Sun Java Enterprise System 5 Update 1 技術の概要

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 820–3093
2007 年 9 月

070914@18741

目次

表目次

図目次

はじめに

『Sun Java Enterprise System 5 Update 1 技術の概要』では、Sun Java™ Enterprise System (Java ES) の技術的および概念上の基礎を紹介します。また、Java ES のコンポーネント、アーキテクチャー、プロセス、および機能についても説明します。

ここでは、Java ES のマニュアルセットで使用される技術的な概念および用語を定義します。各章の最後の項には、重要な技術用語の説明があります。

対象読者

このマニュアルは、ビジネスアナリスト、システム設計者、フィールドエンジニア、システム管理者などを含む、Java ES に基づいてソフトウェアソリューションを設計、配備、または保守するユーザー向けに書かれています。

このマニュアルを読むには、次の各技術に関する一定の知識が必要になります。

- 一般的なネットワークの概念
- 認証および承認に関連するセキュリティーの基礎
- Java プログラミング言語
- Java 2 Platform, Standard Edition (J2SE™ プラットフォーム) コンポーネントおよび Java 2 Platform, Enterprise Edition (J2EE™ プラットフォーム) コンポーネント

内容の紹介

このマニュアルは、次の章で構成されています。

- 第 1 章では、Java ES およびシステムの使用に関連した作業について紹介します。
- 第 2 章では、Java ES ソリューションアーキテクチャーの設計のための、アーキテクチャーのフレームワークについて説明し、そのフレームワークに基づいたアーキテクチャーの例を示します。
- 第 3 章では、Java ES コンポーネントを単一のソフトウェアシステムに統合する上で重要な役割を担う機能についての情報を提供します。
- 第 4 章では、Java ES ソリューションのライフサイクルの各フェーズに関連した、概念および用語について説明します。

- 付録 A では、Java ES コンポーネントの一覧を示します。

Java Enterprise System のマニュアルセット

Java ES システムのマニュアルセットでは、配備計画およびシステムインストールについて説明します。システムマニュアルの URL は http://docs.sun.com/coll/1286.3 です。Java ES の概要を理解するため、次の表に紹介されているマニュアルを、記載されている順番に参照してください。Java ES についての情報およびリソースは、http://www.sun.com/bigadmin/hubs/javaes/ からも利用できます。

表 P–1　Java Enterprise System のマニュアル

マニュアルタイトル	内容
『Sun Java Enterprise System 5 Update 1 Release Notes 』	既知の問題など、Java ES に関する最新の情報が記載されています。これ以外に、コンポーネントごとのリリースノートがリリースノートコレクションに掲載されています (http://docs.sun.com/coll/1315.3)。
『Sun Java Enterprise System 5 Update 1 What's New』	Java ES 5 Update 1 で導入された新しい特長および機能の概要を示します。更新されたマニュアルへのリンクを提供します。
『Sun Java Enterprise System 5 Update 1 技術の概要』	Java ES の技術的および概念的な基礎について説明します。コンポーネント、アーキテクチャー、プロセス、および機能について説明しています。
『Sun Java Enterprise System Deployment Planning Guide 』	Java ES に基づく企業配備ソリューションの計画および設計に関する情報を提供します。配備の計画および設計に関する基本的概念と原則を示し、ソリューションのライフサイクルについて説明し、Java ES に基づくソリューションを計画する際に使用する高度な例と戦略を提供します。
『Sun Java Enterprise System 5 インストール計画ガイド』	Java ES の配備に関し、ハードウェア、オペレーティングシステム、およびネットワーク面の実装仕様の開発に役立つ情報を提供します。インストールおよび設定計画を遂行する上で注意すべきコンポーネントの依存関係などの問題について説明します。
『Sun Java Enterprise System 5 Update 1 Installation Guide for UNIX 』 『Sun Java Enterprise System 5 Installation Guide for Microsoft Windows 』	Java ES のインストール手順について説明します。また、インストール後にコンポーネントを設定する方法、および設定したコンポーネントが正常に機能するかどうかを確認する方法についても説明します。

表 P–1 Java Enterprise System のマニュアル (続き)

マニュアルタイトル	内容
『Sun Java Enterprise System 5 Update 1 インストールリファレンス (UNIX 版)』	パラメータの設定についての追加情報や、設定の計画に使用するワークシートを提供し、Solaris オペレーティングシステムおよび Linux オペレーティング環境でのデフォルトのディレクトリおよびポート番号などの参照資料の一覧が記載されています。
『Sun Java Enterprise System 5 Update 1 Upgrade Guide for UNIX 』	以前にインストールされたバージョンから、Java ES 5 Update 1 へとアップグレードする手順を説明します。
『Sun Java Enterprise System 5 Update 1 Upgrade Guide for Microsoft Windows 』	
『Sun Java Enterprise System 5 Update 1 Monitoring Guide 』	各製品での Monitoring Framework の設定方法と、Monitoring Console を使用してリアルタイムデータを参照し、監視規則を作成する方法を示します。
『Sun Java Enterprise System Glossary 』	Java ES ドキュメントで使用される用語について説明します。

表記上の規則

このマニュアルでは、次のような字体や記号を特別な意味を持つものとして使用します。

表 P–2 表記上の規則

字体または記号	意味	例
`AaBbCc123`	コマンド名、ファイル名、ディレクトリ名、画面上のコンピュータ出力、コード例を示します。	`.login` ファイルを編集します。 `ls -a` を使用してすべてのファイルを表示します。 `machine_name% you have mail.`
`AaBbCc123`	ユーザーが入力する文字を、画面上のコンピュータ出力と区別して示します。	`machine_name%` **`su`** `Password:`
aabbcc123	変数を示します。実際に使用する特定の名前または値で置き換えます。	ファイルを削除するには、`rm` *filename* と入力します。
『 』	参照する書名を示します。	『コードマネージャー・ユーザーズガイド』を参照してください。

表 P–2 表記上の規則 *(続き)*

字体または記号	意味	例
「」	参照する章、節、ボタンやメニュー名、強調する単語を示します。	第 5 章「衝突の回避」を参照してください。 この操作ができるのは、「スーパーユーザー」だけです。
\	枠で囲まれたコード例で、テキストがページ行幅を超える場合に、継続を示します。	`sun% grep '^#define \` `XV_VERSION_STRING'`

コード例は次のように表示されます。

- C シェル

  ```
  machine_name% command y|n [filename]
  ```

- C シェルのスーパーユーザー

  ```
  machine_name# command y|n [filename]
  ```

- Bourne シェルおよび Korn シェル

  ```
  $ command y|n [filename]
  ```

- Bourne シェルおよび Korn シェルのスーパーユーザー

  ```
  # command y|n [filename]
  ```

[ ] は省略可能な項目を示します。上記の例は、*filename* は省略してもよいことを示しています。

| は区切り文字 (セパレータ) です。この文字で分割されている引数のうち 1 つだけを指定します。

キーボードのキー名は英文で、頭文字を大文字で示します (例: Shift キーを押します)。ただし、キーボードによっては Enter キーが Return キーの動作をします。

ダッシュ (-) は 2 つのキーを同時に押すことを示します。たとえば、Ctrl-D は Control キーを押したまま D キーを押すことを意味します。

マニュアル、サポート、およびトレーニング

Sun のサービス	URL	内容
マニュアル	http://jp.sun.com/documentation/	PDF 文書および HTML 文書をダウンロードできます。
サポートおよびトレーニング	http://jp.sun.com/supportraining/	技術サポート、パッチのダウンロード、および Sun のトレーニングコース情報を提供します。

◆ ◆ ◆ 第 1 章

Java Enterprise System の概要

Sun Java™ Enterprise System (Java ES) は、ネットワークを介して、またはインターネット環境で分散しているエンタープライズ版アプリケーションをサポートするために必要なサービスを提供する一連のソフトウェアコンポーネントです。このようなアプリケーションを「分散型のエンタープライズアプリケーション」と呼びます。このマニュアルでは Java ES のソフトウェアコンポーネントとコンポーネントが提供するサービスに重点を置いています。

この章では、Java ES およびシステムの使用に関連する作業を紹介します。この章は、次の節で構成されます。

Java ES が必要な理由

現在のビジネスでは、ネットワークまたはインターネット環境を介して分散する、高いレベルのパフォーマンス、可用性、セキュリティー、スケーラビリティー、および保守容易性を備えたソフトウェアソリューションが必要とされています。

Java ES では、このような分散型のエンタープライズアプリケーション、つまり、通常次のような特徴を持つアプリケーションをサポートするために必要なインフラストラクチャーサービスを提供します。

- 分散型: アプリケーションは、対話型のソフトウェアコンポーネントで構成されています。それらはネットワーク環境にわたって配備され、地理的に離れたサイトを含むこともあります。環境内のさまざまなコンピュータ上で動作するこれらの分散型コンポーネントは、互いに連携して動作することで、エンドユーザーおよびその他のビジネスアプリケーションに対して特定のビジネス機能を提供します。

- エンタープライズ版: アプリケーションの適用範囲と規模が、本稼働環境またはインターネットサービスプロバイダのニーズを満たしています。このアプリケーションは通常、企業全体に及んでおり、多くの部門、オペレーション、およびプロセスを 1 つのソフトウェアシステムに統合します。アプリケーションは、パフォーマンス、可用性、セキュリティー、スケーラビリティー、および保守容易性に関する高度なサービス要件を満たす必要があります。

分散型のエンタープライズアプリケーションでは、分散型コンポーネントが互いに通信できるようにしたり、それらのコンポーネントの動作を調整したり、セキュリティー保護されたアクセスを実装したりするための、基盤となるインフラストラクチャーサービスが必要となります。さらに、これらのインフラストラクチャーサービスは、コンピュータやネットワークリンクから成るハードウェア環境によってサポートされます。このハードウェア環境には、SPARC® と x86 (Intel および AMD) のハードウェアアーキテクチャーが含まれます。

次の図に、全体的な階層スキーマを示します。Java ES は基本的に、図に示す分散型インフラストラクチャーサービス層を提供します。

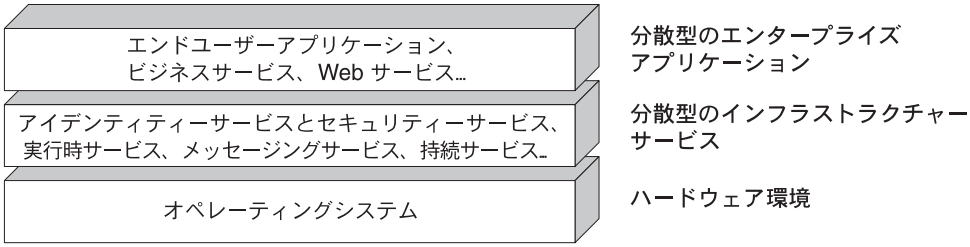

図 1–1 分散型のエンタープライズアプリケーションに必要なサポート

Java ES によって提供される主なサービスには、以下のものがあります。

- ポータルサービス: これらのサービスを使用すると、従業員、在宅勤務者、知識労働者、パートナ企業、仕入先、および顧客が、企業ネットワーク内外から企業のリソースにアクセスできます。これらのサービスは、ユーザーコミュニティーに対する時間と場所を問わないアクセス、配信の個人用にカスタマイズされた統合、集約、セキュリティー、モバイルアクセス、および検索の機能を提供します。
- 通信サービスと共同作業サービス: これらのサービスは、多様なユーザーコミュニティー間で情報を安全に交換することを可能にします。具体的な機能には、ユーザーのビジネス環境で使用する、インスタントメッセージング、会議、カレンダのスケジューリングなどのメッセージング機能によるリアルタイムの共同作業があります。

注 – このマニュアルでは、Sun Java Communications Suite のコンポーネントについて参照します。これらのコンポーネントは、Java ES コンポーネントに依存しており Java ES 配備アーキテクチャー内で使用されます。コミュニケーションコンポーネントおよびコラボレーションコンポーネントは、Java ES には含まれません。

- ネットワークアイデンティティーおよびセキュリティーサービス: グローバルベースのすべてのコミュニティー、アプリケーション、サービスの間で適切なアクセス制御が確実に適用されるようにすることで、企業の主要な情報資産のセキュリティーと保護を向上させます。これらのサービスは、アイデンティティープロファイル、アクセス権、およびアプリケーションとネットワークのリソース情報の格納と管理に使用されるリポジトリと連動します。
- **Web** コンテナサービスとアプリケーションサービス: これらのサービスは、分散型コンポーネントが実行時に互いに通信することを可能にするほか、広範なサーバー、クライアント、およびデバイス用のアプリケーションの開発、配備、および管理をサポートします。これらのサービスは、J2EE 技術に基づいています。

Java ES は、可用性、スケーラビリティー、保守容易性、およびその他のアプリケーションまたはシステムの品質を向上させるサービスも提供します。Java ES によって提供されるサービス品質の機能には、以下のものがあります。

- 可用性サービス: これらのサービスは、アプリケーションコンポーネントとそれらをサポートするインフラストラクチャーコンポーネントに対して、ほぼ連続的な可用性を提供します。
- アクセスサービス: これらのサービスは、Java ES サービスへのインターネットまたはブラウザベースのアクセスを提供します。
- 監視サービス: これらのサービスは、Java ES コンポーネントについての情報をリアルタイムで提供します。

各サービスがいくつかの Java ES コンポーネントから構成される、1 つ以上の Java ES サービスを配備できます。

Java ES コンポーネント

Java ES は、個別のソフトウェア製品やソフトウェアコンポーネントを単一のソフトウェアシステムとして統合化したものです。次のような多数のシステムレベルの機能によって、これらの統合が可能になりました。

- 共通に使用される一連の共有ライブラリですべてのコンポーネントの同期をとります。
- すべての Java ES コンポーネントは、単一のインストーラを使ってインストールされます。

- 統合された 1 つのユーザーアイデンティティーおよびセキュリティー管理システムをすべての Java ES コンポーネントが共有できます。
- すべての Java ES コンポーネントには、共通の監視フレームワークがあります。

これらの機能については、このマニュアルの後の章で説明します。ここでは、Java ES に統合されたコンポーネントの説明に重点を置いています。これらの システムコンポーネントは、次の図に示すように、3 つの主要カテゴリに分類できます。

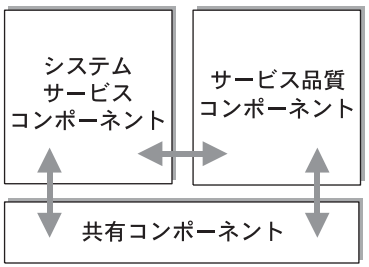

図 1–2　Java ES コンポーネントのカテゴリ

コンポーネントが提供するサービスは、次のとおりです。

- システムサービスコンポーネント: 分散型のエンタープライズアプリケーションをサポートする主な Java ES インフラストラクチャーサービスを提供します。
- サービス品質コンポーネント: システムサービスコンポーネントおよび分散型アプリケーションコンポーネントの可用性、セキュリティー、スケーラビリティー、保守容易性、およびその他の要素を向上させます。
- 共有コンポーネント: 多くのシステムサービスコンポーネントおよびサービス品質コンポーネントを実行する環境を提供します。

Java ES コンポーネントのリストについては、付録 A を参照してください。

システムサービスコンポーネント

いくつかの Java ES コンポーネントが分散型ソフトウェアソリューションをサポートする主なサービスを提供します。これらのシステムサービスには、ポータルサービス、アイデンティティーサービスとセキュリティーサービス、Webコンテナサービス、J2EE アプリケーションサービス、および持続サービスが含まれます。

これらの分散型サービスを提供するシステムサービスコンポーネントとそれらが提供するサービスをアルファベット順に一覧表示して、簡単に説明したのが、次の表です。各システムサービスコンポーネントはマルチスレッド対応のサーバープロセスであり、多数のクライアントをサポートします。コンポーネントの詳細については、73 ページの「システムサービスコンポーネント」を参照してください。

表 1–1　Java ES システムサービスコンポーネント

コンポーネント	提供されるシステムサービス
Sun Java System Access Manager	アクセス管理サービスおよびデジタルアイデンティティー管理サービスを提供します。アクセス管理サービスには、シングルサインオンを含む認証と、アプリケーションまたはサービス、あるいはその両方へのアクセスに対するロールに基づく承認が含まれます。管理サービスには、個々のユーザーアカウント、ロール、グループ、およびポリシーの集中管理が含まれます。
Sun Java System Application Server	セッション Beans、エンティティー Beans、メッセージ駆動型 Beans など、EJB (Enterprise JavaBeans™) コンポーネントの J2EE コンテナサービスを提供します。コンテナは、密接に結合された分散コンポーネント間の対話に必要なインフラストラクチャーサービスを提供し、Application Server を e-コマースアプリケーションおよび Web サービスを開発および実行するためのプラットフォームにします。Application Server は Web コンテナサービスも提供します。
Sun Java System Directory Server	アイデンティティープロファイル (従業員、顧客、仕入先など)、ユーザーの信用情報 (公開鍵の証明書、パスワード、PIN 番号)、アクセス特権、アプリケーションリソース情報、ネットワークリソース情報などのイントラネット情報およびインターネット情報を格納および管理するための中央リポジトリを提供します。
Java DB[1]	Java アプリケーションの開発用に、軽量データベースを提供します。Java DB は、100% Java テクノロジによるデータベースであるオープンソース Apache Derby の、Sun がサポートする配布です。
Sun Java System Message Queue	緩やかに結合された分散型のコンポーネントやアプリケーション間における、信頼性の高い非同期のメッセージングを提供します。Message Queue は、JMS (Java Message Service) API 仕様を実装しているほか、さらにセキュリティー、スケーラビリティー、リモート管理などのエンタープライズ機能も備えています。
Sun Java System Portal Server	ビジネスアプリケーションやビジネスサービスにアクセスするブラウザベースのクライアントに対し、コンテンツの集約や個人用のカスタマイズなど、主要なポータルサービスを提供します。Portal Server は、設定可能な検索エンジンも提供します。
Sun Java System Service Registry	サービス指向アーキテクチャー (SOA) に対応した Web アプリケーションをサポートするレジストリおよびリポジトリを提供します。Service Registry は、Web サービスを登録および検出するための業界標準に加え、関連情報や事実、XML スキーマ、ビジネスプロセスルール、アクセス制御、バージョン管理といったアーティファクトなどを管理するための業界標準を実装しています。

