Oracle9*i* Application Server for Windows and UNIX

Forms アプリケーション Web 利用ガイド

リリース 1.0.2

2000 年 11 月
部品番号 : J02409-01

ORACLE

Oracle9*i* Application Server for Windows and UNIX
Forms アプリケーション Web 利用ガイド , リリース 1.0.2

部品番号 : J02409-01

原本名：Oracle9*i* Application Server Release 1.0.2:
Deploying Forms Applications to the Web for Windows and UNIX

原本部品番号：A86783-01

原本著者：Tony Wolfram, Cathy Godwin

原本協力者：Tom Haunert, Joan Carter, Poh Lee Tan

目次

6 Web へのフォームの配置

7 アプリケーション設計に関する考慮事項

8 これまでのアプリケーション資産の Web への移行

11 パフォーマンス・チューニングに関する考慮事項

12 ロード・バランスに関する考慮事項

13 Oracle Enterprise Manager での Forms のサポート

第 II 部　付録

A　Forms Services のパラメータ

B　クライアント・ブラウザのサポート

C Java Importer

はじめに

対象読者

このマニュアルは、Oracle9*i* Application Server を用いた Forms アプリケーションの Web への配置に関心のあるソフトウェア開発者を対象にしています。

構成

このマニュアルには、次の章と付録が含まれています。

第 1 章　概要

アプリケーションを Web に配置する利点について説明します。

第 2 章　Forms Services の概要

Forms Services のアーキテクチャとそのコンポーネントの概要を説明することで、使用する配置ツールを紹介します。

第 3 章　構成時の選択肢の概説

アプリケーションを Web に配置するときの構成の選択肢を概説します。

第 4 章　Forms Services のインストール

Oracle Universal Installer を使用した Forms Services のインストールについて説明します。

第 5 章　Forms Services の構成

Forms Services をサポートするために、ネットワーク環境を手動で構成するときに必要となるステップについて説明します。

第 6 章 Web へのフォームの配置

実行可能ファイルの作成やアプリケーションの URL のブロードキャストなど、アプリケーションを Web に配置するために必要なステップについて説明します。

第 7 章 アプリケーション設計に関する考慮事項

Web へ配置する Forms アプリケーションを設計するためのガイドラインとヒント、および機能制限が記載されています。

第 8 章 これまでのアプリケーション資産の Web への移行

クライアント / サーバー・ベースの実装またはカートリッジ実装から Web ベースの Forms Services 実装へと、現在のアプリケーションを移行する場合のガイドラインが記載されています。

第 9 章 ネットワークに関する考慮事項

Web アプリケーションを配置できるネットワーク・インプリメンテーションと、各タイプに Web アプリケーションを配置するときに考慮する必要のある事項について説明します。

第 10 章 セキュリティに関する考慮事項

Forms Services をネットワーク環境で設定するときに考慮する必要のある、一般的なセキュリティ問題について説明します。

第 11 章 パフォーマンス・チューニングに関する 考慮事項

Forms Services を使用してインターネットまたはその他のネットワーク環境にアプリケーションを配置するときの、チューニングに関する考慮事項を説明します。

第 12 章 ロード・バランスに関する考慮事項

サーブレットのロード・バランスを使用してロード・バランスを行う技法について説明します。

第 13 章 Oracle Enterprise Manager での Forms のサポート

Oracle Enterprise Manager（OEM）システム管理ツールについて説明します。

第 14 章 キャパシティ量計画の考慮事項

Forms Services の拡張性機能について考察します。

第 15 章 トラブルシューティング・ソリューション

Forms Services のトラブルシューティング・ソリューションに関する情報が記載されています。

付録 A　Forms Services のパラメータ

Forms Services の構成に使用するパラメータについて説明します。

付録 B　クライアント・ブラウザのサポート

ネイティブ JVM を使用する Internet Explorer 5、Oracle JInitiator または AppletViewer をユーザーの Web ブラウザとして使用する際の利点を説明します。

付録 C　Java Importer

フォーム開発者を対象として、Java クラスにアクセスするための PL/SQL パッケージを Java Importer を使用して生成する方法、さらに生成された PL/SQL を Forms アプリケーション内で使用してプログラムを作成する方法を説明します。

関連マニュアル

詳細は、次のマニュアルを参照してください。

- Oracle Forms Developer リリースノート、リリース 6*i*
- Oracle Forms Developer for Windows/NT スタート・ガイド
- Oracle Reports Developer パブリッシング・レポート
- Oracle Forms Developer and Oracle Reports Developer アプリケーション作成ガイド
- Oracle Forms Developer Form Builder リファレンス

第I部

Forms アプリケーションの Web への配置

1

概要

1.1 インターネットがすべてを変えた

インターネットによって、企業は、内部のビジネス・プロセスをカスタマイズしたり合理化するための多くの機会に恵まれるようになりました。しかし、これらの機会は新しい課題をもたらします。たとえば、アプリケーションを作成する場合は、インターネット上で利用できるものを迅速に作成する必要がありますし、膨大な数のユーザーを扱えるように拡張する必要もあります。

中間層アプリケーション・サーバーを使用すると、拡張可能なインフラストラクチャ、トランザクション管理、ポータル・サービス、ビジネス・インテリジェント機能の提供およびそれらの統合により、開発に要する期間と費用の削減が期待できます。

しかし、ベンダーのアプリケーション・サーバー製品のほとんどは、残念ながらこれらのサービスのサブセットのみを対象としています。そのため、顧客は複数ベンダーの製品を統合して使用することになり、これが混乱の原因となっています。

1.2 Oracle インターネット・プラットフォーム

オラクル社では、次の 3 つのコア製品から成る、シンプルで完全かつ統合されたインターネット・プラットフォームを提供することにより、素晴らしく成長するためにお客さまが明確な方針を見つけられるよう支援します。

- Oracle8*i* データベース
- Oracle9*i* Application Server
- Oracle9*i* Developer Suite

企業の IT インフラストラクチャを所有する際の総コストを低減するために、Oracle のインターネット・プラットフォームが提供する方法の 1 つは、様々なベンダー製品の統合に要する費用をなくすことです。Oracle インターネット・プラットフォームを使用することにより、お客さまは真の付加価値サービスを作成することに IT リソースを集中させることができ、延いては競合他社との差別化を図ることができます。

Oracle9*i* Application Server では、Oracle8*i* と Oracle9*i* Developer Suite を組み合せることで、インターネット・アプリケーションの作成、配置および管理に必要となるあらゆるものが提供されます。また、これらの製品は、シンプルで完全かつ統合されたインターネット・プラットフォームを構成します。

1.2.1 容易性

Oracle9*i* Application Server、Oracle8*i* および Oracle9*i* Developer Suite は、入手、インストールおよび管理が容易です。Oracle9*i* Application Server Forms Services および Oracle Forms Developer を含む Oracle のコア中間層サービスとコア開発ツールは、すべて統合されています。アプリケーション・サービスは、Oracle9*i* Application Server として統合されています。また、開発ツールは Oracle9*i* Developer Suite として統合されています。これらの統合により、ポータル、トランザクション・アプリケーションおよびビジネス・インテリジェント機能の作成と配置が、3 つの製品のみでできます。

1.2.2 完全性

データの管理に Oracle8*i* を、アプリケーションの作成に Oracle9*i* Developer Suite を、さらにそれらを実行するのに Oracle9*i* Application Server を使用することで、あらゆるタイプのアプリケーションの作成と Web への配置に対し、Oracle インターネット・プラットフォームは完全なソリューションを提供します。これらの Oracle ツールによって、拡張と高度な利用が可能なインフラストラクチャが提供されるため、お客さまはユーザー数の増加に容易に適応できます。

1.2.3 統合性

Oracle9*i* Application Server は、Oracle8*i* データベース、および Oracle 開発ツールで作成したアプリケーションにとって最適なアプリケーション・サーバーです。Oracle9*i* Application Server では、共通技術スタックを使用して、データとアプリケーション・ロジックを中間層にキャッシュ書込みすることで、Oracle データベースを透過的に拡張できます。さらに Oracle9*i* Application Server は、強力な拡張性および可用性機能の多くを、Oracle8*i* の完成したテクノロジから継承しています。

1.3 Oracle インターネット・プラットフォームによるアプリケーションの配置

インターネットでのアプリケーション・アーキテクチャは、標準的な 2 層のクライアント / サーバー・モデルから、プレゼンテーションとビジネス・ロジックをサーバーで管理する様々な多層配置に移行しました。この移行には、次のような利点が数多くあります。

- **新規バージョンの配置は簡単、迅速かつ安価です。**Web アプリケーションを配置する場合、ユーザーにアプリケーションの URL を公開します。この配置メソッドでは、各ユーザーのデスクトップ・マシン上にアプリケーション・ソフトをインストールする必

要がなくなるため、各地に分散した多くのユーザーにアプリケーションを配布するための時間、コストおよび複雑さが低減されます。

- **集中化された配布は、システム所有の総コストが低減されることを意味します。**Web への配置は、情報へのアクセス可能性を向上すると同時に、管理、メンテナンスおよびネットワークのコストを大幅に削減します。複数の場所からシステム管理サポートを提供するかわりに、システム・メンテナンスおよび管理は 1 つのセントラル・ロケーションから実行されます。Web への配置により、アプリケーションの複雑さは各ユーザーのデスクトップから消えて、集中的に配置され専門的に管理されたアプリケーション・サーバーで処理されます。これにより、少数のサーバー上のサイトを専門的に管理できるため、メンテナンス作業の単純化、高速化および標準化が行えます。したがって、コストを大幅に低減できます。
- **業界標準に準拠した開発は統合性に優れています。**Oracle アプリケーションの開発は、World Wide Web 全体で使用されている既存または最新の標準に準拠しています。これには Java、Enterprise JavaBeans、HTML、XML、CORBA、HTTP、HTTPS などが含まれます。共通言語は、新規にまたは個別に開発されたアプリケーションの容易かつ迅速な統合を意味します。
- **コンポーネントベースの開発は、生産性が向上し、メンテナンスが容易になり、再利用が可能になることを意味します。**各ユーザーの各リクエストに応じて、アプリケーションを迅速にカスタマイズします。企業の開発者は関連するコンポーネントのみを変更し、アプリケーション全体を変更する必要はありません。通常の方法で変更されたコンポーネントは他のアプリケーションで簡単に再利用できます。これらは、組織が "Web 時間 " 内でユーザーのリクエストに応えるための方法の一部です。

アプリケーションの配布および管理が次のように単純化されているため、Oracle インターネット・プラットフォームは非常に低コストの配置プラットフォームです。

- サーバーの拡張性により、コストを低減し管理しやすくするためにサーバーの台数を減らすことが可能です。
- サーバー側にアプリケーションおよびデータ処理を配置することで、ネットワークの使用率を適正に保ちます。
- クライアント側では、必要なものはブラウザのみです。データベースに接続されているデスクトップで必要となる追加ソフトウェアのコストは不要です。

1.4 Oracle9*i* Application Server

Oracle9*i* Application Server は、Oracle インターネット・プラットフォームの重要なコンポーネントです。あらゆるベンダーの中で最も広い範囲の中間層サービスが用意されており、ポータルとトランザクション・アプリケーションの開発、柔軟な配置、企業での統合およびビジネス・インテリジェント・サービスのすべてをサポートします。

Oracle9*i* Application Server を使用すると、低コストで迅速に、新規および既存のアプリケーションをインターネット上で実行できるようにすることができます。また、拡張性、可用性およびロード・バランス・サービスを介した、パフォーマンス面の利点もあります。

1.4.1 拡張性

Oracle9*i* Application Server では、次の 3 つの主要な方法により、Web アプリケーションの高度な拡張性を実現しています。

- 多くの種類のハードウェアおよびオペレーティング・システム上で動作します。これにより、ユーザーはアプリケーションを変更することなくハードウェアをアップグレードできます。
- 中間層にあるデータベースのデータおよびストアド・プロシージャをキャッシュ書込みすることで、システムの拡張性を向上できます。これにより、バックエンド・データベースではより多くの同時ユーザーを扱えます。
- 単一ノード・クラスタまたは複数ノード・クラスタに配置できます。これにより、小規模なアプリケーションと大規模なアプリケーションの両方を拡張できます。

1.4.2 可用性

ハードウェアおよびソフトウェアがなんらかの理由で停止したとしても、Oracle9*i* Application Server 上でアプリケーションを実行しているクライアントがサービスの低下に気付くことはほとんどありません。Oracle9*i* Application Server には、サーバーに障害が発生した場合でもお使いのシステムを利用できるようにするために設計された、次のような機能とメカニズムが数多く用意されています。

- シングル・ポイント障害はありません。
- 途切れたセッションの影響を最小化するために、そのセッションを切り離します。
- 障害の検出、接続ルートの再構築およびプロセスの再起動は自動的に行われます。

1.4.3 ロード・バランス

ロード・バランスを使用すると、現在お使いのハードウェアの限界に近づいたときにマシンのアップグレードや破棄をせずに、ノードを追加するだけで、増大する負荷をいくつかのマシン間に分散できます。

効果的なロード・バランスにより、システムの処理リソースを効率的に利用することで拡張性を最大化できます。Oracle9*i* Application Server では、単一ノード上のスレッドとプロセスの間、および複数ノード配置でのノードとノードの間の両方で、ロード・バランスが効率的に保たれます。さらに、Oracle9*i* Application Server は、中間層サーバー・ファームとして知られている、集中化されたホスト・マシンの集合上に配置できます。

1.4.4 Oracle9*i* Application Server のサービス

Oracle9*i* Application Server は、サービスとユーティリティのセットで構成されています。これらのサービスとユーティリティは、拡張性と信頼性を目的とした分散環境でアプリケーションをインプリメントするために使用できます。これには、次のものが含まれます。

- 通信サービス
- プレゼンテーション・サービス
- データ管理サービス
- システム・サービス
- ビジネス・ロジック・サービス

1.4.4.1 通信サービス

Oracle9*i* Application Server の通信サービスでは、サーバーに送信されたリクエストが処理されます。サービスは、Oracle HTTP Server と Oracle HTTP Server モジュールの組合せを介して提供されます。Oracle HTTP Server は Apache によって強化された、インターネット上の業界標準の Web リスナーです。Apache は全世界の 60 パーセント以上のインターネット・サイトで使用されており、強力で拡張可能なテクノロジを提供します。Oracle HTTP Server モジュールは HTTP Server のプラグインで、固有のサービスの提供または外部プロセスへのリクエストの発信により、機能を拡張します。

1.4.4.2 プレゼンテーション・サービス

Oracle9*i* Application Server のプレゼンテーション・サービスでは、グラフィカル表現の出力が、通常は HTML の形式で処理されます。クライアントのプレゼンテーションを生成するために、スクリプトによる低レベルのプログラミングから Oracle Portal を使用した高レベルのフレームワークまで、様々な方法がサポートされています。このサービスには、Oracle Portal、Apache JServ、OracleJSP、PL/SQL および Perl のサポートが含まれています。

1.4.4.3 データ管理サービス

バックエンド・データベース・インスタンスの負荷を低減し、ネットワークでの読取り専用データの往復通信を回避するために、Oracle9*i* Application Server には Oracle 9*i* Application Server Cache が含まれています。

Oracle 9*i* Application Server Cache は読取り専用データであり、Oracle9*i* Application Server のコンポーネントとして中間層に常駐するアプリケーション・キャッシュです。中間層マシン上の頻繁に使用されるデータおよびストアド・プロシージャをキャッシュ書込みすることにより、Oracle データベースにアクセスするアプリケーションのパフォーマンスと拡張性が改善されます。Oracle 9*i* Application Server Cache により、ある種のアプリケーションでは、その本来のキャパシティ内でできる限り多くのリクエストを処理することが可能となりました。これらのパフォーマンスの改善は、主として次の 2 つの要因に基づいています。

- 中間層でデータベース問合せを処理することにより、ネットワーク上でデータを送受信するための時間が短縮されます。
- データベース・サーバー層での負荷の低減により、既存のデータベースでより多くのユーザーをサポートできるようになります。

Oracle 9*i* Application Server Cache によって最大限の恩恵を被るアプリケーションには、ネットワークを介して Oracle データベースのデータをアクセスするもの、読取り専用の動的かつ重要なコンテンツを持つもの、ほとんど変更されることのないディスクリート表を含むもの、などがあります。

1.4.4.4 システム・サービス

システム管理サービスおよびセキュリティ・サービスを提供するために、Oracle9*i* Application Server には Oracle Enterprise Manager（OEM）と Oracle Advanced Security が含まれています。これらのサービスでは、Secure Sockets Layer ベースの暗号化および認証機能を使用したネットワーク・セキュリティに加えて、お使いの Oracle 環境全体に対する総合管理フレームワークが提供されます。

OEM には、お使いの Oracle プラットフォームを集中管理するための統合されたソリューションが用意されています。これには GUI コンソール、Oracle Management サービス、Oracle Intelligent Agent および管理用ツールが含まれます。

OEM は次の目的で使用します。

- お使いの Oracle 製品とサードパーティ・サービスの状態の監視および対応
- 複数ノード上のアクティビティのスケジューリング
- ネットワーク化されたサービスのイベントの監視
- サーバー・コンポーネントおよびサービスに関する考慮事項の論理的な管理グループへの編成
- アプリケーションのパフォーマンスの測定（OEM の Oracle Trace を使用)

1.4.4.5 ビジネス・ロジック・サービス

Oracle9*i* Application Server には、Java 開発環境と高レベルのモデル・ドリブン技術の両方を使用してビジネス・ロジックを開発するための、いくつかの方法が用意されています。これには、Oracle Forms Developer および Oracle Reports Developer を使用する充実した GUI 指向のアプローチだけでなく、Java 2 Platform、Enterprise Edition（J2EE）、Enterprise Java Beans（EJB）、Oracle Business Components for Java（BC4J）などの Java テクノロジのサポートも含まれます。

1.5 Oracle9*i* Application Server を使用したフォームの配置

Oracle9*i* Application Server の Forms Services を使用すると、Oracle Forms Developer で作成したアプリケーションを、機能やインタフェースを損なうことなくインターネットまたは社内のイントラネットで実行できます。

Web 配置を目的とした新しいアプリケーションを作成したり、クライアント / サーバーに配置されている既存のフォーム、メニューおよびライブラリを取り出して、ほとんど変更することなく Web 配置に移行できます。

Oracle9*i* Application Server の Forms Services を使用する場合、他にも多くの利点があります。次にその利点の一部を紹介します。

- **拡張可能な最適化 Java クライアント。**企業の開発者は、Forms アプリケーションに JavaBeans を組み込んだり、Java クラスを再利用したりできます。これにより、Forms Services のクライアント Java アプレット部分が拡張され、企業の開発者は高機能のユーザー・インタフェースを作成できます。これらのインタフェースは Java 言語の長所を利用しており、Java コンポーネントを再利用できます。
- **あらゆるネットワークで自動拡張性を実現。**Oracle Forms Services では初めから、ロード・バランス機能が提供されます。ロード・バランスはクライアントのリクエストを、使用可能なシステム・リソース全体に効率的に配置します。これは、企業のイントラネット、エクストラネットおよびインターネットでの構築用に最適化されています。このアプリケーションは、LAN、WAN およびダイアルアップ・ネットワーク・アーキテクチャで使用できます。
- **ビルトイン最適化により高性能を実現。**Oracle Forms Services では、3 層アーキテクチャで標準となっている 2 つの主要な制約に対応するため、多くのビルトイン最適化が行われています。2 つの制約とは、ネットワーク帯域幅およびクライアントとアプリケーション・サーバー間の待ち時間です。

  Forms Services は、先進的なアルゴリズムを使用してデータ・ストリームを高度に圧縮することにより、ネットワーク帯域幅を縮小します。

  Oracle Forms Services が待ち時間を短縮するために、次のようにイベント・バンドルを使用する方法があります。ユーザーが項目 A から項目 B にナビゲートする場合（1 つのエントリ・フィールドから別のエントリ・フィールドに移動するためにタブを選択する場合など）、事前トリガーおよび事後トリガーの範囲が起動される場合があります。これらはサーバーで処理される必要があります。イベント・バンドルは、2 つのオブジェクト間のナビゲート中にトリガーされたすべてのイベントを " 収集し "、これらを 1 つの処理用パケットとしてサーバーに配置します。ナビゲーション時に多くのオブジェクトに問い合せる必要がある場合（離れたオブジェクトをマウスでクリックした場合など）、イベント・バンドルは問い合されたすべてのオブジェクトからすべてのイベントを収集し、これらを 1 つのネットワーク・メッセージとしてサーバーに配置します。
- **生産性が高い宣言型 Rapid Application Development（RAD）ツールとの統合。**Oracle9*i* Application Server の Forms Services は、Oracle Forms アプリケーション用として開発されました。したがって、異なるベンダーのツールで作成されたアプリケー

ションおよびサーバーを統合する際に生じる時間を要さないため、開発の後速やかに配置できます。

1.6 このマニュアルの使用方法

アプリケーションをインターネット上に配置することを選択した場合、その実現方法に関して多くのことを決定する必要があります。このマニュアルでは、これらの決定に関する情報を提供します。また、アプリケーションを Web に配布するシステムを構成するための提案およびメソッドを提供します。

弊社は次のものを提供します。

- Oracle9*i* Application Server アーキテクチャに含まれる Forms Services コンポーネントの概要
- 様々な Web への配置シナリオで Forms Services コンポーネントをインストールおよび構成するためのガイド
- これまでのクライアント / サーバー・アプリケーション資産の Web への移行に関する項
- 増大する作業負荷に対応するために機能し相互通信する複数のサーバーをセットアップする際に役立つ、キャパシティ計画およびロード・バランスを行うための項
- ネットワークおよびセキュリティ考慮事項に関する項
- Web アプリケーションのアプリケーション設計上の考慮点およびパフォーマンスを最適化する際のチューニングに関する項

2

Forms Services の概要

2.1 概要

Oracle9*i* Application Server はスケーラブルで安全な、中間層アプリケーション・サーバーです。これにより、Web コンテンツとホスト Web アプリケーションの配信、およびバックオフィス・アプリケーションへの接続が可能となります。Forms Services は Oracle9*i* Application Server の必須部分であり、インターネット・コンピューティングの利点を最大限に実現するテクノロジを提供します。この章では、Forms Services のアーキテクチャの概要から、特にフォームのインターネット上での配置に関する概要を説明します。

Forms Services は、新規および従来の Oracle Forms アプリケーションを、World Wide Web 上に配置できるようにする新世代の開発ツールです。アプリケーションは、社内イントラネットや社外のエクストラネットまたはインターネット上に配置できます。

Forms Services は、Oracle Forms アプリケーションを多層環境に配置するために最適化されたアプリケーション・サーバーです。Oracle Developer Server では、Web の使用やアクセスが簡単なことを利用し、Web に単なる静的な情報公開メカニズムを超えた、複雑で動的なアプリケーションをサポートできる環境としての機能を与えます。

2.2 Forms Services のアーキテクチャ

Forms Services では 3 層のアーキテクチャを使用して、データベース・アプリケーションを配置します。図 2-1 は、Forms Services のアーキテクチャを構成する 3 層を示します。

- **クライアント層**には、アプリケーションを表示する Web ブラウザが含まれます。
- **中間層**は、アプリケーション・ロジックとサーバー・ソフトウェアが格納されるアプリケーション・サーバーです。
- **データベース層**は、企業データが格納されるデータベース・サーバーです。

.

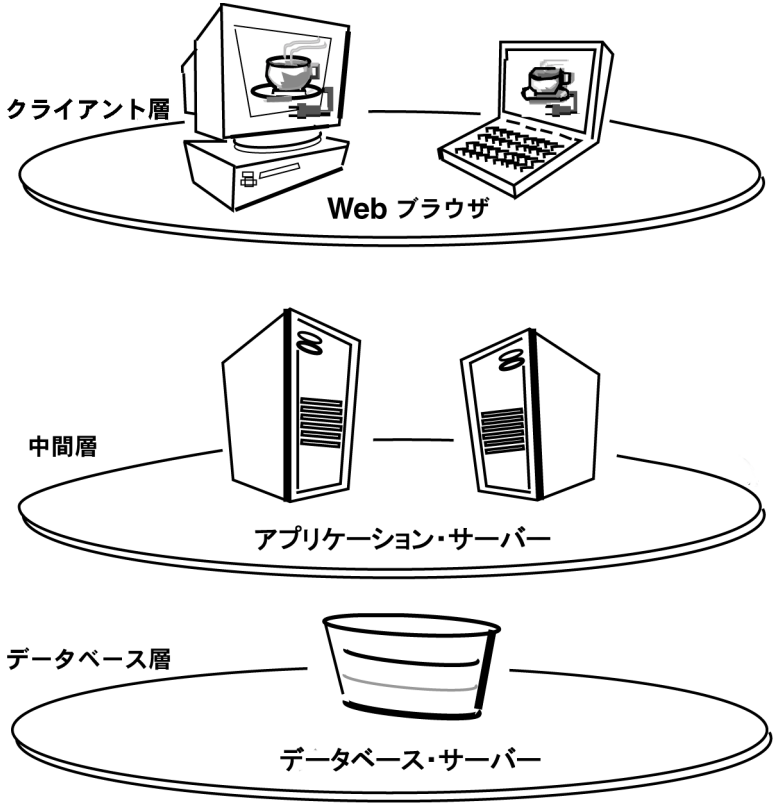


図 2-1 Forms Services のアーキテクチャ

2.3 Forms Services のコンポーネント

Forms Services は中間層のアプリケーション・サーバーで、複雑なトランザクション・フォーム・アプリケーションをインターネット上に配置します。開発者は Oracle Forms Developer で新規アプリケーションを構築し、Forms Services を使用してインターネット上に配置できます。また、開発者は従来のクライアント / サーバー型アプリケーションを、そのアプリケーション・コードを変更することなく 3 層のアーキテクチャに移行することもできます。

Forms Services は図 2-2 に示されるように、3 つの主要なコンポーネントで構成されます。

- **Forms アプレット** クライアントにダウンロードされ、Web ブラウザで参照します。
- **Forms Listener** 中間層にあります。
- **Forms Runtime エンジン** これも中間層にあります。

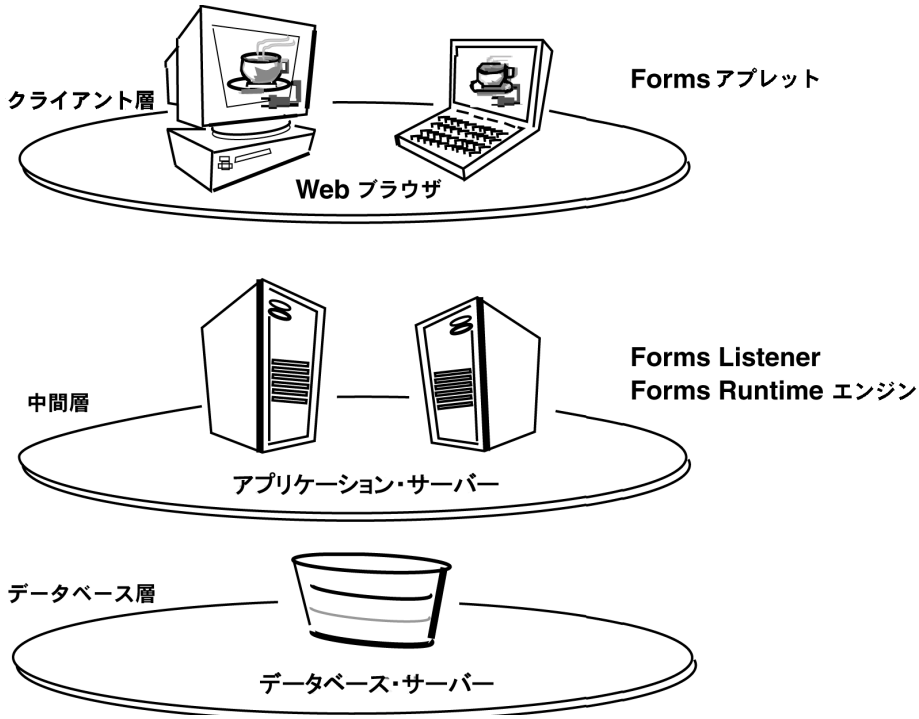


図 2-2 Web 上でフォームを実行する 3 層の構成

2.3.1 Forms アプレット

ユーザーが Web 上でフォーム・セッションを実行すると、Java ベースの小さな Forms アプレットがアプリケーション・サーバーから動的にダウンロードされて、自動的に Java クライアント・マシンにキャッシュされます。

Forms アプレットは、Forms Services Runtime エンジンにユーザー・インタフェースを提供します。拡張可能な最適化 Java アプレットとして、クライアントの Web ブラウザのフレームワーク内で操作できます。項目間の移動や、チェックボックスにチェックを付ける処理は対話的に行われ、その際に生成される情報がビジュアルに返されます。アプリケーションの表示のみを行い、特定のアプリケーション・ロジックは含まれません。

フォームのサイズや複雑度にかかわらず、どのフォームに対しても同じ Java アプレット・コードを使用できます。これは Web 上に配置するアプリケーションやフォームごとに、Java コードを記述する必要がないことを意味します。

2.3.2 Forms Listener

Forms Listener は、Java クライアントと Forms Services Runtime プロセス間におけるブローカとして機能します。Java クライアント・プロセスから接続リクエストを受け取り、Forms Services Runtime プロセスを開始します。またリスナーは Java クライアント完了後できるだけ早く接続できるように、実行するエンジンのプールを保持できます。

2.3.3 Forms Runtime エンジン

Forms Runtime エンジンはアプリケーション・ロジックと処理を管理します。Java クライアントのためにデータベース接続を保持します。クライアント / サーバー・モードで実行するのに使用されたのと同じフォーム、メニューおよびライブラリ・ファイルを使用します。これまでのクライアント / サーバー・アプリケーション資産をインターネット上に配置するのに、アプリケーション・コードの変更は必要ありません。

Forms Runtime エンジンは 2 つの役割を果たします。クライアントのブラウザと通信するときは、クライアントからのリクエストを処理するサーバーとして機能します。データベース・サーバーと通信するときは、要求されたデータをデータベース・サーバーに対して問い合せるクライアントとして機能します。

2.4 Forms Services の動き

Web 上で Forms アプリケーションの実行を開始するには、Java 対応の Web ブラウザを使用して、URL にアクセスします。Forms Services に関するプロセス・フロー中に発生する一連のイベントを説明します。

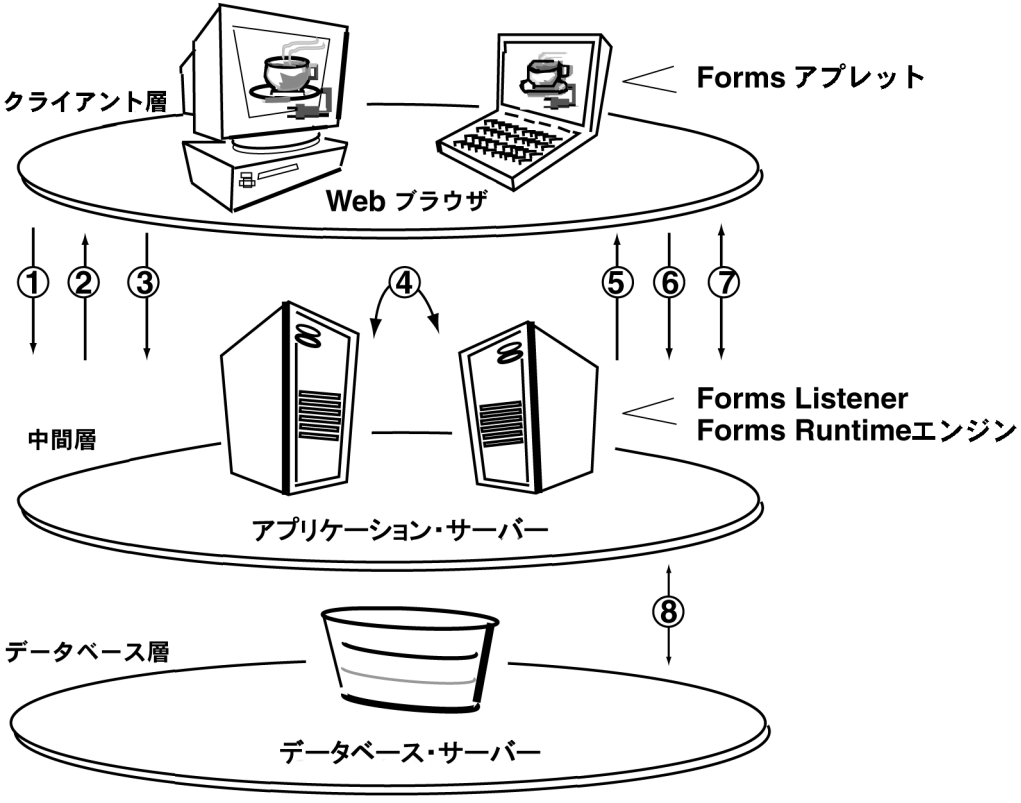


図 2-3 Forms Services のプロセス・フロー

ユーザーが Web 上で Forms アプリケーションを実行するとき、次のイベントが順に起こります。

1. ユーザーが Forms アプリケーションの実行を指示する HTML ページの URL にアクセスします。
2. その HTML ページが Web ブラウザにダウンロードされます。必要であれば、クライアントは Forms アプレットを含む Java アーカイブ・ファイルもダウンロードします。Forms アプレットがインスタンス化され、HTML ページからのパラメータを使用して実行する Forms アプリケーションが決定されます。
3. Forms アプレットは Forms Listener（Forms アプレットをダウンロードしたマシンの特定のポートに存在します）にリクエストを送信します。
4. Forms Listener は Forms Runtime エンジンに連絡して、Forms Services Runtime プロセスに接続します。HTML ページに含まれている場合は、Forms コマンドライン・パラメータ（フォーム名、ユーザー ID およびパスワード、データベース SID、メニュー名などの）および任意のユーザー定義 Form Builder パラメータが、Forms Listener によってプロセスに渡されます。
5. リスナーは Runtime エンジンとの接続を確立して、接続情報を Forms アプレットに送信します。
6. その後 Forms アプレットは、Runtime エンジンとの直接接続を確立します。
7. この時点で Forms アプレットと Runtime エンジンは通信を直接開始し、他のユーザーからの開始リクエストを受け取れるようにリスナーを開放します。Forms アプレットはアプリケーションのユーザー・インタフェースを、ユーザーの Web ブラウザのメイン・ウィンドウに表示します。
8. Runtime エンジンで実行中のアプリケーションは、データベースと直接通信します。

3

構成時の選択肢の概説

3.1 概要

この章では、Forms Services の構成時に表示されるオプションを説明します。構成時の選択項目には次のようなものがあります。

- ソケット接続、HTTP 接続、または SSL（secure sockets layer）の HTTP 接続のいずれにするか。
- ネイティブ JVM を使用する Internet Explorer でフォームを表示するか、Oracle JInitiator または AppletViewer を使用する Internet Explorer または Netscape Navigator でフォームを表示するか。
- ロード・バランスにするか、スタンドアロン構成にするか。
- Forms Servlet 実装か、CGI 実装か。

3.2 ソケット、HTTP または HTTPS

Forms Services は、アプリケーションを配置するために次の 3 つのモードで使用できます。

- ソケット
- HTTP
- HTTPS（SSL 付きの HTTP 1.1）

ユーザー固有のネットワーク環境に最適な Forms Services の実装の詳細は、9.3 項「ネットワーク環境における Forms Services の配置」を参照してください。

3.2.1 ソケット

他の多くのインターネット・ベースのテクノロジと同様に、Forms Services は当初、ソケットを使用して通信するよう設計されました。ソケット接続では、TCP/IP に対する標準のプログラミング・インタフェースが使用されます。

ソケットがどのようなものかは、ネットワーク上で通信するプログラムのナンバリング・システムを想像してもらえば一番簡単です。一般にこれらのプログラムは共通のソケット番号を共有する、クライアント部分とサーバー部分があります。サーバーはクライアントからのリクエストを共通のソケット・ポートでリスニングします。プログラムのクライアント部分とサーバー部分間の通信は、通称「ソケット接続」で行われます。

ソケットの代表的な使用例を示します。クライアントが標準以外のポート番号（たとえば、http://www.xyz.com:9000）を持つ URL にリクエストを送信します。これはクライアントのブラウザがソケット番号 9000 へ接続しようとすることを意味します。また、これはポート 9000 上で接続をリスニングするサーバーが www.xyz.com 上で稼働されていることも意味します。

ソケット・モードでの配置は効率的で簡単に使用できます。Forms Services はネットワーク化されたホスト・マシン上で稼動し、ユーザーのマシン上で実行されるクライアントからの接続を、特定のソケット上またはポートでリスニングします。このメソッドが機能するには、クライアントおよびサーバーのマシンがネットワーク上でお互いを識別できるか、通信できる必要があります。このモードではサーバー側でプロキシを使用できません。

注意： サーバー側のプロキシとは、インターネットに接続またはサービスを提供するときに、サーバー・ソフトウェアを稼動させているマシンを不明または匿名にしておくメソッドです。これはクライアントには認識されずに、サーバーに対する権限のないアクセスを拒否するために使用されるセキュリティ機能です。

サーバーとクライアントが、インターネットなどの安全を保障されていないネットワークで分断されている場合は、ソケット・ベースでの配置は危険にさらされる可能性があります。

3.2.2 HTTP

注意： クライアントのブラウザで Oracle JInitiator を使用する場合、HTTP モードおよび HTTPS モードを使用するには JInitiator のバージョン 1.1.7.30 が必要です。

HTTP モードでは同じくソケット接続を介して通信が確立されますが、この場合は HTTP ソケット接続になります。Forms Services はソケットによる独占接続ではなく、クライアントからの HTTP 接続をリスニングします。Forms Services とクライアント間のすべての内部メッセージが、HTTP パケット内にカプセル化されます。

HTTP ソケット接続では、サイトのクライアントとサーバー間でファイアウォールを通した安全な通信を実現します。HTTP 通信のみを許可するサイトは、構成をまったく変更しないかほとんど変更せずに既存のファイアウォールを通して Forms アプリケーションを配置できます。プロキシが使用されているという事実、はクライアントには完全にわかりません。クライアントから見ると、Forms Services に直接接続しているのと変わりありません。

ファイアウォールが存在する場合は、ソケット・モードは機能しません。ファイアウォールを通したソケット・モード接続が機能するには、Forms Services によって使用される特定の

ソケットまたはポートが開かれていて、ファイアウォールで使用可能になっていることが必要ですが、その場合は開かれたソケットの場所を突き止めるトラフィックにネットワークがさらされることになります。これではファイアウォールに穴が開けられ、その目的が本質的に損なわれます。

HTTP はインターネット上にアプリケーションを配置するために最もよく使用されるプロトコルです。企業はファイアウォールをロックして HTTP 通信のみを許可することで、プライベート・ネットワークのセキュリティを大幅に強化できます。ファイアウォールを提供している企業の多くは、その製品で HTTP 標準をサポートしており、多くの企業は保有するプライベート・ネットワークの中を HTTP 通信が行きかうことを好意的に認めています。

3.2.3 HTTPS

HTTPS モードでは通信は、3.2.2 項「HTTP」に説明があるように、HTTP ソケット接続を介して確立されます。ただし、HTTPS では SSL(secure sockets layer) も実装されます。

注意： クライアントのブラウザで AppletViewer を使用する場合は、HTTPS 接続モードはサポートされません。

注意： クライアントのブラウザで Oracle JInitiator を使用する場合、HTTP モードおよび HTTPS モードを使用するには JInitiator のバージョン 1.1.7.30 が必要です。

Forms Services では SSL をトランスポート・プロトコルとして使用し、機密性、整合性およびサーバー認証を提供できます。SSL は、アプリケーション・レベルの 1 つ下のレベルである、転送レベルで動作します。これは Telnet、FTP および HTTP などのアプリケーションレベルのプロトコルでメッセージが処理される前に、SSL でメッセージの暗号化と複合化ができることを意味します。

- **機密性**は、意図しない受信者によってメッセージが読まれるのを防ぐために、クライアントとサーバー間のメッセージを暗号化することで達成されます。

  サーバーおよびクライアントは、128 ビットまたは 40 ビットの暗号化をサポートします。サーバーで 128 ビットの暗号化を使用している場合、40 ビットの暗号化を使用するクライアントはそのままではサーバーに接続できません。接続するには、サーバー側で環境変数 FORMS60_HTTPS_NEGOTIATE_DOWN を TRUE に設定する必要があります。（デフォルトの設定は FALSE です。）詳細は 5.3 項「環境変数のカスタマイズ」を参照してください。この環境変数を TRUE に設定すると、接続しようとするクライアントによってサポートされる最高レベルの暗号化が常にサーバーで使用されます。FALSE に設定すると、サポートする暗号化のレベルがサーバーのレベルより低いクライアントは接続できません。次の表に実現例を示します。

サーバーの暗号化レベル	クライアントの暗号化レベル	FORMS60_HTTPS_NEGOTIATE_DOWN の設定	接続
128 ビット	40 ビット 128 ビット	TRUE	可。あるクライアントに対しては 40 ビット、他のクライアントに対しては 128 ビットの暗号化をサポートします。
128 ビット	40 ビット	FALSE	なし
40 ビット	128 ビット	TRUE	可。40 ビットの暗号化をサポートします
40 ビット	40 ビット	TRUE	可。40 ビットの暗号化をサポートします
40 ビット	40 ビット	FALSE	可。40 ビットの暗号化をサポートします

- **整合性**は、メッセージが変更されるのを防ぎます。メッセージが変更されると正しく復号化できません。
- **サーバー認証**は、そのサーバーが対象サーバーに間違いないことをクライアント・マシンで検証するプロセスです。たとえば、クライアントが機密データをサーバーに送信する場合、クライアントは相手側のサーバーが安全で、送信した機密データの正しい受信者であることを検証できます。サーバー認証は、デジタル証明を使用して行われます。クライアントのブラウザがサーバーに接続したとき、サーバーは証明書を検証のために提示します。証明書は認証局（CA）と呼ばれる第三者機関によって発行されます。

  ネイティブ JVM を使用する Internet Explorer がクライアントのブラウザの場合、ブラウザよって信頼されるすべての証明書を使用できます。ブラウザのデフォルトでは信頼されない CA を使用する場合は、その CA の指示を参照してください。また、証明書要求の作成および証明書の管理のために、Forms Services に Oracle Wallet Manager をインストールする必要があります。詳細は 5.7 項「HTTPS 接続モードの設定」を参照してください。

  JInitiator を使用するクライアント・ブラウザの場合、HTTPS モードでは、（デフォルトで）次の CA から発行された証明書を信頼します。

  - VeriSign, Inc. - クラス 1、2、3 Public Primary Certification Authority
  - RSA Data Security Inc. - Secure Server Authority
  - GTE CyberTrust Solutions Inc. - CyberTrust Global Root
  - GTE Corporation - CyberTrust Root

他の CA または他のタイプの証明書を使用する場合、その証明書はデフォルトでは信頼されないため、追加の構成ステップが必要となります。また、証明書要求の作成および証明書の管理のために、Forms Services に Oracle Wallet Manager をインストールする必要があります。詳細は 5.7 項「HTTPS 接続モードの設定」を参照してください。

3.3 ネイティブ JVM、Oracle JInitiator または AppletViewer を使用するクライアントのブラウザ

次のブラウザ構成のいずれかを使用することにより、ユーザーは Oracle Forms アプリケーションを Web 上で表示できます。

- ネイティブ JVM（Internet Explorer 5 を使用）
- Oracle JInitiator プラグイン（Netscape Navigator または Internet Explorer を使用）
- AppletViewer

注意： クライアントのブラウザで AppletViewer を使用する場合は、HTTPS 接続モードはサポートされません。

注意： クライアントのブラウザで Oracle JInitiator を使用する場合、HTTP モードおよび HTTPS モードを使用するには JInitiator のバージョン 1.1.7.30 が必要です。

3.3.1 Internet Explorer 5 を使用するネイティブ JVM

オラクル社は、Microsoft 固有の署名済 CAB ファイル（f60all.cab）を提供します。これにより、Internet Explorer 5 内で、Oracle Forms Java アプレットを信頼されたアプレットとして実行できます。このブラウザ・オプションを使用すると、ブラウザのエンド・ユーザー構成を実行する必要性が少なくなります。

詳細は、B.3 項「ネイティブ JVM を使用する Internet Explorer 5」を参照してください。

3.3.2 Oracle JInitiator

Oracle JInitiator は Web ブラウザ内で稼動します。ブラウザのデフォルトの JVM ではなく、クライアント上の特定の Java 仮想マシン (JVM) を使用するように指定する機能が提供されます。Oracle JInitiator はブラウザによって提供されるデフォルトの JVM を、置き換えたり変更せずに、代替の JVM をプラグイン形式で提供します。かわりに、プラグインのフォームで代替の JVM が提供されます。

Oracle JInitiator は Java ソフトウェア・プラグインのオラクル社版です。これは Netscape Navigator ではプラグインとして実行され、Internet Explorer では ActiveX コンポーネントとして実行されます。

オラクル社からは 2 つの JAR ファイル（f60all.jar および f60all_jinit.jar）が提供されます。これらはともに、ネットワークを介したクライアントへの効率的な配布を目的として、クラスをグループ化して zip 圧縮したものです。f60all_jinit.jar ファイルは高圧縮された JAR

ファイルで、ダウンロード時のパフォーマンスを向上させるために Oracle JInitiator でのみ使用できます。これらのファイルをクライアントで使用すると、今後の使用のためにキャッシュされます。

注意： クライアントのブラウザで Oracle JInitiator を使用する場合、HTTP モードおよび HTTPS モードを使用するには JInitiator のバージョン 1.1.7.30 が必要です。

詳細は、B.4 項「Oracle JInitiator」を参照してください。

3.3.3 AppletViewer

ユーザーは AppletViewer を使用してもアプリケーションを参照できます。AppletViewer は Java Developer Kit (JDK) コンポーネントの 1 つで、クライアント・マシンで使用して、Forms Services 上で実行されるアプリケーションを参照できます。

注意： クライアントのブラウザで AppletViewer を使用する場合は、HTTPS 接続モードはサポートされません。

アプリケーションを AppletViewer で実行する詳細は、B.5 項「AppletViewer」を参照してください。

3.4 ロード・バランスまたはスタンドアロン構成

Forms Services には、1 人から数千人までのユーザーに比類のないパフォーマンスを提供するために、ハードウェアのリソースを最適化するためのロード・バランス機能が含まれています。ロード・バランスを使用すると、ハードウェアの限界に近づいたときにマシンのアップグレードや交換をしなくても、単にアプリケーションを実行するマシンを追加して負荷をいくつかのマシン間に分散することで解決できます。

ロード・バランスのインプリメントに関する特定の情報は、第 12 章「ロード・バランスに関する考慮事項」を参照してください。

3.5 Forms Servlet 実装または CGI 実装

Forms Servlet および Forms CGI の各コンポーネントは、Forms Services とともにインストールされます。サーブレット実装および CGI 実装のいずれにおいてもロード・バランスが提供され、HTML ファイルをその場で作成できます。

サーブレット実装と CGI 実装の主な違いは次のとおりです。

- サーブレットを使用すると HTML ファイルはその場で作成されるので、特にネットワークの通信量が多い場合には、CGI 使用時よりも迅速に作成されます。
- サーブレットの場合、複数のエンド・ユーザー・ブラウザ構成を使用できます。Forms サーブレットはクライアントのブラウザ種別を自動的に検出し、正しいタグと正しいアーカイブを判別しながら HTML ページをその場で生成します。

サーブレット実装および CGI 実装のいずれにおいても、formsweb.cfg を使用して構成パラメータを定義します。

3.6 次のステップ

選択肢を決定後、必要な Forms Services コンポーネントを構成できます。Forms Services をインストールするための Oracle Universal Installer の使用方法は、第 4 章「Forms Services のインストール」を参照してください。Forms Services の構成の詳細は、第 5 章「Forms Services の構成」を参照してください。

4

Forms Services のインストール

4.1 概要

Forms Services は、Oracle9*i* Application Server の Enterprise Edition の一部としてインストールされます。大量のトランザクションを処理する中規模から大規模の Web サイトでは、Enterprise Edition を使用することをお薦めします。

Forms Services のインストールの詳細は、『Oracle9*i* Application Server インストレーション・ガイド』を参照してください。必要となるすべての要件と作業は、インストレーション・ガイドに記述されています。

4.2 Oracle Universal Installer について

Oracle9*i* Application Server では、Java ベースのツールである Oracle Universal Installer を使用して、環境変数の構成ならびにコンポーネントのインストールをします。インストーラではインストール・プロセスのステップごとに説明が表示されるので、異なる構成オプションを選択できます。

インストーラには、次の処理を実行する機能があります。

- 製品のインストール・オプションの参照および提供
- 事前設定された環境変数および構成設定の検出
- 環境変数および構成設定のインストール時の設定
- 製品の削除

4.3 Forms Services の起動

インストールが完了すると、Forms Services が自動的に起動されます。

Forms Services を手動で起動するには、次のように入力します。

```
<ORACLE_HOME>/6iserver/ forms60_server start
```

Forms Services を停止するには、次のように入力します。

```
<ORACLE_HOME>/6iserver/ forms60_server stop
```

4.4 次のステップ

実際にアプリケーションを配置するには、いくつかのステップを実行する必要があります。そのステップにはランタイム実行可能ファイルの作成、Web サーバーへの実行可能ファイルの配置およびアプリケーションの URL のブロードキャストが含まれます。これらのステップは第 6 章「Web へのフォームの配置」で説明します。

5

Forms Services の構成

5.1 概要

この章では、Forms Services 用の環境を構成するために必要なステップを説明します。インストールの完了後、この章に記載されている情報を使用して、初期構成の変更や必要な変更を行うことができます。

この章には、次の項が含まれています。

- Web サーバーの構成
- 環境変数のカスタマイズ
- 構成ファイルのカスタマイズ
- サーブレット・エラー・ログの読込み
- HTTPS 接続モードの設定

5.2 Web サーバーの構成

Oracle9*i* Application Server によって、Oracle HTTP Server が Web サーバーとしてインストールおよび構成されます。追加の構成は必要ありません。

次のパスが作成されます。

仮想パス	物理ディレクトリ	説明
/forms60java/	<ORACLE_HOME>/6iserver/forms60/java/	Forms Java ファイル
/dev60html/	<ORACLE_HOME>/6iserver/tools/web60/html/	Forms を実行する初期の HTML ファイル
/servlet/	<ORACLE_HOME>/6iserver/forms60/java/oracle/forms/servlet	サーブレット実行可能ファイル
/dev60cgi/	<ORACLE_HOME>/6iserver/tools/web60/cgi/	CGI 実行可能ファイル
/jinitiator/	<ORACLE_HOME>/6iserver/jinit/	JInitiator（ダウンロード用）
/dev60temp/	<ORACLE_HOME>/6iserver/tools/web60/temp/	Forms テンポラリ・ファイル

