設定された属性が管理サーバから管理対象サーバに伝播されないバグを修正した。
040405	`config.xml` ファイルが Apache XML シリアライズ クラスを使用してシリアライズされるようにした。これにより、組み込みエンティティのエスケープやエンコーディングを指定した XML ヘッダなど、XML が正しく生成されるようになった。使用されるエンコーディングはプラットフォームのデフォルト値。また、「&」文字に関する問題も修正された。
040659	`getParameter()` エンコーディングを iso-8859-1 として指定した場合、POST メソッドが正しく動作するようになった。
040688	クラスタを構成するサーバをメール セッションのターゲットに指定すると、メール セッションがシリアライズ可能でないため、例外 `java.io.NotSerializableException: javax.mail.Session` が送出された。この問題は修正済み。
040698	ポーラーがまだコピー処理中のアプリケーションをデプロイしようとしても、アプリケーション マネージャで不必要な例外が送出されなくなった。
040899	変換ユーティリティが起動 `weblogic.properties` を `.xml` に変換する場合のスタートアップ クラスに関する問題を修正した。

040950	JTA での問題を修正した。XA 接続に関して、接続プールのテスト接続ロジックでは、分散トランザクションを起動してから SQL のテストを実行しようとしていた。これが `javax.transaction.NotSupported` 例外を生成する可能性があった。
041157	MBean 名に漢字が使われていた場合、コンソールのダイアログ タブが正しく動作しなかった。
041290	ユーザ名プロパティを `weblogic.Admin` または `weblogic.Server` コマンド ラインから設定する場合、そのユーザ名は system とする必要がある。ただし、ユーザのスクリプトを壊さないために、ユーザ名の構文に従う。`weblogic.management.password` プロパティ値はシステム パスワードでなければならず、ユーザ名は `config.xml` ファイルでは変更できない。
041662	JTA を改良するため、JNDI でリソースを公開するようにした。リソースはトランザクションの進行に伴って動的に検出されたが、どのリソースがどのサーバでサポートされているかについても検出できるようにした。
041757	JTA 関連の問題を修正した。リソースの公開（JNDIAdvertiser）が呼び出し側の ID で正しく行われなかった。この処理ではシステム ID を使用するようにした。
041845	NTRealm を更新して、全般的な操作性と機能性を向上させた。
041854	WebLogic Server で Java WebStart をサポートするために、この MIME タイプに合わせてコンフィグレーションされた `web.xml` を Web アプリケーションに提供するようにした。その他の MIME タイプのマッピングも提供している。
041995	アクティブ サーバを再開した場合にコンソールで予期しないエラーが表示される問題を修正した。また、ユーザを追加する場合にコンソールに表示される無関係なメッセージを削除した。
042183	`config.dtd` をコンソールで生成する場合に、エラーが発生した。この問題は修正済み。
042362	JTA 仕様のマニュアルでは `setTransactionTimeout()` について、次のように記述されている。 「seconds: タイムアウト値を秒単位で表した値。値がゼロの場合、トランザクション サービスはデフォルト値を復元する」 値がゼロの場合、WebLogic Server はデフォルト値をリセットしなかった。この問題は修正済み。

042794	Administration Console の [クラスタ] パネルを使ってメッセージ駆動型 Bean をクラスタ全体にデプロイできなかった。[サーバ] パネルを使用して、各サーバを対象として個々に指定する必要がある。
043369	ドメイン全体にデプロイされたアプリケーションで、元のパッケージ化を保持できるようになった。つまり、アプリケーションが管理サーバで展開されても、管理対象サーバで展開されたままの状態になる。
043388	WebLogic Server のクラスタでは、共有ネットワーク ドライブを使う必要がない。WebLogic Server とアプリケーションの両方をローカル ファイル システムにインストールできるようになった。
049417	ベータ版に付属のパスワード変換ツールが修正され、すべてのファイルに対する SAX パーサ エラーが返されなくなった。