[1] Java ES 5 は、Java DB を製品コンポーネントとして含む最初のリリースです。Java DB は最初 Derby Database という名前の共有コンポーネントとしてリリースされ、Java ES 2005Q4 に含まれていました。

表 1–1 Java ES システムサービスコンポーネント (続き)

コンポーネント	提供されるシステムサービス
Sun Java System Web Server	Java サーブレットコンポーネントや JSP™ (JavaServer Pages™) コンポーネントなどの Java Web コンポーネントに対して、J2EE Web コンテナサービスを提供します。Web Server は、CGI スクリプトや Sun Java™ System Active Server Pages など、静的および動的な Web コンテンツを配信するためのその他の Web アプリケーション技術もサポートしています。

サービス品質コンポーネント

Java ES には、表 1–1 に示したシステムサービスコンポーネントのほかに、システムサービスコンポーネントが提供するサービスの品質を高めるためのコンポーネントがいくつか含まれています。また、サービス品質コンポーネントを使えば、カスタム開発されたアプリケーションサービスも改善できます。サービス品質コンポーネントは次のカテゴリに分類されます。

- 可用性コンポーネント
- アクセスコンポーネント
- 監視コンポーネント

可用性コンポーネント

可用性コンポーネントは、システムサービスコンポーネントおよびカスタムアプリケーションサービスがほぼ連続的に稼動することを可能にします。Java ES に含まれる可用性コンポーネントおよびそれらのコンポーネントが提供するサービスを次の表に示します。各コンポーネントの詳細については、78 ページの「可用性コンポーネント」を参照してください。

表 1–2 Java ES 可用性コンポーネント

コンポーネント	提供される可用性サービス
High Availability Session Store	障害発生時でも、アプリケーションのデータ、特にセッション状態データを利用可能にするデータストアを提供します。
Sun Cluster	Java ES の高可用性サービスとスケーラビリティーサービス、Java ES インフラストラクチャーの最上部で実行されるアプリケーション、およびサービスとアプリケーションの両方が配備されるハードウェア環境を提供します。

表 1–2 Java ES 可用性コンポーネント (続き)

コンポーネント	提供される可用性サービス
Sun Cluster Geographic Edition[1]	地理的に離れた複数のクラスタを使用し、これらのクラスタ間でデータを複製する冗長なインフラストラクチャーを使用することによって、予期しない中断からアプリケーションを保護します。Sun Cluster Geographic Edition ソフトウェアは、Sun Cluster ソフトウェアを階層的に拡張したものです。

[1] Java ES 5 は、Sun Cluster Geographic Edition を Java ES 製品コンポーネントとして含む最初のリリースです。

アクセスコンポーネント

アクセスコンポーネントは、システムサービスへのフロントエンドアクセスを可能にし、多くの場合、エンタープライズファイアウォールの外にあるインターネットからのセキュリティー保護されたアクセスを可能にします。そのようなアクセスを可能にすることに加えて、多くの場合、ルーティング機能およびキャッシュ機能も提供します。Java ES に含まれるアクセスコンポーネントおよびそれらのコンポーネントが提供するサービスを次の表に示します。コンポーネントの詳細については、80 ページの「アクセスコンポーネント」を参照してください。

表 1–3 Java ES アクセスコンポーネント

コンポーネント	提供されるアクセスサービス
Sun Java System Portal Server (Secure Remote Access を含む)	企業ファイアウォールの外から、内部ポータルを含む Portal Server のコンテンツやサービスへの、セキュリティー保護されたインターネットアクセスを提供します。
Sun Java System Web Proxy Server	Web コンテンツをキャッシュ、フィルタリング、および配信する機能を、送信インターネット要求と受信インターネット要求の両方に対して提供します。

監視コンポーネント

Java ES には、リアルタイムのシステム状態とカスタマイズ可能な監視ジョブを提供する監視機能が含まれています。監視は、Sun Java System Monitoring Console 製品コンポーネント によって実装され、これは Sun Java System Monitoring Framework 共有コンポーネント (shared component)によってサポートされます。詳細は、81 ページの「監視コンポーネント」を参照してください。

共有コンポーネント

Java ES には、多くのシステムサービスコンポーネントおよびサービス品質コンポーネントが依存する、ローカルにインストールされる共有ライブラリがいくつか含まれています。Java ES 共有コンポーネントは、同じホストコンピュータ上で稼働する Java ES 製品コンポーネントにローカルサービスを提供します。

共有コンポーネントは、多くの場合、異なるオペレーティングシステム間の移植性を提供するために使用されます。Java ES 共有コンポーネントの例として、Java 2 Platform, Standard Edition (J2SE)、Netscape Portable Runtime (NSPR)、Network Security Services (NSS)、Java Security Services for Java (JSS) などがあります。完全な一覧については、81 ページの「共有コンポーネント」を参照してください。

共有コンポーネントのインストールは、インストール対象のシステムサービスやサービス品質コンポーネントに応じて、Java ES インストーラによって自動的に行われます。

Sun Java Suites のコンポーネント

Java ES は、単一のエンドツーエンドのインフラストラクチャーソフトウェア配布として、およびクリティカルなビジネスのニーズを満たすことを目的とした個別のスイート配布としての両方で利用可能です。Java ES には、すべての Java ES コンポーネントが含まれ、一方 Sun Java System Suites には特定の業務のニーズを満たすものを選択したコンポーネントのサブセットが含まれます。Java ES インストーラおよびアンインストーラはすべてのスイート配布に含まれますが、スイート内のコンポーネントだけを扱うように軽減されています。すべての共有コンポーネントも、すべてのスイート配布に含まれています。

個々のスイートの内容およびそれぞれのスイートが意図されているビジネス要件を、次の表に一覧表示します。

表 1–4　Sun Java Suites のコンポーネント

スイート	ビジネス要件	内容
Sun Java Application Platform Suite	次世代のサービス指向アーキテクチャー (SOA) の開発、配備、および管理	Access Manager Application Server Directory Server HADB Java DB Message Queue Monitoring Console Portal Server (Secure Remote Access および Mobile Access を含む) Service Registry Web Proxy Server Web Server
Sun Java Availability Suite	ミッションクリティカルなアプリケーションの障害回復および高可用性	Sun Cluster ソフトウェア Sun Cluster エージェント Sun Cluster Geographic Edition

表 1–4 Sun Java Suites のコンポーネント (続き)

スイート	ビジネス要件	内容
Sun Java Communications Suite [1]	セキュリティー保護され信頼できるメッセージングおよび共同作業サービス	Access Manager Application Server Calendar Server* Communications Express* Delegated Administrator* Directory Server HADB Instant Messaging* Java DB Message Queue Messaging Server* Monitoring Console Web Proxy Server Web Server
Sun Java Identity Management Suite	コンピューティングインフラストラクチャーおよびアプリケーション環境間のユーザーアイデンティティー管理	Access Manager Application Server Directory Server HADB Java DB Message Queue Monitoring Console Web Server

[1] アスタリスク (*) のついたコンポーネントはコミュニケーションコンポーネントで、Java ES には含まれなくなった、または Java ES インストーラを介してはインストールされません。これらのコンポーネントは、Sun Java Communications Suite の一部として利用可能です。

表 1–4　Sun Java Suites のコンポーネント　　(続き)

スイート	ビジネス要件	内容
Sun Java Web Infrastructure Suite	小規模から中規模企業向けの Web アプリケーションおよび Web サービス	Access Manager Application Server Directory Server HADB Java DB Message Queue Monitoring Console Service Registry Web Proxy Server Web Server

Java ES での作業

Java ES ソフトウェアに基づくビジネスソリューションの作成には、いくつかの標準的な作業が含まれます。それらの作業の範囲や難易度は、Java ES 導入時の出発点や、作成および配備しようとしているソリューションの性質によって異なります。

ここでは、Java ES での作業における 2 つの側面、つまり、Java ES ソリューションのライフサイクルと通常使用される導入シナリオについて説明します。

Java ES ソリューションのライフサイクル

Java ES ソフトウェアに基づくビジネスソリューションの作成に伴う作業は、次の図に示すように、いくつかのフェーズに分けられます。この図は、作業を通常実行する Java ES ユーザーのカテゴリも示しています。

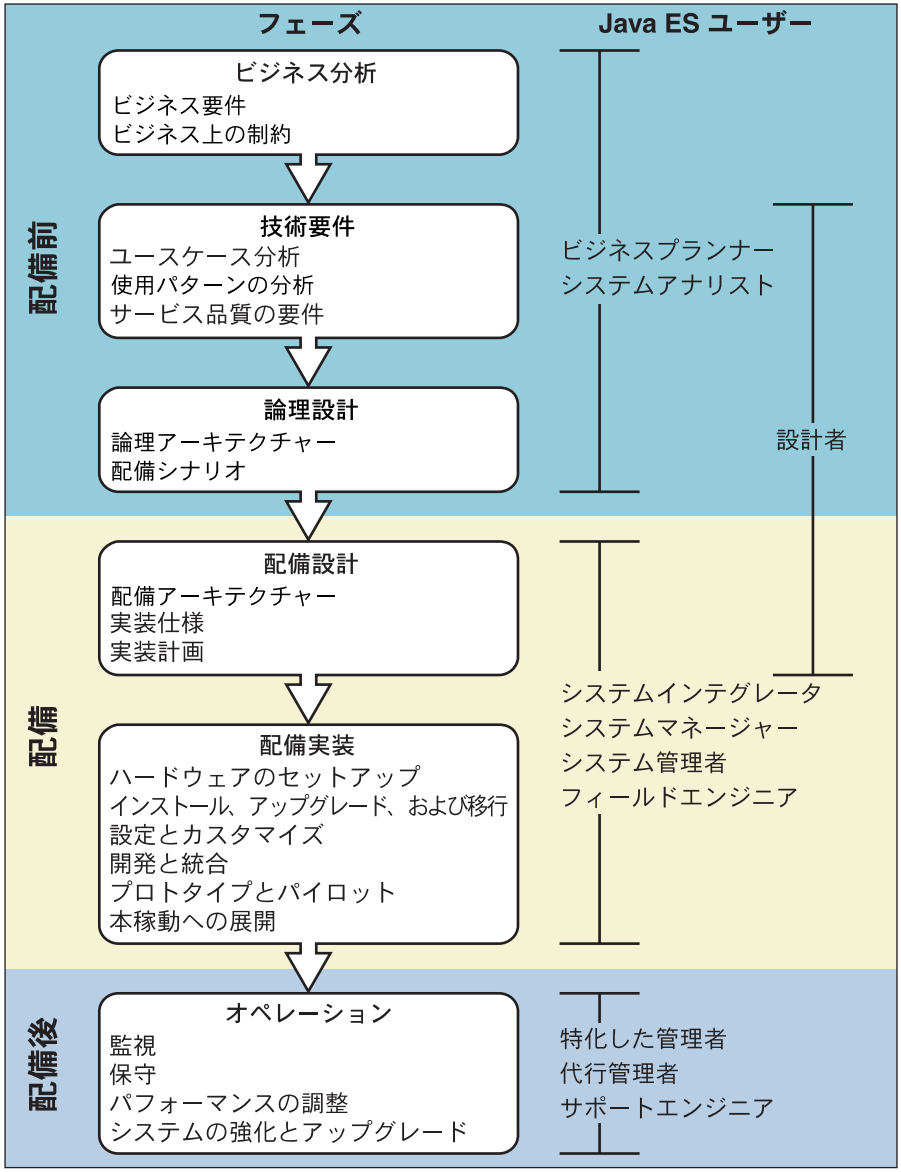

図 1–3　ソリューションライフサイクルのフェーズとユーザーカテゴリ

先の図に示すライフサイクルのフェーズは、次の一般的なグループに分類できます。

- 配置前: このフェーズでは、ビジネスのニーズを配備シナリオ、つまり論理アーキテクチャーと一連のサービス品質の要件に変換します。配備シナリオは、配備アーキテクチャーを設計する仕様としての役割を果たします。
- 配備: このフェーズでは、配備シナリオに基づいて配備アーキテクチャーを作成します。このアーキテクチャーをプロジェクト承認および予算設定の基礎として使用できます。この配備アーキテクチャーは、ソフトウェアソリューションを本稼動環境に配備 (構築、テスト、および展開) するために必要な詳細情報が含まれる実装仕様の基礎にもなります。
- 配備後: オペレーションフェーズでは、配備されたソリューションを本稼働条件下で実行し、パフォーマンスを監視および最適化します。また、配備されたソリューションを必要に応じてアップグレードし、新しい機能を組み入れます。

図 1–3 に示すソリューションのライフサイクルおよび各フェーズの作業については、第 4 章で詳しく説明します。

図 1–3 は、ライフサイクルの各フェーズに示された作業を通常実行する Java ES ユーザーを示しています。各カテゴリのユーザーのスキルとバックグラウンドは、次の表のとおりです。

表 1–5 ライフサイクルの作業に関連する Java ES ユーザーカテゴリ

ユーザー	スキルとバックグラウンド	フェーズ
ビジネスプランナ システムアナリスト	専門的ではなくても、一般的な技術知識を持っている。 企業の戦略上の方向性を理解している。 ビジネスのプロセス、目的、要件を理解している。	ビジネス分析 技術の要件 論理設計
設計者	専門的である。 配備アーキテクチャーの幅広い知識を持つ。 最新の技術に精通している。 ビジネスの要件および制約を理解している。	技術の要件 論理設計 配備設計
システムインテグレータ フィールドエンジニア システム管理者 システムマネージャー	専門的である。 IT 環境に精通している。 分散型のソフトウェアソリューションを実装した経験がある。 ネットワークのアーキテクチャー、プロトコル、デバイス、セキュリティーの知識を持つ。 スクリプト言語およびプログラミング言語の知識を持つ。	配備設計 配備実装

表 1–5 ライフサイクルの作業に関連する Java ES ユーザーカテゴリ (続き)

ユーザー	スキルとバックグラウンド	フェーズ
特化したシステム管理者 代行管理者 サポートエンジニア	特化した技術知識または製品知識を持つ。 ハードウェア、プラットフォーム、ディレクトリ、およびデータベースに精通している。 ソフトウェアの監視、トラブルシューティング、およびアップグレードに熟練している。 オペレーティングシステムプラットフォームのシステムの管理の知識を持つ。	操作

Java ES の導入シナリオ

Java ES の導入につながるビジネスのニーズは多様です。ただし、ほとんどすべての Java ES 配備のハイレベルな目標は、次のいずれかの導入シナリオに該当します。

- 新しいシステム: 既存のソフトウェアシステムが存在しない状態から、新しいビジネスソリューションをサポートするために Java ES ソフトウェアを配備します。
- 強化: 既存の情報技術 (IT) インフラストラクチャーで始め、そのシステムの一部、大部分、または全部を Java ES ソフトウェアで置き換えます。通常は、システムまたはサブシステムが複雑すぎるか、制約が多すぎるか、維持するのに高価であるという理由でシステムやサブシステムを置き換えます。たとえば、セキュリティーの向上、高い可用性、スケーラビリティーの向上、柔軟性の向上、複雑さの解消、追加機能 (シングルサインオンなど)、または IT リソースの有効利用が必要であるなどです。
- 拡張: 既存の IT インフラストラクチャーで始め、現在システムに含まれない Java ES ソフトウェアを配備します。通常、新しいビジネスのニーズに対応するために、ソフトウェアシステムを拡張します。Java ES ポータルによる既存のサービスの個人用にカスタマイズされた集約や既存のサービスに対する Java 認証および承認などの新しい機能が必要になる場合があります。
- アップグレード: Java ES の以前のバージョンまたは Java ES よりも前の Sun 製品から構成される IT インフラストラクチャーで始め、Java ES コンポーネントの最新のバージョンにアップグレードします。

各導入シナリオには、それぞれ考慮しなければならない点と克服しなければならない点があります。導入シナリオによって、図 1–3 で示すライフサイクルの各フェーズで対処する必要のある問題や投資する必要のあるリソースが異なります。

導入シナリオには、次に示す考慮事項が程度の差はありますが適用されます。

- 移行: 既存のインフラストラクチャーを新しいソフトウェアで強化またはアップグレードするには、多くの場合、既存のシステムから新しいシステムにデータを移行する必要があります。データは、設定情報、ユーザー情報、アプリケー

ションの情報などになります。新しいプログラミングインタフェースのために、ビジネスロジックやプレゼンテーションロジックを移行する必要がある場合もあります。

- 統合: 新しいソフトウェアを既存のシステムに追加したり、既存のソフトウェアサブシステムを置き換えたりするには、多くの場合、新しいソフトウェアコンポーネントを残りのサブシステムに統合する必要があります。統合には、新しいインタフェースレイヤーの開発、J2EE コネクタまたはリソースアダプタの使用、既存のコンポーネントの再設定、データ変換スキーマの実装などが含まれます。
- トレーニング: インフラストラクチャーの変更はほとんどの場合、IT 手順やスキルセットの変更を意味します。IT 部門は、Java ES テクノロジをサポートするための、新しいスキルの習得、または古いスキルの移行のために十分な時間を確保できる必要があります。
- ハードウェア: 既存のシステムまたはサブシステムを置き換えたり強化したりする場合、ビジネス上の制約により、既存のハードウェアを再利用しなければならない可能性があります。導入シナリオによっては、ハードウェアリソースが重要な要素になります。