注意： これらの仮想ディレクトリは、<ORACLE_HOME>/6iserver ディレクトリにある 6iserver.conf ファイルで指定されます。

5.3 環境変数のカスタマイズ

この項では、Forms Services の環境変数をカスタマイズする方法について説明します。

UNIX の場合、<ORACLE_HOME>/6iserver ディレクトリにある forms60_server シェル・スクリプト内でこれらの環境変数を設定できます。そうしておくと、次のコマンドラインを使用して Forms Services Listener を起動する時に、Forms Services に必要なすべての環境変数が自動的に設定されます。

```
forms60_server start
```

注意： forms60_server 起動スクリプトの実行後は、Forms Services で使用するために、ORACLE_HOME は元の設定から <ORACLE_HOME>/6iserver に変更されます。

NT の場合、A.2 項「Windows 95 および Windows NT のレジストリ」で説明されているとおり、HKEY_LOCAL_MACHINE¥software¥oracle にある Forms 6*i* に対応する <ORACLE_HOME> レジストリ内の環境変数を設定します。

Forms Services 用の環境変数は次のとおりです。

環境変数	デフォルト値と説明
FORMS60_PATH	<ORACLE_HOME>/6iserver/forms60 実行する Form を検索するときに、Forms が検索するパスを指定します。パスはセミコロン (;) で区切ります。
FORMS60_OUTPUT	<ORACLE_HOME>/6iserver/tools/web60/temp 生成したレポート・ファイルを格納するアプリケーション・サーバー上の物理ディレクトリ。Reports を使用していない場合は、この環境変数は必要ありません。詳細は、7.5 項「レポートの統合」を参照してください。
FORMS60_MAPPING	/dev60temp FORMS60_OUTPUT 変数で定義された物理ディレクトリを指す仮想ディレクトリ。Reports を使用していない場合は、この環境変数は必要ありません。詳細は、7.5 項「レポートの統合」を参照してください。
FORMS60_MESSAGE_ENCRYPTION	設定されません。 有効な値は TRUE または FALSE です。環境変数は RC4 40 ビットの暗号化機能を使用して、Forms メッセージを暗号化します。ソケットおよび HTTP 通信モードにのみ適用されます。デフォルトの設定で、通信は暗号化されます。
FORMS60_WALLET	<ORACLE_HOME>/6iserver/forms60/wallet HTTPS 通信モードで使用されます。詳細は 5.7 項「HTTPS 接続モードの設定」を参照してください。
FORMS60_HTTPS_NEGOTIATE_DOWN	FALSE HTTPS 通信モードのみに使用されます。5.7 項「HTTPS 接続モードの設定」を参照してください。

たとえば、次のように環境変数を定義できます。

```
FORMS60_PATH=/<ORACLE_HOME>/6iserver/forms60
FORMS60_OUTPUT=/<ORACLE_HOME>/6iserver/tools/web60/temp
FORMS60_MAPPING=/dev60temp
FORMS60_MESSAGE_ENCRYPTION=TRUE
FORMS60_WALLET=/<ORACLE_HOME>/6iserver/forms60/wallet
FORMS60_HTTPS_NEGOTIATE_DOWN=FALSE
```

注意： FORMS60_MAPPING 環境変数によって設定された仮想ディレクトリは、FORMS60_OUTPUT 環境変数によって設定された物理ディレクトリに対応している必要があります。

注意： これらの変更を行うには管理者権限が必要です。また、構成の変更を有効にするには、サーバーの再起動が必要です。

5.4 Forms Services 起動パラメータの説明

Forms Services の起動時には、次のパラメータが使用されます。

- Port パラメータ
- Mode パラメータ
- Pool パラメータ
- Log パラメータ

UNIX の場合、これらのパラメータを修正するには、<ORACLE_HOME>/6iserver ディレクトリにある forms60_server シェル・スクリプトを編集し、次のコマンドを修正します。

```
f60ctl start
```

例：

```
f60ctl start port=9001 mode=socket pool=5 log=/tmp/app.log
```

NT の場合、コマンドラインにこれらのパラメータを指定することで、パラメータを修正できます。例：

```
ifsrv60 start port=9001 mode=socket pool=5 log=c:¥tmp¥app.log
```

NT 上で、Forms Services がサービスとして起動されている場合にパラメータを修正するには、「**サービス**」ダイアログの「**スタート パラメータ**」フィールドにパラメータを追加します。

5.4.1 Port パラメータ

サーバー・プロセスが開始されるポートを決定します。Forms Services プロセスの開始時にポート番号を指定しないと、デフォルトのポート 9001 上でプロセスが開始されます。サーバー・プロセスを開始するポート番号は、アプリケーションの HTML ファイル、構成パラメータまたは URL に指定する serverPort 番号と一致する必要があります。

5.4.2 Mode パラメータ

Forms Services を、ソケット・モード（ソケットの直接接続を使用）、HTTP モード（ファイアウォールを越えることが可能）または HTTPS モード（ファイアウォールを越えることが可能で、SSL (secure sockets layer) を追加で使用してサーバー認証およびメッセージの暗号化を行う）のいずれかで実行するかを決定します。デフォルトのモードはソケットです。各モードの詳細は、3.2 項「ソケット、HTTP または HTTPS」を参照してください。

5.4.3 Pool パラメータ

後から使用するユーザーが利用できる、アクティブなスペアの接続数を決定します。たとえば "pool" が 5 に設定された場合は、5 つのアクティブなスペア接続があります。

5.4.4 Log パラメータ

パス名およびログ・ファイル名が提供されると、サーバーのログ・ファイルを生成します。たとえば、log=/PathName/LogFileName とします。

5.5 構成ファイルのカスタマイズ

インストール時に、次の構成ファイルがシステムにインストールされています。

- FormsServlet.initArgs
- formsweb.cfg
- base.htm、basejini.htm および baseie.htm

ユーザーが最初に（アプリケーションの URL へのリンクをクリックすることで）Web 対応のアプリケーションを開始すると、ベース HTML ファイルが Forms Servlet または CGI によって読み込まれます。ベース HTML ファイル内の任意の変数 (%*variablename*%) は、formsweb.cfg ファイルに指定された適切なパラメータ値によって置換され、URL リクエストがある場合はその問合せパラメータの値によって置換されます。

サーブレット実装では、baseHTML、baseHTMLJInitiator および baseHTMLIE の各タグは、FormsServlet.initArgs ファイルに指定した値で置換されます。

変更が必要な場合は構成ファイルを変更できます。

5.5.1 FormsServlet.initArgs

このファイルは次の場所にあります。

```
<ORACLE_HOME>¥6iserver¥apache¥jserv¥servlets¥oracle¥
forms¥servlet¥FormsServlet.initArgs
```

このファイルは、サーブレット実装を使用している場合のみ編集してください。次のパラメータが含まれています。

パラメータ	必須 / 任意	パラメータ値
baseHTML	**必須**	アプレット・タグを含む HTML ファイルへの絶対パス。 デフォルトのパスは <ORACLE_HOME>/forms60/server/base.htm です。
baseHTMLJinitiator	**必須**	JInitiator タグを含む HTML ファイルへの絶対パス。 デフォルトのパスは <ORACLE_HOME>/forms60/server/baseJini.htm です。
baseHTMLIe	**必須**	Internet Explorer 5 のタグ（CABBASE タグなど）を含む HTML ファイルへの絶対パス。 デフォルトのパスは <ORACLE_HOME>/forms60/server/baseie.htm です。
configFileName	**必須**	構成ファイル formsweb.cfg を指す絶対パス。 デフォルトのパスは <ORACLE_HOME>/forms60/server/formsweb.cfg です。

注意： 絶対パスでは、いかなる環境変数も参照しないでください。

注意： UNIX および NT のいずれにおいても、絶対パスを指定するには円記号（¥）ではなくスラッシュ（/）を使用してください。

注意： FormsServlet.initArgs ファイル内のパラメータ名では、大文字と小文字が区別されます。

5.5.2 formsweb.cfg

インストール時に設定した、サーブレットおよび CGI 構成パラメータのほとんどの設定が、このファイルに含まれます。変更が必要な場合はこれらのパラメータをカスタマイズできます。

ベース HTML ファイル内の変数 (%*variablename*%) は、formsweb.cfg ファイルに指定された適切なパラメータ値によって置換されるか、URL リクエストがある場合はその問合せパラメータからの値によって置換されます。

構成の変更は formsweb.cfg ファイルに入力して、変数はベース HTML ファイル内で使用することをお薦めします。

5.5.2.1 formsweb.cfg での特別な構成の作成

名前を付けた固有の構成を formsweb.cfg ファイル内に作成できます。これらの構成は、フォームの実行に使用するエンド・ユーザーの URL 問合せ文字列から要求できます。

特別な構成を作成するには、大カッコ（[]）で囲んだ構成の名前を formsweb.cfg ファイルの最後に追加します。次に、この特別な構成用のパラメータを指定します。（変更するパラメータのみを指定してください。）

たとえば、フレームに分かれているブラウザ・ウィンドウで "generic" ルック・アンド・フィールを使用してフォームを実行するための構成を作成するには、formsweb.cfg ファイルに次のコマンドを追加します。

```
[sepwin]
separateFrame=True
lookandfeel=Generic
```

エンド・ユーザーが "sepwin" 構成を使用するフォームを起動するには、次の URL を入力することになります。

```
http://server:port/servlet/f60servlet?config=sepwin
```
（サーブレット構成の場合）

```
http://myhost.mydomain.com/dev60cgi/ifcgi60.exe?config=sepwin
```
（CGI 構成の場合）

特別な構成のその他の例は、5.5.2.3 項「デフォルトの formsweb.cfg ファイル」を参照してください。

5.5.2.2 formsweb.cfg ファイル内のパラメータ

パラメータ	必須 / 任意	パラメータ値
baseHTML	**必須**	アプレット・タグを含む HTML ファイルへの物理パス。
baseHTMLJInitiator	**必須**	JInitiator タグを含む HTML ファイルへの物理パス。
baseHTMLIE	**必須**	Internet Explorer 5 のタグ（CABBASE タグなど）を含む HTML ファイルへの物理パス。デフォルトのパスは <ORACLE_HOME>/6iserver/forms60/server/baseie.htm です。
ie50	Internet Explorer 5 ブラウザを使用するユーザーがいる場合に推奨	クライアントが Internet Explorer 5 ブラウザを使用している場合は、ネイティブ JVM、JInitiator または AppletViewer のいずれかを使用できます。"JInitiator" の設定では basejini.htm ファイルと JInitiator を使用します。"Native" の設定ではブラウザのネイティブ JVM を使用します。
HTML delimiter	**必須**	変数名のデリミタ。デフォルトで % になります。

パラメータ	必須 / 任意	パラメータ値
MetricsServerHost	任意	ロード・バランス用。第 12 章「ロード・バランスに関する考慮事項」を参照してください。
MetricsServerPort	任意	ロード・バランス用。第 12 章「ロード・バランスに関する考慮事項」を参照してください。
MetricsServerErrorURL	任意	ロード・バランス用。第 12 章「ロード・バランスに関する考慮事項」を参照してください。
MetricsTimeout	任意	ロード・バランス用。第 12 章「ロード・バランスに関する考慮事項」を参照してください。
leastloadedhost	任意	ロード・バランス用。第 12 章「ロード・バランスに関する考慮事項」を参照してください。 ベース HTML ファイルまたは formsweb.cfg ファイルのどちらかに指定できる変数です。ロード・バランスの設定では、負荷の最も低いマシン名を指定する必要があります。推奨するデフォルトのベース HTML ファイルを使用して、ロード・バランスを設定するときは必ず serverHost=%loastloadedhost% を formsweb.cfg ファイルに指定してください。 このプレース・ホルダは、ロード・バランスが行われる間に、負荷の最も低いシステムの名前に動的に置き換えられます。
標準のアプレットまたはオブジェクトのパラメータ 注意 : 次のすべてをベース HTML ファイルに %*variablename*% として指定できます。例 : `<PARAM NAME="connectMode"  VALUE="%connectMode%">` ベース HTML ファイル内のすべての変数は、formsweb.cfg ファイルに指定された適切なパラメータ値で置き換えられます。		
codebase	**必須**	物理ディレクトリ <ORACLE_HOME>/6iserver/forms60/java をポイントするように定義した仮想ディレクトリ。
code	**必須**	コード・パラメータは削除や変更をしないでください。常に次の値にします。oracle.forms.engine.Main。
connectMode	**HTTP および HTTPS 接続では必須**、ソケット接続では任意	Forms Services で使用する接続プロトコルのタイプをクライアントに指定します。有効な値はソケット、http および https です。デフォルトはソケットです。詳細は 3.2 項「ソケット、HTTP または HTTPS」を参照してください。
archive_ie	任意	カンマで区切った CAB ファイルのリスト。ネイティブ JVM を使用する Internet Explorer がブラウザとして検出された場合に、CAB ファイルが使用されます。（デフォルトは f60all.cab。）

パラメータ	必須 / 任意	パラメータ値
archive_jinit	任意	カンマで区切った JAR ファイルのリスト。検出されたブラウザが JInitiator の場合に JAR ファイルが使用されます。（デフォルトは f60all_jinit.jar。）
archive	任意	カンマで区切ったアーカイブ・ファイルのリスト。検出されたブラウザがネイティブ JVM を使用する Internet Explorer または JInitiator のいずれでもない場合に、アーカイブ・ファイルが使用されます。（デフォルトは f60all.jar。）
width	**必須**	Form の幅をピクセルで指定。
height	**必須**	Form の高さをピクセルで指定。
align	任意	left | center | right | top | middle | bottom
alt	任意	アプレットのかわりに表示されるテキスト（ブラウザがアプレットをサポートしない場合）。
hspace	任意	水平方向の余白をピクセルで指定。
vspace	任意	垂直方向の余白をピクセルで指定。
type	**必須**	ハードコードされた値（JInitiator の場合は "application/x-jinit-applet"、AppletViewer の場合は値は不要）。
name	任意	アプレットのインスタンス名。
title	任意	アドバイザリ・タイトル文字列。
border	任意	表示する境界線。
standby	任意	ロード時に表示するテキスト。
codetype	任意	タイプするための初期設定。
Forms アプレットに固有のパラメータ（PARAM タグ内）		
serverHost	任意	Forms Services（NT の場合は ifsrv60.exe）を実行するホスト（デフォルトは Web リスナーのあるマシン）。
serverPort	**必須**	Forms Services（NT の場合は ifsrv60.exe）がリスニングするポート。ほとんどの場合、ポート番号は 9001（デフォルト）のままです。

パラメータ	必須 / 任意	パラメータ値
serverArgs	**必須**	Runform 用のコマンドライン・パラメータ。次の Runform パラメータを参照してください。 forms_param を任意の有効な Forms Runtime コマンドライン・パラメータで置換します。user_param を任意の有効なユーザー定義パラメータで置換します。 例、<param name="serverArgs" VALUE="module=order.fmx"> **注意** : Forms Runtime コマンドラインとユーザー定義パラメータを複数指定できます。HTML ファイルにディレクトリ・パスを含めるか、FORMS60_PATH 環境変数を定義して、.FMX ファイルの物理ディレクトリ・パスを指定する必要があります。拡張子 .FMX は任意です。
splashScreen	任意	アプレットが表示される前に表示する .GIF ファイルを指定。スプラッシュなしの場合は「NO」に設定します。デフォルトのスプラッシュを使用する場合は空白のままにします。
background	任意	背景に表示する .GIF ファイルを指定。背景なしの場合は「NO」に設定します。デフォルトの背景を使用する場合は空白のままにします。
clientDPI	任意	1 インチ当たりのドット数（DPI）を指定し、JVM によって戻される DPI 設定を上書きします。これにより、各プラットフォームの様々な DPI 設定を管理できます。たとえば、Win32 プラットフォームで開発されたフォームは、DPI 値の違いにより、UNIX プラットフォーム上では正しく表示されない可能性があります。clientDPI の値には、すべての正の整数を指定できます。Oracle は 50 から 200 の整数を使用することをお薦めします。 <param name="clientDPI" value="200">
separateFrame	任意	アプレットを分割フレーム内に表示するかどうかを指定。有効な値 : TRUE または FALSE。
lookAndFeel	任意	アプリケーションのルック・アンド・フィールを指定。有効な値 : Oracle または Generic（Windows 95 のルック・アンド・フィール）。
colorScheme	任意	アプリケーションの配色を指定。有効な値 : Teal、Titanium、Red、Khaki、Blue、Olive または Purple。 **注意** : lookAndFeel が Generic に設定されている場合、colorScheme は無視されます。
serverApp	任意	アプリケーションのクラス名がある場合に、デフォルトを置き換えます。アプリケーション固有のフォント・マッピングの作成およびアイコン・パスの設定には、アプリケーション・クラスを使用します。

パラメータ	必須 / 任意	パラメータ値
heartBeat	任意	このパラメータを使用して、クライアントが稼動中であることを示すためにサーバにパケットを送る頻度を設定します。整数値で分を定義するか、あるいは分に対する小数値で秒を定義します。たとえば、0.5 は 30 秒を示します。デフォルトは 2 分です。
imageBase	任意	このパラメータを使用して、アイコン・ファイルが格納される場所を指定します。次の中から選択します。 ■ **codeBase** は、アイコン検索パスが Java クラスを含むディレクトリに対応することを示します。アイコンを JAR ファイルに格納する場合にこの値を使用します（推奨）。 ■ **documentBase** は、デフォルトです。Forms Services CGI を使用した配置では、アイコン・パスをカスタム・アプリケーション・ファイル中に指定する必要があります。
registryPath	任意	このパラメータを使用して、serverApp パラメータで名前を付けたアプリケーション・ファイルが格納されている仮想ディレクトリをリスト表示します。
webformsTitle	任意	このパラメータを使用して、フォームの表示ウィンドウの上端に表れるタイトルを変更します。
Runform パラメータ（serverArgs パラメータ）		
MODULE	**必須**	Form のモジュール名（任意でパスを含みます）。
USERID	任意	scott/tiger@ORA8 などのログイン文字列。
ユーザー定義パラメータ	任意	任意の名前 / 値のペア。

5.5.2.3 デフォルトの formsweb.cfg ファイル

デフォルトの formsweb.cfg ファイルには次の内容が含まれます。

```
; Forms Web CGI Configuration File
; --------------------------------
; This file defines parameter values used by the Forms Web CGI

; ********************************
; PARAMETER VALUES USED BY DEFAULT
; ********************************
  ; SYSTEM PARAMETERS
  ; -----------------
  ; These have fixed names and give information required by the Forms
  ; Web CGI in order to function.  They cannot be specified in the URL query
  ; string.  But they can be overriden in a named configuration (see below).
baseHTML=d:¥orant¥forms60¥server¥base.htm
```

```
baseHTMLJInitiator=d:¥orant¥forms60¥server¥basejini.htm
baseHTMLie=d:¥orant¥forms60¥server¥baseie.htm
HTMLdelimiter=%
MetricsServerPort=9020
MetricsServerErrorURL=
  ; The next parameter specifies how to execute the Forms applet under
  ; Microsoft Internet Explorer 5.0.  Put IE50=native if you want the
  ; Forms applet to run in the browser's native JVM.
IE50=native
  ; USER PARAMETERS
  ; ---------------
  ; These match variables (e.g. %form%) in the baseHTML file. Their values
  ; may be overridden by specifying them in the URL query string
  ; (e.g. "http://myhost.mydomain.com/ifcgi60.exe?form=myform&width=700")
  ; or by overriding them in a specific, named configuration (see below)

  ; 1) Runform arguments:
form=test.fmx
otherparams=
userid=

  ; 2) HTML page title, attributes for the BODY tag, and HTML to add before and
  ;    after the form:
pageTitle=Oracle Forms Server
HTMLbodyAttrs=
HTMLbeforeForm=<B>Hello</B>
HTMLafterForm=

  ; 3) Values for the Forms applet parameters:
width=650
height=500
separateFrame=false
splashScreen=no
    ; select default background by not specifying a value
background=
lookAndFeel=Oracle
colorScheme=teal
serverApp=default
serverPort=9000
serverHost=rlouis-lap
connectMode=Socket
archive=f60all.jar
archive_ie=f600all.cab
archive_jinit=f60all_jinit.jar

  ; 4) Parameters for JInitiator
    ; Page displayed to Netscape users to allow them to download JInitiator.
```

```
    ; If you create your own version, set this parameter to point to it.
jinit_download_page=/jinitiator/us/jinit_download.htm
    ; Parameters related to the version of JInitiator.
jinit_classid=clsid:21157916-4d49-11d4-a3e0-00c04fa32518
jinit_exename=jinit.exe#Version=1,1,7,30
jinit_mimetype=application/x-jinit-applet;version=1.1.7.30

; ********************************
; SPECIFIC CONFIGURATIONS
; ********************************
;  You may define your own specific, named configurations (sets of parameters)
;  by adding special sections as illustrated in the following examples.
;  Note that you need only specify the parameters you want to change.  The
;  default values (defined above) will be used for all other parameters.
;  Use of a specific configuration can be requested by including the text
;  "config=<your_config_name>" in the query string of the URL used to run
;  a form.  For example, to use the sepwin configuration, your could issue
;  a URL like "http://myhost.mydomain.com/ifcgi60.exe?config=sepwin".

; Example 1: configuration to run forms in a separate browser window with
;            "generic" look and feel (include "config=sepwin" in the URL)
[sepwin]
separateWindow=True
lookandfeel=Generic

; Example 2: configuration affecting users of MicroSoft Internet Explorer 5.0.
;            Forms applet will run under the browser's native JVM rather than
;            using Oracle JInitiator.
[ie50native]
IE50=native

; Example 3: configuration forcing use of the base.htm base HTML file in all
;            cases (means applet-style tags will always be generated and
;            JInitiator will never be used).
[applet]
baseHTMLJInitiator=
```

5.5.3 base.htm、basejini.htm および baseie.htm

Forms Services のインストールと構成時に、Oracle Universal Installer によって 3 つのベース HTML ファイルがシステムに作成されます。**ほとんどの場合、これらのファイルの変更は必要ありません。**

ユーザーが最初に（アプリケーションの URL へのリンクをクリックすることで）Web 対応のアプリケーションを開始すると、ベース HTML ファイルが Forms Servlet または CGI によって読み込まれます。

ベース HTML ファイル内の任意の変数（%*variablename*%）は、5.5.2 項「formsweb.cfg」での説明のように、formsweb.cfg ファイルに指定された適切なパラメータ値によって置換されるか、URL リクエストがある場合はその問合せパラメータの値によって置換されます。

サーブレット実装では、baseHTML、baseHTMLJInitiator および baseHTMLIE の各タグは、5.5.1 項「FormsServlet.initArgs」での説明のように、FormsServlet.initArgs ファイルに指定した値で置き換えられます。

次に、ベース HTML ファイルがユーザーの Web ブラウザにダウンロードされます。

注意： 変更する任意のベース HTML 変数は、5.5.1 項「FormsServlet.initArgs」で説明されている FormsServlet.initArgs ファイル、および 5.5.2 項「formsweb.cfg」で説明されている formsweb.cfg ファイル、それぞれのファイル内の対応するパラメータ値を変更することで変更できます。

次のベース HTML 初期ファイルを、<ORACLE_HOME>/6iserver/forms60/server ディレクトリで利用できます。

- **basejini.htm**: これは、Oracle JInitiator を使用する Forms アプレットの実行に必要なタグが含まれる HTML ファイルです。オラクル社によってこの方法での動作が確認されたブラウザ (および標準の APPLET タグを使用して動作しないブラウザ) に適しています (Windows プラットフォームのみ)。例は 5.5.3.4 項「デフォルトの basejini.htm ファイル」を参照してください。JInitiator 設定の詳細は、付録 B「クライアント・ブラウザのサポート」を参照してください。
- **base.htm**: これは、AppletViewer または（ネイティブ JVM が Forms で動作することがオラクル社によって確認済の）任意の Web ブラウザで Forms アプレットを実行するために必要な APPLET タグが含まれている、ベース HTML ファイルです。例は 5.5.3.3 項「デフォルトの base.htm ファイル」を参照してください。ネイティブ JVM および AppletViewer 設定の詳細は、付録 B「クライアント・ブラウザのサポート」も参照してください。
- **baseie.htm**: これは、Internet Explorer 5 でネイティブ JVM を使用するために必要な Internet Explorer 5 のタグが含まれている、ベース HTML ファイルです。例は、5.5.3.5 項「デフォルトの baseie.htm ファイル」を参照してください。Internet Explorer とネイティブ JVM の詳細は、付録 B「クライアント・ブラウザのサポート」も参照してください。

ベース HTML ファイルを新規作成する場合は次の作業を行います。

1. <ORACLE_HOME>/6iserver/forms60/server ディレクトリにある、basejini.htm または base.htm 初期ファイルをコピーします。
2. ファイル名を変更します（たとえば、`order.htm`）。
3. ユーザーに表示されるテキストを追加または変更します（たとえば、<TITLE> および <BODY> タグ内のテキスト）。
4. 必要に応じてパラメータを変更します。5.5.1 項「FormsServlet.initArgs」および 5.5.2 項「formsweb.cfg」の説明にあるように、ベース HTML ファイルでは変数を使用して、

実際の値は FormsServlet.initArgs と formsweb.cfg の各ファイルに指定することをお薦めします。

5. 新しいベース HTML ファイルを任意のディレクトリ内に置きます。FormsServlet.initArgs と formsweb.cfg の各ファイル内の baseHTML、baseHTMLJInitiator または baseHTMLIE パラメータを更新して、ベース HTML ファイルの絶対パスが含まれるようにします。

5.5.3.1 ベース HTML ファイル内のパラメータと変数

注意： base.htm または basejini.htm ファイルで提供されるパラメータ・タグを使用しない場合は、ファイルから削除してください。

パラメータ	必須 / 任意	パラメータ値
leastloadedhost	任意	ロード・バランス用。第 12 章「ロード・バランスに関する考慮事項」を参照してください。 ベース HTML ファイルまたは formsweb.cfg ファイルのどちらかに指定できる変数です。ロード・バランスの設定では、負荷の最も低いマシン名を指定する必要があります。推奨するデフォルトのベース HTML ファイルを使用する場合、ロード・バランスを使用するときは formsweb.cfg ファイルに serverHost=%leastloadedhost% を必ず指定してください。 ロード・バランスを使用している場合は、負荷が最も低いシステムの名前によって、このプレースホルダが動的に置き換えられます。
cabbase	任意	ネイティブ JVM を使用する Internet Explorer の場合、使用される CAB ファイル（f60all.cab）を含みます。

注意 : 残りのパラメータ値を変数 (%*variablename*%) としてベース HTML ファイルで指定することをお薦めします。例 :

```
<PARAM NAME="connectMode"  VALUE="%connectMode%">
```

または

```
<PARAM NAME="cabbase"  VALUE="%archive_ie%">
```

次に、実際のパラメータ値を 5.5.2.2 項「formsweb.cfg ファイル内のパラメータ」で定義される formsweb.cfg ファイルに指定します。すべての変数が実行時に適切なパラメータ値で置き換えられます。

5.5.3.2 使用方法

- 変数の値はベース HTML ファイル内のどこででも使用できます。変数は特別のデリミタで囲まれた名前として指定されます。(デフォルトのデリミタは % です。) たとえば、HTML ファイルに次の行を置くことができます。

  ARCHIVE="%Archive%"

  次に値を formsweb.cfg ファイル (または URL 問合せ文字列内) の %Archive% に割り当てる必要があります。

- すべての変数は実行時に値を受け取る必要があります。変数が値を受け取らないと、ユーザーの Web ブラウザに渡す HTML ファイルを Forms Services が構築できず、エラーが発生します。
- パフォーマンスを向上するには、JAR ファイルのダウンロード用のソースとして 1 つの Web サーバーのみを使用してください。同じファイルを異なるサーバーから複数回ダウンロードすることを防げます。

5.5.3.3 デフォルトの base.htm ファイル

```
<HTML>
<!-- FILE: base.htm (Forms Server)                          -->

<!-- This is the default base HTML file for running a form on the   -->
<!-- web using APPLET-style tags to include the Forms applet.       -->
<!-- This file will be REPLACED if you reinstall "Forms Web CGI and -->
<!-- cartridge", so you are advised to make your own version if you -->
<!-- want to make any modifications.  You should then set the       -->
<!-- baseHTML parameter in the Forms web CGI configuration file     -->
<!-- (formsweb.cfg) to point to your new file instead of this one.  -->

<!-- IMPORTANT NOTE: default values for all the variables which     -->
<!-- appear below (delimited by the percent character) are defined  -->
<!-- in the formsweb.cfg file. It is preferable to make changes in  -->
<!-- that file where possible, and leave this one untouched.        -->

<HEAD><TITLE>%pageTitle%</TITLE></HEAD>

<BODY %HTMLbodyAttrs%>
%HTMLbeforeForm%

<!-- Forms applet definition (start) -->
<APPLET CODEBASE="/forms60java/"
        CODE="oracle.forms.engine.Main"
        ARCHIVE="%archive%"
        WIDTH="%Width%"
        HEIGHT="%Height%">
```

```
<PARAM NAME="serverPort" VALUE="%serverPort%">
<PARAM NAME="serverHost" VALUE="%serverHost%">
<PARAM NAME="connectMode" VALUE="%connectMode%">
<PARAM NAME="serverArgs"
       VALUE="module=%form% userid=%userid% %otherParams%">
<PARAM NAME="separateFrame" VALUE="%separateFrame%">
<PARAM NAME="splashScreen"  VALUE="%splashScreen%">
<PARAM NAME="background"  VALUE="%background%">
<PARAM NAME="lookAndFeel"  VALUE="%lookAndFeel%">
<PARAM NAME="colorScheme"  VALUE="%colorScheme%">
<PARAM NAME="serverApp" VALUE="%serverApp%">

</APPLET>
<!-- Forms applet definition (end) -->

%HTMLafterForm%

</BODY>
</HTML>
```

5.5.3.4 デフォルトの basejini.htm ファイル

```
<HTML>
<!-- FILE: basejini.htm (Oracle Developer Forms)                    -->

<!-- This is the default base HTML file for running a form on the   -->
<!-- web using JInitiator-style tags to include the Forms applet.   -->
<!-- This file will be REPLACED if you reinstall "Forms Web CGI and -->
<!-- cartridge", so you are advised to make your own version if you -->
<!-- want to make any modifications.  You should then set the       -->
<!-- baseHTML parameter in the Forms web CGI configuration file     -->
<!-- (formsweb.cfg) to point to your new file instead of this one.  -->

<!-- IMPORTANT NOTE: default values for all the variables which     -->
<!-- appear below (delimited by the percent character) are defined  -->
<!-- in the formsweb.cfg file. It is preferable to make changes in  -->
<!-- that file where possible, and leave this one untouched.        -->

<HEAD><TITLE>%pageTitle%</TITLE></HEAD>

<BODY %HTMLbodyAttrs%>
%HTMLbeforeForm%

<!-- Forms applet definition (start) -->
<OBJECT classid="%jinit_classid%"
        codebase="/jinitiator/%jinit_exename%"
```

```
        WIDTH="%Width%"
        HEIGHT="%Height%"
        HSPACE="0"
        VSPACE="0">
<PARAM NAME="TYPE"       VALUE="%jinit_mimetype%">
<PARAM NAME="CODEBASE"   VALUE="/forms60java/">
<PARAM NAME="CODE"       VALUE="oracle.forms.engine.Main" >
<PARAM NAME="ARCHIVE"    VALUE="%archive%" >

<PARAM NAME="serverPort" VALUE="%serverPort%">
<PARAM NAME="serverHost" VALUE="%serverHost%">
<PARAM NAME="connectMode" VALUE="%connectMode%">
<PARAM NAME="serverArgs"
       VALUE="module=%form% userid=%userid% %otherParams%">
<PARAM NAME="separateFrame" VALUE="%separateFrame%">
<PARAM NAME="splashScreen"  VALUE="%splashScreen%">
<PARAM NAME="background"  VALUE="%background%">
<PARAM NAME="lookAndFeel"  VALUE="%lookAndFeel%">
<PARAM NAME="colorScheme"  VALUE="%colorScheme%">
<PARAM NAME="serverApp" VALUE="%serverApp%">
<COMMENT>
<EMBED SRC="" PLUGINSPAGE="%jinit_download_page%"
        TYPE="%jinit_mimetype%"
        java_codebase="/forms60java/"
        java_code="oracle.forms.engine.Main"
        java_archive="%archive%"
        WIDTH="%Width%"
        HEIGHT="%Height%"
        HSPACE="0"
        VSPACE="0"

        serverPort="%serverPort%"
        serverHost="%serverHost%"
        connectMode="%connectMode%"
        serverArgs="module=%form% userid=%userid% %otherparams%"
        separateFrame="%separateFrame%"
        splashScreen="%splashScreen%"
        background="%background%"
        lookAndFeel="%lookAndFeel%"
        colorScheme="%colorScheme%"
        serverApp="%serverApp%"
>
<NOEMBED>
</COMMENT>
</NOEMBED></EMBED>
</OBJECT>
<!-- Forms applet definition (end) -->
```

```
%HTMLafterForm%

</BODY>
</HTML>
```

5.5.3.5 デフォルトの baseie.htm ファイル

```
<HTML>
<!-- FILE: base.htm (Oracle Developer Forms)                          -->

<!-- This is the default base HTML file for running a form on the   -->
<!-- web using APPLET-style tags to include the Forms applet.       -->
<!-- This file will be REPLACED if you reinstall "Forms Web CGI and -->
<!-- cartridge", so you are advised to make your own version if you -->
<!-- want to make any modifications.  You should then set the       -->
<!-- baseHTML parameter in the Forms web CGI configuration file     -->
<!-- (formsweb.cfg) to point to your new file instead of this one.  -->

<!-- IMPORTANT NOTE: default values for all the variables which     -->
<!-- appear below (delimited by the percent character) are defined  -->
<!-- in the formsweb.cfg file. It is preferable to make changes in  -->
<!-- that file where possible, and leave this one untouched.        -->

<HEAD><TITLE>%pageTitle%</TITLE></HEAD>

<BODY %HTMLbodyAttrs%>
%HTMLbeforeForm%

<!-- Forms applet definition (start) -->
<APPLET CODEBASE="/forms60java/"
        CODE="oracle.forms.engine.Main"
        WIDTH="%Width%"
        HEIGHT="%Height%">

<PARAM NAME="cabbase" VALUE="%archive_ie%">
<PARAM NAME="serverPort" VALUE="%serverPort%">
<PARAM NAME="serverHost" VALUE="%serverHost%">
<PARAM NAME="connectMode" VALUE="%connectMode%">
<PARAM NAME="serverArgs"
       VALUE="module=%form% userid=%userid% %otherParams%">
<PARAM NAME="separateFrame" VALUE="%separateFrame%">
<PARAM NAME="splashScreen"  VALUE="%splashScreen%">
<PARAM NAME="background"  VALUE="%background%">
<PARAM NAME="lookAndFeel"  VALUE="%lookAndFeel%">
<PARAM NAME="colorScheme"  VALUE="%colorScheme%">
<PARAM NAME="serverApp" VALUE="%serverApp%">
```

```
</APPLET>
<!-- Forms applet definition (end) -->

%HTMLafterForm%

</BODY>
</HTML>
```

5.6 サーブレット・エラー・ログの読込み

Forms Servlet 実装を使用している場合、formsweb.cfg ファイルおよび FormsServlet.initArgs ファイル内の構成エラーはすべて jserv.log ファイルに記録されます。このファイルは <ORACLE_HOME>/apache/Jserv/logs にあります。

5.7 HTTPS 接続モードの設定

HTTPS 接続モードではファイアウォールを越えるために、HTTP を使用して通信します。また、Forms Services は SSL をトランスポート・プロトコルとして使用して、機密性、整合性およびサーバー認証を提供します。この通信モードの説明は、3.2.3 項「HTTPS」を参照してください。

HTTPS モードを使用するには、次のことが必要となります。

- **Web サーバーに対して :** ネイティブ JVM を使用する Internet Explorer がエンド・ユーザーのブラウザである場合、Web サーバー上で証明書の使用を必要とする SSL を使用するように Web サーバーを構成します。これを実行するステップは Web サーバーによって異なりますので、詳細はお使いの Web サーバーのマニュアルを参照してください。ネイティブ JVM を使用する Internet Explorer がエンド・ユーザーのブラウザである場合、ユーザーは初期フォーム起動 HTML ページを HTTPS モードでダウンロードする必要があります。（このステップは、Oracle JInitiator ではオプションです。）
- **Forms Services に対して :**
  - HTTPS 環境変数のカスタマイズ
  - クライアントの構成に応じて、次のステップのいずれかを使用してください。
    Wallet の作成と証明書の要求、または
    Wallet の作成とデフォルトでは JInitiator に信頼されない証明書の要求

注意 : どちらのステップを使用すべきかを判断するには、クライアントのブラウザの説明書を参照してください。

- **クライアントのブラウザに対して :** クライアントで使用されているブラウザによっては、Web サーバーと Forms Services のそれぞれにインストールされている証明書がクライアントのブラウザで信頼されることを確認するステップが必要になる場合があります。

フォームを表示するために、**ネイティブ JVM を使用する Internet Explorer 5 がクライアントのブラウザとして使用されている場合**、「Wallet の作成と証明書の要求」で説明されているステップに従ってください。

フォームを表示するために、**Oracle JInitiator がクライアントのブラウザで使用されている場合**、JInitiator ではデフォルトで次の CA および証明書が信頼されます。次の証明書のいずれかを使用している場合は、「Wallet の作成と証明書の要求」で説明されているステップに従ってください。

- VeriSign, Inc. - クラス 1、2、3 Public Primary Certification Authority
- RSA Data Security Inc. - Secure Server Authority
- GTE CyberTrust Solutions Inc. - CyberTrust Global Root
- GTE Corporation - CyberTrust Root

Oracle JInitiator がクライアントのブラウザで使用されているが、これらの証明書のいずれも使用していない場合は、「Wallet の作成とデフォルトでは JInitiator に信頼されない証明書の要求」で説明されているステップに従ってください。

注意： クライアントのブラウザで AppletViewer を使用する場合は、HTTPS 接続モードはサポートされません。

注意： HTTPS 接続モードを使用するには、Forms Services ならびにサーバー認証を提供するすべての Forms Services マシンに、Oracle Wallet Manager をインストールしておく必要があります。

5.7.1 HTTPS 環境変数のカスタマイズ

Forms Services のインストール中に、HTTPS モードに関連付けられた 2 つの環境変数が設定されます。これらの環境変数がセキュリティのニーズに合うように設定されていることを確認し、必要な場合は HTTPS モードで実行するすべての Forms Services マシン上で変更します。環境変数の変更方法は、5.3 項「環境変数のカスタマイズ」を参照してください。

環境変数	値
FORMS60_HTTPS_NEGOTIATE_DOWN	デフォルト値は FALSE です。 有効な値は TRUE および FALSE です。TRUE に設定されると、128 ビットの暗号化を使用するサーバーは、クライアントによってサポートされる最高レベルに合せて暗号化を調整します。FALSE に設定すると、サーバーは 128 ビットの暗号化をサポートしないクライアント接続を拒否します。詳細は 3.2.3 項「HTTPS」を参照してください。
FORMS60_WALLET	デフォルト値は /<ORACLE_HOME>/6iserver/forms60/wallet です。 サーバー認証に使用される証明書を保持する "Wallet" を含むディレクトリ。

5.7.2 Wallet の作成と証明書の要求

公開鍵暗号ではとりわけ証明書が要求されます。ユーザー証明書は認証局（CA）と呼ばれる第三者機関によって発行されます。証明書は安全な方法で取得され、アクセスのたびに証明書の妥当性チェックは必要ありません。

Forms Services および HTTPS モードを使用しているクライアントの場合、クライアントはサーバーの証明書を検証することで、その Forms Services が妥当なサーバーであることをチェックします。Wallet を作成して証明書を要求するには、Oracle Wallet Manager を使用します。

Forms Services に Oracle Wallet Manager をインストールした後で、証明書を取得するために次の作業を行う必要があります。

- Wallet の作成
- 証明書要求の作成
- 証明書要求の送信
- 証明書のインポート
- 「Auto Login」を「ON」に設定

次の項では、Oracle Wallet Manager における前述のステップを完了する方法の概要を提供します。詳細は、『Oracle Wallet Manager セキュリティ・ガイド』ドキュメントを参照してください。

注意： 複数の Forms Services マシンがある場合は、各マシンごとに一意の証明書を要求することも、同じ証明書をすべてのマシン上で使用することもできます。ライセンス条件については CA に連絡してください。

- 一意の証明書を各マシンごとに使用するには、この項のすべてのプロシージャを、HTTPS モードで実行する各 Forms Services マシン上で実行します。
- 同じ証明書をすべてのマシンで使用するには、この項のすべてのプロシージャを、証明書を含む Wallet を作成する Forms Services マシンのいずれかで実行します。次に、HTTPS モードで実行する他の Forms Services マシンに Wallet ファイル（ewallet.der）をコピーします。そのファイルを FORMS60_WALLET 環境変数で指定したディレクトリにコピーします。最後に、「「Auto Login」を「ON」に設定」の説明にあるように、「Auto Login」がすべてのマシンで「ON」に設定されていることを確認します。

5.7.2.1 Wallet の作成

UNIX では、<ORACLE_HOME>/6iserver/bin ディレクトリにある owm を実行します。

NT では、**「スタート」→「プログラム」→「Oracle for Windows NT」→「Oracle Wallet Manager」**をクリックして、Oracle Wallet Manager を実行します。

Wallet を次のように作成します。

1. メニューバーから**「Wallet」→「New」**をクリックします。「New Wallet」ダイアログ・ボックスが表示されます。
2. 「Wallet Password」フィールドにパスワードを入力します。
3. 「Confirm Password」フィールドにパスワードを再入力します。
4. 「**OK**」をクリックして続行します。新規に空の Wallet が作成されたことを告げるメッセージが表示され、証明書要求を作成するかどうかがプロンプトで求められます。
5. 「**はい**」をクリックして、次の項を参照します。

5.7.2.2 証明書要求の作成

証明書要求を次のように作成します。

1. 次の情報を「Certificate Request」ダイアログ・ボックスに入力します。
   - Common Name: 証明書アイデンティティの名前を、名前が先で名字が後の書式で入力します。たとえば、サーバー管理者の名前を使用できます。
   - Organizational Unit: 組織単位の名前を入力します。たとえば、Finance とします。
   - Organization: 組織名を入力します。たとえば、XYZ Corp とします。
   - Locality/City: 市町村名を入力します。
   - State/Province: 都道府県名を入力します。California を CA とするような略称は使用しないでください。
   - Country: リストをクリックして国名の略称リストを表示します。組織が置かれている国をクリックして選択します。
   - Key Size: ドロップダウン・ボックスをクリックして、暗号 / 復号鍵のペアを作成するときの鍵のサイズを参照します。
   - Advanced: 「Advanced」をクリックして、「Advanced Certificate Request」ダイアログ・パネルを表示します。このフィールドを使用して、識別名（DN）を編集またはカスタマイズします。
2. 「**OK**」をクリックします。証明書要求が正常に作成されたことが「Oracle Wallet Manager」メッセージ・ボックスに表示されます。
3. **「Wallet」→「Save」**をクリックして、Wallet をディスクに保存します。ディレクトリ名を指定するようプロンプトが表示されます。

5.7.2.3 証明書要求の送信

信頼された CA のいずれかに証明書要求を送信する方法は数多くあります。最も一般的な方法は、Oracle Wallet Manager から証明書要求を切り取って、Web 上の CA の証明書要求フォームに貼り付けることです。Oracle Wallet Manager メッセージ・ボックスの本文からテキストをコピーして電子メールのメッセージに貼り付け、認証局に要求を送信することもできます。ただし、この方法は認証局がその形式での要求を受け付けている場合に限ります。

次に、Oracle Wallet Manager ウィンドウに戻って「**OK**」をクリックします。証明書要求が正常に作成されたことが「Oracle Wallet Manager」メッセージ・ボックスに表示されます。

5.7.2.4 証明書のインポート

要求した証明書を CA から受け取った後、作成済の Wallet にそれをインポートする必要があります。2 つのいずれかの方法でインポートできます。

- 認証局から受信した電子メールから、証明書を貼り付けます。
- ファイルから証明書をインポートします。

証明書を貼り付けるには、次のようにします。

1. メニューバーから「**Operations**」→「**Import User Certificate**」をクリックします。「Import User Certificate」ダイアログ・ボックスが開きます。
2. 「**Paste the Certificate**」ラジオ・ボタンをクリックして、「**OK**」をクリックします。「Import User Certificate」ダイアログ・ボックスが開き、次のメッセージが表示されます。"Please provide a base64 format certificate and paste it below"。
3. 受信した電子メールの本文から証明書をコピーします。
4. 証明書をウィンドウに貼り付けて、「**OK**」をクリックします。ウィンドウの下部に、証明書が正常にインストールされましたというメッセージが表示されます。
5. 「**OK**」をクリックします。Oracle Wallet Manager のメイン・パネルが再び表示され、証明書が User Certificates ツリーの一番下に表示されます。
6. 「**Wallet**」→「**Save**」をクリックして、Wallet をディスクに保存します。

証明書を含むファイルをインポートするには、次のようにします。

1. メニューバーから「**Operations**」→「**Import User Certificate**」をクリックします。「Import User Certificate」ダイアログ・ボックスが開きます。
2. 証明書のあるパス名またはフォルダ名を入力します。
3. 証明書ファイルの名前（cert.txt など）をクリックして選択します。
4. 「**OK**」をクリックします。ウィンドウの下部に、証明書が Wallet に正常にインポートされましたというメッセージが表示されます。

5. 「**OK**」をクリックして、ダイアログ・ボックスを閉じます。Oracle Wallet Manager のメイン・パネルが再び表示され、証明書が User Certificates ツリーの一番下に表示されます。
6. 「**Wallet**」→「**Save**」をクリックして、Wallet をディスクに保存します。

5.7.2.5 「Auto Login」を「ON」に設定

Oracle Wallet Manager Auto Login 機能により、Wallet のコピーが自動的に開かれます。そのため、Wallet にパスワードを提供する必要がなく、サーバー認証が行われます。

「Auto Login」を「ON」に設定するには、次のようにします。

1. メニューバーから「**Wallet**」をクリックします。
2. 「**Auto Login**」メニュー項目の隣のチェックボックスをクリックします。これにより、cwallet.sso という名前のファイルが作成されます。このファイルはマシン依存で、あるマシンから別のマシンにはコピーできません。
3. 「Autologin enabled」というメッセージが、ウィンドウの下に表示されます。

注意： 「Auto Login」メニュー項目の隣のチェックボックスは、クリックするたびにオンとオフが切り替わります。チェックマークを消去するには、再度チェックボックスをクリックします。今度は autologin が使用不可になります。

注意： サーバー認証を提供するすべての Forms Services マシンに対して、「Auto Login」を「ON」に設定する必要があります。

5.7.3 Wallet の作成とデフォルトでは JInitiator に信頼されない証明書の要求

注意： この項では、JInitiator のデフォルトでは信頼されない証明書の例として VeriSign Trial Certificate を使用します。

注意： この項は、デフォルトでは Oracle JInitiator に信頼されない証明書を Web サーバーおよび Forms Services 上で使用するというシナリオに適用されます。Oracle JInitiator では、次の CA と証明書が信頼されます。

- VeriSign, Inc. - クラス 1、2、3 Public Primary Certification Authority
- RSA Data Security Inc. - Secure Server Authority
- GTE CyberTrust Solutions Inc. - CyberTrust Global Root
- GTE Corporation - CyberTrust Root

注意： これらの証明書のいずれかを使用している場合は、「Wallet の作成と証明書の要求」のステップに従ってください。

公開鍵暗号ではとりわけ証明書が要求されます。ユーザー証明書は認証局（CA）と呼ばれる第三者機関によって発行されます。証明書は安全な方法で取得され、アクセスのたびに証明書の妥当性チェックは必要ありません。

Forms Services および HTTPS モードを使用しているクライアントの場合、クライアントはサーバーの証明書を検証することで、その Forms Services が妥当なサーバーであることをチェックします。Wallet を作成して証明書を要求するには、Oracle Wallet Manager を使用します。

Forms Services に Oracle Wallet Manager をインストールした後で、証明書を取得するために次の作業を行う必要があります。

- Wallet の作成
- 証明書要求の作成
- 証明書要求の送信
- VeriSign Trial CA ルート証明書のクライアント・マシンへのインストール
- 証明書のインポート
- 「Auto Login」を「ON」に設定

注意： 複数の Forms Services マシンがある場合は、各マシンごとに一意の証明書を要求することも、同じ証明書をすべてのマシン上で使用することもできます。

- 一意の証明書を各マシンごとに使用するには、この項のすべてのプロシージャを、HTTPS モードで実行する各 Forms Services マシン上で実行します。
- 同じ証明書をすべてのマシンで使用するには、この項のすべてのプロシージャを、証明書を含む Wallet を作成する Forms Services マシンのいずれかで実行します。次に、HTTPS モードで実行する他の Forms Services マシンに Wallet ファイル（ewallet.der）をコピーします。そのファイルを FORMS60_WALLET 環境変数で指定したディレクトリにコピーします。最後に、「「Auto Login」を「ON」に設定」の説明にあるように、「Auto Login」がすべてのマシンで「ON」に設定されていることを確認します。

5.7.3.1 Wallet の作成

UNIX では、<ORACLE_HOME>/6iserver/bin ディレクトリにある owm を実行します。

NT では、**「スタート」→「プログラム」→「Oracle for Windows NT」→「Oracle Wallet Manager」**をクリックして、Oracle Wallet Manager を実行します。

Wallet を次のように作成します。

1. メニューバーから**「Wallet」→「New」**をクリックします。「New Wallet」ダイアログ・ボックスが表示されます。
2. 「Wallet Password」フィールドにパスワードを入力します。
3. 「Confirm Password」フィールドにパスワードを再入力します。
4. 「**OK**」をクリックして続行します。新規に空の Wallet が作成されたことを告げるメッセージが表示され、証明書要求を作成するかどうかがプロンプトで求められます。
5. 「**はい**」をクリックして、次の項を参照します。

5.7.3.2 証明書要求の作成

証明書要求を次のように作成します。

1. 次の情報を「Certificate Request」ダイアログ・ボックスに入力します。
   - Common Name: 証明書アイデンティティの名前を、名前が先で名字が後の書式で入力します。たとえば、サーバー管理者の名前を使用できます。
   - Organizational Unit: 組織単位の名前を入力します。たとえば、Finance とします。
   - Organization: 組織名を入力します。たとえば、XYZ Corp とします。
   - Locality/City: 市町村名を入力します。
   - State/Province: 都道府県名を入力します。California を CA とするような略称は使用しないでください。
   - Country: リストをクリックして国名の略称リストを表示します。組織が置かれている国をクリックして選択します。
   - Key Size: ドロップダウン・ボックスをクリックして、暗号 / 復号鍵のペアを作成するときの鍵のサイズを参照します。
   - Advanced: 「Advanced」をクリックして、「Advanced Certificate Request」ダイアログ・パネルを表示します。このフィールドを使用して、識別名（DN）を編集またはカスタマイズします。
2. 「**OK**」をクリックします。証明書要求が正常に作成されたことが「Oracle Wallet Manager」メッセージ・ボックスに表示されます。
3. **「Wallet」→「Save」**をクリックして、Wallet をディスクに保存します。ディレクトリ名を指定するようプロンプトが表示されます。