プラグイン

変更要求番号	説明
038831	高負荷の状態でも Apache プラグインに障害が発生しなくなった。
040817	JavaBean を呼び出す通常の JSP ページを閲覧および呼び出しを行うために IIS を使用したり、ResultSet 参照を取得したりする場合、FORM POST メソッドを使って提出するときに問題が発生した。HTML ストリームが IIS に戻るときに切り捨てられた、つまりページが部分的にしか描画されなかった。この問題は修正済み。
041332	プラグインを使用する場合、`MatchExpression` にカンマで区切った複数の式を格納できるようになった。
041534	`mod_wl.so` を使用する場合、`Debug` と `DebugConfigInfo` が `httpd.conf` に入っていないと Apache はプロキシ処理を実行しなかった。この問題は修正済み。
041580	Apache プラグイン `MatchExpression` は、エンコードされた URL を使った場合でも機能するようになった。
041745	信頼された CA ファイルが指定されていない場合に、Isapi プラグインで SSL を使用するときのログ メッセージを追加した。
041754	ISAPI が SSL 対応でコンフィグレーションされているにもかかわらず、非セキュア ポートを使用した場合に、無限ループに入ることがないように修正した。
041939	Apache プラグインがヘッダ `"transfer-encoding"="chunked"` をフィルタ処理しなかった。そのため、ブラウザがこれらの応答を処理できず、Netscape はハングし、IE はエラー ページを表示した。この問題は修正済み。
042482	NSAPI および Apache プラグインで、HPUX-11 上で READ タイムアウト設定がなくなった。
044988	プラグイン側でソケットが適切にクローズされない、という NSAPI プラグインに関する問題を修正した。Web サーバ側のクローズされないソケットは CLOSE_WAIT 状態のままになる。
046018	Apache 2.0 に対応した新しいバイナリ ファイルが追加された。

046392	Apache プラグインの使用時に、異なる <Location> ブロックに対して異なるクッキー名を設定できるようになった。
046423	Weblogic Server で Apache に対応した静的ライブラリがサポートされるようになった。
046424	Apache 2.0 プラグインに対応した、`KeepAliveEnabled` パラメータおよび `KeepAliveSecs` パラメータが追加された。
046546	永続セッションに対してクッキー名をコンフィグレーションできるようになった。`weblogic.xml` に新しいパラメータ PersistentStoreCookieName（デフォルト =WLCOOKIE）が追加された。
046946	クラスタ ノードが、そうでないのに、自身がクラスタ内の唯一のメンバーであるかのように動作してしまう、WebLogic Server のデプロイメントに関する問題を修正した。`ListenDelaySecs` を使用すると、他のメンバーが見つかるまで HTTP リクエストの処理を遅らせることができるようになった。次に例を示す。 `<Server> ListenDelaySecs="30" </Server>` 新しいノードで他のクラスタ メンバーの存在を確認する時間を持てるように、HTTP リスン スレッドをコンフィグレーション可能な時間、遅らせる。
046956	Apache Stronghold/Raven プロキシで、`DefaultFileName` および `VirtualHost` によって引き起こされる問題が解決された。
047987	SSL を使用してブラウザからプロキシへ、およびプロキシからサーバへ大量のバイト数（500,000 バイト）をストリーミングする場合の NSAPI に関する問題が解決された。
047636	WebLogic Server で、Apache 1.3.X に対応する静的にリンクされたモジュールと DSO がサポートされるようになった。

サーブレットと JSP

変更要求番号	説明
026488	メソッド `println(String)` のパラメータが null だった場合のサーブレットに関する問題を修正した。
029989	`UnavailableException` がサーブレット内で生成された場合に、間違ったエラー メッセージが送信されなくなった。

030880	デバッグを有効にした場合の JSP のコンパイルに関する問題を修正した。コンパイラ フラグを weblogic.xml に指定できるようになった。次に例を示す。
	`<!-- weblogic.xml: weblogic-specific web app descriptor -->`
	`<weblogic-web-app>`
	`<session-descriptor>`
	`</session-descriptor>`
	`<jsp-descriptor>`
	`<jsp-param>`
	`<param-name>compileCommand</param-name>`
	`<param-value>jikes</param-value>`
	`</jsp-param>`
	`<jsp-param>`
	`<param-name>compileFlags</param-name>`
	`<param-value>+E -nowarn</param-value>`
	`</jsp-param>`