次の表は、Java ES の各導入シナリオに該当する考慮事項の性質を要約しています。

表 1–6　Java ES 導入シナリオに関する考慮事項

導入シナリオ	移行	統合	トレーニング	ハードウェア
新しいシステム	問題なし	新しいコンポーネントの統合は比較的簡単	かなりの考慮が必要な場合がある	機器のコストと労力のコストのバランス[1]
強化	重要な考慮事項になる場合がある	新しいコンポーネントを既存のシステムと統合する必要がある	かなりの考慮が必要な場合がある	既存の機器によるかなりの制約がある場合がある
拡張	通常は問題なし	新しいコンポーネントを既存のシステムと統合する必要がある場合がある	かなりの考慮が必要な場合がある	通常、新しいシステムと同じトレードオフの新しいハードウェアが必要になる
アップグレード	かなりの考慮が必要な場合がある	アップグレードされたコンポーネントの統合は比較的簡単	比較的考慮事項が少ない	比較的考慮事項が少ない

[1] 少数の強力なコンピュータを使用すると、通常、機器のコストは増えますが、必要な IT リソースは少なくなります。多数の小型のコンピュータを使用すると、通常、機器のコストは少なくなりますが、より多くの IT リソースが必要になります。

この章の重要な用語

この節では、この章で使用されている重要な技術用語について説明します。ここでは、Java ES の文脈でどのように使用されているかの説明に重点を置いています。

採用シナリオ **(adoption scenario)**	Java ES ソフトウェアを配備する総合的な理由。着手するソフトウェアシステムや達成すべき目標を説明します。Java ES には、4 つの基本的な導入シナリオがあります。新規システム、置換、拡張、およびアップグレードの 4 つです。
構成要素 (コンポーネント)	分散型アプリケーションを構築するときの基本となるソフトウェアロジックの単位。コンポーネントは、Java ES に含まれるシステムコンポーネント、カスタム開発されたアプリケーションコンポーネント (application component) のいずれかになります。通常、アプリケーションコンポーネントは、CORBA や J2EE™ などの分散型コンポーネントモデルに準拠していて、特定のコンピューティング機能を実行します。これらのコンポーネントを単独でまたは複数を組み合わせて、ビジネスサービスを提供します。また、これらを Web サービスとしてカプセル化することもできます。
分散エンタープライズアプリケーション **(distributed enterprise application)**	そのロジックがネットワーク環境またはインターネット環境に及んでおり (分散の側面)、かつその適用範囲と規模が本稼働環境またはサービスプロバイダのニーズを満たしている (エンタープライズの側面) アプリケーション。
エンドユーザー **(end user)**	分散型アプリケーションを利用するユーザー。エンドユーザーは通常、インターネットブラウザやモバイルデバイス GUI などのグラフィカルユーザーインタフェースを使用します。アプリケーションがサポートする同時接続エンドユーザーの数は、そのアプリケーションの配備アーキテクチャー (deployment architecture) を決定する重要な要因の 1 つになります。
サービス	1 つ以上のクライアントに対して実行されるソフトウェア機能。この機能は、メモリー管理など下位レベルのものであることも、信用調査ビジネスサービス (business service) などの上位レベルのものであることも考えられます。高レベルサービスは、個々のサービスのファミリで構成できます。またサービスには、ローカルクライアントが使用可能なローカルサービスや、リモートクライアントが使用可能な分散サービスがあります。
製品コンポーネント	Java ES システムサービスコンポーネントは、主要な Java ES インフラストラクチャーサービスを提供し、サービス品質コンポーネントはこれらのシステムサービスを拡張します。製品コンポーネントは、Java ES インストーラで選択可能です。
サービス品質コンポーネント **(service quality component)**	Java ES に含まれる何種類かのシステムコンポーネント (system component) のうちの 1 つ。システムサービスコンポーネントおよび分散型アプリケーションコンポーネントの可用性、セキュリティー、スケーラビリティー、保守容易性、およびその他の品質を向上させます。
共有コンポーネント **(shared component)**	Java ES に含まれる何種類かのシステムコンポーネント (system component) のうちの 1 つ。ライブラリなどの共有コンポーネントは、ほかのシステムコンポーネントにローカルサービスを提供します。

システムコンポーネント **(system component)**	Java ES に含まれていて、Java ES インストーラによってインストールされるソフトウェアパッケージまたは一連のパッケージ。何種類かのシステムコンポーネントがあります。Java ES インフラストラクチャーサービスを提供する製品コンポーネント、およびローカルサービスを他のシステムコンポーネントに提供する共有コンポーネントです。
システムサービス	Java ES によって提供される固有の機能を定義する、1 つ以上の分散型サービス。システムサービスは通常、いくつかのサービス品質コンポーネント、またはいくつかの共有コンポーネント、あるいはその両方のサポートを必要とします。
システムサービスコンポーネント **(system service component)**	Java ES に含まれる何種類かのシステムコンポーネント (system component)のうちの 1 つ。システムサービスコンポーネントは、Java ES の主要なインフラストラクチャーサービスを提供します。これには、ポータルサービス、アイデンティティーサービスとセキュリティーサービス、Web サービスとアプリケーションサービス、および可用性サービスが含まれます。

◆ ◆ ◆ 第 2 章

2 Java ES ソリューションアーキテクチャー

この章では、Java ES ソリューションの基礎となるアーキテクチャー概念の概要について説明します。この章では、コンポーネントである、システムサービスコンポーネントおよびサービス品質コンポーネントを使用してどのように分散型エンタープライズソリューションをサポートするかを示します。

Java ES ソリューションアーキテクチャーには 2 つの側面があります。論理アーキテクチャー (logical architecture)と配備アーキテクチャー (deployment architecture)です。論理アーキテクチャーは、ソリューションの論理的な構築ブロック (ソフトウェアコンポーネント) 間の対話を示します。配備アーキテクチャーは、論理アーキテクチャーから物理的なコンピューティング環境へのマッピングを示します。Java ES コンポーネントは、論理アーキテクチャーと配備アーキテクチャーの両方で重要な役割を果たします。

この章では、Java ES ソリューションのアーキテクチャーの設計のためのアーキテクチャーのフレームワーク、およびそのフレームワークに基づいたアーキテクチャーの例について説明します。この章は、次の節で構成されます。

Java ES アーキテクチャーのフレームワーク

Java ES コンポーネントは、分散型ソフトウェアソリューションの配備をサポートします。ビジネス要件によって課されるパフォーマンス、可用性、セキュリティー、スケーラビリティー、および保守容易性のレベルで必要な機能を実現するには、これらのソフトウェアソリューションを適切に設計する必要があります。

エンタープライズ版ソリューションの設計には、アーキテクチャーの次元の多くが関係します。それらの次元は、そのようなシステムの構築に使用される多数のソフ

トウェアコンポーネント間の対話をさまざまな観点から見ることができる観点を表します。特に、分散型システムの設計にはアーキテクチャーの次の 3 つの次元が関係します。

- インフラストラクチャーサービスの依存関係: この次元では、分散型ソリューションのサポートにおけるシステムサービスコンポーネント (20 ページの「システムサービスコンポーネント」を参照) の役割に重点を置いています。
- 論理層: この次元では、ソリューションコンポーネントをネットワークまたはインターネット環境に配備するためのソリューションコンポーネントの論理的および物理的な独立性に重点を置いています。
- サービス品質: この次元では、サービス品質コンポーネント (22 ページの「サービス品質コンポーネント」を参照) の役割も含め、可用性、セキュリティー、スケーラビリティー、保守容易性といったサービス品質の要件をどのようにして実現するかに重点を置いています。

ソリューションアーキテクチャーのこれらの 3 つの次元を次の図に示します。

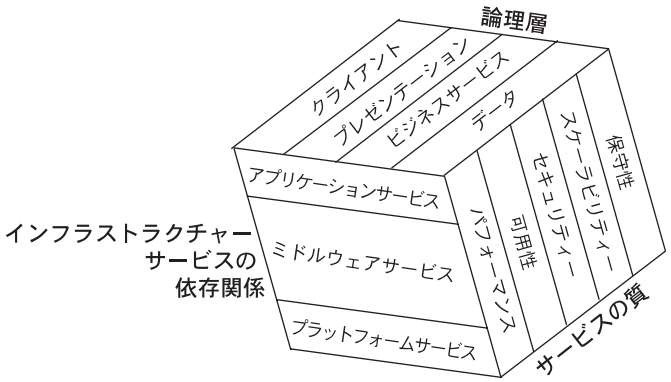

図 2–1 Java ES ソリューションアーキテクチャーの次元

これらの 3 つの次元によって、ソフトウェアソリューションに必要なサービス機能とサービス品質の実現に必要な、アプリケーションコンポーネントとインフラストラクチャーコンポーネントの両方の、ソフトウェアコンポーネント間の関係を統合する単一のフレームワークを表します。

次の各項では、3 つの各次元を個別に説明し、その後、3 つの次元を 1 つのフレームワークに統合して説明します。

次元 1: インフラストラクチャーサービスの依存関係

分散型のエンタープライズアプリケーションの対話型ソフトウェアコンポーネントには、基本となるインフラストラクチャーサービスが必要です。これに基づいて、分散しているコンポーネント間で相互に通信したり、それぞれの動作を調整したり、セキュリティー保護されたアクセスを実装することなどが可能になります。ここでは、これらのインフラストラクチャーサービスを提供するためにいくつかの Java ES コンポーネントが果たす主な役割について説明します。

インフラストラクチャーサービスレベル

分散型のソフトウェアシステムを設計する場合、そのほとんどがカスタム開発コンポーネントで構成されるか、または追加設定の必要ない Java ES コンポーネントで構成されるかにかかわらず、多数のインフラストラクチャーサービスを組み込む必要があります。これらのサービスは、多数のレベルで機能します。

ソリューションアーキテクチャーのインフラストラクチャーサービスの依存関係を、図 2–2 に示します。この図に示されているレベルは、図 1–1 のインフラストラクチャーサービス層を詳細化したものです。図 2–2 のサービス階層とサービス間の依存関係が、ソリューションの論理アーキテクチャーの重要な次元を構成します。これらのインフラストラクチャーサービスは、JavaESシステムサービスコンポーネント(20 ページの「システムサービスコンポーネント」を参照) の主な論理的根拠になります。

一般的に、次の図に示したサービスは、大きく3つのグループに分けられます。下位レベルのプラットフォームサービス、上位レベルのアプリケーションサービス、およびミドルウェアサービスのグループです。なお、ミドルウェアサービスという名前は、ほかの 2 つのグループの間にあることから付けられたものです。

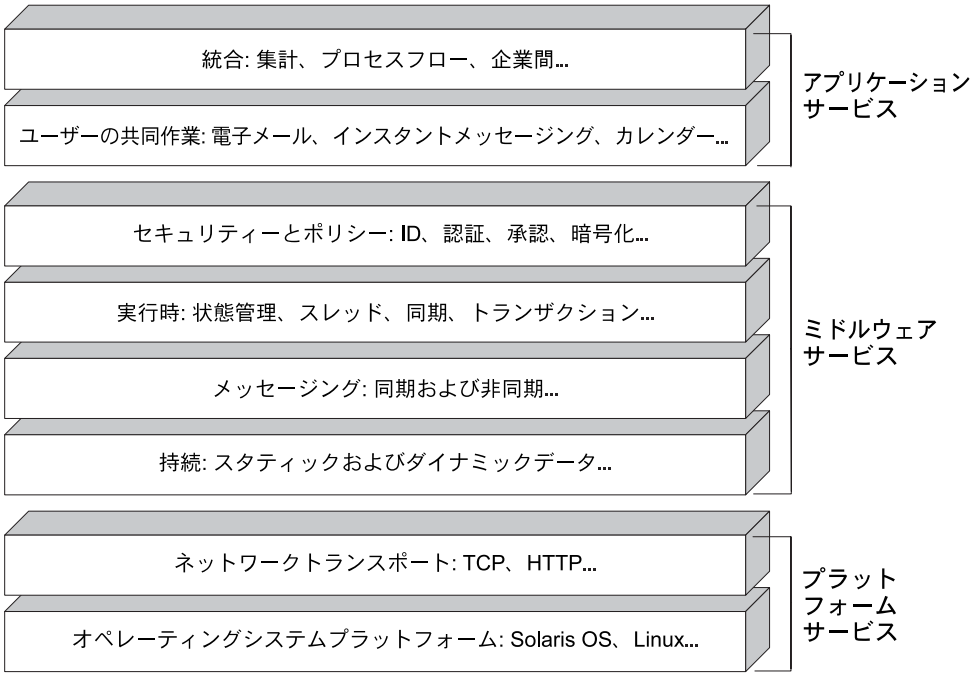

図 2–2　次元 1: インフラストラクチャーサービスレベル

次に、さまざまなインフラストラクチャーサービスレベルについて説明し、関連する場合には Java プログラミング言語のアーティファクトの参考情報も示します。図 2–2 に示された各サービスレベルについて、最下位レベルから順に説明します。

- オペレーティングシステムプラットフォーム: コンピュータ上で実行されるすべてのプロセスに対し、基本的なサポートを提供します。オペレーティングシステムは、メモリー、スレッド、および Java 仮想マシン (JVM™) のサポートに必要なその他のリソースに加えて、物理デバイスも管理します。
- ネットワークトランスポート: 異なるコンピュータ上で実行されている分散型アプリケーションコンポーネント間での通信に必要な、基本的なネットワークサポートを提供します。これらのサービスには、TCP や HTTP などのプロトコルに対するサポートも含まれます。上位レベルのその他の通信プロトコル (「メッセージングレベル」を参照) は、これらの基本的なトランスポートサービスに依存しています。
- 持続: 静的データ (ユーザー、ディレクトリ、設定情報など) と動的アプリケーションデータ (頻繁に更新される情報) の両方に対するアクセスや格納に必要なサポートを提供します。
- メッセージング: アプリケーションコンポーネント間の同期および非同期の通信に対するサポートを提供します。同期メッセージングでは、メッセージをリアルタイムで送受信します。これには、J2EE コンポーネント間のリモートメソッドの

呼び出し (RMI) や Web サービスとの SOAP 対話も含まれます。非同期メッセージングの通信では、直後に受信するコンシューマの準備状況に関係なく、メッセージを送信します。JMS (Java Message Service) や ebXML などの非同期メッセージングの仕様では、信頼性の保証およびその他のメッセージング方式をサポートします。

- 実行時: J2EE モデルや CORBA モデルなどの分散型のコンポーネントモデルで必要となるサポートを提供します。実行時サービスには、密接に結合された分散型コンポーネントに必要なリモートメソッドの呼び出しの他に、コンポーネントの状態 (ライフサイクル) の管理、スレッドプールの管理、同期 (相互排他ロック)、持続サービス、分散するトランザクションの監視、分散する例外の処理などが含まれます。J2EE 環境では、これらの実行時サービスはアプリケーションサーバーまたは Web サーバーの EJB、Web、およびメッセージ駆動型 Bean コンテナから提供されます。
- セキュリティーとポリシー: アプリケーションリソースへのセキュリティー保護されたアクセスに必要なサポートを提供します。これらのサービスには、グループまたはロールに基づく分散型リソースへのアクセスを制御するポリシーのサポートや、シングルサインオン 機能が含まれます。シングルサインオンを使用すると、ある分散型システム内の 1 つのサービスに対するユーザー認証を、同じシステム内の他のサービス (J2EE コンポーネント、ビジネスサービス、Web サービスなど) に自動的に適用できます。
- ユーザーの共同作業: ユーザー間の直接通信およびエンタープライズ内とインターネット環境内でのユーザー間の共同作業に対するサポートで重要な役割を果たすサービスを提供します。これらのサービスは、一般的にスタンドアロンサーバー (電子メールサーバーやカレンダサーバーなど) から提供されるアプリケーションレベルのビジネスサービスです。
- 統合: 既存のビジネスサービスを集約するサービスを提供します。サービスにアクセスするための共通インタフェースを、ポータルと同様に提供するか、またはこれらのサービスを本稼動ワークフロー内で調整するプロセスエンジンを使用して統合することによって提供します。統合は、異なる企業間における企業間 (B2B) 統合として行なわれることもあります。

図 2–2 に示したサービスレベルは、最下位レベルのオペレーティングシステムサービスから最上位レベルのアプリケーションサービスや統合サービスまで、インフラストラクチャーサービス間の依存関係を反映しています。通常、各サービスはその下にあるサービスに依存し、その上にあるサービスをサポートします。ただし、図 2–2 は、インフラストラクチャーサービスの厳密な階層を表しているわけではありません。上位レベルのサービスは、中間のレベルに依存せずに、下位レベルのサービスと直接対話することができます。たとえば、一部の実行時サービスは、中間にあるサービスレベルを介さずに、プラットフォームサービスに直接依存する場合があります。さらに、監視サービスや管理サービスなどのその他のサービスレベルも、ここでの概念的な説明に含まれることがあります。

Java ES インフラストラクチャーサービスコンポーネント

Java ES コンポーネントは、図 2–2 に示した分散型インフラストラクチャーサービスレベルを実装したものです。システムサービスコンポーネントのさまざまなレベル内における位置関係を、次の図に示します。