5.7.3.3 証明書要求の送信

CA に証明書要求を送信する方法は数多くあります。最も一般的な方法は、Oracle Wallet Manager から証明書要求を切り取って、Web 上の CA の証明書要求フォームに貼り付けることです。Oracle Wallet Manager メッセージ・ボックスの本文からテキストをコピーして電子メールのメッセージに貼り付け、認証局に要求を送信することもできます。ただし、この方法は認証局がその形式での要求を受け付けている場合に限ります。

次のステップでは、VeriSign の Trial Server Certificate を例として使用します。

1. ブラウザを使用して、www.verisign.com にナビゲートします。
2. ホーム・ページ上に "Trial Server Certificate" へのリンクがない場合は、それを検索します。
3. 実行すべき 5 つのステップが、VeriSign の Web サイトにリストされます。最初の 3 つのステップを実行することから始めます。

- **ステップ 1: Generate CSR**。このステップは、Oracle Wallet Manager を使用して完了しています。
- **ステップ 2: Submit CSR**。Oracle Wallet Manager から証明書要求情報を切り取り、VeriSign の「Trial Server Certificate」Web ページの「Enter CSR information」フィールドに貼り付けます。
- **ステップ 3: Complete Application**。証明書の送信先とする電子メールアドレスなどの連絡先の情報を、VeriSign の「Trial Server Certificate」Web ページに入力します。

4. ここで、Oracle Wallet Manager ウィンドウに戻って「**OK**」をクリックします。証明書要求が正常に作成されたことが「Oracle Wallet Manager」メッセージ・ボックスに表示されます。
5. 次に示す最後の 2 つのステップを完了させます。これらのステップは後続の項で説明されています。
   - **ステップ 4: Install Test CA Root**。次の項で実行します。
   - **ステップ 5: Install your Test Server ID**。次の項で実行します。

5.7.3.4 VeriSign Trial CA ルート証明書のクライアント・マシンへのインストール

CA ルート証明書をインストールして、Oracle Wallet Manager で読み込むことができる Base64 encoded X.509（.CER）ファイルにエクスポートするには、Internet Explorer 5.0 を使用する必要があります。（Netscape ではルート証明書をファイルにエクスポートできないので、Netscape は使用できません。）

1. Internet Explorer 5.0 を使用して、http://www.verisign.com/server/trial/welcome/caroot.html にナビゲートします。
2. 指示に従って、お使いのブラウザに CA ルート証明書をダウンロードします。
3. **「ツール」→「インターネット オプション」→「コンテンツ」さらに「証明書」**をクリックします。
4. 「証明書」ダイアログが表示されたら、「**目的**」オプションが「**すべて**」に設定されていることを確認し、「**信頼されたルート証明機関**」をクリックします。
5. 「**発行先**」列の値が「**For VeriSign authorized testing only...**」となっている証明書を選択します。
6. **「エクスポート」→「次へ」**をクリックし、**「Base64 encoded X.509(.CER)」**を選択します。
7. vrsnca.cer というファイル名で保存します。
8. Oracle Wallet Manager に戻ります。
9. **「Operations」→「Import Trusted Certificate」**をクリックします。

10. 「**Select a file that contains the certificate**」をクリックします。
11. 保存済の vrsnca.cer ファイルをオープンします。
12. 「Trusted Certificates」の中に「**For VeriSign authorized testing only**」がリスト表示されていることを確認します。
13. 「**Operations**」→「**Export All Trusted Certificates**」をクリックして、信頼された証明書をすべてエクスポートします。
14. vrsndb.txt というファイル名で保存します。
15. クライアント・マシンで、¥Program Files¥Oracle¥Jinitiator¥lib¥security¥certdb.txt のバックアップ・コピーをとることにより、JInitiator の certdb.txt を新しいバージョンで置き換えます。次に、vrsndb.txt を ¥Program Files¥Oracle¥Jinitiator¥lib¥security¥certdb.txt にコピーします。このステップによって、クライアントで信頼される CA のリストが更新されます。

5.7.3.5 証明書のインポート

VeriSign によって要求が処理されると、次のような証明書を含む電子メールが VeriSign から送信されます。

```
-----BEGIN CERTIFICATE-----
MIICETCCAXqgAwIBAgICAkkwDQYJKoZIhvcNAQEEBQAwazELMAkGA1UEBhMCVV
Mx
DzANBgNVBAoTBk9yYWNsZTEoMCYGA1UECxMfRW50ZXJwcmlzZSBBcHBsaWNhdG
lv
biBTZXJ2aWNlczEhMB8GA1UEAxMYRUFTUUEgQ2VydGlmaWNhdGUgU2VydmVyMB
4X
DTk5MDcyNjE3MzkyNloXDTAwMDEyMjE3MzkyNlowPTELMAkGA1UEBhMCVVMxDz
A
BgNVBAoTBm9yYWNsZTEOMAwGA1UECxMFZm9ybXMxDTALBgNVBAMTBGFtYXIw
-----END CERTIFICATE-----
```

証明書を受け取った後、作成済の Wallet にそれをインポートする必要があります。2 つのいずれかの方法でインポートできます。

- 認証局から受信した電子メールから、証明書を貼り付けます。
- ファイルから証明書をインポートします。

証明書を貼り付けるには、次のようにします。

1. Oracle Wallet Manager のメニューバーから、「**Operations**」→「**Import User Certificate**」をクリックします。「Import User Certificate」ダイアログ・ボックスが開きます。
2. 「**Paste the Certificate**」ラジオ・ボタンをクリックして、「**OK**」をクリックします。「Import User Certificate」ダイアログ・ボックスが開き、次のメッセージが表示されます。"Please provide a base64 format certificate and paste it below"。
3. 受信した電子メールの本文または Web ページから、証明書をコピーします。
4. 証明書をウィンドウに貼り付けて、「**OK**」をクリックします。ウィンドウの下部に、証明書が正常にインストールされましたというメッセージが表示されます。
5. 「**OK**」をクリックします。Oracle Wallet Manager のメイン・パネルが再び表示され、証明書が User Certificates ツリーの一番下に表示されます。
6. 「**Wallet**」→「**Save**」をクリックして、Wallet をディスクに保存します。

証明書を含むファイルをインポートするには、次のようにします。

1. メニューバーから「**Operations**」→「**Import User Certificate**」をクリックします。
2. 証明書が保存されている場所のパスを「Import User Certificate」ダイアログ・ボックスに入力します。
3. 証明書ファイルの名前（cert.txt など）をクリックして選択します。
4. 「**OK**」をクリックします。ウィンドウの下部に、証明書が Wallet に正常にインポートされましたというメッセージが表示されます。
5. 「**OK**」をクリックして、ダイアログ・ボックスを閉じます。Oracle Wallet Manager のメイン・パネルが再び表示され、証明書が User Certificates ツリーの一番下に表示されます。
6. 「**Wallet**」→「**Save**」をクリックして、Wallet をディスクに保存します。

5.7.3.6「Auto Login」を「ON」に設定

Oracle Wallet Manager Auto Login 機能により、Wallet のコピーが自動的に開かれます。そのため、Wallet にパスワードを提供する必要がなく、サーバー認証が行われます。

「Auto Login」を「ON」に設定するには、次のようにします。

1. メニューバーから「**Wallet**」をクリックします。
2. 「**Auto Login**」メニュー項目の隣のチェックボックスをクリックします。これにより、cwallet.sso という名前のファイルが作成されます。このファイルはマシン依存で、あるマシンから別のマシンにはコピーできません。
3. 「Autologin enabled」というメッセージが、ウィンドウの下に表示されます。

注意： 「Auto Login」メニュー項目の隣のチェックボックスは、クリックするたびにオンとオフが切り替わります。チェックマークを消去するには、再度チェックボックスをクリックします。今度は autologin が使用不可になります。

注意： サーバー認証を提供するすべての Forms Services マシンに対して、「Auto Login」を「ON」に設定する必要があります。

5.8 次のステップ

Forms Services の構成が完了したら、アプリケーションを Web に配置できます。詳細は、第 6 章「Web へのフォームの配置」を参照してください。

6

Web へのフォームの配置

6.1 概要

この章には、Oracle Forms アプリケーションを Web に配置するための情報が含まれます。Forms Services の構成後、実行可能ファイルを配置してアプリケーションの URL をブロードキャストできます。Forms Services の構成に関する情報は、第 5 章「Forms Services の構成」を参照してください。

6.2 Forms アプリケーションの配置

Forms アプリケーションを配置するには、次のステップに従います。

- ランタイム実行可能ファイルを作成します。
- 実行可能ファイルをサーバー上に配置します。
- アプリケーションの URL をブロードキャストします。

6.2.1 ランタイム実行可能ファイルの作成

.FMX ランタイム実行可能ファイルは、配置する先のアプリケーション・サーバーと同じプラットフォーム上で作成する必要があります。

たとえば、アプリケーション・サーバーのオペレーティング・システムが Sun Solaris の場合は、Web に配置する .FMX ファイルを作成するのに、Solaris 版の Forms Compiler コンポーネントを使用する必要があります。

Sun Solaris オペレーティング・システム用に .FMX ファイルをコンパイルするには、次の f60genm コマンドラインを使用します。

```
f60genm module=mymodule.fmb userid=scott/tiger
```

Forms Compiler オプションの詳細は、オンライン・ヘルプを参照してください。

6.2.2 実行可能ファイルのサーバー上への配置

Forms アプリケーションの実行可能ファイルは、サーバー上の任意のディレクトリから配置できます。FORMS60_PATH 環境変数でこのディレクトリを指定しておく必要があります。

6.2.3 アプリケーションの URL のブロードキャスト

アプリケーションの URL のブロードキャストで必要なのは、対象ユーザーへの通知のみです。ユーザーは Java 対応の Web ブラウザを使用してその URL に接続し、該当するアプリケーションを実行できます。アプリケーション用の HTML ページを作成した場合は、ユーザーに与える URL は単にそのページを指すようにします。

たとえば、ABC 社で新しい受注追跡アプリケーションが使用可能になったことをアナウンスするには、次の URL をブロードキャストします。

```
http://www.abc.com:80/servlet/f60servlet?config=order
```

注意： HTTPS モードで実行している場合は、"http" ではなく "https" を使用してください。（Oracle JInitiator ではオプションです。）

ABC 社の URL は次のコンポーネントで構成されます。

- **プロトコル :** http（または https）
- **ドメイン :** www.abc.com
- **Web サーバー・リスナー・ポート :** 80（黙示的）
- **Forms Servlet:** /servlet/f60servlet
- **formsweb.cfg での特別な構成設定 :** config=order

5.5.2.1 項「formsweb.cfg での特別な構成の作成」で説明されているように、formsweb.cfg 内に特別な構成を作成した場合、エンド・ユーザーは次のようにしてアプリケーションを起動します。

```
http://server:port/servlet/f60servlet?config=specialConfigName
```
（サーブレット構成の場合）

```
http://myhost.mydomain.com/dev60cgi/ifcgi60.exe?config=specialConfigName
```
（NT 上での CGI 構成の場合）

6.2.4 Servlet エラー・ログ

Forms Servlet 実装を使用している場合、formsweb.cfg ファイルおよび FormsServlet.initArgs ファイル内の構成エラーはすべて jserv.log ファイルに記録されます。このファイルは <ORACLE_HOME>/apache/Jserv/logs にあります。

6.3 次のステップ

実行可能ファイルを Web サーバー上に配置し、アプリケーションの URL をブロードキャストした後は、Web ブラウザからのアプリケーションのテストと最適化を行いたくなるでしょう。

Forms アプリケーションを Web 上に配置するためのガイドラインとヒントは、第 7 章「アプリケーション設計に関する考慮事項」を参照してください。

Forms Services を使用してアプリケーションをインターネット上または他のネットワーク環境に配置する際に、チューニングで考慮すべき点の詳細は、第 11 章「パフォーマンス・チューニングに関する 考慮事項」を参照してください。

7

アプリケーション設計に関する考慮事項

7.1 概要

この章では、Web で利用できる Forms アプリケーションの設計に関するガイドラインとヒントを示します。次の項が含まれています。

- 一般的なガイドライン
- Forms アプリケーション設計のためのガイドライン
- Forms Services で使用されるアイコンとイメージの配置
- レポートの統合
- Web 上の Forms アプリケーションの機能制限

7.2 一般的なガイドライン

Web 配置アプリケーションの設計について一般的なガイドラインを次に示します。

- Web アプリケーションのパフォーマンスに影響を与えるネットワーク要因について十分に考慮します（セキュリティ・ファイアウォールとの相互作用、多量のユーザー負荷、およびアプリケーションやデータベース・サーバーに対する頻繁なネットワーク・ラウンドトリップなど）。
- フォームおよびレポートに挿入するイメージ項目と背景イメージの数を制限します。イメージを必要とするたびに、アプリケーション・サーバーからダウンロードする必要があります。
- ネットワーク接続を見直せる箇所では、接続を最適化します。
- 問合せを、できる限り効率的に実行できるように設計し、PS/SQL プログラム単位のコンパイルもれがないようにします。

7.3 Forms アプリケーション設計のためのガイドライン

Web で利用できる Forms アプリケーションの設計に関するヒントを示します。これらは次の項で詳細に説明します。

- ユーザー独自のテンプレート HTML ファイルの作成
- HTML アプリケーション・メニューの作成
- Forms Services での Oracle Designer の使用
- ネットワーク通信量の削減
- 不要なグラフィックとイメージの削除
- 標準フォントの選択

7.3.1 ユーザー独自のテンプレート HTML ファイルの作成

（Oracle が提供するテンプレートを変更して）ユーザー独自の HTML ファイル・テンプレートを作成することを検討してください。ユーザー独自のテンプレートを作成すると、標準の Forms Client アプレット・パラメータおよびパラメータ値をテンプレートに直接に指定できます。作成したテンプレートには、標準テキスト、ブラウザ・ウィンドウ・タイトル、またはイメージ（会社のロゴなど）を挿入でき、これらは Web で使用できるフォームを実行するときに参照できる、最初の Web ページに表示されます。標準パラメータ、値、および追加のテキストまたはイメージを追加すると、特定のアプリケーションのテンプレートをカスタマイズするために必要な作業量を減らすことができます。テキスト、イメージ、またはウィンドウ・タイトルを追加するには、テンプレート HTML ファイルに適切なタグを挿入します。

7.3.2 HTML アプリケーション・メニューの作成

Web に追加のアプリケーションを配置するときは、Web で使用できる様々なアプリケーションを 1 つにまとめたメニューを提供するために、単一の HTML ページを作成するようにしてください。このアプローチにより、利用または削除するすべてのアプリケーションの URL をブロードキャストする手間が省けます。使用できるアプリケーションの登録を変更する場合は、Web メニュー上のリンクが列挙されている箇所を変更します。このことで、ユーザーはメニュー URL に接続して、使用できるアプリケーションのリストから選択できるようになります。

7.3.3 Forms Services での Oracle Designer の使用

Forms Services は、Oracle Designer（32 ビット、リリース 1.3.2 以降）で生成されたフォームをサポートします。標準の Oracle Designer のフォーム生成テンプレート（ofg4pc1t.fmb および ofg4pc2t.fmb）を使用してフォーム定義とメニュー定義を生成する場合、Forms Services を使用して .FMX および .MMX ファイルをコンパイルし、ただちに Web 上でアプリケーションを実行できます。

7.3.4 ネットワーク通信量の削減

ユーザーが Web 上で Form Builder アプリケーションを操作する場合に発生するネットワーク・ラウンドトリップ数を減らすためには、アプリケーションにおける次の Form Builder 機能の一部またはすべてを削除する必要があります。

- **マウス・トリガー。**フォームに、When-Mouse-Click、When-Mouse-DoubleClick、When-Mouse-Down、および When-Mouse-Up トリガーを含めると、スピードとパフォーマンスに影響を与えます。Forms Client は、これらのトリガーのどれかが実行されるたびに、Forms Services と通信する必要があります（ネットワーク・ラウンドトリップが必要になります）。When-Mouse-Move トリガーは、実行ごとに発生するネットワーク・ラウンドトリップ数が多いため、サポートされません。
- **タイマー。**100 分の 1 秒ごとに実行されるタイマー指定をフォームに含めると、毎分 60,000 回のネットワーク・ラウンドトリップというパフォーマンスの劣化が起こります。フォーム内のタイマー数を削減するか、タイマーが実行されるタイミング間隔を変更します。

7.3.5 不要なグラフィックとイメージの削除

アプリケーションに表示されるイメージ項目と背景イメージの数を可能な限り減らします。アプリケーション・ユーザーに対してイメージが表示されるたびに、イメージはアプリケーション・サーバーからユーザーの Web ブラウザにダウンロードする必要があります。

Web アプリケーションで会社のロゴを表示する場合には、アプリケーションの起動時にダウンロードされる HTML ファイルに会社のロゴ・イメージを含めてください。会社のロゴを背景イメージとしてアプリケーションに挿入するかわりにこれを行ってください。会社ロゴを背景イメージとして指定した場合データベースまたはファイルシステムから検索して、ユーザーのマシンにダウンロードするトラフィックが繰返し発生します。

7.3.6 標準フォントの選択

すべてのプラットフォーム間でサポートされているフォントは、あまり有りません。たとえば、Sans Serif は Microsoft Windows アプリケーションで一般的に使用されていますが、UNIX では使用できません。フォントがプラットフォームで使用できない場合、Form Builder は類似したフォントの使用を試みます。このため、Web 上で利用するフォームを設計するときには、必ず次に示すフォント・ガイドラインに従ってください。

実行時に、Forms Services はフォームのフォントを Java 等価フォントに対応付けます。次に、JAVA は配置プラットフォームに定義済のフォントでフォントを表します。フォームのフォントを Java 等価フォントに変換するために、Java は Registry.dat と呼ばれるファイル内の別名リストを使用します。

次の表は、Java フォントと主要な配置プラットフォーム上の等価フォントの一覧です。

表 7-1

Java フォント	Windows フォント	X Windows フォント	Macintosh フォント
Courier	Courier New	adobe-courier	Courier
Dialog	MS Sans Serif	b&h-lucida	Geneva
DialogInput	MS Sans Serif	b&h-lucidatypewriter	Geneva
Helvetica	Arial	adobe-helvetica	Helvetica
Symbol	WingDings	itc-zapfdingbats	Symbol
TimesRoman	Times New Roman	adobe-times	Times Roman

Form Builder フォント別名表を用いてフォーム上のフォントを Java フォントに対応付けする際に対応付けできないフォントについては、Java は自動的に Java フォントを割り当てます。

7.4 Forms Services で使用されるアイコンとイメージの配置

この項では、アイコンとイメージのデフォルト・ディレクトリおよび検索パスの指定方法を説明します。

7.4.1 アイコン

Web 上に Forms アプリケーションを配置する場合、(アイコン・ボタン、メニューまたはウィンドウに指定した）ICO 形式のアイコン・ファイルは使用しません。Web を介して接続できるファイル形式は、GIF または JPG ファイルのみです（GIF がデフォルト形式です)。

デフォルトでは、アイコンは HTML ファイルを含むディレクトリ、DocumentBase ディレクトリにあります。アイコンを別のディレクトリに格納する場合は、アプリケーション・ファイルを作成して、アイコン・ファイルを常駐させる仮想ディレクトリおよび使用するファイル形式（GIF または JPG）を指定する必要があります。このアプリケーション・ファイルは HTML ファイルで参照する必要があります。

カスタム・アプリケーション・ファイルを作成するには、次の手順に従います。

1. <ORACLE_HOME>/6iserver/forms60/java/oracle/forms/registry ディレクトリにある registry.dat テキスト・ファイルを別のディレクトリへコピーします。このディレクトリは Web サーバーの仮想ディレクトリ（/appfile など）へマップする必要があります。
2. 新規ファイルをリネームします（myapp.dat など)。
3. アイコンのディレクトリを指定する iconpath パラメータを次のとおりに変更します。

default.icons.iconpath=/mydir or http://myhost.com/mydir
（絶対パスの場合）

または

default.icons.iconpath=mydir
（DocumentBase ディレクトリから始まる相対パスの場合）

4. iconextension パラメータを次のように変更します。

   default.icons.iconextension=gif

   または

   default.icons.iconextension=jpg

HTML ファイルでアプリケーション・ファイルを参照するには、次の手順に従います。

formsweb.cfg ファイルまたは HTML ファイルで、serverApp パラメータの値を変更し、値をアプリケーション・ファイルのディレクトリおよび名前に設定します。

```
<PARAM NAME="serverApp" VALUE="/appfile/myapp">
```
（絶対パスの場合）

または

```
<PARAM NAME="serverApp" VALUE="appfile/myapp">
```
（CodeBase ディレクトリを基準とする相対パスの場合）

7.4.2 スプラッシュ画面イメージおよびバックグラウンド・イメージ

アプリケーションを Web で実行する場合、（接続中に表示される）スプラッシュ画面イメージおよびバックグラウンド・イメージ・ファイルを指定するための機能が必要です。

これらのイメージは、次に示すように HTML ファイルまたは formsweb.cfg ファイルで定義します。

```
<PARAM NAME="splashScreen" VALUE="splash.gif">

<PARAM NAME="background" VALUE="back.gif">
```

スプラッシュ画面およびバックグラウンド・イメージ・ファイルのデフォルト・ディレクトリは、ベース HTML ファイルが含まれている DocumentBase ディレクトリ内にあります。

7.4.3 アイコンおよびイメージを含むカスタム JAR ファイルの使用

（スプラッシュ画面またはバックグラウンドの）アイコンまたはイメージを使用するたびに、HTTP リクエストが Web サーバーに送信されます。クライアントとサーバー間の HTTP ラウンドトリップ数を減らすには、Java アーカイブ（JAR）ファイルにアイコンおよびイメージを格納するための機能が必要です。この方法を使用すると、JAR ファイルをダウンロードするのに、1 回の HTTP ラウンドトリップのみで済みます。

7.4.3.1 JAR ファイルの作成

SunSoft JDK には、*jar* と呼ばれる実行可能ファイルが含まれています。このユーティリティを使用すると、Java アーカイブ内にファイルを格納できます。詳細は、www.java.sun.com を参照してください。

例 :

```
jar -cvf myjar.jar Splash.gif Back.gif icon1.gif
```

このコマンドにより、myjar.jar と呼ばれる単一の JAR ファイルに 3 つのファイル（Splash.gif、Back.gif、icon1.gif）が格納されます。

7.4.3.2 JAR ファイル内でのファイルの使用

アイコンおよびイメージのデフォルトの検索パスは、DocumentBase を基準とした相対パスです。ただし、それらのファイルを格納するために JAR ファイルを使用する場合は、検索パスは、Java アプレットを含むディレクトリ、CodeBase ディレクトリを基準とする相対パスにする必要があります。

JAR ファイルを使用してアイコンおよびイメージを格納する場合は、ベース HTML ファイル内の imageBase パラメータを使用して、検索パスが CodeBase を基準とした相対パスであることを指定する必要があります。

このパラメータは次の 2 つの異なる値が指定可能です。

- **DocumentBase** 検索パスは DocumentBase ディレクトリを基準とした相対パスです。これはデフォルトの動作です。
- **CodeBase** 検索パスは JAR ファイルを使用できるようにする CodeBase ディレクトリを基準とした相対パスです。

この例では、アイコンを含む JAR ファイルを使用して、検索が CodeBase の相対パスになるように指定します。パラメータ "imageBase" を設定していない場合は、DocumentBase を基準とした相対的な検索となり、アイコンは JAR ファイルからは検索されません。

例 :

<PARAM NAME="archive" VALUE="icons.jar">

<PARAM NAME="imageBase" VALUE="CodeBase">

7.4.4 アイコンおよびイメージの検索パス

アイコンおよびイメージの検索パスは次の内容によって異なります。

- カスタム・アプリケーション・ファイルで指定した内容（アイコンの場合）
- HTML ファイルの SplashScreen パラメータおよび Background パラメータで指定した内容（イメージの場合）
- HTML ファイルの imageBase パラメータで指定した内容（アイコンとイメージの両方の場合）

Forms Services では、指定した内容に応じてアイコンが検索されます。この例では、次のように仮定します。

- *host* はホスト名。
- *documentbase* は HTML ファイルを指す URL。
- *codebase* は、（HTML ファイルで指定した）開始クラス・ファイルのディレクトリを指す URL。
- *mydir* は、アイコンまたはイメージ・ディレクトリを指す URL。

7.4.4.1 DocumentBase

デフォルトの検索パスは、DocumentBase を基準とした相対パスです。この場合、imageBase パラメータを指定する必要はありません。

表 7-2

	指定ディレクトリ	Forms Services で使用される検索パス
アイコン	デフォルト	http://host/documentbase
	iconpath=mydir （アプリケーション・ファイルで指定）	http://host/documentbase/mydir （相対パス）
	iconpath=/mydir （アプリケーション・ファイルで指定）	http://host/mydir （絶対パス）
イメージ	file.gif （HTML ファイルで指定）	http://host/documentbase/file.gif
	mydir/file.gif （HTML ファイルで指定）	http://host/documentbase/mydir/file.gif （相対パス）
	/mydir/file.gif （HTML ファイルで指定）	http://host/mydir/file.gif （絶対パス）

7.4.4.2 CodeBase

次に示すように、基本の HTML ファイルで imageBase=CodeBase パラメータを使用して、JAR ファイル内でのアイコンおよびイメージの検索を可能にします。

表 7-3

	指定ディレクトリ	Forms Services で使用される検索パス
アイコン	デフォルト	http://host/codebase または JAR ファイルのルート
	iconpath=mydir （アプリケーション・ファイルで指定）	http://host/codebase/mydir または JAR ファイル内の mydir ディレクトリ （相対パス）
	iconpath=/mydir （アプリケーション・ファイルで指定）	http://host/mydir （絶対パス） JAR ファイルは使用されない。
イメージ	file.gif （HTML ファイルで指定）	http://host/codebase/file.gif または JAR ファイルのルート
	mydir/file.gif （HTML ファイルで指定）	http://host/codebase/mydir/file.gif または JAR ファイル内の mydir ディレクトリ （相対パス）
	/mydir/file.gif （HTML ファイルで指定）	http://host/mydir/file.gif （絶対パス） JAR ファイルは使用されない。

7.5 レポートの統合

Web で使用できるフォームからレポートを起動するには、RUN_PRODUCT ビルトイン・サブプログラムを使用します。

RUN_PRODUCT を使用して Web 上で実行中のフォームからレポートを実行するには、次に示す 3 つの環境変数を設定する必要があります。

表 7-4

環境変数	説明
FORMS60_OUTPUT	生成したレポート・ファイルを格納するアプリケーション・サーバー上の物理ディレクトリ。 例 : <ORACLE_HOME>/6iserver/tools/web60/temp
FORMS60_MAPPING	FORMS60_OUTPUT 変数で定義された物理ディレクトリを指す仮想ディレクトリ。 例 : /dev60temp/
FORMS60_REPFORMAT	生成したレポート出力を格納する形式。 例 : PDF または HTML

Windows NT では、レジストリ内で環境変数を定義します。UNIX では、コマンド・シェル内で環境変数を定義します。環境変数設定の詳細は、付録 A「Forms Services のパラメータ」を参照してください。

前述の環境変数を設定すると、Web 上で実行中のフォームが RUN_PRODUCT をコールしてレポートを起動する場合に、次の順序で自動的に発生します。

レポートの出力形式が SCREEN または PREVIEW の場合は、次のようになります。

- 結果出力は、FORMS60_OUTPUT 環境変数で指定した物理ディレクトリに（自動生成したファイル名のテンポラリ・ファイルとして）格納されます。
- Web サーバーは、（FORMS60_MAPPING 環境変数で定義した仮想ディレクトリ内で）テンポラリ・ファイル名を検索します。
- Web サーバーは、FORMS60_REPFORMAT 環境変数で指定した目的の表示形式をチェックして、ユーザーのブラウザにその形式でレポートを表示します。

レポートの出力形式が FILE の場合は、次のようになります。

- レポートはユーザーのブラウザで表示されません。
- 結果を示すファイルは、FORMS60_OUTPUT 環境変数で指定された物理ディレクトリに格納されます。
- レポート・ファイルのファイル名は、フォーム定義で定義した名前と同じ名前になります。

7.6 Web 上の Forms アプリケーションの機能制限

Web に配置するフォームを作成するときは、Forms の機能には、Web に配置されると異なる動作をする場合や、あるいはまったく動作しない場合があるということに注意してください。表 7-5 には、フォームの機能を一覧表示しています。機能が Web でサポートされているかどうか、またその機能に関するガイドラインやメモが記載されています。

表 7-5

機能	サポート	ガイドラインと注意事項
ActiveX、OCX、OLE、VBX	なし	ユーザーが出力を表示できないためアプリケーション・サーバー上に画面表示するサード・パーティ・コントロールはサポートされていません。
When-Mouse-Enter / Leave / Move トリガー	なし	トリガーを実行するたびに、ネットワーク・ラウンドトリップが必要になり、その結果パフォーマンスが低下します。
コンソール	あり	（ステータスとメッセージ行を含む）コンソールを表示するには、フォーム・レベル・プロパティのコンソール・ウィンドウを、コンソールを表示するウィンドウに設定します。
ファイアウォール	あり	Forms Services を HTTP または HTTPS モードで実行する必要があります。また、HTTP 1.1 プロトコルをサポートするファイアウォールが必要です。
HOST_COMMAND、ORA_FFI、USER_EXIT	あり	これらの機能をコールすると、可視出力または GUI 要素が、クライアント / サーバー・モードのユーザーのマシン上に表示されることがよくあります。Web 実装では、同じ機能をコールすると、アプリケーション・サーバー上に出力と GUI 要素を表示します（ユーザーは、それらを参照またはそれらと対話できない）。
アイコン・ボタン	あり	アイコン・イメージ・ファイルは GIF 形式にします（ICO 形式は使用できない）。
NLS、BIDI	あり	8 ビット言語のみサポートされます。

8

これまでのアプリケーション資産の Web への移行

8.1 概要

現在 Forms Server のクライアント / サーバー・バージョンを使用している場合は、アプリケーションを Web 用の Forms Services へ簡単に移行できます。この章では、クライアント / サーバー実装と Web 実装間の違いを簡単に説明し、現在使用しているアプリケーションをクライアント / サーバー・ベースから Web ベースの Forms Services へ移行するためのガイドラインを示します。

従来は、Oracle Forms Server のロード・バランス・サービスはカートリッジを介して提供されていました。カートリッジのかわりにサーブレットの実装によってフォームを Web に配置する場合は、ロード・バランスは使用できませんでした。

Oracle9*i* Application Server Forms Services のリリースにより、サーブレットの実装を介して Web に配置した Forms アプリケーションでも、ロード・バランスを使用できるようになりました。ロード・バランスを使用すると、ハードウェアの使用限界に近づいたときにマシンのアップグレードや交換をしなくても、単にアプリケーションを実行するマシンを追加して、サーバー通信量の負荷をいくつかのマシン間に分散することで問題を解決できます。

すでにカートリッジを使用して Web ベースの Forms Developer アプリケーションを配置済で、これからサーブレットに切り替える場合は、Forms Services をインストールし、サーブレット用に構成する必要があります。この章では、既存のカートリッジ・ベースの実装を使用しているユーザーを対象として、カートリッジからサーブレットへ切り替えるためのインストールまたは再構成について説明します。

8.1.1 クライアント / サーバー・ベースのアーキテクチャ

クライアント / サーバー・ベースの実装では、図 8-1 に示すように、Forms Server Runtime エンジンおよびすべてのアプリケーション・ロジックはユーザーのデスクトップ・マシンにインストールされます。いくつかのアプリケーションで指定されるデータベース・サーバー側のトリガーおよびロジック以外の、すべてのユーザー・インタフェース処理およびトリガー処理は、クライアント上で行われます。

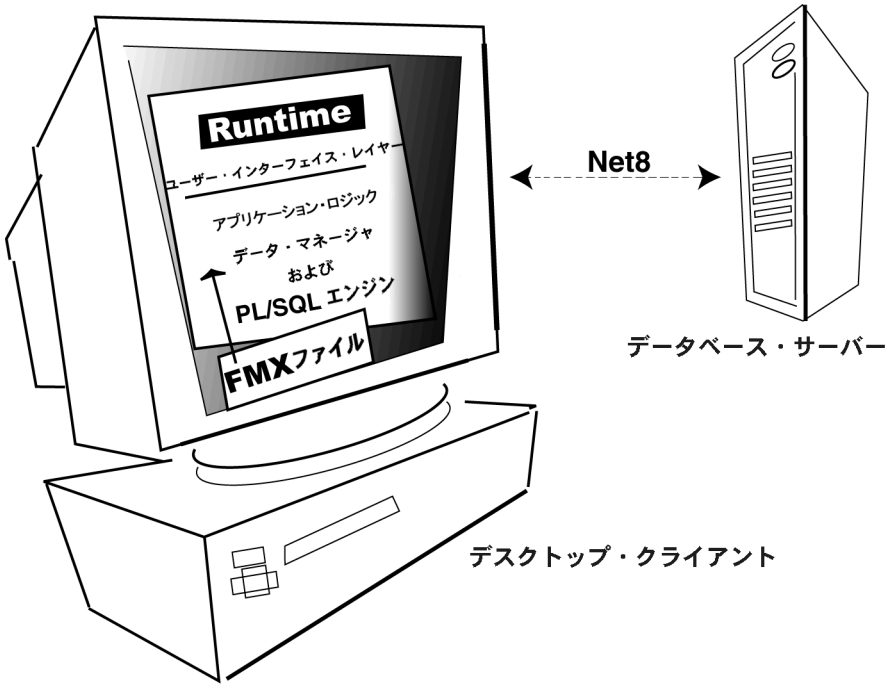


図 8-1 Forms Server のクライアント / サーバー・ベースのアーキテクチャ

8.1.2 Web ベースのアーキテクチャ

Web ベースの実装では、図 8-2 に示すように、Forms Services Runtime エンジンおよびすべてのアプリケーション・ロジックは、クライアント・マシンではなくアプリケーション・サーバーにインストールされます。すべてのトリガー処理はデータベースおよびアプリケーション・サーバーで行われますが、ユーザー・インタフェース処理は、ユーザーのマシンにある Forms クライアントで行われます。

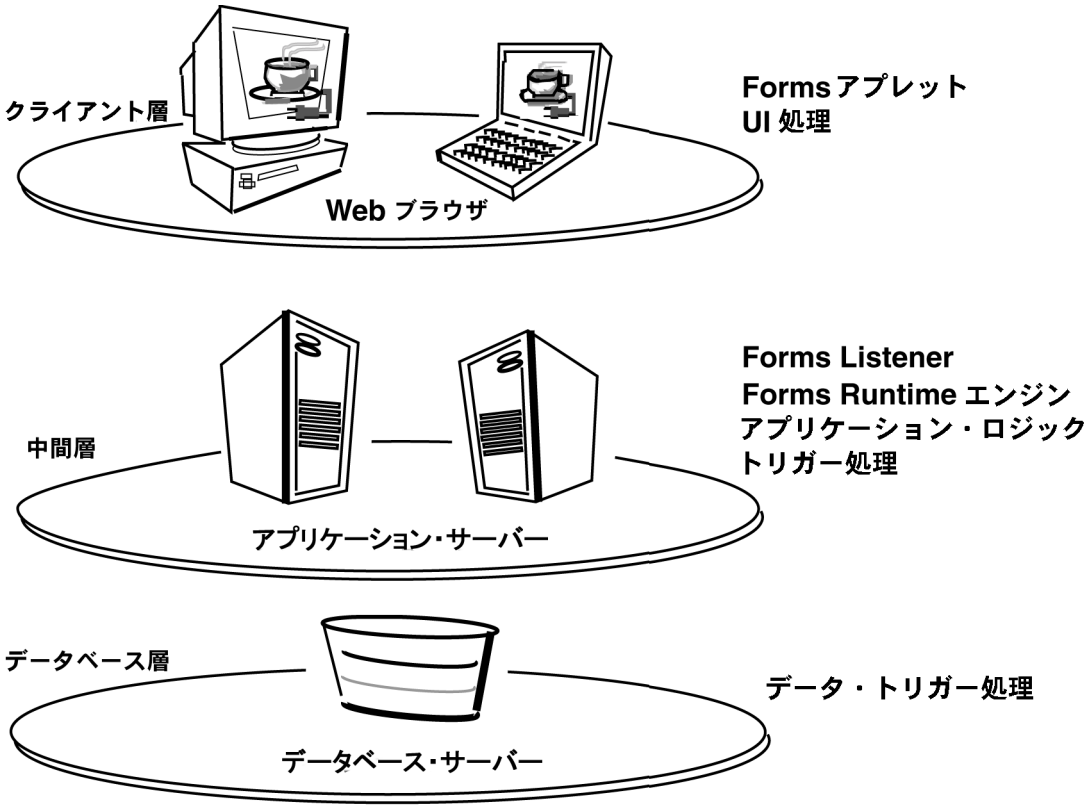


図 8-2 Forms Services の Web ベースのアーキテクチャ

8.1.3 この章の対象読者

この章は、次の項目がユーザーの配置環境に当てはまる場合に役立ちます。

- 現在、Web ベースの Oracle Forms Developer アプリケーションを配置している。
- Oracle Application Server を使用して Web サーバーをサポートしている。
- Web ベースの Oracle Forms Developer アプリケーションを、カートリッジを使用して配置している。
- カートリッジでの配置からサーブレットへ移行したい。

8.2 カートリッジ実装とサーブレット実装の比較

カートリッジ実装とサーブレット実装ではいずれも、ポート番号や対象ファイルの場所などの値を定義するサーバー操作パラメータの設定が必要となりますが、それらを設定する場所が異なります。Oracle Application Server では、Oracle Application Server Manager を開き、異なる配置エンティティのパラメータを設定するために様々な接続先にナビゲートします。Forms Services では、構成上の複雑さは 1 箇所に集中しています。インストール中にユーザーが行う構成上の選択によって、多くの操作パラメータが自動的に設定されます。ユーザーは、インストール中に作成される Forms Services の formsweb.cfg ファイルを変更したり、追加の操作パラメータを設定したりできます。

カートリッジ実装とサーブレット実装のいずれにおいても、標準のベース HTML ファイルに基づいた HTML ファイルがその場で作成されます。カートリッジ実装では、HTML ファイルは cartridg.html ファイル、カートリッジ構成設定、およびアプリケーションの URL を組み合せて作成されます。Forms サーブレット実装では、HTML ファイルは base.htm または basejini.htm ファイル、formsweb.cfg ファイル、およびアプリケーションの URL を組み合せて作成されます。

カートリッジ・ベースとサーブレット・ベースのいずれの HTML ファイルでも、パラメータを変数で定義できます。変数値は、アプリケーションのカートリッジ設定（Oracle Application Server）、formsweb.cfg ファイル（Forms Services）、またはアプリケーションの URL 内の問合せ文字列（Oracle Application Server と Forms Services の両方）で定義できます。

カートリッジ実装とサーブレット実装の大きな違いは、Oracle Application Server 以外の Web サーバーを介して提供されるサービスの種類とパフォーマンスのレベルにあります（Oracle Application Server を介して提供されるものと比較して）。Forms Services では広範囲の操作パラメータが利用できるようになり、Forms Services のインストールと formsweb.cfg ファイルによって、フォーム・パラメータの設定処理が大幅に単純化されました。

8.3 再構成の方法

この項では、構成処理の高レベルな概要を説明します。Oracle Application Server、Forms Services、ベース HTML ファイルなどを技術的に理解しているユーザー向けです。

カートリッジ配置からサーブレットへ再構成するには、次の 2 つの基本的な方法があります。

- formsweb.cfg ファイルのカートリッジ・パラメータを複製し、すべてを同じにする。
- デフォルトの Oracle9*i* Application Server のインストールを使用する。

最初の方法は、複雑なベース HTML ファイルを使用するユーザーに適しています。このファイルには、Forms アプレット・タグに加え、多くの外部テキスト、イメージおよびその他のオブジェクトが含まれます。2 番目の方法は、単純なベース HTML ファイルを使用するユーザーに適しています。

8.3.1 複雑なベース HTML ファイルを使用するユーザー向けの方法

複雑なベース HTML ファイルを使用するユーザー向けの方法は、すべてを同じままにすることです。

1. 使用しないすべての Oracle Application Server インスタンスを停止します。
2. Oracle9*i* Application Server をインストールします。
3. formsweb.cfg ファイルに、カートリッジでの構成時に使用していたパラメータをコピーします。

   現在のカートリッジ・パラメータの場所を知るには、Oracle Application Server を起動し、それぞれの Forms アプリケーションの Cartridge Configuration フォルダへナビゲートします。各フォルダ内で、「カートリッジ・パラメータ」をクリックします。これにより、カートリッジ・パラメータ設定が表示されます。また、Forms カートリッジ・アプリケーション用に作成したベース HTML ファイルでもパラメータ設定を参照できます。
4. Forms カートリッジ用に複数のカートリッジ定義を使用していた場合（つまり、複数のベース HTML ファイルを使用していた場合）は、カートリッジごとに別々の構成セクションを formsweb.cfg ファイルに定義します。（いずれの方法でも、formsweb.cfg ファイルにおいて、名前が付けられたセクション外のファイルの先頭にパラメータを指定するのではなく、カートリッジ・パラメータ用の新しい構成セクションを作成することをお薦めします。）
5. Oracle Application Server 以外の Web リスナーの場合は、Oracle Application Server で使用していた仮想パスと同じパスを定義します。次のように、サーブレットがあるディレクトリをポイントする、スクリプトの新しい仮想パスを追加します。

   virtual_path_name = /servlet/
   physical_path = <ORACLE_HOME>/6iserver/forms60/java/oracle/forms/servlet

6. フォームの実行に使用するすべての URL を、カートリッジではなくサーブレットをポイントするように変更します。たとえば、元の URL が次のとおりであったとします。

   ```
   http://servername.my.domain.com/developerforms/forms60cart?module=emp.fmx
   ```

   この URL を次のように変更します。

   ```
   http://server:port/servlet/f60servlet?config=emp
   ```

注意： HTTPS 通信モードを使用している場合は、上記の例で "http" ではなく "https" を使用してください。（Oracle JInitiator ではオプションです。）

この例では、"myconfig" は、formsweb.cfg ファイルに定義した構成セクションの名前です。このファイルには、以前のカートリッジ・パラメータと同じパラメータが含まれています。

8.3.2 単純なベース HTML ファイルを使用するユーザー向けの方法

単純なベース HTML ファイルを使用するユーザー向けの方法は、デフォルトの Oracle9*i* Application Server のインストールを使用することです。

カートリッジ実装で単純なベース HTML ファイルを使用していた場合、サーブレットへの再構成では、インストール中に自動的に作成されるデフォルトの構成を簡単に利用できます。

1. 使用しないすべての Oracle Application Server インスタンスを停止します。
2. Oracle9*i* Application Server をインストールします。
3. カートリッジを使用していたときと同じ効果を得るために、フォームの実行に使用していた URL を応用します。formsweb.cfg ファイルで定義されたユーザー独自のパラメータを使用します。このようにアプリケーションの URL に値を指定することによって、すべての HTML と Forms アプレットのパラメータ値を変更できます。同じ URL は、AppletViewer、Oracle Jinitiator と組み合せて使用する Web ブラウザ、または Internet Explorer 5.0 の各ユーザーも使用できます。
4. runform.htm ファイルを使用して、アプリケーションの URL での異なるパラメータ設定をテストできます。たとえば、ページ・タイトルが "My Form"、ページ幅が 400、ページ高が 550 のフォームを実行するために使用する URL は次のように指定します。

   ```
   http://server:port/servlet/f60servlet?pagetitle=My+Form&width=400&height=550
   ```

注意： HTTPS 通信モードを使用している場合は、上記の例で "http" ではなく "https" を使用してください。（Oracle JInitiator ではオプションです。）

これらの例では、疑問符はアプリケーション URL 内の問合せ文字列の始まりを表します。問合せ文字列では、ページ・タイトル、幅、高さの各パラメータの値を指定します。

8.4 Forms Web カートリッジからサーブレットへの再構成

フォームの配置をカートリッジからサーブレットに再構成するには、次のステップを実行します。

1. 使用しない Oracle Application Server Web リスナー・インスタンスを停止します。
2. Oracle9*i* Application Server をインストールします。
3. Forms Services の formsweb.cfg ファイルを構成します。
4. オプションで、Forms Services の base.htm ファイルおよび basejini.htm ファイルを構成します。
5. アプリケーションの URL をブロードキャストします。

8.4.1 Oracle Application Server Web リスナー・インスタンスの停止

Oracle Application Server を停止するには、次の 2 つの方法があります。

- 完全に停止する。
- 特定のインスタンスのみを停止し、他のインスタンスは Oracle Application Server で継続してサポートする。

8.4.1.1 Oracle Application Server の完全停止

Oracle Application Server によって提供されるすべてのサービスの使用を停止する場合、この方法を使用します。

1. Oracle Application Server を起動します。
2. Oracle Application Server Manager を開きます。
3. Oracle Application Server インストールのトップレベルのサイトにナビゲートします。
4. すべてを選択します。
5. 「停止」ボタンをクリックします。

8.4.1.2 Oracle Application Server の特定インスタンスの停止

いくつかの Oracle Application Server HTTP リスナーのみを停止し、他の実行は継続する場合、この方法を使用します。

1. Oracle Application Server を起動します。
2. Oracle Application Server Manager を開きます。
3. HTTP リスナーにナビゲートします。

4. カートリッジからサーブレットへ変換するポートで実行している HTTP リスナーを選択します。
5. 「停止」ボタンをクリックします。

8.4.2 formsweb.cfg ファイルの構成

formsweb.cfg ファイルは、Forms Services に含まれる、強力で便利な新しいファイルです。このファイルは、サーブレット実装において Oracle フォームを Web 上で実行するために必要となる、すべての設定のリポジトリとして使用します。このファイルはインストーラによって、<ORACLE_HOME>/6iserver/forms60/server に格納されます。

formsweb.cfg ファイルは、サーブレット実装において Forms アプリケーションを Web 上で実行するための構成パラメータを含むテキスト・ファイルです。formsweb.cfg ファイル内の構成パラメータは、Forms カートリッジで使用されるカートリッジ・パラメータと等価なものです。formsweb.cfg ファイルは、次に示す 3 つの主なセクションに分けられます。

- SYSTEM PARAMETERS
- USER PARAMETERS
- SPECIFIC CONFIGURATIONS

formsweb.cfg ファイルの構成の詳細は、第 5.5.2 章「formsweb.cfg」を参照してください。

8.4.2.1 SYSTEM PARAMETERS

SYSTEM PARAMETERS セクションでは、Forms サーブレットが必要とする情報を提供します。formsweb.cfg 内の他の多くのパラメータとは異なり、SYSTEM PARAMETERS は URL 問合せ文字列で指定できません。ただし、代替のパラメータ、値のセットを formsweb.cfg 内の SPECIFIC CONFIGURATION セクションに指定し、その構成をアプリケーション URL で呼び出すことによって、SYSTEM PARAMETERS の値を上書きできます。

8.4.2.2 USER PARAMETERS

USER PARAMETERS セクションでは、ベース HTML ファイルの変数で定義されたパラメータの実際の値を指定します。たとえば、base.htm ファイルに次のように定義されているとします。

<PARAM NAME="separateFrame" VALUE="%separateFrame%">

formsweb.cfg では、次のように %separateFrame% 変数に特定の値を設定できます。

separateFrame=false

指定した USER PARAMETER 値は、formsweb.cfg 内の SPECIFIC CONFIGURATION セクションまたはアプリケーションの URL の問合せ文字列で上書きできます。

例 :

```
http://server:port/servlet/f60servlet?separateFrame=true
```

注意： HTTPS 通信モードを使用している場合は、上記の例で "http" ではなく "https" を使用してください。（Oracle JInitiator ではオプションです。）

これらの例では、?separateFrame=true という問合せ文字列は、formsweb.cfg ファイルで指定されている separateFrame の値を上書きします。

formsweb.cfg ファイルとアプリケーションの URL の両方でパラメータの特定の値が定義されている場合、URL で定義された値が使用されます。

8.4.2.3 SPECIFIC CONFIGURATIONS

同じフォームを複数の構成で実行する場合、formsweb.cfg ファイル内の SPECIFIC CONFIGURATIONS セクションでカスタム値を設定することにより、カスタム構成を定義できます。

アプリケーションの URL の問合せ文字列でカスタム構成を呼び出す場合は、formsweb.cfg 内の USER PARAMETERS セクションで定義したパラメータがカスタム値で上書きされます。SPECIFIC CONFIGURATIONS セクションを設定する場合は、必要となるのは変更するパラメータを指定することだけです。USER PARAMETERS セクションで指定したデフォルト値は、他のすべてのパラメータで使用されます。

特定の SPECIFIC CONFIGURATION セクションを呼び出すには、アプリケーションの URL で "config" パラメータを使用します。たとえば、次の URL では、[myconfig] という SPECIFIC CONFIGURATION セクションを呼び出します。

```
http://server:port/servlet/f60servlet?config=myconfig
```

注意： HTTPS 通信モードを使用している場合は、上記の例で "http" ではなく "https" を使用してください。（Oracle JInitiator ではオプションです。）

formsweb.cfg ファイルの構成の詳細は、第 5.5.2 章「formsweb.cfg」を参照してください。

8.4.3 base.htm ファイルまたは basejini.htm ファイルの構成

Web 対応のアプリケーションを（アプリケーションの URL へのリンクをクリックすることにより）起動するとき、Forms サーブレットは特別なファイルを読み込みます。このファイルには、選択したアプリケーションを Web 上で実行するために必要となる、すべてのアプレット・タグ、パラメータ、パラメータ値（またはこれらの値用の変数）が含まれています。これがベース HTML ファイルです。

Oracle Universal Installer によって、2 つのベース HTML ファイルがディレクトリ <ORACLE_HOME>/6iserver/FORMS60/server に格納されます。

- basejini.htm

  このファイルには、ユーザーの Web ブラウザと Oracle JInitiator の組合せによって Forms アプレットを実行するために必要となるタグが含まれています。

- base.htm

  このファイルには、AppletViewer または（ネイティブな JVM が Forms で動作することがオラクル社によって確認済の）任意の Web ブラウザで Forms アプレットを実行するために必要となるタグが含まれています。

base.htm ファイルおよび basejini.htm ファイルの詳細は、第 5.5.3 章「base.htm、basejini.htm および baseie.htm」を参照してください。

