Sun Java System Access Manager 7 2005Q4 配備計画ガイド

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

Part No: 819–4121
2005 年 10 月

目次

表目次

図目次

はじめに

『Sun Java System Access Manager 7 2005Q4 配備計画ガイド』では、ソリューションライフサイクルに基づいて、Sun Java™ System Access Manager のための計画と配備のソリューションについて説明します。

Access Manager は Sun Java™ Enterprise System (Java ES) のコンポーネントの 1 つで、ネットワーク環境またはインターネット環境に分散した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サービスを提供するソフトウェアコンポーネントで構成されています。

対象読者

本書は、Access Manager 配備の計画、分析、および設計を担当する配備設計者およびビジネスプランナ向けに書かれています。また、本書は Access Manager 配備の特定の分野で設計や実装に携わるシステムインテグレータにとっても役立ちます。

お読みになる前に

次のコンポーネントおよび概念に精通することをお勧めします。

- 『Sun Java System Access Manager 7 2005Q4 Technical Overview』に記載されている Access Manager の技術的な概念
- 配備先のプラットフォーム: Solaris™ オペレーティングシステムまたは Linux オペレーティングシステム
- Access Manager を実行する Web コンテナ: Sun Java System Application Server、Sun Java System Web Server、BEA WebLogic、または IBM WebSphere Application Server

- 技術的な概念: LDAP (Lightweight Directory Access Protocol)、 Java™ テクノロジ、JSP (JavaServer Pages™) テクノロジ、HTTP (HyperText Transfer Protocol)、HTML (HyperText Markup Language)、および XML (eXtensible Markup Language)

内容の紹介

本書は次のように構成されています。

第 1 章では、Sun Java System Access Manager の概要を述べます。

第 2 章では、ソリューションのライフサイクルにおけるビジネス分析段階、つまり業務上の問題を分析してビジネス目的を定義し、その目的を達成するために考慮する必要のあるビジネス要件や制約を特定する作業について説明します。

第 3 章では、ソリューションのライフサイクルにおける技術要件段階、つまり使用状況を分析し、ユースケースを確認し、提案されている配備ソリューションの Quality of Service (QoS) 要件を決定する作業について説明します。

第 4 章では、ソリューションのライフサイクルにおける論理設計段階、つまりソリューションの論理コンポーネントの相互関係を示す論理アーキテクチャーを設計する作業について説明します。

第 5 章では、ソリューションのライフサイクルにおける配備設計段階、つまりハイレベルの配備アーキテクチャーとローレベルの実装仕様を設計し、ソリューションの実装に必要な一連の計画と仕様を準備する作業について説明します。

第 6 章では、ソリューションのライフサイクルにおける実装段階について説明します。たとえば、Access Manager を複数のサーバーに配備したり、Access Manager セッションフェイルオーバーをインストールしたり、設定したりする作業が含まれます。

付録 A では、Access Manager をインストールしたあとのディレクトリレイアウトについて説明します。

付録 B では、パスワード暗号化鍵、およびインストールしたあとのパスワード暗号化鍵の変更について説明します。

関連マニュアル

次の関連マニュアルを利用できます。

Access Manager コアマニュアル

Access Manager コアマニュアルには、次のマニュアルが含まれます。

- 『Sun Java System Access Manager 7 2005Q4 リリースノート』は、製品リリース後にオンラインで入手できます。このリリースノートでは、現行リリースの新機能、既知の問題と制限、インストール上の注意、およびソフトウェアやマニュアルで見つかった問題を報告する方法を含む、リリース時点における各種の情報について説明します。
- 『Sun Java System Access Manager 7 2005Q4 Technical Overview』では、Access Manager の各コンポーネントがどのように連携してアクセス制御機能を統合し、企業資産やWeb ベースアプリケーションを保護するかについて、その概要を説明します。 また、Access Manager の基本概念や基本用語についても解説します。
- 『Sun Java System Access Manager 7 2005Q4 配備計画ガイド』(本書) は、既存の情報技術インスラストラクチャーでのAccess Manager 配備の計画に関する情報を提供します。
- 『Sun Java System Access Manager 7 2005Q4 Performance Tuning Guide』では、Access Manager およびその関連コンポーネントが最適なパフォーマンスを得られるように調整する方法について説明します。
- 『Sun Java System Access Manager 7 2005Q4 管理ガイド』では、Access Manager コンソールの使用方法、およびコマンド行インタフェースを介してユーザーおよびサービスデータを管理する方法について説明します。
- 『Sun Java System Access Manager 7 2005Q4 Federation and SAML Administration Guide』では、Liberty Alliance Project 仕様に基づいた連携モジュールについて説明します。その中には、この仕様に基づいた統合サービスに関する情報、Liberty ベースの環境を有効にするための手順、およびフレームワークを拡張するためのアプリケーションプログラミングインタフェース (API) の要約が含まれます。
- 『Sun Java System Access Manager 7 2005Q4 Developer's Guide』は、Access Manager をカスタマイズする方法や Access Manager の機能を組織の現在の技術的なインフラストラクチャーに統合する方法についての情報を提供します。製品のプログラミング上の側面および API についても詳しく説明します。
- 『Sun Java System Access Manager 7 2005Q4 C API Reference』は、Access Manager 公開 C API を構成するデータ型、構造、および関数を要約して説明します。

- 『Sun Java System Access Manager 7 2005Q4 Java API Reference』は、Access Manager でのJava パッケージの実装について説明します。
- 『Sun Java System Access Manager Policy Agent 2.2 User's Guide』は、Access Manager で使用可能なポリシー機能とポリシーエージェントの概要を示します。

『Access Manager リリースノート』の更新状況およびコアマニュアルの修正情報へのリンクは、 Access Manager マニュアル Web サイト (`http://docs.sun.com/app/docs/coll/1367.1`) にあります。

Sun Java Enterprise System 製品マニュアル

関連製品に関する役立つ情報については、Sun Java Enterprise System マニュアル Web サイト (`http://docs.sun.com/prod/entsys.05q4`) の次のマニュアルコレクションを参照してください。

- Sun Java System Directory Server。

  `http://docs.sun.com/app/docs/coll/1372.1`
- Sun Java System Web Server。

  `http://docs.sun.com/app/docs/coll/1378.1`
- Sun Java System Application Server。

  `http://docs.sun.com/app/docs/coll/1369.1`
- Sun Java System Message Queue。

  `http://docs.sun.com/app/docs/coll/1374.1`
- Sun Java System Web Proxy Server。

  `http://docs.sun.com/app/docs/coll/1377.1`

サードパーティーの関連 Web サイト

このリリースノートに掲載されているサードパーティーの URL を参照すると、追加および関連情報を入手できます。

注 – Sun は、本書に記載されたサードパーティーの Web サイトの有効性について責任を負いません。Sun は、これらのサイトまたはリソースを通じて入手可能なコンテンツ、広告、製品、その他の内容についていかなる保証もせず、かつ責任や義務を負いません。Sun は、これらのサイトやリソースを通じて入手したコンテンツ、製品、またはサービスを使用または信頼することに起因または関連する、またはそう主張された、現実のまたは主張されたいかなる損害や損失についても責任や義務を負わないものとします。

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。

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–1** 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・ユーザーズガイド』を参照してください。
「 」	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。

ご意見、ご要望をお寄せください

Sun はマニュアルをより良いものにするために、ご意見やご提案をお待ちしております。

ご意見をお寄せいただく場合は、http://docs.sun.com にアクセスし、「コメントの送信」をクリックしてください。オンラインフォームの場合は、マニュアルのタイトルとパーツ番号を指定してください。パーツ番号は、マニュアルのタイトルページまたはマニュアルの上部にある7 桁または9 桁の番号です。

たとえば、本書のタイトルは、『Sun Java System Access Manager 7 2005Q4 配備計画ガイド』で、パーツ番号は 819-4121 です。

第 1 章

Access Manager の配備計画について

Sun Java™ System Access Manager (Access Manager) は Sun アイデンティティー管理インフラストラクチャーの一部を成すもので、エンタープライズ内および企業間 (B2B) のバリューチェーンにおいて、組織が Web アプリケーションやほかのリソースへのセキュリティー保護されたアクセスを管理できるようにします。この章では、Access Manager の配備を計画する際の基本的な方針について説明します。次の節で構成されています。

Access Manager について

Access Manager は Sun Java™ Enterprise System (Java ES) のコンポーネントの 1 つで、ネットワーク環境またはインターネット環境に分散した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のサービスを提供するソフトウェアコンポーネントで構成されています。Access Manager には、次の主な機能が備えられています。

- ロールとルールの両方に基づくアクセス制御を使用した集中化された認証および認証サービス
- 組織の Web ベースアプリケーションへのシングルサインオン (SSO) アクセス
- Liberty Alliance Project および SAML (Security Assertions Markup Language) と連携したアイデンティティーサポート
- 管理者およびユーザーのアクティビティーを含む重要な情報の Access Manager コンポーネントによるロギング。この記録をあとで分析、報告、監査に使用できます。ロギングは J2SE ロギング API (`java.util.logging`) をベースに実行されます。

Access Manager は、ディレクトリサービス、アクセス管理、プロビジョニング、連携などのアイデンティティー情報の利用、共有、および管理を行うために必要な機能を提供する Sun アイデンティティー管理製品群の一部でもあります。アイデンティティー管理製品群には次の製品があります。

- Sun Java System Access Manager
- Sun Java System Directory Server Enterprise Edition
- Sun Java System Federation Manager
- Sun Java System Identity Auditor
- Sun Java System Identity Manager
- Sun Java System Identity Manager Service Provider Edition

各コンポーネントの詳細については、次の Sun ソフトウェア Web サイトを参照してください。`http://jp.sun.com/products/software/`

次の図に、Access Manager、Identity Manager、および Directory Server のアイデンティティー管理コンポーネントを示します。

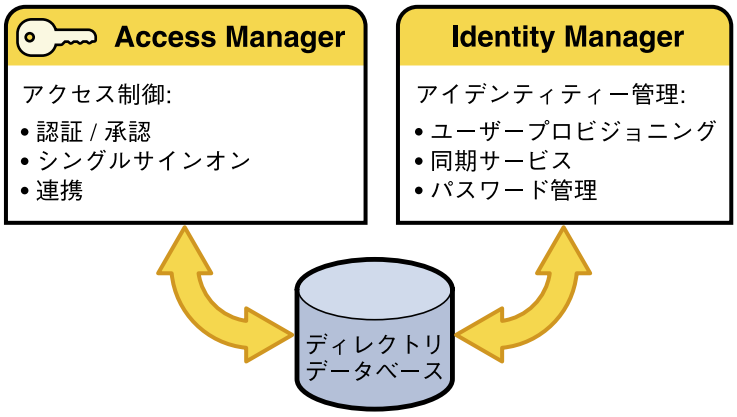


図 **1–1** Sun アイデンティティー管理コンポーネント

Sun Java System Identity Manager には、ユーザープロビジョニング、パスワード管理、同期サービス、包括的な監査とレポート作成、および管理の委譲といった機能があります。Identity Manager は Sun Java Enterprise System のコンポーネントではありません。Identity Manager をご使用の配備環境で使用する場合、または詳細情報の入手をご希望の場合は、Sun Microsystems の技術担当者、または次のサイトにある Sun 事業所にご連絡ください。`http://www.sun.com/sales-n-service/WWSales.html`

Access Manager に関する詳細な説明については、『Sun Java System Access Manager 7 2005Q4 Technical Overview』を参照してください。

Access Manager の配備計画

アイデンティティー管理ソリューションの実装を成功させる上で、配備計画の段階は非常に重要です。目標、要件、そして優先事項は、企業によって異なります。配備計画がうまくいくかどうかは、入念な準備、分析、および設計にかかっています。計画作業のどこかにミスや手違いがあると、いろいろな面でシステムに不具合が生じることになりかねません。システムの計画がずさんなために、重大な問題が起きる可能性もあります。たとえば、システムのパフォーマンスが上がらない、維持管理が面倒である、操作コストがかかりすぎる、リソースに無駄がある、需要の増大に見合った拡張ができないといった問題が生じることがあります。

このマニュアルで説明する Access Manager 配備計画は、ソリューションのライフサイクルに沿ったものです。ソリューションのライフサイクルには、Java Enterprise System をベースにした Access Manager エンタープライズソフトウェアソリューションを計画し、設計し、実装するという各プロセスが含まれます。

ソリューションのライフサイクル

次の図に示されているソリューションのライフサイクルは、配備プロジェクトの計画と進捗管理の手だてとして役立ちます。このライフサイクル構造には、配備計画を成功させるために必要な準備、分析、および設計が一連の段階として順番に組み込まれています。それぞれの段階は関連するいくつかのタスクからなり、1 つの段階で行われたタスクのアウトプットが次の段階のタスクのインプットとなります。それぞれの段階で行われるタスクを繰り返し行い、分析と設計を尽くした結果としてその段階のアウトプットを生成することが必要となります。

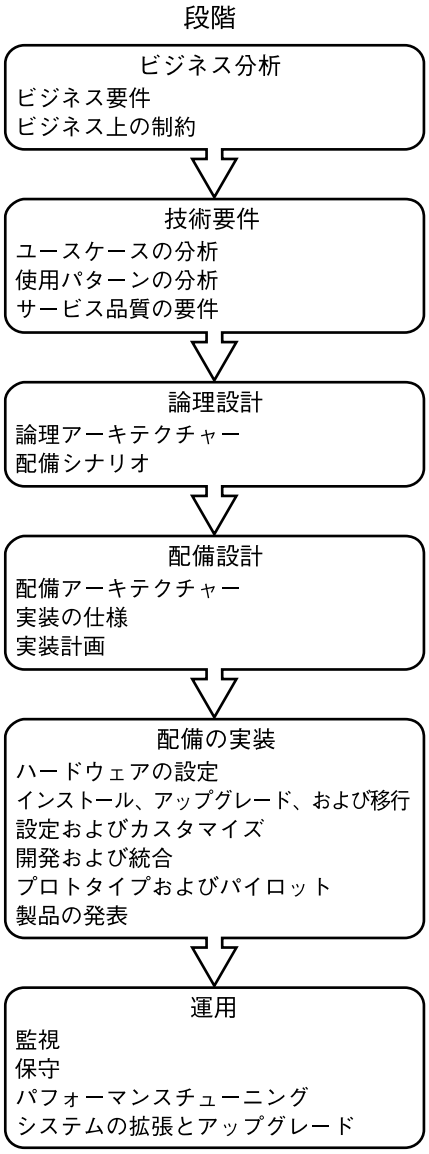


図 **1–2** ソリューションのライフサイクル

このマニュアルは、ソリューションのライフサイクルにおける各段階に基づいた構成になっています。この章の次の項では、ライフサイクルの各段階の要約を示します。これらの段階の詳細な説明については、『Sun Java Enterprise System 2005Q4 Deployment Planning Guide』を参照してください。

ビジネス分析段階

ビジネス分析を行う際には、配備プロジェクトの業務目標を定義し、その目標を達成するために満たす必要のあるビジネス要件を明示します。ビジネス要件を明示するにあたっては、業務目標の達成に影響を与える可能性のあるビジネス上の制約すべてを考慮します。適正なビジネス分析を行わないと、ソリューションが不完全になる危険性があります。

ビジネス分析段階の間に、ビジネス要件に関する資料を作成し、あとで技術要件段階のタスクのインプットとして使用できるようにしておきます。

第 2 章を参照してください。

技術要件段階

技術要件の段階では、ビジネス分析段階で定義したビジネス要件とビジネス上の制約の処理から始まり、それらの情報を次の配備アーキテクチャーの設計に使用できる技術仕様に書き換えます。技術要件では、パフォーマンス、可用性、セキュリティーなどの QoS (Quality of Service) 特性を指定します。

技術要件段階の間に、次の情報を含む資料を作成します。

- ユーザーのタスクと使用状況パターンの分析
- 計画しているシステムでのユーザー対話のユースケース
- ビジネス要件から導き出したQoS 要件。ユーザーのタスクと使用状況パターンの分析を考慮に入れることも可能

作成した使用状況分析、ユースケース、およびQoS 要件の資料は、ソリューションのラ イフサイクルにおける論理設計のインプットとなります。使用状況分析は、配備設計段階でも重要な役割を果たします。

第 3 章を参照してください。

論理設計段階

論理設計では、技術要件段階で作成したユースケースをインプットとして使用し、ソリューションを実装するために必要な Access Manager コンポーネントを特定します。Java ES コンポーネントをサポートするコンポーネントや、ビジネス要件を満たすために必要な独自開発のコンポーネントも指定します。次に、それらのコンポーネントを、コンポーネント間の相互関係を示す論理アーキテクチャーにマップします。論理アーキテクチャー上には、ソリューションの実装に必要なハードウェアの指定は行いません。

論理設計段階のアウトプットが論理アーキテクチャーです。論理アーキテクチャーと技術要件段階で作成した QoS 要件とによって配備シナリオが作成され、このシナリオが配備設計段階でのインプットとなります。

第 4 章を参照してください。

配備設計段階

配備設計では、論理アーキテクチャーに指定したコンポーネントを物理的な環境に対応させ、ハイレベルの配備アーキテクチャーを作成します。配備アーキテクチャーを構築する方法のローレベルの詳細を具体的に指定する、実装仕様も作成します。また、ソフトウェアソリューションの実装に関係するさまざまな面を詳述した一連の計画と仕様も作成します。

この配備設計段階でプロジェクトの承認が行われます。プロジェクトの承認にあたっては、配備のコストが評価されます。承認が下りたら、配備の実装に関わる契約を結び、プロジェクトを構築するためのリソースを確保します。通常、プロジェクトの承認は、実装仕様が細部まで明らかになったあとに行われます。しかし、配備アーキテクチャーが完成した時点で承認が下りる場合もあります。

配備設計段階のアウトプットには、次の内容が含まれます。

- 配備アーキテクチャー。コンポーネントとネットワーク上のハードウェアおよびソフトウェアとの対応付けを示すハイレベルの設計資料。
- 実装仕様。配備構築の設計図として使用する詳細な仕様。
- 実装計画。エンタープライズソフトウェアソリューションの実装におけるさまざまな面を網羅した一連の計画と仕様。実装計画には、移行計画、インストール計画、ユーザー管理計画、テスト計画などが含まれます。

第 5 章を参照してください。

実装段階

実装段階では、配備設計で作成した仕様と計画に基づいて、配備アーキテクチャーを構築し、ソリューションを実装します。配備プロジェクトの性質に従って、このマニュアルでは次のタスクについて説明します。

- 複数のホストサーバーへの Access Manager のインストール
- サイトとしての Access Manager 配備の設定
- Access Manager でのロードバランサの使用法
- Access Manager セッションフェイルオーバーの実装
- セッション割り当て制限の設定
- セッションプロパティー変更通知の有効化
- 配備のチューニング

第 6 章を参照してください。

第 2 章

Access Manager のためのビジネス分析

Sun Java™ System Access Manager を使用すると、組織は、従業員、請負業者、顧客、およびサプライヤのためのアイデンティティー管理ソリューションを配備できます。ソリューションライフサイクルのビジネス分析段階では、ビジネス上の問題点を分析することでビジネス上の目標を定義し、その目標を達成するにあたってのビジネス要件とビジネス制約を明確にします。 この章で説明する内容は、次のとおりです。

ビジネス分析について

ビジネス分析は、ビジネス目標を明らかにすることから始まります。次に、解決が必要なビジネス上の問題を分析し、ビジネス目標を達成するために満たす必要のあるビジネス要件を特定します。また、目標の達成能力を制限するビジネス制約がある場合は、それらの制約についても考慮します。ビジネス要件とビジネス制約を分析することにより、一連のビジネス要件ドキュメントを作成します。

ここで作成された一連のビジネス要件ドキュメントを、技術要件段階で技術要件を導き出すための基盤として使用します。ソリューションライフサイクル全体にわたり、このビジネス分析段階で実施された分析に従って、配備計画、そして最終的にはソリューションが成功するかどうかを判定します。

Access Manager ビジネス要件の定義

ここでは、Access Manager のために考慮する必要のある特定のビジネス要件、つまり、Access Manager ソリューションに必要なビジネス要件について説明します。

Sun Java™ System Access Manager は、複雑な分散型アイデンティティー管理システムであり、適切な配備によって、企業の組織にまたがる広範なデータおよびサービスへのアクセスがセキュリティー保護されます。企業リソースの適正な制御を確実にするため、配備プロセスの適切な計画が必要になります。この章では、配備の計画方法について説明します。次の節で構成されています。

リソースの定義

アイデンティティー管理ソリューションには、組織全体にわたる多様なシステムが関係するため、Access Manager を適切に配備するにはさまざまなリソースが必要になります。配備プロセスに関係する、または必要とされる企業のリソースを、以下に示します。

人事情報

組織内のさまざまな取引関係および政治的な関係を考慮する必要があります。直接的または多面的な報告体制を備えたチームを編成する必要があります。通常、Access Manager の配備は、1 人のプロジェクトマネージャーと数人の専任システム管理者で構成される小規模なチームで行われます。これらの人々は、チームリーダー、さらには多数の関連プロジェクトの責任を担うオーナーに報告を行います。また、経営権を持つスポンサーに直接報告することもよくあります。このグループは往々にして、Sun の技術リソースや、必要に応じて使用される LOB アプリケーション管理者で構成される仮想のチームメンバーによって増強されます。

この構成では実際のニーズと完全には一致しない可能性もありますが、ほぼ一般的な配備チームモデルを表しています。個々の役割を必ずしも明確に識別する必要はありませんが、さまざまなスキルセットを表す次の抽象技術ロールを使用することで、Access Manager の典型的な配備チームをさらに定義できます。

経営権を持つスポンサー

従来より、アイデンティティー管理の配備を成功させるには、組織的および政治的な境界を超えた企業の決定権を持つ人物の承認やサポートが必要です。このため、経営権を持つスポンサーが管理に関係することは重要です。プランニングのためのミーティングは、配備に関する既得権を保持する人物からの見識を得る上で重要なプロセスです。プロジェクト計画を策定する際、成果が全体としての企業目標に沿っていることを確認してください。たとえば、コスト削減が主な経営目標である場合、現在のアイデンティティー管理コストに関する統計を収集し、パスワードリセット関連のヘルプデスク利用コストなどのコストを判断します。具体的な統計が入手できれば、配備チームが経営陣のサポートを得るために必要な、明確な投資収益率 (ROI) の定義に役立ちます。関連するほかの問題には、以下が含まれます。

- 誰がアイデンティティー管理の配備からメリットを得るか
- アイデンティティー管理ソリューションは、どのような組織上の問題を解決するか
- 配備を遅らせる可能性のある内部的な課題にどのように取り組んでいくか

多くの場合、アイデンティティー管理の概念や Access Manager 配備の価値を、ほかの経営陣にも納得させることが必要になります。経営面および技術面の「エバンジェリスト」は、新しいインフラストラクチャーを経営陣に売り込み、統合化への要求を喚起したり、インフラストラクチャーの変更を受け入れさせて、最終的な成功を収めるのに寄与します。

チームリーダー

プロジェクトの成功に責任を持つ当事者として、チームリーダーを 1 人選ぶ必要があります。このチームリーダーは、プロジェクトの目標を達成するという責任を担い、そのための権限を持ちます。このチームリーダーは、テクニカルリーダー、プロジェクトマネージャー、執行責任者などに論理的に分散されたロールである場合があります。このロールをどのように定義するとしても、その目標は配備プロセスが着実に前進し、成功していることを示して、経営陣の支援を維持することです。

プロジェクト管理

プロジェクトマネージャーは、スケジュールの調整を担当します。また、利用可能なサービス、コア IT グループの提供するサポート、およびさまざまな事業分野 (LOB) のアプリケーション統合に関連するスケジュールを管理します。この役割を担う担当者は、優れたコミュニケーション能力を持ち、企業の政治的側面を理解できる必要があります。プロジェクトマネージャーはまた、環境に追加される新規アプリケーションが円滑に機能するために、内部顧客のニーズとリソースの利用状況とのバランスを取る必要もあります。

LOB アプリケーションは、組織の運営に欠かすことのできないものです。通常、これらは、データベースおよびデータベース管理システムとの接続機能を持つ大規模なプログラムです。会計、サプライチェーン管理、およびリソース計画アプリケーションがこれに含まれます。現在、LOB アプリケーションは、ユーザーインタフェースを備えたネットワークアプリケーションや、電子メールや住所録などの個人向けアプリケーションとの接続機能を持つようになりつつあります。

システムアナリスト

システムアナリストは、Access Manager 配備に統合されるさまざまなデータやサービスの評価および分類を担当します。システムアナリストは、LOB アプリケーションのオーナーにインタビューを行い、プラットフォーム、アーキテクチャー、およびその配備スケジュールの技術要件に関する詳細情報を収集します。システムアナリストは、この情報を使用して顧客の要件を満たすようにアプリケーションを配備に統合する方法を計画します。システムアナリストは、さまざまなアプリケーションのアーキテクチャーおよびプラットフォームに関する広範な知識を持つ、IT ゼネラリストでなければなりません。また、Access Manager のアーキテクチャー、サービス、エージェント、および API に関する詳細な知識も必要になります。

事業分野別 (LOB) アプリケーション管理者

LOB アプリケーション管理者は、LOB アプリケーションに関する詳細な知識を持つと同時に、それらのアプリケーションを管理することのできる技術の専門家であり、Access Manager ポリシーエージェント (ポリシー適用ポイント) のアプリケーションへの統合を担当します。LOB アプリケーション管理者には、LOB アプリケーションのアーキテクチャー、統合ポイント、および適切なスケジュールに関する明確な意思伝達能力が必要です。通常、LOB アプリケーション管理者は、Access Manager ポリシーに示されるアクセス制御モデルの定義を担当します。この人物は、カスタムプログラミングを行って、Access Manager とそのアプリケーション (セッション調整など) 間の統合を拡張することもあります。さらに、通常は品質保証 (QA) および新たに配備された環境でのアプリケーションの回帰試験を担当します。

システム管理者

Access Manager を配備し、その可用性を維持する上で、適切なリソースが存在することは重要です。以下に示すレベルで、システム管理者が必要です。補助的な管理者には、Access Manager の配備先ソフトウェアコンテナの配備およびパフォーマンスを担当する Web コンテナ管理者も含まれます。