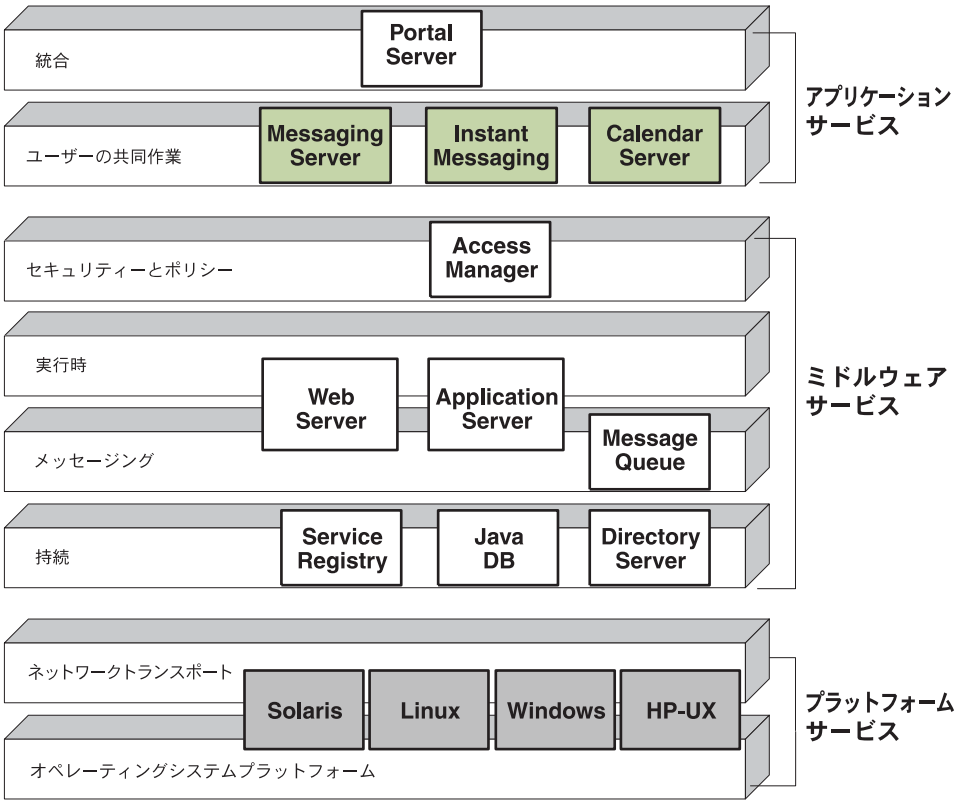

図 2–3　Java ES システムサービスコンポーネント

注 – 図の中で影付きのボックスの部分は、Java ES に含まれていないコンポーネントを示します。ユーザーの共同作業コンポーネントは、Java ES の一部ではありませんが、多くの場合 Java ES コンポーネントとともに配備され、Java ES アーキテクチャー内で使用されます。これらのコンポーネントは Sun Java Communications Suite の一部であり、このマニュアルでは例示目的でのみ参照されています。また、オペレーティングシステムプラットフォームも正式には Java ES の一部ではありませんが、Java ES コンポーネントがサポートされているオペレーティングシステムプラットフォームを示すために図に含まれています。

Java ES インフラストラクチャーサービスの依存関係

一般に、図 2–3 に示した各 Java ES システムサービスコンポーネントは、インフラストラクチャー内でその下にあるコンポーネントに依存し、その上にあるコンポーネントをサポートします。それらの依存関係とサポートの関係は、論理アーキテクチャーを設計する上で重要な要素です。

次の表に、Java ES システムサービスコンポーネント間の具体的な関係を示します。この表では図 2–3 と同様に、最上位から順に記載しています。

表 2–1　Java ES システムサービスコンポーネント間の関係

構成要素	依存するコンポーネント	サポートするサーバー
Portal Server	Application Server または Web Server Access Manager Directory Server 対応するチャネルを使用するように設定されている場合: Calendar Server、Messaging Server、および Instant Messaging[1]	なし
Access Manager	Application Server または Web Server Directory Server	Portal Server シングルサインオン用に設定されている場合: Calendar Server、Messaging Server、および Instant Messaging
Application Server	Message Queue Directory Server (管理オブジェクト用)	Portal Server Access Manager
Message Queue	Directory Server (管理オブジェクト用)	Application Server
Web Server	Access Manager (アクセス制御用)	Portal Server Access Manager
Directory Server	なし	Portal Server Access Manager Calendar Server Messaging Server Instant Messaging

[1] Calendar Server、Messaging Server、および Instant Messaging コンポーネントは、Sun Java Communications Suite の一部として利用可能です。

表 2–1　Java ES システムサービスコンポーネント間の関係　　(続き)

構成要素	依存するコンポーネント	サポートするサーバー
Service Registry	Java DB	Applcation Server に基づくコンポーネント
Java DB	なし	Service Registry

次元 2: 論理層

分散型エンタープライズアプリケーションの相互に作用するソフトウェアコンポーネントは、いくつかの論理層に存在するとみなすことができます。これらの層は、提供するサービスの性質に基づいて、ソフトウェアコンポーネントの論理的および物理的な独立性を表しています。

ソリューションアーキテクチャーの論理層の次元を、次の図に示します。

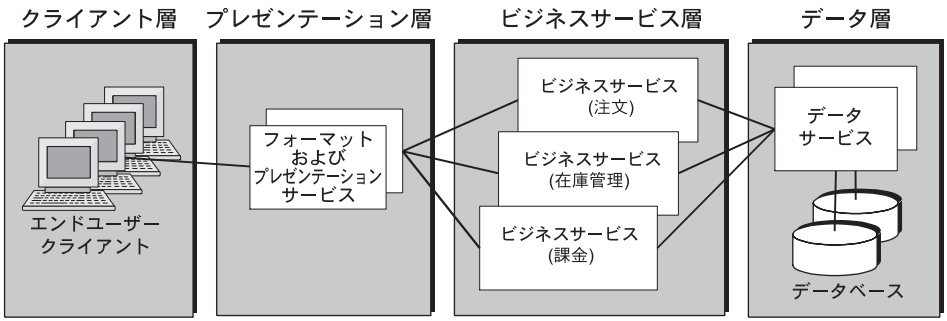

図 2–4　次元 2: 分散型エンタープライズアプリケーションの論理層

論理層アーキテクチャーは基本的に、図 1–1 の分散型エンタープライズアプリケーション層を表します。37 ページの「インフラストラクチャーサービスレベル」で説明した Java ES システムサービスコンポーネントは、図 2–4 に示したすべての論理層に含まれるアプリケーションコンポーネントをサポートします。論理層の概念は、主にカスタムエンタープライズアプリケーションに当てはまりますが、Sun Java Communications Suite コンポーネントおよび一部のポータルサービスが提供する共同作業サービスにも当てはまります。

論理層について

ここでは、図 2–4 に示した 4 つの論理層について簡単に説明します。この説明では、J2EE プラットフォームコンポーネントモデルを使用して実装されたアプリケーションコンポーネントを取り上げます。ただし、CORBA など、その他の分散型のコンポーネントモデルも、このアーキテクチャーをサポートしています。

- クライアント層: クライアント層は、エンドユーザーがユーザーインタフェースを通して直接アクセスするアプリケーションロジックによって構成されます。クライアント層のロジックには、ブラウザベースのクライアント、デスクトップコンピュータ上で実行される Java™ コンポーネント、または携帯型のデバイス上で実行される J2ME™ プラットフォーム (Java 2 Platform, Micro Edition) モバイルクライアントが含まれることがあります。
- プレゼンテーション層: プレゼンテーション層は、クライアント層に配信するデータを準備し、クライアント層からのバックエンドビジネスロジックへの配信の要求を処理するアプリケーションロジックで構成されます。プレゼンテーション層のロジックは通常、Java サーブレットコンポーネントや JSP コンポーネントなどの J2EE コンポーネントから構成されます。これらのコンポーネントは、HTML 形式や XML 形式の配信用データを準備したり、処理要求を受信したりします。また、この層にはポータルサービスが含まれることもあります。ポータルサービスは、ビジネスサービス層のビジネスサービスへのパーソナル化、セキュリティー保護、およびカスタマイズされたアクセス機能を提供できます。
- ビジネスサービス層: ビジネスサービス層は、アプリケーションの主要機能を実行するロジックで構成されます。それらの機能には、データの処理、ビジネスルールの適用、複数ユーザーの調整、データベースや旧バージョンシステムのような外部リソースの管理などがあります。通常、この層は、Java オブジェクト、EJB コンポーネント、メッセージ駆動型 Beans などの J2EE 分散型コンポーネントモデルに準拠する、密接に結合されたコンポーネントで構成されます。個々の J2EE コンポーネントは、在庫情報サービスや税額計算サービスなどの複雑なビジネスサービスを配信するように組み立てることができます。個々のコンポーネントやサービスアセンブリは、サービス指向アーキテクチャーモデル内で緩やかに結合された、SOAP (Simple Object Access Protocol) インタフェース標準に準拠した Web サービスとしてカプセル化できます。ビジネスサービスは、エンタープライズカレンダサーバーやメッセージングサーバーのようなスタンドアロンのサーバーとして構築することもできます。
- データ層: データ層は、ビジネスロジックで使用される持続データを提供するサービスで構成されます。このデータは、データベース管理システムに格納されているアプリケーションデータであることも、LDAP (Lightweight Directory Access Protocol) データストアに格納されているリソース情報およびディレクトリ情報であることもあります。このデータサービスには、外部ソースからのデータ供給や旧バージョンのコンピューティングシステムからアクセス可能なデータが含まれることがあります。

論理的および物理的な独立性

図 2–4 に示したアーキテクチャーの次元は、コンポーネントの論理的および物理的な独立性に重点を置いており、4 つの異なる層として表現されています。これらの層は、ネットワーク環境内のさまざまなコンピュータ間でのアプリケーションロジックの区分を表しています。

- 論理的な独立性: アーキテクチャーモデルの 4 つの層は、論理的な独立性を表しています。ある層 (ビジネスサービス層など) のアプリケーションロジックは、ほかの層のロジックとは無関係に変更することができます。プレゼンテーション層またはクライアント層のロジックを変更またはアップグレードしなくても、ビジネスロジックの実行を変更できます。このような独立性により、たとえば、ビジネスサービスコンポーネントを変更しなくても、新しいタイプのクライアントコンポーネントを導入できます。
- 物理的な独立性: 4 つの層は、物理的な独立性も表しています。各層内のロジックは、それぞれ異なるハードウェアプラットフォーム上に (つまり、異なるプロセッサ構成、チップセット、およびオペレーティングシステム上に) 配備できます。この独立性により、個々のコンピューティング要件および最大限に拡張されたネットワーク帯域幅に最適に対応するように、コンピュータ上の分散型アプリケーションコンポーネントを実行することが可能になります。

  アプリケーションコンポーネントやインフラストラクチャーコンポーネントをハードウェア環境 (つまり、配備アーキテクチャー) にマッピングする方法は、ソフトウェアソリューションの規模や複雑さに応じて、さまざまな要因によって決定されます。小規模の配備の場合は、配備アーキテクチャーに含まれるのは数台のコンピュータのみである場合があります。大規模の配備の場合は、ハードウェア環境へのコンポーネントのマッピングには、各コンピュータの速度と演算能力、ネットワークリンクの速度と帯域幅、セキュリティーおよびファイアウォールの考慮事項、高可用性およびスケーラビリティーのためのコンポーネントのレプリケーションの方針などの要素を考慮する場合があります。

システムコンポーネントに適用される層によるアーキテクチャー

図 2–3 で示したように、Java ES インフラストラクチャーサービスコンポーネントは、分散型ソフトウェアソリューションの基盤となるインフラストラクチャーサポートを提供します。それらのソリューションの一部には、Sun Java Communications Suite コンポーネントおよび一部の Java ES コンポーネントが提供するアプリケーションレベルサービスが含まれます。それらのソリューションは、論理層の設計方法を使用します。

たとえば、Messaging Server が提供する電子メール通信サービスは、いくつかの論理的に区別された Messaging Server 設定を使用して実装されます。これらの異なる設定はそれぞれ、一連の異なるサービスを提供します。メッセージングソリューションを設計するときには、これらの異なる設定は、次の図に示すように別々の論理層に存在する個々のコンポーネントとして表されます。図中のコンポーネント間を結ぶ線は対話を表しています。

注 – 次の図は、完全な論理アーキテクチャーを表しているわけではありません。図を簡略化するために、多くの Java ES コンポーネントが省略されています。

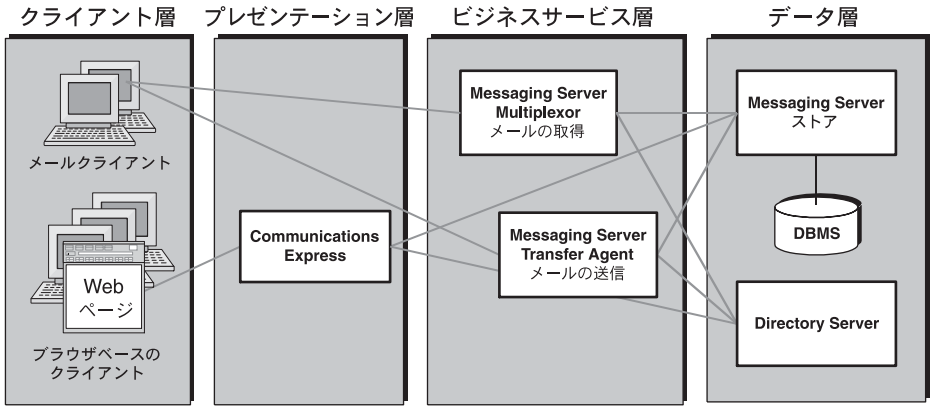

図 2–5　Messaging Server: 層によるアーキテクチャーの例

注 – コミュニケーションコンポーネントは Java ES の一部ではありませんが、多くの場合 Java ES コンポーネントとともに配備され Java ES アーキテクチャー内で使用されます。これらのコミュニケーションコンポーネントは Sun Java Communications Suite の一部であり、このマニュアルでは例示のためにのみ参照されています。

Messaging Server 機能を論理的に異なる層に分けることにより、Messaging Server の論理的に区別された設定を物理環境内の異なるコンピュータに配備できます。物理的に分離すると、サービス品質の要件 (45 ページの「次元 3: サービスの品質」を参照) を柔軟に満たすことができます。たとえば、インスタンスごとに異なる可用性ソリューションを提供したり、Messaging Server 機能ごとに異なるセキュリティーを実装できます。

次元 3: サービスの品質

すでに説明したアーキテクチャーの 2 つの次元 (インフラストラクチャーサービスの依存関係および論理層) では、主にアーキテクチャーの論理的な面に焦点を当てました。つまり、サービスをエンドユーザーに配信するために、どのようなコンポーネントが必要で、どのような方法で対話するのかについてです。配備される別のソリューションで同様に重要な次元は、サービス品質の要件を満たすためのソリューションの機能です。

ソリューションアーキテクチャーのサービスの品質の次元は、Java ES サービス品質コンポーネントが果たす役割を明らかにします。

サービス品質

ビジネスの運営におけるインターネットサービスや e コマースサービスの重要性が増しているため、これらのサービスのパフォーマンス、可用性、セキュリティー、スケーラビリティー、および保守容易性は、高いパフォーマンスを備えた大規模な配備アーキテクチャーの重要なサービス品質要件になりました。

優れたソフトウェアソリューションを設計するには、適切なサービス品質要件を設定し、それらの要件を満たすアーキテクチャーを設計する必要があります。いくつかの重要なサービス品質により、サービス品質要件を指定します。これらのサービス品質を次の表に要約してあります。

表 2–2 ソリューションアーキテクチャーに影響するサービス品質

システムサービス品質	説明
パフォーマンス	ユーザーの負荷条件に関する応答時間の測定
可用性	システムのリソースやサービスがエンドユーザーにアクセス可能になる頻度 (システムの「稼働時間」) の測定。
セキュリティー	システムとユーザーの整合性を説明する、要因の複雑な組み合わせ。セキュリティーには、システムの物理セキュリティー、ネットワークセキュリティー、アプリケーションおよびデータのセキュリティー (ユーザーの認証および承認)、またセキュリティー保護された情報のトランスポートも含まれます。
スケーラビリティー	配備されたシステムに対して、随時、容量を拡張する機能。通常、スケーラビリティーにはシステムへのリソースの追加が関連しますが、配備アーキテクチャーの変更は含まれません。
潜在能力	システムでリソースを追加せずに、異常なピーク負荷使用を処理する機能
保守容易性	システムの監視、発生した問題の解決、ハードウェアおよびソフトウェアコンポーネントのアップグレードなどを含めた、配備されたシステムの保守の容易さ。

サービス品質の次元は、ソリューションの配備アーキテクチャーに、つまりアプリケーションコンポーネントとインフラストラクチャーコンポーネントが物理環境内でどのように配備されるかに、大きな影響を与えます。

配備アーキテクチャーに影響を与えるサービス品質は、互いに密接に関係しています。あるシステム品質に対する要件がほかのサービス品質の設計に影響を与えることが、よくあります。たとえば、セキュリティーのレベルを高めることは、パフォーマンスに影響し、それが可用性に影響する可能性があります。冗長性を使用して可用性の問題に対処するためにコンピュータを追加すると、保守コスト (保守容易性) に影響を与える可能性があります。

ビジネスの要件と制約の両方を満たす配備アーキテクチャーを設計するには、複数のサービス品質が相互に関連する仕組み、およびこれらのかね合いを理解しておくことが重要です。

Java ES サービス品質コンポーネント

システムサービスコンポーネントまたは分散型アプリケーションコンポーネントが提供するサービス品質を向上するために、主にいくつかの Java ES コンポーネントが使用されます。これらのソフトウェアコンポーネントは、多くの場合、ロードバランサやファイアウォールなどのハードウェアコンポーネントとともに使用されます。