037771	再ロードする場合の JSP のチェックを最適化した。
038739	JSP のデバッグのサポートを改良した。
039083	`RequestDispatcher` と JSP ページの転送が正しく機能するようになった。
039193	ユーザがサーブレット クラスをパブリックにしなかった場合、紛らわしく参考にならないエラーが送出された。こうしたエラーは発生しなくなった。
039530	リリース時に状態をリセットするように内部サーブレット タグを修正した。同じタグを 1 つのページで複数回使った場合、2 回目以降のタグは前のタグのデータを保持していた。
040171	Administration Console で `SingleThreadedServletPoolSize` 属性を設定できるようになった。

040338	動作中のすべてのサーブレットが、コンソールの [mydomain ｜ Web アプリケーション ｜ DefaultWebApp_myserver ｜ ActiveServlets] ウィンドウに表示されるようになった。
040662	JSP ファイルにページ タグがあった場合、JSP コンパイラがクラスまたは java ファイルを生成しなかった。この問題は、コマンド ライン `jspc` を使った場合にだけ発生した。この問題は修正済み。
040766	`response.sendRedirect` を実行する場合に相対 URL として ../ を使用したときのサーブレットの問題を修正した。
040814	サーブレット ユーザのエラーで例外が発生した場合に、わかりやすいエラー メッセージが表示されるようにした。
040843	仮想ホストを使用して Web Application Components をデプロイする場合の問題を修正した。
040981	パイプ文字を使用した WAP およびセッション クッキーに関連する問題を修正した。
041234	POST および HTTPS を使用する JSP/ サーブレットでの問題を修正した。
041478	セッション クッキーのフォーマットを変更することで、URL 書き換えに関する問題が修正された。
041605	JSP コンパイラのエラー ページで、javac から文字化けしたエラー メッセージが表示されなくなった。
041684	サーブレットでの問題を修正した。XML コンテンツの処理中に、XSLT タグによって SAXParseException が送出されていた。
041729	`weblogic.servlet.jsp.Precompiler` が、コンパイル対象の JSP の `workingdir` や `packagePrefix` などの `weblogic.xml` パラメータを受け付けるようになった。
041945	`web.xml` に設定 `inputCharset./A*=Shift_JIS inputCharset./B*=EUC-JP` が入っており、2 つのクライアントが同時に JSP/ サーブレットにアクセスした場合、WebLogic Server はこれらの文字列を正しく処理できなかった。一方のクライアントは正しい結果を得たが、他方は得られなかった。この問題は修正済み。

042031	Web アプリケーション データ テーブルで、任意の Web アプリケーションのモニタ アイコンをクリックしてから、[サーブレット] カラムのモニタ リンクをクリックした場合、正しい見出しが表示されるようになった。
042293	サーブレットの URL 書き換えは IE 5.0 で正しく動作したが、Netscape 4.7 では動作しなかった。この問題は修正済み。
042509	2 つの Web アプリケーション間で両方で認証を使用して転送する場合の問題が修正された。セッション データがアプリケーション間で不適切に共有されることがなくなった。
044873	`weblogic.jspc` を使用して JSP を手動でコンパイルする場合に、`-version` 引数が機能するようになった。
044926	WebLogic Server 6.1 では、JSP から 2 バイト コードを含む Java コードが適切に生成されるようになった。
049760	フォーム検証タグ用の JSP タグ ライブラリを含む `weblogic-vtags.jar` ファイルが、ドキュメントの記載どおり `ext` ディレクトリに追加された。