Access Manager 管理者

Access Manager 管理者は、Access Manager の配備と保守を担当します。この管理者は、共通サービスの可用性を保証し、一般的にインフラストラクチャーに必要な拡張を加え、特にポリシーとロールを設定します。また、ガイドラインを策定して統合作業のサポートに役立てるとともに、LOB アプリケーション管理者への技術サポートも行います。Java、XML、LDAP、HTTP、および Web アプリケーションアーキテクチャーの理解が必須です。

Directory Server 管理者

Access Manager の配備を考慮する前から、認証および承認に使用される企業のディレクトリサービスが組織内のグループにより管理されていることがよくあります。Directory Server 管理者は、ディレクトリサービスの可用性の管理だけでなく、アイデンティティー管理インフラストラクチャーのサポートに必要な変更も含め、現在定義されているLDAP スキーマやアイデンティティーデータへの追加や変更の受け入れおよび統合も担当します。

ハードウェア、データセンター、ネットワーク管理者

大規模な組織は、通常、ハードウェア、オペレーティングシステム、データセンター、およびネットワーク管理をミドルウェア管理から切り離すことでスケールメリットを追求します。この種の企業の場合、これらの管理者間で明確に意思を疎通させることが重要になります。配備を成功させる上で、特定のマシンにアクセスできることや、特定のネットワーク構成を確立することが重要な場合があります。これらの管理者に常にプロジェクトの主要管理点や要件を意識させることで、スムーズなロールアウトが可能になります。

独立系ソフトウェアベンダー

Sun Microsystems およびほかの独立系ソフトウェアベンダー (ISV) は、Access Manager の配備を成功させる上で重要なパートナーです。パッケージソフトウェアを購入することで、企業は複数の組織にまたがるソフトウェア開発を行うコストとリスクを軽減および分散させることができます。

ISV は、1 つ以上のタイプのコンピュータハードウェアまたはオペレーティングシステムプラットフォームで実行可能な製品を製造および販売します。プラットフォームを製造する企業 (Sun、IBM、Hewlett-Packard、Apple、Microsoft など) は、ISV を支援 し、サポートを提供します。

ISV に関係する当事者すべてにとって最大の関心事は、協力関係を強化して配備を成功させることです。Sun テクニカルサービスやほかの ISV に働きかけ、プロジェクトの立ち上げの支援や、以前の Access Manager 配備で得た知識を提供してもらってください。対策チーム (Access Manager のエンジニアと配備チームとの仲介役として機能できる) と率直な討議を行うとともに、テクニカルサービスを利用すると、投資を有効に活用したり、配備を成功させるのに役立ちます。

提携先のサードパーティー

Access Manager の連携管理機能を活用することを計画している場合、外部パートナーや提携しているサードパーティーと共同して作業を行います。連携管理機能の初期配備を、独自の内部配備とともに考慮してください。この場合、重要なのは、提供するビジネス機能を保持する LOB アプリケーションを含めること、および当事者すべての技術リソースとの通信を管理することです。弁護士も、関係する当事者間の良好な関係を築く助けになります。

資金の調達

多くの場合、配備プロジェクトのコスト面の責任はコア IT グループが担います。実際、内部資金を LOB アプリケーションからコアグループに移して、アイデンティティー管理プロジェクトの資金の一部にあてるのが一般的な方法です。ただし、単一の LOB アプリケーショングループが内部資金を提供する場合でも、より大きな組織のニーズと資金調達グループのニーズとのバランスを取る必要があります。

目標の設定

組織は、目標を設定することにより、Access Manager 配備が完了したあとのあるべき状態を定義します。配備の戦略とは、これらの目標に達するためのロードマップを計画し、目標に向かって進むことです。目標は、関係する当事者すべてが期待することを定義し、プロセスの初期にその承認を得て作成します。

一般に、アイデンティティー管理ソリューションでは、セキュリティーを拡張し、インフラストラクチャーの管理機能を向上させるとともに、コストを削減します。より具体的にいえば、Access Manager が組織に設定を許可する一般的な目標 (およびそのメリット) には、以下が含まれます。

- 予想されるデジタルアイデンティティー (従業員、パートナー、顧客) の増加に対応する、スケーラブルなインフラストラクチャーを実装する
- アイデンティティープロファイルの作成および管理を、独自データを制御する各グループと統合する
- ベンダーの統合、ユーザーの自己管理、および関連する管理コストによりコストを削減する
- アイデンティティープロファイルの迅速な終了により、セキュリティーを改善する
- セキュリティーモデルおよびアクセス権の透過性を改善する
- クリティカルシステムへのアクセスに必要な時間を短縮する
- 組織内部のロールまたは連携が変更されたら、クリティカルシステムへのユーザー権限を削除する

最終的に、これらの目標を、関係するグループすべてのモチベーションの理解および実地調査から得られた情報と結び付けて、配備用のインフラストラクチャーの設計に使用できます。また、これらを配備プロセス全体で使用して、当事者の関係を維持し、プロジェクトの支持を得られるようにします。

情報の収集

実地調査を行い、配備に統合するアプリケーションやデータストアに関する情報を収集できます。さらに、これらの部門間の情報収集は、特定の機能および目標を定義し、関係するグループのモチベーションの理解を深めるのに役立ちます。収集が済んだら、情報を設計の青写真として、および経営権を持つスポンサーから確実な承認を得るために活用できます。実地調査の際、以下のグループの支援を得られます。

- ユーザーは、日常業務で使用しているアプリケーションに関するフィードバックを提供する
- 人事担当者は、雇用および雇用終了処理に関する情報を提供する
- サポート担当者は、組織の境界をまたがる問題に関して貴重な情報を提供する
- アプリケーション管理者および開発者は、配備に統合する事業分野別 (LOB) アプリケーションに関する技術情報を提供できる
- ネットワーク管理者は、組織のパフォーマンスや標準に関する技術的基盤の知識を保持している

初期調査には、次の項目に関する情報の収集が含まれます。

- 31 ページの「ビジネスプロセス」
- 32 ページの「IT インフラストラクチャー」
- 32 ページの「仮想データ」

ビジネスプロセス

ビジネスプロセスとは、組織内のさまざまなグループがそれぞれの業務の実行を定義する手順です。プロセスには、次の手順を含めることができます。

- 給与の支給
- 購買および買掛金勘定
- 従業員の出張の承認
- 部門の予算管理
- 従業員の雇用終了

通常、これらのプロセスは、業務単位ごとに使用されるアプリケーションによりサポートされるため、これらのプロセスの評価は必須です。考慮すべき内容には、以下が含まれます。

- 現在のプロセスで遅延が発生するかどうか
- 同じ機能を実行する異なるプロセスが多数存在するかどうか
- 業務単位の境界をまたがってプロセスを標準化できるかどうか
- プロセスはどの程度複雑か。プロセスを集約したり簡略化したりできるか
- 現在のプロセスで組織上の変更を処理できるか

プロセスに加える変更はすべて、配備を開始する前に行う必要があります。

IT インフラストラクチャー

IT インフラストラクチャーには、Access Manager の配備に統合されるすべてのハードウェアサーバー、オペレーティングシステム、および統合化アプリケーションが含まれます。次の点について考慮します。

- Access Manager を利用するアプリケーション

  アプリケーションには、人事および会計用のアプリケーションなどの重要な内部アプリケーション、またはあまり重要性の高くない従業員のポータルを含めることができます。また、Access Manager の機能を利用するアプリケーションとして、機密性の高い財務情報と機密性の高くない販売物資の両方を処理する外部 B2B アプリケーション、またはクレジットカードデータや購入履歴に関係する B2C のショッピングカートがあります。

- Access Manager を利用するシステム

  アプリケーションの配備先のハードウェアとそのオペレーティングシステムについて考慮してください。Access Manager の配備には、アプリケーションを実行するための Web コンテナ、Sun Java System Directory Server (または既存のデータストア)、および Access Manager が最低限含まれます。また、企業リソースを使用して独自の Web コンテナを実行する、追加のハードウェアサーバーが含まれることもあります。さらに、セキュリティーを向上させるために、そこに Access Manager ポリシーエージェントをインストールすることもできます。

- 各部門が利用する Access Manager のサービス

  Access Manager 内に統合するデフォルトのサービスやカスタムサービスについて考慮します。ロールおよびポリシーの戦略は、部門ごとに割り当て、定義する必要があります。認証モジュールを評価する必要があります。カスタムサービスが存在する場合はそれを整備する必要があります。

さらに考慮する必要のある技術的な内容は、次のとおりです。

- インフラストラクチャー内に非互換性が存在するかどうか
- 現在のシステムで性能低下やダウンタイムが発生しているかどうか
- アプリケーションは十分にセキュリティー保護されているか
- ウイルスを制御する手段が存在するか
- アプリケーションは、ユーザーの資格に基づいてカスタマイズ可能か

詳細は、33 ページの「アプリケーションの評価」を参照してください。

仮想データ

仮想データとは、Access Manager にアクセスするプロファイル、Access Manager からアクセス可能な設定、およびAccess Manager によってセキュリティー保護されるデータなど、あらゆる状況で利用されるデータを指します。仮想データには、ユーザープロファイル (従業員、顧客など)、データおよびサービスアクセスルール、およびほかのタイプの企業データが含まれます。ただし、含まれるデータはこれだけに限定されるものではありません。

- Access Manager で保護する対象

Access Manager は、すべての種類のデータおよびサービスへのアクセスをセキュリティー保護します。管理者は、Access Manager データの表示や設定が可能なユーザーを制限したり、アプリケーション、ポータル、およびサービスへのアクセスを制御できます。

- Access Manager を利用するユーザー

  ユーザーには、従業員、ビジネスパートナー、サプライヤ、現在の顧客、および潜在顧客が含まれます。各ユーザーが保持するプロファイルには、最低限、ユーザー ID とパスワードが含まれます。従業員は、外部の販売情報にアクセスする顧客よりも、明らかにより広範で機密性の高いプロファイルを保持します。

- アクセス可能なデータ

  データには、公開情報、内部情報、機密情報、秘密データなどが含まれます。さらに、外部 Web サイトの販売情報、従業員の機密プロファイル、企業のリソースを保護するアクセスルール、サーバー設定情報、連携顧客プロファイルなどが含まれる場合もあります。

- 信頼すべきデータの入手元

  多くの場合、異なる種類のデータを定義する複数のスキーマが使用されます。これらの定義を、配備内で調和させる必要があります。データの所有権の問題を銘記しておき、必要な場合には、さまざまな LOB アプリケーションがデータの制御を維持できるようにしてください。より大規模な組織にはすべてのサービスが重要であるため、企業全体を代表するサービスを提供するために、サテライトグループの需要のバランスを取ることが重要です。

さらに考慮する必要のある技術的な内容は、次のとおりです。

- 複数の属性内で同じ情報が定義されているか
- ユーザーは組織の境界をまたがる複数のプロファイルを保持しているか
- データストアは、ファイアウォールの内側に存在するか
- データは異なるデータストア間で一貫しているか
- 新規データが追加されたり、既存のデータが変更されたりする頻度はどれくらいか

詳細は、35 ページの「データの分類」を参照してください。

アプリケーションの評価

アイデンティティー管理サービスは、一般に、拡張されたシステムを構成する企業および業務単位向けアプリケーションを備えた、集中管理された IT 機能として提供されます。このシステム階層の維持には、サーバーインフラストラクチャーを管理および保守するコア IT グループ、および LOB アプリケーションを保守する従業員のサテライトグループが関係します。

大規模な組織には、たいてい、数百 (または数千) の内部アプリケーションがあります。それらのすべてを評価するには時間と費用がかかります。アプリケーションの調査を実行する場合は、次の条件を満たすアプリケーションを集中的に調査してください。

- 組織にとって特に大きな価値を持つアプリケーション
- シングルサインオンインフラストラクチャーへの統合で、メリットが期待されるアプリケーション
- 組織内部の標準的なプログラミングおよび配備プラットフォームを示すアプリケーション
- 通常、アイデンティティー管理インフラストラクチャーに受け入れられるアプリケーション
- 現在、配備の初期段階にあり、論理的に Access Manager の配備とスケジュールが一致する可能性のあるアプリケーション

スプレッドシートを作成すると、最も将来性の高いアプリケーションから得られた情報の整理に活用できます。全体的な測定基準を策定して、アプリケーション間の統合化の複雑性を比較できます。この測定基準により、アプリケーションがどの程度配備に適しているかを判断できます。適合性の高いアプリケーションの例は、セキュリティー目的で Access Manager ポリシーエージェントがインストールされたアプリケーションサーバーに認証を委任する Web アプリケーションです。すべてのユーザー情報は、LDAP ディレクトリに格納されます。

適合性の低いアプリケーションの例には、メインフレームコンピュータ上で動作する、テキストベースのインタフェースを備えたアプリケーションがあります。この場合、メインフレームアプリケーションの新しいバージョンを待つ間に、ほかのアプリケーションを統合する方が好都合です。

次の節では、組織のアプリケーションを評価する際に収集可能な情報の種類について説明します。この手順は、保護されるリソースを判別するのにも役立ちます。

プラットフォームの情報

既存のテクノロジおよびハードウェアに基づく一般的なプラットフォーム情報を使用して、アプリケーションが統合化に適切かどうかを評価できます。収集されるプラットフォーム情報には、次のものが含まれます。

- アプリケーションが動作するオペレーティングシステム (バージョンを含む)
- アプリケーションが動作する Web コンテナ (バージョンを含む)
- アプリケーションの開発に使用したプログラミングモデル(Java、ASP/.NET、C など)
- アプリケーションをアップグレードする計画があるかどうか。あるとすれば、そのスケジュールはいつか

LOB アプリケーションも、サードパーティー製のアプリケーション (ポータル、コンテンツ管理データベース、人事管理システムなど) を実行している場合があります。これらのアプリケーションが、Access Manager ポリシーエージェントでサポートされるプラットフォーム上に常に配備されているとは限りません。ポリシーエージェントが必要な場合は、これらのアプリケーションの配備基準を確認し、ポリシーエージェントの可用性に基づいてその配備のスケジュール設定を行ってください。

セキュリティーモデル

LOB アプリケーション内で使用する既存のセキュリティーモデルをドキュメント化しておくことは重要です。通常、外部の認証や承認を使用するアプリケーションは、外部のディレクトリサービスに依存するアプリケーションとともに、配備の候補になります。セキュリティー情報には、次のものが含まれます。

- 現在どの認証メカニズムを使用しているか
- 特殊な認証要件 (2 ファクター認証など) は存在するか
- 外部認証メカニズム用のプラグイン可能なインタフェースが存在するか
- 現在どの承認メカニズムを使用しているか
- 承認を外部で行うことは可能か (または、そうすべきか)
- どのユーザーデータリポジトリを使用しているか。それを外部で行うことは可能か
- 誰がアプリケーションにアクセス可能か。既存のロールまたはグループが所定の位置に存在しているか。どのような特殊条件が存在する場合、彼らにアクセスが許可されるのか

セッションのライフサイクル

アイデンティティーのセッションライフサイクルは、認証アプリケーションを評価する場合に考慮する重要な項目です。ユーザーセッションが作成、管理、および破棄される方法について明確に把握しておいてください。アプリケーションの統合時に参照できるよう、このプロセスを明確にドキュメント化してください。

カスタマイズおよびブランド設定

アプリケーションにブランド設定やルックアンドフィールに関する特定の要件がある場合は、それについて検討します。多くの場合、アプリケーション単体のルックアンドフィールを重視するか、ユーザーにとって一貫した使い勝手を重視するかは重要な問題です。カスタマイズやブランド設定を行う場合は、そのための時間もスケジュールに組み込む必要があるため、アプリケーションの評価にその要件が含まれているかどうかを確認してください。

データの分類

アプリケーションの分析と、その適切なレベルへの分類を完了したら、次に、これらのアプリケーションから提供されるデータおよびサービスの分類を開始する必要があります。この情報は、セキュリティーモデルの構築に使用されます。分類自体は、データおよびサービスを分類するプロセスです。それに続き、既存の認証および承認システムのカタログ作成が行われます。

この前者のプロセスに、アプリケーションの評価で収集された情報が使用されます。収集した情報をさまざまなセキュリティー層に編成するのは、良い方法です。これらの層は、データの紛失、アプリケーションの妥協的使用、誤用、その他の不正なアクセスタイプに関係したリスクの量を示すものとなります。正しく定義されたカテゴリを使用すると、リソースをセキュリティモデルにマッピングして、認証および承認要件を組み込む作業が簡単になります。

データまたはサービスは、4 つのセキュリティレベルに分類されます。X 軸はデータまたはサービス、Y 軸は関連付けられるセキュリティレベルを表します。層 1 は、セキュリティーが最小であることを示します。これは、公開された Web サイトに適用可能なデータです。一方、層 4 は、セキュリティーが最大であることを示します。これは財務や人事 (HR) データなどに適用されます。実際の組織の分類に使用される層はこれより多い場合も少ない場合もありますが、この図は、大量のデータには関連するリスクも低く、そのためセキュリティー要件も少なくなるという典型的な例を示しています。関連するリスクが大きくなるにつれ、セキュリティー要件も多くなります。通常、高度なセキュリティー要件が必要なデータはごくわずかであり、多くのデータはセキュリティー要件をほとんど必要としません。

次の図は、典型的な組織内のデータおよびサービスのセキュリティー要件を示しています。

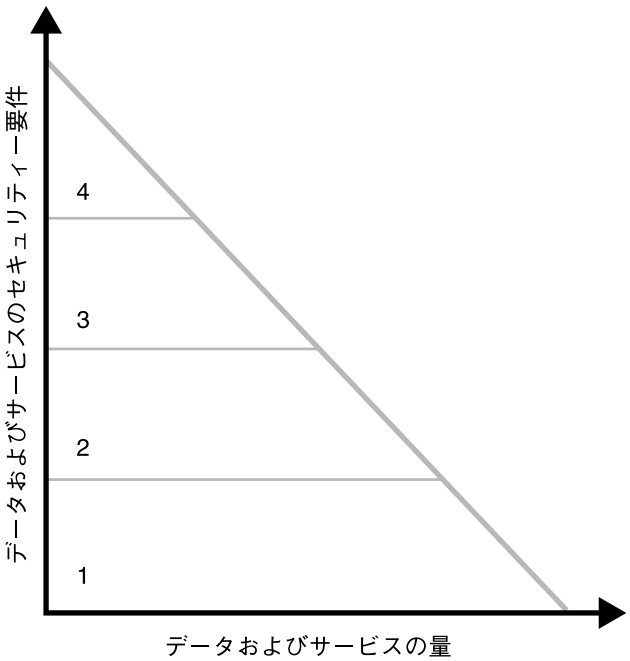


図 **2–1** データおよびサービスのセキュリティー要件

認証および承認機能を割り当てることができるように、データおよびサービスタイプの機能グループを構築しようとしていることを念頭に置いてください。層の数が多すぎるとプロセスが過度に複雑になり、層の数が少なすぎると柔軟性に欠けたものにな

ります。さらに、ネットワーク上に配置すること自体が非常に危険なデータもあることも考慮しておくことは重要です。可能な場合は、内部で利用可能なデータと外部で利用可能なデータを明確に区別するようにしてください。これらの層を構築する際、認証および承認要件とともに、データのアクセス時刻およびネットワーク上の位置などの修飾条件も念頭に置くようにしてください。

認証へのマッピング

データをセキュリティーレベルに応じて分類できたら、次の段階は、認証および承認メカニズムの一覧を作成することです。利用可能な認証メカニズムに関する手持ちのリストを使用して、これらのメカニズムを定義済みのセキュリティー層に関連付けます。たとえば、次の関連付けは、前の図で分類されたデータに対応しています。

- 層 1 のデータは、アクセス制御なしの匿名認証が適切と考えられます。
- 層 2 のデータは、パスワード保護のみを必要とします。
- 層 3 のデータは、ハードトークンまたは証明書認証が必要です。
- 層 4 のデータは、マルチファクター認証を必要とします。または、ネットワーク上には絶対に配置しないようにします。

認証要件とデータやサービスの分類とのマッピングが、単純で明快なものであるようにしてください。そうでない場合、一致しない項目間で共通の基準を見つけるようにしてください。論理的な差異が存在するなら、ためらわずに複数の図を作成してください。

たとえば、イントラネットとエクストラネット用のアプリケーションでは、それぞれ別個の図を作成できます。また、人事 (HR) や財務などの機能セキュリティドメインに基づいて、データを分類することもできます。このようなデータ分類手法は万能とは言えませんが、セキュリティー要件を理解し、論理的に管理可能なグループにマッピングする場合に役立ちます。

承認へのマッピング

アプリケーション評価により入手したデータを使用して各アプリケーションを検証し、スケーラブルな承認モデルを決定します。通常、最善の方法は、複数のアプリケーションにわたって使用する共通のグループやロールを見つけることです。これらのグループやロールが、組織内の機能ロールにマッピングされていると理想的です。また、これらのグループやロールのソース (メンバーシップデータの存在位置およびそのモデリング方法) を判別する必要もあります。たとえば、データは Sun Java System Directory Server 内に存在する可能性があります。

存在しない場合は、カスタムプラグインが必要になる場合があります。堅牢なグループモデルが存在する場合、最初に、各アプリケーションを既存のグループまたはロールに関連付けてください。存在しない場合は、最初にロールまたはグループメカニズムを計画し、機能的観点からユーザータイプ間の共通の関係を見つけてから、特定のアプリケーションにとりかかります。作業の完了時点で、以下のものが入手できます。

- 既存のグループおよびロールの明確なマッピング
- データの存在する位置、その品質と管理に関する権限を持つ人物に関する明確な理解
- 配備を促進したり、配備のコストや複雑性を低減したりするために作成する必要のある新しいグループまたはロールについての明確な理解
- 既存および将来のグループメカニズムの、分類されたアプリケーションへのマッピング
- 特定のグループやロールへのアクセスを可能にするため、アプリケーションが必要とする追加条件に関する注意事項

この基本的なセキュリティーモデル (認証および承認メカニズムとの関連を利用したデータの分類) を使用して、配備を実行するスケジュールをまとめることができます。

スケジュールの作成

収集した情報から、予備段階のスケジュールを作成する必要があります。次の節では、一般的な配備スケジュールを作成するための手順について説明します。

配備の設計

スケジュールのこの段階では、これまでに入手した概念、ビジネスニーズ、およびユーザー要件を適切なコンテキストに配置します。ここで、配備の全体像が明確になります。コンポーネントが記述され、技術要件が定義され、完成したアーキテクチャーが立案作成されます。この設計段階を開始する 2 つの方法は、ログイン時の画面案を作成すること、およびデータフローチャートを作成することです。

コンセプト証明

コンセプト証明を使用すると、設計をビジネス環境でテストできます。組織には、たいていの場合、期待される結果と一体になった設定済みの一連のテストケースである、テストケースデータベースが存在します。このテストデータベースには、コンセプト証明を適用できます。そして、すべてが順調に進めば、ドキュメント化された結果は新しい結果と等しくなります。コンセプト証明の目的は、「配備の設計」で提起されたすべての疑問に答えることにより、すべてのニーズを効果的に、最小限のリスクで満たせることを証明することです。

これは一般にすばやく実行されるため、限られたデータセットに基づいて設計を改良するための十分な時間を取ることができます。通常は、コンセプト証明とそれに続く設計改良を、数回繰り返す必要があります。最後に実行するコンセプト証明は、いく

つかの内部アプリケーションが統合されたものになるはずです。企業の共有サービス統合は、一般に、初期採用者によるサインオン標準モデル、次に一般の参加者によるサインオン標準モデル、最後に残された者たちによるサインオン標準モデルに準拠したものになります。初期の採用者で成功すると、そのアプリケーションを一般の採用での参照用として使用しやすくなります。

初期採用

ミッションクリティカルなアプリケーションや収益を創出するためのアプリケーションは、最初のアプリケーションとして選択すべきではありません。よりリスクの少ない戦略は、重要なアプリケーションの中でもロールアウト時に問題が発生しても企業運営に大きな混乱をきたさないものを選択することです。たとえば、部門ポータルは、会計年度末の会計システムよりも、シングルサインオン (SSO) ロールアウト用の拠点として適しています。

また、プロセスの弱点を排除し、結果が立証され、成功が迅速に認識されるように、初期段階でアプリケーションロールアウトの数を制限してください。最良のロールアウトの戦略は、可視性を最大限に高めながら、組織上のリスクを最小限に抑えることです。このため、製品に関する適切な経験を持つ配備チームがクリティカルアプリケーションを担当するようにします。

一般の参加

配備プロジェクトは単一のアプリケーションで始まりますが、多目的システムを構築できるように、他の内部顧客の要件も同時に評価する必要があります。中心的な IT グループは、より大規模な組織を代表するサービスを提供するため、サテライトグループのさまざまな基準およびスケジュールを受け入れることができなければなりません。スケジュールは十分に大きなウィンドウに表示し、サテライトグループが変更およびアップグレードを自分たちのアプリケーションの配備および品質保証 (QA) サイクルに組み入れる時間を取れるようにする必要があります。

製品環境

コンセプト証明が終了したら、改良された設計を製品環境内にレプリケートできます。製品環境の目的は、通常の環境で設計の機能を実証し、正しく動作することを確認することです。この環境は、コンセプト証明で観察された動作、および配備設計で定義された動作と比較されます。このテストは、安定性を確認する目的でも行われます。

評価が行われ、レポートが生成されます。初期採用アプリケーションは、準備段階が完了しているため、製品環境でも稼働します。新規アプリケーションを、テスト段階から製品段階へ、徐々に移行させてゆきます。その他のアプリケーションは、初期採用と同じようにコンセプト証明サイクルで稼働させたあとで、徐々に製品環境に追加されていきます。

スケジュールはプロジェクトの複雑さに応じて変動するため、サンプルスケジュールは利用できません。ただし、このプロセスは通常、2 ～ 3 か月の期間をかけて行われます。

配備ロードマップ

Access Manager 統合を確実に成功させるには、詳細な計画が必要不可欠です。このプロセスには、ハードウェア、現在配備されているアプリケーション、アイデンティティーデータ、およびアクセス階層に関する情報の収集が含まれます。Access Manager の配備は、次の各段階に細分化できます。

1. 次に示すようなビジネスの目的を特定する
   - 業務効率を改善する
   - データのセキュリティーを確保する
   - 組織内の範囲と関係を理解し、ビジネス目的のサポートに必要な行動の変化を分析することにより、確実に生産性を進展させる
2. ビジネス目的の達成に必要なテクノロジサービスおよびツールを列挙することによって、高度なテクノロジ分析を開発し、それをビジネスの目的に割り当てる
3. 次に示すようなテクノロジサービスの具体策を定義する
   - パーソナライズにより蓄積された従業員の履歴およびデータを保管する
   - アイデンティティー管理を使用して、パスワード同期およびアイデンティティー管理を実行する
   - ロールの戦略を開発して、企業のセキュリティ保護を実現する
4. 統計の精度、予測可能性、範囲、費用、影響、複雑性、行動、インフラストラクチャー、利点、サポート、依存関係などの項目に基づいて、各具体策に優先順位をつける