22 ページの「サービス品質コンポーネント」で紹介した Java ES サービス品質コンポーネントについて、次に要約します。

- 可用性コンポーネント: 配備されたソリューションがほぼ連続的に稼働することを可能にします。
- アクセスコンポーネント: システムサービスへのインターネットからのセキュリティー保護されたアクセスを可能にし、また多くの場合ルーティング機能も提供します。
- 監視コンポーネント: Java ES コンポーネントについての情報をリアルタイムで提供します。

次の表は、アーキテクチャーの観点からの最も重要な Java ES サービス品質コンポーネントと、それらの各コンポーネントが最も影響を及ぼすシステム品質を一覧表示しています。

表 2–3　サービス品質コンポーネントと影響を受けるシステム品質

構成要素	影響を受けるシステム品質
High Availability Session Store	有効な Solaris のバージョン
Monitoring Console	サービス利用可能性
Portal Server Secure Remote Access	セキュリティー スケーラビリティー
Sun Cluster	有効な Solaris のバージョン スケーラビリティー
Sun Cluster Geographic Edition	有効な Solaris のバージョン スケーラビリティー

表 2–3 サービス品質コンポーネントと影響を受けるシステム品質 (続き)

構成要素	影響を受けるシステム品質
Web Proxy Server	セキュリティー 性能 スケーラビリティー

Sun Cluster ソフトウェア

Sun Cluster ソフトウェアは、高可用性サービスおよび高スケーラビリティーサービスを、Java ES コンポーネントおよび Java ES インフラストラクチャーがサポートするアプリケーションに対して提供します。クラスタとは緩やかに結合された一連のコンピュータのことであり、サービス、システムリソース、およびデータの単一のクライアントビューを一括して提供します。クラスタの内部では、冗長コンピュータ、インターコネクト、データ記憶域、およびネットワークインタフェースを使用して、クラスタベースのサービスおよびデータに高可用性を提供します。

Sun Cluster ソフトウェアは、メンバーノードおよびその他のクラスタリソースの健全性を継続的に監視します。障害が発生した場合、Sun Cluster ソフトウェアは監視対象のリソースのフェイルオーバーを開始するために介入し、内部の冗長性を使用して、リソースへのほぼ連続的なアクセスを可能にします。

Sun Cluster データサービスパッケージ (Sun Cluster エージェントと呼ばれることもある) が、すべての Java ES システムサービスコンポーネントに利用できます。カスタム開発されたアプリケーションコンポーネント用のエージェントを記述することもできます。

Sun Cluster ソフトウェアによる制御が行われるので、スケーラブルなサービスも提供できます。クラスタのグローバルファイルシステムおよびクラスタ内の複数のノードの機能を利用して、インフラストラクチャーサービスやアプリケーションサービスを実行することにより、サービスの複数の並行インスタンス間で、これらのサービスに対して増加する要求のバランスを取ることができます。したがって、正しく設定されていれば、Sun Cluster ソフトウェアは分散型のエンタープライズアプリケーションに高可用性とスケーラビリティーの両方を提供できます。

Sun Cluster 環境をサポートするのに必要な冗長性のために、ソリューションに Sun Cluster を含めると、物理環境に必要なコンピュータやネットワークリンクの数がかなり増えます。

Sun Cluster の可用性サービスは、ほかの Java ES コンポーネントが提供するサービスとは異なり、分散型のピアツーピアサービスです。したがって、Sun Cluster ソフトウェアは、クラスタ内のすべてのコンピュータにインストールする必要があります。

Sun Cluster ソフトウェアへの拡張は Sun Cluster Geographic Edition によって提供されており、地理的に離れた場所にある複数のクラスタおよびクラスタ間のデータを複製するインフラストラクチャーを使用して、予期しない中断からアプリケーションを保護します。

注 – Sun Cluster および Sun Cluster Geographic Edition は、 Solaris™ Operating System (Solaris OS) でのみサポートされます。

アーキテクチャーの 3 つの次元の統合

図 2–1 に示し、前のいくつかの節で説明したアーキテクチャーの 3 つの次元を 1 つのものとして捉えると、分散型ソフトウェアソリューションを設計するためのフレームワークが得られます。3 つの次元 (インフラストラクチャーサービスの依存関係、論理層、およびサービス品質) は、ソリューションアーキテクチャーで Java ES コンポーネントが果たす役割を明らかにします。

各次元は、それぞれアーキテクチャーの異なる面を表します。すべてのソリューションアーキテクチャーが、それらのすべての次元を考慮する必要があります。たとえば、ソリューションアーキテクチャーの各論理層の分散型コンポーネント (次元 2) は、適切なインフラストラクチャーコンポーネント (次元 1) と適切なサービス品質コンポーネント (次元 3) のサポートが必要です。

同様に、ソリューションアーキテクチャーに含まれるコンポーネントは、アーキテクチャーの次元ごとに異なる役割を果たします。たとえば、Directory Server はデータ層のバックエンドコンポーネント (次元 2) と持続サービスのプロバイダ (次元 1) の両方とみなされます。Directory Server はこれらの 2 つの次元の中心に位置するため、この Java ES コンポーネントには、サービスの品質の問題 (次元 3) が最も重要です。Directory Server の障害はビジネスシステムに多大な影響を及ぼすので、このコンポーネントの高可用性設計は非常に重要です。Directory Server はユーザーや設定に関する機密情報の格納に使用されるため、このコンポーネントのセキュリティーの設計も非常に重要です。

Java ES コンポーネントに関するこれらの 3 つの次元の相互作用は、ソリューションの論理アーキテクチャーとソリューションの配備アーキテクチャーの設計に影響します。

35 ページの「Java ES アーキテクチャーのフレームワーク」で説明したアーキテクチャーフレームワークに基づく詳細な設計方法論は、このマニュアルでは説明されていません。ただし、3 次元のアーキテクチャーフレームワークは、Java Enterprise System に基づいたソフトウェアソリューションの配備を理解するのに重要な設計の面を明らかにします。

Java ES ソリューションアーキテクチャーの例

Java ES は、広範なソフトウェアソリューションをサポートします。多くのソリューションは、開発作業を行わずに、Java ES に含まれるコンポーネントを使ってすぐに設計および配備できます。その他のソリューションには、新しいビジネスまたはプレゼンテーションサービスを提供するカスタム J2EE コンポーネントの開発を必要とする、広範な開発が必要になる場合があります。これらのカスタムコンポーネントを SOAP インタフェース標準に準拠する Web サービスとしてカプセル化することができます。多くのソリューションは、この 2 つの方法を組み合わせて使用します。

ここでは、前の節のアーキテクチャーの概念に基づき、Java ES がすぐに使用できるソリューションをどのようにしてサポートするのかを示す例を取り上げます。

エンタープライズ通信のシナリオ

企業は、一般に従業員間の通信、特に電子メールサービスやカレンダサービスをサポートする必要があります。そのような企業は、内部の Web サイトやその他のリソースに対する個人向けにカスタマイズされたアクセスを企業全体にわたる認証および承認サービスに基づいて従業員に提供すると便利であるとみなします。また、それらの企業は、シングル Web サインオンによりすべてのエンタープライズサービスへのアクセスが可能になるように、すべてのエンタープライズサービスで従業員のアイデンティティーを追跡することを望みます。

多数のビジネス要件の一部の例である、このような特定のビジネス要件を次の表にまとめてあります。

表 2–4　ビジネス要件の要約: 通信のシナリオ

ビジネス要件	説明	必要なサービス
シングルサインオン	セキュリティー保護されたエンタープライズリソースおよびサービスへのアクセス (Web アクセスはシングルサインオンの単一の ID に基づく)。	アイデンティティーサービス
メッセージング カレンダ	従業員と外部との電子メールメッセージング。 電子的な従業員用のカレンダ機能や会議の設定。	通信および共同作業サービス
ポータルアクセス	電子メール、カレンダ、内部 Web ページなどの通信サービスへの単一の Web ベースの個人向けにカスタマイズされたアクセスポイント。	ポータルサービス

さらに、企業には、これらのサービスを提供するソフトウェアシステムのパフォーマンス、可用性、ネットワークセキュリティー、およびスケーラビリティーに関する要件があります。

シナリオ例の論理アーキテクチャー

表 2–4 で示したポータルサービス、通信サービス、およびアイデンティティーサービスを Java ES コンポーネントおよび Sun Java Communications Suite コンポーネント (Messaging Server、Calendar Server、Instant Messaging など) を使用して配信するための論理アーキテクチャーを次の図に示します。このアーキテクチャーは、Messaging Server の論理的に異なる設定を、それぞれが異なるサービスを提供するために、別々のコンポーネントとして扱います。

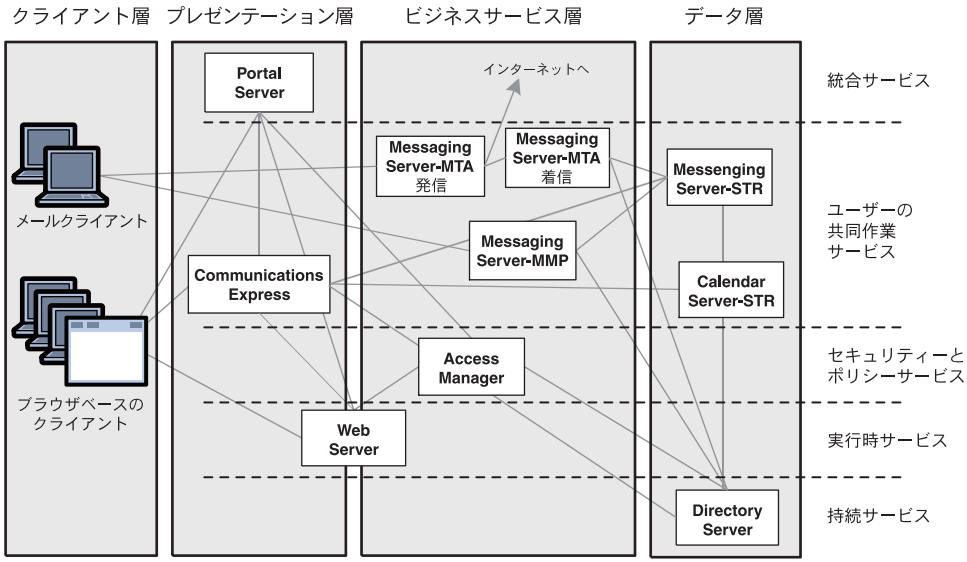

図 2–6 エンタープライズ通信のシナリオの論理アーキテクチャー

コンポーネントは、標準の論理層を表す横の次元、またインフラストラクチャーサービスレベルを表す縦の次元に配置されています。コンポーネント間の対話は、分散型インフラストラクチャーサービスとしての各コンポーネントの機能 (インフラストラクチャーサービスレベル間の対話) または階層アプリケーションアーキテクチャー内の各コンポーネントの役割 (論理層内または論理層間の対話) に依存します。

このアーキテクチャーでは、Directory Server に格納されたユーザー情報にアクセスする Access Manager が、プレゼンテーション層の Portal Server およびその他の Web ベースコンポーネントに対する、シングルサインオン認証および承認の仲介役として機能します。Messaging Server コンポーネントには、データ層のメッセージストア (Messaging Server-STR)、ビジネスサービス層の送信および取得コンポーネント、プレゼンテーション層の HTTP アクセスコンポーネントおよび Communications Express が含まれます。

論理アーキテクチャーは、さまざまなコンポーネント間のインフラストラクチャーサービスの依存関係も示します。たとえば、Portal Server は、メッセージングチャネルとカレンダチャネルには Communications Express を利用し、認証および承認サービスには Access Manager を利用します。これらのコンポーネントは、今度は、ユーザー情報および設定データのために Directory Server を利用します。多数のコンポーネントは、Web Server が提供する Web コンテナサービスを必要とします。

Java ES ソリューションの論理設計の詳細については、『Sun Java Enterprise System Deployment Planning Guide 』を参照してください。

シナリオ例の配備アーキテクチャー

論理アーキテクチャーから配備アーキテクチャーに移行する際には、サービス品質要件が最も重要になります。たとえば、保護されたサブネットやファイアウォールを使用して、バックエンドデータへのセキュリティーバリアを設けることができます。多くのコンポーネントの可用性とスケーラビリティーの要件は、複数のコンピュータにコンポーネントを配備し、ロードバランサを使用してレプリケートしたコンポーネント間に要求を分散することによって満たすことができます。

ただし、より高い可用性要件の水準が適用された場合や大量のディスク容量が必要な場合は、他の可用性ソリューションの方が適しています。たとえば、Messaging Server ストアに Sun Cluster を、Directory Server にマルチマスターレプリケーションを使用できます。

Java ES ソリューションの配備設計の詳細については、『Sun Java Enterprise System Deployment Planning Guide 』を参照してください。

この章の重要な用語

この節では、この章で使用されている重要な技術用語について説明します。ここでは、Java ES の文脈でどのように使用されているかの説明に重点を置いています。

アプリケーションコンポーネント **(application component)**
: 特定のコンピューティング機能を実行し、ビジネスサービスをエンドユーザーまたはほかのアプリケーションコンポーネントに対して提供する、カスタム開発されたソフトウェア構成要素 (コンポーネント)。通常、アプリケーションコンポーネントは、CORBA や J2EE プラットフォームなどの分散型コンポーネントモデルに準拠しています。これらのコンポーネントを単独であるいは組み合わせて、Web サービスとしてカプセル化できます。

アーキテクチャー **(architecture)**	分散型アプリケーション (またはその他のソフトウェアシステム) の論理的および物理的な構築ブロックと、それら構築ブロック間の相互関係を示した設計。分散エンタープライズアプリケーション (distributed enterprise application)の場合、アーキテクチャー設計には一般的に、アプリケーションの論理アーキテクチャー (logical architecture)と配備アーキテクチャー (deployment architecture)の両方が含まれます。
ビジネスサービス **(business service)**	複数のクライアントに代わってビジネスロジックを実行するアプリケーションコンポーネント (application component)またはコンポーネントアセンブリ (したがって、マルチスレッド対応プロセスである)。また、ビジネスサービスは、Web サービス (web service)としてカプセル化された分散型コンポーネントアセンブリであってもかまいませんし、スタンドアロンのサーバーであってもかまいません。
クライアント	ソフトウェアサービスを要求するソフトウェア。クライアントは、別のサービスを要求するサービスであったり、エンドユーザーがアクセスする GUI コンポーネントであったりします。
配備アーキテクチャー **(deployment architecture)**	論理アーキテクチャー (logical architecture)から物理的なコンピューティング環境へのマッピングを示す上位レベルの設計。物理的な環境は、イントラネットまたはインターネット環境にあるコンピュータ、それらのコンピュータ間のネットワークリンク、ソフトウェアをサポートするために必要なほかの物理的デバイスなどで構成されます。
論理アーキテクチャー **(logical architecture)**	分散型アプリケーションの論理的な構築ブロックとそれら構築ブロック間の関係 (またはインタフェース) を記述した設計。論理アーキテクチャーには、分散型アプリケーションコンポーネント、およびこれらのアプリケーションのサポートに必要なインフラストラクチャーサービスの両方が含まれます。
サーバー	分散型のサービスまたは関連する一連のサービスを、外部インタフェースを使ってサービスにアクセスするクライアントに対して提供する、マルチスレッド対応のソフトウェアプロセス。ハードウェアのサーバーとは区別されます。
Web サービス **(web service)**	アクセス可能性、サービスのカプセル化、および検出に関する標準インターネットプロトコルに準拠したサービス。この標準インターネットプロトコルには、SOAP メッセージングプロトコル、WSDL (Web Service Description Language) インタフェース定義、および UDDI (Universal Description, Discovery and Integration) レジストリ標準が含まれます。

◆ ◆ ◆ 第 3 章

Java ES 統合機能

この章では、Java ES コンポーネントを 1 つのソフトウェアシステムに統合する際に主な役割を果たす機能を理解するための概念および技術的な背景について説明します。これらの機能は、別のインフラストラクチャー製品を手動で統合する場合と比較した、Java ES を使用した場合の利点を理解するのに役立ちます。

この章は、次の節で構成されます。

Java ES 統合インストーラ

Java ES のすべてのコンポーネントは、単一のインストーラを使ってインストールされます。Java ES インストーラは、Java ES ソフトウェアを特定のホストシステムに転送します。インストーラを使用すると、コンピューティング環境内の特定のホストに必要な数の Java ES コンポーネントを選択してインストールできます。インストーラは、インストール対象の特定の Java ES コンポーネントに応じて、一定のインストール時の設定も可能にします。

Java ES インストーラそのものは、分散インストールを実行しません。分散型 Java ES ソフトウェアソリューションを配備するには、Java ES インストーラを使用して、環境内のコンピュータごとに 1 台ずつ、適切なコンポーネントをインストールします。配備アーキテクチャーおよびコンポーネントの依存関係に基づいて、インストールセッションおよび設定手順を適切な順序で使用する必要があります。

このインストーラは、グラフィカルモードおよびテキストベースモードの両方で対話的に実行できる一方、パラメータ駆動型のサイレントインストールモードでも実

行できます。このインストーラは英語のほかに、次の言語をサポートします。フランス語、ドイツ語、日本語、韓国語、スペイン語、簡体字中国語、および繁体字中国語です。

このセクションでは、Java ES の統合インストーラについて説明します。詳細については、『Sun Java Enterprise System 5 Update 1 Installation Guide for UNIX 』を参照してください。

既存のソフトウェアのチェック

インストーラは、インストール先のホストを検証し、すでにインストールされている Java ES コンポーネントを確認します。次に、インストーラは複数のレベルでチェックを実行して、既存のすべてのコンポーネントが、相互に正常に機能するように適切なリリースレベルにあるかどうかを確認します。これにより、互換性がなく、アップグレードまたは削除する必要があるソフトウェアコンポーネントが表示されます。