Web サービス

変更要求番号	説明
046032、046808	WebLogic Web サービスをホストする WebLogic Server インスタンスのホストおよびポートをデフォルトで動的に設定する、WebLogic Web サービスの WSDL が JSP によって生成されるようになった。wsgen Ant タスクを使用して Web サービスをアセンブルするときに、値を明示的に指定することによって、WSDL でホストおよびポートをハード コード化することを指定できる。
046215	wsgen Java Ant タスクを使用して、WebLogic Web サービスをアセンブルする場合、Web サービスの呼び出し時に（デフォルトの HTTP ではなく）HTTPS を使用することを指定できるようになった。
047794	wsgen Java Ant タスクを使用して、WebLogic Web サービスをアセンブルする場合、生成されたエンタープライズ アプリケーション アーカイブには、Web アプリケーション ディレクトリも Web アプリケーション *.war ファイルも含まれず、*.war ファイルだけが含まれるようになった。
048797	WebLogic Web サービスが、適切に Microsoft SOAP ツールキットの配列タイプと相互運用できるようになった。

XML

変更要求番号	説明
034785	WebLogic Server で、XML レジストリでのネームスペースがサポートされるようになった。
034785	localRoot、qualifiedRoot、namespaceURI など、ドキュメントに関するネームスペース情報を取り出すメソッドが提供されるようになった。XML ドキュメントに定義されたネームスペースがある場合には、レジストリでのルックアップは限定名に基づく。
036428	XML サブシステムの以前のリリースには、`xmlx.jar` ファイルに含まれる `javax.xml.*.parse(File)` メソッドの Apache の実装にエラーが含まれていた。現在では WebLogic Server にこの JAR ファイルが含まれておらず、WebLogic Server 独自のメソッド実装が正しくなっているため、この問題は現在のリリースには適用されない。
036642	作成されたカスタム パーサが、属性とタグ本体に埋め込まれた改行文字をスペースに変換しなくなった。 **注意：** カスタム パーサの作成機能は、WebLogic Server のこのリリースでは非推奨になっている。代わりに、WebLogic Server の高性能パーサを使用する。
039096	XML レジストリに新しく追加されたエントリを認識するために WebLogic Server を再起動する必要がなくなった。
039247	カスタム パーサで解析するコンフィグレーションの XML ドキュメントを、サーブレット API の `setAttribute()` メソッドを使用して解析する場合、カスタム パーサが正しく呼び出されるようになった。
040479	`config.xml` ファイルに XML が正しく書き込まれなかった `JMSHelper` のバグを修正した。
041540	WebLogic サーバ付属の Xerces パーサのシリアライザで、`Shift_JIS` や `EUC-JP` などのマルチバイト エンコーディングがサポートされるようになった。
041541	WebLogic サーバ付属の Xerces パーサのメソッド `apache.xml.serialize.BaseMarkupSerializer.printEscaped()` で、`Shift_JIS` や `EUC-JP` などのマルチバイト エンコーディングがサポートされるようになった。

041734	WebLogic Server の JAXP 実装で、特定のパーサに解析を委託されようとしている XML ドキュメントを格納する場合に入力ストリームがリセット可能である必要がなくなった。
041778	XML レジストリに正しく登録されている外部エンティティを格納する XML ドキュメントを解析する場合に、`NullPointerException` エラーは送出されなくなった。
041797	新しいサーバを作成した後に、サーバの再起動が失敗することがなくなり、`config.xml` ファイルに不正な解析済みヘッダが含まれることがなくなった。
46951	`org.xml.sax.InputSource` クラスの WebLogic Server 拡張機能の一部である、メソッド `weblogic.xml.sax.XMLInputSource.getNamespaceURI()` が現在のリリースで実装された。
48809	SAX バージョン 2 サーブレット属性からのドキュメントの解析時に、`SaxParserFactory` が適切に選択されるようになった。
48810	パーサのクラス名をパブリック ID にマッピングすることで、特定のドキュメント タイプ用のパーサをコンフィグレーションできるようになった。特定のドキュメント タイプ用のパーサをコンフィグレーションするときのパーサ クラス名の指定は、このリリースで非推奨になっている。代わりに、ファクトリ名を指定する。