Forms Web サーブレットの実装では、アプリケーション起動プロセスが開始するときに、ベース HTML ファイル内のすべての変数（%variablename%）が適切なパラメータ値と置き換えられます。このパラメータ値は、formsweb.cfg ファイルで指定するか、アプリケーションの URL に含まれる問合せ文字列で指定します。一度すべての値が定義されると、HTML ファイルが生成され、その後ユーザーの Web ブラウザにダウンロードされて、選択した Forms アプリケーションが起動します。

formsweb.cfg ファイルとアプリケーションの URL の両方でパラメータの特定の値が定義されている場合、URL で定義された値が使用されます。

ほとんどの場合、デフォルトのベース HTML ファイルを変更する必要はありません。かわりに、それらのパラメータを変数として指定できます。変数の実際の値は、formsweb.cfg またはアプリケーションの URL で定義できます。

たとえば、ベース HTML ファイルでは、splashScreen というパラメータを次のように指定します。

<PARAM NAME="splashScreen" VALUE="%splashScreen%">

その後、実際の値を formsweb.cfg ファイルで次のように指定します。

splashScreen=virtual_path/mysplashscreen.gif

注意： HTTPS 通信モードを使用している場合は、上記の例で "http" ではなく "https" を使用してください。（Oracle JInitiator ではオプションです。）

ベース HTML ファイル内で値のかわりに変数を使用すると、すべてのフォーム・アプリケーションで同一のベース HTML ファイルを汎用的に使用でき、formsweb.cfg（またはアプリケーションの URL）のみで複雑な構成を一元管理できます。

ベース HTML ファイルでパラメータ値を直接指定する場合、オラクル社から提供される元のベース HTML ファイルは修正しないでください。かわりに、別名で保存したコピーのファイルを修正します。その際、formsweb.cfg ファイル内の baseHTML（または basejiniHTML）パラメータを必ず変更して、修正したファイルの場所を指定するようにしてください。

8.4.4 アプリケーションの URL のブロードキャスト

アプリケーションの URL をブロードキャストするには、URL を対象ユーザーに知らせます。ユーザーは Java 対応の Web ブラウザを使用してその URL に接続し、該当するアプリケーションを実行できます。たとえば、ABC 社で新しい受注追跡アプリケーションが使用可能になったことをアナウンスするには、次の URL を電子メールで従業員に通知します。

```
http://server:port/servlet/f60servlet?config=myconfig&form=tracker.fmx
```

注意： HTTPS 通信モードを使用している場合は、上記の例で "http" ではなく "https" を使用してください。（Oracle JInitiator ではオプションです。）

ABC 社の URL は次のコンポーネントで構成されます。

http または https	接続プロトコル
server:port	アプリケーション・サーバーを管理するマシン名
servlet	Web サーバーで定義された、サーブレットをポイントする仮想パス
f60servlet	Forms サーブレット
?config=myconfig&form=tracker.fmx	formsweb.cfg ファイル内のユーザー作成の "myconfig" セクションで定義したカスタム構成、および tracker.fmx というフォーム・モジュールを指定する問合せ文字列。 ベース HTML ファイルでは "form" が "module" の変数値として定義されているため、ここではパラメータ "form" が使用されます。つまり、次のように定義されています。 <PARAM NAME="serverArgs" VALUE="module=%form%"> ベース HTML の JInitiator ファイルでは、次のように構文が少し異なります。 serverArgs="module=%form%"

8.5 移行に関するガイドライン

アプリケーションをクライアント / サーバーでの配置から Web へ移行する場合、Web ベースのアプリケーションでは次の点に注意してください。

- JPEG および GIF イメージ・タイプのみがサポートされているので、既存のイメージをこれらの形式に変換してください。
- ファイル転送用に圧縮された JAR（Java アーカイブ）ファイルの使用がサポートされているので、Forms Services と Java クライアント間で大きなファイルの転送が必要な場合は、常に JAR ファイルを使用してください。
- ユーザー・インタフェースでは ActiveX、OCX、OLE または VBX コントロールはサポートされません。かわりに、JavaBeans をユーザー・インタフェースで使用して、機能を再現してください。その他の Microsoft Windows ユーザー・インタフェースに依存するものも JavaBeans で置換してください。
- When-Mouse-Enter、When-Mouse-Leave および When-Mouse-Move などの MouseMove トリガーはサポートされません。
- クライアントのハード・ドライブへの書込みアクセスは本来サポートされません。これは、トランスポータブルな Forms ユーザー・インタフェース用の JavaBeans を書くことで実行できます。
- Java フォントのみがサポートされているので、使用するフォントのタイプについてアプリケーションをチェックしてください。必要に応じて、Java フォントに変換します。Java では、Registry.dat ファイルにあるフォント別名リストが使用されます。次の表 8-1 に示すフォント別名がサポートされます。

表 8-1 Web ベースのアプリケーションでのフォントのサポート

Java フォント	Windows フォント	XWindows フォント	Macintosh フォント
Courier	Courier New	adobe-courier	Courier
Dialog	MS San Serif	b&h-lucida	Geneva
DialogInput	MS San Serif	b&h-lucidatypewriter	Geneva
Helvetica	Arial	adobe-helvetica	Helvetica
Symbol	Wingdings	itc-zapfdingbats	Symbol
Times Roman	Times New Roman	adobe-times	Times Roman

この章では、Forms カートリッジの実装をサーブレットで再構成するための情報を提供しました。移行を円滑かつ確実に行えるよう、説明している構成オプションは限定的なものになっています。

9

ネットワークに関する考慮事項

9.1 概要

Forms Services を適切に実装するためには、次の点を決定する必要があります。

- Web アプリケーションを配置するネットワークのタイプ
- ネットワークおよびセキュリティに関する問題の管理方法
- ネットワークにアクセスすることが予想されるユーザーの数とタイプ

この章では、Web アプリケーションを配置できるネットワークキング実装のタイプと、各タイプで Web アプリケーションを配置する場合に必要な事項について説明します。

9.2 ネットワーク・トポロジー

アプリケーションを配置できる様々なネットワークキング実装について説明する際には、多くの専門用語を使用します。通常、ネットワークは次の項目にグループ化できます。

- インターネット これは、インターネット・サービス・プロバイダ（ISP）へアクセスするすべての人にオープンなネットワークです。Internet Engineering Task Force (IETF) が起草したデータ転送標準を使用します。
- イントラネット これは、セキュリティー・ルールとネットワーク管理をコントロールする、単一の企業が「所有」しているネットワークです。
- エクストラネット これは、複数の企業によって「所有」されているネットワークで、各企業は独自のネットワーク・インフラストラクチャ、セキュリティ・ルール、およびユーザーを持つため、ネットワーク管理とセキュリティに関する統合的アプローチが必要になります。

インターネット、イントラネット、およびエクストラネットの主な違いは、イントラネットとエクストラネットはコントロールしている単一または複数の企業によって明確に定義され、ユーザーについて把握しているという点です。反対に、インターネットはユーザーについての明確な情報は入手できません。インターネットを介して通信するコンピュータとネッ

トワークは、接続するまで相手のことが分かりません。つまり、暗号化標準、ユーザー認証、認証など前もって調整できません。

これらの実装は次の項で詳細に説明します。

- インターネット
- イントラネット
- エクストラネット

9.2.1 インターネット

インターネットは、インターネット・サービス・プロバイダ（ISP）へアクセスするすべての人にオープンなネットワークです。ユーザーは、インターネットに接続して、世界中の他のネットワーク化されたコンピュータへアクセスします。インターネットに接続しているコンピュータが、ハードウェアやソフトウェアのセキュリティ・メソッドを使用して保護されていない場合、そのコンピュータ上のデータはインターネット上の他のどのユーザーからもアクセスされる可能性があります。

9.2.2 イントラネット

イントラネットは、セキュリティ・ルールとネットワーク管理をコントロールする、単一の企業が「所有」しているネットワークです。ネットワーク化されたコンピュータは、物理的に 1 つの場所（製造工場の目録管理に使用されるコンピュータなど）に置かれるか、または物理的に異なる場所（保険会社の各支店で使用されるコンピュータなど）にあります。

イントラネットは単一の企業によってコントロールされているので、ネットワークにアクセスしようとしているすべてのユーザーについて把握しており、ネットワーク構造、セキュリティ・ルール、およびソフトウェアを自由に選択することもできます。

次に示すのは、イントラネット・スタイルのネットワークの例です。

- ローカル・エリア・ネットワーク（LAN）。
- ダイヤルアップ・アクセスによって使用範囲をリモートの従業員まで拡張する LAN で構成される広域ネットワーク（WAN）。
- 専用通信回線によってインターコネクトされた LAN で構成される WAN。
- 仮想プライベート・ネットワーク（VPN）。これは、通常は公共回線、ときにはインターネット・サービス・プロバイダ（ISP）を介して暗号化された安全な接続を作成する特別な " トンネリング " ソフトウェアを使用して、その使用範囲をリモートの従業員やネットワークに拡張する LAN または WAN で構成されます。

9.2.3 エクストラネット

エクストラネット は、複数の企業が「所有する」ネットワークで、各企業は、独自のネットワーク・インフラストラクチャ、セキュリティ・ルールおよびユーザーを持ちます。ネットワーク化されたコンピュータは、通常物理的に異なる場所に置かれます。多くの場合、異なる企業がネットワーク・データ部分を互いに共有します。たとえば、旅行業界でエクストラネットを使用すると、旅行代理店は航空会社やツアー・オペレータが所有しているネットワークからのデータを利用して、航空機を予約したり旅程を調整できます。

イントラネットと同様、エクストラネットではユーザーについて把握しています。ただし、エクストラネットは複数の企業によってコントロールされているので、ネットワーク管理およびセキュリティに関して統合化アプローチが必要です。旅行業界の例では、旅行代理店と航空会社は、旅行代理店が航空機予約情報にアクセスするためにネットワーキングとセキュリティの問題を調整する必要があります。

次に示すのは、エクストラネット・スタイルのネットワークの例です。

- リモート・ダイアルアップを使用してインターコネクトおよびアクセスする、複数の企業に属する LAN または WAN。
- 専用回線を使用してインターコネクトおよびアクセスする、複数の企業に属する LAN または WAN。
- 仮想プライベート・ネットワーク（VPN）。これは、通常は公共回線、ときには ISP を介して暗号化された安全なネットワーク接続を作成する特別な " トンネリング " ソフトウェアを使用して、使用範囲をリモート・ユーザーまで拡張する、複数の企業に属する LAN または WAN で構成されます。

エクストラネットを介して、ネットワーク化されたデータおよびアプリケーションを共有している企業は、ユーザー認証、認証およびデータ暗号化に関してセキュリティ・プロトコルに同意する必要があります。ファイアウォールやルータなどのセキュリティ・ハードウェアには、互換性が必要です。

9.3 ネットワーク環境における Forms Services の配置

Forms Services がどのように機能するかを検討し、自社にとって最適なネットワーク設定タイプの決定が完了したら、ネットワークに Forms Services をインプリメントできます。次の 5 つの項で、ネットワーキング・オプションと関連するリスクについて説明します。

- インターネットを介した配置
- ローカル・エリア・ネットワーク（Local Area Network: LAN）上での 配置
- リモート・ダイアルアップ・アクセスによるネットワークでの配置
- 公共回線でテレコムが提供する VPN を介したネットワークでの配置
- インターネットでの VPN アクセスを介したネットワークでの配置

9.3.1 インターネットを介した配置

Forms Services を使用すると、HTTP 1.1 パケットに Forms メッセージをカプセル化して、Forms アプリケーションをインターネットに配置できます。HTTP はインターネット上にアプリケーションを配置するために最もよく使用されるプロトコルです。

多くの企業は、HTTP 通信のみを使用可にしてファイアウォールを " ロック・ダウン " することで、プライベート・ネットワークのセキュリティを大幅に向上させます。ファイアウォールを提供している会社の多くは、その製品で HTTP 標準をサポートしており、多くの企業は保有するプライベート・ネットワークの中を HTTP 通信が行きかうことを好意的に認めています。HTTP 通信のみを使用できるサイトでは、構成にほとんど変更を加えず、さらにクライアントに対して完全に透過な状態で、既存のファイアウォールを介して簡単に Forms Services を配置できます。

社内ネットワークを保護するために、厳格なセキュリティ・ルールが必要な場合でも、社内ネットワーク内のファイアウォールの後ろと非武装ゾーン（DMZ）にアプリケーション・サーバーを置くことができます。ファイアウォール内の HTTP フィルタは、VPN を使用しなくても受信トラフィックを制限するのに十分です。

また、より安全な通信を行うために、HTTP 1.1 とともに SSL（secure sockets layer）を使用できます。SSL はプライバシー、整合性および認証を与える転送プロトコルです。SSL は、アプリケーション・レベルの 1 つ下のレベルである、転送レベルで動作します。つまり、SSL は、HTTP などのアプリケーション・レベルのプロトコルで処理される前に、メッセージを暗号化、復号化できます。

インターネット上に Forms Services を配置すると、Web 上の各ユーザーおよびエクストラネットのカスタマは、他のネットワーク配置オプションと比べて低コストでアプリケーションを使用できるようになります。企業は、スケーラブルで、安全で洗練された新規または既存の Forms アプリケーションをインターネット上で実行できます。

9.3.1.1 リスク

HTTP ソケット接続を使用してインターネット上にアプリケーションを配置する場合、ユーザーの Forms Client PC では、等価のパフォーマンスを提供するために、Forms Services の以前のバージョンよりも多少高性能の CPU が必要となります。

HTTP ラッパーで Forms データを送信するとネットワーク・トラフックが増える可能性があり、またより低スピードの接続で同時に実行できるセッション数に影響を与えることがあります。

9.3.1.2 その他のインターネット配置オプション

HTTP ソケット接続メソッドを使用しない場合、保護されたネットワークの外にアプリケーション・サーバーを含む DMZ を設定する他のオプションもあります。IP ルータをセットアップして、DMZ を保護するために、ポート 80（HTTP 通信）と 9001（Forms Listener のデフォルト・ポート）の宛先となるパケット以外のすべての受信パケットをブロックできます。このアプローチを使用する場合、Forms Services Listener ポートが無防備であるという

リスクがあります。複数の Forms Services Listener を使用する場合（たとえば、複数アプリケーションや複数言語をホストする場合）、リスクはさらに高くなります。

さらに、IP ルーターは、DMZ 内に常駐する複数ホームのファイアウォールによって支援する必要があります。このファイアウォールによって、すべての受信トラフィックが IP ルーターから DMZ 内のアプリケーション・サーバーに再経路指定されます。アプリケーション・サーバーは、信頼性のある企業ネットワーク内のデータベースに接続する必要があるため、複数ホームのファイアウォールも、すべての Net8 トラフィックを信頼性のある企業ネットワーク内のデータ・サーバーに再経路指定する必要があります。

ローテーション・スケジュールは、異なる Forms Services Listener を異なる時間に使用して侵入を防ぎたい場所にセットアップできますが、これでは重大なハッカーを防げません。

内部ネットワークをハッカーの進入から守るためには、複数ホームのファイアウォールと内部ネットワーク間に追加のファイアウォールをセットアップして、IP パケットをフィルタし Net8 トラフィックのみ渡すようにすることをお薦めします。

9.3.2 ローカル・エリア・ネットワーク（Local Area Network: LAN）上での配置

Forms アプリケーションにアクセスするすべてのユーザーが LAN 内に存在する場合、基本的な内部ネットワーク・セキュリティは十分であり、Forms Services を特別な構成にする必要はありません。

9.3.3 リモート・ダイアルアップ・アクセスによるネットワークでの配置

一部のユーザーが LAN 外または保護 WAN 外に存在し、ダイアル・インで Forms アプリケーションへアクセスする場合、リモート・アクセス・セキュリティ用に特別に設計されたサーバーが必要になります。このシナリオは、オフサイトで作業している従業員や貴社の LAN や WAN にアクセスする必要がある信頼できるカスタマには理想的です。このソリューションは、LAN にリモートでアクセスする必要があるユーザーが 1000 人より多い場合の実装には適しません。

リモート・アクセス・サーバーに登録されたユーザーが有効なユーザーです。登録されていないユーザーにはアクセス権がありません。リモート・アクセス・サービス（Remote Access Service : RAS）は、Windows NT サーバーの機能の 1 つです。Windows NT RAS サーバーは、このシナリオでリモート・アクセス・サーバーとして使用できます。

プライベート WAN は、専用線で構築されることがよくあります。侵入者は専用線の位置とデータを送信するのに使用する線の電線コードを知らなければ、侵入できません。このような条件下では、侵入されることはまず考えられません。

公共の電話回線を介してダイアルアップする場合は、機密データを送信時に暗号化することをお薦めします。Windows NT RAS サーバーには、Point-to-Point-Tunneling Protocol (PPTP) がインクルードされており、PPTP は、公共の電話回線を介して通信する機密データを暗号化する場合に使用できます。暗号化プロトコルを備えるリモート・アクセス・サー

バーを使用していない場合は、以降の項を参照してください。ネットワークで Forms Services を構成するための、その他のより安全なオプションについて説明されています。

侵入者がリモート・アクセス・サーバーの電話番号にランダムにダイアルし、複数のユーザー名 / パスワードを組み合せて LAN にログインを試みるという場合、リスクはほとんどありません。ただし、リモート・アクセス・サーバーは、サーバーへのアクセス方法をすでに承知している悪意のある元従業員やカスタマに対しては非常に無防備です。

この問題を回避するには、次の予防策をお薦めします。

- 厳格なセキュリティ・レコードのメンテナンス。このメンテナンスにより、元従業員およびカスタマのエントリがリモート・アクセス・サーバー、自動ダイアルバック装置、およびすべての内部システムから削除されているか確認します。
- コール側 ID の検証。これは、登録されている電話番号のみがリモート・アクセス・サーバーにアクセスできるようにする方法です。
- 自動ダイアル・バック装置。この装置は以前に登録した電話番号を使用してコール側にコール・バックします。

9.3.4 公共回線でテレコムが提供する VPN を介したネットワークでの配置

前項で説明したように、従来型の WAN は通常専用線で構築されています。ただし、公共の電話回線を介してダイアルアップしている場合、ユーザー認証およびデータ送信には、より安全な方法を使用されることをお薦めします。

遠距離通信プロバイダから利用できる、VPN（仮想プライベート・ネットワーク）を使用するオプションがあります。遠距離通信プロバイダは、承認されたユーザーのリストを保持しており、認証済のユーザーがダイアル・インする場合はいつでも VPN を作成します。使用しているネットワークに前項で説明したリモート・アクセス・サーバーが必要な場合は、前項のセキュリティの利点とリスクのすべてが適用されます。（このソリューションは、LAN にリモートでアクセスする必要があるユーザーが 1000 人より多い場合の実装には適しません。）

主なリスクは、サーバーへのアクセス方法をすでに知っており、VPN プロバイダの登録済ユーザー・リスト上に存在する、悪意のある元従業員またはカスタマに対して無防備であるということです。このリスクを回避するためには、リモート・アクセス・サーバーおよび VPN プロバイダの登録済ユーザー・リストの両方について、認証済ユーザーのリストを最新のものにしておくことを励行してください。

9.3.5 インターネットでの VPN アクセスを介したネットワークでの配置

ダイアルアップ・アクセスの方法としてインターネットを使用する予定がある場合は、ユーザー認証およびデータ送信について安全なメソッドを使用することをお薦めします。オプションの 1 つは、Forms Services の HTTP ソケット構成または HTTPS（改良済プライバシ、整合性、および認証のための安全なソケット層を持つ HTTP 1.1 ソケット構成）を使用することです。HTTP ソケットの詳細は、3.2 項「ソケット、HTTP または HTTPS」を参照してください。

インターネット上で VPN を使用する別のオプションもあります。このメソッドを使用すると、データは IP（インターネット・プロトコル）パケットのフォームでインターネットを介して転送されます。IP パケットは送信側および受信側の IP アドレスおよびビット（データ）のグループです。

インターネット上に VPN をセットアップすると、遠距離通信費を節約できます。リモート・ユーザーは、専用線や 800 番ではなくローカルの IPS にダイアルします。ネットワークで VPN ソフトウェアを構成およびメンテナンスする必要があり、ダイアル・インするユーザーには互換 VPN ソフトウェアが必要になります。インターネットを介して 2 つの LAN で通信している場所にエクストラネット接続をセットアップする場合は、すべての当事者が互換性のあるファイアウォールを使用する必要があります。リモートの作業者が存在する場合、リモートの作業者が使用できるモバイルのファイアウォールを提供するベンダーもありますが、これには多大な費用と管理時間が必要となります。

ほとんどの大手ファイアウォール・ベンダーは、インターネット上に VPN を実装するためのオプションを用意しています。お薦めできる VPN は、次のものを使用しています。

- 強力なユーザー認証。単純なユーザー・パスワードのメカニズムではなくリクエスト / 応答のメカニズムを含みます。
- ネットワークのより機密度の高い部分へのアクセスをコントロールする内部ファイアウォール。
- 公共ネットワーク間でのデータ転送時にデータを保護するためのデータ暗号化（これは、各 IP パケットのデータを公共ネットワーク間で転送する前に暗号化し、受信先で非暗号化する「IP トンネリング」と呼ばれるものです）。

インターネット上での VPN のセットアップに関連するリスクを次に示します。

- HTTP ソケット接続を使用しない場合、ファイアウォールではデータを通過させません。ファイアウォールと Forms Services を構成すると、汎用プロキシをセットアップすることでこの問題を回避できる場合があります。
- 強力な認証とデータ暗号化に必要な追加の処理のために、ネットワーク・パフォーマンスが低下する場合があります。
- キーは適切に構成および管理してください。
- ファイアウォール構成は厳格に管理して、元従業員や元カスタマを登録解除してください。
- 潜在的なリスクとしてファイアウォール騙しがあります。（ファイアウォール騙しとは、侵入者が IP パケットに偽のアドレスを付与し、ネットワーク上の信頼性のあるノードと偽って侵入し、それらのパケットをユーザーのネットワークに送信することです。侵入者は、ネットワーク上のトラフィックを監視し、自分が付与したアドレスが受け付けれらるか否かを幾ケースもテストし当ネットワークで受け付けれられる IP アドレスを割り出します。）ファイアウォールにフィルタを使用すると、この妨害から守ることができます。

9.4 ネットワーク・セキュリティをメンテナンスするための ガイドライン

Forms Services を使用しているミッション・クリティカルなアプリケーションをインプリメントする場合、セキュリティは重要な問題になります。必要なネットワーク環境のタイプを判断してから、ネットワークを保護するためのセキュリティ方策を明文化します。詳細は、第 10 章「セキュリティに関する考慮事項」を参照してください。

アプリケーション・サーバーを起動して実行した後に、セキュリティを継続的にメンテナンスする必要があります。インターネットを介してアプリケーションにアクセスする場合は、サイトがハッカーによって侵入される場合があるので、メンテナンスは特に重要です。セキュリティ・ルールの強化は継続的なプロセスです。

イントラネット、エクストラネット、およびインターネットの Forms アプリケーションについて、いくつかの配置オプションを説明し、セキュリティに関連する影響を確認してきました。この結果、次のようにまとめることができます。

- ダイアルアップWANまたはダイアルアップVPNを使用しているイントラネットおよびエクストラネットの実装では、適切量の管理作業で合理的に安全性を確保できます。LAN の場合、内部から妨害を受けることが多いので、サーバーの保護とデータベース・ユーザーの管理を改善する必要があります。暗号化メカニズムは、認証されていないユーザーから機密データを保護するためにぜひ使用してください。
- インターネット VPN を介してイントラネットおよびエクストラネットを実装する場合は、強力なアクセス・コントロールのみならず強力な認証と暗号化を使用してください。ほとんどの大手ファイアウォール・ベンダーは VPN オプションを備えており、認証されていないユーザーに対するアクセスのブロック、公共ネットワーク上でのデータの暗号化、およびユーザー認証を行うことができます。

インターネットでのセキュリティ方法の現実的な実装は、次の要素の組合せに基づきます。

- HTTP または HTTPS ソケット通信
- DMZ でのアプリケーション・サーバー
- 内部ネットワークを DMZ から保護するファイアウォール
- 可能な場合のデータ暗号化

10

セキュリティに関する考慮事項

10.1 概要

World Wide Web への関心が急速に高まる前から、インターネット上でユーティリティまたはプログラムを実行して、特定のリモート・コンピュータに接続し友人や仲間を見つけたり、相手側がログオンしているかどうかを確認することはよく行われていました。また、ネットワークを介してリアルタイムにその友人達と通信したり、相手側のディスク・ドライブに一時的に接続してファイルを交換することもできました。

インターネットは実際広範囲にオープンされていて、操作上、信頼レベルは高くても、セキュリティ・レベルは低いものでした。現在は、無数のユーザーが存在するため、セキュリティは重要な問題の 1 つになっています。企業は、ネットワークを保護することで、外部から独自のプライベート・ネットワークへ、コントロールされていないまたは不要なアクセスが行われないようにしています。

この章では、ネットワーク・セキュリティに関する問題を取り扱います。

10.2 共通システム・セキュリティの問題

次の項では、ネットワーク化された環境で Forms Services をセットアップする場合に考慮する必要がある、共通のセキュリティの問題について説明します。

- ユーザー認証
- サーバー認証
- 認証
- 保護送信（暗号化）
- ファイアウォール
- 仮想プライベート・ネットワーク（Virtual Private Network）（VPN）
- 非武装ゾーン（DMZ）

10.2.1 ユーザー認証

認証とは、ネットワークまたはデータベースにログインするユーザーにログイン権限を付与する検証プロセスのことです。認証の例としては、ローカル・エリア・ネットワーク（LAN）へログインする場合のユーザー名とパスワードの使用、およびインターネット上で安全な電子メールを送受信する場合のデジタル証明が含まれます。企業は、目標のセキュリティ・レベルおよび保護するネットワークやデータベースのタイプに応じて、様々なタイプの認証プロセスを使用できます。ただし、最終的な認証の目的は、承認されたユーザーのみがネットワークまたはデータベースおよびそのリソースにアクセスできるようにすることです。

Forms Services の場合、Web 上での Forms アプリケーションの実行は従来のクライアント / サーバー環境と似ており、アプリケーション・ユーザーは、ユーザー名とパスワードを組み合せてユーザー自身を特定することで、データベース・ユーザーとしてログオンします。

Forms Services を使用すると Forms アプリケーションをインターネット上の数百のユーザーに配置できるため、権限のないユーザーがネットワーク上で送信されるデータを（スニッファを使用して）不法に取り込んだり、認証情報を傍受したり、アプリケーションやサーバー環境へアクセスしたりする危険があります。このため、インターネット上にアプリケーションを配置する場合は、暗号化およびファイアウォールなどの追加のセキュリティ機能をインプリメントする必要があります。

10.2.2 サーバー認証

サーバー認証では、クライアント・マシンは、サーバーがリクエスト対象であるか検証します。たとえば、クライアントが機密データをサーバーに送信する場合、クライアントは相手側のサーバーが安全で、送信した機密データの正しい受信者であることを検証できます。

HTTPS 通信モード（SSL（secure sockets layer）を持つ HTTP 1.1 を使用するモード）を使用する場合、インターネット上でのデータの送信は暗号化されサーバー認証が行われます。サーバー認証は、デジタル証明を使用して行われます。クライアントのブラウザがサーバーに接続する場合、サーバーはその証明書を表示します。サーバーは認証局（CA）から証明書を取得します。CA は、個人または企業の識別情報を検証した後でのみ個人または企業に証明書を発行する企業です。

- **JInitiator を使用するクライアント・ブラウザの場合**、Forms Services の HTTPS モードでは、（デフォルトで）次の CA から発行された証明書を信頼します。
  - VeriSign, Inc. - クラス 1、2、3 Public Primary Certification Authority
  - RSA Data Security Inc. - Secure Server Authority
  - GTE CyberTrust Solutions Inc. - CyberTrust Global Root
  - GTE Corporation - CyberTrust Root

  他の CA または他のタイプの証明書を使用する場合、その証明書は Oracle Forms で自動的には信頼されないため、追加の構成ステップが必要となります。JInitiator とともに HTTPS モードを使用する場合は、証明書要求の作成と管理のために、Oracle Wallet

Manager をインストールする必要があります。詳細は 5.7 項「HTTPS 接続モードの設定」を参照してください。

- **ネイティブな Internet Explorer を使用するクライアント・ブラウザの場合**、Internet Explorer によって信頼されたすべての証明書を使用できます。Internet Explorer のデフォルトでは信頼されない CA を使用する場合は、その CA の指示を参照してください。詳細は 5.7 項「HTTPS 接続モードの設定」を参照してください。

10.2.3 認証

認証とは、ユーザーが必要とするネットワークまたはデータベース・リソースに対するアクセス権を認証済ユーザーに付与するプロセスです。また、認証により、ユーザーは不要または使用する権限のないリソースへアクセスできません。たとえば、マネージャは従業員の給与台帳情報が記載してある表へのアクセスは認められても、在庫管理事務員はこの情報へのアクセスは認められません。ネットワークおよびデータベース・リソースでの認証を実行する場合に使用するメソッドは、目標のセキュリティのレベル、および保護されているネットワークまたはデータベースのタイプによって異なります。

Forms Services の場合、ユーザーが認証されるとデータベース・ロールがユーザーに割り当てられ、その結果データベース内のデータを参照または変更する権限が与えられます。（これは認証のフォームの 1 つです。）ユーザー ID もアプリケーション・ロールを設定する場合に使用されます。

10.2.4 保護送信（暗号化）

情報が通信回線を介して送信されると、その通信回線が同軸ケーブル、電話回線、光ファイバー、衛星のいずれであろうとも、通信はサードパーティにより傍受可能であるというリスクがあります。データが漏洩していることを送受信者が気づかれずに、情報が傍受される可能性があります。

最もよく使用される送信保護のためのメソッドは、データを暗号化することです。暗号化を使用すると、データの送受信者は情報をエンコードおよびデコードできる「キー」を持ちます。データを送信すると、送信者側のキーは数学的アルゴリズムを使用して情報をエンコードするために使用されます。受信者側のキーは情報をデコードします。サード・パーティがデータの送信中にエンコードされたデータを傍受しても、サード・パーティがキーへのアクセス権を取得するかまたはアルゴリズムのコードを「破らない」限り、データは判読不能で使用できません。

データを暗号化するのに使用するメソッドは、目標のセキュリティ・レベルとデータを送信するネットワーク・タイプによって異なります。たとえば、対称型暗号化は、ネットワークのスピードが重要である場合に使用できます。ポピュラーな対称型暗号化システムでは、RC-4 およびデータ暗号化規格（DES）を使用します。非対称型暗号化は非常に安全ですが、ネットワーク・パフォーマンス維持に費用がかかります。ポピュラーな非対称型暗号化システムでは、Diffie-Hellman（DH）および Rivest-Shamir-Adleman（RSA）を使用します。

ネットワーク、ファイアウォールまたは VPN にインクルードされている暗号化メソッドを確認する必要があります。Forms Services では、データ送信の安全性を高めるため、次の暗号化オプションが用意されています。

- HTTPS 通信モード : このモードでは、HTTP 1.1 を SSL（secure sockets layer）とともに使用します。HTTPS 通信モードでは、128 ビットまでの暗号化が提供されます。また、デジタル証明を使用したサーバー認証も提供されます。HTTPS モードのセットアップ方法の詳細は、5.7 項「HTTPS 接続モードの設定」を参照してください。
- ORA_ENCRYPT_LOGIN: この環境変数を使用して、Forms Services へログインするためのユーザー名とパスワードを暗号化します。
- DBLINK_ENCRYPT_LOGIN: この環境変数を使用して、データベースへログインするためのユーザー名とパスワードを暗号化します。
- FORMS60_MESSAGE_ENCRYPTION: この環境変数を使用して、RC4 40 ビットの暗号化を使用する Forms メッセージを暗号化します。ソケットおよび HTTP 通信モードにのみ適用されます。（デフォルトで、通信は暗号化されます。）
- FORMS60_HTTPS_NEGOTIATE DOWN: この環境変数を使用して、128 ビットのサーバーに、より低いレベルの暗号化で構成されているクライアントの処理方法を指示します。TRUE を設定すると、サーバーはクライアントで使用可能な最高レベルの暗号化を使用できます。FALSE に設定すると、サーバーは、クライアントが 128 ビットの暗号化を使用しない限り、クライアント・リクエストを拒否します。
- DSA（デジタル署名アルゴリズム）: このアルゴリズムは、Forms Services アプレットでデジタル署名を行う場合に使用します。
- Net8 SNS/ANO: この暗号化方式は、データベースと Forms Services 間の送信を暗号化するために使用します。

10.2.5 ファイアウォール

ファイアウォールとは、通常、ネットワークで受信可能なデータのタイプをフィルタするハードウェアとソフトウェアの組合せです。たとえば、ファイアウォールは保護されたネットワークへ HTTP 通信のみ通れるよう設定できます。また、ファイアウォールは、ネットワークの IP アドレスを無名にしておくことで、外部コンピュータからアクセスできないようにします。ネットワークへのアクセスを認証および権限付与された外部のトラフィックは、ファイアウォールの IP アドレスからネットワークの IP アドレスに再送信されます。ファイアウォールは、プライベート・ネットワークの、侵入に対する防御の最初のラインです。

ネットワーク・セキュリティ・システムにファイアウォールを導入する場合は、標準のソケット接続ではなく、必ず HTTP ソケット接続または HTTPS ソケット接続を使用するように Forms Services リスナーを構成します。これは、ファイアウォールが、標準の Forms メッセージングを含む、パケットまたはポート・レベルでの多くの共通サービスを使用不可にするためです。HTTP はファイアウォールを介して渡すことができるサービスです。

10.2.6 仮想プライベート・ネットワーク（Virtual Private Network）（VPN）

仮想プライベート・ネットワーク（VPN）とは、通信が完全なプライベートとみなされる場所の 2 つのネットワーク間またはネットワークとリモート・ユーザー間の認証済接続です。ネットワークとリモート・ユーザーのコンピュータの両方にある特別な「トンネルリング」ソフトウェアにより、インターネット・サービス・プロバイダ（ISP）を介して、公共回線で保護され、暗号化された接続が作成されます。リモート・ユーザーが VPN ソフトウェアを正しく設定していなかった場合は、ネットワークに VPN を作成できません。

VPN セットアップにファイアウォールが導入されていることがよくあります。標準のソケット接続ではなく、必ず HTTP ソケット接続または HTTPS ソケット接続を使用するように Forms Services リスナーを設定してください。これは、ファイアウォールが、標準の Forms メッセージングを含む、パケットまたはポート・レベルでの多くの共通サービスを使用不可にするためです。

注意： HTTP およびソケットの詳細は、第 3.2 章「ソケット、HTTP または HTTPS」を参照してください。

10.2.7 非武装ゾーン（DMZ）

非武装ゾーン（DMZ）とは、機密情報を含まないネットワーク内の隔離された環境のことです。たとえば、アプリケーション・サーバーを非武装ゾーン内に置き、すべてのデータベース・サーバーは保護されたネットワーク内に置くネットワークを持つことができます。そのようにすると、非武装ゾーンのセキュリティが危険な状態である場合でも、機密データは侵入者に漏洩されません。

10.3 セキュリティ改善のための簡単なステップ

次に示すステップにより、ネットワーク・セキュリティに関するリスクを減らすことができます。

- ユーザーが自分のユーザー名 / パスワードを、承認されていないユーザーに貸与しないようにします。
- 注文受付事務員、役員、製品販売員などの、様々なユーザーのプロファイルに合致する明確なデータベース・ロールによって、厳格な認証方式を実行します。各ロールにより、ユーザー・プロファイルに応じてデータを変更する権限または参照する権限も制限します。
- サーバーまたはデータベースにアクセスする必要のないユーザーを削除するか、パスワード・エイジングを実行して、ユーザー・アカウントを慎重に管理します。
- 暗号化およびデジタル証明認証には、HTTPS 接続モードを使用します。
- ORA_ENCRYPT_LOGIN および DBLINK_ENCRYPT_LOGIN を使用して、送信時にユーザー名とパスワードを暗号化します。

- 侵入者に対する機密データの漏洩を避けることができる場合は常に、FORMS60_MESSAGE_ENCRYPTION および Net8 SNS/ANO などの暗号化を使用します。

次に、確実に実行するようで見落としやすいネットワーク・セキュリティの考慮事項を示します。

- 未認証のユーザーが建物に侵入したりアクセスできないようにするために、サーバー・マシンに対する物理アクセスをコントロールします。
- バックアップ・メディアの保護ストレージなど、厳格なデータ・バックアップ・システムをインプリメントします。
- telnet や ftp などの簡単に侵入できるサービスの使用をやめるか、または最小限に抑えます。
- すべてのセキュリティ関連オペレーティング・システム・パッチをインストールします。

11

パフォーマンス・チューニングに関する考慮事項

11.1 概要

この章では、Forms Services を使用してインターネットまたはその他のネットワーク環境上にアプリケーションを配置する場合に発生する、チューニング上の考慮事項について説明します。この章では、アプリケーション・サーバー上のネットワークおよびリソースについて取り扱います。次の項が含まれています。

- Forms Services のビルトイン最適化機能
- Forms Services アプリケーションのチューニング
- Performance Collection Services
- Trace 収集

Forms Services とデータベース・サーバー間の接続のチューニングについては取り扱いません。

11.2 Forms Services のビルトイン最適化機能

Forms Services および Java クライアントには、いくつかの最適化機能が含まれており、大きく次の項目に分類できます。

- クライアント・リソース要件の最小化
- Forms Services リソース要件の最小化
- ネットワーク使用量の最小化
- ネットワークを介して送信されるパケットの効率の拡大
- クライアントでのアプリケーション画面の効率的なレンダリング

11.2.1 クライアント・リソース要件の最小化

Java クライアントは、主にアプリケーション画面のレンダリングを行います。Java クライアントには、埋込みアプリケーションのロジックはありません。Java クライアントをロードすると、複数のフォームを同時に表示できます。すべての Forms アプリケーションに対応する汎用 Java クライアントを使用すると、アプリケーションごとにカスタマイズされた Java クライアントと比較して、クライアント上のリソースが少なくて済みます。

Java クライアントは、多くの Java クラスで構成されています。これらのクラスは、スプラッシュ画面の表示、ネットワーク通信およびルック・アンド・フィールの変更などの、機能サブコンポーネントにグループ化されます。機能サブコンポーネントを使用すると、Forms Developer および Java 仮想マシン（JVM）は、すべての機能クラスを一度にダウンロードせず、必要に応じて機能をロードできます。

11.2.2 Forms Services リソース要件の最小化

フォーム定義が FMX ファイルからロードされる時、実行プロセスのプロファイルは次のものに要約できます。

- 暗号化されたプログラム単位
- ボイラープレート・オブジェクト / イメージ
- データ・セグメント

これらの中で、データ・セグメント・セクションのみがアプリケーションの指定したインスタンスに対して一意です。暗号化されたプログラム単位およびボイラープレート・オブジェクト / イメージはすべてのアプリケーション・ユーザーに対して共通です。Forms Services は共有コンポーネントを物理メモリーにマップして、同じ FMX ファイルにアクセスするすべてのプロセスでそのコンポーネントを共有します。

指定した FMX ファイルをロードする最初のユーザーは、そのフォームに必要な全メモリー量を使用します。ただし、後続のユーザーの場合は必要なメモリー量が大幅に減らされているので、ローカル・データのエクステントにのみ依存します。共有コンポーネントをマップするこのメソッドを使用すると、指定したアプリケーションに必要な、ユーザーごとの平均メモリー量を減らすことができます。

11.2.3 ネットワーク使用量の最小化

帯域幅は重要なリソースで、インターネット・コンピューティングの一般的な広がりとともに、インフラストラクチャにますます大きな負担を強いるようになっています。このため、アプリケーションはネットワークの容量を節約して使用することが重要です。

Forms Services は、メタデータ・メッセージを使用する Java クライアントと通信します。メタデータ・メッセージは、実行対象のオブジェクトとその実行方法をクライアントに通知する名前と値のペアのコレクションです。パラメータのみを Java クライアント上の汎用オブジェクトに送信することで、(同じ効果になるよう新規コードを送信した場合と比較して) 通信量を約 90% 減らすことができます。

Forms Services では、次の 3 つの方法で効果的にデータ・ストリームを圧縮します。

- 同じようなメッセージの集合（名前と値のペアのコレクション）を送信すると、2 番目以降のメッセージには、前のメッセージとの相違点のみが含まれます。この結果、ネットワーク・トラフィックを大幅に減らすことができます。このプロセスは、*message diff-ing* と呼ばれます。
- 同じ文字列がクライアント画面で繰り返されると（たとえば、同じ企業名が記載されている複数行のデータが表示される場合)、Forms Services はその文字列を一度のみ送信し、後続のメッセージではその文字列を参照します。参照によって文字列を渡すことで、帯域幅の効率は向上します。
- データ・タイプはその値に必要な最小のバイト数で送信されます。

11.2.4 ネットワークを介して送信されるパケットの効率の拡大

待ち時間は、アプリケーションの応答時間に影響を与える最も重要な要因です。待ち時間の影響をなるべく受けないようにする最もよい方法の 1 つは、Java クライアントと Forms Server 間で、対話中に送信されるネットワーク・パケットの数を最小限にすることです。

Forms Developer モデル内のトリガーを多数使用すると大きな効果がありますが、各トリガーにネットワークの往復が必要なため、待ち時間の影響が大きくなります。トリガーに関連する待ち時間を避ける方法の 1 つは、イベント・バンドルを介してトリガーをグループ化することです。たとえば、ユーザーが項目 A から項目 B にナビゲートする場合（あるエントリ・フィールドから別のフィールドへタブする場合など)、トリガー実行前後の範囲には、それぞれ Forms Server 上での処理が必要です。

イベント・バンドルは、2 つのオブジェクト間をナビゲートしている間にトリガーされたすべてのイベントを集めて、それらを単一のパケットとして Forms Services に配布して処理します。ナビゲートに多くのオブジェクトの問合せが関連している場合（離れているオブジェクト上でマウスのクリックを行った場合など)、イベント・バンドルは問い合されたすべてのオブジェクトからすべてのイベントを集めて、そのグループを単一のネットワーク・メッセージとして Forms Services に配布します。

11.2.5 クライアントでのアプリケーション画面の効率的なレンダリング

指定したフォーム内のすべてのボイラープレート・オブジェクトは仮想グラフィック・システム（VGS）ツリーの一部です。VGS は、すべての Forms Developer 製品に共通の図形サブコンポーネントです。VGS ツリー・オブジェクトは、座標、カラー、線幅およびフォントなどの属性を使用して記述します。オブジェクトの VGS ツリーを Java クライアントに送信する場合、送信する属性のみが指定したオブジェクト・タイプのデフォルトと異なる属性になります。

イメージは圧縮された JPEG イメージとして送信および格納されます。これにより、ネットワーク・オーバーヘッドとクライアントの必要なメモリー量の両方を減らすことができます。

リソースの最小化には、クライアントおよびサーバー・プロセスのメモリー・オーバーヘッドの最小化も含まれます。ネットワークを最適な状態で使用するには、帯域幅を最小に維持し、ネットワークの待ち時間の影響も含まれるため、クライアントおよび Forms Services 間の通信で使用するパケット数を最小化することが必要です。

11.3 Forms Services アプリケーションのチューニング

アプリケーションの開発者は、Forms Server のビルトイン・アーキテクチャの最適化機能により最大の利益を得ることができます。この章の後半では、多くのアプリケーションに影響を与える主要なパフォーマンスの問題および開発者がアプリケーションをチューニングしてパフォーマンスを改善し、Forms Server 機能を活用するための方法について説明します。

取り上げる内容は次のとおりです。

- データ・サーバーに対する Forms Services の位置
- アプリケーションの起動時間の最小化
- 必須ネットワーク帯域幅の削減
- パフォーマンスを改善するためのその他の方法

11.3.1 データ・サーバーに対する Forms Services の位置

Java クライアントを Forms Services へ接続する場合、イベント・バンドルなどの機能を使用してネットワーク待ち時間の影響を効率的に抑えることができます。*message diff-ing* を使用して、ネットワーク帯域幅を削減します。一方、Forms Services とデータ・サーバー間に存在するクライアント / サーバーの関係では、往復通信の遅延やネットワークの混雑に関する許容度はさらに小さくなります。

これらの理由から、Forms Services はデータ・サーバーと同じ高速 LAN 上に置くことが最良であり、この結果 Forms Services をユーザーからさらに離れた場所に置くことがあります。これは、ユーザーの近くにサーバーを置くという標準規則に反しているように思えますが、従来のクライアント / サーバー実装と比較して、ネットワーク上での Forms Services の効率を改善した結果です。

最適構成では、図 11-1 に示すように、Forms Services およびデータ・サーバーはデータ・センター内の同じ場所に配置されます。クライアントは低帯域幅（モデム）と長い待ち時間（衛星）の接続を介してサーバーにアクセスしますが、このように設定することをお薦めします。

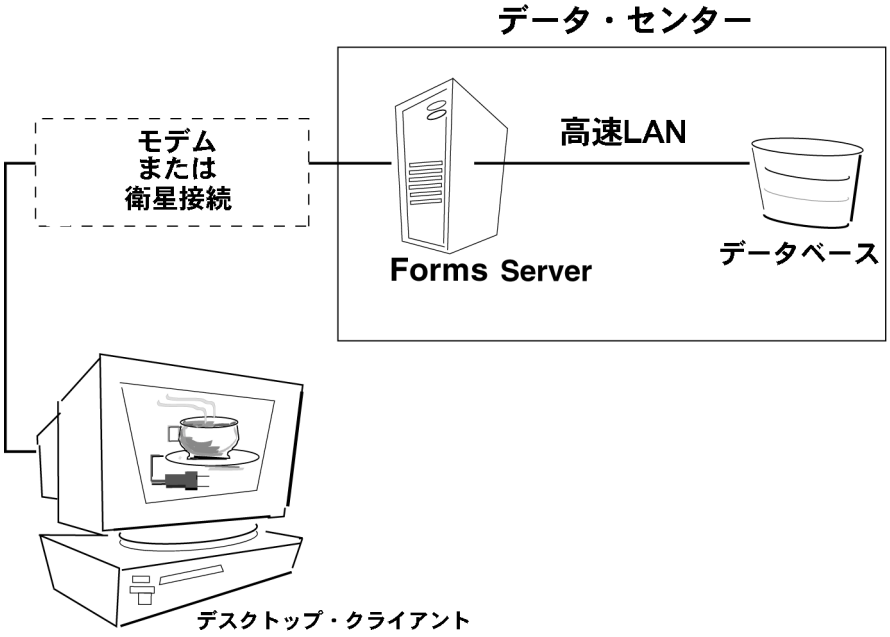


図 11-1 Forms Services およびデータ・サーバーの同じ場所への配置

11.3.2 アプリケーションの起動時間の最小化

アプリケーションをロードするためにかかる時間は、第一印象として重要であり、またどのユーザーにとっても主要な基準となります。起動時間は、オーバーヘッドとみなされます。起動時間は、今後のパフォーマンスを期待させるものでもあります。業務でクライアント・テクノロジを使用する場合、クライアント・コードのロードに必要な追加のオーバーヘッドは、ユーザーにあまりよい影響を与えません。したがって、可能な限りロード時間を最小化することが重要です。

Forms アプリケーションを要求した後に、次のステップを実行してからアプリケーションを使用します。

1. Java 仮想マシン (JVM) を起動します。
2. すべての初期 Java クライアント・クラスをロードし、クラスのセキュリティを認証します。
3. スプラッシュ画面を表示します。
4. フォームを次に示す方法で初期化します。
   a. 必要に応じて、追加の Java クラスをロードします。
   b. クラスのセキュリティを認証します。
   c. ボイラープレート・オブジェクトとイメージをレンダリングします。
   d. 初期画面ですべての要素をレンダリングします。
5. スプラッシュ画面を削除します。
6. フォームを使用できます。

アプリケーション開発者は、JVM を起動するのにかかる時間についてほとんど何も行うことができません。ただし、Java 配置モデルおよび Form Java クライアントの構造では、開発者はロードする Java クラスとその方法を決定できます。これにより、Java クラスに必要なロード時間を最小化します。

Java クライアントには、基本機能のクラス（ウィンドウを開くなど）と特定の表示オブジェクトの追加クラス（LOV 項目など）のコア集合が必要です。これらのクラスはサーバーに始めから常駐している必要がありますが、次の方法を使用すると、これらのクラスをクライアントの JVM にロードするのに必要な時間を短縮できます。

- JAR ファイルの使用
- キャッシュの使用
- 需要に応じた遅延ロード

11.3.2.1 JAR ファイルの使用

Java は Java アーカイブ（JAR）メカニズムを提供して、クライアントにネットワークを介して効率的な配布を行うために、クラスをまとめてグループ化し、圧縮（Zip 形式）することができるファイルを作成します。このファイルをクライアントで使用すると、今後の使用のためにキャッシュされます。

Forms Services では、通常の配置シナリオをサポートするために、次に示す構成済の JAR ファイルが提供されます。

ファイル名	使用方法	説明
f60all_jinit.jar	任意	Oracle JInitiator 専用で、すべてのランタイム状況で使用できる高圧縮された Java クラス・ファイルの集合を含む。
f60all.jar	任意	すべてのランタイム状況で使用できる Java クラス・ファイルの全体的な集合を含む。
f60common.jar	必須	アプレットごとに必要。

f60generic_laf.jar	任意	アプリケーションを Generic lookAndFeel ランタイム設定で配置する場合、または lookAndFeel 設定を指定していない場合にロードする。
		`<APPLET ...>`
		`<PARAM NAME="lookAndFeel" VALUE="Generic">`
		`...`
		`</APPLET>`

f60oracle_laf.jar	任意	アプリケーションを Oracle lookAndFeel ランタイム設定で配置する場合にのみロードする。
		`<APPLET ...>`
		`<PARAM NAME="lookAndFeel" VALUE="Oracle">`
		`...`
		`</APPLET>`

f60splash.jar	必須	アプレットごとに必要。
f60tree.jar	任意	Forms アプリケーションは、階層ツリー・コントロールを使用する場合にのみロードする。

1 つ以上の JAR ファイルをアプレットに指定するには、参照している HTML ファイルの <APPLET> タグで ARCHIVE パラメータを指定します。例:

```
<APPLET CODEBASE="http://www.server.com/webcode/"
ARCHIVE="f60all.jar, icons.jar"
CODE="oracle.forms..">
```

11.3.2.2 キャッシュの使用

Forms Services でサポートされている JVM（Oracle JInitiator および Oracle JDK）は、両方とも JAR ファイルのキャッシュをサポートします。JVM がクラスを参照する場合、最初にローカルのクライアント・キャッシュをチェックして、クラスがキャッシュ済 JAR ファイル内に存在するか確認します。クラスがキャッシュ内に存在する場合、JVM はサーバーをチェックして、JAR ファイルの現行バージョンがあるか確認します。見つからない場合は、クラスはネットワークを介してではなくローカル・キャッシュからロードされます。