第 3 章

技術要件

ソリューションのライフサイクルにおける技術要件段階では、使用状況を分析し、ユースケースを特定して、提案された配備ソリューションの QoS 要件を決定します。この章では、Sun Java™ System Access Manager 7 2005Q4 での、この処理に関する要件の大まかな技術概要を示します。次の節で構成されています。

配備オプション

Access Manager の配備を計画する際に、組織が考慮する必要のある重要な要素がいくつかあります。これらは、通常、リスク評価および成長戦略と関連があります。次に例を示します。

- 配備環境でどれくらいのユーザーをサポートすることが見込まれているか。予測される成長率はどれくらいか

  ユーザーの成長およびシステムの使用状況を監視し、この実データを予測されるデータと比較して、現在の能力で予測される成長を処理可能であることを確認することは重要です。
- 環境にサービスを追加する計画はあるか。それは現在の設計に影響を及ぼす可能性があるか

  これで、開発中のアーキテクチャーは、現在のサービスに対して最適化されます。将来の計画も検討する必要があります。

さらに、アーキテクチャーは、以降の節で説明する目標を達成するための基礎を提供する必要があります。

セキュリティー

セキュリティーの確保された内部および外部ネットワーク環境を計画している場合には、以下の選択肢について考慮してください。

- サーバーベースのファイアウォールは、サーバーへのポートレベルのアクセスを制限することにより、追加セキュリティーレイヤーを提供します。標準のファイアウォールと同様、サーバーベースのファイアウォールは、着信および送信 TCP/IP トラフィックを制限します。
- 最小化とは、システムの脆弱性が悪用される可能性を最小限に抑えるために、不要なソフトウェアおよびサービスをすべてサーバーから削除することをいいます。
- 分割 DNS インフラストラクチャーには、1 つのドメイン内に 2 つのゾーンが作成されます。1 つのゾーンは組織の内部ネットワーククライアントにより使用され、もう1 つのゾーンは外部ネットワーククライアントにより使用されます。この手法は、より高度なセキュリティーレベルを実現するために推奨されています。DNS サーバーでロードバランサを使用することによって、パフォーマンスを向上させることもできます。

高可用性

IT の配備では、ユーザーに対して可用性を継続するとともに、SPOF (Single Point Of Failure) を発生させないことが重要です。可用性を高めるための手法は、クラスタリングやマルチマスターレプリケーションなど、製品ごとに異なります。望ましい高可用性とは、システムやコンポーネントが一定の期間、連続的に使用可能であるということです。システムは一般に複数のホストサーバーで構成されますが、ユーザーには 1 つの高可用性システムのように見えます。すべてのアプリケーションが 1 台のサーバーで動作する、最小構成の配備の場合、SPOF に含まれる要素として次のものが考えられます。

- Access Manager Web コンテナ
- Directory Server
- Java™ 仮想マシン (JVM)
- Directory Server ハードディスク
- Access Manager ハードディスク
- ポリシーエージェント

高可用性を実現する場合は、バックアップやフェイルオーバー処理、およびデータストレージやデータアクセスを中心に計画します。ストレージに関する 1 つの手法は、RAID (redundant array of independent disks) です。より高い可用性が求められるシステムでは、システムの各部分が適切に設計され、本稼働に先立ち、十分にテストされていることが必要です。たとえば、テストが十分ではない新規のアプリケーションプログラムほど、本番での稼働中に、システム全体に影響するエラーを引き起こす可能性が高くなります。

クラスタリング

クラスタリングとは、単一の高可用性システムを構築するために複数のコンピュータを使用することを指します。Sun Java System Directory Server のデータストアでは、クラスタリングが非常に重要な手法となる場合が多くあります。たとえば、クラスタ化された 1 組の MMR サーバーでは、可用性が確保されることにより、各マスターインスタンスの可用性を向上させることができます。

スケーラビリティー

「水平スケーリング」は、複数のホストサーバーを接続して1 つの装置として動作させることで実現します。ロードバランスに対応したサービスは、サービスの速度と可用性が向上するため、水平スケーリングが行われていると見なされます。一方、「垂直スケーリング」は、1 台のホストサーバー内部にリソースを追加することにより、既存のハードウェアの容量を拡張します。スケーリング可能なリソースには、CPU、メモリー、および記憶装置が含まれます。水平スケーリングおよび垂直スケーリングは相互排他的なものではないため、配備ソリューションとして両者を併用することができます。通常、環境内のサーバーに容量いっぱいまでリソースがインストールされることはないため、垂直スケーリングを使用してパフォーマンスを改善します。サーバーの容量が限界に近づいた場合も、水平スケーリングを使用して、ほかのサーバーに負荷を分散することができます。

ハードウェア要件

Access Manager を配備する場合の最小構成は、Access Manager と Sun Java System Web Server などの Web コンテナを実行する 1 台のホストサーバーです。Directory Server が稼働するサーバーは、Access Manager と同じでも、違ってもかまいません。複数のサーバーが配備されている環境では、Access Manager インスタンスとそれに対応する Web コンテナは異なるホストサーバーにインストールされ、クライアントの要求は、ロードバランサによって各 Access Manager インスタンスに分配されます。通常、Directory Server と Access Manager は別々のサーバーにインストールされます。

パフォーマンスを最適化するために、Access Manager は 100M バイト以上の Ethernet ネットワーク上で実行してください。Access Manager と 1 つのWeb コンテナを実行する Access Manager 配備の最小構成には、1 個以上の CPU を搭載している必要がありますが、5 個以上の CPU を搭載してもプロセッサのパフォーマンスはそれほど向上しません。ホストサーバーごとに 2 ～ 4 個の CPU を強くお勧めします。ソフトウェアの基本的なテストを実行するために、512M バイト以上の RAM が必要です。

実際の配備では、スレッド、Access Manager SDK、HTTP サーバー、およびほかの内部処理用に 1G バイトの RAM、基本操作およびオブジェクト割り当て領域に 2G バイトの RAM、さらに 10,000 並行セッションごとに 100M バイトの RAM が推奨されて

います。各 Access Manager は、並行セッションが 100,000 でキャップアウトすることが推奨されており、それ以降は、水平ロードバランスを適用する必要があります (32 ビットアプリケーションの 4G バイトメモリー制限を前提とする)。

ソフトウェア要件

Access Manager には、次のような固有のソフトウェア要件があります。

サポートされるリリース、必要なパッチ、および既知の制限を含むソフトウェア要件の最新情報については、『Sun Java System Access Manager 7 2005Q4 リリースノート』を参照してください。

オペレーティングシステム要件

Access Manager 7 2005Q4 は、次のプラットフォームでサポートされています。

- Solaris™ Operating System (OS)、 SPARC® ベースシステム、バージョン 8、9、および 10
- Solaris™ OS、x86 プラットフォーム、バージョン 9 および 10
- Red Hat™ Linux、Advanced Server および Enterprise Server

OS パッチおよびパッチクラスタのダウンロード方法については、`http://sunsolve.sun.com/` にある SunSolve Online にアクセスしてください。

現在 Solaris システムにインストールされているパッチを表示するには、`showrev -p` コマンドを使用します。

Web コンテナ要件

Access Manager 7 2005Q4 は、完全インストール、または SDK のみのインストールのいずれかでの使用に次の Web コンテナをサポートしています。

- Sun Java System Web Server
- Sun Java System Application Server

- BEA WebLogic
- IBM WebSphere Application Server

これらの Web コンテナのサポートされているバージョンについては、『Sun Java System Access Manager 7 2005Q4 リリースノート』を参照してください。

ポリシーエージェントを Access Manager Web コンテナにインストールする場合は、約 10M バイトのディスク容量が使用されます。Web コンテナを設定するときには、この追加容量を考慮に入れる必要があります。詳細は、『Sun Java System Access Manager Policy Agent 2.2 User's Guide』を参照してください。

Directory Server 要件

Access Manager 7 2005Q4 には、LDAP ディレクトリサーバーに対する次の要件があります。

- Access Manager 情報ツリーが Sun Java System Directory Server に格納されていること。情報ツリーには次の情報が含まれます。
  - ユーザー認証の方法
  - ユーザーがアクセス可能なリソース
  - ユーザーがリソースへのアクセス権を取得したあとに、アプリケーションで使用可能になる情報
- Access Manager には、ユーザーやグループなどのユーザーデータを格納するためのアイデンティティーリポジトリが必要です。Access Manager 7 2005Q4 は、Sun Java System Directory Server または LDAP バージョン 3 (LDAP v3) 互換のディレクトリサーバーをアイデンティティーリポジトリとして使用できます。

Access Manager 情報ツリーおよびアイデンティティーリポジトリについては、『Sun Java System Access Manager 7 2005Q4 Technical Overview』を参照してください。

Java Development Kit (JDK) ソフトウェア要件

Access Manager 7 2005Q4 に必要な JDK ソフトウェアに固有のバージョンについては、『Sun Java System Access Manager 7 2005Q4 リリースノート』を参照してください。

Access Manager セッションフェイルオーバー要件

Access Manager セッションフェイルオーバーを実装することを計画している場合、以下のコンポーネントが必要です。

- Access Manager を実行する Web コンテナ: Sun Java System Web Server、Sun Java System Application Server、IBM WebSphere Application Server、または BEA WebLogic。

- Sun Java System Directory Server: すべての Access Manager インスタンスが、同じ Directory Server にアクセスする必要があります。
- Sun Java System Message Queue。Message Queue ブローカクラスタは、Access Manager インスタンスとセッションストアデータベースの間のセッションメッセージを管理します。
- Sleepycat Software, Inc. の Berkeley DB (`http://www.sleepycat.com/`) が、デフォルトのセッションストアデータベースです。Sun Java Enterprise System 2005Q4 リリースによって配布されるバーションを使用します。

Access Manager のセッションフェイルオーバーは以下のプラットフォームでサポートされています。

- Solaris™ OS、SPARC® ベースシステム、バージョン 8、9、および 10
- Solaris™ OS、x86 プラットフォーム、バージョン 9 および 10
- Red Hat™ Linux、Advanced Server および Enterprise Server

これらのプラットフォームおよびコンポーネントがサポートされるバージョンに関する最新の情報については、『Sun Java System Access Manager 7 2005Q4 リリースノート』を参照してください。

詳細は、89 ページの「Access Manager セッションフェイルオーバーの実装」を参照してください。

Web ブラウザ要件

Access Manager 管理者およびエンドユーザーは、Web ブラウザを使用して、管理タスクおよびユーザー管理タスクを実行します。このリリースでサポートされる Web ブラウザについては、『Sun Java System Access Manager 7 2005Q4 リリースノート』を参照してください。

Access Manager スキーマ

スキーマとは、データに課されるルールセットのことで、通常はデータの格納方法の定義に利用されます。Sun Java System Directory Server は LDAP (Lightweight Directory Access Protocol) スキーマを使用して、データの格納方法を定義します。オブジェクトクラスは、LDAP スキーマ内の属性を定義します。Directory Server では、各データエントリは、内部の属性セットを記述および定義するオブジェクトのタイプを指定するため、1 つ以上のオブジェクトクラスを保持する必要があります。基本的に、各エントリは、属性セットとその対応する値、およびこれらの属性に対応するオブジェクトクラスのリストになります。

Access Manager は、Sun Java System Directory Server をデータリポジトリとして使用します。このリポジトリには Directory Server スキーマを拡張する Access Manager スキーマが組み込まれています。Access Manager のインストール時に、`ds_remote_schema.ldif` と `sunone_schema2.ldif` のファイルで構成される Access Manager スキーマは、Directory Server スキーマと統合されます。`ds_remote_schema.ldif` ファイルは、Access Manager が固有に使用するLDAP オブジェクトクラスと属性を定義します。`sunone_schema2.ldif` ファイルは、Access Manager LDAP スキーマのオブジェクトクラスと属性をロードします。

`ds_remote_schema.ldif`、`sunone_schema2.ldif` 、およびその他の Access Manager LDIF ファイルは、以下のディレクトリにあります。

- Solaris システム: `/etc/opt/SUNWam/config/ldif`
- Linux システム: `/etc/opt/sun/identity/config/ldif`

マーカーオブジェクトクラス

Access Manager コンソールを使用して作成し、Directory Server 内に格納したアイデンティティーエントリには、マーカーオブジェクトクラスが追加されます。マーカーオブジェクトクラスは、指定されたエントリを Access Manager が管理するエントリとして定義します。オブジェクトクラスは、サーバーやハードウェアなど、ディレクトリツリーのほかの面には影響を与えません。また、既存のアイデンティティーエントリは、これらのオブジェクトクラスを含めるようにエントリを変更しない限り、Access Manager を使用して管理することはできません。マーカーオブジェクトクラスについては、『Access Manager 開発者ガイド』を参照してください。既存の Directory Server データを Access Manager で使用するために移行する方法については、『 Sun Java Access Manager 6 2005Q1 Migration Guide』を参照してください。

管理ロール

LDAP エントリの委任された管理 (Access Manager 内の各アイデンティティー関連オブジェクトにマップされる) は、定義済みのロールおよびアクセス制御命令 (ACI) を使用して実装されます。デフォルトの管理ロールおよびその定義済み ACI は、Access Manager のインストール時に作成され、Access Manager コンソールを使用して表示および管理できます。Access Manager 7 2005Q4 のレルムモードでは、ロールは ACI ではなくポリシーに依存します。

Access Manager のアイデンティティー関連オブジェクトが作成されると、適切な管理ロール (および対応する ACI) も作成され、そのオブジェクトの LDAP エントリに割り当てられます。そのあと、ロールは、そのオブジェクトのノード制御を管理する個々のユーザーに割り当てることができます。たとえば、Access Manager が組織を新規作成すると、いくつかのロールが自動的に作成され、Directory Server に格納されます。

- 組織管理者は設定済み組織のすべてのエントリに対する読み取りアクセス権と書き込みアクセス権を持っています。

- 組織のヘルプデスク管理者は、設定済み組織のすべてのエントリに対する読み取りアクセス権、およびこれらのエントリ内の `userPassword` 属性に対する書き込みアクセス権を持っています。
- 組織のポリシー管理者は、組織のすべてのポリシーに対する読み取りアクセス権と書き込みアクセス権を持っています。

これらのロールのいずれかをユーザーに割り当てると、そのロールに与えられたすべてのアクセス権がユーザーに与えられます。

次の表に、Access Manager 管理ロール、および各ロールに適用される権限の要約を示します。

表 3–1 デフォルトおよび動的なロールとそのアクセス権

ロール	管理サフィックス	アクセス権
最上位組織の管理者 (amadmin)	ルートレベル	最上位組織内のすべてのエントリ (ロール、ポリシー、グループなど) に対する読み取りおよび書き込みアクセス権。
最上位組織のヘルプデスク管理者	ルートレベル	最上位組織内のすべてのパスワードに対する読み取りおよび書き込みアクセス権。
最上位組織のポリシー管理者	ルートレベル	すべてのレベルのポリシーに対する読み取りおよび書き込みアクセス権。参照ポリシー作成を委任するため、下位組織により使用されます。
組織管理者	組織のみ	作成された下位組織内のすべてのエントリ (ロール、ポリシー、グループなど) に対する読み取りおよび書き込みアクセス権のみ。
組織のヘルプデスク管理者	組織のみ	作成された下位組織内のすべてのパスワードに対する読み取りおよび書き込みアクセス権のみ。
組織ポリシー管理者 (Organization Policy Admin)	組織のみ	作成された下位組織内のすべてのポリシーに対する読み取りおよび書き込みアクセス権のみ。
コンテナ管理者 (Container Admin)	コンテナのみ	作成されたコンテナ内のすべてのエントリ (ロール、ポリシー、グループなど) に対する読み取りおよび書き込みアクセス権のみ。
コンテナヘルプデスク管理者 (Container Help Desk Admin)	コンテナのみ	作成されたコンテナ内のすべてのパスワードに対する読み取りおよび書き込みアクセス権のみ。

表 3–1 デフォルトおよび動的なロールとそのアクセス権　(続き)

ロール	管理サフィックス	アクセス権
グループ管理者	グループのみ	作成されたグループ内のすべてのエントリ (ロール、ポリシー、グループなど) に対する読み取りおよび書き込みアクセス権のみ。
ピープルコンテナ管理者	ピープルコンテナのみ	作成されたピープルコンテナ内のすべてのエントリ (ロール、ポリシー、グループなど) に対する読み取りおよび書き込みアクセス権のみ。
ユーザー (自己管理者)	ユーザーのみ	ユーザーエントリ内のすべての属性に対する読み取りおよび書き込みアクセス権のみ (nsroledn や inetuserstatus などのユーザー属性を除く)。

グループベースの ACI の代わりにロールを使用すると、効率を高め、保守の手間を少なくすることができます。フィルタ処理されたロールは、ユーザーの nsRole 属性の確認のみを行うため、LDAP クライアントの処理が簡略化されます。ロールは、そのメンバーの親のピアでなければならない、という範囲制限の影響を受けます。

デフォルトACI については、Access Manager コンソールのオンラインヘルプを参照してください。

Access Manager の管理アカウント

Access Manager のインストール時には、次の管理アカウントが作成されます。

- 管理者ユーザー ID (amadmin) は、Access Manager の最上位管理者で、Access Manager によって管理されるすべてのエントリに無制限にアクセスできます。デフォルト名の amadmin を変更することはできません。

  インストール中に、amadmin のパスワードを入力する必要があります。インストール後に amadmin のパスワードを変更するには、Directory Server コンソールか ldapmodify ユーティリティーを使用します。

- LDAP、メンバーシップ、およびポリシーサービスのバインド DN ユーザー (amldapuser) は、すべての Directory Server エントリに対する読み取りおよび検索アクセス権を持つ管理ユーザーです。デフォルト名の amldapuser を変更することはできません。

  インストール中に、amldapuser のパスワードを入力する必要があります。amadmin と同じパスワードは使用しないでください。インストール後に amldapuser のパスワードを変更するには、Directory Server コンソールか ldapmodify ユーティリティーを使用します。

  amldapuser のパスワードを変更した場合は、LDAP 認証サービスとポリシー設定サービスにも、この変更を反映させる必要があります。amldapuser は、これらのサービスで使用されているデフォルトユーザーだからです。この変更は、これ

らのサービスを登録している組織ごとに行う必要があります。

- プロキシユーザー (puser) は、任意のユーザー (組織管理者またはエンドユーザーなど) の権限を引き受けることができます。
- 管理ユーザー (dsameuser) は、Access Manager SDK が、特定のユーザーにリンクされていない Directory Server 上で、サービス設定情報の取得などの操作を実行するときバインドするために使用されます。

puser および dsameuser は関連付けられたパスワードを所有し、それぞれのパスワードは serverconfig.xml に暗号化された形式で格納されています。このファイルは次のディレクトリ内にあります。

- Solaris システム: /etc/opt/SUNWam/config
- Linux システム: /etc/opt/sun/identity/config

インストール後に、puser および dsameuser のパスワードを変更することをお勧めします。ただし、amadmin や amldapuser に使用したものと同じパスワードを使用しないでください。puser または dsameuser のパスワードを変更するには、ampassword ユーティリティーを使用します。

- ampassword --admin (または -a) オプションでは、dsameuser のパスワードが変更されます。(このオプションでは、amadmin のパスワードは変更されません。)
- ampassword --proxy (または -p) オプションでは、puser のパスワードが変更されます。

puser と dsameuser のどちらのパスワードを変更するかは、ユーザーの配備によって決まります。

Access Manager が単一のホストサーバー上に配備されている場合は、次の手順を実行します。

1. ampassword ユーティリティーを使用して、Directory Server とローカルの serverconfig.xml ファイル内のそれぞれのパスワードを変更します。
2. Access Manager Web コンテナを再起動します。

Access Manager が複数のホストサーバー上に配備されている場合は、次の手順を実行します。

1. 最初のサーバー上で、ampassword ユーティリティーを使用して、Directory Server とローカルの serverconfig.xml ファイル内のそれぞれのパスワードを変更します。
2. ampassword --encrypt (または -e) オプションを使用して、新しいパスワードを暗号化します。
3. Access Manager の配備されているその他の各サーバー上で、手順 2 で暗号化した新しいパスワードを使用して、serverconfig.xml ファイル内のパスワードを手動で変更します。
4. パスワードを変更した各サーバー上 (最初のサーバーを含む) で、Access Manager Web コンテナを再起動します。

ampassword ユーティリティーについては、『Sun Java System Access Manager 7 2005Q4 管理ガイド』を参照してください。

スキーマの制限

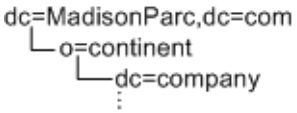
Access Manager は、管理するエントリを抽象的に表現します。したがって、たとえば、Access Manager 内の組織は、Directory Server 内の組織とは必ずしも同じにはなりません。特定の DIT (Directory Infromation Tree) を管理できるかどうかは、ディレクトリエントリを表すまたは管理する方法と、DIT が各 Access Manager タイプの制限に適しているかどうかによります。

以下の項で、Access Manager スキーマに課される制限について説明します。この節の最後には、53 ページの「サポートされない DIT の例」も複数記載しています。

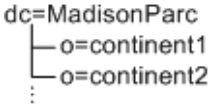
組織としてマークできるエントリのタイプは 1 つに限られる

Access Manager sunISManagedOrganization 予備クラスを、任意のエントリに追加することにより、 Access Manager では、このエントリを組織であるように管理できます。ただし、組織としてマークできるエントリのタイプは 1 つに限られます。たとえば、DIT にエントリ o=sun と別のエントリ dc=ibm がある場合、両方のエントリを組織としてマークすることはできません。

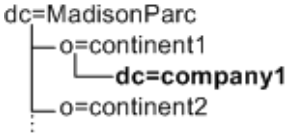
次の例では、dc と o の両方のエントリを組織にしようとしていますが、この DIT 構造は Access Manager で管理できません。



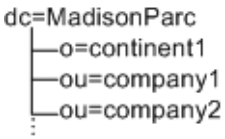
ただし、Access Manager ルートサフィックスでのエントリは、1 つのエントリには数えられません。したがって、次の例のDIT 構造は、Access Manager で管理できます。



o=continent1 の下に dc=company1 を追加した場合、dc がコンテナとしてマークされている場合にのみ、このDIT の管理が可能になります。コンテナは、Access Manager の別の抽象タイプであり、通常、OrganizationalUnit にマッピングされます。ほとんどの DIT では、 iplanet-am-managed-container エントリをすべての OrganizationlUnits に追加します。



ただし、このマーカーオブジェクトクラスはどのエントリタイプにも追加できます。次の例の DIT 構造が可能です。



この例では、o= と ou= の両方のエントリを組織としてマークすることはできないため、o= エントリを organization としてマークし、ou= エントリを containers としてマークしています。コンソールに表示されるときに、組織とコンテナで利用できるオプションは同じです。従属または下位区分、ピープルコンテナ、グループ、ロール、およびユーザーは、両方の内部で作成できます。

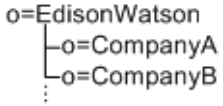
ピープルコンテナをユーザーの親エントリにする必要がある

もう 1 つの抽象エントリタイプはピープルコンテナです。Access Manager タイプは、このエントリがユーザーの親エントリであると想定します。ピープルコンテナとしてエントリに iplanet-am-managed-people-container のマークが付けられていると、UI は、このコンテナには下位ピープルコンテナまたはユーザーだけが存在すると見なします。属性 OrganizationUnit が通常、ピープルコンテナとして使用されますが、iplanet-am-managed-people-container オブジェクトクラスを保有し、Access Manager の管理可能な親タイプの organization または container を保持する限り、Access Manager のどのエントリでもピープルコンテナとして使用できます。

Access Manager XML で可能な組織の説明は 1 つに限られる

Access Manager の組織は、amEntrySpecific.xml で定義されます。このファイルでは、1 つの組織の説明だけを記述できます。この結果、ディレクトリエントリのプロパティーをカスタマイズしたり、管理ページや検索ページを UI 内に作成すると、カスタム属性は Access Manager 設定全体に適用されます。この Access Manager 要件は、特にホスティングサービスを行う企業など、配備における組織ごとに異なる表示属性を必要とする諸企業のニーズに合わない場合があります。

次の例で、Edison-Watson 社はホスティング企業として、インターネットサービスを多数の企業に提供しています。企業 A では、ユーザーの姓、名、バッジ番号を入力するフィールドを表示する必要があります。企業 B では、ユーザーの姓、名、社員番号を入力するフィールドを表示する必要があります。

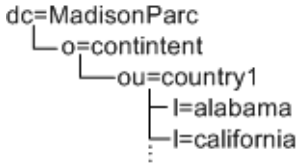


組織の説明は、組織レベルではなくルートレベル (o=EdisonWatson) で定義されます。デフォルトでは、企業 A と 企業 B の両方の UI を同一にする必要があります。また、すべてのサービスは、下位スキーマタイプユーザーの属性になるように、属性をグローバルに定義します。したがって、企業 A が、予備クラス CompanyA-user にそのユーザー用の属性を保持し、企業 B が、CompanyB-user に属性を保持している場合、企業 B の属性は上書きされ、表示されません。

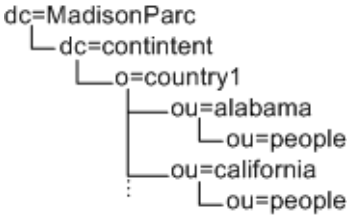
回避策としては、ユーザー表示に対処するように ACI を修正する方法があります。ただしこの回避策は、「検索」および「作成」ウィンドウでの属性には対処しません。

サポートされない DIT の例

次の例では、次の 3 タイプの組織マーカーが必要になります。o、ou、および l。l=california および l=alabama がピープルコンテナではないと見なされるため、この DIT は Access Manager では動作しません。



次の例では、3 タイプの Access Manager マーカー (dc、o、ou)、およびピープルコンテナ (ou=people) が必要になります。この条件下では、DIT は Access Manager で動作しません。


dc=MadisonParc
dc=contintent
o=country1
ou=alabama
ou=people
ou=california
ou=people