同様に、インストーラは、J2SE や NSS など、すでにインストールされている Java ES 共有コンポーネントを確認し、互換性のない共有コンポーネントを検出すると、その一覧が表示されます (23 ページの「共有コンポーネント」を参照)。インストールを続行すると、インストーラによって、自動的に共有コンポーネントが新しいバージョンにアップグレードされます。

依存性の確認

インストーラは、コンポーネントのチェックを広範囲に行い、ユーザーが選択したインストールコンポーネントが正しく連携して動作するか検証します。多くのコンポーネントには、ほかのコンポーネントに対する依存性があります。このため、インストールするコンポーネントを選択すると、インストーラでは選択したコンポーネントに依存関係のあるコンポーネントおよびサブコンポーネントが自動的に選択されます。選択したコンポーネント間にローカルな依存関係がある場合、このコンポーネントを選択解除することはできません。ただし、ローカルの依存関係ではない場合は、警告メッセージが表示されますが、別のホストコンピュータ上のコンポーネントによって依存関係が満たされるという前提に基づき、操作は続行できます。

初期設定

多くの Java ES コンポーネントは、初期設定をしてから起動する必要があります。コンポーネントによっては、Java ES インストーラでこの初期設定を実行できます。

インストーラで「今すぐ設定」オプションを選択すると、この初期設定を実行できます。「あとで設定」オプションを選択して初期設定を省略してソフトウェアをインストールすることもできますが、この場合はインストールの完了後にインストールした各コンポーネントの初期設定を明示的に行う必要があります。

インストーラが初期設定を行うようにする場合は、インストール時に必要な設定情報を入力します。具体的には、管理者 ID やパスワードなど、すべてのコンポーネント製品に共通する一連のパラメータ値を指定できます。

アンインストール

Java ES には、アンインストールプログラムも用意されています。このプログラムを使用して、Java ES のインストーラによってローカルコンピュータにインストールされたコンポーネントを削除できます。アンインストーラはローカルな依存関係を確認し、依存関係が検出された場合は警告メッセージを出力します。アンインストーラは Java ES 共有コンポーネントを削除しません。インストーラと同様、アンインストーラはグラフィカルモード、テキストベースモード、またはサイレントモードで実行できます。

システム監視サービス

Java ES には、システムサービスをリアルタイムで監視する監視機能が含まれています。監視は、Sun Java System Monitoring Framework (共有コンポーネント (shared component))、および Sun Java System Monitoring Console (製品コンポーネント) によって実装されます。Monitoring Framework は自動的に設定および有効にされ、インストールされた各 Java ES コンポーネントのデータを収集します。Monitoring Console は監視対象データを表示するために使用するグラフィカルインタフェースです。Monitoring Console は Java ES のインストール中に選択可能なコンポーネントで、Monitoring Framework は自動的にインストールされます。

監視とは、実行時データを収集、公開し、サービス品質基準の計算を行うプロセスで、それによりシステム管理者はパフォーマンスを評価したり、警告メッセージを受け取ることができます。実行時の運用中に、管理者は Monitoring Console と対話してパフォーマンス統計を表示したり、しきい値を設定して動的に監視したり、カスタムジョブを定義したり、警告を確認したりします。

統合されたアイデンティティーサービスとセキュリティーサービス

Java ES の重要な機能として、統合されたユーザーアイデンティティー管理と、統合された認証および承認フレームワークが挙げられます。この項では、Java ES から提供される、統合されたアイデンティティーサービスとセキュリティーサービスを理解するための技術的な背景について説明します。

単一アイデンティティー

Java ES 環境では、エンドユーザーに単一統合アイデンティティーが割り当てられています。この単一アイデンティティー (single identity)に基づいて、ポータル、Web ページなどの各種リソースへのアクセスや、メッセージング、カレンダ、インスタントメッセージングなどの各種サービスへのアクセスを、ユーザーに許可することができます。

この統合されたアイデンティティーおよびセキュリティー機能は、Directory Server、Access Manager、およびその他の Java ES コンポーネント間の緊密な連携動作に基づいています。

Java ES のサービスやリソースへのユーザーアクセスは、ユーザー固有の情報をユーザーリポジトリまたはディレクトリ内の単一のユーザーエントリ内に格納することで実現されます。通常、このユーザー固有の情報には、一意の名前とパスワード、電子メールアドレス、組織内のロール、Web ページの設定などのデータが含まれます。ユーザーエントリ内の情報を使用して、ユーザーを認証したり、特定リソースへのアクセスを承認したり、そのユーザーに各種のサービスを提供したりできます。

Java ES の場合、Directory Server が提供するディレクトリ内に、ユーザーエントリが格納されます。ユーザーが Java ES コンポーネントから提供されるサービスを要求すると、そのサービスでは Access Manager を使用してユーザーを認証し、特定リソースへのアクセスを承認します。要求されたサービスは、ユーザーのディレクトリエントリ内のユーザー固有の設定情報を確認します。サービスは、その情報を使用してユーザーが要求した作業を実行します。

次の図は、ユーザーの認証および承認を実行するため、またユーザーに対してサービスを提供するためのユーザーエントリへのアクセスを示しています。

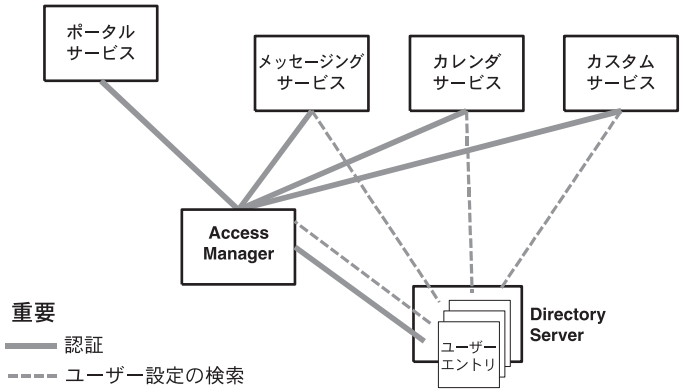

図 3–1 単一ユーザーエントリによる多数のサービスのサポート

このシステムによって可能になる機能の 1 つに、ユーザーが Web ベースですべての Java ES サービスにサインオンできる機能、つまり他のシステムサービスに対しても自動的に認証される機能があります。シングルサインオンとして知られるこの機能は、Java ES が提供する強力な機能のうちの 1 つです。

認証とシングルサインオン

Java ES の認証および承認サービスは、Access Manager によって提供されます。Access Manager は、Directory Server 内の情報を使用することで、ユーザーと Java ES Web サービスまたは企業内のその他の Web ベースのサービスとの対話を仲介します。

Access Manager は、ポリシーエージェントと呼ばれる外部コンポーネントを使用します。ポリシーエージェントは、サービスをホストする Web サーバーまたは Access Manager によってセキュリティー保護されているリソースにプラグインとして追加されます。ユーザーがセキュリティー保護されたリソースに対する要求を送信したときに、ポリシーエージェントは Access Manager の代わりに対話を仲介します。Portal Server などの一部の Java ES コンポーネントでは、Access Manager SDK のサブコンポーネントによってポリシーエージェントの機能が提供されます。

認証

Access Manager には、HTTP または HTTPS を介して企業内の Web サービスへのアクセスを要求したユーザーのアイデンティティーを検証する認証サービスが含まれています。たとえば、企業の従業員が同僚の電話番号を調べる必要がある場合に、ブラウザを使用してその企業のオンライン電話帳にアクセスします。電話帳サービスにログインするには、ユーザーが自分のユーザー ID とパスワードを入力する必要があります。

認証が行われる順序を図 3–2 に示します。ポリシーエージェントは、電話帳へのログオン要求を仲介し (1)、認証サービスに要求を送信します(2)。認証サービスでは、Directory Server に格納されている情報と照合して、ユーザー ID とパスワードを確認します (3)。ログイン要求が有効と認められると、そのユーザーは認証され (4)、(5)、および (6)、企業の電話帳が従業員に表示されます (7)。ログイン要求が有効と認められない場合は、エラーが生成されて認証に失敗します。

認証サービスは、証明書ベースの HTTPS を介した認証もサポートします。

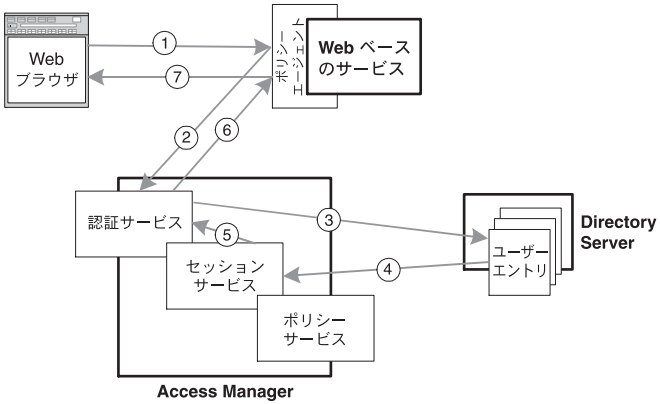

図 3–2　認証順序

シングルサインオン

前の段落で説明した認証の例では、重要な手順を説明しています。ユーザーの認証要求が検証されると、図 3–2 に示すように Access Manager のセッションサービスが起動されます (4)。セッションサービスによってセッショントークンが生成され、ここにユーザーのアイデンティティー情報およびトークン ID が保持されます (5)。セッショントークンはポリシーエージェントに返され (6)、さらに認証要求を発行したブラウザに Cookie として転送されます (7)。

認証されたユーザーがセキュリティー保護された別のサービスにアクセスしようとすると、ブラウザはセッショントークンを対応するポリシーエージェントに渡します。ポリシーエージェントはセッションサービスを使用して、そのユーザーの以前の認証が現在も有効であることを確認します。これにより、ユーザー ID およびパスワードの再入力を求めずに、そのユーザーに対して別のサービスへのアクセスが許可されます。

したがって、ユーザーは 1 回のサインオンだけで、Java ES が提供する Web ベースの複数のサービスから認証されます。シングルサインオン認証は、ユーザーが明示的にサインオフするか、セッションが期限切れになるまで有効です。

承認

Access Manager には、Java ES 環境の Web ベースのリソースに対するアクセス制御機能を提供するポリシーサービスも含まれています。ポリシーとは、特定の条件下で特定のリソースへのアクセスが承認されるユーザーを記述したルールのことです。次の図に、承認が行なわれる順序を示します。

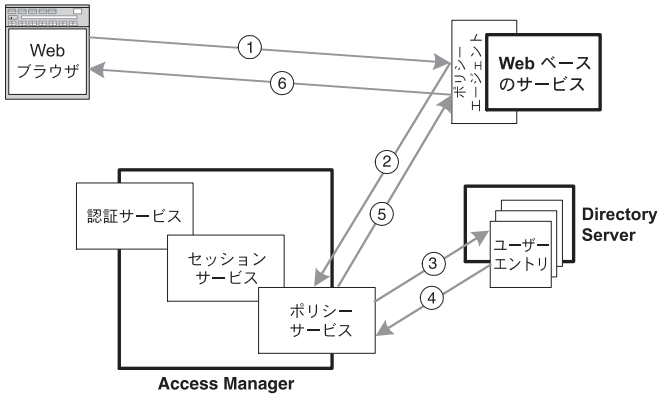

図 3–3　承認順序

認証されたユーザーが Access Manager によってセキュリティー保護されたリソースに対する要求を発行すると (1)、ポリシーエージェントはポリシーサービスに通知します (2)。ポリシーサービスでは、Directory Server の情報を使用して (3)、そのユーザーにリソースへのアクセスポリシーがあるかどうかを確認するために、そのリソースを管理するアクセスポリシーを評価します (4)。ユーザーにアクセス権がある場合は(5)、リソース要求が受け入れられます (6)。

Access Manager を使用すると、1 つの企業内のポリシーを定義、変更、付与、取り消し、および削除することができます。ポリシーは Directory Server に格納され、組織エントリのポリシー関連の属性によって設定されます。ロールは、ユーザーに対して定義したり、ポリシー定義に組み込んだりできます。

ポリシーを適用するのは、Access Manager のポリシーエージェントの役目です。ポリシーサービスがアクセス要求を拒否すると、ポリシーエージェントは要求を発行したユーザーがセキュリティー保護されたリソースへアクセスすることを禁止します。

この章の重要な用語

この節では、この章で使用されている重要な技術用語について説明します。ここでは、Java ES の文脈でどのように使用されているかの説明に重点を置いています。

ディレクトリ	データの書き込みよりも読み取りに対して最適化された特殊なデータベース。ほとんどのディレクトリは、業界標準のプロトコルである LDAP (Lightweight Directory Access Protocol) に基づいています。
ポリシー	特定の条件下で特定のリソースへのアクセスが承認されるユーザーを記述したルール。この規則は、組織内のユーザーグループまたはロールに基づいて規定することができます。
単一アイデンティティー **(single identity)**	Java ES ディレクトリ内の単一ユーザーエントリに基づいてユーザーが所有するアイデンティティー。この 1 つのユーザーエントリに基づき、ユーザーは、ポータルや Web ページなどの各種 Java ES リソースや、メッセージング、カレンダ、およびインスタントメッセージングなどのサービスにアクセスできます。
シングルサインオン	分散型システム内のあるサービスに対するユーザー認証を、システム内のほかのサービスにも自動的に適用できるようにする機能。

◆ ◆ ◆ 第 4 章

Java ES ソリューションのライフサイクル

この章では、Java ES ソリューションのライフサイクルの各フェーズに関する概念と用語について説明します。この章では、配備作業、特に配備設計作業と配備実装作業に重点を置いています。

この章では、ライフサイクルの各フェーズで行う作業について説明します。この章は、次の節で構成されます。

ソリューションのライフサイクルの作業

ソリューションのライフサイクルについては、Java ES ソフトウェアを使ってビジネスソリューションを実装するための標準的な方法として、第 1 章で紹介しました。簡単に参照できるようにライフサイクルの図を再度示してあります。

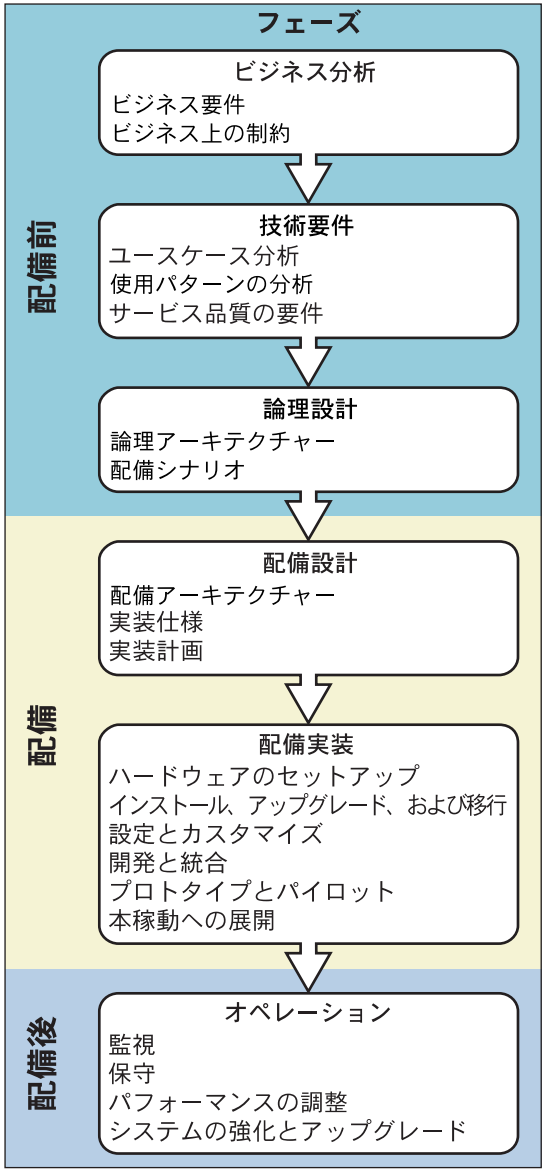

図4–1 ソリューションのライフサイクルの作業

配置前

ライフサイクルの配備前 (predeployment)フェーズでは、ビジネスに関するニーズの分析に基づいて配備シナリオ (deployment scenario)を作成します。配備シナリオは、配備を設計するための仕様としての役割を果たします。

配備前の作業は、図 4–1 に示すように 3 つのフェーズに分けられます。

- ビジネス分析: 提示する配備作業のビジネス上の目的を定義し、その目的を達成するために満たす必要があるビジネス上の要件および制約を記述します。
- 技術要件: ビジネス分析の結果に基づいてユースケースを作成します。ユースケースは、作成予定のソフトウェアシステムとユーザーとの対話をモデル化したものです。これらのユースケースに予測される使用パターンも決定します。ビジネス分析と使用分析の両方に基づいて、提案する配備が満たす必要のあるサービス品質要件 (表 2–2 を参照) を定式化します。
- 論理設計: 技術要件のフェーズで開発したユースケースを分析して、エンドユーザーにサービスを提供するために必要な Java ES インフラストラクチャーコンポーネントおよびカスタム開発されたアプリケーションコンポーネントを特定します。第 2 章で説明した概念に基づいて、論理アーキテクチャーを設計します。論理アーキテクチャーは、特定のソフトウェアソリューションのユースケースを実現するために必要なすべてのコンポーネント、およびコンポーネント間のすべての対話を示します。

次の図に示すように、論理アーキテクチャーは、パフォーマンス、可用性、セキュリティー、およびその他のサービス品質要件と組み合わされて、配備シナリオにカプセル化されます。ライフサイクルの配備前のフェーズの詳細については、『Sun Java Enterprise System Deployment Planning Guide 』を参照してください。