キャッシュは、効率性を最大化するために適切なサイズにします。キャッシュ・サイズが小さすぎると、有効な JAR ファイルが上書きされてしまう場合があります。その結果、アプリケーションを再度起動すると、別の JAR ファイルのダウンロードが必要になります。デフォルトのキャッシュ・サイズは 20MB です。このサイズは、アプリケーションを正常に実行した後のキャッシュ容量のサイズと比較する必要があります。

JAR ファイルはロード元のホストに関連してキャッシュされます。これには、異なるサーバーからの同一の JAR ファイルがキャッシュを埋めることができるロード・バランス・アーキテクチャという含意があります。JAR ファイルを中央に置き、ロード・バランス構成の各サーバーでそれらを参照することで、開発者は各 JAR ファイルの 1 つのコピーのみがクライアントのキャッシュでメンテナンスされていることを確認できます。この方法を使用すると、JAR ファイル内の特定のクラスに署名して、ロード元のサーバー以外のサーバーに接続を戻せるようにする必要があります。Oracle が提供する JAR ファイルでは、事前にクラスに署名してあります。

11.3.2.3 需要に応じた遅延ロード

JAR メソッドの欠点は、実行を継続する前に JAR ファイル内のすべてのクラスをロードし、JVM によって妥当性チェックする必要があるという点です。JAR ファイルの便利な点として、他の JAR ファイルを参照して、指定したアーカイブ内に格納するクラスの数を制限できるという機能があります。JVM は、アプリケーションが要求する順序で必要な JAR ファイルにナビゲートできます。

Oracle が提供する f60splash.jar ファイルには、クライアントを初期化し、初期スプラッシュ画面を表示するのに十分なロジックがあります。また、このファイルには、他の JAR ファイルに含まれるファイルの遅延参照もあるので、これらのファイルは需要に応じて続いてロードされます。需要に応じた遅延ロードを使用するには、f60splash.jar ファイルを HTML ページが参照する最初の JAR ファイルに設定する必要があります。

11.3.3 必須ネットワーク帯域幅の削減

開発者は、メッセージ間で異なる情報のみを送信する *message diff-ing* を使用して、データ・ストリーム圧縮を最大限利用できるようアプリケーションを設計できます。次のステップを実行すると、メッセージ間の相違部分を削減できます。

- **メッセージ送信順序のコントロール。**メッセージの送信順序は次の 2 つの基準で管理されます。
  - 初期画面の場合は、オブジェクト・ナビゲータの表示順
  - 実行中、プログラムの順序は項目プロパティに変更

  その結果が有用性に影響を与えない場合は、同じキャンバス上にある似たようなオブジェクトをオブジェクト・ナビゲータ内で並べて配置するようにします。たとえば、ボタンの次にボタン、テキスト項目の次にテキスト項目と配置します。（項目プロパティ「次ナビゲーション項目」を使用する場合、フォーム内の項目でもナビゲーションと同じ順序が使用されます。）オブジェクト・ナビゲータで似たような項目をまとめて順序付けすると、最初のフォームを表示するためにクライアントに送信された項目プロパティには、多くの似たような項目が連続して含まれるので、*message diff-ing* アルゴリズムが効率的に機能します。

  さらに、トリガーまたは他のロジックを使用して項目プロパティを変更する場合、別の表示タイプの項目プロパティを変更する前に似たような項目のプロパティをまとめてグループ化する必要があります。例:

```
set_item_property(text_item1_id, FONT_WEIGHT, FONT_BOLD);
set_item_property(text_item2_id, FONT_WEIGHT, FONT_BOLD);
set_item_property(text_item3_id, FONT_WEIGHT, FONT_BOLD);
set_item_property(button_item1_id, LABEL, 'Exit');
...
```

- **オブジェクト間の類似点を活用。**似たようなオブジェクトを使用すると、（ユーザーに視覚的によりアピールする上に）*message diff-ing* の効率性が向上します。次の手順で、オブジェクト間の一貫性が図られます。
  - プロパティのデフォルト値を受け入れ、オブジェクトに必要な属性のみを変更します。
  - スマート・クラスを使用して、オブジェクトのグループを記述します。
  - ルック・アンド・フィールを少数の可視属性にロックします。
- **ボイラープレート・テキストの使用を削減。**開発者である場合、可能な限り、ボイラープレート・テキストではなく PROMPT 項目プロパティを使用してください。Forms Developer 6.0 以降には、Associate Prompt 機能が含まれており、この機能を使用して、ボイラープレート・テキストを指定した項目のプロンプトとして再設計できます。

- **ボイラープレート項目 (円弧、円、多角形など) の使用を削減。**指定したフォームのすべてのボイラープレート項目をフォームの初期化時にロードします。ボイラープレート項目をロードしてクライアント上でリソースを使用するには、表示の有無にかかわらず時間がかかります。共通のボイラープレート項目（矩形と線）は最適化されます。このため、アプリケーションをこれらの基本的なボイラープレート項目に制限すると、起動時間を短縮しながらネットワーク帯域幅とクライアント・リソースを削減できます。
- **ナビゲーションを最小に維持。**Event Bundle は、2 つまたはそれ以上のオブジェクトにナビゲーショを拡張する場合でも、ナビゲーション・イベントが完了するたびに送信されます。デフォルト値が受け入れられているときに、フィールド間をナビゲートする必要がないフォームを設計します。フォームは、完了したらユーザーが素早く終了できるようにしてください。このようにすると、すべての追加のナビゲーション・イベントは 1 つの Event Bundle として実行されます。
- **初期画面を表示する時間を短縮。**Java クライアントは必須クラスをロードすると、初期画面を表示する前に、表示するすべてのオブジェクトをロードして初期化する必要があります。項目数を最小限に抑えることで、初期画面は移入され、より迅速に表示されます。初期画面を表示する時間を短縮する方法は次のとおりです。
  - オブジェクトの集合（タイトル、小さなロゴ、ユーザー名およびパスワードなど）を制限して、アプリケーションのログイン画面を提供します。
  - Form の初期画面では、要素の非表示はただちに必要としません。次のキャンバス・プロパティを使用します。

```
RAISE ON ENTRY = YES （キャンバスのみ）
VISIBLE = NO
```

  1 つのシートのみ表示されるいくつかのシートで構成される TAB キャンバスには注意してください。タブ間で応答を切り替える場合、キャンバス上のすべてのシートに対するすべての項目がロードされます。この中には初期タブに後ろに隠れている項目も含まれます。この結果、TAB キャンバスをロードして初期化するためにかかる時間は、初めに可視できるオブジェクトのみではなく、キャンバス上のすべてのオブジェクトに関連します。

- **MENU_BUFFERING を使用不可に設定。**デフォルトでは、MENU_BUFFERING は TRUE に設定されます。これは、変更されたメニューが完全な状態で再送信されるときに、メニューに対する変更内容が今後の " 同期化 " イベントのためにバッファされることを意味します。（ほとんどのアプリケーションは同時に複数の変更を行うことも、まったく行わないこともできます。したがって、クライアント側のメニューを更新する最も効率のよい方法は、すべてのメニューを一度に送信することです。）ただし、指定したアプリケーションはメニューに最小限の変更しか加えない場合があります。この場合、変更するたびに変更内容を送信した方が効率的です。これは、次の文を使用して実行できます。

```
Set_Application_Property (MENU_BUFFERING, 'false');
```

メニューのバッファは、LABEL、ICON、VISIBLE および CHECKED のメニュー・プロパティにのみ適用します。ENABLE/DISABLE イベントは常に送信されますが、メニュー全体を再送信する場合は不要です。

11.3.4 パフォーマンスを改善するためのその他の方法

次に示す方法を使用すると、アプリケーションを実行するのに必要なリソースをさらに削減できます。

- **MOUSE-UP、MOUSE-DOWN トリガーの使用を制限。**Java モデルでは、イベントはマウス・ボタンの動作を検出するとトリガーが発行される必要があります。イベントは Forms Services に渡されて、このイベントが MOUSE-UP または MOUSE-DOWN のどちらであるか判別されます。指定したアプリケーションは 1 つのトリガーのみ（MOUSE-DOWN など）定義できますが、イベントを処理するためにトリガー・コードを指定していない場合でも、イベントは、関連（MOUSE-UP）イベントについてクライアントにより生成されます。マウス・イベントは非同期のため、通常のイベント・バンドルモデルの外部で処理されます。
- **タイマーを確認して JavaBeans で置換。**タイマーを起動すると、非同期イベントが生成されます。このイベントとまとめられる他のイベントはキュー内にありません。タイマーのサイズはほんの数バイトですが、毎秒実行されるタイマーは、通常の作業日には毎分 60 のネットワーク・トリップと、およそ 30,000 パケットを生成します。多くのタイマーは、時計または動画を提供するために使用されます。これらのコンポーネントを、Forms Services やネットワークの介入がなくても同じ効果をもたらす、自己完結型の JavaBeans と置換します。
- **入力項目の妥当性チェックのローカライズについて考慮。**When-Validate-Item トリガーを使用して項目に対する入力を処理することはよく行われます。トリガー自体は、Forms Services で処理されます。移動可能な Java コンポーネントを使用して、標準のクライアント項目（テキスト・ボックスなど）のデフォルト機能を置換することを考慮する必要があります。次に、日付や最大 / 最小値などの項目の妥当性チェックを項目内に含めます。この方法を使用すると、より複雑な、入力の自動フォーマットなど（(XXX) XXX-XXXX の書式を持つ電話番号など）のアプリケーション固有の妥当性チェックを行うことができます。
- **アプリケーションを、大きな 1 つのフォームではなく多数の小さいフォームに縮小。**細かく分けられたアプリケーションを指定すると、ユーザーのナビゲーションは、Forms Services からロードおよび初期化されるオブジェクトを定義します。大きなフォームの場合、オブジェクトの初期化中にアプリケーションが遅延して、アプリケーションの多くが参照できなくなるという危険性があります。フォームをまとめて連鎖する場合は、ビルトインの OPEN_FORM および NEW_FORM を使用することを検討します。
  - OPEN_FORM を使用すると、コールしているフォームはクライアントとサーバーにオープンされたままの状態になるので、クライアントとサーバー両方の追加のフォームは多くのメモリーを消費します。ただし、フォームが別のユーザーによっ

て使用中である場合、サーバーのメモリー使用量は、データ・セグメントにのみ制限されます。ユーザーが初期フォームに戻ると、フォームはすでにローカル・メモリー内に常駐し追加のネットワーク・トラフィックを再表示する必要はありません。

- NEW_FORM を使用すると、コールしているフォームはクライアントとサーバー上でクローズされて、すべてのオブジェクト・プロパティが破棄されます。このため、サーバーおよびクライアント上で使用するメモリー量は少なくなります。初期フォームに戻るには、クライアントに再度ダウンロードすることが必要ですが、この場合ネットワーク・リソースが必要で、起動時間が遅延します。初期フォームを再度コールしない限り（ログイン・フォームなど）、OPEN_FORM を使用して、次のフォームをアプリケーションで表示します。

11.4 Performance Collection Services

Performance Collection Services の追加によって、Forms のランタイム診断が強化されました。これによって、お使いのアプリケーションの理解と改善に役立つ情報が提供されます。

11.4.1 Performance Collection Services の使用方法

Performance Collection Services をアクティブにするには、'record=performance' をコマンドラインの引数に含めるか（クライアント / サーバー環境でのランタイムの場合)、HTML ファイル内の 'serverArgs' パラメータの一部として指定します（Web 配置の場合)。

たとえば、クライアント / サーバー・モードで実行する場合は次のようにします。

```
ifrun60  module=.. userid=.. record=performance log=yourlogname
```

結果はファイル *yourlogname* に書き込まれます。ファイル名を指定しない場合は、一意の名前のファイルが作成されます。この名前は 'perf_xxxx' という書式になります。ここで、'xxxx' は実行中のランタイム・プロセスのプロセス ID です。

HTML ファイルでは、次のように指定します。

```
<param name= "serverArgs" value = "module=.. userid=.. record=performance
log=yourlogname">
```

11.4.2 Performance Services によって収集されるイベント

Performance Services によって収集されるイベントには、次のものがあります。

イベント	説明
ClientTime	クライアントで消費された時間
Logon Time	データベース・サーバーへのログオン時刻
Logoff Time	データベース・サーバーからのログオフ時刻
DB Time	データベース操作に要した時間
APServerTime	Forms Services での処理時間

11.4.3 パフォーマンス・データの分析

Performance Services によって収集されたデータは、f60parse.pl という PERL スクリプトを使用して分析します。このスクリプトは、<ORACE_HOME>¥6iserver¥forms60¥perl¥ ディレクトリにあります。

データ分析を開始するには、次のコマンドを入力します。

```
perl f60parse.pl -input=infile -eventf='evfile' -outputf='ofile'
```

各項目の内容は次のとおりです。

- **infile** はアプリケーションの実行中に記録されたデータ。
- **evfile** はイベント記述ファイル。
- **ofile** は PERL スクリプトによって生成される結果ファイル。
- **eventf** および **outputf** はオプションのパラメータ。

デフォルトでは、ブラウザで表示可能な次の HTML ファイルに結果が出力されます。

ファイル名	説明
index.html	ユーザー・アクションの概要
detailed1.html	イベントの詳細
detailed2.html	イベント収集の詳細
event.html	イベント定義

結果ファイルを指定した場合は、指定した名前の XLS 出力ファイルが作成されます。

11.5 Trace 収集

Forms アプリケーションを開発して配置する場合、アプリケーションのどの部分を最適化できるのか、どんなアクションが時間やリソースを要するのかを把握したいときがあります。Oracle Trace を統合することによって、お使いの Forms アプリケーションのパフォーマンスとリソース消費を徹底的に分析できます。

Oracle Trace は Oracle Enterprise Manager（OEM）の一部です。Oracle Trace は、Oracle データベース、Forms Services、およびトレースのために Oracle Trace API をインプリメントするその他の製品のパフォーマンスを収集し、分析するための Oracle ツールです。Oracle Trace を最大限に活用するには、Oracle Enterprise Manager 製品一式に含めて提供されるオプション・パックの 1 つである、Diagnostics Pack をインストールする必要があります。Diagnostics Pack には、Oracle Trace による収集を GUI インタフェースでリモート管理し、Oracle Trace の出力を効率的に表示するための一連のツールが含まれています。

この項では、次のことを説明します。

11.5.1 Oracle Trace でトレースされる Forms イベントのタイプ

Oracle Trace では、事前定義済イベントのデータのみを収集します。これには次の 2 種類があります。

- 期間イベント
- ポイント・イベント

期間イベントには開始点と終了点があります。たとえば、トランザクションは期間イベントです。期間イベントでは、ネストされたイベントが発生することがあります。トランザクション内で発生するエラーなどがこれに該当します。ポイント・イベントは、製品内で瞬間的に発生するイベントです。たとえば、キャンバスまたはダイアログの表示はポイント・イベントです。

Forms Services では、これらのイベントはバイナリ・ファイル（oforms.fdf）で定義されます。このファイルは Forms Services の一部として提供され、次のディレクトリに格納されます。

NT の場合： <ORACLE_HOME>¥6iserver¥net80¥admin¥fdf

UNIX の場合： <ORACLE_HOME>/6iserver/network/admin/fdf

11.5.1.1 Forms 期間イベントおよび項目

次の表では、Forms 期間イベントの番号と名前をリストし、イベントとそれに関連する固有項目および汎用項目について説明します。項目は、特定のイベントに対して表示される情報です。

イベント番号と名前	説明および項目
1. Session	Forms セッション内でエンド・ユーザーによって消費された合計時間（ログインおよびログアウトを含む）。 固有項目 = セッション名、IP アドレス。 汎用項目 = UCPU、SCPU、INPUT_IO、PAGEFAULTS、PAGEFAULTS_IO、MAXRS_SIZE
2. Form	固有のフォームおよびそのフォームからコールされたフォーム内でエンド・ユーザーによって消費された合計時間。 固有項目 = フォーム名。 汎用項目 = UCPU、SCPU、MAXRS_SIZE、CROSS_FAC（セッション ID）。
3. Query	固有の問合せ内でエンド・ユーザーによって消費された合計時間。 固有項目 = ブロック名。 汎用項目 = UCPU、SCPU、MAXRS_SIZE、CROSS_FAC（セッション ID）。
4. Trigger	固有のトリガー内でエンド・ユーザーによって消費された合計時間。 固有項目 = ブロック名、項目名、トリガー ID。 汎用項目 = UCPU、SCPU、MAXRS_SIZE、CROSS_FAC（セッション ID）。
5. LOV	LOV 内でエンド・ユーザーによって消費された合計時間。 固有項目 = LOV 名、ブロック名、項目名。 汎用項目 = UCPU、SCPU、MAXRS_SIZE、CROSS_FAC（セッション ID）。
11. Built-in	ビルトイン内でエンド・ユーザーによって消費された合計時間。 固有項目 = ビルトイン名。 汎用項目 = UCPU、SCPU、MAXRS_SIZE、CROSS_FAC（セッション ID）。

イベント番号と名前	説明および項目
12. User Exit	ユーザー・イグジット内でエンド・ユーザーによって消費された合計時間。 固有項目 = ユーザー・イグジット名。 汎用項目 = UCPU、SCPU、MAXRS_SIZE、CROSS_FAC（セッション ID）。
13. SQL	SQL コード内でエンド・ユーザーによって消費された合計時間。 固有項目 = SQL 文。 汎用項目 = UCPU、SCPU、MAXRS_SIZE、CROSS_FAC（セッション ID）。
14. Menu Create	メニュー作成時にエンド・ユーザーによって消費された合計時間。 固有項目 = メニュー名。 汎用項目 = UCPU、SCPU、MAXRS_SIZE、CROSS_FAC（セッション ID）。
41. ServerTime	Forms Services での処理で消費された時間。 固有項目 = なし。 汎用項目 = UCPU、SCPU、MAXRS_SIZE、CROSS_FAC（セッション ID）。
42. DBTime	データベースで消費された時間。 固有項目 = SQL 文。 汎用項目 = UCPU、SCPU、MAXRS_SIZE、CROSS_FAC（セッション ID）。
43. DBLogon	データベースへのログオンに消費された時間。 固有項目 = なし。 汎用項目 = UCPU、SCPU、MAXRS_SIZE、CROSS_FAC（セッション ID）。
44. DBLogoff	データベースからのログオフに消費された時間。 固有項目 = なし。 汎用項目 = UCPU、SCPU、MAXRS_SIZE、CROSS_FAC（セッション ID）。

注意 : 各ポイント・イベントにはタイムスタンプも付けられます。期間イベントには開始タイムスタンプと終了タイムスタンプが付けられます。

11.5.1.2 Forms ポイント・イベントおよび項目

次の表では、Forms ポイント・イベントの番号と名前をリストし、イベントとそれに関連する固有項目および汎用項目について説明します。

イベント番号と名前	説明および項目
1. Alert	アラートが発生した瞬間。 固有項目 = アラート名。 汎用項目 = UCPU、SCPU、CROSS_FAC（フォーム ID）。
2. Editor	エディタが起動された瞬間。 固有項目 = エディタ名。 汎用項目 = UCPU、SCPU、CROSS_FAC（フォーム ID）。
3. Window	ウィンドウが作成された瞬間。 固有項目 = ウィンドウ名。 汎用項目 = UCPU、SCPU、CROSS_FAC（フォーム ID）。
4. Canvas	ユーザー側から見て、ユーザーがキャンバスを訪れた瞬間。 固有項目 = キャンバス名。 汎用項目 = UCPU、SCPU、CROSS_FAC（フォーム ID）。
5. Timer	タイマーがアクティブになった瞬間。 固有項目 = タイマー名。 汎用項目 = UCPU、SCPU、CROSS_FAC（フォーム ID）。
11. Dialog	ダイアログがアクティブになった瞬間。 固有項目 = ダイアログ名。 汎用項目 = UCPU、SCPU、CROSS_FAC（フォーム ID）。

11.5.2 Diagnostics Pack がない場合の Forms および Oracle Trace の使用方法

Oracle Enterprise Manager および Oracle Diagnostics Pack をインストールしていない場合でも、Oracle Trace を使用して、Forms アプリケーションの側面を知ることができます。Forms Services をインストールする際、コマンドラインから実行するトレース実行ファイルが自動的に <ORACLE_HOME> にインストールされ、Forms アプリケーションのパフォーマンスを手動で分析できます。

Diagnostics Pack なしで Forms および Oracle Trace を使用するには、次のことを実行する必要があります。

- コマンドラインを使用して Trace 収集を開始し、Form アプリケーションをトレース・モードで実行します。

- Trace の出力をフォーマットして、Forms アプリケーションに関する情報を含むテキスト・ファイルを作成します。

11.5.2.1 収集の開始

Oracle Diagnostics Pack をインストールしていない場合は、コマンドライン・インタフェース（CLI）を使用してデータ収集を管理する必要があります。CLI は、Windows ベースのシステムでは MS/DOS ウィンドウから、UNIX ベースのシステムではコンソールから利用できます。

1. otrccol コマンドを使用して収集を開始

データ収集を開始するには、次のコマンドを入力します。

```
otrccol start job_id input_parameter_file
```

各項目の内容は次のとおりです。

- **job_id** は任意の数値（たとえば、1234）。
- **input_parameter_file** は、収集を開始するのに必要な固有のパラメータ値を持つテキスト・ファイル。

  このファイルは次の書式でなければなりません。

  `col_name = my collection`
  col_name は収集に付ける一意の名前。

  `dat_file= <通常、収集名と同じ名前>.dat`
  dat_file は作成するトレース・ファイルの名前。

  `cdf_file= <通常、収集名と同じ名前>. cdf`
  cdf_file は作成するトレース定義ファイルの名前。

  `fdf_file= oforms.fdf`
  fdf-file はトレース対象のフォーム・イベントを含んでいるファイルの名前。このファイルは常に oforms.fdf です。

  `regid= 1 192216243 0 0 45 <データベース SID>`
  regid は、会社のコード（オラクル社は 192216243）、製品のコード（45=Forms）およびその他の内部情報を含む登録識別子。<SID> は必須ですが、Diagnostics Pack がない場合は使用されません。

収集を開始するとプロセスが実行中となり、Forms アプリケーションがトレース・モードで実行されるのを待ちます。

2. Forms をトレース・モードで起動

フォームをトレース・モードで起動するには、パラメータ pecs=trace をコマンドラインに追加する必要があります。

- Windows NT 上のクライアント / サーバー・モードの場合は、次のように指定します。

```
ifrun60 module=myModule userid=scott/tiger@orcl pecs=trace
```

- Forms Services を使用した Web の場合は、HTML ページで次のようにします。

  フォームの実行に使用する HTML ファイルで定義されている **serverArgs** パラメータの一部として、**pecs=trace** を指定します。

- Forms Services と Forms サーブレットを使用した Web の場合は、次のようにします。

  formsweb.cfg ファイル内の固有セクションの一部として、**otherParams=pecs=trace** を指定します。

これでアプリケーションはトレース・モードで実行されるようになります。アプリケーションの実行中に発生するすべてのイベントは、収集の開始に使用したパラメータ・ファイル中に指定した 2 つの出力ファイルに書き込まれます。

3. 出力ファイルの場所

Forms アプリケーションをトレース・モードで実行すると、次の 2 つのファイルが生成されます。

- バイナリ形式のトレース出力を持つ、拡張子が .DAT のファイル。
- トレース定義を持つ、拡張子が .CDF のファイル。

これらのファイルは次のディレクトリにあります。

NT の場合 : <ORACLE_HOME>¥6iserver¥otrace80¥admin¥cdf

UNIX の場合 : <ORACLE_HOME>/6iserver/otrace/admin/cdf

4. otrccol コマンドを使用して収集を停止

データ収集を停止するには、次のコマンドを入力します。

```
otrccol stop job_id input_parameter_file
```

ここで、job_id と input_parameter_file は、収集の開始時に使用したパラメータと同じです。

11.5.2.2 出力のフォーマット

トレース・ファイルが出力されたら、出力をフォーマットして、Forms セッション中に収集した情報を表示できます。Oracle Diagnostics Pack をインストールしていない場合は、Oracle Trace 統計情報レポート・ユーティリティを使用して、トレース・ファイルからテキスト・ファイルを作成できます。

Oracle Trace 統計情報レポート・ユーティリティでは、サーバー・イベントの各オカレンスに関連するすべての項目の統計が表示されます。これらのレポートは膨大な量になることがあります。コマンド・パラメータを使用することにより、レポート出力を制御できます。次のコマンドとオプションのパラメータを使用して、レポートを作成してください。

```
otrcrep [optional parameters] collection_name.cdf
```

Oracle Trace によってイベントごとに 1 つのテキスト・ファイルが作成され、そのイベント内の項目の全体的な集合がインクルードされます。たとえば、トリガーに対して 1 ファイル、ビルトインに対して 1 ファイル、ネットワークに対して 1 ファイル、となります。

11.5.2.3 オプションのレポート・パラメータの使用方法

次のオプションのパラメータを使用することにより、Oracle Trace 統計情報レポート・ユーティリティの出力を操作できます。

パラメータ	説明
output_path	レポート・ファイルの出力先をフル・パスで指定します。指定しない場合、ファイルは現行ディレクトリに配置されます。
-p [<pid>]	イベント・データをプロセスごとに編成します。プロセス ID（pid）を指定すると、そのプロセスによって生成されたすべてのイベントを時間順に含むファイルが 1 つ作成されます。プロセス ID を指定しない場合は、収集に関係したプロセスごとに 1 つずつファイルが作成されます。 出力ファイルの名前は **collection_Ppid.txt** です。
- P	**collection_PROCESS.txt** という名前のレポートを作成します。このレポートには、収集に関係したすべてのプロセスがリストされます。これにはイベント・データは含まれません。このレポートを最初に作成して、より詳細なレポートを作成する固有のプロセスを決定することもできます。
-w#	-w132 のように指定して、レポートの幅を設定します。デフォルトは 80 文字です。
-i#	ページあたりのレポート行数を設定します。デフォルトはページあたり 63 行です。
-h	すべてのイベントおよび項目のレポート・ヘッダーを抑止して、短縮したレポートを作成します。
-s	Net8 データ（または Oracle7 用の SQL*Net）でのみ使用できます。
-a	全製品のすべてのイベントをデータ収集（.dat）ファイル内での出現順に含むレポートを作成します。

11.5.3 Diagnostics Pack がある場合の Forms および Oracle Trace の使用方法

Oracle Diagnostics Pack をインストールしている場合は、Oracle Trace Manager による Oracle Trace 収集のリモート管理ならびに Trace Data Viewer による収集データの表示のために提供されているツールを最大限に活用できます。

11.5.3.1 収集の開始

Oracle Trace Manager を使用するには、まず中間層マシンに Oracle Enterprise Manager をインストールし、さらにデータを収集するノードに Intelligent Agent をインストールする必要があります。

OEM のインストール方法の詳細は、Oracle Enterprise Manager のマニュアルを参照してください。

Oracle Trace Manager を使用すると、次のことができます。

- データ収集を実行するノードをリモートで検出できます。
- ウィザードを使用して、そのノード上で収集を開始できます。ウィザードでは、収集の名前といくつかのオプション・パラメータの入力が求められます。

ノード上で収集を開始すると、Oracle Trace Manager の主画面に収集の実行状況が表示されます。この主画面からリモートで、いつでも収集を停止できます。データ収集の実行中であれば、Forms アプリケーションをトレース・モードで実行できます。

11.5.3.2 出力のフォーマット

Trace Manager を使用した場合の Trace 出力は、コマンドライン・ユーティリティから Oracle Trace を使用した場合の出力と同一です。異なるのは、収集を開始する方法だけです。つまり、アプリケーションをトレース・モードで実行すると、CDF ディレクトリに 2 つのトレース・ファイル（.CDF および .DAT ファイル）が作成されます。

Trace Data Viewer でトレースの内容を表示するためには、トレース出力をデータベース・スキーマに合せてフォーマットしておく必要があります。Oracle Trace Manager を使用してこれを実行するには、データベースにアップロードする収集の上でマウスを右クリックし、フォーマット・オプションを選択します。この結果、Trace Manager 作業環境で指定したスキーマで、トレースがデータベースに自動的にフォーマットされます。

11.5.3.3 Trace Data Viewer の使用方法

ここでは、トレース出力が固有のスキーマでデータベースに格納されているとします。このトレースの内容を表示および分析するには、Trace Data Viewer を起動し、データをアップロードした時のユーザー名でデータベースに接続します。

Trace Data Viewer では、次の分析ができます。

- ネットワーク・トラフィック
- 様々なイベントで消費された時間
- 様々なイベントのための CPU 消費

Trace Data Viewer の主画面から、Forms セッションに関する固有の情報へドリルダウンできます。たとえば、ネットワーク・トラフィック統計には、送受信されたバイト数とパケット数、接続された IP アドレスおよびセッション内のラウンドトリップ数が記録されています。

概要から特定のイベントのドリルダウン分析へと移ることにより、詳細な情報を得ることができます。特定のイベントには、Server、Trigger、Builti-in、Query およびその他のタイプのイベントがあります。それぞれのイベントについて、イベントの消費時間と CPU 消費を詳細表示できます。詳細表示を選択した場合、表示する情報を CPU 消費順または経過時間順でソートできます。そのためには、ソート・キーにする列のヘッダー上でマウスをクリックします。これは、どのトリガーが最も実行時間を要しているか、あるいはどのビルトインが CPU リソースを一番消費しているかを確認する場合などに役立ちます。

最後に、Trace Data Viewer は、各フォームの実行中に発生したすべての期間イベントとポイント・イベントを、すべての期間イベントに対する CPU 時間と経過時間の平均および標準偏差をもって要約します。

11.5.4 Load Balancer Server トレース・ログの設定

この項では、Load Balancer Server トレース・メッセージの形式について説明します。トレースを開始するには、Load Balancer Server を再起動し、forms60_server シェル・スクリプト内の <traceLevel> パラメータを指定する必要があります。<traceLevel> のデフォルトは 0 で、この場合トレースは実行されません。10 を指定すると、Laod Balancer Server 用のトレース出力を作成できます。

11.5.4.1 トレース・レベル 1

トレース・レベル 1 には次のヘッダーが含まれます。

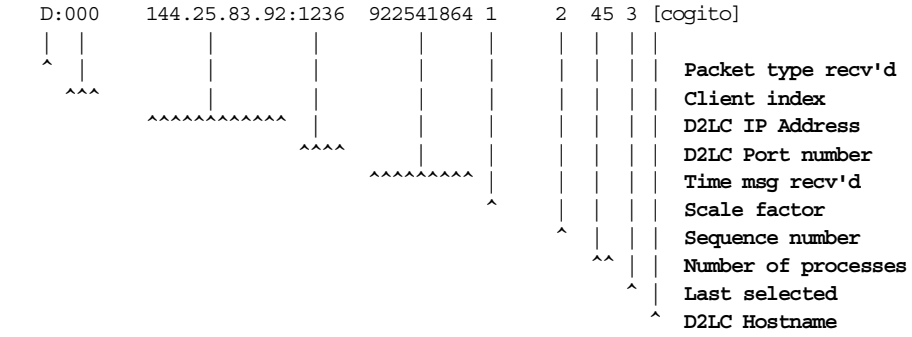
```
HOSTNAME:          neko.us.oracle.com    IP ADDRESS: 144.25.83.146
Data port number:  1234  Request port number:  1235
Maximum number of clients: 10    Trace level: 2
```

- **Hostname** および **IP Address**: D2LS サーバーのホスト名とアドレス。
- **Data port number**: D2LS サーバーが D2LC クライアント・メッセージをリスニングするポート番号。このポートは D2LC クライアント・プロセスを構成するために使用されます。
- **Request port number**: サーバーが最小負荷のホスト情報リクエストをリスニングするポート番号。

- **Maximum number of clients**: D2LC クライアントに割り当てられたスロット数。クライアントごとに 1 つのスロットが必要です。
- **Trace level**: サーバー・ログ・ファイルに印刷されたトレース情報量。

11.5.4.2 トレース・レベル 2

トレース・レベル 2 では、メッセージは D2LC クライアントから次の形式で提供されます。各フィールドの記述は次のとおりです。



トレース・ログの各フィールドの記述は次のとおりです。

- **Packet Type Received**: 受信したパケットのタイプ。次のタイプを指定できます。
  - **D**: D2LC クライアントから受信されたデータ。タイプ "D" のパケットの場合、トレース行の残りのデータはクライアントから送信された情報と一致します。
  - **S**: 最小負荷ホスト用に選択されたクライアント。タイプ "S" のパケットの場合、トレース行の残りのデータは最小負荷ホストとして選択および戻されたクライアントと一致します。
- **Client index**: D2LC クライアントの内部索引。この索引は、リスエストがクライアントから最初に受信されるときに割り当てられます。0 から始まります。
- **D2LC IP Address**: メッセージを送信するクライアントの IP アドレス。
- **D2LC Port number**: メッセージを送信するクライアントが使用する IP ポート番号。
- **Time message received**: メッセージをクライアントから受信した時刻。1970 年 1 月 1 日の 00:00:00 UTC 以降の秒単位の時刻です。
- **Scale factor**: クライアントに割り当てられたスケール・ファクタ。スケール・ファクタは、最小負荷ホストを選択する場合のプロセス数に対する乗数として使用されます。
- **Sequence number**: クライアントが D2LS サーバーにメッセージの送信を試みた回数。
- **Number of processes**: クライアントによって報告されたプロセス数。

- **Last selected**: クライアントが負荷が最も少ないホストとして最後に選択された時刻。サーバー内の内部カウンタは時間経過とともに増加します。クライアントが負荷が最も少ないホストとして選択されると、このカウンタは「**Last Selected**」フィールドに格納されます。最小の「**Last Selected**」フィールドを持つ D2LC クライアントは、最近の使用頻度が最も低いことになります。負荷が最も少ないホストをリクエストするとプロセス数が最も少なくなり、最近の使用頻度が最も低いクライアントが選択されてその結合をブレークします。
- **D2LC Hostname**: D2LC クライアントのホスト名。

11.5.4.3 トレース・ファイルのサンプル

次に、2 つのサーバー構成のサンプルのトレース・ファイルを示します。Formsvr1 は D2L クライアントと D2L サーバーを実行します。Formsvr2 は D2L クライアントを実行します。

```
HOSTNAME:          formsvr1.us.oracle.com       IP ADDRESS:   144.25.87.101
Data port number:  1234 Request port number:  1235
Maximum number of clients: 10   Trace level: 2

D:000   144.25.87.101:1000  925260387 1     2    0 0 [formsvr1]
D:000   144.25.87.101:1000  925260387 1     3   43 0 [formsvr1]
D:001   144.25.87.102:1001  925260388 1     2    0 0 [formsvr2]
D:001   144.25.87.102:1001  925260388 1     3   43 0 [formsvr2]
S:000   144.25.87.101:1000  925260387 1     3   44 1 [formsvr1]
D:000   144.25.87.101:1000  925260387 1     4   45 1 [formsvr1]
D:001   144.25.87.102:1001  925260388 1     4   45 0 [formsvr2]
S:001   144.25.87.102:1001  925260388 1     4   46 2 [formsvr2]
D:000   144.25.87.101:1000  925260387 1     5   45 1 [formsvr1]
D:001   144.25.87.102:1001  925260388 1     5   45 2 [formsvr2]
S:000   144.25.87.101:1000  925260387 1     5   46 3 [formsvr1]
D:000   144.25.87.101:1000  925260387 1     6   47 3 [formsvr1]
D:001   144.25.87.102:1001  925260388 1     6   47 2 [formsvr2]
S:001   144.25.87.102:1001  925260388 1     6   48 4 [formsvr2]
D:000   144.25.87.101:1000  925260387 1     7   47 3 [formsvr1]
D:001   144.25.87.102:1001  925260388 1     7   47 4 [formsvr2]
```

12

ロード・バランスに関する考慮事項

12.1 概要

この章では、Forms Services のロード・バランスに関する考慮事項について説明します。ロード・バランスにより、中間層マシンのプール（サーバー・ファーム）をメンテナンスし、これらのマシン間におけるサーバー・トラフィックの負荷のバランスをとります。ロード・バランスは、任意の Web サーバーで実行できるサーブレットを使用してインプリメントされます。

この章には、次のトピックに関する情報が含まれています。

- ロード・バランスに関する用語
- ロード・バランスのアクション
- Forms Services のロード・バランスの構成

12.2 ロード・バランスに関する用語

ロード・バランスの設定に必要な用語を示します。

- **Load Balancer Server**: 様々なロード・バランス・プールのすべての Forms Services を追跡するコンポーネント。指定したプールでサーバーのステータスを追跡して、その負荷を示す統計を保持します。指定したプールで要求を満たすことができる最小負荷のサーバーに、各フォームの実行リクエストを送ります。
- **Load Balancer Client**: マシンで現在実行している Forms プロセス数などの負荷情報を Load Balancer Server に送信するコンポーネント。Load Balancer Client は、Forms Services とともに各マシン上で実行されます。
- **Servlet**: Forms Servlet は単一の jar ファイルとしてインプリメントされます。
- **プライマリ・ノード**: フォームを実行するすべての URL リクエストを扱う Forms リスナー（および関連ソフトウェア）。ロード・バランスを使用中の場合、各フォームの実行リクエストは、Forms Services を実行している最小負荷マシンに経路指定されます。Load Balancer Server から最小負荷マシンの名前を取得します。
- **セカンダリ・ノード**: Forms Services、Runtime Client および Load Balancer Client を実行しているマシン。フォームの実行リクエストは、ロード・バランスが使用されているときにプライマリ・ノードからセカンダリ・ノードに送信されます。

  多くの場合、プライマリ・ノードはセカンダリ・ノードとしても動作します（たとえば、Forms Services をインストールして実行している場合）。

12.3 ロード・バランスのアクション

ロード・バランスを使用する場合に発生するイベントを図 12-1 に示します。

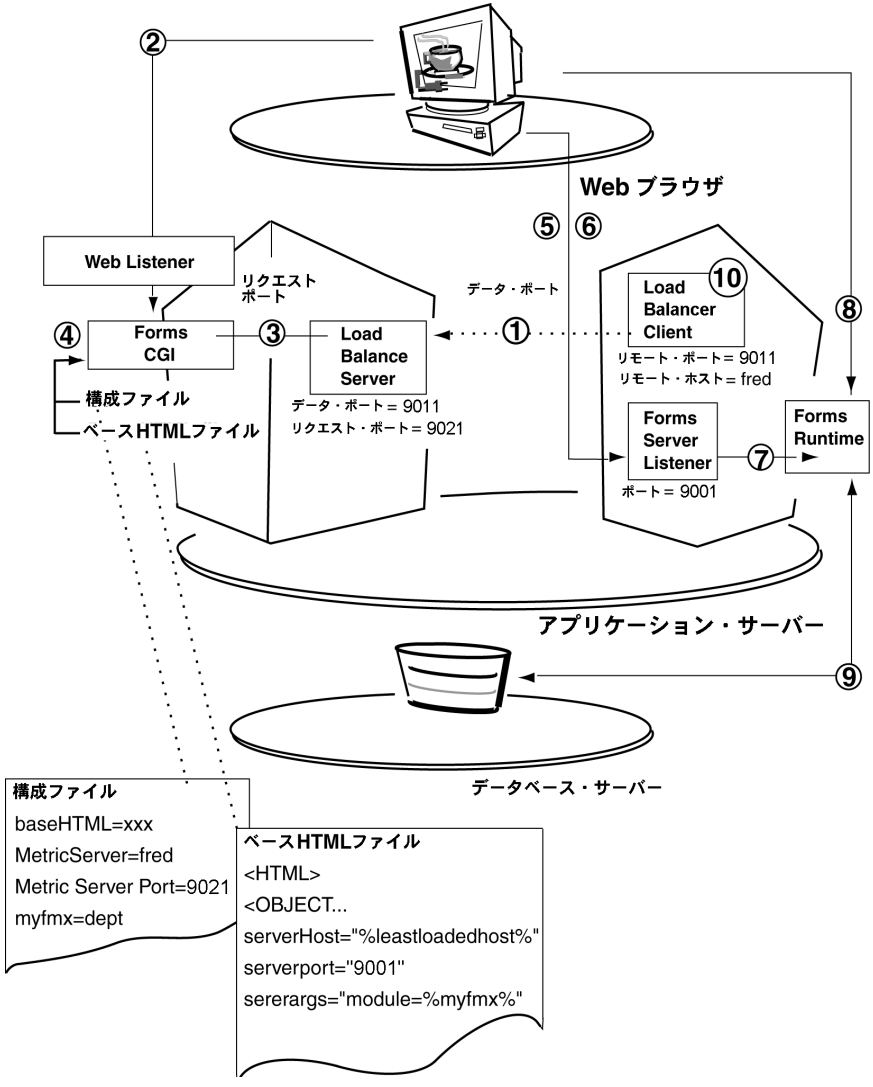


図 12-1 Forms Services のロード・バランス

Forms Services のロード・バランスを使用すると、次のイベントが発生します。

1. Load Balancer Client は、負荷情報を Load Balancer Server へ定期的に送信します。この負荷情報には、各 Load Balancer Client で実行しているプロセスの総数が含まれます。
2. ユーザーは、Forms サーブレットを指す URL にアクセスします。
3. Forms サーブレットは、使用できる最小負荷システムの名前を Load Balancer Server に尋ねます。
4. Forms サーブレットは、Forms Services を実行しているシステムとして指定された最小負荷システムの名前で HTML ページを動的に作成し、その HTML ページをユーザーの Web ブラウザへ戻します。
5. ユーザーの Web ブラウザは、Java アプレットを HTML ページで指定されたホストからダウンロードすることを要求します。
6. Java アプレットは、特定の Form Builder アプリケーション（.FMX）を求めるリクエストを Forms Services に送信します。
7. サーバーは、Forms Services Runtime エンジンと交信します。（サーバーは、アプリケーションの起動遅延を最小化するために、使用できるランタイム・エンジンのプールをメンテナンスします。）各アクティブ・ユーザーは専用のランタイム・エンジンを受信します。
8. サーバーは、ランタイム・エンジンとダイレクト・ソケット、HTTP または HTTPS との接続を確立し、ソケット、HTTP または HTTPS 情報を Java アプレットに送信します。次に Java アプレットで、ランタイム・エンジンとダイレクト・ソケット、HTTP または HTTPS との接続を確立します。Java アプレットとランタイム・エンジンは直接通信し、サーバーを解放して他のユーザーからの起動リクエストを受けられるようになります。（この時点では、アプリケーション・サーバーおよび Forms Services は、アプレットと Runtime エンジン間の通信には関係していません。）Java アプレットはアプリケーションのユーザー・インタフェースをユーザーの Web ブラウザのメイン・ウィンドウに表示します。
9. ランタイム・エンジンは、データ・ソースにより、Net8 または ODBC (Open Database Connectivity) を介してデータベースと直接通信します。
10. Load Balancer Client は、負荷情報を Load Balancer Server へ送信し続けます。すべての新しいサービス・リクエストは、その情報に基づいて経路指定されます。

注意： Load Balancer Server が使用できない場合、ステップ 3 で、Forms サーブレットは最小負荷システムに関する情報を取得できません。そのかわり、Forms サーブレットは、ユーザーのブラウザを MetricsServerErrorURL パラメータで指定した URL にリダイレクトします。リダイレクトはユーザーから見えないため、ユーザーはリダイレクトが行われていることを知る必要はありません。

12.4 Forms Services のロード・バランスの構成

Forms Services とともに提供される次の実行可能ファイルを使用して、ロード・バランスをインプリメントできます。

- Forms Services Listener（f60ctl）
- Load Balancer Server（d2ls60）
- Load Balancer Client（d2lc60）

ロード・バランスを行う各マシンでロード・バランス・コンポーネントをインストールおよび構成する必要があります。これには、プライマリ・ノードを持つマシンおよびセカンダリ・ノードを持つすべてのマシンが含まれます。

また、ロード・バランスを使用するマシンごとに、forms60_server シェル・スクリプトを編集する必要もあります。forms60_server シェル・スクリプトは、<ORACLE_HOME>/6iserver ディレクトリにあります。

次の点を確認します。

- Load Balancer Server の Data Port 値が、すべての Load Balancer Client の Data Port 値と一致しているか。
- ロード・バランス化されるすべての Forms Services には、同じ Forms Services Port 値があるか。

変更するには管理者権限が必要です。また、構成変更を有効にするためには、そのプロセスを停止して再起動する必要があります。

ロード・バランス用に構成するには、forms60_server シェル・スクリプト内の各マシンの次のパラメータを設定する必要があります。

- Forms Services Listener パラメータ
- Load Balancer Server パラメータ
- Load Balancer Client パラメータ

12.4.1 Forms Services Listener パラメータ

<ORACLE_HOME>/6iserver ディレクトリにある forms60_server シェル・スクリプトを編集して、Forms Services Listener が使用するポート番号とプロトコルを設定します。Forms Services Listener を起動するには、次の構文を使用します。

```
f60ctl start port=<Forms Services Port> mode=<Protocol>
```

例 :

```
f60ctl start port=9001 mode=socket
```

Forms Services Port: デフォルトは 9001。Forms Services がフォームの実行リクエストをリスニングする TCP/IP ポート番号を入力します。

注意 : ロード・バランス化されるすべての Forms Services には、同じ Forms Services Port 値が必要です。

Protocol: デフォルトはソケットです。これは、Forms Runtime エンジンと Forms Java アプレット間の通信に使用するプロトコルです。この値は、ファイアウォールを介して通信を行う場合にのみ、HTTP または HTTPS に変更する必要があります。(たとえば、このマシンがファイアウォールの内側にあり、Forms アプリケーションをファイアウォールの外側のユーザーが利用する必要がある場合、HTTP を選択します。HTTP 1.1 を SSL（secure sockets layer）とともに使用する場合は、HTTPS を選択します。)

デフォルトのパラメータ値を受け入れるか、または Forms Services の起動パラメータ値を変更します。このポート番号は、別のプログラムによって使用されている場合のみ変更します。

12.4.2 Load Balancer Server パラメータ

<ORACLE_HOME>/6iserver ディレクトリにある forms60_server シェル・スクリプトを編集して、Forms Load Balancer Server が使用するポート番号を設定します。Load Balancer Server を起動するには、次の構文を使用します。

```
d2ls60 <Data Port> <Request Port> <Maximum Clients> <Trace Level>
```

例 :

```
d2ls60 9011 9021 1000 0
```

Data Port: デフォルトは 9011。Load Balancer Client からの負荷データをリスニングする TCP/IP ポート番号を入力します（セカンダリ・ノードで実行されます)。

Load Balancer Server の Data Port 値は、すべての Load Balancer Client の Data Port 値に一致する必要があります。

Request Port: デフォルトは 9021。Forms サーブレットで作成された " 最小負荷ホスト " のリクエストをリスニングする TCP/IP ポート番号を入力します。この値は、MetricServerPort パラメータとして formsweb.cfg ファイルに書き込まれます。

serverHost パラメータは、値 %LeastLoadedHost% に設定します（serverHost=%LeastLoadedHost%）。ドメイン名が名前の解決のためにネットワークで必須の場合、ドメイン名を serverHost パラメータに追加する必要があります。たとえば、serverHost=%LeastLoadedHost%.jp.oracle.com となります。

Maximum Clients: デフォルトは 1000。負荷情報を実行して Load Balancer Server に送信する Load Balancer Client の最大数を指定します。

Trace Level: デフォルトは 0 で、トレースを実行しません。10 を指定すると、Laod Balancer Server 用の出力を作成できます。

注意： セカンダリ・ノードのみとして使用および構成されているすべてのマシンについて、forms60_server シェル・スクリプトを編集し、Load Balancer Server に関連するセクションを削除する必要があります。

このセクションは次の行から始まり、

```
# Stop load lalancing server
```

次の行で終わります。

```
# Stop load balancing client
```

12.4.3 Load Balancer Client パラメータ

<ORACLE_HOME>/6iserver ディレクトリにある forms60_server シェル・スクリプトを編集して、Forms Load Balancer Client が使用する Load Balancer ホスト名とデータ・ポート番号を設定します。Load Balancer Client を起動するには、次の構文を使用します。

```
d2lc60 <Load Balancer Host> <Data Port> 0 [<Scale Factor> <Process Name>]
```

例 :

```
d2lc60 neko 9011 0 1 f60webm
```

Load Balancer Host: 最初は、ソフトウェアをインストールしたローカル・マシン名がデフォルト値として設定されます。この名前は Load Balancer Server を含むホスト名に変更する必要があります。プライマリ・ノード（Load Balancer Server が実行されているマシン）の完全なホスト名を入力します。値は 256 文字以内で入力します。

Data Port: デフォルトは 9011。ロード・バランス・サーバーが負荷データをリスニングする TCP/IP ポート番号を入力します。各 Load Balancer Client の Data Port 値は、Load Balancer Server の Data Port 値に一致する必要があります。

Scale Factor: デフォルトは、Windows NT の場合は 4、UNIX の場合は 1 です。スケール・ファクタを使用すると、各 Load Balancer Client 上で実行される Load Balancer プロセスの容量の違いに起因する不均衡を軽減できます。最小負荷システムと思われるシステムは、新規プロセスを実行するのに必ずしも最適な場所ではありません。容量が少ないシステムには、さらに大きいスケール・ファクタ値を割り当てる必要があります。

Process Name: デフォルトは f60webm。値を設定すると、Load Balancer Client では、実行可能ファイル名が指定した名前と一致する（ロード・バランスが目的の）プロセス数がカウントされます。値を指定しない場合、マシン上のすべてのプロセスがカウントされます。

13

Oracle Enterprise Manager での Forms のサポート

13.1 概要

この章では、Forms とともに使用する Oracle Enterprise Manager（OEM）のインストールと構成の方法について説明します。OEM の特徴と機能についても説明します。OEM は、グラフィカルな Java コンソール、管理サーバー、エージェントおよび Oracle 製品を管理するための統合されたシステム管理プラットフォームを提供するツールから構成されるシステム管理ツールです。

この章には、次の項が含まれています。

- OEM を使用する理由
- OEM コンポーネント
- Forms とともに使用する OEM コンポーネントのインストールと構成
- OEM コンソールからの Forms Services の管理
- OEM メニュー・オプション

詳細な OEM ドキュメントは、次のマニュアルに記載されています。

- 『Oracle Enterprise Manager 概説』
- 『Oracle Enterprise Manager 管理者ガイド』
- 『Oracle Enterprise Manager 構成ガイド』

13.2 OEM を使用する理由

OEM Forms 管理者インタフェースには、次の基本機能があります。

- **ノードおよびサービスの自動検出** : Forms Listener、Forms Services、Load Balancer Server および Load Balancer Client は、管理対象のノード上で OEM の Intelligent Agent によって自動的に検出され、OEM コンソールのナビゲータ・ツリーに表示されます。
- **ノードおよびサービスの制御** : 検出されたノードとサービスに対して、起動や停止などのいくつかの基本制御が提供されます。
- **ノードおよびサービスの監視** : 検出された Forms Listener、Forms Services、Load Balancer Server および Load Balancer Client の、次のイベントを監視します。サービス停止、過度のメモリ使用、過度の CPU 使用。これらのイベントのいずれかが発生すると、事前にプログラムされたアクションが実行され、システム管理者に警告されるか、問題の自動修正が試行されます。