第 4 章

Access Manager を使用する場合の論理設計

ソリューションライフサイクルの論理設計段階では、ソリューションにおける論理コンポーネントの相互関係を示す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と技術要件段階で作成した使用状況分析とによって配備シナリオが作成され、このシナリオが配備設計段階でのインプットとなります。この章は、Sun Java ™ System Access Manager の論理設計に関する次の節で構成されています。

論理アーキテクチャーについて

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ソフトウェアコンポーネントが明確にされ、それらコンポーネントの相互関係が示されます。論理アーキテクチャーと技術要件段階で決定した QoS 要件とによって配備シナリオが作成されます。配備シナリオは、次の配備設計段階で行う配備アーキテクチャー設計のための基礎となります。

論理アーキテクチャーの設計

論理アーキテクチャーを設計する際には、技術要件段階で確認したユースケースを使用して、ソリューションに必要なサービスを提供する Java Enterprise System (Java ES) コンポーネントを決定します。最初に指定したコンポーネントに対してサービスを提供するコンポーネントについても、すべて列挙する必要があります。

Java ES コンポーネントは、提供するサービスに応じて、多層アーキテクチャーのコンテキスト内に配置します。コンポーネントが多層アーキテクチャーの一部であることを理解すると、コンポーネントによって提供されるサービスを分配する方法を決めたり、スケーラビリティーや可用性などの Quality of Service (QoS) を実現する方針を決めたりするのに役立ちます。

論理アーキテクチャーおよびソリューションライフサイクルの詳細は、『Sun Java Enterprise System 2005Q4 Deployment Planning Guide』を参照してください。

Access Manager コンポーネント

Access Manager の配備には、次の製品およびコンポーネントが含まれます。

- 56 ページの「Web コンテナ」
- 56 ページの「Directory Server」
- 57 ページの「セッションフェイルオーバー用の Message Queue および Berkeley DB」

Web コンテナ

Access Manager は、次のいずれかの Web コンテナ内で実行する必要があります。

- Sun Java System Web Server
- Sun Java System Application Server
- BEA WebLogic Server
- IBM WebSphere Application Server

サポートされる各 Web コンテナの特定のバージョンについては、『Sun Java System Access Manager 7 2005Q4 リリースノート』を参照してください。

Directory Server

Access Manager には、次のエンティティー用に LDAP ディレクトリサーバーが必要です。

- 56 ページの「Access Manager 情報ツリー」
- 57 ページの「アイデンティティーリポジトリ」

Access Manager 情報ツリー

Access Manager 7 2005Q4 では、Access Manager 情報ツリーを格納するために Sun Java System Directory Server が必要です。Access Manager が作成および維持する Access Manager 情報ツリーには、システムアクセスに関係する次の情報が保持されます。

- ユーザー認証の方法
- ユーザーがアクセス可能なリソース
- ユーザーがリソースへのアクセス権を取得したあとに、アプリケーションで使用可能になる情報

アイデンティティーリポジトリ

Access Manager には、ユーザーやグループなどのユーザーデータを格納するためのアイデンティティーリポジトリが必要です。以前のバージョンの Access Manager では、アイデンティティーリポジトリとして Sun Java System Directory Server が必要でした。しかし、Access Manager 7 2005Q4 では、Sun Java System Directory Server に加えて、LDAP バージョン 3 (LDAP v3) 互換のディレクトリサーバーもサポートされています。

セッションフェイルオーバー用の Message Queue および Berkeley DB

セッションフェイルオーバーの実装を計画している場合は、Access Manager に次のコンポーネントが必要です。

- Sun Java System Message Queue。Message Queue ブローカクラスタは、Access Manager インスタンスとセッションストアデータベースの間のセッションメッセージを管理します。
- Sleepycat Software, Inc. の Berkeley DB (`http://www.sleepycat.com/`)。デフォルトのセッションストアデータベースです。

Access Manager を使用する Java ES コンポーネント

通常、Access Manager は次の Java ES コンポーネント製品とともに配備されます。

- Sun Java System Portal Server、Sun Java System Messaging Server、Sun Java System Calendar Server、Sun Java System Instant Messaging、および Sun Java System Communications Express。シングルサインオン (SSO) を有効にするために必要です。
- Sun Java System Web Server。アクセス制御を任意に設定するために必要です。

Access Manager 論理アーキテクチャーの例

ここでは、Access Manager ソリューションの論理アーキテクチャーの例として、次のシナリオを取り上げます。

Access Manager の Web 配備

Access Manager 配備では一般に Web ブラウザを使用して、Web サーバー上に配備されたアプリケーションまたはリソースにアクセスします。アプリケーションまたはリソースは Access Manager により保護されるため、通信は、Web サーバーにインストールされたポリシーエージェントを使用して行われます。さらに、Web サーバーに Access Manager SDK を配備することもできます。このシナリオの場合、マシン上の Web サーバーの数や、複数のマシンに配備される Access Manager のインスタンスに関して制限はありません。たとえば、1 台のマシンで複数の Web サーバーを稼働させ、それぞれに Access Manager のインスタンスを配備することもできます。同様に、複数の Web サーバーを別々のマシン上で稼働させ、それぞれに Access Manager のインスタンスを配備することもできます。

次の図に、Access Manager の Web 配備シナリオを示します。

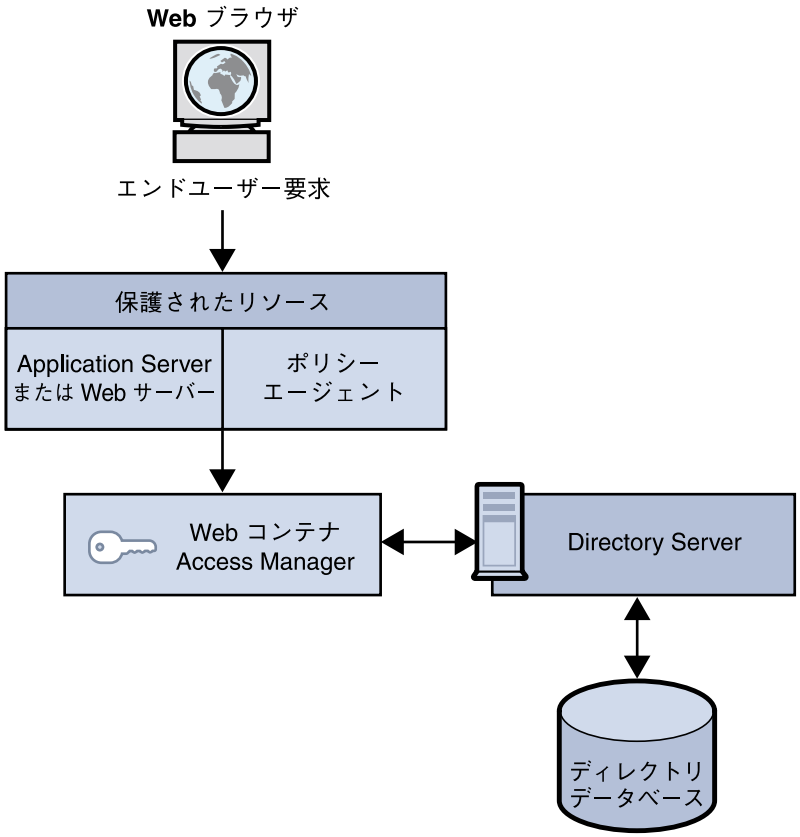


図 **4–1** Access Manager の Web 配備

Access Manager の複数サーバー配備

Access Manager を複数のサーバーに配備する場合、2 台以上のホストサーバーに、それぞれ 1 つ以上の Access manager インスタンスを配備します。それぞれの Access Manager インスタンスは同じ Directory Server にアクセスします。配備環境に応じて、Directory Server インスタンスをマルチマスターレプリケーション (MMR) 設定にすることができます。

1 番目のホストサーバーにインストールされた Access Manager インスタンスは、Directory Server のインスタンスを指し示します。Java ES インストーラを使用したインストールでは、実際の配備に応じて既存の Directory Server が、既存のディレクトリ情報ツリー (DIT) を使用するか、使用しないかを選択できます。

Java ES インストーラを実行して、その他のサーバーに Access Manager のそれ以外のインスタンスをインストールし、既存の DIT を使用して、Access Manager インスタンスが Directory Server を指し示すようにします。このとき Access Manager は、Directory Server がすでに存在しているものと認識しているため、情報を一切 Directory Server に書き込みません。

次の図には、1 つの Directory Server に対して、異なるホストサーバー上に配備された複数の Access Manager インスタンスを示しています。

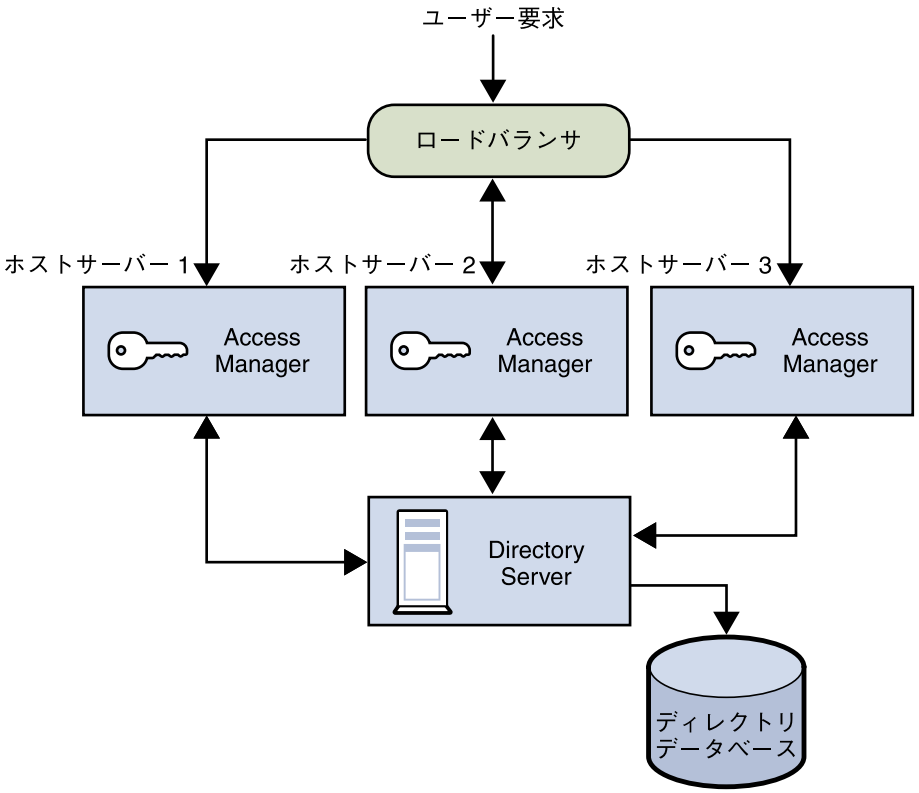


図 **4–2** 1 つの Directory Server に対する複数の Access Manager インスタンス

詳細については、77 ページの「複数のホストサーバーへの Access Manager のインストール」を参照してください。

Java アプリケーションの配備

別の一般的な Access Manager のシナリオでは、Java アプリケーションは配備先のサーバーに直接インストールされた Access Manager SDK にアクセスできます。このシナリオでは、1 つ以上の Access Manager インスタンスが稼働する Sun Java System

Web Server または Sun Java System Application Server など Web コンテナのインスタンスを持つ追加サーバーが必要になります。 このサーバーは、情報を管理してシングルサインオン (SSO) を提供します。次の図は、Java アプリケーションの配備シナリオを示しています。

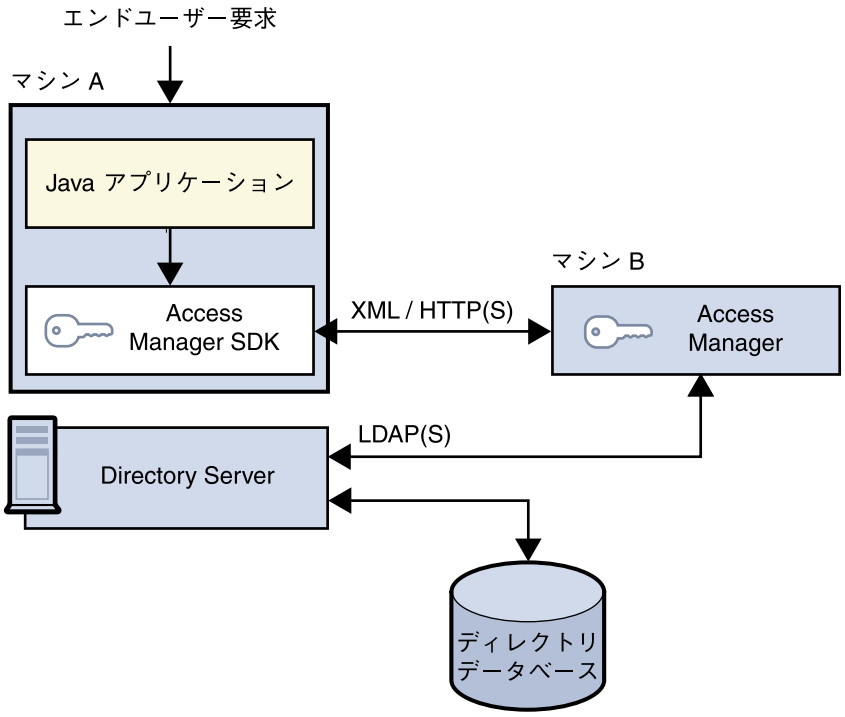


図 **4–3** Java アプリケーションの配備

Access Manager セッションフェイルオーバー配備

Access Manager は、Web コンテナから独立したセッションフェイルオーバーの実装を提供しています。その際、Sun Java System Message Queue (Message Queue) を通信ブローカとして使用し、Sleepycat Software, Inc. の Berkeley DB をセッションストアデータベースとして使用します。Access Manager セッションフェイルオーバーは、単一のハードウェアまたはソフトウェアの障害発生時に、ユーザーの認証セッション状態を維持するため、セッション情報を失ったり、ユーザーに再ログインを要求したりせずに、ユーザーのセッションをセカンダリ Access Manager インスタンスにフェイルオーバーできます。

Access Manager セッションフェイルオーバーの概要

Access Manager 7 2005Q4 セッションフェイルオーバーには、次のコンポーネントが含まれます。

- Access Manager 7 2005Q4 の複数のインスタンス。各インスタンスは 2 つ以上のホストサーバー上で、サポートされる Web コンテナで実行されます。
- Message Queue ブローカクラスタ。Access Manager インスタンスとセッションストアデータベースとの間のセッションメッセージを管理します。
- セッションストアデータベースとして Sleepycat Software, Inc. (`http://www.sleepycat.com/`) によって提供されている Berkeley DB。Berkeley DB クライアントデーモンは `amsessiondb` です。

Access Manager のセッションフェイルオーバーは、次の Message Queue のパブリッシュ/サブスクライブ (トピック送信先) 配信モデルに従います。

1. ユーザーがセッションの開始、更新、または終了を行うと、Access Manager はセッションの作成、更新、または削除に関するメッセージを Message Queue ブローカクラスタにパブリッシュします。
2. Berkeley DB クライアント (`amsessiondb`) は Message Queue ブローカクラスタにサブスクライブし、セッションメッセージを読み取って、データベースにセッション操作を格納します。

Access Manager インスタンスが、単一のハードウェアまたはソフトウェアの問題によって失敗した場合、そのインスタンスに関連付けられているユーザーのセッションが、次のように、セカンダリ Access Manager インスタンスにフェイルオーバーします。

1. セカンダリ Access Manager インスタンスは、Message Queue ブローカクラスタに、ユーザーのセッション情報に対するクエリ要求をパブリッシュします。
2. Message Queue ブローカクラスタの同じセッション要求トピックにサブスクライブしている Berkeley DB クライアント (`amsessiondb`) は、クエリー要求を受信し、セッションデータベースから対応するエントリを取得して、ユーザーのセッション情報をセッション応答トピックとともに Message Queue ブローカクラスタにパブリッシュします。
3. セッション応答トピックにサブスクライブしているセカンダリ Access Manager インスタンスは、ユーザーのセッションを伴う応答を受信し、セッション情報を失ったり、ユーザーが再度ログオンしたりすることなく続行できます。

Message Queue ブローカが失敗すると、Access Manager は非セッションフェイルオーバーモードで動作します。あとで Message Queue ブローカが再起動すると、Access Manager はセッションフェイルオーバーモードに戻ります。

Message Queue コンポーネントおよびパブリッシュ/ サブスクライブ配信モデルについては、『 Sun Java System Message Queue 技術の概要』を参照してください。

セッションフェイルオーバー配備シナリオ

次の図に、2 台のホストサーバーから構成され、それぞれ Access Manager インスタンス (Web コンテナ上に配備)、Message Queue ブローカークラスタ、および Berkeley DB クライアント (`amsessiondb`) を実行している基本的なシナリオを示します。ロードバランサはクライアント要求を Access Manager インスタンスに分配しています。両方の Access Manager インスタンスは同じ Directory Server にアクセスします。これは図に示されていません。

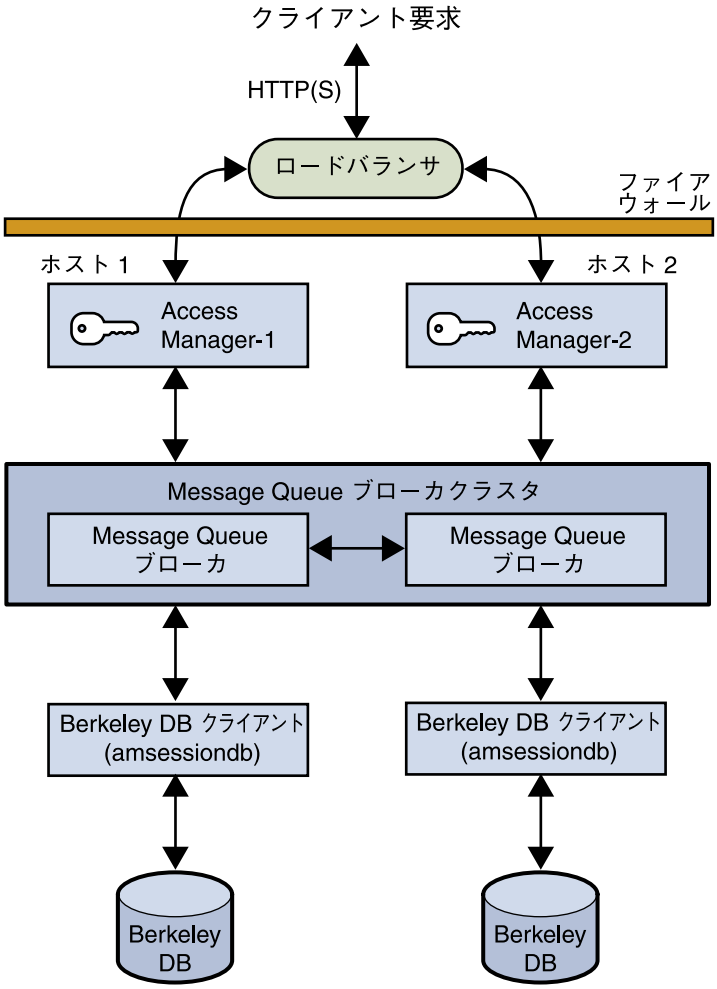


図 **4–4** Access Manager セッションフェイルオーバー基本配備シナリオ

図に示すような、それぞれ同じ Directory Server にアクセスするサイトを追加できます。ただし、セッションフェイルオーバーは、サイト内の Access Manager に対してのみ実行され、現在のリリースでは、サイト間のセッションフェイルオーバーはサポートされていません。

詳細については、89 ページの「Access Manager セッションフェイルオーバーの実装」を参照してください。

Access Manager および Portal Server の配備

Java Enterprise System 2005Q4 リリースでは、Access Manager とPortal Server を同じ物理サーバーにインストールするか、複数のサーバーにインストールできます。

単一のサーバーへのインストール

このシナリオでは、Access Manager と Portal Server を同じ物理サーバーにインストールします。同じサーバーまたはリモートサーバーに、Directory Server をインストールするか、すでにインストールされている Directory Server にアクセスする必要があります。

これらのコンポーネントをインストールするには、Java Enterprise System インストーラを単独のセッションで実行し、次のように選択します。

- 「コンポーネントの選択」パネルで、次の製品とサブコンポーネントを選択します。
  - 「Communication & Collaboration Services」で、Portal Server を選択します。
  - 「Directory & Identity Services」で、Access Manager 7 2005Q4 とそのサブコンポーネントを選択します。
    - アイデンティティー管理およびポリシーサービスコア
    - Access Manager 管理コンソール
    - 連携管理の共有ドメインサービス
    - Access Manager SDK

      デフォルトでは、Portal Server を選択すると、インストーラは Access Manager SDK だけを選択します。したがって、その他のサブコンポーネントには別にチェックを付ける必要があります。

    次の Web コンテナの 1 つをインストールし、設定します。
  - Sun Java System Application Server
  - Sun Java System Web Server

複数のサーバーへのインストール

このシナリオでは、Portal Server は、リモートサーバーからローカルサーバー上の Access Manager にアクセスします。ローカルサーバーまたはリモートサーバーに、Directory Server をインストールするか、すでにインストールされている Directory Server にアクセスする必要があります。

- ローカルサーバーで、Access Manager と Web コンテナをインストールします。リモートサーバーでコンポーネントのインストールと設定を行う前に、このサーバーでコンポーネントのインストールと設定を行う必要があります。
- リモートサーバーで Portal Server と Access Manager SDK をインストールします。リモートサーバーでその他の Access Manager サブコンポーネントを選択する必要はありません。

Access Manager および Portal Server の配備については、『Sun Java System Portal Server 配備計画ガイド』を参照してください。

連携管理

2001 年、Sun Microsystems はほかの企業とともに Liberty Alliance Project に加わりました。このプロジェクトでは、アイデンティティーベースのインフラストラクチャー、ソフトウェア、および Web サービスを開発するための標準を定義しました。

まず Access Manager が実装したのは、アカウント連携、認証ドメイン、およびシングルサインオン (SSO) を含む Identity Federation Framework (Liberty ID-FF) 仕様です。それ以降のリリースの Access Manager には、Liberty ID-FF 仕様のバージョン 1.2 および Liberty Identity Web Services Framework (Liberty ID-WSF) のバージョン 1.0 仕様で定義されている新機能が追加されました。Web サービスには、アイデンティティーベースのサービスプロバイダに格納された種々の属性からなるアイデンティティーデータをインターネットを介して取得および更新するためのフレームワークが含まれます。アイデンティティープロバイダとサービスプロバイダ間の通信に必要な、クライアントアプリケーションプログラミングインタフェース (API) も提供されています。

Access Manager 7 2005Q4 では、さらに機能が追加されています。たとえば、Access Manager は、ビジネスパートナにアウトソーシングされたアプリケーションに対するユーザーアカウントを一括連携することや、アイデンティティープロバイダとサービスプロバイダ間で設定済みロールをマッピングすることを可能にします。

詳細は、『Sun Java System Access Manager 7 2005Q4 Federation and SAML Administration Guide』を参照してください。このマニュアルでは、上記の機能の開発に使用されたオープンな標準仕様を紹介し、それらがどのように Access Manager に実装されているかを説明しています。さらに、統合 Web サービスに関する情報およびアプリケーションプログラミングインタフェース (API) の要約も記載されています。

第 5 章

Access Manager での配備設計

ソリューションライフサイクルの配備設計段階では、ハイレベルの配備アーキテクチャーとローレベルの実装仕様を設計し、ソリューションを実装するために必要な一連の計画および仕様を準備します。この配備設計段階でプロジェクトの承認が行われます。この章は、Sun Java™ System Access Manager での配備設計に関する次の節で構成されています。

ロードバランサの使用法

ほとんどの配備では、ユーザー要求を 2 つ以上の Access Manager インスタンスに分散するために、Access Manager はロードバランサを使用して設定されます。ロードバランサは、ハードウェア、ソフトウェア、またはその両方の組み合わせを使用して実装できます。次の図は、ロードバランサを使用した Access Manager 配備を示しています。

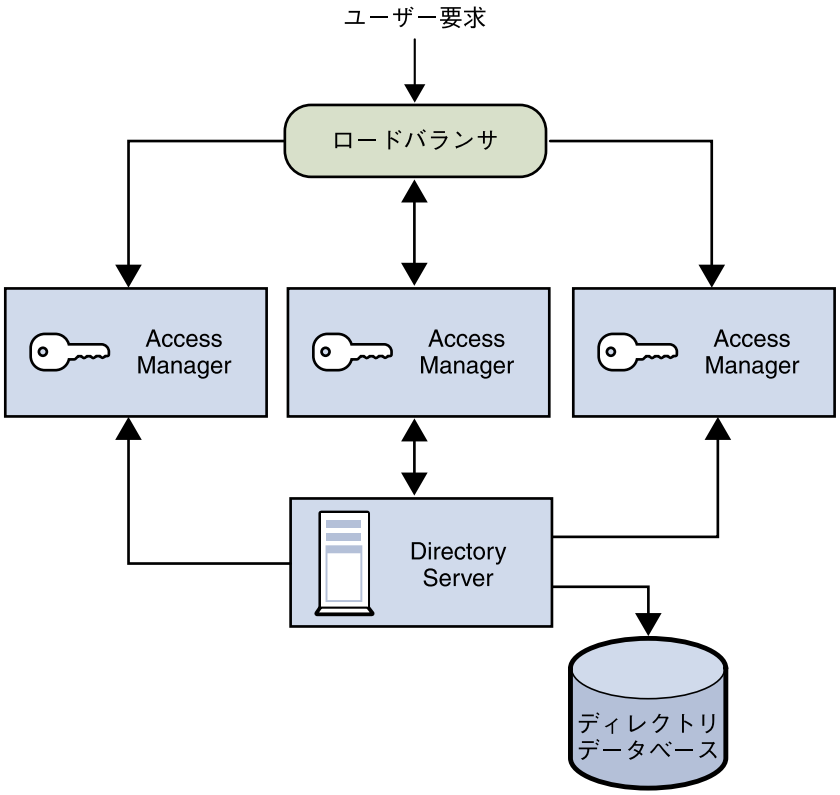


図 **5–1** ロードバランサを使用した Access Manager の設定

スティッキーセッション用のロードバランサの設定

Access Manager とともに配備するロードバランサは、スティッキーセッションをサポートしている必要があります。スティッキーセッションでは、セッションが特定の Access Manager インスタンスによって作成されると、セッション情報を保持するために、ユーザーからのそれ以降の要求は引き続きその同じインスタンスに配信されます。Access Manager は Cookie を使用してセッション情報を中継するため、ロードバランサは、そのセッションを作成した Access Manager インスタンスに要求をリダイレクトする必要があります。スティッキーセッションが存在しない場合は、すべての Access Manager インスタンスを信頼する必要があるため、パフォーマンスが低下します。スティッキーセッションは、`setcookie` 関数またはロードバランサ Cookie のどちらかを使用して実装できます。