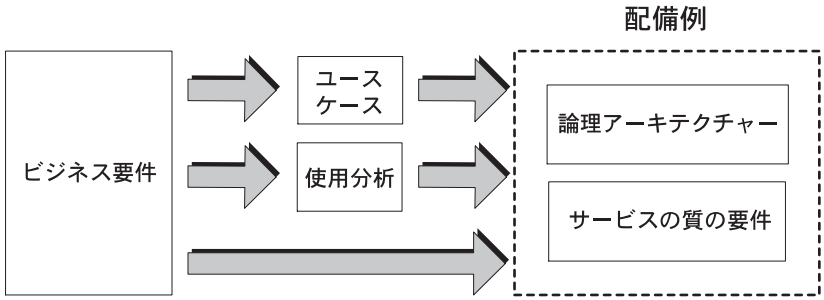

図 4–2 配備シナリオの指定

導入

ライフサイクルの配備のフェーズでは、配備シナリオに基づいて配備設計を作成し、その設計の実装、テスト、および本稼働環境への展開を行います。

通常、配備プロセスには、1 つのソフトウェアソリューションをサポートするために必要な、すべての層およびすべてのインフラストラクチャーサービスレベルのソフトウェアコンポーネントが関連します。通常、カスタム開発されたアプリケーションコンポーネント (J2EE コンポーネント、Web サービス、またはその他のサーバー) と、ソリューションをサポートするために必要な Java ES コンポーネントの両方を配備する必要があります。

配備の作業は、図 4–1 に示すように 2 つのフェーズに分けられます。

- 66 ページの「配備設計」: 配備設計は、ソリューションの論理アーキテクチャーと、ソリューションが満たす必要があるパフォーマンス、可用性、セキュリティー、スケーラビリティー、保守容易性、およびその他のサービス品質の要件の両方に依存しています。配備アーキテクチャーにおけるサービス品質の次元は、配備設計フェーズで重要な役割を果たします。
- 68 ページの「配備の実装」: 配備設計の実装は、ハードウェアのセットアップ、ソフトウェアのインストールおよび設定、開発と統合、テスト、および本稼動への展開のその他の作業を含む反復プロセスです。

  この後の節で、これらの配備プロセスの 2 つのフェーズについてさらに詳しく説明します。

配備設計

配備設計フェーズでは、上位レベルの配備アーキテクチャーを作成したあと、下位レベルの実装仕様を作成します。

配備アーキテクチャー

配備アーキテクチャーは、配備シナリオに指定されたサービス品質の要件を満たす方法で、論理アーキテクチャー、つまりアプリケーションの論理的な構築ブロックを物理的なコンピューティング環境にマッピングすることによって作成されます。次の図に示すように、配備シナリオに基づいて配備アーキテクチャーが作成されます。

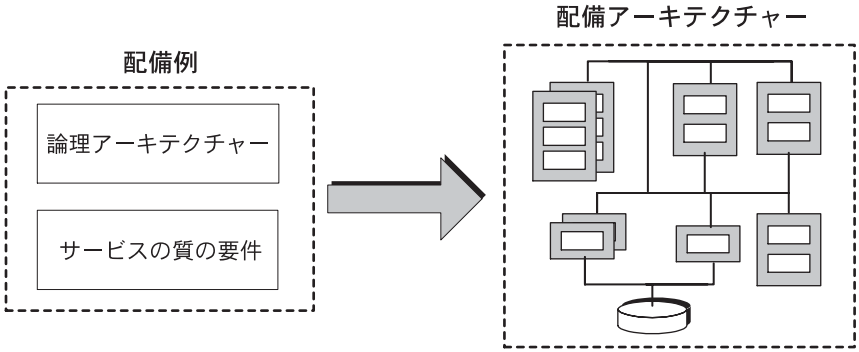

図 4–3　配備シナリオに基づいた配備アーキテクチャーの作成

このアーキテクチャー設計の特徴の 1 つとして、パフォーマンス、可用性、セキュリティー、およびその他のサービス品質の要件を満たすように、物理環境がサイジング (コンピュータの台数を決定し、コンピュータのプロセッサの処理能力および RAM の要件を見積もる) されます。サイジングが完了した後、物理環境内にあるさまざまなコンピュータに Java ES コンポーネントとアプリケーションコンポーネントを割り当てます。配備アーキテクチャーの作成時には、さまざまなコンピュータの能力、システムのインフラストラクチャーサービスの特徴、および総所有コストや総利用コストに対する制限を考慮に入れる必要があります。

配備シナリオに含まれる Java ES コンポーネントの数やサービス品質の要件で要求される事項が多ければ多いほど、設計上、より高機能なコンピュータやより広いネットワーク帯域幅が要求されます。ハードウェアが高価でその使用が制限される場合は、固定コスト (ハードウェア) と変動コスト (人的リソースの要件) のバランスやサービス品質の各要件のバランスを再検討したり、設計を見直して効率性を高める必要があります。

配備アーキテクチャーの設計では、試行錯誤の繰り返しが必要な場合もあります。ただし、Java Enterprise System は配備設計の出発点として、一連のリファレンス配備アーキテクチャーを開発します。

参照アーキテクチャーは、特定の配備シナリオ、つまり特定のサービス品質の要件が指定された論理アーキテクチャーに基づいています。参照アーキテクチャーでは、ソフトウェアソリューションは、指定されたサービス品質の要件を満たす方法で、特定の物理環境に配備されます。指定された負荷でのパフォーマンステストは、配備シナリオの開発に使用されたものと同じユースケースのセットに基づいています。参照アーキテクチャーのドキュメントは、非開示の条件付きで Java ES の顧客に提供されます。

リファレンス配備アーキテクチャーまたは複数の参照アーキテクチャーの組み合わせに基づいて、ユーザー独自の配備シナリオの要件を満たす配備アーキテクチャーに関する最初の概要を設計できます。ユーザー独自の配備シナリオと参照アーキテクチャーの元になっている配備シナリオの違いを考慮したうえで、参照アーキテク

チャーを調整したり、参照アーキテクチャーを参照点として使用したりできます。このようにして、ユーザー独自のサイジングの影響、パフォーマンス、セキュリティー、可用性、容量、および保守容易性のニーズを評価できます。

実装仕様

実装仕様によって、配備アーキテクチャーの実装に必要な詳細情報が提供されます。この仕様には、通常、次の情報が含まれます。

- コンピュータ、ストレージデバイス、ロードバランサ、ネットワークケーブルなどの実際のハードウェア
- オペレーティングシステム
- サブネットおよびセキュリティーゾーンを含む、ネットワーク設計
- 可用性の設計の詳細
- セキュリティー設計の詳細
- エンドユーザーのプロビジョニングに必要なディレクトリ設計情報

実装計画

実装計画は、配備実装フェーズのさまざまな作業をどのように実行するかを示します。この計画には、通常、次の作業が含まれます。

- ハードウェアの設定
- ソフトウェアのインストール、アップグレード、および移行
- システムの設定とカスタマイズ
- 開発および統合
- テスト
- 製品の発表

配備の実装

配備設計の実装は、前の節と図 4–1 で示した作業から構成されます。実際の配備プロセスでは作業の反復が発生するので、これらの各作業の順序は固定的なものではありません。以下の各項では、主要な各配備実装作業について、これらが通常実行される順序に従って個別に説明します。

ハードウェアのセットアップ

実装仕様には、物理環境に関するすべての詳細情報が含まれています。たとえば、コンピュータ、ネットワーク設計、ネットワークハードウェア (ケーブル、スイッチ、ルータ、ロードバランサなど)、ストレージデバイスなどです。これらのハードウェアはすべて、Java ES ソリューションをサポートするプラットフォームとして設定する必要があります。

ソフトウェアのインストール、アップグレード、および移行

配備アーキテクチャー、および実装仕様が提供する詳細情報によって、物理環境の各コンピュータに配備するアプリケーションコンポーネントおよびJava ESコンポーネントが明らかになります。Java ESの統合インストーラを使って配備アーキテクチャー内の各コンピュータに適切なJava ESコンポーネントをインストールします(55ページの「Java ES統合インストーラ」を参照)。

インストール計画は、インストーラセッションの順序および範囲を示します。ただし、インストールの実行方法は、初めてJava ESのインストールを実行するのか、以前インストールしたJava ESコンポーネントをアップグレードするのか、またはサードパーティのコンポーネントをJava ESと交換するのかどうかによって異なる場合があります。これらのJava ES導入シナリオの最後の2つは、多くの場合、互換性のためにデータやアプリケーションコードの移行を必要とします。

システムの設定とカスタマイズ

さまざまなシステムコンポーネントを統合されたシステムとして連携させるには、いくつかのシステム設定作業を完了させる必要があります。これらの作業の中で最初に完了する必要がある作業は、個々のシステムコンポーネントを起動させるために必要な初期設定です。次に、各Java ESコンポーネントが対話するコンポーネントと通信できるように設定する必要があります。

各コンポーネントの可用性ソリューションに応じて、高可用性も設定する必要があります。ユーザーが各種サービスにアクセスできるようにユーザーのプロビジョニングを行う必要があります。また、認証と承認のポリシーおよび制御を設定する必要もあります(58ページの「統合されたアイデンティティーサービスとセキュリティーサービス」を参照)。

ほとんどの場合、設定作業には、要件どおりの機能セットを実現するために、ある程度のJava ESコンポーネントのカスタマイズが必要です。たとえば、通常は、ポータルチャネルを提供するためにPortal Serverを、承認タスクを実行するためにAccess Managerをカスタマイズします。

開発と統合

一般に、配備シナリオに指定された論理アーキテクチャーによって、ソリューションの実装に必要なカスタム開発(development)作業の範囲が決まります。

配備によっては、Application ServerまたはWeb Server環境で実行されるJ2EEコンポーネントを使用して新しいビジネスサービスおよびプレゼンテーションサービスを最初から開発する必要があるので、開発は非常に大規模な作業になります。このような場合には、ソリューションのプロトタイプを作成して、開発成果の全体を実装する前に概念実証テストを実行します。

大規模な開発を必要とするソリューション用に、Sun Java™ Studioソフトウェアでは分散型コンポーネントまたはビジネスサービスをプログラミングするツールを提供

しています。Sun Java Studio 開発ツールは、Java ES インフラストラクチャーによりサポートされるアプリケーションのプログラミングとテストを簡素化します。

状況によっては、Java ES コンポーネントを旧バージョンのアプリケーションやサードパーティーのサービスと統合できます。このような統合には、データ層の既存のディレクトリまたはデータサービス、あるいはビジネスサービス層の既存のコンポーネントが含まれることがあります。Java ES コンポーネントをこれらのシステムと統合するには、データまたはアプリケーションコードの移行が必要になる場合があります。

J2EE プラットフォームでは、J2EE リソースアダプタを開発することによって既存のアプリケーションを Application Server 環境にプラグインするコネクタフレームワークを提供し、Message Queue では、さまざまなアプリケーションを統合するための強力な非同期メッセージング機能を提供します。

プロトタイプとパイロットのテスト

必要なカスタマイズ作業または開発作業の量によっては、ある時点で配備アーキテクチャーを検証し、ユースケースと比較して、ソリューションがサービス品質要件を満たしているかどうかをテストする必要があります。

カスタム開発サービスが比較的少ない (ほとんど追加設定の必要がない配備) 場合は、ソリューションで単に Java ES コンポーネントのカスタマイズとシステムのパイロットテストのみが必要になることがあります。

一方、重要な新規のアプリケーションロジックを開発してカスタムサービスを作成した場合、このテストはプロトタイプのテスト、統合テストなどを実施する大規模なものになる可能性があります。

このテストによって配備アーキテクチャーの欠点が判明した場合は、アーキテクチャーを修正してテストを再び実行します。最終的に、この反復プロセスによって、本稼動環境への配備が可能な配備アーキテクチャーおよび実装が完成します。

本稼動への展開

本稼動への展開では、本稼動環境に配備実装を組み入れます。このフェーズでは、本稼動環境での分散型アプリケーションおよびインフラストラクチャーサービスのインストール、設定、起動、本稼動システムのエンドユーザーのプロビジョニング、シングルサインオンおよびアクセスポリシーの設定などを行います。通常、まず限定的な配備を行い、その後に組織全体の実装に移行します。このプロセスでは試験稼動を実行し、適用する負荷を徐々に増やして、サービス品質の要件を満たしていることを確認します。

配置後

ライフサイクルの配備後 (postdeployment)の段階では、本稼働環境に配備されたソリューションを実行します。ライフサイクルのオペレーションフェーズでは、次の作業を行います。

- 監視: システムのパフォーマンスおよびシステムの機能を定期的に監視します。
- 保守: 新規エンドユーザーのシステムへの追加、パスワードの変更、新規管理ユーザーの追加、アクセス権限の変更、定期的なバックアップの実行などの日常的な管理機能を実行します。
- パフォーマンスの調整: システムの運用上のボトルネックを見つけるために定期的に情報を監視し、設定のプロパティーの変更、容量の追加などによりボトルネックの除去を試みます。
- システムの強化とアップグレード: システムに新しい Java ES コンポーネントを追加することで新しい機能を追加したり Java ES 以外のコンポーネントを置き換えたりする作業が含まれます。どちらの場合も、これらの変更によりソリューションのライフサイクルの最初のフェーズから、システムの再設計が必要になる場合があります。アップグレード作業は限定的で、通常は Java ES コンポーネントのアップグレードを実行します。

この章の重要な用語

この節では、この章で使用されている重要な技術用語について説明します。ここでは、Java ES の文脈でどのように使用されているかの説明に重点を置いています。

配備	Java ES ソリューションのライフサイクルの段階の 1 つ。この段階では、配備シナリオに基づいて、配備設計の作成、実装、プロトタイプ作成、本稼働環境への展開が行われます。このプロセスの最終結果は配備または配備ソリューションとも呼ばれます。
配備シナリオ **(deployment scenario)**	特定の Java ES ソリューションに対する論理アーキテクチャー (logical architecture)と、そのソリューションがビジネスニーズを満たすために必要なサービス品質の要件。サービス品質要件には、パフォーマンス、可用性、セキュリティー、保守容易性、スケーラビリティーまたは潜在容量などに関する要件があります。配備シナリオは、配備設計の開始点です。
開発 **(development)**	Java ES ソリューションの配備プロセスに含まれる作業の 1 つ。この作業によって、配備アーキテクチャー (deployment architecture)のカスタムコンポーネントがプログラミングおよびテストされます。
配備前 **(predeployment)**	Java ES ソリューションのライフサイクルプロセスの段階の 1 つ。この段階では、ビジネスニーズに基づいて配備シナリオ (deployment scenario)が作成されます。このシナリオには、論理アーキテクチャー (logical architecture)およびソリューションが満たす必要のある一連のサービス品質の要件が含まれます。

配備後 **(postdeployment)**	Java ES ソリューションのライフサイクルプロセスの段階の 1 つ。この段階では、分散型アプリケーションの起動、監視、調整によるパフォーマンスの最適化、および動的アップグレードによる新機能の追加が行われます。
リファレンス配備アーキテクチャー **(reference deployment architecture)**	パフォーマンス確認用に設計、実装、およびテストされた配備アーキテクチャー (deployment architecture)。リファレンス配備アーキテクチャーは、カスタムソリューションの配備アーキテクチャーを設計する場合の開始点として使用されます。
ユースケース **(use case)**	分散エンタープライズアプリケーション (distributed enterprise application)によって実行される、1 つまたは一連のエンドユーザータスク。アプリケーションの設計、テスト、およびパフォーマンスの測定を行うための基礎情報として使用されます。

◆ ◆ ◆ 付 録 A

Java ES コンポーネント

Java ES は、製品コンポーネントおよび共有コンポーネントのコレクションで構成され、これらは互いに連携して、ネットワーク経由の分散型アプリケーションをサポートします。インストール中、Java ES インストーラは選択可能なコンポーネントを提示し、その多くは選択可能なサブコンポーネントを含んでいます。これらのコンポーネントおよびサブコンポーネントがこの付録に一覧表示されています。

この付録には、Java ES コンポーネントの概要として簡単な説明が示されています。特定のコンポーネントに関する詳細は、`http://docs.sun.com/app/docs/prod/entsys.5` で利用可能なマニュアルセットを参照してください。広範囲にわたる Java ES についての情報およびリソースは、`http://www.sun.com/bigadmin/hubs/javaes/` からも利用できます。

この付録に一覧表示する Java ES コンポーネントは、カテゴリごとにグループ化され、次の節で説明されています。

システムサービスコンポーネント

Java ES システムサービスコンポーネントは、分散型エンタープライズアプリケーションをサポートするために必要なインフラストラクチャーサービスを提供します。これらのサービスには、17 ページの「Java ES が必要な理由」で説明したように、ポータルサービス、アイデンティティーサービスとセキュリティーサービス、Web サービスとアプリケーションサービス、および可用性サービスが含まれます。次の各節で Java ES システムサービスコンポーネントについて説明します。

Access Manager 7.1

Sun Java System Access Manager (Access Manager) は、認証、承認サービス、ポリシーエージェント、およびアイデンティティー連携を統合し、ネットワークリソースを保護するための包括的なソリューションを提供します。Access Manager は、Web サービスアプリケーションおよび Web コンテンツに対する権限のないアクセスを防ぎ、Web ベースのサービスおよび Web ベースでないアプリケーションを使用する顧客、従業員、およびパートナのデジタル ID の管理プロセスを、組織が管理するためのインフラストラクチャーを提供します。これらのリソースは内部および外部の広範なコンピューティングネットワークで利用される可能性があるため、ID ごとに属性、ポリシー、資格付与が定義および適用されて、これらのテクノロジへのアクセスが管理されます。