13.3 OEM コンポーネント

Forms Services を管理するには、次の 3 つの OEM コンポーネントをインストールする必要があります。

- **OEM Management Server（OMS）** : OMS は、OEM の中央リポジトリを制御し、中央リポジトリとして機能するソフトウェアです。OMS は、1 台のマシンにのみインストールします。この OMS マシンが他のマシンを管理します。
- **OEM コンソール** : このソフトウェアは、OMS のユーザー・インタフェースを提供します。
- **OEM エージェント** : このソフトウェアは、Forms Services のデータを収集し、OMS に送信します。OEM エージェントは、OMS によって管理されるすべての Forms Services マシンにインストールする必要があります。

13.4 Forms とともに使用する OEM コンポーネントのインストールと構成

OEM Management Server（OMS）、OEM コンソールおよび OEM エージェント・ソフトウェアは、Oracle9*i* Application Server の一部としてインストールされます。

13.4.1 OEM での Forms サポートの構成

Forms と OMS をインストールした後で、次の操作を実行します。次のステップを実行している間は、OMS サービスを実行しないでください。

1. OMS がインストールされているマシンで、ディレクトリを $ORACLE_HOME¥sysman¥admin に変更します。
2. システム権限を持つログイン（system など）を使用して、OEM リポジトリ・データベースに接続します。
3. "createOEMFormsUser.sql" スクリプトを実行して、OEM リポジトリ内の Forms 固有データをサポートする OEM Forms ユーザーを作成します。（このスクリプトを変更して、デフォルトの表領域や割当て制限などを追加できます。ただし、ユーザー名とパスワードをスクリプトで変更することはできません。）
4. OEM Forms ユーザーとしてデータベースに接続します。(そのユーザー名とパスワードで実行した SQL スクリプトを参照してください。）
5. "createOEMFormsTables.sql" スクリプトを実行して、OEM リポジトリ内に必要な表を作成します。
6. コンソールがインストールされているマシンで、OEM リポジトリ・データベースに接続するために、Tnsnames.ora ファイルに TNS エントリを作成します。OMS マシン上の EM リポジトリへの接続に使用した TNS 別名と同一の名前を使用してください。

13.4.2 OMS サービスの開始

OMS サービスを開始するには、次のように入力します。

```
oemctrl start oms
```

または、Windows NT のコントロール・パネルからサービスを開始します。

13.5 OEM コンソールからの Forms Services の管理

既存の Forms Listener は、OEM から管理できません。最初に、OEM コンソールから Forms Listener を作成する必要があります。

13.5.1 ノードの検索

OEM でリモートの Forms Services マシンを管理するには、そのマシンの場所を探す必要があります。場所を探す手順は、次のとおりです。

1. OEM コンソールで、メニューから「**ノードの検出**」を選択します。
2. ノード名を入力します。たとえば、formssrv-pc と入力します。

13.5.2 OEM コンソールでの管理ユーザーの資格証明の入力

OEM コンソールで管理ユーザーの資格証明を入力する手順は、次のとおりです。

1. OEM コンソールを開始します。
2. 「システム」メニューから、「**作業環境**」を選択します。
3. 「**優先接続情報リスト**」タブを選択します。
4. 「サービス名」列で、管理対象のリモート Forms Services マシンの名前を検索します。サービス・タイプがノードの行を選択してください。
5. このノードで Oracle9*i* Application Server のインストールを実行したユーザーの、オペレーティング・システムのユーザー名とパスワードを入力します。

   注意 : Windows NT では、ユーザーはユーザー・マネージャによって、「サービスとしてログオン」するユーザー権利を付与されていなければなりません。

13.5.3 OEM コンソールからの Forms Runtime インスタンスの表示

OEM コンソールから Forms Runtime インスタンスを表示する手順は、次のとおりです。

1. OEM コンソールで、「**Developer Server**」、「**Forms_Listeners_**<*RemoteMachineName*>」を選択します。
2. マウスの右ボタンをクリックして、「**ランタイム・プロセスのリスト**」を選択します。

13.6 OEM メニュー・オプション

次のメニュー・オプションは、Forms Listener、Forms Services、Load Balancer Servere および Load Balancer Client の管理に使用できます。

13.6.1 Forms Listener グループの制御

マウスの右ボタンで表示されるメニューで使用可能なコマンドは、次のとおりです。

- **新規作成**: 新規リスナー・プロセスを作成する前に、パラメータのリストを求めるプロンプトが表示されます。リスナー・プロセスが作成されると、エントリがナビゲータ・ツリーに表示され、リスナーが開始します。
- **ランタイム・プロセスのリスト**: このコマンドは、このノードで実行されている Forms Runtime プロセスのリストを別のウィンドウに表示します。「ランタイム・プロセス・リスト」ウィンドウを参照してください。
- **リフレッシュ**: このコマンドは、このノードで実行されている既存の Forms Listener を検出し、このノード上のすべての Forms Listener インスタンスの実行中 / 非実行中ステータスもリフレッシュします。

13.6.2 Forms Listener インスタンスの制御

マウスの右ボタンで表示されるメニューで使用可能なコマンドは、次のとおりです。

- **開始**: リスナーが現在ダウンしている場合に、リスナーを開始します。
- **停止**: リスナーがシャットダウンし、Listener インスタンスがダウンしていることが特殊なアイコンによって示されます。
- **類似作成**: コピー・コマンドと非常によく似ていて、現在のリスナーと同じパラメータを使用して別のリスナーを作成します。「新規作成」コマンドと同様のダイアログによって、必要な変更を行うように求められます。
- **変更**: ダイアログ・ボックスで、起動パラメータと環境変数を変更できます。
- **削除**: Listener インスタンスがナビゲータ・ツリーから削除されます。Forms Listener インスタンスは、そのリスナーに関連付けられている Runtime プロセスがない場合のみ削除できます。削除されたリスナーは、ノードから自動的にシャットダウンします。
- **プロパティ**: この Forms Listener インスタンスに関連付けられているパラメータ、環境変数および Runtime プロセスのリストを表示します。

13.6.3「ランタイム・プロセス・リスト」ウィンドウ

このウィンドウは、ある特定のノード上の現在の Forms Runtime プロセスをすべて示す表タイプのリストです。各行が 1 つの Runtime プロセスを表します。次のフィールドが表示されます。

- リスナー名
- ノード
- IP アドレス
- ユーザー名
- プロセス ID
- 接続時間
- 動的ロギング・ステータス
- メモリー使用量
- CPU %

13.6.4 Forms Runtime プロセスの制御

マウスの右ボタンで表示されるメニューで使用可能なコマンドは、次のとおりです。

- **削除** : 実行を停止するための削除シグナルが Runtime インスタンスに送信されます。このコマンドは主に、不正なランタイム・プロセスを停止して、それ以上損害が与えられないようにするために使用します。
- **ロギング使用可能** : ランタイム・インスタンスの動的ロギングをオンにします。ログは、生成されたファイル名でテンポラリ・ファイルに書き込まれます。ファイル形式は、Forms Runtime Diagnostic（FRD）で生成される形式と同じです。
- **ロギング使用不可** : ランタイム・インスタンスの動的ロギングをオフにします。
- **ログの表示** : 動的ロギング・コマンドで生成されたログ・ファイルを表示します。

13.6.5 Load Balancer Server グループの制御

マウスの右ボタンで表示されるメニューで使用可能なコマンドは、次のとおりです。

- **新規作成** : Load Balancer Server インスタンスが作成されます。サポートされるパラメータは次のとおりです。*<port #1> <port #2> <max. no. of client> <trace level>*。

Load Balancer Server は、Metrics Server とも呼ばれています。

13.6.6 Load Balance Server インスタンスの制御

マウスの右ボタンで表示されるメニューで使用可能なコマンドは、次のとおりです。

- **開始** : Load Balancer Server を開始します。
- **停止** : Load Balancer Server をシャットダウンします。
- **類似作成** : コピー・コマンドと非常によく似ていて、現在と同じパラメータを使用して別の Load Balancer Server を作成します。
- **変更** : 起動パラメータと環境変数を変更するためのダイアログ・ボックスが表示されます。
- **削除** : Load Balancer Server がナビゲータ・ツリーから削除されます。削除されたサーバーは、ノードから自動的にシャットダウンします。
- **プロパティ** : この Load Balancer Server に関する関連情報を表示する別のウィンドウを起動します。

Load Balancer Server は、Metrics Server とも呼ばれています。

13.6.7 Load Balancer Client グループの制御

コマンドは、Load Balancer Server のオブジェクト・タイプとまったく同じです。サポートされるパラメータは、次のとおりです。

<Master Server host name> <Remote port> <Local port> <Scale Factor>

Load Balancer Client は、Metrics Client とも呼ばれています。

13.6.8 Load Balancer Client インスタンスの制御

コマンドは、Load Balancer Server のオブジェクト・インスタンスとまったく同じです。

Load Balancer Client は、Metrics Client とも呼ばれています。

13.6.9 監視機能

イベントは、OEM コンソールのイベント管理画面にリストされます。これらのイベントは、OEM で登録または登録解除することによって、オンまたはオフにできます。イベントが作成されて OEM に登録されると、OEM はイベントが発生したときに、システム管理者に通知するか、回復処理を実行できます。

次のイベントを登録できます。

- **リスナー停止 :** このイベントは、リスナー回復処理ありまたはなしでスケジュールできます。リスナー回復処理は、このイベントが発生したときに Listener を再起動するために使用できます。Listener がダウンした場合は、イベント・ログにエントリが書き込まれます。このエントリは、OEM コンソールから表示できます。

注意 : 回復処理は、回復処理を使用するイベントをスケジュールする前にスケジュールする必要があります。

- **Load Balancer Server 停止 :** リスナー停止と同様です。このイベントは、Load Balancer Server 回復処理ありまたはなしでスケジュールできます。
- **Load Balancer Client 停止 :** Load Balancer Server 停止と同様です。このイベントは、Load Balancer Client 回復処理ありまたはなしでスケジュールできます。
- **ランタイム・プロセスによる過度の CPU 使用 :** ランタイム・プロセスが CPU 時間を消費しすぎると、システム管理者に通知されます。このイベントは、*X* 秒ごとにチェックされます。時間間隔を設定します。アラート基準値、警告基準値および発生回数を選択できます。
- **ランタイム・プロセスによる過度の仮想メモリー使用 :** ランタイム・プロセスによって消費されている仮想メモリーが一定量を超えると、システム管理者に通知されます。このイベントは、過度の CPU 使用イベントと同様です。次のパラメータを設定できます。イベント間隔、警告基準値（KB 単位の仮想メモリー）、アラート基準値（KB 単位の仮想メモリー）および発生回数の各パラメータ。

14

キャパシティ量計画の考慮事項

14.1 概要

この章では、Forms Services の拡張性機能について考察します。広く使用されているハードウェア・プラットフォームとオペレーティング・システムを用いて、いくつかのベンチマーク・テストを実行することによって、サーバーの拡張性を調査しました。

次のベンチマークを測定しました。

- ユーザーあたりの RAM
- CPU あたりのユーザー数

Forms Server 6.0 では、次の結果が得られました。

Windows NT の場合 :

表 14-1 Windows NT でのベンチマーク

アプリケーションのサイズ / 複雑さ	ユーザーあたりの RAM (MB)	CPU あたりのユーザー数
標準 / 普通	2.5 ～ 6.0	100 ～ 300
小 / 単純	1.0 ～ 2.5	150 ～ 300

Sun Solaris の場合 :

表 14-2 Sun Solaris でのベンチマーク

アプリケーションのサイズ / 複雑さ	ユーザーあたりの RAM (MB)	CPU あたりのユーザー数
標準 / 普通	2.0 ～ 5.0	200 ～ 400
小 / 単純	1.0 ～ 2.0	300 ～ 500

注意： この章に記載されている結果は、Forms Services のリリース 6*i* に固有であり、以前のリリースの製品には適用されません。このリリースは、以前のリリースと比較してパフォーマンスが改善されています。パフォーマンスの改善は、次のようないくつかのアーキテクチャおよびコードの最適化によるものです。

- Windows NT のもとでの動的リンク・ライブラリ共有の改善
- 中間層レコードのキャッシングの改善
- メッセージ・レイヤーの改善によるサーバー上の全体的な処理の削減

ベンチマーク・テストは、オラクル社で現在でも進行中の作業です。ここに示す図は、本書の記述時点で利用可能な情報を表しています。追加の結果は、利用可能になりしだい公開されます。

14.2 拡張性とは

拡張性とは、基盤となるソフトウェアは変更しないでシステムにハードウェア・リソースを追加することで、ある単一システム上のユーザー数の増加に適応できる能力です。拡張可能なシステムは、企業のニーズの拡大に適応できます。

パフォーマンス・ニーズに合せて拡張できるハードウェアとソフトウェアを選択することは、パフォーマンス・ニーズが変わるたびに新しいソフトウェアを購入することよりもはるかに優れた方針です。

次の事項を検討してください。

- アプリケーションまたはオペレーティング・システムは、どのようにすれば追加システム・リソースを有効に利用できるか。
- *n* 人のユーザーをサポートするためには、メモリーがどの程度必要か。
- より高速なプロセッサまたは複数のプロセッサに簡単にアップグレードできるか。
- 追加のプロセッサによって処理能力はどの程度増加するか。
- パフォーマンスを向上させるために後で追加できる機能はあるか（キャッシュやドライブ・アレイ・コントローラなど）。

これらの質問に対する回答は、使用するハードウェア、オペレーティング・システムおよびアプリケーション・ソフトウェアに大きく依存します。

14.3 システム・キャパシティの評価基準

ネットワーク化されたアプリケーションの拡張性は、アプリケーション・サーバーの能力と、ユーザー負荷の増加に予想どおりに適応するためのネットワーク・トポロジに関連します。

この項で説明する各コンポーネントの役割と、それらのコンポーネントが特に Forms Services 環境で、システムの全体的な拡張性にどのように影響するかを理解しておくと役に立ちます。

この章では、最もよく使用されるサーバー・ハードウェアとオペレーティング・システムの 2 つの組合せを例として使用します。その組合せとは、Sun UltraSparc アーキテクチャ上で実行される Sun Solaris と、Intel アーキテクチャ上で実行される Microsoft Windows NT です。

次の領域は、Forms Services をベースとするシステムの評価で重要です。

- プロセッサ
- メモリー
- ネットワーク
- 共有リソース
- ユーザー負荷
- アプリケーションの複雑さ

14.3.1 プロセッサ

より高速な動作か、より効率的なな動作か。プロセッサ・テクノロジは、両方のアプローチを探索しました。通常、企業は 2 ～ 3 年ごとに新世代アーキテクチャ（より効率的な動作）をリリースします。これらのリリースの間に、プロセッサ速度が向上します（より高速な動作）。クロック速度とも呼ばれるプロセッサの速度は、通常はメガヘルツ（MHz）で表されます。プロセッサ速度は、コンピュータ・システムがどの程度高速に稼動できるかをよく示します。通常は、サーバーとして使用されるコンピュータは、複数のプロセッサを使用し、マルチプロセッサ・システムと呼ばれます。

Forms Services に関して我々が実際に興味を持つ基準値は、各プロセッサ上の同時ユーザー数です。この基準値は、プロセッサあたりのユーザー数と呼ばれることもあります。この数値は、プロセッサのタイプによって大きく異なります。この変化の例は、14-1 ページの表 14-1 と 14-1 ページの表 14-2 を参照してください。

ベンチマークで収集された経験データによると、400MHz の Intel Pentium II Xeon プロセッサと 1MB の L2 キャッシュを搭載したコンピュータは、200MHz Pentium Pro システムと比較して、約 2 倍のユーザー数をサポートできます。

14.3.2 メモリー

メモリーは、コンピュータ・システムがプログラムの起動と実行に使用できる RAM の容量です。コンピュータ・システムの RAM の容量は、通常はメガバイト（MB）で表されます。

プログラムの通常の実行では、プログラムは RAM にロードされ、プログラムが非アクティブになるたびに、オペレーティング・システムがプログラムをディスクにスワップします。オペレーティング・システムは、プログラムがアクティブになると、そのプログラムを RAM に戻します。

このアクティビティは、一般にスワッピングと呼ばれています。Sun Solaris や Microsoft Windows NT などのほとんどのオペレーティング・システムは、通常の操作中にスワッピングを実行します。スワッピングによって、プロセッサの需要が増加します。過度のスワッピングは、システムの処理速度をかなり低下させる傾向があります。パフォーマンスの低下を防ぐには、サーバー・ホスト・マシンに十分な RAM を搭載してください。

重要な基準値は、Forms Services を介してアプリケーションに接続し実行するすべての追加ユーザーが必要とする RAM です。この基準値は、ユーザーあたりのメモリーとも呼ばれます。通常は、パフォーマンス測定ツールは、ユーザーあたりのメモリーを正確に測定しません。この基準値を入念に調査して、メモリー要件を判断します。ユーザーあたりのメモリーの例は、14-1 ページの表 14-1 と表 14-2 を参照してください。

14.3.3 ネットワーク

Forms Services のような多層のインターネットベース・アーキテクチャでは、クライアントを Forms Services に接続する物理的なネットワーク、および Forms Services とデータベース間の接続は、システムの全体的な拡張性の重要な要因となります。Forms Services をベースとするシステムのパフォーマンスを測定するときは、物理ネットワークのパフォーマンスに注意してください。

14.3.4 共有リソース

マルチユーザー、マルチプロセス環境での個々のプロセスのパフォーマンスは、メイン・メモリーで処理される個々のプロセスの能力に直接比例します。つまり、他のプロセス用の領域を空けるために、必要なページが仮想メモリーにスワップされると、パフォーマンスが悪影響を受けます。必要なページがメイン・メモリーに見つかる可能性を高める 1 つの技法は、イメージ・マップ・メモリーを使用して共有メモリー・モデルをインプリメントすることです。イメージ・マップ・メモリーは、メモリー内のファイルの内容を、プロセス間で共有される特定のアドレス空間に関連付けます。

Forms Services はイメージ・マップ・メモリーを使用します。個々の Forms プロセスは、FMX ファイル・イメージの大部分を共有するので、個々のメモリー要件が低減し、全体的な拡張性が向上します。

14.3.5 ユーザー負荷

ベンチマーク・シナリオでは、実際のアプリケーション環境を正確に作り出すために多数のクライアント・マシン（およびユーザー）を設定するのは、実際的ではありません。ベンチマークでは、負荷シミュレータを使用して、アプリケーション・サーバーでトランザクションを実行する実際のユーザーをシミュレートします。Oracle Tools の開発部門は、負荷シミュレータを開発しました。このシミュレータは、サーバーにメッセージを送信して負荷をシミュレートすることで、実世界の Forms Services ユーザーを模倣します。負荷シミュレータは、Forms Services と UI クライアントの間に位置し、これら 2 つのコンポーネント間のメッセージ・トラフィックをインターセプトする小さな Java アプリケーションです。

クライアントからのイベント・メッセージが記録されると、そのメッセージをサーバーに再生できます。これにより、実際のユーザー・セッションがシミュレートされます。（UI クライアントは、再生モードには関係しないことに注意してください。）サーバーへの再生中に、負荷シミュレータは多数のユーザー・セッションを再生できます。この方法では、負荷シミュレータは、クライアントとサーバー間のメッセージの往復時間の合計を判断することによって、ユーザーへの合計応答時間を計算できます。あるビジネス・トランザクション全体の合計応答時間を累計することによって、アプリケーション・パフォーマンスの測定可能な基準値を取得できます。

14.3.6 アプリケーションの複雑さ

値リスト（LOV）とポップアップ・ウィンドウを含む単純な単一 Form から、複数の Forms と PL/SQL ライブラリ（PLL）を同時にオープンする複雑なアプリケーションにいたるまで、様々な複雑さの Forms アプリケーションをテストしました。アプリケーションの複雑さは、ある 1 つのモジュールに固有の複雑さではなく、ユーザーが一度にアクセスできるモジュール数に関連付けました。

複雑さを判断するのによい方法は、Form に追加されたすべての依存性を参照することです。たとえば、フォームは、CALL_FORM または OPEN_FORM ビルトインを介して他のフォームをコールすることがあります。また、メニュー（MMX ファイル）に接続することや、PL/SQL ライブラリ（PLL ファイル）を使用して外部ビジネス・ロジックをロードすることもあります。これらすべての要因は、ユーザーあたりのメモリー使用量に寄与します。

次の表に、Oracle Forms アプリケーションの複雑さのレベルを分類します。

表 14-3 アプリケーションの複雑さの判断

アプリケーションのサイズ / 複雑さ	メモリー内の同時モジュールの合計サイズ
大 / 複雑	> 10MB
標準 / 普通	2 ～ 10MB
小 / 単純	< 2MB

複雑さの異なる 2 つのアプリケーションをテストしました。

- 最初のアプリケーションは、適切なメニューと値リストを含む単純な「顧客注文入力」画面でした。どの時点でも、アクティブなフォームは 1 つのみでした。
- 2 番目のアプリケーションは、複雑さが普通のアプリケーションでした。実際の顧客アプリケーション、ヘルプ・デスクおよび顧客サポート・システムを使用しました。このアプリケーションには、同時にオープンされる多数のモジュールがあり、個々のモジュール内には複雑なビジネス・ロジックがありました。

現実的なユーザー・コミュニティ、つまり複合的な作業負荷が存在するコミュニティを表すために、テストでは、45 分間のシナリオに、サービス・デスク要員が実行するアクティビティを模倣したいくつかのトランザクションを含めました。

インプリメントした作業のステップごとの定義。

ステップ	実行した作業
1	サービス表示アプリケーションの起動 - ログイン
2	通知画面へ [ナビゲート]
3	進捗画面を [コール] トランザクション : パラメータ化した問合せの入力
4	問題画面を [オープン] 各種タブ（PL/SQL の実行）へ [ナビゲート] 画面上のすべてのフィールドへ [ナビゲート]
5	サービス画面を [コール] トランザクション : ブラインド問合せの入力 問い合せたすべてのフィールドへ [ナビゲート]
6	シナリオ 2 ～ 5 の [繰返し]

14.4 拡張性の基準値の判断

ユーザー負荷が増えたときに感じられるパフォーマンスの低下感を測定するためには、最初に、特定のユーザーが特定のアプリケーション作業を実行するのにかかる時間を判断する必要があります。この合計応答時間基準値は、単に特定の物理トランザクションやネットワークの往復の応答時間をテストするのとは異なります。このメトリックは、(平均的なユーザーが）当面のビジネス・タスクを実行するのにかかる合計時間（つまり、ビジネス・トランザクションの一部として Forms Services とデータベースの間で行われるすべての対話の合計）を参照します。

全体的なシステム・リソースに関する経験的な情報を得るために、拡張性テストでは、オペレーティング・システムに固有の監視ユーティリティ（Windows NT のパフォーマンス・モニタなど）も使用して、物理メモリーと仮想メモリーの使用量および CPU の合計使用量の値を判断します。

合計応答時間基準値を経験的な測定値とともに使用することで、ユーザー負荷が増えたときに、特定のユーザーのパフォーマンスが大幅に低下するポイントを判断できました。許容されるパフォーマンスでサポートできるユーザー数を判断したところ、個々のメモリー消費は、アプリケーションにアクセスするユーザー数で除算した合計使用可能メモリーの単純な等式になりました。

例 :

512MB の RAM のある特定のハードウェア・プラットフォームでは、60 人の同時ユーザーまではパフォーマンスが一定です。それを超えると、パフォーマンスは大幅に低下します。これにより、サポートされる最大ユーザー数は 60 人であると規定できます。

通常のオペレーティング・システム・オーバーヘッド（～ 32MB）を考慮すると、個々のメモリー使用量は、(512-32) / 60、つまりユーザーあたり 8MB になります。

14.5 サンプルのベンチマーク結果

次の項では、次のシナリオについて、テストしたシステム、テストの結果および簡単な分析を定義します。

- 低コストの Intel Pentium ベース・システム上の、標準的な複雑さのアプリケーション
- Intel Pentium II Xeon-Base システム上の、標準的な複雑さのアプリケーション
- エントリレベルの Sun UltraSparc サーバー上の、標準的な複雑さの アプリケーション
- Intel Pentium II Xeon-Base システム上の単純なアプリケーション
- エントリレベルの Sun UltraSparc サーバー上の単純なアプリケー ション

14.5.1 低コストの Intel Pentium ベース・システム上の、標準的な複雑さのアプリケーション

パラメータ:

アプリケーションのサイズ / 複雑さ	CPU	RAM	オペレーティング・システム	スワップ
標準（2 ～ 10MB）	2 つの 200MHz Pentium Pro	512MB	Windows NT 4.0 Server（SP 3）	2GB

結果:

CPU あたりのユーザー数	ユーザーあたりのメモリー
100	2.4MB

分析:

このシステムは、標準的な複雑さのアプリケーションの拡張性をテストするために使用した最も安価なシステムの 1 つです。システムは、約 200 ユーザーを非常に効率よく処理できました。200 ユーザーを超えると、パフォーマンスが劇的に低下しました。このシステムは、標準クラスの複雑さに分類されるアプリケーションを最大 200 ユーザーが使用する小規模部門サーバーとして、費用効果があります。

14.5.2 Intel Pentium II Xeon-Base システム上の、標準的な複雑さのアプリケーション

パラメータ:

アプリケーションのサイズ / 複雑さ	CPU	RAM	オペレーティング・システム	スワップ
標準（2 ～ 10MB）	1MB L2 キャッシュを搭載した 2 つの 400MHz Pentium II Xeon	512MB	Windows NT 4.0 Server（SP 3）	2GB

結果:

CPU あたりのユーザー数	ユーザーあたりのメモリー
200	1.2MB

分析:

このシステムは、標準的な複雑さのアプリケーションの拡張性をテストするために使用した最新の Intel Pentium II Xeon-Base サーバーの 1 つです。システムは、約 400 ユーザーを非常に効率よく処理しました。400 ユーザーを超えると、パフォーマンスが劇的に低下しました。システムは、大規模な部門サーバーまたは小～標準規模ビジネス用のエントリレベル Enterprise Server として、費用効果があります。

14.5.3 エントリレベルの Sun UltraSparc サーバー上の、標準的な複雑さのアプリケーション

パラメータ:

アプリケーションのサイズ / 複雑さ	CPU	RAM	オペレーティング・システム	スワップ
標準	2 つの 248MHz Ultra Sparc	512MB	Solaris 2.5.1	2GB

結果:

CPU あたりのユーザー数	ユーザーあたりのメモリー
200	1.3MB

分析:

システムは、約 375 ユーザーを非常に効率よく処理しました。375 ユーザーを超えると、パフォーマンスが劇的に低下しました。システムは、過度のページングとスワッピング・アクティビティにより速度が低下するようで、実際のボトルネックは物理メモリーであったことを示しています。このシステムは、標準的な複雑さのアプリケーションを実行する大規模部門または小〜標準規模のビジネスで、費用効果があります。

14.5.4 Intel Pentium II Xeon-Base システム上の単純なアプリケーション

パラメータ:

アプリケーションのサイズ / 複雑さ	CPU	RAM	オペレーティング・システム	スワップ
小（2MB 未満）	1MB L2 キャッシュを搭載した 2 つの 400MHz Pentium II Xeon	512MB	Windows NT Server 4.0（SP 3）	2GB

結果:

CPU あたりのユーザー数	ユーザーあたりのメモリー
250	1MB

分析:

Pentium II Xeon-Base のサーバーは、小規模なアプリケーションで 500 ユーザーを非常に効率よく処理しました。

14.5.5 エントリレベルの Sun UltraSparc サーバー上の単純なアプリケーション

パラメータ:

アプリケーションのサイズ / 複雑さ	CPU	RAM	オペレーティング・システム	スワップ
小（2MB 未満）	2 つの 248MHz Ultra Sparc	512MB	Solaris 2.5.1	2GB

結果:

CPU あたりのユーザー数	ユーザーあたりのメモリー
240	1MB

分析:

このシステムは、エントリレベルの Sun Ultra Sparc システムです。システムは、約 480 ユーザーを非常に効率よく処理しました。480 ユーザーを超えると、パフォーマンスが劇的に低下しました。

15
トラブルシューティング・ソリューション

15.1 概要

この章では、Forms Services のトラブルシューティング・ソリューションに関する情報が次の項に記載されています。

- Forms Services のステータスのチェック
- Forms Services の開始
- Forms Services プロセスの停止
- Forms Services ログの開始
- トラブルシューティングの FAQ

15.2 Forms Services のステータスのチェック

Forms Services のステータスをチェックする手順は、次のとおりです。

Microsoft Windows NT の場合 :

1. **[Control]+[Alt]+[Delete]** を押して、「Windows NT のセキュリティ」ダイアログを表示します。
2. 「**タスク・マネージャ**」を選択します。
3. タスク・マネージャで、「**プロセス**」タブをクリックします。

サーバー・プロセスが実行中の場合、タスク・マネージャは、IFSRV60.EXE と呼ばれるプロセスと、IFWEB60.EXE と呼ばれる複数のプロセス（アクティブな接続ごとに 1 つずつ）を表示します。

UNIX の場合 :

UNIX プロンプトで `ps -ef | grep f60srvm` と入力し、**[Enter]** キーを押します。

プロセス ID のリストが画面に表示されます。Listener が実行されている場合は、リストには f60srvm というプロセスと、f60webm プロセスの複数のオカレンスが含まれます。（アクティブな接続ごとに 1 つのプロセスがあり、*pool* のデフォルト値が使用されている場合は次のユーザーに備える予備の接続が 1 つあります。*pool* が 5 に設定されている場合は、5 つの予備接続があります。）

15.3 Forms Services の開始

Forms Services を開始する手順は、次のとおりです。

Microsoft Windows NT 上のサービスとして開始する場合 :

既存の Forms Services のサービスを削除し、新しい起動パラメータを使用して再インストールできます。

1. コマンド・ウィンドウで、次のように入力します。

```
ifsrv60 -remove <FormsServerServiceNameToBeRemoved>
```

2. 次のように入力します。

```
ifsrv60 -install <NewFormsServerServiceName> port=<portNum> mode=<socket/http/https>
[pool=<numOfRunforms> log=<logfilePath> exe=<RunformexeName>]
```

3. **[Enter]** キーを押します。サーバー・プロセスが、指定されたポート番号で実行を開始します。

起動パラメータ定義は、5.4 項「Forms Services 起動パラメータの説明」を参照してください。

Microsoft Windows NT 上のコンソール・モードで開始する場合 :

1. タスクバーで、「**スタート**」→「**ファイル名を指定して実行**」を選択します。
2. 次のように入力します。

```
<ORACLE_HOME>¥6iserver¥bin¥ifsrv60 <FormsServerName> port=<portNum>
mode=<socket/http/https> [pool=<numOfRunforms> log=<logfilePath>
exe=<RunformexeName>]
```

3. **[Enter]** キーを押します。サーバー・プロセスが、指定されたポート番号で実行を開始します。

起動パラメータ定義は、5.4 項「Forms Services 起動パラメータの説明」を参照してください。

UNIX の場合 :

1. UNIX プロンプトで、次のように入力します。

```
cd <ORACLE_HOME>
```

2. **[Enter]** キーを押します。
3. 次のように入力します。

```
forms60_server start
```

4. **[Enter]** キーを押します。サーバーがバックグラウンドで実行を開始します。

起動パラメータ定義は、5.4 項「Forms Services 起動パラメータの説明」を参照してください。

15.4 Forms Services プロセスの停止

Forms Services プロセスを停止する手順は、次のとおりです。

Microsoft Windows NT 上の NT サービスとして停止する場合 :

1. コントロール・パネルで、「**サービス**」を選択します。
2. Forms Services プロセスを探して選択します。
3. 「**停止**」をクリックします。

Microsoft Windows NT 上のコンソール・モードで停止する場合 :

1. Forms Services のステータスをチェックします。サーバーが実行中の場合、タスク・マネージャは、IFSRV60.EXE と呼ばれるプロセスを表示します。
2. IFSRV60.EXE を選択し、「**プロセスの終了**」をクリックします。

UNIX の場合 :

1. Forms Services のステータスをチェックします。プロセス ID のリストが画面に表示されます。f60srvm プロセスのプロセス ID に注意してください。
2. UNIX プロンプトで、次のように入力します。

```
kill process_ID
```

または、次のように入力します。

```
kill -g
```

3. **[Enter]** キーを押します。

15.5 Forms Services ログの開始

次のように log オプションを使用してサーバーを開始すると、Forms Services によってログ・ファイルが作成されます。

```
ifsrv60 -install Forms60Server log=<¥PathName¥LogFileName> port=<portNum>
mode=<socket/http/https>
```

ログには、診断情報が含まれます。

15.6 トラブルシューティングの FAQ

問題	ソリューション
Web 対応の Forms アプリケーションを、Java 対応でない Web ブラウザで実行できない。	Web ブラウザが Java 対応かどうかが不明な場合は、Web ブラウザのネットワーク作業環境をチェックします。「Java を使用可能にする」および「JavaScript を使用可能にする」チェック・ボックスがチェックされている必要があります。
Forms Services を開始しようとすると、エラー・メッセージ " ポート 9000 にバインドできません " が表示される。	別のプロセスがポートを使用している可能性があります。そのプロセスは、Forms Services の別のオカレンスである可能性があります。Forms Services がまだ実行されていないことをチェックしてください。直前に Forms Services を停止した場合は、ポート 9000 への既存の接続が再オープンされるまでに 1 ～ 2 分かかることがあります。
Forms クライアントが Web ブラウザにダウンロードされない。	Oracle Java クラス・ファイル（コードベース）をポイントする仮想ディレクトリが定義されていることをチェックします。
すべての接続データが正しいにもかかわらず、クライアントがサーバーに接続できない。	サーバーが 128 ビットの暗号化を使用し、クライアントが（40 ビットの暗号化を使用しているため）この暗号化をサポートできない場合は、FORMS60_HTTPS_NEGOTIATE_DOWN 環境変数をチェックします。この変数が FALSE に設定されている場合は、サーバーはクライアントの接続リクエストを拒否します。必要に応じて、Java コンソールとサーバー・ログ・ファイル（使用可能な場合）をチェックして、クライアントとサーバーが使用する暗号化のレベルを確認します。
Forms Services が、アプリケーションのベース HTML ファイルで渡したユーザー ID、パスワードおよびデータベース SID パラメータ値を無視しているように見える。	値の前にパラメータ "userid=" があることを確認します。例 : userid=scott/tiger@inventory
Forms Services が、変数の変更を認識しないように見える。	Forms Services を停止し、再起動します。
セキュリティ・ファイアウォールを使用しているときに、プロキシ・サーバーを使用してファイアウォールの外部にアクセスすると、問題が発生する。	プロキシが手動構成に設定されていることを確認します。

問題	ソリューション
HTML ページとアプレットが起動時にダウンロードされ、アプレットが実行を開始するが、他のものは実行されていないように見える。	次の項目をチェックします。 最初に、Forms クライアントが本当に実行されていることを確認します。実行されている場合は、Web ブラウザのステータス・バーに「applet oracle.forms.engine.Main を実行中」というメッセージが表示されます。 このメッセージが表示されているにもかかわらず、アプリケーションが表示されない場合は、次の項目をチェックします。 1. Forms Services と Web サーバーが、同じアプリケーション・サーバーにインストールされていることを確認します。現在の Java 制限により、これらは同じサーバーにインストールする必要があります。 2. アプリケーションのベース HTML ファイルと構成ファイルをチェックして、.FMX ファイルに対して有効なディレクトリ・パスとファイル名が指定されていることを確認します。仮想ディレクトリ・パスではなく、物理ディレクトリ・パスを使用する必要があります。 3. Java コンソールを表示するように Web ブラウザの作業環境を設定してみます。この設定によって、ランタイム Java エラー・メッセージを参照できます。
アプレットが Forms Services に接続できない。	サーバー上の "mode" 設定が、ベース HTML ファイルの "connectionType" と一致していることを確認します。
ローカル・データベースへの接続で問題が発生する。	次の原因が考えられます。 * Net8 v2 接続文字列を指定していない場合は、エラーが発生します。Forms Services Runtime エンジンは、LOCAL、TWO_TASK などの型の接続文字列をアクセプトしません。 * Net8 v2 接続文字列を使用しているにもかかわらず、データベースに接続できない場合は、Forms Services が実行されていることを確認します。ほとんどのインストールでは、サーバーはリブート後に自動的に再起動されません。 * クライアント・マシンではなくアプリケーション・サーバー上の TNSNAMES.ORA ファイルに、有効な接続文字列が必要です。アプリケーション・ロジックは、ユーザーのクライアント・マシンではなくアプリケーション・サーバー上で実行されます。
CLASSPATH 環境変数を変更した後に、予期しない動作が発生する。	アプリケーション・サーバーまたはユーザーのマシン上で CLASSPATH 環境変数の設定を変更すると、予期しない結果が生成される可能性があります。この変数を、Forms Java クラス・ファイルが存在する場所とオーバーラップするディレクトリに設定すると、ファイル名のオーバーラップの原因となることがあります。

問題	ソリューション
いくつかの使用されていないプロセスがサーバー上で実行されているように見える。	Web 対応の Form Builder アプリケーションを実行している各ユーザーについて、アプリケーション・サーバー上で Forms Services ランタイム・プロセス（Windows 上の ifweb60.exe と ifsrv60、UNIX 上の f60webm と f60srvm）が開始することを思い出してください。各ランタイム・プロセスは、ユーザーがアプリケーションを終了したときに終了する必要があります。ユーザーがアプリケーションを正しく終了しないでブラウザを終了すると、プロセスがサーバー上に残ります。アプリケーションを正しく終了するには、メニューまたは [終了 / 取消し] キー機能を使用してからブラウザを終了する必要があります。

第II部

付録

A

Forms Services のパラメータ

A.1 概要

この付録には、Forms Services を構成するために使用するパラメータが含まれています。

A.2 Windows 95 および Windows NT のレジストリ

Windows 95 および Windows NT の場合、Oracle Universal Installer は新規の ORACLE セクションをレジストリ内に作成します。Oracle レジストリには、Oracle ホーム・ディレクトリ名、製品の作業環境ファイルの場所、およびヘルプ・ファイルの場所などを管理する構成パラメータが含まれています。Windows の Net8 を使用する場合、構成パラメータはネットワーク通信に使用するドライバおよび Net8 が操作パラメータのために使用する値も判断します。

A.2.1 レジストリの表示および変更

レジストリ・エディタを使用して Microsoft Windows レジストリを表示したり、オプションで編集できます。このエディタは、Windows ソフトウェアがインストールされているディレクトリにあります。

エディタを起動するには、次の手順を実行します。

1. 「**スタート**」→「**ファイル名を指定して実行**」を選択します。
2. `REGEDIT` と入力します。
3. 「**OK**」をクリックします。
4. レジストリ・エディタで、HKEY_LOCAL_MACHINE ノードを拡張します。
5. SOFTWARE ノードを拡張します。
6. Oracle 構成パラメータを表示するには、［ORACLE］キーをクリックします。
7. パラメータ名をダブルクリックして「文字列の編集」ダイアログ・ボックスを表示し、パラメータ値を変更できます。

8. 「値のデータ」テキスト・ボックス内の値を変更します。
9. 「**OK**」をクリックして、新しい値を使用します。

A.3 構成パラメータ

Oracle Installer は多くのパラメータを自動的に設定します。Oracle 製品には一部のパラメータが必要です。これらのパラメータは表 A-1 に列挙されています。その他のパラメータを使用すると、製品の動作をカスタマイズできます。これらについては、A.3.2 項「カスタマイズ可能パラメータ」で説明されています。

A.3.1 必須パラメータ

この項で列挙するパラメータは、Oracle Installer によって自動的に設定されたり、削除されます。これらのパラメータは、様々な Oracle 製品を正しく機能させるために必要です。

注意： この項で列挙されているパラメータの設定を変更しないでください。変更した場合、1 つ以上の Oracle 製品が正しく機能しなくなることがあります。

次に列挙するパラメータ内の *nn* は、製品またはコンポーネントのリリース番号を指定します。Oracle 製品の新規リリースにアップグレードする場合、この番号は変更されることがあります。

表 A-1 必須パラメータ

パラメータ	設定
BROWSER*nn*	<*ORACLE_HOME*>¥6iserver¥BROWSE*nn*
DE*nn*	<*ORACLE_HOME*>¥6iserver¥TOOLS¥COMMON*nn*
FORMS*nn*	<*ORACLE_HOME*>¥6iserver¥FORMS*nn*
GRAPHICS*nn*	<*ORACLE_HOME*>¥6iserver¥GRAPH*nn*
MM*nn*	<*ORACLE_HOME*>¥6iserver¥TOOLS¥COMMON*nn*
OCL*nn*	<*ORACLE_HOME*>¥6iserver¥GRAPH*nn*
PRO*nn*	<*ORACLE_HOME*>¥6iserver¥PRO*nn*
RDBMS*nn*	<*ORACLE_HOME*>¥6iserver¥RDBMS*nn*
RW*nn*	<*ORACLE_HOME*>¥6iserver¥REPORT*nn*
TK*nn*	<*ORACLE_HOME*>¥6iserver¥TOOLS¥COMMON*nn*
VGS*nn*	<*ORACLE_HOME*>¥6iserver¥TOOLS¥COMMON*nn*

A.3.2 カスタマイズ可能パラメータ

この項で列挙されているパラメータは、Oracle 製品の様々な外観を制御します。動作をカスタマイズするために、これらのパラメータ設定を変更できます。

次の項では、各パラメータのデフォルト設定（存在する場合）を列挙します。デフォルト値に自動設定されないパラメータを説明します。パラメータ・リストには、有効値の説明および例が含まれています。

FORMS60_PATH

デフォルト : <*ORACLE_HOME*>¥6iserver¥FORMS60¥PLSQLLIB

有効値 : すべてのドライブ上のすべてのディレクトリ

例 :

FORMS60_PATH=C:¥oracle¥apps¥forms;C:¥myfiles

このパラメータは、Form Builder ランタイム・アプリケーションで使用されるファイルの検索パスを指定します。これらには、フォーム・ファイル（.fmx）、メニュー・ファイル（.mmx）、PL/SQL ライブラリ（.pll）および実行時にアプリケーションがファイルからロードしようとするその他のオブジェクトが含まれます。たとえば、イメージ・ファイル scooter.tif をインポートする場合、Form Builder はこのファイルを見つけるために、FORMS60_PATH で指定されたディレクトリを検索します。

FORMS60_PATH では複数のディレクトリを指定できます。パス・リスト内でディレクトリ名を分離するには、セミコロン（;）を使用します。

FORMS60_REPFORMAT

デフォルト : なし

有効値 : HTML、PDF

例 :

FORMS60_REPFORMAT=HTML

RUN_PRODUCT を使用してフォームからレポートを実行するためにブラウザを起動する場合、FORMS60_REPFORMAT 環境変数を設定する必要があります。このパラメータはレポート・フォーマットを指定します。

FORMS60_TIMEOUT

デフォルト : 15

有効値 : 1 ～ 1440（1 日）

例 :

FORMS60_TIMEOUT=1440

このパラメータは、クライアントが Forms Services と通信しない場合に Forms Services プロセスが終了するまでの所要時間（分）を指定します。

GRAPHICS60_PATH

デフォルト : なし

有効値 : すべてのドライブ上のすべてのディレクトリ

例 :

GRAPHICS60_PATH=C:¥oracle¥apps¥graphics;C:¥myfiles

このパラメータは、Graphics ランタイム・アプリケーションで使用されるファイルの検索パスを指定します。これらには、図表ファイル（.ogr）、イメージ、外部問合せ、および実行時にアプリケーションがファイルからロードしようとするその他のオブジェクトが含まれます。たとえば、イメージ・ファイル scooter.tif をインポートする場合、Graphics Builder はこのファイルを見つけるために、GRAPHICS60_PATH で指定されたディレクトリを検索します。

GRAPHICS60_PATH では複数のディレクトリを指定できます。パス内のディレクトリを分離するには円記号（¥）を使用し、完全パスを分離するにはセミコロン（;）を使用します。

NLS_LANG

デフォルト : AMERICAN_AMERICA.WE8ISO8859P1

有効値 : 行の使用可能な値リストの詳細は、『NLS ガイド』を参照してください。または、CD の中の ¥bonus¥nls¥nlsd2r1.wri を参照してください。

例 :

NLS_LANG=AMERICAN_AMERICA.WE8ISO8859P1

このパラメータは、メッセージ・ファイルで表示される言語を設定します。NLS_LANG の構文の一例を次に示します。

NLS_LANG=<*language*>_<*territory*>.<*char_set*>

各項目の内容は次のとおりです。

- *Language* は、メッセージ、曜日および月の名前を表示するための言語およびその規則を指定します。

- *Territory* は、週および日数を計算するための地域およびその規則を指定します。
- *Char_set* は、UPPER、LOWER および INITCAP ファンクションで使用されるキャラクタ・セット、および ORDER BY 問合せで使用されるソート・タイプを指定します。この引数も、メッセージを表示するために使用されるキャラクタ・セットを制御します。

ORACLE_HOME

デフォルト : Window95 の場合は C:¥ORAWIN95、Windows NT の場合は C:¥ORANT

有効値 : すべてのドライブ上のすべてのディレクトリ

例 :

ORACLE_HOME=C:¥orawin95

このパラメータは、Windows 版 Oracle 製品がインストールされるホーム・ディレクトリを指定します。このディレクトリは、Oracle ディレクトリ階層で一番上のディレクトリです。

B

クライアント・ブラウザのサポート

B.1 概要

次のブラウザ構成のいずれかを使用することにより、ユーザーは Oracle Forms アプリケーションを Web 上で表示できます。

- ネイティブ JVM を使用する Internet Explorer 5
- Oracle JInitiator プラグイン（Netscape Navigator または Internet Explorer を使用）
- AppletViewer

注意： クライアントのブラウザで AppletViewer を使用する場合は、HTTPS 接続モードはサポートされません。

注意： クライアントのブラウザで Oracle JInitiator を使用する場合、HTTP モードおよび HTTPS モードを使用するには JInitiator のバージョン 1.1.7.30 が必要です。

B.2 構成パラメータとベース HTML ファイルはどのようにしてクライアントのブラウザに関連付けられるか

エンド・ユーザーが Web 対応のアプリケーションを起動すると (アプリケーションの URL へのリンクをクリックすることで)、次の処理が実行されます。

1. Forms サーブレットまたは CGI が、エンド・ユーザーで使用されているブラウザの種別を検出します。
2. formsweb.cfg ファイルを読み込んで、IE50 パラメータ設定を判別します（エンド・ユーザーが Internet Explorer 5 ブラウザを使用している場合）。

3. 次の表に従って、適切なベース HTML ファイルを選択します。

検出されたブラウザ	IE50 パラメータ設定	使用されるベース HTML ファイル	この付録内の該当する項
Internet Explorer 5	native	baseie.htm	B.3 項「ネイティブ JVM を使用する Internet Explorer 5」
Internet Explorer 5	jinitiator	basejini.htm	B.4 項「Oracle JInitiator」
Netscape Navigator またはバージョン 5 より前のバージョンの Internet Explorer	適用不可	basejini.htm	B.4 項「Oracle JInitiator」
その他のすべてのブラウザ	適用不可	base.htm	B.5 項「AppletViewer」

4. ベース HTML ファイル内の変数（%*variablename*%）を、FormsServlet.initArgs ファイル（サーブレット実装のみ）、formsweb.cfg ファイル（サーブレット実装と CGI 実装の両方）に指定された適切なパラメータ値で置き換え、URL リクエストがある場合はその問合せパラメータの値で置き換えます。
5. HTML ファイルをエンド・ユーザーのブラウザに送信します。

B.3 ネイティブ JVM を使用する Internet Explorer 5

オラクル社は、Microsoft 固有の署名済 CAB ファイル（f60all.cab）を提供します。これにより、Internet Explorer 5 内で、Oracle Forms Java アプレットを信頼されたアプレットとして実行できます。このブラウザ・オプションを使用すると、ブラウザのエンド・ユーザー構成を実行する必要性が少なくなります。

B.3.1 ソフトウェアのインストール

この項では、Forms アプリケーションをそのまま Microsoft Internet Explorer 5 で実行するために、クライアント・マシンにインストールする必要のあるソフトウェアを説明します。

Microsoft Internet Explorer 5 をクライアント・マシンにインストールします。このブラウザは Microsoft の Web サイト（http://www.microsoft.com/japan/ie/）からダウンロードできます。

Microsoft Internet Explorer 5 をインストールする場合、Microsoft Virtual Machine for Java コンポーネントもインストールする必要があります。インストール・オプションの「完全」または「カスタム」のいずれかを選択します。カスタム・インストールを選択した場合は、利用可能なコンポーネントのリストから「Microsoft Virtual Machine for Java」を手動で選択する必要があります。

B.3.2 Microsoft Internet Explorer のテスト

ブラウザが使用する Microsoft VM for Java がインストールされていること、ならびにそのバージョン・レベルを確認します。また、JDK 1.1 アプレットがそのままでブラウザで実行されることも確認します。

B.3.2.1 Microsoft JVM のチェック

1. Microsoft Internet Explorer 5 を起動します。
2. **「表示」→「Java コンソール」**を選択して、Java コンソールを表示します。
3. Java コンソールが表示され、Microsoft VM for Java のバージョンが 5.0.0.3167（またはそれ以降）であることが通知されます。

B.3.2.2 Java 1.1 アプレットのテスト

JavaSoft の Web サイト（http://www.javasoft.com/applets/jdk/1.1/index.html）にある例を使用して、ブラウザで Java 1.1 アプレットを実行できるかどうかをテストします。アプレットが表示されない場合は、実施済のブラウザおよび JVM の構成ステップを再確認し、テストを繰り返します。

B.3.3 Oracle Forms アプリケーションの起動

インストールと構成が完了し、さらにブラウザでの Java アプレットの実行テストが成功したら、Oracle Forms アプリケーションを Microsoft Internet Explorer 5 で正常に実行できます。

アプリケーションのベース HTML ファイルで、標準の Java の <APPLET> タグを使用してください。Oracle JInitiator 固有の <EMBED> タグと <OBJECT> タグは使用しないでください。標準の Java の <APPLET> タグを使用した HTML ファイルの例は、B.3.5 項「baseie.htm ファイルの変更」を参照してください。

B.3.4 トラブルシューティング

Oracle Forms Services アプリケーションが、実行中に表示されない。

通常は Internet Explorer 5 の構成エラーが原因です。Java コンソールに出力されるエラー・メッセージを参照して情報を収集します。最上位メニューから、**「表示」→「Java コンソール」**を選択してください。コンソールに出力されるエラー・メッセージをチェックします。一般的なエラー・メッセージは次のとおりです。

アプレットが起動せず、エラー・メッセージ「java.lang.ClassNotFoundException: sun.applet.AppletViewer」が表示される。

このエラー・メッセージは、Oracle Forms Services のバージョンが 6.0.5.30.2 以降でないことを示します。これより前のバージョンの Oracle Forms Server 6.0 では、このクラス・ファ