詳細は、83 ページの「Access Manager でのロードバランサの使用法」を参照してください。

複数の JVM 環境

Access Manager サービスは、複数の Java 仮想マシン (JVM) 環境でサポートされています。これは、Sun Java System Application Server のインスタンスは複数の JVM を保持するように設定でき、そのすべてで Access Manager サービスが稼働可能であることを意味します。Access Manager のアーキテクチャーでは、マシン内の Sun Java System Application Server インスタンスの数、複数マシンをまたがる Access Manager サービスの数、単一の Application Server が保持できる JVM の数などに関する配備に制限を課しません。

複数の JVM 環境については、次の Sun Java System Application Server のマニュアルを参照してください。`http://docs.sun.com/coll/1310.1`

Directory Server レプリケーションに関する考慮事項

Access Manager のパフォーマンスと応答時間を改善するには、レプリケートされた Directory Server 間でロードバランスを使用する方法と、レプリケートされたサーバーをユーザーの近くに配置する方法の2 つの方法があります。Directory Server は、シングルサプライヤ構成またはマルチサプライヤ構成でセットアップできます。Sun Java System Directory Proxy Server などのロードバランスアプリケーションも使用できます。Directory Proxy Server は、設定された Directory Server セット間の LDAP 操作のプロポーショナルなロードバランスを動的に実行します。1 つ以上の Directory Server インスタンスが利用できない場合、負荷は残りのサーバー間でバランス良く再配分されます。サーバーが復帰すると、負荷がバランス良くかつ動的に再配分されます。

Access Manager をインストールする前に、Directory Server レプリケーションを設定する必要があります。この設定によって、サプライヤとコンシューマのデータベースが正しく同期されるため、参照や更新が適切に同期されていることを確認する時間が取れるようになります。

Access Manager をレプリケーション目的でインストールした場合、Directory Server の各インスタンスおよび Access Manager の各インスタンスは、以下に対して同じ値を使用して設定する必要があります。

- ディレクトリマネージャー
- ディレクトリマネージャーのパスワード
- Directory Server の管理者 ID
- サーバー管理者のパスワード
- ベースサフィックス

- デフォルトの組織

レプリケーション用の設定

Access Manager は、シングルサプライヤまたはマルチサプライヤのレプリケーションで動作するように設定できます。次の図は、コンシューマが読み取り専用のデータベースであるシングルサプライヤ構成を示しています。書き込み操作要求の参照は、サプライヤデータベースに対して行われます。この設定により、負荷が複数のディレクトリに分散させられるため、サーバーのパフォーマンスを向上させる手段として利用できます。

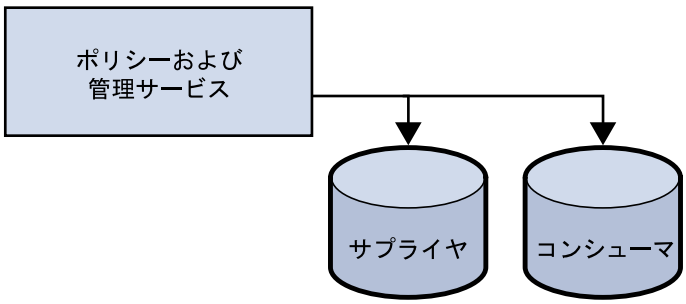


図 **5–2** シングルサプライヤの Directory Server レプリケーション

次の図は、Access Manager の複数インスタンスを使用したマルチサプライヤ構成、またはマルチマスターレプリケーション (MMR) を示しています。この設定によりフェイルオーバー保護および高可用性が実現されるため、サーバーのパフォーマンスはさらに向上します。

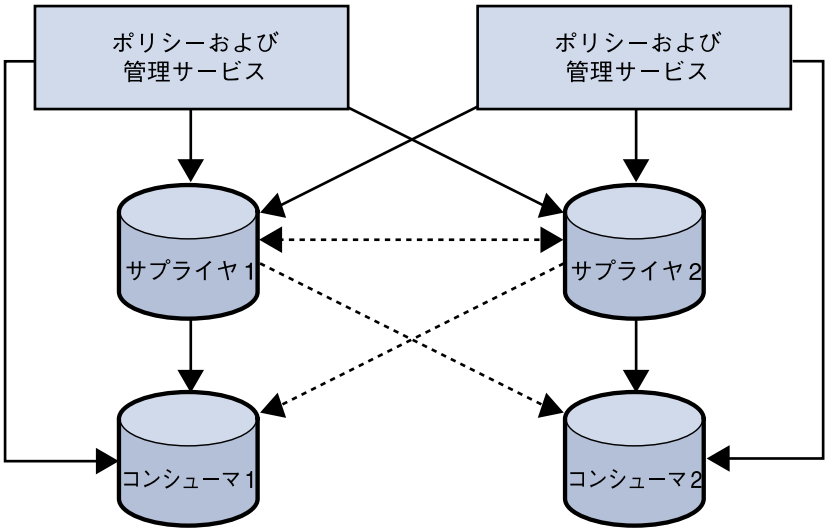


図 **5–3** マルチサプライヤの Directory Server 構成

以下の手順を使用すると、Access Manager がまだインストールされていない場合に、Access Manager ディレクトリツリーのルートまたは最上位レベルでレプリケーションを設定したり、デフォルトの組織レベルでレプリケーションを設定したりすることができます。

1. サプライヤおよびコンシューマ Directory Server インスタンスをインストールします。

   手順については、『Sun Java Enterprise System 2005Q4 Installation Guide for UNIX』を参照してください。

2. サプライヤおよびコンシューマ間のレプリケーションアグリーメントを設定し、ディレクトリ参照および更新が正しく機能することを確認します。

   このバージョンの Access Manager で機能するように、既存の Directory Server データの移行が必要になる場合があります。詳細は、『Sun Java System Access Manager 6 2005Q1 Migration Guide』を参照してください。

3. Access Manager および Directory Server を初めて配備する場合や、既存のユーザーデータを使用する予定がない場合は、Java ES インストールプログラムを実行して Access Manager をインストールしてください。

   インストール時に、既存の Directory Server が存在するかどうか尋ねられたら「はい」を選択し、70 ページの「レプリケーション用の設定」でインストールしたサプライヤ Directory Server のホスト名とポート番号を指定します。

4. Access Manager がインストールされているホストサーバーで、使用しているプラットフォームに応じて、次のディレクトリの `AMConfig.properties` ファイルを修正します。

   - Solaris システム: `/etc/opt/SUNWam/config`
   - Linux システム: `/etc/opt/sun/identity/config`

5. 次のプロパティーを修正して、70 ページの「レプリケーション用の設定」 でインストールしたコンシューマ Directory Server のホストおよびポート番号を反映します。

   - `com.iplanet.am.directory.host`
   - `com.iplanet.am.directory.port`

6. 次のプロパティーを修正して、要求されたエントリが見つからない場合に Access Manager が同じ要求を繰り返す回数を指定します。

   `com.iplanet.am.replica.retries`

7. 次のプロパティーを修正して、Access Manager が再試行を行うまでの時間をミリ秒単位で指定します。

   `com.iplanet.am.replica.delay.between.retries`

8. 有効になっている Access Manager 認証モジュールごとに、Access Manager コンソールを使用して、70 ページの「レプリケーション用の設定」でインストールしたコンシューマディレクトリを指定します。

   - 最初の LDAP サーバーとポートには、プライマリ (コンシューマ) Directory Server のホスト名とポート番号を指定します。たとえば、`consumer1.example.com:389` と指定します。
   - 2 番目の LDAP サーバーとポートには、セカンダリ (サプライヤ) Directory Server のホスト名とポート番号を指定します。たとえば、`supplier1.example.com:389` と指定します。

9. `serverconfig.xml` ファイルで、70 ページの「レプリケーション用の設定」でインストールしたコンシューマディレクトリのホスト名とポート番号を指定します。`serverconfig.xml` ファイルの例を次に示します。
10. Web コンテナを再起動して Access Manager を再起動します。

`serverconfig.xml` ファイルの例

次の例は、`serverconfig.xml` のレプリケーションの変更を示しています。

```
<iPlanetDataAccessLayer>
<ServerGroup name="default" minConnPool="1"
maxConnPool="10">
<Server name="Server1"
host="consumer1.example.com" port="389"
type="SIMPLE" />
```

ロードバランサを使用する場合の設定

次の図は、Directory Proxy Server またはハードウェアロードバランサを含むマルチサプライヤ構成を示しています。この設定により、Access Manager によりサポートされているフェイルオーバー、高可用性、および管理されたロードバランスをうまく活用することができます。

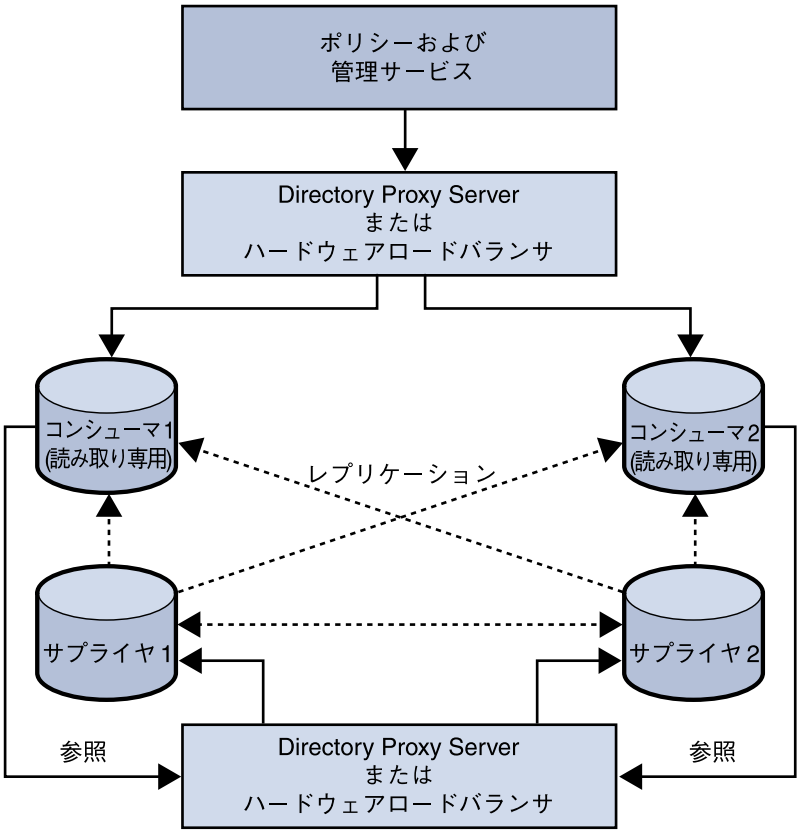


図 **5–4** ロードバランサを使用したマルチサプライヤ構成

LDAP ロードバランサを使用することにより、Access Manager で提供されるレベルを上回る高可用性とディレクトリフェイルオーバー保護の機能が追加されます。たとえば、Directory Proxy Server は、各サーバーに再配分される負荷の割合を指定できます。また、すべてのバックエンド LDAP サーバーが使用不可になった場合は、Directory Proxy Server が引き続き要求を管理し、クライアントのクエリーを拒否します。ロードバランサをインストールする場合は、このアプリケーションを認識するように Access Manager を設定する必要があります。

1. Access Manager を設定する前に、Directory Server をレプリケーション用にセットアップします。ディレクトリレプリケーションおよびセットアップ手順については、Sun Java System Directory Server のマニュアルを参照してください。`http://docs.sun.com/coll/1316.1`
2. LDAP ロードバランサをインストールおよび設定します。使用しているロードバランサに同梱されているマニュアルの指示に従ってください。

3. AMConfig.properties ファイルで、 com.iplanet.am.directory.host および com.iplanet.am.directory.port プロパティーを、コンシューマ Directory Server のロードバランサのホストおよびポート番号を指すように修正します。
4. 有効になっている Access Manager 認証モジュールごとに、Access Manager コンソールを使用して、コンシューマ Directory Server を指定します。次の手順では、例としてLDAP 認証モジュールを使用します。
   - 最初の LDAP サーバーとポートには、プライマリ(コンシューマ) Directory Server のホスト名とポート番号を、proxyhostname:port の形式で入力します。
   - 2 番目の LDAP サーバーとポートには何も入力しないでください。
5. serverconfig.xml ファイルで、コンシューマ Directory Server のホスト名とポート番号を指定します。serverconfig.xml ファイルの例を次に示します。
6. Web コンテナを再起動して Access Manager を再起動します。

serverconfig.xml ファイルに対するロードバランサの変更

次の例は、serverconfig.xml ファイルに対するロードバランサの変更を示しています。

```
<iPlanetDataAccessLayer>
<ServerGroup name="default" minConnPool="1"
maxConnPool="10">
<Server name="Server1"
host="idar.example.com" port="389"
type="SIMPLE"
```

ファイアウォールを使用した Directory Server

Access Manager とDirectory Server の間にファイアウォールが設定されている場合、ファイアウォールのアイドル接続タイムアウト値が、Directory Server のアイドル接続タイムアウト値 (nsslapd-idletimeout 属性) を下回ると、Access Manager 接続がタイムアウトすることがあります。この問題は通常、Access Manager の負荷が低く、使用率がピークに達していないときに発生します。

Directory Server 接続がファイアウォールによって切断されると、Access Manager は、この接続が切断されていることを認識せず、LDAP 接続プールで使用可能なすべての接続を使い果たします。LDAP 接続プールを新規に作成するには、Access Manager を再起動する必要があります。この問題を回避するには、次の解決策を検討してください。

グローバルタイムアウト属性の設定

Directory Server の `nsslapd-idletimeout` グローバル属性を、ファイアウォールのアイドル接続タイムアウト値より小さい値に設定できます。ただし、`nsslapd-idletimeout` は Access Manager 以外のアプリケーションにも影響するグローバル設定属性であるため、この解決策を採用できない場合もあります。

個々のクライアント接続のタイムアウトの設定

Directory Server では、個々のクライアント接続に対して個別の属性を設定できます。`nsIdleTimeout` 属性では、個々のクライアントのアイドル接続タイムアウト値を指定します。この値は、Directory Server のグローバル設定で指定した `nsslapd-idletimeout` 値よりも優先されます。

LDAP ディレクトリにバインドした Access Manager ユーザーについて、`nsIdleTimeout` 属性を設定します。このユーザーは、デフォルトでは `amldapuser` になっています。この属性は、`dsameuser` および `puser` ユーザーにも適用されます。

`amldapuser` に `nsIdleTimeout` 属性を追加するには、Directory Server コンソールまたは `ldapmodify` ツールのどちらかを使用します。次に例を示します。

```
ldapmodify -h host-name -p port
-D "cn=Directory Manager" -w password
dn: cn=amldapuser,ou=DSAME Users, dc=example,dc=com
changetype: modify
add: nsIdleTimeout
nsIdleTimeout: timeout-value
```

timeout-value には、ファイアウォールについて設定したアイドル接続タイムアウト値よりも低い値を指定します。このように設定すると、`amldapuser` ユーザーの Access Manager 接続は、ファイアウォールによって切断される前に、Directory Server によって切断されます。

`dsameuser` または `puser` にタイムアウトを追加する場合にも、上の構文を使用しますが、`dn` オプションを `dsameuser` または `puser` ユーザーに設定します。

`AMConfig.properties` ファイルの `com.sun.am.event.connection.idle.timeout` プロパティーでは、持続検索が再起動するまでのタイムアウト値 (分) を指定します。このプロパティーを指定しておけば、接続が切断されたときに持続検索が確実に再起動します。この値は、ロードバランサまたはファイアウォール TCP のタイムアウト値よりも低い値にしておくことが理想的です。そうすれば、接続が切断する前に持続検索が再起動するようになるからです。デフォルト値はゼロ (0) で、持続検索は再起動されません。

Directory Server 属性と `ldapmodify` ツールについては、次の Sun Java System Directory Server のマニュアルを参照してくださ
い。`http://docs.sun.com/coll/1316.1`

第 6 章

Access Manager 設計の実装

ソリューションライフサイクルの実装段階では、配備のためのさまざまなソリューションを実装します。この章では、Sun Java™ System Access Manager の次の実装シナリオについて説明します。

複数のホストサーバーへの Access Manager のインストール

それぞれが同じ Directory Server にアクセスする Access Manager インスタンスを複数のホストサーバーにインストールするには、次の手順に従います。

Access Manager インスタンスの配備

それぞれが同じ Directory Server にアクセスする Access Manager インスタンスを複数のホストサーバーにインストールするには、次の手順に従います。

1. Java Enterprise System (Java ES) インストーラを実行して、Access Manager をホストサーバーにインストールします。インストーラを実行したら、「今すぐ設定」または「あとで設定」のいずれかのオプションを指定します。インストーラの実行については、『Sun Java Enterprise System 2005Q4 Installation Guide for UNIX』を参照してください。

   インストーラの実行中には、Access Manager Web コンテナとして Web Server または Application Server を インストールすることもできます。Web コンテナとして BEA WebLogic Server または IBM WebSphere Application Server を使用するには、まずそれらの製品をインストールしてから、次の手順で `amconfig` スクリプトを実行する必要があります。インストールの手順については、対応する BEA または IBM 製品のマニュアルを参照してください。

2. インストール時に「あとで設定」オプションを指定した場合や、Access Manager インスタンスを (たとえば、Web コンテナとして BEA WebLogic Server または IBM WebSphere Application Server を使用するために) 再設定する必要がある場合は、`amconfig` スクリプトを実行する必要があります。`amconfig` スクリプトと `amsamplesilent` 設定ファイルは、*AccessManager-base*`/bin` ディレクトリに格納されています。ここで、*AccessManager-base* は、デフォルトのインストールディレクトリを表します。Solaris システムでは `/opt/SUNWam`、Linux システムでは `/opt/sun/identity` です。

   `amconfig` スクリプトを次のように実行します。

   a. `amsamplesilent` ファイルを書き込み可能なディレクトリに コピーし、そのディレクトリを現在のディレクトリにします。たとえば、`/newinstances` というディレクトリを作成します。

   b. 設定する新しいインスタンスを判別できるように `amsamplesilent` ファイルのコピーの名前を変更します。たとえば、Web Server 6.1 用の新しい Access Manager インスタンスを作成する場合は、ファイル名を `amwebsvr6` に変更します。

   c. `amwebsvr6` ファイル内の変数を設定して、新しいインスタンスを設定します。たとえば、次のように Access Manager をレルムモードで設定します。

      ```
      AM_REALM=true
      DEPLOY_LEVEL=1
      NEW_INSTANCE=true
      WEB_CONTAINER=WS6 # Web Server is the web container
      DIRECTORY_MODE=1
      ...
      ```

      あとでこのインスタンスの再設定またはアンインストールが必要になった場合に備えて、新しい `amwebsvr6` ファイルを保存します。

   d. 新しい `amwebsvr6` ファイルをサイレント設定用入力ファイルに指定して、`amconfig` スクリプトを実行します。たとえば、Access Manger がデフォルトディレクトリにインストールされた Solaris システムでは、次のように入力します。

      ```
      # cd /opt/SUNWam/bin/
      # ./amconfig -s ./newinstances/amwebsvr6
      ```

amsamplesilent ファイル、またはこのファイルのコピーへのフルパスを指定して、amconfig を実行します。スクリプトは amwebsvr6 ファイル内の変数を読み取り、サイレントモード (-s オプション) で実行されて、Web コンテナ用に Access Manager を設定します。 amsamplesilent ファイルと amconfig スクリプトの実行については、『Sun Java System Access Manager 7 2005Q4 管理ガイド』を参照してください。

3. これらの手順を、追加の Access Manager インスタンスを配備するほかのホストサー バーで繰り返します。

追加の Access Manager インスタンスを配備する場合の考慮事項には次のものがあります。

- Java ES インストーラを実行しており、1 番目のインスタンスと同じ Directory Server を使用する場合は、「Directory Server にユーザーデータが準備されていますか?」に対して「はい」を選択します。
- amconfig スクリプトを実行している場合は、amsamplesilent ファイルのコピーに変数を設定します。たとえば、次のように Access Manager をレルムモードで配備します。

```
AM_REALM=true
DEPLOY_LEVEL=1
NEW_INSTANCE=true
WEB_CONTAINER=WS6 # Web Server is the web container
DIRECTORY_MODE=4  # Directory Server is provisioned with user data
AM_ENC_PW=password-encryption-key-value-from-the-first=instance
...
```

- デフォルト以外のネーミング属性とオブジェクトクラスを使用している場合、ユーザーのネーミング属性と組織のネーミング属性およびオブジェクトクラスに対して適切なカスタム値を指定します。また、Web アプリケーションのすべての配備 URI (SERVER_DEPLOY_URI、CONSOLE_DEPLOY_URI、PASSWORD_DEPLOY_URI、および COMMON_DEPLOY_URI) が以前のインストールと一致する必要があります。
- 次の注に説明されているように、1 番目のインスタンスと同じパスワード暗号化鍵を使用します。

注意 – 複数サーバーの配備では、すべての Access Manager インスタンスで、パスワード暗号化鍵に同じ値を使用する必要があります。

複数サーバーの配備で、Java ES インストーラを実行して Access Manager を以降の (2 番目、3 番目、それ以降の) サーバーにインストールすると、インストーラは各サーバーに対して新しいランダムなパスワード暗号化鍵を生成します。そのため、以降のサーバーでインストーラを実行する場合は、1 番目の Access Manager インスタンスの暗号化鍵値を使用してください。これは、`AMConfig.properties` ファイル内の `am.encryption.pwd` 属性からコピーできます。次のように設定します。

- 「今すぐ設定」オプション。インストーラによって生成された新しいランダムな暗号化鍵を、1 番目のインスタンスの暗号化鍵値に置き換えます。
- 「あとで設定」オプション。`amconfig` スクリプトを実行する前に、`amsamplesilent` ファイルのコピー内にある `AM_ENC_PWD` 変数を、1 番目のインスタンスの暗号化鍵値に設定します。

ただし、Access Manager インスタンスのパスワード暗号化鍵を変更する必要がある場合は、付録 B を参照してください。

プラットフォームサーバーリストおよびレルムまたは DNS エイリアスへのインスタンスの追加

異なるホストサーバーに Access Manager の複数インスタンスをインストールする場合、追加のインスタンスは、プラットフォームサーバーリスト、あるいはレルムまたは DNS のエイリアスに追加されません。次のように、追加の Access Manager インスタンスの値を明示的に追加する必要があります。

1. 1 番目の Access Manager ホストサーバーで、`amadmin` として Access Manager 7 2005Q4 コンソールにログインします。
2. Access Manager コンソールで、「設定」、「システムプロパティー」、「プラットフォーム」の順にクリックします。
3. 追加の各 Access Manager インスタンスを、「インスタンス名」の下にあるプラットフォームサーバーリストに追加します。
   a. 「インスタンス名」のプラットフォームサーバーリストで、「新規」をクリックします。
   b. 「新規サーバーインスタンス」で、「サーバー」と「インスタンス名」を追加します。次に例を示します。
      - サーバー: `http://amserver2.example.com:80`
      - インスタンス名: `02`
   c. 「了解」をクリックしてインスタンスを追加します。
   d. すべてのインスタンスを追加したら、「保存」をクリックします。
4. 追加の各 Access Manager インスタンスについて、レルムまたは DNS のエイリアスを追加します。

a. Access Manager コンソールで、「アクセス制御」をクリックし、「レルム名」の下にあるルート (最上位レベル) レルムをクリックします。

b. 「レルム属性」で、「レルムまたは DNS のエイリアス」に Access Manager インスタンスを追加し、「追加」をクリックします。たとえば、`amserver2.example.com` と指定します。

c. すべてのインスタンスを追加したら、「保存」をクリックします。

サイトとしての Access Manager 配備の設定

Access Manager 7 2005Q4 では、Access Manager 配備の集中的な設定管理を行えるようにする「サイトの概念」が導入されました。Access Manager がサイトとして設定されている場合は、クライアント要求は常にロードバランサを経由します。それにより、配備が簡略化されるとともに、クライアントとバックエンド Access Manager サーバーの間にあるファイアウォールなどの問題も解決されます。サイトには次のコンポーネントが含まれます。

- 複数の Access Manager インスタンスが少なくとも 2 台の異なるホストサーバーに配備されます。たとえば、1 台のサーバーに2 つのインスタンスを配備し、もう 1 台のサーバーに 3 番目のインスタンスを配備することができます。または、すべてのインスタンスを別々のサーバーに配備することもできます。また、配備に必要な場合は、Access Manager インスタンスをセッションフェイルオーバーモードに設定することもできます。
- 1 つ以上のロードバランサが、クライアント要求をさまざまな Access Manager インスタンスに経路指定します。 配備の要件 (たとえば、ラウンドロビンまたは負荷平均値の使用) に従って各ロードバランサを設定し、Access Manager インスタンス間で負荷を分散します。
- すべての Access Manager インスタンスが、同じ Directory Server にアクセスします。

次の手順は、レルムモードでの Access Manager 7 2005Q4 コンソールを示しています。

サイトの設定

Access Manager の複数サーバー配備が存在する場合は、次のどちらかの方法を使用して、配備をサイトとして設定します。

- Access Manager セッションフェイルオーバーの実装を計画している場合は、`amsfoconfig` スクリプトによって配備がサイトとして設定されます。89 ページの「Access Manager セッションフェイルオーバーの実装」を参照してください。

- セッションフェイルオーバーの実装を計画していない場合は、この節の手順に従ってください。

配備をサイトとして設定する場合は、Access Manager コンソールで次の機能を実行します。

- ロードバランサの URL を「サイト名」(サイト ID) に追加します。
- ロードバランサのサイト名 (サイト ID) を、プラットフォームサーバーリスト内の各 Access Manager インスタンスにマップします。
- ロードバランサを「レルムまたは DNS のエイリアス」に追加します。

また、Access Manager によって `fqdnMap` プロパティー (メモリー内) がロードバランサを含むように自動的に設定されるため、`AMConfig.properties` ファイルでこのプロパティーを明示的に設定する必要はありません。