Access Manager には、次のサブコンポーネントが含まれます。

- **Access Manager** コアサービス**:** これを使用すると、ユーザーのアイデンティティーを作成および管理したり、ユーザーのアイデンティティーに基づいて Java ES リソースへのアクセスを許可するためのポリシーを定義および評価したりできます。
- **Access Manager** 管理コンソール**:** アイデンティティーサービスとポリシー管理を統合し、ユーザーが Directory Server でユーザーアカウント、サービス属性、アクセス規則を作成、管理するための単一グラフィカルインタフェースとなります。
- 連携管理の共有ドメインサービス**:** ユーザーは、複数の関連するサービスプロバイダが提供するアプリケーションに単一の ID でアクセスできます。
- **Access Manager SDK:** Access Manager へのリモートインタフェースを提供します。このサブコンポーネントは、Access Manager にリモートアクセスする Java ES コンポーネントをホストするすべてのコンピュータ上にインストールする必要があります。
- **Access Manager** 分散認証ユーザーインタフェース **:** セキュリティー保護されていない領域に配備されたポリシーエージェントまたはアプリケーションが、セキュリティー保護された配備の領域にインストールされた Access Manager 認証サービスと通信することを可能にするユーザーインタフェースを提供します。
- **Access Manager** クライアント **SDK:** Access Manager サーバーにアクセス可能なスタンドアロンのアプリケーションを実装して、認証、SSO、承認、監査、ロギング、SAML などのサービスを使用できるようにします。
- **Access Manager** セッションフェイルオーバークライアント**:** Access Manager セッションフェイルオーバーを設定するために必要です。

Application Server Enterprise Edition 8.2

Sun Java System Application Server (Application Server) は、サーバーサイド Java アプリケーションと Web サービスの開発および配信に使用する J2EE 互換プラットフォームを提供します。主要な機能には、スケーラブルなトランザクション管理、コンテナ管理された持続ランタイム、Web サービスパフォーマンス、クラスタ、高可用性セッション状態、セキュリティー、および統合機能が含まれます。

Application Server には、次のサブコンポーネントが含まれます。

- ドメイン 管理サーバー**:** Application Server の管理や設定、また J2EE コンポーネントおよびアプリケーションの配備などのサーバー側の管理機能を提供します。
- **Application Server** ノードエージェント**:** サーバーインスタンスをホストするすべてのマシンで実行する軽量のプロセスで、サーバーインスタンスの停止、起動、および再起動などの多くの管理作業を行います。
- コマンド行管理ツール**:** Application Server インストールおよびホストされたアプリケーションの管理および設定を可能にする、コマンド行管理クライアントを提供します。このツールは、アプリケーションの配備も支援します。
- ロードバランスプラグイン。複数の Application Server インスタンス (スタンドアロンまたはクラスタのどちらか) の間で均等にワークロードを分配するために使用し、システムの全体のスループットを向上させます。また、1 つのサーバーインスタンスから別のサーバーインスタンスへのフェールオーバーの要求を有効にするためにも使用します。
- アプリケーションのサンプル。Application Server の完全インストールでインストールされます。

Directory Server Enterprise Edition 6.2

Sun Java System Directory Server (Directory Server) は LDAP ベースのディレクトリサーバーで、イントラネット、ネットワーク、およびエクストラネットの情報に対する、一元化されたディレクトリサービスを提供します。Directory Server は既存のシステムに統合され、従業員、顧客、仕入先、およびパートナ企業の情報を統合化に対応した集中的リポジトリとして機能します。Directory Server を拡張することで、ユーザーのプロファイルや設定情報、およびエクストラネットのユーザー認証を管理できます。

Directory Server には、次のサブコンポーネントが含まれます。

- **Directory Server 6.2** コアサーバー**:** アイデンティティーデータを保存し管理するための、スケーラブルで安全な、柔軟性の高い手段を提供します。
- **Directory Service Control Center:** ブラウザベースの管理インタフェースを提供し、ディレクトリサービスおよびディレクトリプロキシサービスを設定します。

- **Directory Server** コマンド行ユーティリティー**:** コマンド行から管理作業を行えるようになります。
- **Directory Proxy Server 6.2** コアサーバー**:** 仮想ディレクトリ機能を提供し、ディレクトリサービスの可用性およびスケーラビリティーを向上させることによって、セキュリティーを強化します。

Java DB 10.2

Java DB は、Java アプリケーションの開発用に軽量なデータベースを提供します。Java DB は、100% Java テクノロジによるデータベースであるオープンソース Apache Derby の、Sun がサポートする配布です。Java ES 5 は、Java DB を製品コンポーネントとして含む最初のリリースでした。Java DB は最初 Derby Database という名前の共有コンポーネントとしてリリースされ、Java ES 2005Q4 に含まれていました。

Java DB には、次のサブコンポーネントが含まれます。

- Java DB クライアント
- Java DB サーバー

Message Queue 3.7 UR2

Sun Java System Message Queue (Message Queue) は、アプリケーション間通信および信頼性の高いメッセージ配信の問題に対する、標準ベースのソリューションです。Message Queue は、JMS (Java Message Service) オープン標準を実装した企業向けのメッセージングシステムです。

Message Queue の機能は、JMS プロバイダであることに加え、JMS 仕様の最小要件を満たしています。Message Queue ソフトウェアを使用することで、異なるプラットフォームおよびオペレーティングシステム上で稼動するプロセスが共通の Message Queue サービスに接続して、情報を送受信できます。アプリケーション開発者は、ネットワーク間の通信方法に関する低レベルの詳細ではなく、アプリケーションのビジネスロジックに集中して作業を行うことができます。

Java ES インストーラでは、Message Queue はインストール可能な単一のコンポーネントとして提供されます。

Portal Server 7.1 Update 2

Sun Java System Portal Server (Portal Server) は、アイデンティティーに対応したポータルサーバーソリューションです。Portal Server はパーソナル化、集約、セキュリティー、統合、検索などの主なポータルサービスを組み合わせます。

Java ES インストーラでは、Portal Server はインストール可能な単一のコンポーネントとして提供されます。

Service Registry 3.1

Sun Java System Service Registry (Service Registry) は、Web サービス (UDDI) レジストリとしても、エンタープライズビジネス XML (ebXML) レジストリとしても機能するリポジトリであり、Web ベースのサービス指向アーキテクチャー (SOA) アプリケーションをサポートします。UDDI レジストリは、Web サービスの登録と検索に使用され、ebXML レジストリは、ビジネスプロセス統合のサポートに必要な情報アーティファクトの格納および管理に使用されます。これらのアーティファクトには、XML スキーマ、ビジネスプロセスルール、Web サービスアクセス制御、バージョン管理、分類スキーマなどのメタデータが含まれます。

Service Registry には、次のサブコンポーネントが含まれます。

- Service Registry クライアントサポート
- Service Registry 配備サポート

Web Server 7.0

Sun Java System Web Server (Web Server) は、マルチプロセスとマルチスレッドに対応するセキュリティー保護された Web サーバーであり、業界標準に基づいて構築されています。Web Server は、高いパフォーマンス、信頼性、スケーラビリティー、および管理能力を、中規模から大規模の企業に対して提供します。

Web Server には、次のサブコンポーネントが含まれます。

- Web Server CLI
- Web Server コア
- Web Server サンプル

サービス品質コンポーネント

Java ES サービス品質コンポーネントにより、システムサービスコンポーネントまたは分散型アプリケーションコンポーネントが提供するサービス品質が向上します。システムがほぼ連続的に稼動することを可能にする可用性コンポーネント、システムサービスへのセキュリティー保護されたエンドユーザーアクセスを可能にするアクセスコンポーネント、また Java ES ソリューションの保守容易性の向上に使用するシステム管理コンポーネントがあります。

Java ES サービスコンポーネントをサポートするコンポーネントは、次のカテゴリにグループ化され、この節で説明しています。

- 78 ページの「可用性コンポーネント」
- 80 ページの「アクセスコンポーネント」
- 81 ページの「監視コンポーネント」

可用性コンポーネント

可用性コンポーネントは、システムサービスコンポーネントおよびアプリケーションコンポーネントがほぼ連続的に稼動することを可能にします。ここでは、次の Java ES 可用性コンポーネントについて説明します。

High Availability Session Store 4.4

Sun Java System High Availability Session Store (HADB) が提供するデータストアを使用すれば、障害発生時でもアプリケーションのデータが利用可能になります。この機能は、クライアントセッションに関連付けられた状態情報の復元に特に重要です。この機能がないと、セッション中に障害が発生した場合、セッションの再確立時にすべてのオペレーションを繰り返す必要があります。

次の Java ES コンポーネントは、セッション状態情報を格納するサービスを提供します。Application Server、Access Manager、および Message Queue。ただし、これらのコンポーネントの中で、障害時にセッション状態を保持するために HADB サービスを使用できるのは Application Server のみです。

Java ES インストーラでは、HADB はインストール可能な単一のコンポーネントとして提供されます。ただし、HADB サービスを提供するにはサーバーとクライアントの両方のサブコンポーネントが必要です。

Sun Cluster 3.1 8/06 および Sun Cluster エージェント 3.1

注 – Sun Cluster コンポーネントは、Solaris プラットフォームでのみサポートされます。

Sun Cluster ソフトウェアは、高可用性サービスおよび高スケーラビリティーサービスを、Java ES とJava ES インフラストラクチャーに基づくアプリケーションに対して提供します。

クラスタとは緩やかに結合された一連のコンピュータ (クラスタノード) のことであり、サービス、システムリソース、およびデータの単一のクライアントビューを一括して提供します。クラスタの内部では、冗長コンピュータ、インターコネクト、データ記憶域、およびネットワークインタフェースを使用して、クラスタベースのサービスおよびデータに高可用性を提供します。 Sun Cluster ソフトウェアは、メンバーノードおよびその他のクラスタリソースの健全性を継続的に監視し、障害が発生した場合でも、内部の冗長性を利用してそれらのリソースへのほぼ連続的なアクセスを提供します。

Java ES インストーラでは、Sun Cluster コアサブコンポーネントおよび Sun Cluster エージェントが、個別にインストール可能なコンポーネントとして提供されます。次の Sun Cluster エージェントが Java Enterprise System に含まれます。

注 – 次のリストの *HA* は、高可用性を表します。

- HA Application Server
- HA Message Queue
- HA Directory Server
- HA Messaging Server
- HA Application Server EE (HADB)
- HA/Scalable Web Server
- HA Instant Messaging
- HA Calendar Server
- HA Apache Tomcat
- HA Apache
- HA DHCP
- HA DNS
- HA MySQL
- HA Sun N1 Service Provisioning
- HA NFS
- HA Oracle
- HA Samba
- HA Sun N1 Grid Engine
- HA Solaris Containers

注 – エージェントのリストは、SPARC と x86 では異なります。Sun Cluster Agents についての詳細は、`http://docs.sun.com/app/docs/prod/entsys.5` の Sun Cluster のマニュアルを参照してください。

Sun Cluster Geographic Edition 3.1 2006Q4

Sun Cluster Geographic Edition は、Sun Cluster ソフトウェアを階層的に拡張したものです。この拡張は、地理的に離れた複数のクラスタを使用し、これらのクラスタ間でデータを複製する冗長なインフラストラクチャーを使用することによって、予期しない中断からアプリケーションを保護します。Java ES 5 は、Sun Cluster Geographic Edition を Java ES 製品コンポーネントとして含む最初のリリースです。

Sun Cluster Geographic Edition には、次のサブコンポーネントが含まれます。

- Sun Cluster Geographic Edition コア
- Sun StorEdge Availability Suite
- Hitachi Truecopy Data Replication Support (SPARC のみ)
- EMC SRDF Data Replication

注 – Sun Cluster Geographic Edition は、Solaris x86 ではサポートされていません。

アクセスコンポーネント

アクセスコンポーネントは、システムサービスへのフロントエンドアクセスを可能にし、多くの場合、エンタープライズファイアウォールの外にあるインターネットからのアクセスを可能にします。ここでは、次の Java ES アクセスコンポーネントについて説明します。

- 80 ページの「Portal Server Secure Remote Access 7.1」
- 81 ページの「Web Proxy Server 4.0.5」

Portal Server Secure Remote Access 7.1

Sun Java System Portal Server Secure Remote Access (Portal Server Secure Remote Access) は、Portal Server のコンテンツとサービスに対するブラウザベースのセキュリティー保護されたリモートアクセスを任意のリモートブラウザに対して提供することで、Portal Server を拡張します。これにより、クライアントソフトウェアを用意する必要がなくなります。Portal Server との統合により、ユーザーはセキュリティー保護されたアクセスで、アクセス権があるコンテンツやサービスにアクセスすることができます。

Portal Server Secure Remote Access には、次のサブコンポーネントが含まれます。

- **Portal Server Secure Remote Access** コア**:** コア機能を提供します。
- ゲートウェイ**:** インターネットから送信されるリモートユーザーセッションと企業イントラネットの間のインタフェースおよびセキュリティーバリアとして機能します。ゲートウェイは単一のインタフェースにより、内部 Web サーバーとアプリケーションサーバーのコンテンツをリモートユーザーに安全に提供し、Portal Server とさまざまなゲートウェイインスタンスとの通信を制御します。
- **Netlet Proxy:** インターネットなどのセキュリティー保護されていないネットワークを通じて、一般的な TCP/IP サービスを安全に実行できます。Netlet を使用することで、telnet、SMTP、HTTP、固定ポートアプリケーションなどのアプリケーションを実行できます。Netlet により、ファイルシステムおよびディレクトリのリモートアクセスおよび操作が可能になり、クライアントブラウザ上の Netlet アプレット、ゲートウェイ、およびアプリケーションサーバー間のセキュリティー保護された通信を確保します。
- **Rewriter Proxy:** ゲートウェイとイントラネットコンピュータの間の HTTP トラフィックをセキュリティー保護することができます。Rewriter は、Web リンクを転送し、イントラネットの Web ページの扱いに関するルールセットを作成することで、イントラネット外から企業イントラネットの Web ページへのセキュリティー保護されたアクセスを提供します。

Web Proxy Server 4.0.5

Sun Java System Web Proxy Server (Web Proxy Server) は、Web コンテンツのキャッシュ機能、フィルタリング機能、および配信機能を提供します。Web Proxy Server は通常、企業のファイアウォールの内側ではリモートのコンテンツサーバーに対する要求数を低減するために使用され、ファイアウォールの外側では受信されたインターネット要求に対するセキュリティー保護されたゲートウェイを提供するために使用されます。

Java ES インストーラでは、Web Proxy Server はインストール可能な単一のコンポーネントとして提供されます。

監視コンポーネント

Sun Java System Monitoring Console 1.0 (Monitoring Console) には、Java ES 配備内のすべてのノードエージェントに接続するマスターエージェントが含まれます。Monitoring Console は、Sun Java System Monitoring Framework 2.0 (Monitoring Framework) によってサポートされ、監視対象コンポーネントの属性を公開するために、すべての監視対象コンポーネントに必要なインストゥルメンテーションとノードエージェントを提供する、共有コンポーネントです。それぞれの製品コンポーネントは監視可能な属性を表すオブジェクトを公開し、ノードエージェントはホスト上の複数のコンポーネントの表示を集約します。監視についての詳細は、『Sun Java Enterprise System 5 Update 1 Monitoring Guide』を参照してください。

共有コンポーネント

共有コンポーネントは、Java ES のシステムサービスコンポーネントとサービス品質コンポーネントが依存するローカルサービスとテクノロジサポートを提供します。特定のホストコンピュータ上で稼動するすべての Java ES コンポーネントが共有できるローカルライブラリです。Java ES インストーラは、ホストコンピュータにインストールされたほかの Java ES コンポーネントをサポートするために必要なすべての共有コンポーネントを自動的にインストールします。

Java ES には、次の共有コンポーネントが含まれます。

- ACL (Apache Common Logging) 1.0.4
- ANT (Jakarta ANT Java/XML ベースのビルドツール) 1.6.5
- BDB (Berkeley Database) 4.2.52
- Common Agent Container 2.1 (Sun Cluster のみ)
- Common Agent Container 2.1
- FastInfoSet 1.0.2
- ICU 3 ((International Components for Unicode) 3.2
- J2SE (Java 2 Platform, Standard Edition) 5.0 Update 6
- JAF (JavaBeans™ Activation Framework) 1.0.3

- JATO (Java Studio Web Application Framework) 2.1.5
- JavaHelp™ 2.0
- JavaMail™ API 1.3.2
- JAXB (Java Architecture for XML Binding) 1.0.6
- JAXP (Java API for XML Processing) 1.3.1_01
- JAXR (Java API for XML Registries) 1.0.8
- JAXRPC (Java API for XML-based Remote Procedure Call) 1.1.3_01
- JAXWS (Java API for Web Services) 2.0
- JDMK (Java Dynamic Management Kit) 5.1_03
- JSS (Java Security Services) 4.2.4
- JSS3 (Java 用のネットワークセキュリティーサービス) 3.1.11
- JSTL (JavaServer Pages™ Standard Tag Library) 1.0.6
- KTSE (KT Search Engine) 1.3.4
- LDAP C SDK 6.0
- LDAP Java SDK 4.19
- MA Core (Mobile Access Core) 6.3.1
- NSPR (Netscape Portable ランタイム) 4.6.3
- NSPRD (Netscape Portable Runtime Development) 4.6
- NSS (ネットワークセキュリティーサービス) 3.11
- NSSU (Network Security Service Utilities) 3.11
- SAAJ (SOAP with Attachments API for Java) 1.3
- SASL (Simple Authentication and Security Layer) 2.19
- Sun Explorer Data Collector (Solaris OS のみ) 4.3.1
- Sun Java System Monitoring Framework 2.0 (Monitoring Console 1.0 をサポート)
- Sun Java Web コンソール 3.0.2
- WSCL (Web Services Common Library) 2.0
- XWSS (XML Web Services Security) 2.0

索引

ひ

ふ

ほ

み