イルがクライアントのマシンに存在している必要がありました。Oracle Forms Services *6i* Patch2 をインストールしてください。

アプレットが起動せず、エラー・メッセージ「com.ms.security.SecurityException[oracle/forms/engine/Main.init]: cannot access file C:¥WINNT¥Java¥.hotjava」が表示される。

このエラー・メッセージは、イントラネット・ゾーンの Java アプリケーションは Java のサンドボックス外では実行できないようにセキュリティ設定が構成されていることを示します。Microsoft Internet Explorer 5 では、どのゾーンからページがロードされたかが右下角に表示されます。「イントラネット・ゾーン」と表示されるはずです。プロキシ・サーバーを使用している場合は、実行中のホスト Forms Services でプロキシ・サーバーをバイパスしていることを確認してください。

B.3.5 baseie.htm ファイルの変更

標準の Java の <APPLET> タグを使用して Forms Services Java クライアントを起動する、ベース HTML ファイルの例を示します。baseie.htm は標準の Applet タグの使用に適した HTML ページの例であり、Forms Services 製品に同梱されています。

```
<HTML>
<BODY>
<APPLET
    CODEBASE="/web_forms/"
    CODE="oracle.forms.engine.Main"
    WIDTH="800"
    HEIGHT="600">
    <PARAM NAME="serverPort" VALUE="9000">
    <PARAM NAME="CABBASE" VALUE="f60all.cab">
    <PARAM NAME="serverArgs" VALUE="module= grid2.fmx userid=scott/tiger ">
    <PARAM NAME="lookAndFeel" VALUE="oracle">
    <PARAM NAME="colorScheme" VALUE="Titanium">
</APPLET>
</BODY>
</HTML>
```

B.4 Oracle JInitiator

この項では、Oracle JInitiator をユーザーの Web ブラウザのプラグインとして使用する利点について説明します。Oracle JInitiator を使用すると、Netscape Navigator または Internet Explorer を使用して Forms Services アプリケーションを実行できます。ブラウザのデフォルトの JVM ではなく、クライアント上の特定の Java 仮想マシン (JVM) を使用するように指定する機能が提供されます。

Oracle JInitiator は、Netscape Navigator のプラグイン、および Internet Explorer の ActiveX コンポーネントとして実行されます。Oracle JInitiator はブラウザによって提供されるデフォ

ルトの JVM を、置き換えたり変更せずに、代替の JVM をプラグイン形式で提供します。かわりに、プラグインのフォームで代替の JVM が提供されます。

オラクル社からは 2 つの JAR ファイル（f60all.jar および f60all_jinit.jar）が提供されます。f60all.jar は標準の JAR ファイルであり、f60all_jinit.jar は Oracle JInitiator でのみ使用できる高圧縮された JAR ファイルです。

B.4.1 Oracle JInitiator を使用する理由

Oracle JInitiator では、Web ブラウザを使用して透過的に起動できる、保証されサポート可能な Java Runtime Environment（JRE）がクライアントのデスクトップに提供されます。

Oracle JInitiator は、JavaSoft の Java Plug-in の Oracle バージョンです。JavaSoft Plug-in は、ブラウザ内から起動できる JavaSoft JRE の配布メカニズムです。また、Oracle JInitiator では、Oracle が保証した JRE の配布メカニズムが提供されます。これにより、Forms Developer アプリケーションを安定しサポートされた方法で、ブラウザ内から実行できます。

Oracle JInitiator では、Forms Developer アプリケーションを実行するために保証されたプラットフォームが提供されるのみではなく、標準 JavaSoft Java Plug-in の他に多くの追加機能も提供されます。これらの機能には、JAR ファイルのキャッシュ書込み、増分の JAR ファイルのロード、およびアプレットのキャッシュ書込みが含まれます。

B.4.2 Oracle JInitiator の利点

Oracle JInitiator には次の利点があります。

- 旧リリースのブラウザで、最新の Oracle が保証する JVM を実行できます。
- 異なるブラウザ間で JVM の一貫性が保証されます。
- 信頼性が高い実行プラットフォームです。JInitiator は、Forms Services での使用が完全にテストされ、保証されています。
- 高性能な実行環境です。JInitiator によって、アプリケーション・クラス・ファイルがキャッシュに自動的に書き込まれます。これにより、アプリケーションを高速起動できます。
- 自己インストールおよび自己メンテナンスが可能な実行環境です。JInitiator はプラグインまたは ActiveX コンポーネントのように、自身を自動的にインストールおよび更新します。ローカルにキャッシュされたアプリケーション・クラス・ファイルは、アプリケーション・サーバーから自動的に更新されます。

B.4.3 Oracle JInitiator の使用方法

クライアントのブラウザが Oracle JInitiator を使用するように指定された HTML ファイルを最初に見つけたとき、Oracle JInitiator はアプリケーション・サーバーからクライアント・マシンに自動的にダウンロードされます。これにより、Windows 95 および Windows NT 4.0 プラットフォーム上の Netscape Navigator または Internet Explorer 内で Forms および Graphics アプリケーションを直接実行できます。

Oracle JInitiator は、ブラウザによって提供される標準プラグイン・メカニズムを使用してインストールおよび更新されます。Oracle JInitiator のインストールでは、Forms Developer アプリケーションを信頼されたアプレットとして Oracle JInitiator 環境で実行するために必要なステップが実行されます。

注意： クライアントのブラウザで Oracle JInitiator を使用する場合、HTTP モードおよび HTTPS モードを使用するには JInitiator のバージョン 1.1.7.30 が必要です。

B.4.4 サポートされる構成

Oracle JInitiator では、次の構成がサポートされています。

	Internet Explorer 4.0	Internet Explorer 5	Navigator 4.0	Navigator 4.5
Windows 95	○	○	○	○
Windows NT	○	○	○	○

B.4.5 システム要件

Oracle JInitiator の最小システム要件を次に示します。

- Windows 95 または Windows NT 4.0
- Pentium 90 MHz 以上のプロセッサ
- 12MB のハード・ディスク空き領域（20MB を推奨）
- 16MB のシステム RAM（24MB を推奨）

B.4.6 Netscape Navigator での Oracle JInitiator の使用方法

Oracle JInitiator では、QuickTime ムービーまたは Shockwave アニメーション機能など他のプラグインと同様にブラウザ内で実行できるように、Netscape Navigator プラグイン・アーキテクチャが使用されています。Web アプリケーション開発者は Netscape HTML <EMBED> タグを使用して、プラグインを Web ページの一部として実行するように指定できます。これにより、ユーザーの介在を最小にして、Oracle JInitiator を Web ブラウザ内で実行できます。

Navigator が Oracle JInitiator の使用が指定された HTML ページを最初に見つけたとき、Oracle JInitiator ダウンロード・ページを指示する HTML ページに "Plug-in Not Loaded" ダ

イアログが表示されます。続いて、各オペレーティング・システム用の Oracle JInitiator バージョンをダウンロードし、インストールできます。

Oracle JInitiator のインストール後、Navigator をシャットダウンして、再起動します。続いて、元の HTML ページを再び表示します。Oracle JInitiator はアプレットを解放するために、<EMBED> タグ内のパラメータを実行および使用します。Navigator が Oracle JInitiator の使用が指定された Web ページを次に見つけたとき、ユーザーが介在することなく、Navigator はプラグインをローカル・ディスクから透過的にロードおよび実行します。

B.4.7 Microsoft Internet Explorer での Oracle JInitiator の使用方法

Oracle JInitiator は Microsoft Internet Explorer 拡張メカニズムを使用して、ActiveX コントロールおよび COM コンポーネントのダウンロードおよびキャッシュ書込みを行います。Web アプリケーション開発者は HTML <OBJECT> タグを使用して、ActiveX コントロールまたは COM コンポーネントを Web ページの一部として実行するように指定できます。このようなコンポーネントには Oracle JInitiator が含まれます。

Internet Explorer は Oracle JInitiator 使用を指定するように変更された HTML ファイルを最初に見つけたとき、オラクル社によって VeriSign デジタル署名が行われた ActiveX コントロールをダウンロードするかどうかをユーザーに確認します。「はい」をクリックすると、Internet Explorer は Oracle JInitiator のダウンロードを開始します。Oracle JInitiator が実行され、アプレットの内容を表現するために <OBJECT> タグ内のパラメータが使用されます。Internet Explorer が Oracle JInitiator をサポートするように変更された Web ページを次に見つけたとき、ユーザーが介在することなく、Oracle JInitiator をローカル・ディスクから透過的にロードおよび実行します。

B.4.8 Oracle JInitator プラグインの設定

Oracle JInitator プラグインを設定するには、次の手順を実行します。

- Oracle JInitiator HTML マークアップをベース HTML ファイルに追加します。
- Oracle JInitiator をサーバーにインストールします（サーバー・ベースのテスト用のみ）。
- Oracle JInitiator ダウンロード・ファイルをカスタマイズします。
- Oracle JInitiator をダウンロード可能にします。

B.4.8.1 Oracle JInitiator マークアップのベース HTML ファイルへの追加

Oracle JInitiator マークアップをベース HTML ファイルに追加するには、次の手順を実行します。

1. ベース HTML ファイルをテキスト・エディタでオープンします。
2. OBJECT および EMBED タグを追加します。

   追加マークアップの例は B.4.10 項「ベース HTML ファイルの Oracle JInitiator タグ」を参照してください。

B.4.8.2 Oracle JInitiator ダウンロード・ファイルのカスタマイズ

Oracle JInitiator のダウンロード・ファイル（JINIT_DOWNLOAD.HTM）は、ユーザーが Oracle JInitiator ファイルをダウンロードできるテンプレート HTML ファイルです。

Oracle JInitiator ダウンロード・ファイルをカスタマイズするには、次の手順を実行します。

1. JINIT_DOWNLOAD.HTM ファイルを HTML またはテキスト・エディタでオープンします。
2. 必要に応じてテキストを変更します。
3. 変更を保存します。

B.4.8.3 Oracle JInitiator をダウンロード可能にする

Oracle JInitiator をダウンロード可能にするには、次の手順を実行します。

1. jinit11*x*.EXE を Web サーバーにコピーします。

   jinit11*x*.EXE は、ベース HTML ファイル内で指定された位置にコピーする必要があります。

2. JINIT_DOWNLOAD.HTM を Web サーバーにコピーします。

   JINIT_DOWNLOAD.HTM は、ベース HTML ファイル内で指定された位置にコピーする必要があります。

B.4.9 Oracle JInitiator プラグインの変更

Oracle JInitiator プラグインを変更するには、次の手順を実行します。

- Oracle JInitiator のキャッシュ・サイズを変更します。
- Oracle JInitiator のヒープ・サイズを変更します。
- **Oracle JInitiator** 用にプロキシ・サーバー設定をチェックおよび変更します。
- Oracle JInitiator 出力を表示します。

B.4.9.1 Oracle JInitiator キャッシュ・サイズの変更

Oracle JInitiator のキャッシュ・サイズを変更するには、次の手順を実行します。

1. 「スタート」メニューで、**「スタート」→「プログラム」→「JInitiator Control Panel」**を選択します。
2. 「**Basic**」タブをクリックします。
3. 「Java Run Time Parameters」フィールドで、Dcache サイズを指定します。たとえば、Dcache.size=20000000 と指定すると、キャッシュ・サイズは 20MB に設定されます。

Oracle JInitiator のデフォルト・キャッシュ・サイズは 20000000 です。これは Oracle JInitiator のインストール時に自動設定されます。

B.4.9.2 Oracle JInitiator ヒープ・サイズの変更

Oracle JInitiator のヒープ・サイズを変更するには、次の手順を実行します。

1. 「スタート」メニューで、**「スタート」→「プログラム」→「JInitiator Control Panel」**を選択します。
2. 「**Basic**」タブをクリックします。
3. 「Java Run Time Parameters」フィールドで、mx サイズを指定します。たとえば、mx64m と指定すると、最大ヒープ・サイズは 64MB に設定されます。

Oracle JInitiator のデフォルト最大ヒープ・サイズは 64MB です。これは Oracle JInitiator のインストール時に自動設定されます。

B.4.9.3 Oracle JInitiator 用のプロキシ・サーバー設定のチェックおよび変更

Oracle JInitiator 用にプロキシ・サーバー設定をチェックおよび変更するには、次の手順を実行します。

1. 「スタート」メニューで、**「スタート」→「プログラム」→「JInitiator Control Panel」**を選択します。
2. 「**Proxies**」タブをクリックします。
3. ブラウザの構成ダイアログ・ボックスの設定を **Oracle JInitiator** で使用できるようにするために、「**Use browser setting**」チェックボックスを選択します。別のプロキシ・サーバー設定を使用する場合は、このボックスにはチェックを付けないでください。次に、「**Proxy Address**」フィールドにプロキシ・サーバーのホスト名を入力します。

B.4.9.4 Oracle JInitiator 出力の表示

Oracle JInitiator 出力を表示するには、次の手順を実行します。

1. 「スタート」メニューで、**「スタート」→「プログラム」→「JInitiator Control Panel」**を選択します。
2. 「**Basic**」タブをクリックします。
3. デバッグ出力を使用可能にするために、「**Show Java Console**」チェックボックスをオンにします。

B.4.10 ベース HTML ファイルの Oracle JInitiator タグ

この例では、Microsoft Internet Explorer および Netscape Navigator 用の Oracle JInitiator マークアップを示します。これらのタグをベース HTML ファイルに追加すると、Netscape および Microsoft のブラウザ内でアプリケーションを実行できます。

```
<HTML>
<BODY>
<P>
<OBJECT classid="clsid:9F77a997-F0F3-11d1-9195-00C04FC990DC"
WIDTH=600
HEIGHT=480
codebase="http://acme.com/jinit11711.exe#Version=1,1,7,11">
<PARAM NAME="CODE" VALUE="oracle.forms.engine.Main" >
<PARAM NAME="CODEBASE" VALUE="/forms60code/" >
<PARAM NAME="ARCHIVE" VALUE="/forms60code/f60all.jar" >
<PARAM NAME="type" VALUE="application/x-jinit-applet;version=1.1.7.11">
<PARAM NAME="serverPort" VALUE="9000">
<PARAM NAME="serverArgs" VALUE="module=order.fmx">
<PARAM NAME="serverApp" VALUE="default">
<COMMENT>
<EMBED type="application/x-jinit-applet;version=1.1.7.11"
java_CODE="oracle.forms.engine.Main"
java_CODEBASE="/forms60code/"
java_ARCHIVE="/forms60code/f60all.jar"
WIDTH=600
HEIGHT=480
serverPort="9000"
serverArgs="module=order.fmx"
serverApp="default"
pluginspage="http://acme.com/jinit_download.htm">
<NOEMBED>
</COMMENT>
</NOEMBED></EMBED>
</OBJECT>
</BODY>
</HTML>
```

B.4.11 Oracle JInitiator FAQ

次の項では、Oracle JInitiator に関してよく聞かれる質問について詳しく説明します。

B.4.11.1 保証および可用性

Oracle JInitiator はいつから利用可能ですか？

カスタム Oracle Developer アプリケーションの配置のために Forms JInitiator は 1998 年 9 月から利用可能です。Oracle Applications では、1999 年 2 月から Oracle JInitiator の保証を行っています。

Oracle JInitiator はどのようにして配布されるのですか？

Forms Developer リリース 6*i* から、Oracle JInitiator は Forms Developer の配布 CD に収録されています。Oracle JInitiator の更新情報は、オラクル社カスタマ・サポートからも入手できます。

Oracle JInitiator は Windows 以外のプラットフォームで動作するようになりますか？

オラクル社では現在、Oracle JInitiator を Microsoft Windows 以外のプラットフォームに移植する予定はありません。ただし、弊社は Forms Developer アプリケーションを Microsoft Windows 以外のプラットフォームで実行するためのサポートおよび保証を行うため、多くのハードウェア・ベンダーと密接な関係を築いています。

Oracle JInitiator は、Netscape Navigator および Internet Explorer のどのバージョンで保証されていますか？

各 Oracle JInitiator リリースの最終品質保証テスト時に、Oracle JInitiator はこれらのブラウザの最新リリースに対して保証されます。オラクル社は、これらのブラウザの旧リリースについてもサポートを行う予定です。保証されるブラウザのバージョンの詳細は、Oracle JInitiator リリースの添付ドキュメントを参照してください。

JavaSoft Java Plug-in と Oracle JInitiator の違いは何ですか？

主な違いは、Oracle JInitiator には Oracle が保証する JRE が含まれているのに対し、JavaSoft Java Plug-in は JavaSoft JDK リファレンス・インプリメンテーションに同梱されることです。JavaSoft のインプリメンテーションは、Forms Developer アプリケーションで保証されていません。Forms Developer では JRE に極度の要求が行われるので、弊社は JavaSoft の JRE を厳しい条件下で実行できるように変更しました。

オラクル社は JavaSoft に対して、拡張機能の追加を依頼していますが、JavaSoft が新規バージョンに拡張機能を組み込むことを待てません。

JavaSoft Plug-in は、ブラウザ内から起動できる JavaSoft JRE の配布メカニズムです。また、Oracle JInitiator では、Oracle が保証した JRE の配布メカニズムが提供されます。これにより、Forms Developer アプリケーションを安定しサポートされた方法で、ブラウザ内から実行できます。

オラクル社は Oracle JInitiator 製品を管理しており、製品を完全にサポートしています。オラクル社のお客さまには、オラクル社カスタマ・サポートから、アプリケーションをサポートするために必要な適切なレベルのサポートが提供されます。

Oracle JInitiator では、Forms Developer アプリケーションを実行するために保証されたプラットフォームが提供されるのみではなく、標準 JavaSoft Java Plug-in の他に多くの追加機能も提供されます。これらの機能には、JAR ファイルのキャッシュ書込み、増分の JAR ファイルのロード、およびアプレットのキャッシュ書込みが含まれます。

なぜ、Oracle は JavaSoft が提供する JRE を使用しないで、特定の JRE を保証し配布するのですか？

Forms Developer はコンピュータ化へのコストを削減する方法として、サーバー・ベースの配布に移行するお客さまに対応してきました。また、ビジネスに必要な既存アプリケーションへの投資を保護する必要性を認識しています。

お客さまに Java の一意な要求が行われる Java プラットフォームで、特に与えられたアプリケーションが大規模で複雑であっても、完全に変更のない既存のアプリケーションの実行機能を提供します。

JavaSoft Java Plug-In を使用して Forms Developer アプリケーションを実行できますか？

JavaSoft Plug-In を使用して Forms Developer アプリケーションを配置することは保証されていません。したがって、これはサポートされる配置構成ではありません。現在、Oracle JInitiator によって提供される JRE には、JavaSoft によって提供される JRE ではまだ使用できない多くの拡張機能が組み込まれています。さらに、オラクル社は、オラクル社カスタマ・サポートを通して Oracle JInitiator を完全にサポートしています。

オラクル社ではネイティブ・ブラウザ配布をサポートする予定はありますか？

ネイティブ・ブラウザ・サポートを行う際の主な問題は、Forms Services が必要とする Java の同じバージョンおよび品質レベルをサポートするために、ブラウザ・ベンダーおよびプラットフォーム提供者に依存する必要があることです。オラクル社はこの依存があるため、お客さまが必要とする期間にネイティブ・ブラウザの配布を、配布オプションとして保証できませんでした。したがって、弊社は Oracle JInitiator をインターネットによるアプリケーション配布方法として完全に保証しています。これにより、Forms Services アプリケーションを利用するプラットフォームの安定性およびサポートが保証されます。

B.4.11.2 サポート

誰が Oracle JInitiator のサポートを提供するのですか？

オラクル社がオラクル社カスタマ・サポートを通して Oracle JInitiator を完全にサポートします。

Oracle JInitiator は Forms Services のどのバージョンをサポートしていますか？

Oracle は、クライアント上で Oracle JInitiator を実行する Forms Services リリース 1.6 以降をサポートします。

Oracle JInitiator は Oracle Application でサポートされていますか？

はい。Oracle Application グループでは、Netscape Navigator 4.06 以降および Microsoft Internet Explorer 4.0 以降で Oracle Application を実行する場合、Oracle JInitiator の使用が保証されています。

B.4.11.3 インストール

Forms Developer アプリケーションを Web ブラウザで実行するために、クライアント上に何をインストールする必要がありますか？

ブラウザが Oracle JInitiator を必要とする HTML ページを最初に見つけたとき、Oracle JInitiator は Netscape Navigator および Microsoft Internet Explorer によって提供される標準ブラウザ拡張メカニズムを使用して、自身をクライアント・マシンに自動的にダウンロードできます。続いて、プラグインまたは ActiveX オブジェクトの追加に必要なメソッドを使用して、Oracle JInitiator は現在使用しているブラウザにインストールされます。

クライアントにダウンロードされる Oracle JInitiator はどの程度の大きさですか？

圧縮された Oracle JInitiator 配布版は約 8MB です。クライアントに完全インストールする場合は、約 10MB になります。

ユーザーが詳細を入力する必要のない、Oracle JInitiator のサイレント・インストールは実行できますか？

Oracle JInitiator では、ユーザーが InstallShield によって提供されるインストール・プロセスをアクティブに実行する必要がないサイレント・インストール・モードがサポートされています。サイレント・インストールを実行するには、Oracle JInitiator 配布版をマシンにダウンロードする必要があります。続いて、ダウンロードした実行ファイルの実行時に、コマンドラインからまたは Windows の「ファイル名を指定して実行」ダイアログから "`-s -sm`" を指定します。

たとえば、サイレント・インストールをコマンドラインから実行する場合、DOS シェルをオープンし、次のように入力します。

```
C:¥TEMP> jinit.exe -s -sm
```

Windows の「ファイル名を指定して実行」ダイアログを使用してサイレント・インストールを実行するには、**「スタート」→「ファイル名を指定して実行」**をクリックしてから、表示される「ファイル名を指定して実行」ダイアログで `jinit.exe -s -sm` と入力します。

ユーザーの介在が不必要な Oracle JInitiator インストールをセントラル・サーバーから実行できますか？

ホスト・オペレーティング・システムによって提供されるこの機能を使用すると、ユーザーが介在することなく、Oracle JInitiator を各クライアント・デスクトップにインストールできます。これには、システム管理者が各クライアント・マシンにアクセスし、サイレントで GUI を使用しない Oracle JInitiator インストール・オプションを実行することも含まれます。

Oracle JInitiator で、実行中のブラウザと同じプロキシ・サーバー構成などを使用できますか？

Oracle JInitiator の操作は、Oracle JInitiator の「コントロール・パネル」を使用して制御されます。Oracle JInitiator の「コントロール・パネル」は、Oracle JInitiator のインストール時にインストールされます。「コントロール・パネル」には、**「スタート」→「プログラム」**メニューからアクセスできます。

Oracle JInitiator の「コントロール・パネル」を使用して、Oracle JInitiator が特定のプロキシ設定または起動ブラウザによって提供されるデフォルトを使用するように構成できます。「**Proxies**」タブを選択し、適切な設定を挿入します。

どのようにして Oracle JInitiator の新規バージョンをブラウザ・クライアントでダウンロードし、インストールするのですか？

Oracle JInitiator は使用しているブラウザのタイプに応じて、Netscape プラグインまたは Microsoft ActiveX オブジェクトとして機能します。ブラウザは MIME タイプを使用して、HTML ページ・リクエストと必要なプラグインまたは ActiveX オブジェクトの間でマッピングを行います。各 Oracle JInitiator インストールには、特定の MIME タイプが対応付けられています。ブラウザが不明な MIME タイプが含まれた HTML ページをロードする場合、このページに必要なプラグインまたは ActiveX オブジェクトが含まれていないことをユーザーに通知します。続いて、これを取り出すためのダイアログ・ボックスが表示されます。

アプリケーションの HTML ページで指定された MIME タイプを以降のバージョンに変更することにより、ブラウザはその MIME タイプにとって有効なプラグインまたは ActiveX オブジェクトが含まれていないことを検出します。続いて、リクエストに完全に応えるために、新規ファイルをダウンロードするようプロンプトが表示されます。

例 :

HTML ページ HR.HTML を使用すると、HR アプリケーションを実行できます。HR.HTML ページは、MIME タイプ値を使用して Oracle JInitiator バージョン 1.1.5.21.1 を使用する必要があることをブラウザに示します。

Oracle JInitiator の以降のリリースを入手しサーバーにインストールしている場合、クライアント・ブラウザでは、新規バージョン・リリース情報が含まれている HR.HTML ファイル内のバージョンに関する行を変更することにより、新規バージョンを使用できます。

Netscape の "Plug-in Not Loaded" ダイアログで「取消し」ボタンをクリックしましたが、Oracle JInitiator をインストールするためのプロンプトが表示されません。どのようにしてプラグインをインストールすればよいですか？

Netscape では、Windows レジストリを使用して、インストール済プラグインに関する情報が格納されています。"Plug-in Not Loaded" ダイアログが表示されると同時に、Netscape はプラグインが実際にインストールされているのかどうかにかかわらず、プラグインの詳細をこのレジストリに書き込みます。特定のプラグインを使用する必要があるページが見つかった場合、Netscape はレジストリの設定に従って、プラグインがインストールされていると判断します。したがって、"Plug-in Not Loaded" ダイアログが再び表示されることはありません。これを解決するには、Netscape で「プラグイン欠落」アイコンをクリックして、プラグインをロードします。これにより、Netscape で「Plug-In を入手」ダイアログ・ボックスが表示されます。

異なる MIME タイプを含む多くの HTML ページを持っています。Oracle JInitiator の最新リリースは、これらの旧 MIME タイプで実行できますか？

Netscape ブラウザでは現在、特定プラグインの指定 MIME タイプを格納するために使用する場合、256 文字の上限があります。Microsoft Internet Explorer では拡張可能ブラウザ・オブジェクト・アーキテクチャが使用されているため、この制限はありません。Oracle JInitiator はこの制限内で、できる限り多くの旧 MIME タイプに対して逆向きのサポートを提供します。

特定リリースでサポートされている MIME タイプの詳細は、Oracle JInitiator リリースの添付ドキュメントおよびリリース・ノートを参照してください。

Oracle JInitiator のすべてのバージョンで Forms Developer アプリケーションを実行できますか？

はい。Oracle は、Oracle JInitiator のすべてのインストール・バージョンで Forms Developer アプリケーションを実行できる、汎用 MIME タイプを提供しています。この MIME タイプ・アプリケーション x-jinit-applet は、Oracle JInitiator のすべてのバージョンで認識されます。常にこの MIME タイプを使用することで、ブラウザによって Oracle JInitiator の以降のバージョンをアップグレードできます。

B.4.11.4 Oracle JInitiator の操作

Forms アプレット・ウィンドウを同じ起動ブラウザ・ウィンドウ内で実行できますか？

Forms Server リリース *6i* では、同じブラウザ・ウィンドウ内および新規ウィンドウ内での Forms アプレットの実行がサポートされています。これは構成変更可能なオプションであり、ベース HTML ファイルのパラメータとして設定されます。

現行ブラウザ・ページのナビゲートを終了した場合、実行中の Forms Developer アプリケーションはどうなりますか？

Oracle JInitiator には、実行中の Java アプリケーションをキャッシュに書き込み、現行ブラウザ・セッション中必要なときに取り出すことができる追加機能が含まれています。つまり、Forms アプリケーションの実行中に異なるページにナビゲートしてから Forms アプリ

ケーション・ページに戻る場合、実行中の Forms アプリケーションは元の状態のまま表示されます。

Oracle JInitiator を使用してカスタム開発 Java アプリケーションを実行できますか？

Oracle JInitiator では、Oracle 開発チームが拡張した標準 JavaSoft JVM が使用されています。したがって、カスタム Java アプリケーションを実行できます。ただし、Oracle は現在、Forms Developer、Oracle Enterprise Manager および Oracle Discoverer など Oracle Java ベース・アプリケーションの実行時のみ、Oracle JInitiator をサポートしています。Oracle では、カスタム Java アプリケーションを実行するために Oracle JInitiator を使用することはサポートしていません。

Oracle JInitiator と JavaSoft Java Plug-in は同じマシン上で共存できますか？

はい。これらは異なる MIME タイプを使用してプラグインを起動するので、同じブラウザ・インストールで共存できます。

同じブラウザ・インスタンス内で JavaSoft Java Plug-in と同時に使用された場合、Oracle JInitiator は正しく共存し動作しますか？

いいえ。動的ロード可能なライブラリはロードされ、JVM 動的ロード可能なライブラリは指定されるので、Oracle JRE および JavaSoft JRE は同じブラウザ・インスタンス内から同時に実行できません。つまり、JavaSoft Java Plug-in を使用しているブラウザ・ユーザーが、同じブラウザ・インスタンス内で Oracle JInitiator に切り替えることはできません。Oracle JInitiator および JavaSoft の Java Plug-in を使用する異なるアプリケーションを切り替える場合、ブラウザを停止し再起動する必要があります。

JavaSoft Java Plug-in および Oracle JInitiator を使用して、異なる JRE を使用するオプションがあります。Forms Developer アプリケーションを実行するために Oracle 保証 JRE を使用するように構成されている場合、JavaSoft Java Plug-in を使用できますか？

保証およびサポートされている組合せは、Oracle JInitiator と Oracle JRE のみです。JavaSoft 標準に準拠する Oracle JRE には、JavaSoft JRE に対するバグ修正が含まれています。これにより、Forms Developer アプリケーションは正しく実行されます。オラクル社は Oracle の拡張機能が JavaSoft に伝達され、標準 JRE に適用されるように、JavaSoft と密接な関係を築いています。しかし、改良された JavaSoft JRE がリリースされるのを待つことはできません。

次の図は、Oracle JInitiator の「コントロールパネル」および Java ランタイム環境値の正しい設定を示しています。

B.4.11.5 キャッシュ書込み

Oracle JInitiator は、アプリケーション実行時にダウンロードされた Java クラス・ファイルをキャッシュ書込みできますか？キャッシュ書込みできる場合、Java クラス・ファイルは 1 回のみダウンロードされ、アプリケーションが起動されるたびにダウンロードされないのですか？

はい。Oracle JInitiator では、Java アプリケーション実行時にダウンロードする JAR ファイル用の持続キャッシュ書込みメカニズムが提供されます。JAR ファイルは、Java アプリケーションによって使用される一連の Java クラス・ファイルを含む標準 Java アーカイブです。

必要な各クラス・ファイルを複数回ダウンロードするのではなく、すべての必要なクラス・ファイルを 1 つの JAR ファイルに入れることにより、ダウンロードは一度ですみます。

Oracle JInitiator は JAR ファイルをクライアントにキャッシュ書込みすることにより、アプリケーションで必要になるたびに JAR ファイルをダウンロードする必要性が少なくなります。JAR ファイルが最初に必要になったとき、Web サーバーからダウンロードされ、ローカル・クライアント・マシンに保存されます。JAR ファイルが次に必要になったとき、Oracle JInitiator は JAR ファイルが格納されているかどうかを確認するためにキャッシュ・ディレクトリを調べます。格納されている場合、これをローカル・ディレクトリから使用し、ファイルを Web サーバーから再ダウンロードしません。これにより、ユーザーの多くの時間および一般に使用されるアプリケーションのネットワーク・トラフィックを節約できます。たとえば、アプリケーションが 2MB の JAR ファイルを使用しており、2MB のファイルを 5 秒でダウンロードできる高速イーサネット接続を使用する場合、アプリケーション起動時に 5 秒節約できます。2MB のファイルをダウンロードするのに 10 分を要する低速ダイアルアップ・ネットワークで実行する場合、アプリケーション起動時に 10 分節約できます。

Oracle JInitiator キャッシュ書込みテクノロジはどのようにして機能するのですか？

Oracle JInitiator では、ブラウザのセッションとは無関係に JAR ファイルをキャッシュに書き込みます。Oracle JInitiator は、ダウンロードされた JAR ファイルをローカル・クライアント・マシンに格納します。したがって、次に JAR ファイルが必要なときにダウンロードする必要がありません。

JAR ファイルが要求されたとき、Oracle JInitiator は JAR ファイルが前回要求され、ダウンロードされ、格納されていないかどうかを確認するために、キャッシュ・ディレクトリをチェックします。JAR ファイルが存在しない場合、Oracle JInitiator は JAR ファイルを Web サーバーからダウンロードし、次回使用するためにキャッシュに格納します。キャッシュ・ファイルには、Oracle JInitiator が JAR ファイル、および Web サーバーによって報告される要求ファイルの最終変更日付を一意に識別するための追加情報が格納されています。

キャッシュに JAR ファイルが存在する場合、格納された JAR ファイルが現行のものであるかどうかを確認するために、Web サーバーをチェックする必要があります。Oracle JInitiator はキャッシュ書込みされた JAR ファイルに含まれる最終変更日付を調べ、サーバー上の JAR ファイルが変更されているかどうかを Web サーバーに確認します（標準 HTTP 相互作用を使用）。Web サーバーは、サーバーに格納されたファイルの最終変更日付およびタイムスタンプを使用します。続いて、ステータス・コード 200 で新しいファイルを Oracle JInitiator に提供します。または、ステータス・コード 304 を戻します。これは、キャッシュ内のファイルが現行のものであることを示します。

キャッシュ書込みされた JAR ファイルが現行のものではない場合、新しいファイルがダウンロードされ、次回使用するためにキャッシュ・ディレクトリに格納されます。ファイルが現行のものである場合、Oracle JInitiator はこのファイルをキャッシュ・ディレクトリからロードし、前回使用されたことを示すために、キャッシュ書込みされたファイルのタイムスタンプを更新します。

キャッシュ書込みされた JAR ファイルはどこに格納されるのですか？

デフォルトでは、Oracle JInitiator はダウンロードされた JAR ファイルを Oracle JInitiator インストール・ディレクトリの下の jcache サブディレクトリに格納します。

なぜ、jcache ディレクトリには、キャッシュ書込みされた JAR ファイルの見慣れない名前が含まれているのですか？

Web サーバー上の各 JAR は URL（URL = codebase + JAR filename）によって識別されるので、Oracle JInitiator のキャッシュ書込みメカニズムではこれを使用して、JAR ファイルを一意に識別します。Windows オペレーティング・システムでは、完全な URL は有効なファイル名ではないので、Oracle JInitiator は単純なハッシング・アルゴリズムを使用して、これを有効なファイル名に変更してから、格納 JAR ファイル名として使用します。JAR ファイルのリクエストが行われた場合、Oracle JInitiator は完全な URL に対してハッシング・アルゴリズムを実行し、キャッシュ内に結果として生じるファイル名が存在するかどうかをチェックします。

JAR ファイル・キャッシュ書込みはどのようにサーバー・ロード・バランシングを処理するのですか？

前述したように、キャッシュ内の JAR ファイルは、取り出された URL に基づいて識別されます。したがって、異なるサーバー上にある同じ JAR ファイルは、各サーバーからダウンロードされます。セキュリティおよびアプリケーションの整合性を保証するために、これは意識的に行われます。JAR ファイルが名前のみを使用してキャッシュ書込みされている場合、不正なアプリケーションが他のアプリケーションの JAR ファイルを置換することがあります。オリジナルのアプリケーションが実行されている場合、Java クラス・ファイルは異なります。また、JAR ファイルは一意の名前を持つことが保証されていないので、JAR ファイルが衝突することがあります。これは、2 つの異なるアプリケーションが同じ JAR ファイル名を使用しており、JAR ファイルからの異なるクラス・ファイルが必要な場合に起こります。

Forms Developer アプリケーションを実行するたびに、キャッシュ書込みされた JAR ファイルのタイムスタンプが更新されているようです。これは正常ですか？そのたびに、ファイルがダウンロードされているということですか？

いいえ。Oracle JInitiator では、構成変更可能なキャッシュ最大サイズがサポートされています。キャッシュ書込みされた JAR ファイルが使用されるたびに、Oracle JInitiator はキャッシュ書込みされたファイルが前回使用された日時を示すために、タイムスタンプを更新します。

キャッシュ・サイズが、最大キャッシュ・サイズを維持するためにファイル削除が必要になる程度まで大きくなった場合、Oracle JInitiator はキャッシュ・ファイルのタイムスタンプを使用して、最も以前に使用されたファイルを判断し、そのファイルを削除します。

キャッシュが正しく機能していること、および JAR ファイルが常にダウンロードされているわけではないことがどうしたらわかるのですか？

Oracle JInitiator が必要なファイルをダウンロードする必要がある場合、Forms Developer アプリケーションを実行するように構成された Web サーバーを使用します。最新の Web サーバーでは、ダウンロードされたファイル、ダウンロードしたユーザー、およびダウンロード

日時を追跡できるログ・ファイルの使用がサポートされています。Web サーバーのログ・ファイルでは、発生したトランザクションを記述するために標準フォーマットが使用されています。このログ・フォーマットには、リクエスト項目名およびリクエストの結果が含まれています。リクエストの結果は、一連の標準 HTTP ステータス・コードを使用して示されます。

JAR ファイルがクライアントにダウンロードされている場合、ログ・ファイルには、要求された JAR ファイル名および HTTP ステータス・コード 200 が含まれます。JAR ファイルのタイムスタンプがキャッシュ書込みされたファイルのタイムスタンプ以前のものであるため、JAR ファイルがダウンロードされていない場合、ログ・ファイルには、要求された JAR ファイル名および HTTP ステータス・コード 304 が含まれます。

次の例では、キャッシュ内の JAR ファイルが現行のものではないので、Web サーバーからダウンロードする必要がある場合、標準 NCSA ログ・フォーマットを使用してログ・ファイル内で作成されたエントリを示します。

```
ferret.us.oracle.com - - [19/Feb/1999:17:40:12 -0800]
"GET /forms_java/f60all.jar  HTTP/1.0" 200 -
```

次の例では、キャッシュ内の JAR ファイルが現行のものであるので、Web サーバーからダウンロードしない場合、標準 NCSA ログ・フォーマットを使用してログ・ファイル内で作成されたエントリを示します。

```
ferret.us.oracle.com - - [19/Feb/1999:17:42:29 -0800]
"GET /forms_java/f60all.jar  HTTP/1.0" 304 -
```

JAR ファイルのダウンロード時、jcache ディレクトリに .JCX ファイルが作成されるようです。このファイルは何ですか？

JAR ファイルのダウンロード時、JAR ファイルのテンポラリ・コピーがファイル・システムに書き込まれます。このテンポラリ・コピーは、.jcx というファイル拡張子で識別されます。ダウンロードが正常に終了した場合、.jcx ファイルは .jc ファイルになります。ダウンロードまたは接続が中断された場合、操作は完了しないで、テンポラリ・ファイルの拡張子は .jcx のままです。Oracle JInitiator は .jcx という拡張子のファイルは無効なので、ロードしません。

キャッシュ書込みが正常に機能することを確認していますが、アプリケーションの起動に時間がかかりすぎます。これはなぜですか？

Oracle JInitiator によって提供される JAR ファイル・キャッシュ書込みは、システム上の Java のスピードを上げるような技法を使用しません。実行することは、各アプリケーションの起動に必要な JAR ファイルをダウンロードする時間を節約することです。JAR ファイルの Zip 解凍、包含クラスのメモリーへのロード、および改ざんされていないことを保証するための認証操作によって、起動時間は非常に長くなります。実際に、超高速ネットワークでは、JAR ファイルのダウンロードに要する時間は、Java クラスのメモリーへのロードおよび認証に要する時間よりも短くなります。つまり、キャッシュ書込みでは、アプリケーション起動に要する時間をほとんど節約できません。低速ネットワークでは、JAR ファイルのダウンロードに要する時間は、起動時間において比例して長くなります。したがって、JAR ファイルのキャッシュ書込みはさらに重要になります。

B.5 AppletViewer

この項では、AppletViewer について説明します。AppletViewer は JDK コンポーネントであり、クライアント・マシンが Forms Services 上で実行されるアプリケーションを表示するために使用する Oracle サポート製品です。アップグレード・バージョンは、Forms Developer Web サイトからダウンロードできます。

オラクル社は AppletViewer で使用する f60all.jar ファイルを提供します。

注意： AppletViewer は Windows 95 および Windows NT 4.0 でのみサポートされています。

注意： クライアントのブラウザで AppletViewer を使用する場合は、HTTPS 接続モードはサポートされません。

B.5.1 AppletViewer でのアプリケーション実行

AppletViewer でアプリケーションを実行するには、次のステップを実行する必要があります。

- AppletViewer を使用してアプリケーションを実行する準備を行います。
- clientBrowser パラメータをベース HTML ファイルに追加します。
- clientBrowser パラメータを設定します。

AppletViewer でアプリケーションを実行している場合、AppletViewer は URL の表示リクエスト（たとえば、web.showDocument および RUN_PRODUCT）を無視します。この場合、この章の B.5.2.1 項「シグネチャを登録することによる Forms アプレットの信頼」で後述する方法で、Forms アプレットを信頼するためのプロセスを実行する必要があります。

B.5.1.1 AppletViewer を使用したアプリケーション実行準備

AppletViewer 内でアプリケーション実行の準備を行うには、AppletViewer をダウンロード可能にして、ユーザーに AppletViewer をクライアント・マシンにインストールする必要があることを通知します。次のことを実行します。

1. JDK_DOWNLOAD.HTM をカスタマイズします。

   JDK_DOWNLOAD.HTM は、ユーザーが AppletViewer をダウンロードできるテンプレート HTML ファイルです。

2. JDK.EXE を Web サーバーにコピーします。

   JDK.EXE は、JDK_DOWNLOAD.HTM 内で指定された位置にコピーする必要があります。

3. JDK_DOWNLOAD.HTM を Web サーバーにコピーします。

   JDK_DOWNLOAD.HTM は、JDK_DOWNLOAD.HTM 内で指定された位置にコピーする必要があります。

B.5.1.2 clientBrowser パラメータのベース HTML ファイルへの追加

clientBrowser パラメータを使用するには、指定されたアプリケーションを実行するシステム・コールを発行するためのセキュリティ権限を持つ必要があります。通常、Java クラス・ファイルをロードする場合、Forms アプレットは信頼されないので、このようなシステム・コールは発行できません。ただし、Forms アプレットが信頼される場合、これらのコールを発行できます。次のいずれかを満たす場合、Forms アプレットは信頼されていると判断されます。

- B.5.2.1 項「シグネチャを登録することによる Forms アプレットの信頼」で記述したように、Forms アプレット・シグネチャがクライアント・マシンに「登録されている」。
- Forms Java クラス・ファイルがクライアント・システムにローカルにインストールされており、B.5.2.2 項「Forms Java クラス・ファイルをローカルにインストールすることによる Forms アプレットの信頼」で記述したように CLASSPATH 環境変数が設定されている。

これらの HTML ファイルの例では、マシンにシグネチャを登録することで Forms アプレットを信頼していると仮定しています。また、Forms Java クラス・ファイルをローカルにインストールすることで Forms アプレットを信頼した場合、F60ALL.JAR ファイルをダウンロードする必要はありません。したがって、ARCHIVE="/.../f60all.jar" アプレット・タグを HTML ファイルから削除します。

B.5.1.3 clientBrowser パラメータの設定

clientBrowser パラメータを設定するには、次のいずれかを実行します。

- clientBrowser パラメータを HTML ファイルに追加します。
- clientBrowser パラメータを HTML ファイルに追加し、各クライアントに JDK_SETUP.BAT ファイルを変更させます。

clientBrowser パラメータを HTML ファイルに追加します。

このオプションでは、ブラウザの表示パスが HTML ファイルにハードコードされているため、すべてのクライアントがブラウザ実行ファイルを同じ表示ディレクトリにインストールしていると仮定しています。例 :

```
<APPLET CODEBASE="/forms60code/"
     CODE="oracle.forms.engine.Main"
     ARCHIVE="/forms60code/f60all.jar"
     HEIGHT=480
     WIDTH=640>
     <PARAM NAME="serverArgs" VALUE="module=start.fmx userid=scott/tiger">
     <PARAM NAME="clientBrowser"
     VALUE="c:¥programfiles¥netscape¥communicator¥program¥netscape.exe">
</APPLET>
```

clientBrowser パラメータを HTML ファイルに追加し、各クライアントに JDK_SETUP.BAT ファイルを変更させます。

このオプションは、クライアントがブラウザ実行ファイルを異なる表示ディレクトリにインストールしている可能性がある場合に最適です。ただし、すべてのクライアントが同じブラウザを使用していると仮定しています。HTML ファイルの一例を次に示します。

```
<APPLET CODEBASE="/forms60code/"
     CODE="oracle.forms.engine.Main"
     ARCHIVE="/forms60code/f60all.jar"
     HEIGHT=480
     WIDTH=640>
     <PARAM NAME="serverArgs" VALUE="module=start.fmx userid=scott/tiger">
     <PARAM NAME="clientBrowser" VALUE="netscape">
</APPLET>
```

次に、JDK_SETUP.BAT の一例を示します。

```
SET CLASSPATH=C:¥ORANT¥JDK1.1¥JDK¥LIB¥CLASSES.ZIP
PATH C:¥PROGRAM FILES¥NETSCAPE¥COMMUNICATOR¥PROGRAM;
C:¥ORANT¥JDK1.1¥JDK¥BIN;%PATH%
```

B.5.2 Forms アプレット・シグネチャの登録

クライアント・マシンはシグネチャを使用して、有効かつ信頼されたエンティティ（署名者）からファイルがダウンロードされていることを確認できます。これにより、クライアント・マシンは不正行為または Java アーカイブ（JAR）ファイルの誤動作から自身を保護できます。クライアントが JAR ファイルを妥当性チェックするには、クライアント・マシンにそのファイルのシグネチャが登録されていることが必要です。Javakey は、JAR ファイルのデジタル・シグネチャを生成するための Sun Microsystems のコマンドライン・ツールです。

Forms アプレット自体は署名済 JAR ファイルです。Forms アプレット・シグネチャを登録するには 2 つのオプションがあります。次のいずれかを選択します。

- 弊社が提供する Forms アプレット・シグネチャを使用して、シグネチャをクライアント・マシンに登録します。
- ユーザー固有のシグネチャを使用して Forms アプレットを再署名し、そのシグネチャをクライアント・マシンに登録します。このメソッドを選択する場合、JAR ファイルの作成および署名の詳細は http://java.sun.com/security/usingJavakey.html を参照してください。

B.5.2.1 シグネチャを登録することによる Forms アプレットの信頼

シグネチャを登録することにより、Forms アプレットを信頼するには、次の手順を実行します。

1. Forms Developer 認証をクライアント・マシン上の ¥<ORACLE_HOME>¥6iserver¥FORMS60¥JAVA にコピーします。

   この認証は Dev.x509 という名前のファイルです。このファイルは、サーバー上の ¥<ORACLE_HOME>¥6iserver¥FORMS60¥JAVA にあります。

2. DOS コマンド・プロンプトをオープンし、¥<ORACLE_HOME>¥6iserver¥FORMS60¥JAVA にナビゲートします。
3. 次のように入力します。

   javakey -c Developer true

   このコマンドは、認証提供者の正確な名前を使用して、クライアントの認証データベース上で AppletViewer の信頼された認証を作成します。

4. [Enter] キーを押します。
5. javakey -ic Developer Dev.x509 と入力します。

   このコマンドは、Dev.x509 認証をクライアントの JDK 認証データベースにインポートし、この認証をステップ 3 で作成された信頼された認証に関連付けます。

6. [Enter] キーを押します。

B.5.2.2 Forms Java クラス・ファイルをローカルにインストールすることによる Forms アプレットの信頼

Java クラス・ファイルをローカルにインストールすることにより Forms アプレットを信頼するには、次の手順を実行します。

1. ¥<ORACLE_HOME>¥6iserver¥FORMS60¥JAVA ディレクトリをクライアント・マシン上の新規ディレクトリにコピーします。

   このディレクトリは正確にコピーし、ディレクトリ構造は絶対に変更しないでください。

2. 次のようにして、<ORACLE_HOME> ディレクトリの JDK_SETUP.BAT を変更します。
   a. テキスト・エディタで JDK_SETUP.BAT をオープンします。
   b. 新規ディレクトリを参照するために、CLASSPATH 環境変数を変更します。
   c. 変更を JDK_SETUP.BAT に保存します。

B.5.3 ユーザーへの指示

アプリケーションを AppletViewer 内から実行するには、次のステップを実行します。

- AppletViewer をインストールします。
- AppletViewer を実行します。
- Web ブラウザを AppletViewer 内から起動します。

B.5.3.1 AppletViewer のインストール

AppletViewer をインストールするには、Oracle Installer を使用して JDK AppletViewer をインストールします。

1. 起動している Windows アプリケーションをすべて終了します。
2. タスクバーから、**「スタート」→「ファイル名を指定して実行」**を選択します。
3. 「ファイル名を指定して実行」ダイアログに次のように入力します（「D:」は実際の CD-ROM ドライブの文字に置き換えます）。

   ```
   D:¥setup.exe
   ```

   続いて「**OK**」をクリックします。
4. 「Oracle インストール設定」ダイアログ・ボックスで、会社名および <ORACLE_HOME> ディレクトリのデフォルト値をチェックします。
5. 「**Oracle Forms Services**」をクリックします。
6. 「**カスタム**」をクリックします。
7. 「使用可能な製品」リストから、「**JDK AppletViewer**」を選択します。
8. **「インストール」**をクリックします。

B.5.3.2 AppletViewer の実行

AppletViewer を実行するには、次の手順を実行します。

1. DOS コマンドで、AppletViewer 実行ファイル（appletviewer.exe）に移動します。
2. ホスト名、HTML ファイル仮想ディレクトリおよび HTML ファイルを指定して、AppletViewer 実行ファイルを実行します。

   たとえば、次のように入力します。
   appletviewer http://myhost.com/web_html/start.html
3. [Enter] キーを押します。

B.5.3.3 AppletViewer 内からの Web ブラウザ起動

Web ブラウザを AppletViewer 内から起動するには、次の手順を実行します。

1. 次の 2 つのメソッドのいずれかを使用して、Forms を信頼します。
   - Forms アプレット・シグネチャを登録します。
   - Forms Java クラス・ファイルをローカルにインストールします。
2. clientBrowser パラメータをベース HTML ファイルに追加します。

C

Java Importer

Oracle Forms Developer 6*i* には Java Importer が含まれています。この付録には、Java Importer の説明と使用方法ならびに結果ファイルに関する、次の項が含まれています。

C.1 概要

C.1.1 Java のインポートとアプリケーションの作成

フォーム開発者は Java Importer を使用して、Java クラスにアクセスする PL/SQL パッケージを生成できます。さらに、生成された PL/SQL を Forms アプリケーション内で使用してプログラムを作成できます。Java Importer で生成された PL/SQL は強力で、元の Java クラスのコンストラクタ、メソッドおよびフィールドをサポートします。

単に静的メソッドを PL/SQL ファンクションおよびプロシージャにマップするだけでなく、Java Importer は、型のマッピングと配列オブジェクトとともに、永続 Java オブジェクトもサポートします。