Access Manager 配備をサイトとして設定するには、次の手順に従います。

1. `amAdmin` として Access Manager にログインします。
2. ロードバランサの URL を「サイト名」に追加します。
   a. Access Manager コンソールで、「設定」、「システムプロパティー」、「プラットフォーム」の順にクリックします。
   b. 「サイト名」の下で、「新規」をクリックし、ロードバランサに関する次の値を入力します。
      - サーバー: ロードバランサのプロトコル、ホスト名、およびポート。次に例を示します。`http://lb.example.com:80`
      - サイト名: 一意の 2 桁のサイト識別子 (サイト ID)。次に例を示します。10

      入力したら、「了解」をクリックします。
   c. ロードバランサを「サイト名」に追加したら、「保存」をクリックします。これにより、ロードバランサのエントリにサイト ID が追加されます。次に例を示します。`http://lb.example.com:80|10`

      サイト ID は、サーバー ID およびその他のサイト ID に関して一意であることが必要です。たとえば、01 をサイト ID とサーバー ID の両方に使うことはできません。
3. 同じコンソールパネルで、ロードバランサを各 Access Manager インスタンスにマップします。
   a. 「インスタンス名」の下にある「サーバー」リストで各インスタンス名をクリックして、そのインスタンスの「サーバーインスタンスの編集」パネルを表示します。
   b. ロードバランサの「サイト名」(サイトID) を Access Manager インスタンスにマップします。たとえば、「サイト名」が 10 のロードバランサを使用した場合、1 番目のサーバーの「インスタンス名」は `01(|10)` になります。
   c. 「了解」をクリックし、ほかの Access Manager インスタンスに対して上記の手順を繰り返します。

      手順を終了すると、すべての Access Manager インスタンスがロードバランサにマップされます。次に例を示します。

```
http://amserver1.example.com:8080|01|10
http://amserver2.example.com:8080|02|10
http://amserver3.example.com:8080|03|10
```

d. 「保存」をクリックして設定を保存します。

4. ロードバランサのレルムまたは DNS のエイリアスを追加します。

   a. Access Manager コンソールで、「アクセス制御」をクリックし、「レルム名」の下にあるルートまたは最上位レベルのレルムをクリックします。

   b. 「レルム属性」で、ロードバランサを「レルムまたは DNS のエイリアス」に追加し、「追加」をクリックします。次に例を示します。 `lb.example.com`

   c. 「保存」をクリックして変更を保存します。

5. 個々の Access Manager インスタンスとは異なり、ポリシーエージェントなどのクライアントでは、ロードバランサが唯一のエントリポイントになります。たとえば、ポリシーエージェントを使用している場合は、`AMAgent.properties` ファイル内の該当するエントリを、ロードバランサを指し示すように変更します。

Access Manager でのロードバランサの使用法

ロードバランサは、複数サーバー配備の環境で、クライアント要求を複数の Access Manager インスタンスに分散します。この節の情報を利用する前に、81 ページの「サイトとしての Access Manager 配備の設定」の説明に従って Access Manager 配備をサイトとして設定してください。1 つのサイトには、異なるホストサーバーにインストールされた、Access Manager の複数 (2 つ以上) のインスタンスが含まれます。すべての Access Manager インスタンスが 同じ Directory Server にアクセスし、同じパスワード暗号化鍵を使用する必要があります。Access Manager のインストールについては、77 ページの「複数のホストサーバーへの Access Manager のインストール」を参照してください。

この節には、ロードバランサの使用法に関する次の情報が含まれています。

ロードバランサ用の SSL ターミネーションの設定

ロードバランサを設定して SSL 要求を処理できるようにする前に、まず、Access Manger Web コンテナの SSL を設定します。手順については、『Sun Java System Access Manager 7 2005Q4 管理ガイド』の第 3 章「Access Manager を SSL モードに設定する」を参照してください。

ロードバランサと Access Manager サーバー用に SSL を設定するには、次の場合について考慮してください。

- ロードバランサのみの SSL 設定: SSL ターミネーション。

  ロードバランサがクライアントからの SSL 接続を終了し、Access Manager サーバーへの別の SSL 接続を作成します。

- Access Manager サーバーのみの SSL 設定: SSL パススルー。

  ロードバランサがクライアントからのすべての要求を Access Manager サーバーにバイパスします。

- ロードバランサと Access Manager サーバー両方の SSL 設定。

SSL パススルー設定を除き、標準のサーバー証明書を使用してロードバランサの SSL ターミネーションを有効にすることができます。ただし、ロードバランサと Access Manager サーバー用に SSL パススルーを設定し、ロードバランサがクライアントから Access Manager サーバーへのすべての要求をバイパスする場合は、標準のサーバー証明書では SSL に関する次の問題が発生します。

- クライアントがロードバランサ経由で Access Manager サーバーにアクセスした場合、クライアントは Access Manager サーバーからサーバー証明書を取得します。ロードバランサは SSL サーバー証明書を持たず、クライアント要求をバックエンド Access Manager サーバーにバイパスするだけです。その場合、クライアントは、サーバー証明書のホスト名と対象名が異なっているという警告メッセージを受け取ります。
- 上記の問題を回避するには、ロードバランサ名の `SubjectDN` を使用してサーバー証明書をインストールします。ただし、2 台の Access Manager サーバー間のセッション検証で問題が発生します。

  たとえば、あるユーザーが `amserver1` からセッションを取得し、同じユーザーの 2 番目の要求が `amserver2` に送信されると、`amserver2` では `amserver1` に対するユーザーセッションを検証する必要があります。`amserver2` がセッション検証要求を `amserver1` に送信する場合は、`amserver1` への SSL 接続を作成し、次に `amserver1` からロードバランサの `SubjectDN` 付きのサーバー証明書を取得します。この 2 つの名前 (`amserver1` のホスト名と証明書の `subjectDN`) が異なるため、`amserver2` は SSL ハンドシェークを停止し、セッション検証は失敗します。

これらの問題を解決するために、Access Manager には次のプロパティーが用意されています。

- `com.iplanet.am.jssproxy.trustAllServerCerts`

このプロパティーが有効 (true) になっていると、Access Manager は証明書に関連した問題 (ファイル名の競合など) をすべて無視し、SSL ハンドシェークを継続します。

注意 – 発生の可能性があるセキュリティー上のリスクを回避するために、このプロパティーを有効にするのは、テストを目的とする場合、または企業ネットワークが厳重に管理されている場合だけにしてください。セキュリティー上のリスクが発生する可能性がある場合 (たとえば、サーバーが別のネットワークのサーバーに接続されている場合) は、このプロパティーを有効にしないでください。

- `com.iplanet.am.jssproxy.SSLTrustHostList`

  このプロパティーが有効 (true) になっていると、Access Manager は `AMConfig.properties` ファイルのプラットフォームサーバーリストをチェックします。サーバーリスト内の2 つのサーバーのサーバー FQDN が一致すると、Access Manager は SSL ハンドシェークを継続します。

- `com.iplanet.am.jssproxy.checkSubjectAltName`

  このプロパティーが有効 (true) になっており、かつサーバー証明書に対象別名 (`SubjectAltName`) 拡張機能が含まれていると、Access Manager は、この拡張機能内のすべての名前エントリをチェックします。サーバー FQDN と同じ名前が `SubjectAltName` 拡張機能内に 1 つもない場合、Access Manager は SSL ハンドシェークを継続します。このプロパティーの使用は、`com.iplanet.am.jssproxy.trustAllServerCerts` プロパティーを有効にする場合より安全です。公開鍵基盤 (PKIX) の定義では、`SubjectAltName` 拡張機能を使用して証明書に複数の対象名を含めることができます。

  このプロパティーを有効にするには、信頼される FQDN をカンマで区切ったリストで指定します。次に例を示します。

  ```
  com.iplanet.am.jssproxy.checkSubjectAltName=
  amserv1.example.com,amserv2.example.com
  ```

  `SubjectAltName` 拡張機能を使用した証明書の取得については、次の節を参照してください。

SubjectAltName 拡張機能を使用した CSR の生成

`SubjectAltName` 拡張機能を使用して証明書署名要求 (CSR) を生成するには、証明書データベースツール (`certutil`) を使用します。`certutil` が `/usr/sfw/bin` ディレクトリにない場合、まず、Solaris システムの場合は `SUNWtlsu` パッケージを、Linux システムの場合は `sun-nss-sun-nss-devel` RPM をインストールします。必要に応じて、 `LD_LIBRARY_PATH` 環境変数に、適切な `certutil` のパスを設定します。

`certutil` については、次のサイトを参照してください。`http://www.mozilla.org/`

ここでは、Web コンテナとして Web Server または Application Server を使用している場合の certutil の使用方法について説明します。Web コンテナとして BEA WebLogic Server または IBM WebSphere Application Server を使用している場合は、それぞれの BEA または IBM 製品のマニュアルを参照してください。

SubjectAltName 拡張機能を使用して CSR を生成するには、次の手順に従います。

1. スーパーユーザー (root) としてログインするか、スーパーユーザーになります。
2. certutil -N オプションを使用して、新しい証明書データベース (cert8.db) を作成します。必要に応じて、最初に、データベース用のディレクトリを作成します。次に例を示します。

   ```
   # mkdir certdbdir
   # cd certdbdir
   # certutil -N -d .
   ```

   certutil から入力を求められたら、鍵を暗号化するためのパスワードを入力します。

   ```
   Enter a password which will be used to encrypt your keys.
   The password should be at least 8 characters long,
   and should contain at least one non-alphabetic character.

   Enter new password: your-password
   Re-enter password:  your-password
   ```

3. SubjectAltName 拡張機能を使用して CSR を生成します。次に例を示します。

   ```
   # certutil -R -s "cn=lb.example.com,o=example.com,c=us"
   -o server.req -d . -a -8 amserv1.example.com,amserv2.example.com
   ```

   certutil から入力を求められたら、パスワード (または pin) を入力し、次にランダムシードを生成するための鍵を入力して鍵を作成します。

   ```
   Enter Password or Pin for "NSS Certificate DB": your-password

   A random seed must be generated that will be used in the
   creation of your key.  One of the easiest ways to create a
   random seed is to use the timing of keystrokes on a keyboard.

   To begin, type keys on the keyboard until this progress meter
   is full.  DO NOT USE THE AUTOREPEAT FUNCTION ON YOUR KEYBOARD!

   Continue typing until the progress meter is full:

   |************************************************************|

   Finished.  Press enter to continue:

   Generating key.  This may take a few moments...
   ```

4. CSR (この例では server.req ファイル) を認証局 (CA) に送信します。サーバー証明書を取得し、certutil -A オプションを使用して、それを証明書データベースに追加します。
5. 証明書データベース (cert8.db) を Web コンテナのディレクトリにコピーします。

- Web Server。cert8.db および key3.db データベースを /opt/SUNWwbsrv/alias ディレクトリにコピーし、Web Server インスタンスの名前を使用してその名前を変更します。次に例を示します。

  ```
  https-webserver.example.com-webserver-cert8.db
  https-webserver.example.com-webserver-key3.db
  ```

- Application Server。cert8.db および key3.db データベースをインスタンスの /config ディレクトリにコピーします。次に例を示します。

  ```
  /var/opt/SUNWappserver/domains/domain1/config/cert8.db
  /var/opt/SUNWappserver/domains/domain1/config/key3.db
  ```

ロードバランサ Cookie 用の Access Manager の設定

Access Manager をロードバランサ Cookie 用に設定するには、配備内のすべての Access Manager インスタンスの設定を更新して、インスタンスがロードバランサを認識できるようにします。このシナリオでは、異なるホストサーバーに複数 (2 つ以上) の Access Manager インスタンスが配備されています。ロードバランサが、クライアント要求をさまざまな Access Manager インスタンスに経路指定します。すべての Access Manager インスタンスが、同じ Directory Server を使用します。

1. Access Manager コンソールで、81 ページの「サイトとしての Access Manager 配備の設定」の説明に従って Access Manager 配備をサイトとして設定します。配備をサイトとして設定すると、Access Manager はロードバランサを含むようにメモリー内の fqdnMap プロパティーを自動的に設定します。
2. 各 Access Manager の AMConfig.properties ファイルに次のプロパティーを追加します。

   ```
   com.iplanet.am.lbcookie.name=amlbcookie
   com.iplanet.am.lbcookie.value=amserver
   ```

   amlbcookie はロードバランサの Cookie で、*amserver* はそのインスタンスで使われる Access Manager ホストサーバーの名前です。
3. 各 Web コンテナを再起動することによって、すべての Access Manager インスタンスを再起動します。

SAML を使用したロードバランサの設定

このシナリオでは、Access Manager サイトがロードバランサを使用してクライアント要求をさまざまな Access Manager インスタンスに分散しており、サイトには SAML (Security Assertions Markup Language) サービスが実装されています。要求がロードバランサ経由で Access Manager インスタンスに送信された場合、そのインスタンスが SAML 表明を取得するには、配備内のほかのどの Access Manager サーバーによって元の表明またはアーティファクトが発行されたかを認識する必要があります。

まず、配備がサイトとして設定されていることが必要です。ホストサーバーには複数の Access Manager インスタンスがインストールされており、ロードバランサがクライアント要求をさまざまなインスタンスに経路指定します。すべての Access Manager インスタンスが、同じ Directory Server にアクセスします。Access Manager セッションフェイルオーバーはオプションです。

SAML でロードバランサを使用するようにサイトを設定するには、次の手順に従います。

1. SAML ロードバランスが機能するには、Access Manager 配備がサイトとして設定されていることが必要です。Access Manager 配備をサイトとして設定していない場合は、81 ページの「サイトとしての Access Manager 配備の設定」の手順に従います。
2. `amadmin` として Access Manager コンソールにログインします。
3. Access Manager コンソールで、「連携」、次に「`SAML`」をクリックします。
4. 「プロパティー」セクションの「`SAML` プロファイル」で、次のエントリを追加または変更します。
   - サイト ID。配備内の各 Access Manager インスタンスを追加します。すべての Access Manager インスタンスが、同じ「サイト ID とサイト発行者名」を共有する必要があります。
   - 信頼パートナー。パートナーの配備サイトの「ソース ID」(サイト ID)、「発行者名」、および「ホストリスト」を追加します。Access Manager サーバーの一意の「ソース ID」(サイト ID) と「発行者名」、およびロードバランサの URL、IP アドレス、またはホスト名によって配備が識別され、これらの情報が設定のためにパートナーのサイトに送信されます。

     これらのフィールドの詳細については、『Sun Java System Access Manager 7 2005Q4 Federation and SAML Administration Guide』を参照してください。
5. 「保存」をクリックして変更を保存します。

`fqdnMap` プロパティーの設定

Access Manager 配備をサイトとして設定した場合は、Access Manager によってメモリー内の `fqdnMap` プロパティーがロードバランサを含むように自動的に設定されるため、`AMConfig.properties` ファイルでこのプロパティーを設定する必要はありません。ただし、次の Access Manager 配備の場合は、このプロパティーを明示的に設定 する必要があります。

- 配備がサイトとして設定されていない場合。
- 配備に、物理ホストにマップされた仮想ホストが含まれている場合。

`fqdnMap` プロパティーを設定する必要がある場合は、配備内の各 Access Manager インスタンスの `AMConfig.properties` ファイルで、このプロパティーをロードバランサに設定します。必要に応じて、最初に、このプロパティーのコメント文字 (#) を削除します。 次に例を示します。

```
com.sun.identity.server.fqdnMap[lb.example.com]=lb.example.com
```

ロードバランサを経由した Access Manager インスタンスへのアクセス

ロードバランサ経由で Access Manager インスタンスにアクセスする方法は、モード (レルムまたは旧バージョン) と、アクセスするコンソールによって異なります。ロードバランサ経由で Access Manager インスタンスにアクセスするには、次の構文を使用します。

`http://`*loadbalancer*`.`*domain*`:`*port*`/amserver/console|/amconsole`

旧バージョンモードでは、次の両方のコンソールにアクセスできます。

- 新しい Access Manager 7 2005Q4 コンソール。次に例を示します。

  `http://loadbalancer.example.com:80/amserver/console`
- Access Manager 6 2005Q1 コンソール。次に例を示します。

  `http://loadbalancer.example.com:80/amconsole`

レルムモードでは、新しい Access Manager 7 2005Q4 コンソールにのみアクセスできます。次に例を示します。

```
http://loadbalancer.example.com:80/amserver/console
```

Access Manager セッションフェイルオーバーの実装

Access Manager では、Sun Java System Message Queue (Message Queue) を通信ブローカとして、また Sleepycat Software, Inc. の BerkeleyDB をセッションストアデータベースとして使用して、Web コンテナから独立したセッションフェイルオーバー実装を提供しています。Access Manager 7 2005Q4 の拡張機能には、セッションフェイルオーバー環境を設定するための `amsfoconfig` スクリプトと、Message Queue ブローカや Berkeley DB クライアントを起動および停止するための `amsfo` スクリプトが含まれています。

ここでは、次のトピックについて説明します。

■ 107 ページの「amsessiondb クライアントを使用したパフォーマンステスト」

Access Manager セッションフェイルオーバーシナリオ

次の図に、次のコンポーネントを含む Access Manager セッションファイルオーバーの配備シナリオを示します。

- 別々のホストサーバーで、サポートされる Web コンテナ上で実行される 3 つの Access Manager インスタンス。すべての Access Manager インスタンスは同じ Directory Server にアクセスします。これは図に示されていません。
- 別々のサーバーでクラスタモードで実行する Message Queue ブローカ。
- Message Queue ブローカと同じサーバーで実行する Berkeley DB クライアント (amsessiondb)。
- パフォーマンスおよびセキュリティ向上のためのロードバランサ。
- クライアント要求は、Web ブラウザ、Access Manager SDK を使用して C または Java アプリケーション、または J2EE/Web エージェントから発行できます。

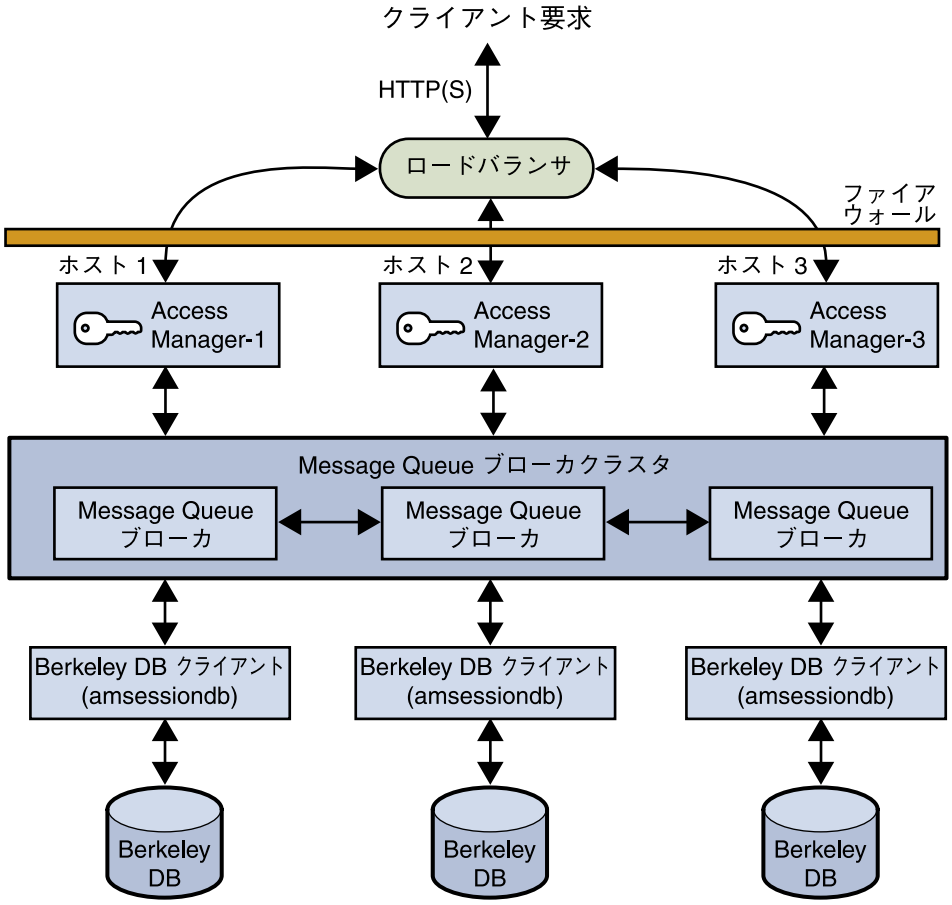


図 **6–1** Access Manager セッションフェイルオーバーシナリオ

セッションフェイルオーバーコンポーネントのインストール

次の表は、Access Manager セッションフェイルオーバーに必要なコンポーネントのインストール方法を説明しています。

表 **6–1** Access Manager セッションフェイルオーバーコンポーネントのインストール

コンポーネント	インストール方法
Access Manager	Java ES インストーラを使用して、各ホストサーバーに Access Manager の 1 番目のインスタンスをインストールします。インストーラによって、必要なセッションフェイルオーバー Solaris パッケージまたは Linux RPM が追加されます。 参照: 『Sun Java Enterprise System 2005Q4 Installation Guide for UNIX』 Java ES インストーラを使用して Access Manager をインストールする場合は、レルムモード (バージョン 7.x) または旧バージョンモード (バージョン 6.x) のどちらかを選択できます。Access Manager セッションフェイルオーバーは、両方のモードでサポートされています。 Java ES インストーラを実行したあと、`amconfig` スクリプトを実行して次のことを行います。 ■ インストール時に「あとで設定」オプションを指定した場合は、1 番目の Access Manager インスタンスを設定します。 ■ インストール済みの Access Manager インスタンスを再配備または再設定します。 詳細については、77 ページの「複数のホストサーバーへの Access Manager のインストール」を参照してください。
Message Queue	Java ES インストーラを使用して、Message Queue をインストールします。 参照: 『Sun Java Enterprise System 2005Q4 Installation Guide for UNIX』
Berkeley DB クライアント (Access Manager のサブコンポーネント)	Java ES インストーラおよび `amconfig` スクリプトによって、Berkeley DB クライアントに必要な Access Manager パッケージまたは RPM が追加されます。ただし、Access Manager がインストールされていないサーバーに Berkeley DB クライアントをインストールする場合は、使用しているオペレーティングシステムに応じて、次のパッケージまたは RPM を手動で追加する必要があります。 Solaris OS の場合は、`pkgadd` コマンドを使用して、次のパッケージを追加します。 `SUNWamsfodb`、`SUNWbdb`、および `SUNWbdbj`。 参照: Solaris のマニュアル Linux OS の場合は、`rpm` コマンドを使用して、次の RPM を追加します。 `sun-identity-sfodb`、`sun-berkeleydatabase-core`、および `sun-berkeleydatabase-java`。 参照: Linux のオンラインマニュアルページ。

注意 – 複数サーバーの配備では、Access Manager のすべてのインスタンスが同じパスワード暗号化鍵値を使用する必要があります。1 番目の Access Manager インスタンスをインストールしたとき、`AMConfig.properties` ファイル内の `am.encryption.pwd` プロパティーからパスワード暗号化鍵値を保存します。次に、Java ES インストーラまたは `amconfig` スクリプトを実行してほかのホストサーバーに Access Manager インスタンスを配備するとき、パスワード暗号化鍵にこの同じ値を使用します。

セッションフェイルオーバー用の Access Manager の設定

Access Manager をセッションフェイルオーバー用に設定するには、次の手順に従います。

- 93 ページの「1–Cookie エンコードの無効化」
- 94 ページの「2–Web コンテナの `server.xml` ファイルの編集 」
- 94 ページの「3–Message Queue サーバーでの新規ユーザーの追加」
- 94 ページの「4–`amsessiondb` スクリプトの編集 (必要な場合)」
- 95 ページの「5–`amsfoconfig` スクリプトの実行」

各手順を、以降の節で詳細に説明します。

配備でセッションフェイルオーバーが有効であるかどうかを判別するには、`AMConfig.properties` ファイルの `com.iplanet.services.debug.level` プロパティーを `error` から `message` に変更します。 次に、Solaris システムでは `/var/opt/SUNWam/debug` ディレクトリ、Linux システムでは `/var/opt/sun/identity/debug` ディレクトリにある `amSession` ログを確認します。

1–Cookie エンコードの無効化

Access Manager インスタンスを実行している各ホストサーバーで、Cookie エンコードを無効にします。

- Web Server が Web コンテナの場合、`AMConfig.properties` ファイルの次のプロパティーが false (Java ES インストーラによって設定されるデフォルト値) に設定されていることを確認してください。

  ```
  com.iplanet.am.cookie.encode=false
  ```

  `sun-web.xml` ファイルの `encodeCookies` プロパティーを false に設定します。次に例を示します。

  ```
  <sun-web-app>
  <property name="encodeCookies" value="false"/>
  ...
  </sun-web-app>
  ```

- Application Server、BEA WebLogic、または IBM WebSphere Application Server が Web コンテナの場合は、AMConfig.properties ファイルの次のプロパティーを false に設定します。

  ```
  com.iplanet.am.cookie.encode=false
  ```

Access Manager クライアントは、Cookie のエンコードもデコードも実行する必要はありません。リモート SDK クライアントは、AMConfig.properties ファイルまたは Web コンテナの sun-web.xml ファイルのどちらかで、Access Manager サーバー側の設定と同期させる必要があります。

2–Web コンテナの server.xml ファイルの編集

Access Manager インスタンスを実行している各ホストサーバーで、Access Manager Web コンテナの server.xml (または同等の) 設定ファイルに、imq.jar と jms.jar がインストールされている場所を追加します。Solaris システムの場合は次のようになります。

```
<JAVA javahome="/usr/jdk/entsys-j2se" serverclasspath=
"/usr/share/lib/imq.jar:/usr/share/lib/jms.jar:
/opt/SUNWwbsvr/bin/https/jar/webserv-rt.jar:
${java.home}/lib/tools.jar:
/opt/SUNWwbsvr/bin/https/jar/webserv-ext.jar:
/opt/SUNWwbsvr/bin/https/jar/webserv-jstl.jar:
/usr/share/lib/ktsearch.jar"
```

3–Message Queue サーバーでの新規ユーザーの追加

Message Queue のユーザー名とパスワードとして guest ユーザーを使用しない場合は、Message Queue がインストールされているサーバーで、Message Queue ブローカに接続するための新しいユーザーとパスワードを追加します。たとえば、Solaris システムで、amsvrusr という新しいユーザーを追加するには、次のように指定します。

```
# /usr/bin/imqusermgr add -u amsvrusr -p password
```