新しい ORA_JAVA パッケージとそのビルトインを使用して、フォーム開発者は、生成された PL/SQL を介して、インポートした Java に簡単にアクセスできます。内部的には、生成された PL/SQL パッケージでは、Java ネイティブ・インタフェース（JNI）標準および PL/SQL と Java 間のブリッジとして機能する内部 JNI パッケージが使用されます。

C.1.2 インポートした Java を用いたアプリケーションの実行

インポートした Java は中間層で動作します。対応する生成された PL/SQL パッケージの Java クラスおよび Java メソッドへのコールは、Forms Services 上の専用 Java 仮想マシン（JVM）で実行されます。専用 JVM は、インポートした Java をコールするために生成 PL/SQL パッケージを使用する、Forms Services のアプリケーション・インスタンスごとに作成されます。

C.2 コンポーネント

Java Importer の機能には、次のものがあります。

- Java Importer ツール

  Importer ツールは Form Builder から実行し、Java クラスをインポートします。また、インポートした Java への PL/SQL アクセスを提供する PL/SQL パッケージを生成します。Importer には、PL/SQL パッケージを生成するための様々なユーザー・オプションが用意されています。

- ORA_JAVA パッケージを含む、ランフォーム API

  ORA_JAVA パッケージは、エラー処理、配列および永続性のサポートを提供する支援パッケージです。（Java メソッドをコールするために使用する内部 JNI パッケージもありますが、このパッケージの内容を知る必要はありません。また、直接コールする必要もありません。）

C.3 インストール要件

Java Importer は、Forms 6*i* Patch2 とともにインストールされます。インストールする際、Java Importer（importer.java）を含む JAR ファイルは CLASSPATH に格納する必要があります。ファイル・システム内の importer.java の場所は、デフォルトでは ORACLE_HOME/TOOLS/COMMON60/JAVA/importer.jar です。

ただし、Importer の操作を可能とするためには、Builder 用マシンとサーバー・マシンの両方に JDK または JRE 1.2 以降をインストールしておく必要があります。

C.3.1 インポートした Java の要件

Java Importer ツールを使用して任意の Java クラスをインポートするには、Java クラスがシステムの CLASSPATH に存在しなければなりません。

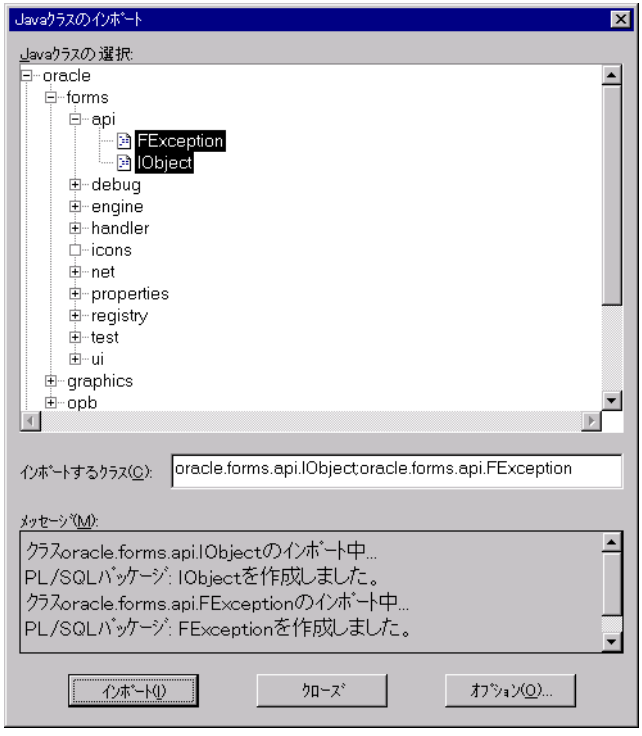
C.4 Java のインポート

C.4.1 Java Importer ツールの使用方法

Java Importer ツールには、複数の Java クラスを容易にアクセスおよびインポートできる、便利な Java クラス・ブラウザが用意されています。選択およびインポートされたそれぞれの Java クラスに対して、オブジェクト・ナビゲータ内のオープンされている現行のモジュールに、ツールによって PL/SQL パッケージが生成されます。

C.4.2「Java クラスのインポート」ダイアログ・ボックスの表示

「Java クラスのインポート」ダイアログ・ボックスを表示するには、メイン・メニューから**「プログラム」→「Java クラスのインポート ...」**を選択します。

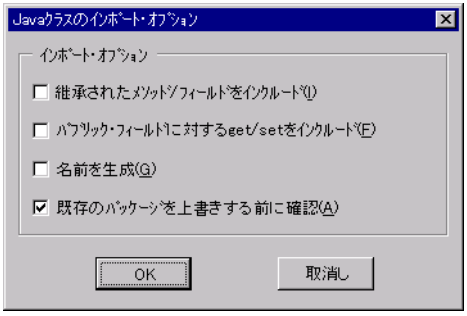


このダイアログ・ボックスには、次のコンポーネントとボタンがあります。

Java クラスの選択（ブラウザ）	Class ブラウザでは、現行の CLASSPATH 環境変数で定義されている順序に従って、すべての Java クラス・ファイルが階層型に表示されます。 （リーフ・ノードで表される）クラス・ファイルは、（フォルダ・ノードで表される）パッケージ内に編成されて表示されます。 フォルダの内容を表示する場合はフォルダを拡張し、内容を隠す場合はフォルダを縮小します。 1 つ以上の Java クラスを選択できます（Windows で複数選択するには、[SHIFT] キーを押しながらクリックします）。
インポートするクラス	このテキスト・ボックスには、選択したクラス・ファイルの完全修飾クラス名が表示されます。複数のクラス名の場合、各クラス名はセミコロン（;）で区切られます。 このフィールドには完全修飾クラス名を入力することもできます。大文字と小文字が区別されるため、たとえば java.lang.String のように入力します。
メッセージ	表示専用のパネルで、インポート・プロセスの状態が表示されます。
インポート	このボタンを押すと、インポート・プロセスが開始されます。 このボタンは、Class ブラウザで 1 つ以上のクラス・ファイルを選択した場合、またはテキスト・ボックスに完全修飾クラス名を入力した場合に有効となります。
オプション ...	「Java クラスのインポート・オプション」ダイアログ・ボックスが表示されます。C.4.3 項「インポート用のオプションの指定」を参照してください。
クローズ	このダイアログ・ボックスをクローズします。

C.4.3 インポート用のオプションの指定

「Java クラスのインポート・オプション」ダイアログ・ボックスを表示するには、「Java クラスのインポート」ダイアログ・ボックスで「**オプション ...**」をクリックします。



オプションは次のとおりです。

- **継承されたメソッド / フィールドをインクルード。**デフォルトではチェックが付いていません。継承されたすべてのメソッドおよびフィールドをプロシージャおよびファンクションにマップする場合、このボックスにチェックを付けます。このオプションを指定すると、指定したクラスおよびその親クラスに含まれるすべてのメソッドに対して、ファンクションとプロシージャが生成されます。
- **パブリック・フィールドに対する get/set をインクルード。**デフォルトではチェックが付いていません。パブリックな非定数フィールドまたは非静的フィールドを、値を取得するファンクションおよび値を設定するプロシージャにそれぞれマップする場合、このボックスにチェックを付けます。
- **名前を生成。**デフォルトではチェックが付いていません。チェックを付けない場合、オーバーロードした Java メソッドでは生成された PL/SQL ファンクションおよびプロシージャの名前を一意の PL/SQL シグネチャとして解決できないため、ファンクション名またはプロシージャ名に追加の識別子が付加されます。

  インポートする Java クラスに変更が生じたとしても、すべての変更をとおして同一の永続 PL/SQL ファンクション名、プロシージャ名、変数名および型名を使用する場合、このボックスにチェックを付けます。(注意 : Java クラスの変更を PL/SQL に反映させるには、PL/SQL パッケージを再生成する必要があります。) チェックを付けた場合、すべての Java メソッドに対する生成された PL/SQL ファンクションおよびプロシージャの名前には、永続的で一意な 4 桁の識別子が付加されます。

  永続識別子番号はメソッドのシグネチャに基づいて生成されます。「名前を生成」を選択すると、Java Importer で同じ Java クラスがインポートされるたびに、同じ Java メソッドに対して生成される PL/SQL ファンクションおよびプロシージャには同一の 4 桁の識別子が付加されます。

  標準命名および永続命名の詳細と例は、C.5.1.4.1 項「永続命名とデフォルト命名の違い」を参照してください。

- **既存のパッケージを上書きする前に確認。**デフォルトではチェックが付いています。既存の PL/SQL パッケージを新しく生成されたパッケージで上書きする前に確認のプロンプトを表示しない場合は、チェックを外します。

C.4.4 PL/SQL への Java クラスのインポート

1 つ以上の Java クラス・ファイルをインポートするには、次のステップを実行します。

1. Form Builder で、生成される PL/SQL パッケージを配置するモジュール（フォーム・モジュール、PL/SQL ライブラリまたはメニュー・モジュール）をオープンします。
2. メニューから**「プログラム」→「Java クラスのインポート ...」**を選択して、「Java クラスのインポート」ダイアログ・ボックスを表示します。
3. Class ブラウザで、1 つ以上の Java クラス・ファイルを選択します。ディレクトリ（フォルダ）は選択できません。

   インポートする Java クラス・ファイルが表示されない場合は、そのクラス・ファイルが現行の CLASSPATH にあるかどうかを確認してください。

   注意 : Form Builder のセッション内で CLASSPATH を編集した場合、CLASSPATH の変更を Java Importer に認識させるために、Builder を再起動する必要があります。
4. 必要であれば、インポート・オプションを指定するために「**オプション ...**」をクリックします。詳細は、C.4.3 項「インポート用のオプションの指定」を参照してください。
5. 「**インポート**」をクリックして、インポート・プロセスを開始します。
6. 処理が完了したら、「**OK**」をクリックします。

C.5 インポートした Java でのアプリケーションの作成

C.5.1 生成された PL/SQL の説明

C.5.1.1 何が生成されるか

Java Importer によって、インポートしたクラスごとに 1 つの PL/SQL パッケージが生成されます。生成されるそれぞれの PL/SQL パッケージ内には、Java のパブリック・フィールド、メソッドおよびコンストラクタに対応する PL/SQL ファンクションおよびプロシージャが Java Importer によって生成されます。プライベート・メソッドおよび保護されたメソッドについては、対応する PL/SQL は生成されません。

C.5.1.2 Java はどのようにして PL/SQL にマップされるか

生成された PL/SQL パッケージでは、Java は次のように PL/SQL にマップされます。

Java	PL/SQL	説明
パブリック・コンストラクタ	ファンクション（"new" と呼ばれる）	パブリック・コンストラクタごとに、異なる "new" ファンクションが生成されます。"new" ファンクションは、オブジェクトのインスタンスを戻します。
パブリック・メソッド（戻り値の型を持つ） 例 : public int getAge()	ファンクション	インスタンス・メソッドの場合、クラスのインスタンスは最初の引数としてファンクションに渡され、オブジェクトの識別に使用されます。静的メソッドの場合は、インスタンス引数は必要ありません。
パブリック・メソッド（戻り値の型 void を持つ） public void setAge(int age)	プロシージャ	インスタンス・メソッドの場合、クラスのインスタンスは最初の引数としてプロシージャに渡され、オブジェクトの識別に使用されます。静的メソッドの場合は、インスタンス引数は必要ありません。
パブリックな静的定数フィールド 例 : public static final int RETIRE_AGE=60;	パッケージ変数（適切な型の）	PL/SQL パッケージが最初に参照されるとき、初期化コードによってパッケージ変数の値が設定されます。
パブリック・フィールド（非静的または非定数） 例 : public String gender;	オプション : 「パブリック・フィールドに対する get/set をインクルード」チェックボックスを選択した場合は、次のとおりです。	
	ファンクション、プロシージャ	パブリック・フィールドは、値を取得するファンクションおよび値を設定するプロシージャにマップされます。インスタンス・フィールドの場合、クラスのインスタンスは最初の引数としてファンクションおよびプロシージャに渡されます。静的フィールドの場合は、インスタンス引数は必要ありません。
	「パブリック・フィールドに対する get/set をインクルード」チェックボックスを選択していない場合は、次のとおりです。 PL/SQL は生成されません。	

C.5.1.3 Importer マッピング・オプションとは

次に示す「Java クラスのインポート・オプション」ダイアログ・ボックス内の Java Importer オプションは、生成される PL/SQL パッケージでの Java から PL/SQL へのマッピングに影響を与えます。

- 「継承されたメソッド / フィールドをインクルード」チェックボックスを選択した場合、継承されたメソッドおよびフィールドは、前述したように Importer によって適切なファンクションおよびプロシージャにマップされます。
- 「パブリック・フィールドに対する get/set をインクルード」チェックボックスを選択した場合、パブリックな非定数フィールドまたは非静的フィールドは、Importer によって、値を取得するファンクションおよび値を設定するプロシージャにそれぞれマップされます。このオプションは前述の表で説明されています。

C.5.1.4 PL/SQL の命名はどのように変わるか

生成された PL/SQL の命名規約では、PL/SQL ファンクションまたはプロシージャに、対応する Java プログラム要素と同じ名前が付けられます。しかし次の場合は、PL/SQL の命名は Java プログラム要素名と同一にはなりません。

- Java 名が長すぎる場合（PL/SQL の制限は 30 文字）、30 文字を超える部分は Importer によって切り捨てられます。
- Java 名の最初の 30 文字が一意でない場合、PL/SQL の最大長である 30 文字以内の一意の名前が Importer によって作成されます。
- Java メソッドがオーバーロードしたために、生成された PL/SQL ファンクションおよびプロシージャの名前が一意の PL/SQL シグネチャとして解決できない場合、Importer によってファンクション名またはプロシージャ名に追加の識別子が付加されます。（注意 : これは、インポート・オプションとして「名前を生成」を選択した場合は適用されません。）

  たとえば、methodA と呼ばれる異なる 2 つの Java メソッド（methodA(int) および methodA(java.long.Float)）があり、それぞれの PL/SQL シグネチャが methodA(NUMBER) と methodA(ORA_JAVA.JOBJECT) であるとします。これらは PL/SQL シグネチャが一意なため、PL/SQL パッケージでも同じ名前になります。

  しかし、たとえば methodB と呼ばれる異なる 2 つの Java メソッド（methodB(char) および methodB(byte)）があり、PL/SQL シグネチャがともに methodB(PLS_INTEGER) であるとします。この場合は PL/SQL シグネチャが同一なため、PL/SQL パッケージでは異なる名前が必要となります。したがって、Java クラスの最初の methodB は PL/SQL パッケージでも同じ名前になりますが、2 番目の methodB は methodB_0 となります。
- Java 名が PL/SQL の予約語である場合、生成された PL/SQL 名にアンダースコアを付けて一意とした名前が、Importer によって作成されます。

  たとえば、value という名前の Java メソッドは、PL/SQL パッケージでは value_ という名前で生成されます。

C.5.1.4.1 永続命名とデフォルト命名の違い

「Java クラスのインポート・オプション」ダイアログ・ボックスで「名前を生成」オプションを選択したかどうかにかかわらず、PL/SL パッケージ内のすべてのファンクションおよびプロシージャには、Importer によって一意の名前が付けられます。しかし、デフォルトの名前（「名前を生成」オプションを選択していない場合）と永続名は異なります。

標準命名とは

標準命名を使用すると、ファンクションおよびプロシージャは対応する Java メソッドと同じ名前で生成されます。この場合、ファンクション名およびプロシージャ名は一意になります。Java メソッドがオーバーロードしたために、生成された PL/SQL ファンクションおよびプロシージャの名前が一意の PL/SQL シグネチャとして解決できない場合、Importer によってファンクション名またはプロシージャ名に追加の識別子が付加されます。

例 1

次の Java クラスを起動します。

```
Class myclass
{
    public void p1(int x);
    public void p1(long x);
    public void p1(char x);
}
```

Java Importer を使用して、クラスをインポートします。永続命名を使用しない場合（「名前を生成」を選択していない場合)、このメソッドに対する PL/SQL は次のようになります。

```
-- Method: p1 (C)V
PROCEDURE p1 (
    obj   ORA_JAVA.JOBJECT,
    a0    PLS_INTEGER);

-- Method: p1 (I)V
PROCEDURE p1_1 (
    obj   ORA_JAVA.JOBJECT,
    a0    NUMBER);

-- Method: p1 (J)V
PROCEDURE p1_2 (
    obj   ORA_JAVA.JOBJECT,
    a0    NUMBER);
```

プロシージャは Java クラス・ファイル内での Java メソッドの出現順どおりには生成されていないことに注意してください。前述の PL/SQL プロシージャの順序は、次の順番の Java メソッドに対応します。

```
    public void p1(char x);
    public void p1(int x);
    public void p1(long x);
```

例 2

次の例では、前の例で示した Java クラスを使用しますが、p1(long x) というメソッドが削除されています。次の Java クラスを起動します。

```
Class myclass
{
    public void p1(int x);
    public void p1(char x);
}
```

Java Importer を使用して、クラスをインポートします。永続命名を使用しない場合（「名前を生成」を選択していない場合）、このメソッドに対する PL/SQL は次のようになります。

```
-- Method: p1 (C)V
PROCEDURE p1 (
    obj   ORA_JAVA.JOBJECT,
    a0    PLS_INTEGER);

-- Method: p1 (I)V
PROCEDURE p1 (
    obj   ORA_JAVA.JOBJECT,
    a0    NUMBER);
```

プロシージャが両方とも p1 となっていることに注意してください。それぞれが一意のシグネチャを持っているため、Importer では各メソッドが一意のプロシージャ・シグネチャとして解決され、識別子は付加されません。Java クラスからのメソッドの削除が、生成されたプロシージャ名に影響を与えた点に注意してください。

永続命名とは

永続命名を使用すると、すべてのファンクション名およびプロシージャ名の末尾に、永続的で一意な 4 桁の識別子が付加されます。Java クラスから PL/SQL を生成するときに常に「名前を生成」を選択していれば、Java クラスが変更されたとしても、（付加された識別子を含む）ファンクション名およびプロシージャ名は変わりません。

永続識別子番号はメソッドのシグネチャに基づいて生成されます。「名前を生成」を選択すると、Java Importer で同じ Java クラスがインポートされるたびに、同じ Java メソッドに対して生成される PL/SQL ファンクションおよびプロシージャには同一の 4 桁の識別子が付加されます。

インポートする Java クラスに変更が生じたとしても、すべての変更をとおして同一の永続 PL/SQL ファンクション名、プロシージャ名、変数名および型名を使用する場合は、永続命名を使用することをお薦めします。（注意 : Java クラスの変更を PL/SQL に反映させるには、PL/SQL パッケージを再生成する必要があります。）

例 3

次の Java クラスを起動します。

```
Class myclass
{
    public void p1(int x);
    public void p1(long x);
    public void p1(char x);
}
```

Java Importer を使用して、クラスをインポートします。永続命名を使用する場合（「名前を生成」を選択した場合）、このメソッドに対する PL/SQL は次のようになります。

```
-- Method: p1 (C)V
PROCEDURE p1_7384 (
    obj   ORA_JAVA.JOBJECT,
    a0    PLS_INTEGER);

-- Method: p1 (I)V
PROCEDURE p1_3150 (
    obj   ORA_JAVA.JOBJECT,
    a0    NUMBER);

-- Method: p1 (J)V
PROCEDURE p1_4111 (
    obj   ORA_JAVA.JOBJECT,
    a0    NUMBER);
```

すべてのプロシージャ名に、一意な 4 桁の識別子が付加されていることに注意してください。

例 4

次の例では、前の例で示した Java クラスを使用しますが、p1(long x) というメソッドが削除されています。次の Java クラスを起動します。

```
Class myclass
{
    public void p1(int x);
    public void p1(char x);
}
```

Java Importer を使用して、クラスをインポートします。永続命名を使用する場合（「名前を生成」を選択した場合）、このメソッドに対する PL/SQL は次のようになります。

```
-- Method: p1 (C)V
PROCEDURE p1_7384 (
    obj    ORA_JAVA.JOBJECT,
    a0     PLS_INTEGER);

-- Method: p1 (I)V
PROCEDURE p1_3150 (
    obj    ORA_JAVA.JOBJECT,
    a0     NUMBER);
```

すべてのプロシージャ名に、一意な 4 桁の識別子が付加されていることに注意してください。これらの識別子の番号は、前の例（例 3）で生成された番号と同じです。Java クラスからメソッドを削除しても、識別子の番号、つまりプロシージャの名前には影響がない点に注意してください。

C.5.1.5 PL/SQL を再生成するとどうなるか

同じ Java クラスから PL/SQL パッケージを再生成する場合、いくつかのオプションがあります。次に示す「Java クラスのインポート・オプション」ダイアログ・ボックスの 2 つのオプションが、どのように PL/SQL パッケージを再生成するかに影響を与えます。

- 「既存のパッケージを上書きする前に確認」チェックボックスにチェックを付けた場合、既存のパッケージを上書きするかどうかを確認するプロンプトが Importer によって表示されます。問題がなければ、パッケージ（およびそれに含まれるすべてファンクションとプロシージャ）を上書きするよう選択します。
- 「名前を生成」にチェックを付けた場合と付けない場合で生成された PL/SQL がどのように異なるかは、C.5.1.4.1 項「永続命名とデフォルト命名の違い」を参照してください。注意 :「名前を生成」オプションを、同じ Java クラスから前回 PL/SQL パッケージを生成したときとは異なる設定にすると、前回とは違うプロシージャ名およびファンクション名が付けられます。これは、ファンクション名およびプロシージャ名に付加される識別子が変更される（または付加されない）ためです。

C.5.2 Java データ型

Java Importer がマップする Java データ型と PL/SQL データ型を次の表にリストします。

Java データ型	変換後の PL/SQL	注意
すべての Java 配列	ORA_JAVA.JARRAY	ORA_JAVA.JARRAY は ORA_JAVA.JOBJECT のサブタイプ
boolean	BOOLEAN	
byte	PLS_INTEGER	
char	PLS_INTEGER	
double	NUMBER	
float	NUMBER	
int	NUMBER	
long	NUMBER	
すべての Java オブジェクト	ORA_JAVA.JOBJECT	
short	PLS_INTEGER	
java.lang.String	VARCHAR2	
すべての Java 例外	ORA_JAVA.JEXCEPTION	ORA_JAVA.JEXCEPTION は ORA_JAVA.JOBJECT のサブタイプ

Java 配列および Java オブジェクトは、特別な ORA_JAVA 型である ORA_JAVA.JARRAY および ORA_JAVA.JOBJECT にマップされる点に注意してください。また、Java の int 型と PL/SQL の INT 型ではサポートが異なるため、Java の int 型は PL/SQL の NUMBER 型にマップされます。

C.5.2.1 PL/SQL パッケージでの Java データ型に関する情報

生成された PL/SQL パッケージ仕様には、インポートした Java のデータ型とシグネチャに関する情報を提供するコメントが含まれます。コメント内のデータ型とシグネチャに関する情報は JNI ベースです。これらのコメントは、生成された項目の PL/SQL シグネチャの直前に置かれます。

たとえば、次のような Java メソッドがあるとします。

```
public void p1(char x);
public void p1(int x);
public void p1(long x);
```

この Java メソッドは、生成された PL/SQL パッケージでは次のようにマップされます。

```
-- Method: p1 (C)V
PROCEDURE p1 (
    obj   ORA_JAVA.JOBJECT,
    a0    PLS_INTEGER);

-- Method: p1 (I)V
PROCEDURE p1_1 (
    obj   ORA_JAVA.JOBJECT,
    a0    NUMBER);

-- Method: p1 (J)V
PROCEDURE p1_2 (
    obj   ORA_JAVA.JOBJECT,
    a0    NUMBER);
```

生成された各プロシージャの上にあるコメントに注意してください。

`-- Method:p1 (C)V`	元の Java は p1 という名前のメソッドであり、引数として char 型を使用することを示します。
`-- Method:p1 (I)V`	元の Java は p1 という名前のメソッドであり、引数として int 型を使用することを示します。
`-- Method:p1 (J)V`	元の Java は p1 という名前のメソッドであり、引数として long 型を使用することを示します。

C.5.2.2 配列

ORA_JAVA パッケージには、配列を作成するためのルーチンや配列要素の値を取得および設定するためのルーチンを提供するビルトインが含まれています。

次の ORA_JAVA ビルトインを参考にしてください。

ORA_JAVA.NEW_<java_type>_ARRAY
ORA_JAVA.GET_<java_type>_ARRAY_ELEMENT
ORA_JAVA.SET_<java_type>_ARRAY_ELEMENT
ORA_JAVA.GET_ARRAY_LENGTH

<java_type> は任意のオブジェクト・タイプまたは Java スカラーを示します。

C.5.3 永続性

ユーザーが PL/SQL 内に Java オブジェクトを作成する場合（コンストラクタをコールするか、ORA_JAVA ビルトインを使用してオブジェクトを作成）、デフォルトでは、そのオブジェクトを作成した PL/SQL トリガーの存続期間がオブジェクトの永続性となります。

ただし、ORA_JAVA パッケージ・ビルトインを使用すると、ユーザーはグローバル参照を使用して永続オブジェクトを管理できます。

C.5.3.1 グローバル参照

オブジェクトを作成した PL/SQL トリガーの存続期間を超えてオブジェクトを永続させるには、ORA_JAVA.NEW_GLOBAL_REFERENCE を使用します。オブジェクトを破棄するには、ORA_JAVA.DELETE_GLOBAL_REFERENCE を使用します。

次の ORA_JAVA ビルトインを参考にしてください。

ORA_JAVA.NEW_GLOBAL_REFERENCE
ORA_JAVA.DELETE_GLOBAL_REFERENCE

参照が完了したら、作成した ORA_JAVA.NEW_GLOBAL_REFERENCE ごとに ORA_JAVA.DELETE_GLOBAL_REFERENCE をコールします。グローバル参照はガーベジ・コレクションでは削除されません。明示的に削除してください。

注意 : グローバル参照の保存に使用する変数は、パッケージ変数として定義する必要があります。

C.5.4 エラー処理

ORA_JAVA パッケージには、PL/SQL が Java をコールしている間に例外やエラーが発生していないかをチェックするためのルーチンを提供するビルトインが含まれています。

PL/SQL が Java をコールするときにエラーが発生すると、次の PL/SQL 例外のいずれかが呼び出されます。

ORA_JAVA.JAVA_ERROR
ORA_JAVA.JAVA_EXCEPTION

C.5.4.1 エラー

PL/SQL が Java メソッドをコールするときに、エラーが発生する場合があります。これは Java メソッドによってスローされる例外ではなく、メソッドをコールしようとしたことに起因するエラー条件です。発生する可能性があるエラーには、次のものがあります。

- Java VM を初期化できません。
- 引数 *n* に null は指定できません。
- 指定された配列のインデックスは範囲外です。

- 指定された配列サイズは無効です。
- 引数 *n* のために範囲外変換エラーが発生しました。
- 引数 *n* のために無効な整数変換エラーが発生しました。
- 引数 *n* のオブジェクト・タイプが無効です。

ORA_JAVA.JAVA_ERROR 例外が呼び出された場合は、ORA_JAVA.LAST_ERROR ビルトインを使用して、エラー・テキストを取得してください。このコンテキストでは、エラーは Forms Services イベントです。

次の ORA_JAVA ビルトインを参考にしてください。

ORA_JAVA.LAST_ERROR
ORA_JAVA.CLEAR_ERROR

C.5.4.2 例外

例外とは、標準の Java 例外です。ORA_JAVA.EXCEPTION_THROWN 例外が呼び出された場合、これはコールされたメソッドから Java 例外がスローされたことを示します。

ORA_JAVA.EXCEPTION_THROWN 例外が呼び出された場合は、ORA_JAVA.LAST_EXCEPTION ビルトインを使用して、例外を取得してください。このコンテキストでは、例外は、Forms Services が検出でき、アプリケーションに伝えることができる Java イベントです。

次の ORA_JAVA ビルトインを参考にしてください。

ORA_JAVA.LAST_EXCEPTION
ORA_JAVA.CLEAR_EXCEPTION

C.6 制限事項

C.6.1 Java および PL/SQL の問題点と要件

- java.lang.String オブジェクトは、サイズが 32KB に制限されている VARCHAR2 にマップされます。
- Forms アプリケーションでは、生成された PL/SQL パッケージを介して無効な Java オブジェクトを参照することは許されません。

C.6.2 Forms Services 内の Java

- Java Importer によってインポートされ、Forms アプリケーションから参照される Java は、アプリケーションの中間層に存在しなければなりません。
- Forms Services にあるインポートされた Java が PL/SQL からコールされると、起動されたランタイム・プロセスごとに別々の Java 仮想マシン（JVM）が起動されます。各 JVM で使用されるメモリーの量には、JVM プロセスのオーバーヘッドに加えて、Java アプリケーションの実行と Java オブジェクトの記憶域のために使用されるメモリー量が含まれます。

C.6.3 Builder の CLASSPATH の更新

- Java クラスが一度 Form Builder のセッションにロードされると、クラスに対する変更はクラスの実行には反映されません。変更を反映させた上でクラスを実行するには、Builder を再起動する必要があります。
- Form Builder のセッション内で Java クラスをインポートし、その後クラスに変更を加えた場合、Builder を再起動してから、変更した Java クラスをインポートする必要があります。前回 Java クラスをインポートしたときと同じ Form Builder のセッション内でそのクラスに加えられた変更は、Java Importer によって認識されません。
- Form Builder のセッション内で CLASSPATH を編集した場合、CLASSPATH の変更を Java Importer に認識させるために、Builder を再起動する必要があります。

  注意 : Builder のセッション内で CLASSPATH に追加したクラスは、「Java クラスのインポート」ダイアログ・ボックスにリストされていなくてもインポートできます。CLASSPATH にはあるものの、「Java クラスのインポート」ダイアログ・ボックスにはリストされないクラスをインポートするには、「インポートするクラス」テキスト・フィールドに完全修飾クラス名を入力してください。

C.6.4 Builder の制限事項

- Java Importer とその機能は、Form Builder の Web プレビュー・モードとともには使用できません。

C.7 ORA_JAVA ビルトイン・リファレンス

NEW_GLOBAL_REF ビルトイン

DELETE_GLOBAL_REF ビルトイン

LAST_EXCEPTION ビルトイン

CLEAR_EXCEPTION ビルトイン

LAST_ERROR ビルトイン

CLEAR_ERROR ビルトイン

次のものについては、NEW_<java_type>_ARRAY ビルトインを参照してください。

- NEW_OBJECT_ARRAY ビルトイン
- NEW_BYTE_ARRAY ビルトイン
- NEW_CHAR_ARRAY ビルトイン
- NEW_SHORT_ARRAY ビルトイン
- NEW_INT_ARRAY ビルトイン
- NEW_LONG_ARRAY ビルトイン
- NEW_FLOAT_ARRAY ビルトイン
- NEW_DOUBLE_ARRAY ビルトイン
- NEW_STRING_ARRAY ビルトイン
- NEW_BOOLEAN_ARRAY ビルトイン

IS_NULL ビルトイン

GET_ARRAY_LENGTH ビルトイン

次のものについては、GET_<java_type>_ARRAY_ELEMENT ビルトインを参照してください。

- GET_OBJECT_ARRAY_ELEMENT ビルトイン
- GET_BYTE_ARRAY_ELEMENT
- GET_CHAR_ARRAY_ELEMENT
- GET_SHORT_ARRAY_ELEMENT
- GET_INT_ARRAY_ELEMENT
- GET_LONG_ARRAY_ELEMENT

GET_FLOAT_ARRAY_ELEMENT

GET_DOUBLE_ARRAY_ELEMENT

GET_STRING_ARRAY_ELEMENT

GET_BOOLEAN_ARRAY_ELEMENT

次のものについては、SET_<java_type>_ARRAY_ELEMENT ビルトインを参照してください。

SET_OBJECT_ARRAY_ELEMENT

SET_BYTE_ARRAY_ELEMENT

SET_CHAR_ARRAY_ELEMENT

SET_SHORT_ARRAY_ELEMENT

SET_INT_ARRAY_ELEMENT

SET_LONG_ARRAY_ELEMENT

SET_FLOAT_ARRAY_ELEMENT

SET_DOUBLE_ARRAY_ELEMENT

SET_STRING_ARRAY_ELEMENT

SET_BOOLEAN_ARRAY_ELEMENT

C.7.1 NEW_GLOBAL_REF ビルトイン

説明

ORA_JAVA.JOBJECT 型のオブジェクトを参照するためにグローバル変数として使用できる参照ハンドルを返します。

構文

```
FUNCTION NEW_GLOBAL_REF
    (obj IN ORA_JAVA.JOBJECT)
    RETURN ORA_JAVA.JOBJECT;
```

パラメータ

obj グローバル参照の作成対象となる Java クラスの有効なインスタンス。実際のパラメータには、ORA_JAVA.JOBJECT 型の任意のオブジェクトを指定できます。

戻り値

PL/SQL データ型が ORA_JAVA.JOBJECT のオブジェクト。

使用上の注意

このビルトインは、オブジェクトを作成した PL/SQL トリガーの存続期間を超えてオブジェクトを永続させる場合に使用してください。これは永続 Java オブジェクトを作成する唯一のメカニズムです。PL/SQL トリガーよりも有効範囲が広いオブジェクトに対して使用してください。

グローバル参照の作成対象となるオブジェクトは、有効でなければなりません。

作成された参照ハンドルは、ORA_JAVA.DELETE_GLOBAL_REF によって明示的に削除されるまでアクティブなままとなります。これは、多くのトリガー・ポイントで同じオブジェクトを使用できることを意味します。

例

```
PROCEDURE foo IS
    obj ORA_JAVA.JOBJECT;
    ...
BEGIN
    obj := myclass.new;
    mypkg.instobj := ORA_JAVA.NEW_GLOBAL_REF(obj);
    ...
END;
```

C.7.2 DELETE_GLOBAL_REF ビルトイン

説明

ORA_JAVA.NEW_GLOBAL_REF によって作成されたグローバル参照オブジェクトを削除します。

構文

```
PROCEDURE DELETE_GLOBAL_REF (obj IN ORA_JAVA.JOBJECT);
```

パラメータ

obj　　削除対象の有効なグローバル参照オブジェクト。

使用上の注意

不要なグローバル参照オブジェクトを削除して、そのオブジェクトに割り当てられたメモリーを解放する場合、このビルトインを使用する必要があります。

例

```
PROCEDURE foo IS
    obj ORA_JAVA.JOBJECT;
    ...
BEGIN
    obj := myclass.new;
    mypkg.instobj := ORA_JAVA.NEW_GLOBAL_REF(obj);
    ...
END;
...
ORA_JAVA.DELETE_GLOBAL_REF (mypkg.instobj);
```

C.7.3 LAST_EXCEPTION ビルトイン

説明

PL/SQL から Java をコールしたときに発生した、最後の Java 例外を返します。呼び出された PL/SQL 例外が ORA_JAVA.EXCEPTION_THROWN の場合に使用します。

構文

```
FUNCTION LAST_EXCEPTION RETURN ORA_JAVA.JEXCEPTION;
```

パラメータ

なし

戻り値

データ型が ORA_JAVA.JEXCEPTION のオブジェクト。これは ORA_JAVA.JOBJECT のサブタイプです。

使用上の注意

PL/SQL ブロックで Java メソッドへのコールを発行するときは、NULL ポインタなどの ORA_JAVA.EXCEPTION_THROWN 型の PL/SQL 例外を処理するために、コールするブロックの例外処理部でこのビルトインを使用することをお薦めします。
ORA_JAVA.EXCEPTION_THROWN がスローされた場合、これはコールされている Java メソッド内から例外がスローされたことを示します。

ORA_JAVA.LAST_ERROR も参照してください。

例

```
/* This example assumes you have imported the
** java.lang.Exception class.
*/
PROCEDURE foo IS
    obj ORA_JAVA.JOBJECT;
    excp ORA_JAVA.JEXCEPTION;
BEGIN
    obj := jfoo.new;
    jfoo.addElement(obj);
EXCEPTION
    WHEN ORA_JAVA.EXCEPTION_THROWN THEN
        excp := ORA_JAVA.LAST_EXCEPTION;
        message('    Java Exception: ' || exception_.toString(excp));
    ...
END;
```

C.7.4 CLEAR_EXCEPTION ビルトイン

説明

ORA_JAVA.LAST_EXCEPTION によって取り出した、最後の例外を削除します。

構文

```
PROCEDURE CLEAR_EXCEPTION;
```

パラメータ

なし

例

```
/* This example assumes you have imported the
** java.lang.Exception class.
*/
PROCEDURE foo IS
    obj ORA_JAVA.JOBJECT;
    excp ORA_JAVA.JOBJECT;
BEGIN
    obj := jfoo.new;
    jfoo.addElement(obj);
    ...
EXCEPTION
    WHEN ORA_JAVA.EXCEPTION_THROWN THEN
        excp := ORA_JAVA.LAST_EXCEPTION;
        message('    Java Exception: ' || exception_.toString(excp));
        ORA_JAVA.CLEAR_EXCEPTION;
END;
```

C.7.5 LAST_ERROR ビルトイン

説明

PL/SQL から Java をコールしたときに発生した、最後の PL/SQL 例外のエラー・テキストを返します。呼び出された PL/SQL 例外が ORA_JAVA.JAVA_ERROR の場合に使用します。

構文

```
FUNCTION LAST_ERROR RETURN VARCHAR2;
```

パラメータ

なし

戻り値

VARCHAR2

使用上の注意

PL/SQL ブロックで Java メソッドへのコールを発行するときは、ORA_JAVA.JAVA_ERROR 型の PL/SQL 例外を処理するために、コールするブロックの例外処理部でこのビルトインを使用することをお薦めします。ORA_JAVA.JAVA_ERROR がスローされた場合、これはコールされている Java メソッド内から例外がスローされたのではないことを示します。

詳細は C.5.4 項「エラー処理」を参照してください。

また、ORA_JAVA.LAST_EXCEPTION も参照してください。

例

```
/*
** Example of an invalid array element error.
*/
PROCEDURE foo IS
    arr   ORA_JAVA.JARRAY;
    n     PLS_INTEGER;
BEGIN
    ...
    arr := ORA_JAVA.NEW_BYTE_ARRAY(5);
    n   := ORA_JAVA.GET_BYTE_ARRAY_ELEMENT(arr, 5);
    ...
EXCEPTION
    WHEN ORA_JAVA.JAVA_ERROR THEN
        message('  Alert: ' || ORA_JAVA.last_error);
END;
```

C.7.6 CLEAR_ERROR ビルトイン

説明

ORA_JAVA.LAST_ERROR によって取り出した、最後のエラー・テキストを削除します。

構文

```
PROCEDURE CLEAR_ERROR;
```

パラメータ

なし

例

```
/*
** Example of retrieving the value of an invalid array element.
*/
PROCEDURE foo IS
    arr   ORA_JAVA.JARRAY;
    n     PLS_INTEGER;
BEGIN
    ...
    arr := ORA_JAVA.NEW_BYTE_ARRAY(5);
    n   := ORA_JAVA.GET_BYTE_ARRAY_ELEMENT(arr, -1);
    ...
EXCEPTION
    WHEN ORA_JAVA.JAVA_ERROR THEN
        message('  Alert: ' || ORA_JAVA.last_error);
        ORA_JAVA.CLEAR_ERROR;
END;
```

C.7.7 NEW_<java_type>_ARRAY ビルトイン

説明

指定された Java データ型の新規配列を作成します。

構文

```
FUNCTION NEW_OBJECT_ARRAY (
    length IN PLS_INTEGER,
    clsname IN VARCHAR2) RETURN ORA_JAVA.JARRAY;


FUNCTION NEW_BYTE_ARRAY (
    length IN PLS_INTEGER) RETURN ORA_JAVA.JARRAY;


FUNCTION NEW_CHAR_ARRAY (
    length IN PLS_INTEGER) RETURN ORA_JAVA.JARRAY;


FUNCTION NEW_SHORT_ARRAY (
    length IN PLS_INTEGER) RETURN ORA_JAVA.JARRAY;


FUNCTION NEW_INT_ARRAY (
    length IN PLS_INTEGER) RETURN ORA_JAVA.JARRAY;


FUNCTION NEW_LONG_ARRAY (
    length IN PLS_INTEGER) RETURN ORA_JAVA.JARRAY;


FUNCTION NEW_FLOAT_ARRAY (
    length IN PLS_INTEGER) RETURN ORA_JAVA.JARRAY;


FUNCTION NEW_DOUBLE_ARRAY (
    length IN PLS_INTEGER) RETURN ORA_JAVA.JARRAY;

FUNCTION NEW_STRING_ARRAY (
    length IN PLS_INTEGER) RETURN ORA_JAVA.JARRAY;


FUNCTION NEW_BOOLEAN_ARRAY (
    length IN PLS_INTEGER) RETURN ORA_JAVA.JARRAY;
```

パラメータ

length 作成する配列のサイズ（つまり、配列要素の数）。

clsname クラス・ファイルの完全修飾名。名前中のセパレータとしては '.'（ピリオド）を使用してください。たとえば、java.lang.String とします。

データ型が OBJECT の配列を作成する場合のみ必要です。

戻り値

PL/SQL データ型が ORA_JAVA.JARRAY のオブジェクト。これは ORA_JAVA.JOBJECT のサブタイプです。

使用上の注意

新規配列は、それを作成した PL/SQL トリガーでのみ有効となります。
ORA_JAVA.NEW_GLOBAL_REF ビルトインは、トリガーの存続期間を超えて配列の永続性を拡張する場合に使用してください。

例

```
/*
** Example of creating an array of data type object.
*/
PROCEDURE create_object_array IS
    arr ORA_JAVA.JOBJECT;
BEGIN
    arr := ORA_JAVA.NEW_OBJECT_ARRAY(3, 'java.lang.String');
    ...
END;


/*
** Example of creating an array of data type char with one element.
*/
PROCEDURE create_char_array IS
    arr ORA_JAVA.JOBJECT;
BEGIN
    arr := ORA_JAVA.NEW_CHAR_ARRAY(1);
    ...
END;
```

C.7.8 GET_<java_type>_ARRAY_ELEMENT ビルトイン

説明

指定された Java データ型の指定された配列にある、指定された要素の現行値を返します。値は、対応する PL/SQL データ型で返されます。

構文

```
FUNCTION GET_OBJECT_ARRAY_ELEMENT (
    arr IN ORA_JAVA.JARRAY,
    pos IN PLS_INTEGER) RETURN ORA_JAVA.JOBJECT;


FUNCTION GET_BYTE_ARRAY_ELEMENT (
    arr IN ORA_JAVA.JARRAY,
    pos IN PLS_INTEGER) RETURN PLS_INTEGER;


FUNCTION GET_CHAR_ARRAY_ELEMENT (
    arr IN ORA_JAVA.JARRAY,
    pos IN PLS_INTEGER) RETURN PLS_INTEGER;


FUNCTION GET_SHORT_ARRAY_ELEMENT (
    arr IN ORA_JAVA.JARRAY,
    pos IN PLS_INTEGER) RETURN PLS_INTEGER;


FUNCTION GET_INT_ARRAY_ELEMENT (
    arr IN ORA_JAVA.JARRAY,
    pos IN PLS_INTEGER) RETURN NUMBER;


FUNCTION GET_LONG_ARRAY_ELEMENT (
    arr IN ORA_JAVA.JARRAY,
    pos IN PLS_INTEGER) RETURN NUMBER;


FUNCTION GET_FLOAT_ARRAY_ELEMENT (
    arr IN ORA_JAVA.JARRAY,
    pos IN PLS_INTEGER) RETURN NUMBER;


FUNCTION GET_DOUBLE_ARRAY_ELEMENT (
    arr IN ORA_JAVA.JARRAY,
    pos IN PLS_INTEGER) RETURN NUMBER;


FUNCTION GET_STRING_ARRAY_ELEMENT (
```

```
    arr IN ORA_JAVA.JARRAY,
    pos IN PLS_INTEGER) RETURN VARCHAR2;


FUNCTION GET_BOOLEAN_ARRAY_ELEMENT (
    arr IN ORA_JAVA.JARRAY,
    pos IN PLS_INTEGER) RETURN BOOLEAN;
```

パラメータ

arr	指定したデータ型の有効な配列。実際のパラメータは、ORA_JAVA.JARRAY 型の配列です。
pos	配列内の要素の位置。最初の要素の位置は 0 であることに注意してください。たとえば、サイズが 3 の配列では、要素の位置は 0、1、2 となります。

戻り値

対応する PL/SQL データ型（PLS_INTEGER、NUMBER、VARCHAR2、BOOLEAN または ORA_JAVA.JOBJECT）の値。

使用上の注意

指定したデータ型の配列は有効でなければなりません。

配列の長さが不明な場合は、まず ORA_JAVA.GET_ARRAY_LENGTH を使用して配列のサイズを取得してください。

一度に取り出せる値は 1 つのみです。

例

```
/*
** Example of getting 3 values of an array of data type object.
*/
PROCEDURE get_object_array IS
    arr ORA_JAVA.JARRAY;
    obj1 ORA_JAVA.JOBJECT;
    obj2 ORA_JAVA.JOBJECT;
    obj3 ORA_JAVA.JOBJECT;
BEGIN
    arr := myclass.getMyArray;
    obj1 := ORA_JAVA.GET_OBJECT_ARRAY_ELEMENT(arr, 0);
    obj2 := ORA_JAVA.GET_OBJECT_ARRAY_ELEMENT(arr, 1);
    obj3 := ORA_JAVA.GET_OBJECT_ARRAY_ELEMENT(arr, 2);
    ...
END;
```

C.7.9 SET_<java_type>_ARRAY_ELEMENT ビルトイン

説明

指定された Java データ型の指定された配列にある指定された要素の値を、指定された値に変更します。

構文

```
PROCEDURE SET_OBJECT_ARRAY_ELEMENT (
    arr IN ORA_JAVA.JARRAY,
    pos IN PLS_INTEGER,
    value IN ORA_JAVA.JOBJECT);


PROCEDURE SET_BYTE_ARRAY_ELEMENT (
    arr IN ORA_JAVA.JARRAY,
    pos IN PLS_INTEGER,
    value IN PLS_INTEGER);


PROCEDURE SET_CHAR_ARRAY_ELEMENT (
    arr IN ORA_JAVA.JARRAY,
    pos IN PLS_INTEGER,
    value IN PLS_INTEGER);


PROCEDURE SET_SHORT_ARRAY_ELEMENT (
    arr IN ORA_JAVA.JARRAY,
    pos IN PLS_INTEGER,
    value IN PLS_INTEGER);


PROCEDURE SET_INT_ARRAY_ELEMENT (
    arr IN ORA_JAVA.JARRAY,
    pos IN PLS_INTEGER,
    value IN NUMBER);


PROCEDURE SET_LONG_ARRAY_ELEMENT (
    arr IN ORA_JAVA.JARRAY,
    pos IN PLS_INTEGER,
    value IN NUMBER);


PROCEDURE SET_FLOAT_ARRAY_ELEMENT (
    arr IN ORA_JAVA.JARRAY,
    pos IN PLS_INTEGER,
    value IN NUMBER);
```

```
PROCEDURE SET_DOUBLE_ARRAY_ELEMENT (
    arr IN ORA_JAVA.JARRAY,
    pos IN PLS_INTEGER,
    value IN NUMBER);

PROCEDURE SET_STRING_ARRAY_ELEMENT (
    arr IN ORA_JAVA.JARRAY,
    pos IN PLS_INTEGER,
    value IN VARCHAR2);

PROCEDURE SET_BOOLEAN_ARRAY_ELEMENT (
    arr IN ORA_JAVA.JARRAY,
    pos IN PLS_INTEGER,
    value IN BOOLEAN);
```

パラメータ

arr	指定したデータ型の有効な配列。実際のパラメータは、ORA_JAVA.JARRAY 型の配列です。
pos	配列内の置換対象の要素の位置。最初の要素の位置は常に 0 であることに注意してください。たとえば、サイズが 3 の配列では、要素の位置は 0、1、2 となります。
value	配列要素を置換する、指定したデータ型の新しい値。

使用上の注意

指定したデータ型の配列および置換対象の配列要素は有効でなければなりません。

配列の長さが不明な場合は、まず ORA_JAVA.GET_ARRAY_LENGTH を使用して配列のサイズを取得してください。

一度に設定できる値は 1 つのみです。

例

```
/*
** Example of changing 3 values of an array of data type object.
*/
PROCEDURE set_object_array IS
    arr ORA_JAVA.JOBJECT;
    obj ORA_JAVA.JOBJECT;
BEGIN
    arr := ORA_JAVA.NEW_OBJECT_ARRAY(3, 'myapp.foo');
    ORA_JAVA.SET_OBJECT_ARRAY_ELEMENT(arr, 0, foo.new('obj1'));
```

```
    ORA_JAVA.SET_OBJECT_ARRAY_ELEMENT(arr, 1, foo.new('obj2'));
    ORA_JAVA.SET_OBJECT_ARRAY_ELEMENT(arr, 2, foo.new('obj3'));
    ...
END;


/*
** Example of changing the value of an array of data type char.
*/
PROCEDURE set_char_array IS
    arr ORA_JAVA.JOBJECT;
BEGIN
    arr := ORA_JAVA.NEW_CHAR_ARRAY(1);
    ORA_JAVA.SET_CHAR_ARRAY_ELEMENT(arr, 0, 2);
    ...
END;
```

C.7.10 IS_NULL ビルトイン

説明

オブジェクトが NULL かどうかを示す BOOLEAN 値を返します。

構文

```
FUNCTION IS_NULL (obj IN ORA_JAVA.JOBJECT) RETURN BOOLEAN;
```

パラメータ

obj Java クラスの有効なインスタンス。実際のパラメータには、ORA_JAVA.JOBJECT 型の任意のオブジェクトを指定できます。

例

```
PROCEDURE foo IS
    obj ORA_JAVA.JOBJECT;
    ...
BEGIN
    obj := myclass.new;
    IF NOT ORA_JAVA.IS_NULL(obj) THEN
        pck.obj := ORA_JAVA.NEW_GLOBAL_REF(obj);
        ...
    END IF;
    ...
END;
```

C.7.11 GET_ARRAY_LENGTH ビルトイン

説明
配列のサイズ（長さ）を返します。

構文
```
FUNCTION GET_ARRAY_LENGTH (
    arr IN ORA_JAVA.JARRAY) RETURN PLS_INTEGER;
```

パラメータ
arr　　有効な配列。

戻り値
PLS_INTEGER。

使用上の注意
このビルトインは、配列の長さを取得する場合に使用してください。有効な配列を指定する必要があります。

例
```
PROCEDURE get_size IS
    n PLS_INTEGER;
    arr ORA_JAVA.JARRAY;
BEGIN
    arr := myclass.getMyArray;
    IF NOT ORA_JAVA.IS_NULL(arr) THEN
        n := ORA_JAVA.GET_ARRAY_LENGTH(arr);
        ...
    END IF;
    ...
END;
```

第III部

索引

索引

数字

A

B

C

D

F

G

H

I

J

せ

そ

た

ち

て

と

に

ね

は