次のコマンドを発行して、guest ユーザーを非アクティブにします。

```
# /usr/bin/imqusermgr update -u guest -a false
```

4–amsessiondb スクリプトの編集 (必要な場合)

amsessiondb スクリプトは、Berkeley DB クライアント (amsessiondb) の起動、データベースの作成、および特定のデータベース値の設定を行うために、amsfo スクリプトから呼び出されます。このスクリプトには、次に示すように、各種のデフォルトのパスやディレクトリを指定する変数が含まれています。

```
JAVA_HOME=/usr/jdk/entsys-j2se/
IMQ_JAR_PATH=/usr/share/lib
JMS_JAR_PATH=/usr/share/lib
```

```
BDB_JAR_PATH=/usr/share/db.jar
BDB_SO_PATH=/usr/lib
AM_HOME=/opt/SUNWam
```

これらのいずれかのコンポーネントがそれらのデフォルトのディレクトリにインストールされていない場合は、必要に応じて amsessiondb スクリプトを編集し、変数に正しい場所を設定します。

5–amsfoconfig スクリプトの実行

Access Manager 7 2005Q4 には、Access Manager 配備をセッションフェイルオーバー用に設定するための amsfoconfig スクリプトが用意されています。

amsfoconfig スクリプトを実行するための要件

amsfoconfig スクリプトを実行するには、Access Manager 配備が次の要件を満たしている必要があります。

- 配備内に 2 つ以上の Access Manager インスタンスがインストールおよび設定されている必要がありますが、配備をサイトとして設定することはできません。amsfoconfig スクリプトによって、配備がサイトとして設定されているか、またはプラットフォームサーバーリスト内の任意のサーバーエントリでサイトが有効になっていると判断された場合、スクリプトはメッセージを表示して終了します。セッションフェイルオーバーを手動で設定するには、103 ページの「セッションフェイルオーバーの手動での設定」を参照してください。
- 配備内の少なくとも 2 つのサーバーに Java Message Queue (MQ) ブローカがインストールおよび設定されている必要があります。
- 配備内に Berkeley DB のクライアントとデータベースがインストールおよび設定されている必要があります。
- Directory Server が実行されており、このスクリプトからアクセス可能であり、かつ Access Manager データを使用して設定されている必要があります。

amsfoconfig スクリプトの機能

amsfoconfig スクリプトは amsfo.conf 設定ファイルを読み取り、次の機能を実行して、Access Manager 配備をセッションフェイルオーバー用に設定します。

- 新規サイトを設定します。スクリプトは、プラットフォームサーバーリスト内の Access Manager インスタンスと amsfo.conf ファイルのロードバランサに関する情報を使用して、Access Manager セッションフェイルオーバー配備用の新規サイトを作成します。このスクリプトは既存のプラットフォームサーバーリストを変更して、サイトが設定されたあと、プラットフォームサーバーリストのすべてのサーバーエントリがそのサイトに属するようにします。

  たとえば、デフォルト値 10 が SiteID として使用されている場合、
  http://server1.example.com:80|01 は
  http://server1.example.com:80|01|10 に変更されます。

- レルムまたは DNS のエイリアスの既存のリストを変更します。スクリプトは、ロードバランサのホスト名をこのリストに追加します。このホスト名は、`amsfo.conf` ファイルの `lbServerHost` 変数から取得されます。
- セッションフェイルオーバー設定 XML を Directory Server にロードします。スクリプトは、設定情報に基づいてセッション設定 XML ファイルを動的に生成し、生成された XML を Directory Server にロードします。この情報は、Access Manager コンソールの「セッション」の「セカンダリ設定インスタンス」に対応しています。

次の表は、Access Manager セッションフェイルオーバーのスクリプトと設定ファイルを示しています。

表 **6–2** Access Manager セッションフェイルオーバーのスクリプトと設定ファイル

名前	説明および場所
`amsofconfig`	Access Manager をセッションフェイルオーバー用に設定するためのスクリプト。
	Solaris システム: *AccessManager-base*`/SUNWam/bin`
	Linux システム: *AccessManager-base*`/identity/bin`
`amsfo`	Message Queue ブローカや `amsessiondb` クライアントを起動および停止するためのスクリプト。
	Solaris システム: *AccessManager-base*`/SUNWam/bin`
	Linux システム: *AccessManager-base*`/identity/bin`
`amsfopasswd`	暗号化された Message Queue ブローカユーザーのパスワードを生成するためのスクリプト。
	Solaris システム: *AccessManager-base*`/SUNWam/bin`
	Linux システム: *AccessManager-base*`/identity/bin`
`amsfo.conf`	セッションフェイルオーバーの設定ファイル。
	Solaris システム: *AccessManager-base*`/SUNWam/lib`
	Linux システム: *AccessManager-base*`/sun/identity/lib`
`amProfile.conf`	セッションフェイルオーバーの環境ファイル。
	Solaris システム: `etc/opt/SUNWam/config`
	Linux システム: `/etc/opt/sun/identity/config`

AccessManager-base は、Access Manager のベースインストールディレクトリを表します。デフォルト値は次のとおりです。

Solaris システム: `/opt`

Linux システム: `/opt/sun`

amsfoconfig スクリプトの実行

amsfoconfig スクリプトを実行して Access Manager をセッションフェイルオーバー用に設定するには、次の手順に従います。

1. スーパーユーザー (root) としてログインするか、スーパーユーザーになります。
2. 表 6–3の説明に従って、amsfo.conf ファイル内の変数を設定します。
3. スクリプトを実行します。たとえば、Access Manager がデフォルトディレクトリにインストールされた Solaris システムでは、次のように入力します。

```
# cd /opt/SUNWam/bin
# ./amsfoconfig
```

   スクリプトの実行時に、状態情報が表示されます。

4. amsfoconfig スクリプトから入力を求められたら、次のパスワードを入力します。
   - Access Manager 管理者 (amAdmin) のパスワード
   - Message Queue ブローカユーザーのパスワード
5. 結果をチェックするには、/var/tmp/amsfoconfig.log ファイルを確認します。

次の表は、amsfoconfig スクリプトで使用される amsfo.conf ファイル内の変数を示しています。amsfoconfig スクリプトを実行する前に、これらの変数を配備の必要に応じて設定します。

表 **6–3** amsfoconfig スクリプトで使用される amsfo.conf ファイル内の変数

変数	説明
CLUSTER_LIST	クラスタに参加している Message Queue ブローカのリスト。形式は次のとおりです。 *host1*:*port*, *host2*:*port*,*host3*: *port* 次に例を示します。 jmq1.example.com:7777,jmq2.example.com:7777,jmq3.example.com:7777 デフォルトはありません。
lbServerPort	ロードバランサのポート。デフォルトは 80 です。
lbServerProtocol	ロードバランサへのアクセスに使用されるプロトコル (http または https)。デフォルトは http です。
lbServerHost	ロードバランサの名前。 次に例を示します。lbhost.example.com

表 **6–3** amsfoconfig スクリプトで使用される amsfo.conf ファイル内の変数　(続き)

変数	説明
SiteID	amsfoconfig スクリプトによって作成される新規サイト(およびロードバランサ) の識別子。 SiteID は、プラットフォームサーバーリスト内にすでに存在しているサーバー ID より大きい任意の値にすることができます。 デフォルトは 10 です。

amsfoconfig スクリプトの実行例

次の例は、amsfoconfig スクリプトの実行例を示しています。

```
Welcome to Sun Java System Access Manager 7 2005Q4

Session Failover Configuration Setup script.
============================================================
============================================================
Checking if the required files are present...
============================================================

Running with the following Settings.
------------------------------------------------
Environment file: /etc/opt/SUNWam/config/amProfile.conf
Resource file: /opt/SUNWam/lib/amsfo.conf
         ------------------------------------------------
Using /opt/SUNWam/bin/amadmin

Validating configuration information.
Done...

Please enter the LDAP Admin password:
(nothing will be echoed): password1
Verify: password1
Please enter the JMQ Broker User password:
(nothing will be echoed): password2
Verify: password2

Retrieving Platform Server list...
Validating server entries.
Done...

Retrieving Site list...
Validating site entries.
Done...

Validating host: http://amhost1.example.com:7001|02
Validating host: http://amhost2.example.com:7001|01
Done...

Creating Platform Server XML File...
Platform Server XML File created successfully.
```

```
Creating Session Configuration XML File...
Session Configuration XML File created successfully.

Creating Organization Alias XML File...
Organization Alias XML File created successfully.

Loading Session Configuration schema File...
Session Configuration schema loaded successfully.

Loading Platform Server List File...
Platform Server List server entries loaded successfully.

Loading Organization Alias List File...
Organization Alias List loaded successfully.

Please refer to the log file /var/tmp/amsfoconfig.log for additional
information.
################################################################
Session Failover Setup Script. Execution end time 10/05/05 13:34:44
################################################################
```

セッションフェイルオーバーコンポーネントの起動

Access Manager 7 2005Q4 には、次の機能を実行するための `amsfo` スクリプトが用意されています。

- セッションフェイルオーバー配備用に指定されている Java Message Queue (MQ) ブローカを起動および停止します。
- セッションフェイルオーバー配備用に指定されている `amsessiondb` クライアントを起動および停止します。
- `amsfo.conf` 設定ファイルを読み取り、ファイル内の変数に基づいた特定の操作を実行します。たとえば、スクリプトで、最初に Berkeley DB データベースを削除し、次に再作成するように設定できます。
- `/tmp/amsession/logs/` ディレクトリ内の `amsessiondb.log`、`jmq.pid`、および `amdb.pid` の各ファイルに書き込みます。デフォルトのログディレクトリは、`amsfo.conf` ファイル内の `LOG_DIR` 変数によって決定されます。

Access Manager セッションフェイルオーバーコンポーネントを起動するには、次の手順に従います。

1. 配備の必要に応じて、`amsfo.conf` 設定ファイル内の変数を設定します。これらの変数については、表 6–4を参照してください。
2. `amsfo` スクリプトを実行して、Java Message Queue (MQ) ブローカおよび `amsessiondb` クライアントを起動します。詳細は、100 ページの「`amsfo` スクリプトの実行」を参照してください。
3. 各 Access Manager インスタンスを、それぞれの Web コンテナを起動することにより起動します。詳細は、『Sun Java System Access Manager 7 2005Q4 管理ガイド』を参照してください。

amsfo スクリプトの実行

amsfo スクリプトには、次の起動および停止オプションが含まれています。

使用法: amsfo { start | stop }

amsfo スクリプトを実行するには、次の手順に従います。

1. スーパーユーザー (root) としてログインするか、スーパーユーザーになります。
2. 配備の必要に応じて、amsfo.conf ファイル内の変数を設定します。これらの変数については、表 6–4を参照してください。
3. スクリプトを実行します。たとえば、Access Manager がデフォルトディレクトリにインストールされた Solaris システムでセッションフェイルオーバーコンポーネントを起動するには、次のように入力します。

```
# cd  /opt/SUNWam/bin
# ./amsfo start
```

4. スクリプトの結果をチェックするには、/tmp/amsession/logs/amsessiondb.log ファイルを確認します。

次の表は、amsfo.conf 設定ファイル内の変数を示しています。amsfo スクリプトを実行する前に、これらの変数を配備の必要に応じて設定します。

表 **6–4** amsfo.conf 設定ファイル

変数	説明
AM_HOME_DIR	Access Manager のデフォルトのインストールディレクトリ。デフォルトディレクトリは、次のようにプラットフォームによって異なります。 Solaris システム: *AccessManager-base*/SUNWam/opt Linux システム: *AccessManager-base*/identity/opt *AccessManager-base* は、Access Manager のベースインストールディレクトリを表します。デフォルト値は、Solaris システムでは /opt、Linux システムでは /opt/sun です。
AM_SFO_RESTART	スクリプトで amsessiondb クライアントを自動的に再起動するかどうかを指定します (true または false)。 デフォルトは true (amsessiondb クライアントを再起動する) です。

表 **6–4** amsfo.conf 設定ファイル　(続き)

変数	説明
CLUSTER_LIST	クラスタに参加している Message Queue ブローカのリスト。形式は次のとおりです。 *host1*:*port*, *host2*:*port*,*host3*: *port* 次に例を示します。 jmq1.example.com:7777,jmq2.example.com:7777,jmq3.example.com:7777 デフォルトはありません。
DATABASE_DIR	セッションデータベースファイルを作成するディレクトリ。 デフォルトは "/tmp/amsession/sessiondb" です。
DELETE_DATABASE	スクリプトで amsessiondb プロセスを再起動するときに、新規データベースを削除してから作成するかどうかを指定します (true または false)。 デフォルトは true です。
LOG_DIR	ログディレクトリの場所。 デフォルトは "/tmp/amsession/logs" です。
START_BROKER	amsessiondb プロセスとともに Message Queue ブローカを起動するかどうかを指定します (true または false)。この変数を次のように設定します。 true - Message Queue ブローカは、amsessiondb プロセスと同じマシン上で実行されます。 false - Message Queue ブローカと amsessiondb プロセスは、別のマシン上で実行されます。 デフォルトは true です。
BROKER_INSTANCE_NAME	起動する Message Queue ブローカインスタンスの名前。 デフォルトは aminstance です。
BROKER_PORT	ローカル Message Queue ブローカインスタンスのポート。 デフォルトは 7777 です。
BROKER_VM_ARGS	Java VM の引数。デフォルトは "-Xms256m -Xmx512m" です。これにより、システムリソースに基づく最大値が設定されます。
USER_NAME	Message Queue ブローカーへの接続に使用されるユーザー名。 デフォルトは「guest」です。94 ページの「3–Message Queue サーバーでの新規ユーザーの追加」 の手順で別のユーザー名を指定した場合は、USER_NAME にその名前を設定します。

表 **6–4** amsfo.conf 設定ファイル　(続き)

変数	説明
PASSWORDFILE	Message Queue ブローカへの接続に使用される暗号化パスワードを含むパスワードファイルの場所。暗号化パスワードを生成するには、102 ページの「amsfopasswd スクリプト」の説明に従って amsfopasswd スクリプトを使用します。 デフォルトは *$AM_HOME_DIR*/.password です。*$AM_HOME_DIR* は、Access Manager のデフォルトのインストールディレクトリを指定します。

amsfopasswd スクリプト

amsfopasswd スクリプトはテキスト形式の Message Queue ブローカパスワードを受け取り、暗号化パスワードをファイル内に格納して返します。次に、このファイルを amsfo スクリプトへの入力 (PASSWORDFILE 変数) として使用できます。

amsfopasswd スクリプトは、次のディレクトリに格納されています。

- Solaris システム: *AccessManager-base* /SUNWam/bin
- Linux システム: *AccessManager-base* /identity/bin

デフォルトの *AccessManager-base* インストールディレクトリは、Solaris システムでは /opt、Linux システムでは /opt/sun です。

次の構文を使用して amsfopasswd スクリプトを実行します。

```
amsfopasswd -f filename | --passwordfile filename
            -e password | --encrypt password
amsfopasswd -h | --help
```

次の表は、amsfopasswd スクリプトの引数を示しています。

表 **6–5** amsfopasswd スクリプトの引数

引数	説明
-f *filename* | --passwordfile *filename*	amsfopasswd が暗号化パスワードを保存するファイルのパス。
-e *password* | --encrypt *password*	amsfopasswd が暗号化するテキストパスワード。
-h | --help	amsfopasswd コマンドの使用例を表示して、終了します。

次の例は、amsfopasswd スクリプトを示しています。暗号化パスワードは、/opt/SUNWam/.password ファイルに格納されます。

```
# ./amsfopasswd -f /opt/SUNWam/.password -e mypassword
```

セッションフェイルオーバーの手動での設定

場合によっては、Access Manager をセッションフェイルオーバー用に手動で設定する必要があります。たとえば、`amsfoconfig` スクリプトの実行を計画していない場合があります。または、`amsfoconfig` スクリプトが設定を終了する前に次のメッセージを表示して終了する場合もあります。「サイトはすでに設定されています」または「サーバーエントリはすでにサイト設定されています」。

次の手順は、Access Manager をセッションフェイルオーバー用に手動で設定する方法について説明しています。

- 103 ページの「1–配備に必要なコンポーネントのインストール」
- 103 ページの「2–サイトとしての Access Manager 配備の設定」
- 103 ページの「3–ロードバランサ用の新規セカンダリ設定インスタンスの作成」
- 104 ページの「4–セッションフェイルオーバーのその他の設定作業の実行」
- 104 ページの「5–セッションフェイルオーバーコンポーネントの起動」

これらの手順は、前に説明した手順、つまり必要なコンポーネントをインストールし、`amsfoconfig` スクリプトを使用してセッションフェイルオーバーを設定したあと、さまざまなコンポーネントを起動する方法の手順に対応しています。

1–配備に必要なコンポーネントのインストール

配備で、Access Manager インスタンス、ロードバランサ、Message Queue、および Berkeley DB クライアントを含むすべてのコンポーネントをインストールします。詳細は、91 ページの「セッションフェイルオーバーコンポーネントのインストール」を参照してください。

2–サイトとしての Access Manager 配備の設定

複数の Access Manager インスタンスとロードバランサをサイトとして設定する `amsfoconfig` スクリプトの実行を計画していない場合は、81 ページの「サイトとしての Access Manager 配備の設定」の説明に従って配備を設定する必要があります。

3–ロードバランサ用の新規セカンダリ設定インスタンスの作成

ロードバランサ用の新規セカンダリ設定インスタンスを作成するには、次の手順に従います。

1. `amAdmin` として Access Manager 7 2005Q4 コンソール にログインします。
2. 「設定」、「グローバルプロパティー」、「セッション」、「セカンダリ設定インスタンス」の順にクリックします。
3. 「新規」をクリックし、次の値を追加します。

- 名前: ロードバランサの URL。次に例を示します。
  `http://lb.example.com:80`
- セッションストアユーザー: Message Queue サーバーへの接続に使用している名前 (「guest」以外に存在する場合)。
- セッションストアパスワード: セッションストアユーザーのパスワード。
- 最大待ち時間: 5000。別の値が必要な場合以外は、デフォルト値を使用してください。
- データベース URL: Message Queue ブローカのアドレスリスト。次に例を示します。

  `mqsvr1.example.com:7777,mqsvr2.example.com:7777,mqsvr3.example.com:7777`

  デフォルトの Message Queue ポートは 7676 です。ただし、Application Server を Web コンテナとして使用している場合は、ポート 7676 はすでに Application Server で使用されている可能性があるため、別のポートを使うことを検討してください。有効なポート番号の範囲については、Message Queue のマニュアルを参照してください。

4. 「追加」をクリックして変更を保存します。

4–セッションフェイルオーバーのその他の設定作業の実行

次のタスクを実行します。これらのタスクは、`amsfoconfig` スクリプトを実行している場合と同じです。

- 93 ページの「1–Cookie エンコードの無効化」
- 94 ページの「2–Web コンテナの `server.xml` ファイルの編集 」
- 94 ページの「3–Message Queue サーバーでの新規ユーザーの追加」
- 94 ページの「4–`amsessiondb` スクリプトの編集 (必要な場合)」

5–セッションフェイルオーバーコンポーネントの起動

`amsfo` スクリプトを実行して、Message Queue ブローカと Berkeley DB クライアント(`amsessiondb`) を起動します。次に、各 Access Manager インスタンスを、それぞれの Web コンテナを起動することにより起動します。99 ページの「セッションフェイルオーバーコンポーネントの起動」を参照してください。

`amsessiondb` スクリプト

`amsessiondb` スクリプトは、Berkeley DB クライアント (`amsessiondb`) の起動、データベースの作成、および特定のデータベース値の設定を行うために、`amsfo` スクリプトから呼び出されます。

注 – Access Manager セッションフェイルオーバーコンポーネントの起動と停止には、amsfo スクリプトを実行し、そのスクリプトから amsessiondb スクリプトを呼び出す方法をお勧めします。次の情報は、amsessiondb スクリプトを単独で実行する必要が生じた場合のためにのみ提供されています。

amsessiondb スクリプトを実行する前に、94 ページの「4–amsessiondb スクリプトの編集 (必要な場合)」の説明に従ってパスが正しく設定されていることを確認してください。

amsessiondb スクリプトを実行する場合、Message Queue ブローカーパスワードをコマンド行にテキストで入力できます (-w または --password オプション)。ただし、ファイル内で暗号化パスワードを使用する場合(-f または --passwordfile オプション) は、最初に amsfopasswd スクリプトを実行してMessage Queue ブローカテキストパスワードをファイル内に暗号化します。次に、このファイルを -f または --passwordfile オプションに使用して、amsessiondb スクリプトを実行します。

次の構文を使用して amsessiondb スクリプトを実行します。

```
amsessiondb [ -u username | --username username ]
[ -w password | --password password |
-f filename | --passwordfile filename ]
[ -c cachesize | --cachesize cachesize ]
[ -b dbdirectory | --dbdirectory dbdirectory ]
-a MQServerAddressList | --clusteraddress MQServerAddressList
[ -s numcleanexpiredsessions | --numcleansessions numcleanexpiredsessions ]
[ -v | --verbose ]
[ -i statsinterval | --statsInterval statsinterval ]
amsessiondb -h | --help
amsessiondb -n | --version
```

次の表は、amsessiondb スクリプトの引数を示しています。

表 **6–6** amsessiondb スクリプトの引数

引数	説明
-u *username* | --username *username*	Message Queue ブローカーに接続するユーザー名。94 ページの「3–Message Queue サーバーでの新規ユーザーの追加」で指定したユーザーを指定します。 デフォルトは「guest」です。
-w *password* | --password *password*	Message Queue ブローカに接続するために使用するユーザー名のテキストパスワード。94 ページの「3–Message Queue サーバーでの新規ユーザーの追加」で指定したパスワードを指定します。 デフォルトは「guest」です。

表 **6–6** amsessiondb スクリプトの引数　(続き)

引数	説明
-f *filename* | --passwordfile *filename*	Message Queue ブローカーにアクセスするための暗号化パスワードを格納するファイル。 このオプションを指定する場合は、-w または --password オプションを指定しないでください。
-c *cachesize* | --cachesize *cachesize*	M バイト単位でのキャッシュサイズ。デフォルトは 8M バイトです。
-b *dbdirectory* | --dbdirectory *dbdirectory*	Berkeley DB データベース (amsessions.db) が作成されるベースディレクトリ。 デフォルトは "sessiondb" です。amsessiondb スクリプトを実行しているディレクトリに作成されます。 注**:** データベースを作成するディスク領域を十分に確保するには、100,000 セッションあたり 1G バイト必要です。
-a *MQServerAddressList* | --clusteraddress *MQServerAddressList*	次の形式の Message Queue ブローカアドレスリスト。 *host*1: *port*[,*host*2: *port*,*host* 3:*port*,...] 次に例を示します。mqsvr1:7777,mqsvr2:7777
-s numcleanexpiredsessions | --numcleansessions numcleanexpiredsessions	クリーンアップ間隔ごとに削除される期限切れのセッション数。 デフォルトは 1000 です。
-v | --verbose	冗長モードで実行します。結果は標準出力に送られます。 デフォルトは、非冗長モードです。
-i statsinterval | --statsInterval statsinterval	要求、読み取り、書き込み、削除の合計の統計情報を標準出力に出力する秒単位の間隔。 デフォルトは 60 秒です。
-h | --help	amsessiondb コマンドの使用例を表示して、終了します。
-n | --version	現在インストールされている Access Manager のバージョンを返し、終了します。

次の例は、amsessiondb スクリプトを示しています。

```
amsessiondb -u amsvrusr -f pwfile -c 128 -b sessiondb
-a host1:7777,host2:7777
```

amsessiondb クライアントを使用したパフォーマンステスト

amsessiondb クライアントを使用したパフォーマンステストの条件は、次のとおりです。

- Berkeley DB 上の処理数は、1 秒間の認証回数の 2 倍でした。
- テストは次の設定で行われました。
  - 書き込みデータ – 3K バイト
  - 持続時間 – 1 分
  - Berkeley DB キャッシュサイズ – 28M バイト (Access Manager 7 2005Q4 のデフォルトキャッシュサイズは 32M バイト)

次の表は、テスト結果を示しています。

表 **6–7** amsessiondb クライアントを使用したパフォーマンステスト

ディスク	備考
標準の IDE ディスク: 1 秒間に 666 回の書き込み	各サイトは 1 秒間に最大 300 回の認証をサポートできます。 そのため、IDE ディスクは推奨されません。
標準の 10K RPM SCSI ディスク (Sun Blade サーバー上): 1 秒間に 1520 回の書き込み	各サイトは 1 秒間に最大 750 回の認証をサポートできます。
Seagate Cheetah 15K RPM SCSI ディスク: 1 秒間に 1860 回の書き込み	各サイトは 1 秒間に最大 900 回の認証をサポートできます。
Sun T-300 ディスクアレイ: 1 秒間に 2700 回の書き込み	各サイトは 1 秒間に最大 1300 回の認証をサポートできます。
/tmp 内のスワップ領域を使用するディスク: 1 秒間に 3300 回の書き込み	各サイトは 1 秒間に最大 1600 回の認証をサポートできます。

セッション割り当て制限の設定

Access Manager 7 2005Q4 には、設定可能な属性に基づいてAccess Manager がユーザー数を特定の数のアクティブな並行セッションに制限できるようにする、新しいセッション割り当て制限の機能が含まれています。Access Manager 管理者は、セッション割り当て制限を次のレベルで設定できます。

- グローバルに。すべてのユーザーに制限が適用されます。
- エンティティー (組織またはレルム、ロール、またはユーザー) に対して。そのエンティティーに属する特定のユーザーにのみ、制限が適用されます。

セッション割り当て制限のための配備シナリオ

セッション割り当て制限は、次の Access Manager 配備でサポートされています。

- Access Manager 単一サーバー配備

  このシナリオでは、Access Manager が単一のホストサーバーに配備されます。Access Manager は、ログインしたすべてのユーザーのアクティブなセッション数をメモリー内に保持します。ユーザーがサーバーにログインしようとすると、Access Manager は、そのユーザーの有効なセッション数がセッション制限を超えているかどうかをチェックしたあと、設定されているセッション割り当て制限オプションに基づくアクションを実行します。

- Access Manager セッションフェイルオーバー配備

  このシナリオでは、Access Manager の複数インスタンスが、異なるホストサーバー上にセッションフェイルオーバー設定で配備されます。Access Manager インスタンスは、Sun Java System Message Queue (Message Queue) を通信ブローカとして、また Sleepycat Software, Inc. の Berkeley DB をセッションストアデータベースとして使用して、セッションフェイルオーバー用に設定されます。Access Manager セッションフェイルオーバーについては、89 ページの「Access Manager セッションフェイルオーバーの実装」を参照してください。

セッションフェイルオーバー配備では、ユーザーがログインしようとすると、セッション作成要求を受信している Access Manager サーバーは、まず、Access Manager アイデンティティーリポジトリからそのユーザーのセッション制限を取得します。次に、そのユーザーのセッション数をセッションリポジトリから直接フェッチし (同じサイト内のすべての Access Manager サーバーからすべてのセッションを蓄積)、セッション制限がいっぱいになっているかどうかをチェックします。そのユーザーのセッション制限がいっぱいになっている場合、Access Manager サーバーは、設定されているセッション割り当て制限オプションに基づく操作を実行します。

セッションフェイルオーバー配備でセッション制限が有効になっており、セッションリポジトリが使用できない場合、ユーザー (スーパーユーザーを除く) のログインは許可されません。

セッションフェイルオーバー配備では、Access Manager インスタンスが停止していると、以前にそのインスタンスによってホストされていたすべての有効なセッションがまだ有効であると見なされ、サーバーが特定のユーザーの実際のアクティブなセッション数を判定するときにカウントされます。セッションフェイルオーバー用に設定されていない Access Manager 複数サーバー配備では、セッション割り当て制限はサポートされません。

セッション割り当て制限の設定

セッション割り当て制限を設定するには、Access Manager の最上位レベル管理者 (`amAdmin` など) が Access Manager コンソールでいずれかの Access Manager インスタンスに対して次の属性を設定する必要があります。これらの属性のいずれかをリセットした場合は、サーバーを再起動して新しい値を有効にする必要があります。

- 「割り当て制限を有効」は、セッション割り当て制限の機能を有効または無効にするためのグローバル属性です。この属性が有効になっていると、ユーザーが新しいクライアント経由でログイン (また、それによって新しいセッションを作成) しようとした場合は常に、Access Manager によってセッション割り当て制限が適用されます。

  デフォルトは無効 (「オフ」) です。

- 「割り当て制限のタイムアウトを読み取り」は、アクティブなユーザーセッション数に対するセッションリポジトリへの問い合わせがタイムアウトになるまでの時間をミリ秒単位で定義します。セッションリポジトリが使用できないために最大待ち時間に到達した場合は、セッション作成要求が拒否されます。

  デフォルトは 6000 ミリ秒です。

- 「セッション制限がいっぱいになった場合に生じる動作」は、ユーザーのセッション割り当て制限がいっぱいになった場合の動作を決定します。この属性は、「割り当て制限を有効」属性が有効になっている場合にのみ有効になります。次のいずれかの値を設定できます。

  - DENY_ACCESS。Access Manager は、新しいセッションに対するログイン要求を拒否します。
  - DESTROY_OLD_SESSION。Access Manager は、同じユーザーで次に有効期限切れとなるセッションを破棄して、新しいログイン要求に成功できるようにします。

  デフォルトは DESTROY_OLD_SESSION です。

- 「トップレベルの管理者に制限の確認を免除」は、セッション割り当て制限を、トップレベルの管理者ロールを持つ管理者に適用するかどうかを指定します。 この属性は、「割り当て制限を有効」属性が有効になっている場合にのみ有効になります。

  デフォルトは「いいえ」です。

  `AMConfig.properties` ファイル (`com.sun.identity.authentication.super.user`) で定義された Access Manager のスーパーユーザーは、常にセッション割り当て制限の確認から免除されます。

- 「アクティブなユーザーセッション」は、ユーザーの並行セッションの最大数を定義します。Access Manager には、同じ属性名を持つ、動的属性とユーザー属性の両方が含まれています。

  デフォルトは 5 です。

セッション制限の複数設定

ユーザーが、異なるレベルで複数のセッション制限を設定している場合、Access Manager は、そのユーザーの実際の割り当て制限を次の優先順位に従って判定します。

- ユーザー (最高)

- ロール/組織/レルム (競合の解決レベルに基づく)
- グローバル (最低)

たとえば、Ken がマーケティングロールと管理ロールの両方のメンバーだとします。セッション制限は、次のように定義されています (競合の解決レベルはすべて同じ)。

- 組織 - 1
- マーケティングロール - 2
- 管理ロール - 4
- ユーザー Ken - 3

Ken の割り当て制限は 3 です。

セッション割り当て制限の属性については、Access Manager コンソールのオンラインヘルプを参照してください。

セッションプロパティー変更通知の有効化

セッションプロパティー変更通知の機能を使用した場合、特定のセッションプロパティーに変更が発生すると、Access Manager は登録されているすべてのリスナーに通知を送信します。この機能は、Access Manager コンソールで「プロパティーの変更通知を有効」属性が有効(「オン」) になっている場合に有効になります。

たとえば、シングルサインオン (SSO) 環境では、複数のアプリケーションで 1 つの Access Manager セッションを共有できます。「通知プロパティー」リストで定義された特定のセッションプロパティーに変更が発生すると、Access Manager は、登録されたすべてのリスナーに通知を送信します。

SSO に参加しているクライアントアプリケーションはすべて、通知モードに設定されていれば、セッション通知を自動的に取得します。クライアントでキャッシュされたセッションは、新しいセッション状態 (任意のセッションプロパティーの変更を含む) に基づいて自動的に更新されます。セッション通知に基づいて特定のアクションを実行しようとするアプリケーションは、`SSOTokenListener` インタフェースの実装を記述し、次にこの実装を `SSOToken.addSSOTokenListener` メソッドを通して登録することができます。詳細は、『Sun Java System Access Manager 7 2005Q4 Developer's Guide』を参照してください。

セッションプロパティー変更通知を設定するには、次の手順に従います。

1. `amadmin` として Access Manager コンソールにログインします。
2. 「設定」タブをクリックします。
3. 「グローバルプロパティー」で「セッション」をクリックします。
4. 「プロパティーの変更通知を有効」を「オン」に設定します。

5. 「通知プロパティー」リストで、プロパティーが変更されたときに通知を送信する各プロパティーを追加します。
6. リストへのプロパティーの追加が終了したら、「保存」をクリックします。

配備のチューニング

Access Manager のインストール後、amtune や関連するスクリプトを使用して配備のチューニングを行い、パフォーマンスを最適化できます。これらのスクリプトを使用すると、Access Manager、Solaris™ Operating System (OS)、Web コンテナ、および Directory Server のチューニングを行うことができます。

Java Enterprise System インストーラにより、チューニングスクリプトと関連ファイルが bin/amtune ディレクトリにインストールされます。

amtune は、対話型のスクリプトではありません。amtune を実行する前に、amtune-env 設定ファイルのパラメータを編集して、特定の環境で amtune が実行するチューニングを指定する必要があります。amtune-env 設定ファイルには、次の 2 つの主要なセクションがあります。

- チューニングを制御するパフォーマンス関連のパラメータ。
- Access Manager エンジニアリングにより管理される内部セクション。変更不可。

amtune スクリプトは次の 2 つのモードで実行できます。

- レビューモード: amtune が推奨チューニングのレポートを行いますが、環境に対して実際の変更は行いません。
- 変更モード: amtune-env 設定ファイルのパラメータに基づいて、amtune が Directory Server 以外に対して実際の変更を行います。

amtune スクリプトが Directory Server のチューニングを自動的に行うことはありません。ほとんどの配備には、Access Manager 以外にも Directory Server にアクセスするアプリケーションがあります。したがって、ほかのアプリケーションに与える影響を考慮せずにチューニングによる変更を行うことは好ましくありません。

Directory Server のチューニングを行う前に、db2bak を使用して Directory Server のデータのバックアップを取ります。

amtune を実行すると、このスクリプトにより、 Directory Server チューニングスクリプト amtune-directory を含む tar ファイルが作成されます。このファイルを一時ディレクトリで解凍し、スクリプトをレビューモードで実行します。この変更が、配備で使用するすべてのアプリケーションで受け入れられるものであることを確認できたら、amtune-directory を変更モードで実行します。

チューニングスクリプトの実行、および amtune-env 設定ファイルでのチューニングパラメータの設定については、『Sun Java System Access Manager 7 2005Q4 Performance Tuning Guide』を参照してください。

付録 A

インストールされる製品のレイアウト

この付録では、Sun Java™ System Enterprise System (Java ES) インストーラを使用して Sun Java™ System Access Manager 7 2005Q4 をインストールしたあとのディレクトリレイアウトについて説明します。

次の表は、インストール後の Access Manager のデフォルトディレクトリの概要を示しています。

Access Manager ディレクトリの概要

表 **A–1** Access Manager ディレクトリの概要

説明	デフォルトディレクトリ
ベースインストールディレクトリ 114 ページの「ベースインストールディレクトリ」を参照してください。	Solaris システム: `/opt/SUNWam` Linux システム: `/opt/sun/identity` インストール中に、必要に応じて `/opt` または `/opt/sun` の代わりに別のベースインストールディレクトリを指定できますが、製品ディレクトリ名 (`/SUNWam` または `/identity`) は変更しないでください。
設定ディレクトリ 119 ページの「設定 (`/config`) ディレクトリ」を参照してください。	Solaris システム: `/etc/opt/SUNWam/config` Linux システム: `/etc/opt/sun/identity/config`
一時ファイルディレクトリ	Solaris システム: `/var/opt/SUNWam/tmp` Linux システム: `/var/opt/sun/identity/tmp`

表 **A–1** Access Manager ディレクトリの概要　(続き)

説明	デフォルトディレクトリ
デバッグファイルディレクトリ	Solaris システム: `/var/opt/SUNWam/debug` Linux システム: `/var/opt/sun/identity/debug`
ログファイルディレクトリ	Solaris システム: `/var/opt/SUNWam/logs` Linux システム: `/var/opt/sun/identity/logs`

ベースインストールディレクトリ

デフォルトのベースインストールディレクトリは、Access Manager をインストールするプラットフォームによって異なります。

- Solaris システム: `/opt`
- Linux システム: `/opt/sun`

Access Manager のマニュアルでは、*AccessManager-base* 変数を使用してベースインストールディレクトリを示しています。

ベースインストールディレクトリ内で、Access Manager パッケージ、共有バイナリファイル、コマンド行ツール、およびその他のファイルは、Solaris システムの `/SUNWam` ディレクトリ、および Linux システムの `/identity` ディレクトリにインストールされます。したがって、デフォルトのベースおよび製品ディレクトリも、プラットフォームによって異なります。

- Solaris システム: `/opt/SUNWam`
- Linux システム: `/opt/sun/identity`

注 – インストール中に、必要に応じて別のベースインストールディレクトリを指定できますが、製品ディレクトリ名 (`/SUNWam` または `/identity`) は変更しないでください。

`/SUNWam` または `/identity` ディレクトリには、次のファイルとディレクトリが含まれます。

- Web アプリケーションアーカイブ (WAR) ファイル (`amcommon.war`、`amconsole.war`、`ampassword.war`、および `amserver.war`)

  WAR ファイルについては、『Sun Java System Access Manager 7 2005Q4 Developer's Guide』を参照してください。

  サブディレクトリ:

  - 115 ページの「/bin ディレクトリ」

Access Manager をインストールしたあと、pkgchk（1M）ユーティリティーを使用して、パッケージのインストールが正しく行われたことを確認してください。次に例を示します。

```
pkgchk -l -p /opt/SUNWam
```

/bin ディレクトリ

次の表では、/bin ディレクトリのコマンド行ツールおよびユーティリティーについて説明しています。これらのツールおよびユーティリティーについては、『Sun Java System Access Manager 7 2005Q4 管理ガイド』を参照してください。

表 **A–2** Access Manager のコマンド行ツールおよびユーティリティー

ユーティリティー	説明
am2bak	Access Manager コンポーネントをバックアップします。
amadmin	XML サービスを Directory Server にロードし、DIT でバッチ管理タスクを実行します。
amsfo、amsfoconfig、amsfopassword	Access Manager セッションフェイルオーバースクリプト。
ampassword	Access Manager 管理者またはユーザーのパスワードを変更します。
amsamplesilent	インストールスクリプトおよび設定スクリプトで使用するサンプルのサイレントインストールファイル。

表 **A–2** Access Manager のコマンド行ツールおよびユーティリティー　*(続き)*

ユーティリティー	説明
amconfig、amutils、amdsconfig 、amsdkconfig、amsvcconfig、amas70config、amwas51config、amwl81config 、amws61config	Access Manager インスタンスのインストール、設定、およびアンインストールに使用する、インストール、および設定スクリプト。これらのスクリプトについては、『Sun Java System Access Manager 7 2005Q4 管理ガイド』を参照してください。
amserver	amunixd デーモンと amsecuridd デーモンを起動および停止します。
amtune	パフォーマンスを改善するために、オペレーティングシステム、Access Manager、Web コンテナ、および Directory Server パラメータを設定します。
amverifyarchive	ログアーカイブを調べて、アーカイブ内のすべてのファイルの改ざんや削除を検出します。
bak2am	am2back ユーティリティーでバックアップした Access Manager コンポーネントを復元します。
ldapmodify	新規エントリを追加するか、既存のエントリを変更して、LDAP ディレクトリの内容を編集します。
ldapsearch	LDAP ディレクトリに検索要求を発行し、結果を LDIF テキストで表示します。
amGenerateLDIF.pl、amGenerateNI.pl	Access Manager の一括連携スクリプト。
am2bak.template、amserver.template、amadmin.template、amverifyarchive.template、ampassword.template、bak2am.template	Access Manager のテンプレートファイル。

/docs ディレクトリ

/docs ディレクトリには、Java API リファレンス (Javadoc) に使用される HTML、JAR、CSS および関連ファイルが含まれます。

/dtd ディレクトリ

/dtd ディレクトリには、Access Manager で使用される DTD (Document Type Definition) ファイルが含まれます。DTD は、Access Manager がアクセスする XML ファイルの構造を定義します。詳細は、『Sun Java System Access Manager 7 2005Q4 Developer's Guide』を参照してください。

次の表では /dtd ディレクトリの Access Manager DTD ファイルについて説明しています。

表 **A–3** Access Manager DTD ファイル

ファイル	説明
Auth_Module_Properties.dtd	認証モジュールがプロパティーの指定に使用するXML ファイルの構造を定義します。
amAdmin.dtd	amAdmin コマンド行ツールを使用してディレクトリツリー上でバッチ LDAP 操作を実行する際に使用する XML ファイルの構造を定義します。
amWebAgent.dtd	Web エージェントからの要求を処理し、応答を Web エージェントに送信する際に使用する XML ファイルの構造を定義します。これは、下位互換性を維持する目的で残されている非推奨のファイルです。
policy.dtd	ポリシーを Directory Server に格納する際に使用する XML ファイルの構造を定義します。
remote-auth.dtd	認証サービスのリモート認証 API により使用される XML ファイルの構造を定義します。
server-config.dtd	すべてのサーバーおよびユーザータイプの ID、ホスト、およびポート情報を記述する serverconfig.xml の構造を定義します。
sms.dtd	XML サービスファイルの構造を定義します。
web-app_2_2.dtd	Access Manager 配備コンテナが J2EE アプリケーションを配備する際に使用する XML ファイルの構造を定義します。

/include ディレクトリ

/include ディレクトリにはヘッダー (.h) ファイルが含まれます。

/ldaplib ディレクトリ

/ldaplib/ldapsdk サブディレクトリには、Access Manager に含まれる LDAP ユーティリティーの実行に必要な共有オブジェクト (.so) ファイルが含まれます。

/lib ディレクトリ

/lib ディレクトリには、JAR ファイルおよび追加の共有オブジェクト (.so) ファイルが含まれます。また、/etc/opt/SUNWam/config/AMConfig.properties ファイルへのリンクも含まれます。

/locale ディレクトリ

/locale ディレクトリには、地域対応化プロパティファイルが含まれます。各プロパティファイルには、英語版の対応するファイルが含まれます。たとえば、amAdminCLI_en.properties は amAdminCLI.properties に対応するファイルです。

/migration ディレクトリ

/migration ディレクトリには、以前のバージョンの Access Manager からのデータの移行に使用するスクリプトとサポートファイルが含まれます。たとえば、/opt/SUNWam/migration/61to62/scripts サブディレクトリには、DIT を Access Manager 2005Q1 へ移行する場合に使用する Upgrade61DitTo62 スクリプトが含まれます。

移行については、『Sun Java Enterprise System 2005Q4 アップグレードガイド』(パーツ番号 819–3456) を参照してください。

/public_html ディレクトリ

/public_html ディレクトリとサブディレクトリには、オンラインヘルプに使用される HTML ファイルおよび関連するファイルが含まれます。

/samples ディレクトリ

/samples ディレクトリには、次のサブディレクトリが含まれます。 /admin、/appserver、/authentication、 /console、/csdk、/liberty、/logging、/phase2、/policy、 /saml、および /sso。

各サブディレクトリには、サブディレクトリ名で示されたそれぞれの機能に応じたサンプルが含まれます。これらのサンプル別の詳細については、Readme.html ファイルを参照してください。

/share ディレクトリ

/share/bin サブディレクトリには、Access Manager 内部で使用される次のユーティリティーが含まれます。

- amtune/amtune-utils
- amsecuridd、amunixd、 amwar、checkport、および wsutils.ksh

/upgrade ディレクトリ

/upgrade ディレクトリには、次のディレクトリが含まれます。

- /scripts には、アップグレードスクリプトおよびファイルが含まれます。
- /services ディレクトリには、Access Manager サービスで使用されるディレクトリが含まれます。

/web-src ディレクトリ

/web-src ディレクトリには、Web コンテナ上での Access Manager J2EE Web アプリケーションの配備先サブディレクトリが含まれます。次のサブディレクトリが含まれます。

- applications/ ディレクトリ。Access Manager コンソールを配備します。index.html ファイルと各種サブディレクトリが含まれます。/console ディレクトリには、各種コンソール関連のサブディレクトリが含まれます。
- /common ディレクトリ (およびサブディレクトリ)。Access Manager Liberty Common Domain コンポーネントを配備します。
- /password ディレクトリ (およびサブディレクトリ) 。Access Manager Password Synchronization コンポーネントを配備します。index.html ファイルと、各種サブディレクトリが含まれます。
- /services ディレクトリ (およびサブディレクトリ) 。Access Manager Core Service を配備します。index.html ファイルと、各種サブディレクトリが含まれます。

設定 (/config) ディレクトリ

設定 (/config) ディレクトリのデフォルトの場所は、Access Manager のインストール先プラットフォームによって異なります。

- Solaris システム: /etc/opt/SUNWam
- Linux システム: /etc/opt/sun/identity

/config ディレクトリには、次のような設定ファイル、XML ファイル、および LDIF ファイルが含まれます。

- .version ファイルには、Access Manager の現在のバージョンが記述されています。
- AMConfig.properties、SSOConfig.properties、および LogConfig.properties の各ファイルには、Access Manager 設定属性が含まれます。

- `serverconfig.xml` ファイルは、Directory Server 用の Access Manager の設定情報を提供します。
- `/ldif` サブディレクトリには、Access Manager のインストール時に Directory Server データストアを生成するために必要な LDIF ファイルが含まれます。次に例を示します。
  - インストール時に、`ds_remote_schema.ldif` ファイルは、Access Manager のデータを Directory Server に格納するために必要な Access Manager 固有の LDAP スキーマオブジェクトクラスおよび属性 (`iplanet-am-managed-people-container` など) をロードします。`sunone_schema2.ldif` ファイルは、Access Manager LDAP 固有のスキーマオブジェクトクラスおよび属性をロードします。
  - アンインストール時に、`ds_remote_schema_uninstall.ldif` ファイルは、Access Manager の LDAP スキーマオブジェクトおよび属性を Directory Server から削除します。
- `/ums` サブディレクトリには、次のような XML ファイルが含まれます。
  - `ums.xml` ファイルは、Access Manager によって管理されるオブジェクトの LDAP 設定情報を含むテンプレートセットを提供します。
  - `/xml` サブディレクトリには、次のファイルが含まれます。
    - `amserveradmin` スクリプトは、Access Manager サービスをロードします。
    - 通常、XML ファイルは設定に使用されません。これらの XML ファイルを修正した場合は、Directory Server データストアに手動で再ロードする必要があります。サーバーでの変更は一切、これらのファイルと同期しません。このディレクトリの XML ファイルについては、『Sun Java System Access Manager 7 2005Q4 Developer's Guide』を参照してください。

付録 B

パスワード暗号化キーの変更

Sun Java™ System Access Manager は、パスワード暗号化キーを使用してユーザーパスワードを暗号化します。すべての Access Manager サブコンポーネントが、同じパスワード暗号化キー値を使用する必要があります。Access Manager の複数のインスタンスを別々のホストサーバーに配備することを計画している場合は、すべてのインスタンスに同じパスワード暗号化キーを使用する必要があります。

インストールに関する考慮事項

Access Manager の 1 番目のインスタンスをインストールすると、Java Enterprise System インストーラによって、デフォルトのパスワード暗号化キー文字列が生成されます。このデフォルトの値を受け入れるか、または J2EE 乱数発生関数によって生成された別の値を指定することもできます。インストーラは、パスワード暗号化キー値を `AMConfig.properties` ファイルの `am.encryption.pwd` 属性に格納します。

パスワード暗号化キーの値を指定する場合は、その文字列の長さを少なくとも 12 文字にする必要があります。

Access Manager の複数インスタンスを配備するには、1 番目のインスタンスのインストール後に、`am.encryption.pwd` 属性からパスワード暗号化キー値を保存します。次に、追加のインスタンスを配備するときは、このキー値を使用して値を設定します。

- Java ES インストーラを実行する場合は、この値を「Access Manager: 管理」パネルの「パスワードの暗号鍵」フィールドにコピーします。
- `amconfig` スクリプトを実行する場合は、`amsamplesilent` 設定ファイル (または、このファイルのコピー) の `AM_ENC_PWD` 変数をこの値に設定します。

次のシナリオでは、パスワード暗号化キーを取得して、変更する必要がある理由について説明します。これらのシナリオでは、各 Access Manager インスタンスが同じ Directory Server を使用しています。

- Access Manager の複数サーバーインストールを実行しており、1 番目の Access Manager インスタンスをインストールしたときにパスワード暗号化キーを保存しなかった場合は、追加のインスタンスを配備するときに、使用するキーを取得する必要があります。
- 追加の Access Manager インスタンスをすでに配備しており、そのインスタンスが 1 番目の Access Manager インスタンスとは別のパスワード暗号化キーを使用している場合は、その暗号化キーの値が 1 番目のインスタンスと一致するように変更する必要があります。

パスワード暗号化キーを変更した場合に必要なほかの変更点

パスワードとパスワード暗号化キーは、配備全体で一貫している必要があります。ある場所やインスタンスでパスワードを変更したら、ほかのすべての場所やインスタンスのパスワードも更新する必要があります。

`serverconfig.xml` ファイルに、暗号化されたユーザーパスワードが収められ、`<DirPassword>` 要素によって識別できます。次に例を示します。

```
<DirPassword>
Adfhfghghfhdghdfhdfghrteutru
</DirPassword>
```

`serverconfig.xml` の `puser` および `dsameuser` パスワードは、`AMConfig.properties` ファイルの `am.encryption.pwd` に定義されたパスワード暗号化キーを使用して暗号化されます。パスワード暗号化キーを変更した場合は、`ampassword` ユーティリティーを使用して、`serverconfig.xml` ファイルのこれらのパスワードも再暗号化する必要があります。

`ampassword` ユーティリティーについては、『Sun Java System Access Manager 7 2005Q4 管理ガイド』を参照してください。

キー値の変更

パスワード暗号化キーを変更するには、次の手順に従います。

1. 1 番目の Access Manager インスタンスがインストールされているホストサーバーにスーパーユーザー (`root`) としてログインするか、スーパーユーザーになります。
2. 1 番目の Access Manager インスタンスの `AMConfig.properties` ファイルで、次の属性の値を取得し、保存します。
   - パスワード暗号化キー: `am.encryption.pwd`
   - 共有シークレット: `com.iplanet.am.service.secret`

   `AMConfig.properties` ファイルは次のディレクトリにインストールされます。
   - Solaris システム: `/etc/opt/SUNWam/config`

- Linux システム: /etc/opt/sun/identity/config

3. 2 番目の Access Manager インスタンスが配備されているサーバーにスーパーユーザー (root) としてログインするか、スーパーユーザーになります。
4. 念のため、/config ディレクトリの AMConfig.properties ファイルと serverconfig.xml ファイルをバックアップします。
5. 2 番目の Access Manager インスタンスの Web コンテナを停止します。たとえば、Solaris システムで、Web Server が Web コンテナの場合は次のようになります。

```
# cd /opt/SUNWwbsvr/https-host2-name
#./stop
```

6. AMConfig.properties ファイルを編集して、am.encryption.pwd と com.iplanet.am.service.secret の値を、手順 2 で 1 番目の Access Manager インスタンスから保存した値に置き換えます。
7. am.encryption.pwd で定義されている暗号化キーを変更したため、ampassword ユーティリティーを実行して、serverconfig.xml ファイルのパスワードを再暗号化し、置き換える必要があります。serverconfig.xml のパスワードは、<DirPassword> 要素によって識別されます。次の場合について考慮します。

   パスワードが同じ場合: puser と dsameuser のパスワードが serverconfig.xml の amadmin パスワードと同じ場合は、ampassword を実行して amadmin パスワードを再暗号化します。Solaris システムの場合は次のようになります。

```
# cd /opt/SUNWam/bin
# ./ampassword --encrypt password
```

   ここで *password* は、1 番目のインスタンスをインストールしたときに amadmin に使用したパスワードです。ampassword の出力 (新しい暗号化パスワード) を使用して、2 番目のインスタンスの serverconfig.xml の 2 つのパスワードを置き換えます。

   パスワードが異なる場合: puser と dsameuser のパスワードが serverconfig.xml の amadmin パスワードと異なる場合は、ampassword を実行して各パスワード (type="proxy" と type="admin") を再暗号化します。

   ampassword の出力 (新しい暗号化パスワード) を使用して、2 番目のインスタンスの serverconfig.xml の puser と dsameuser のパスワードを置き換えます。
8. 2 番目の Access Manager インスタンスの Web コンテナを再起動します。たとえば、Solaris システムで、Web Server が Web コンテナの場合は次のようになります。

```
# cd /opt/SUNWwbsvr/https-host2-name
# ./start
```

9. 配備されている Access Manager のほかの追加インスタンスについて、手順 3 から手順 8 を繰り返します。

索引