Sun Java™ System

Access Manager 6
配備計画ガイド

2005Q1

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

Part No: 819-1942

目 次

図目次

表目次

コード例一覧

はじめに

本書『Sun Java™ System Access Manager 配備計画ガイド』では、Sun Java System Access Manager (以前の Sun™ Java System Identity Server) を計画し配備するための技術情報と簡単なシナリオを提供しています。また、アイデンティティ管理および配備チームの選択方法に関する一般的な情報も提供します。

この序文は、次の節で構成されています。

対象読者

この『配備計画ガイド』は、Sun Java System のサーバーおよびソフトウェアを使用した統合アイデンティティサービスおよび Web アクセスプラットフォームを実装する IT 管理者向けに書かれています。管理者は次の技術に精通していることが必要です。

- 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

- Solaris™ または Linux オペレーティングシステムの概念
- LDAP (Lightweight Directory Access Protocol) ディレクトリサーバーの概念
- Java™ テクノロジ
- JavaServer Pages™ (JSP) テクノロジ
- HTTP (HyperText Transfer Protocol)
- HTML (HyperText Markup Language)
- XML (eXtensible Markup Language)

お読みになる前に

Access Manager は、ネットワークまたはインターネット環境に分散するエンタープライズ アプリケーションをサポートするソフトウェアインフラストラクチャである Sun Java Enterprise System のコンポーネントです。Sun Java Enterprise System で提供されているマニュアルに精通することをお勧めします。このマニュアルは次の URL からオンラインでアクセスできます。

http://docs.sun.com/prod/entsys.05q1

Sun Java System Directory Server は Access Manager 配備のデータストアとして使われるため、Directory Server のマニュアルに精通することをお勧めいたします。このマニュアルは次の URL からオンラインでアクセスできます。

http://docs.sun.com/coll/DirectoryServer_05q1

内容の紹介

次の表に、本書の内容をまとめています。

表 1 『Access Manager 配備計画ガイド』の内容

章または付録	説明
第 1 章「はじめに」	Sun Java™ System Access Manager の概要を説明します。
第 2 章「配備の計画」	配備の計画方法を説明します。
第 3 章「Access Manager のアーキテクチャ」	Access Manager のコアアーキテクチャと関連するインタフェースについて、およびサービスのアーキテクチャ上の詳細について説明します。
第 4 章「配備前の考慮事項」	配備に関連する高度な技術的概要を説明します。

表 1　『Access Manager 配備計画ガイド』の内容 (続き)

章または付録	説明
第 5 章「配備シナリオ」	Access Manager の配備のさまざまなシナリオについて説明します。
付録 A「インストールされる製品のレイアウト」	Access Manager のインストール後のディレクトリレイアウトについて説明します。
付録 B「ユーザーセッションのライフサイクル」	Access Manager コンポーネントのプロトコルのやり取りを追跡して、セッションのライフサイクルについて説明します。
付録 C「Active Directory に対する認証」	Microsoft® Active Directory® に対して認証する方法を説明します。
付録 D「chroot 環境のインストール」	chroot 環境について説明します。
付録 E「ロードバランサの設定」	Access Manager をロードバランサで動作するように設定する方法を説明します。
付録 F「RADIUS サーバーに対する認証」	RADIUS (Remote Authentication Dial-In User Service) サーバーでユーザーを認証する方法について説明します。
用語集	最新の『Sun Java™ Enterprise System 用語集』のリンクを紹介します。

表記上の規則

この節の各表に、本書の表記規則を説明します。

表記上の規則

次の表に、本書の表記上の変更を説明します。

表 2　表記上の規則

書体	意味	例
AaBbCc123 (モノスペース)	API およびプログラミング言語の要素、HTML タグ、Web サイト URL、コマンド名、ファイル名、ディレクトリ パス名、画面のコンピュータ出力、サンプルコード。	`.login` ファイルを編集します。 すべてのファイルを表示するには、`ls -a` を使用します。 `%` メールを受信しました。

表 2　表記上の規則 (続き)

書体	意味	例
AaBbCc123 (モノスペースボールド)	画面のコンピュータ出力と対比させる場合に、ユーザーが入力したテキスト。	% **su** Password:
AaBbCc123 (イタリック体)	コマンドまたはパス名の中の実際の名前や値に置き換えられるプレースホルダ。	このファイルは *install-dir*/bin ディレクトリにあります。

記号

次の表に、本書の記号の表記規則を説明します。

表 3　記号の表記規則

記号	説明	例	意味
[ ]	オプションのコマンドオプションを囲みます。	ls [-l]	-l オプションは必須ではありません。
{ | }	必須コマンドオプションの一連の選択肢を囲みます。	-d {y|n}	-d オプションには y 引数または n 引数のどちらかを使う必要があります。
-	同時の複数のキー入力を結合します。	Control-A	Control キーを押しながら、A キーを押します。
+	連続した複数のキー入力を結合します。	Ctrl+A+N	Control キーを押し、それを離してから、続きのキーを押します。
>	グラフィカルユーザーインタフェースのメニュー項目を示します。	「ファイル」-「新規作成」-「テンプレート」	「ファイル」メニューから、「新規作成」を選択します。「新規作成」サブメニューから、「テンプレート」を選択します。

デフォルトのパスとファイル名

次の表に、本書のデフォルトのパスとファイル名の表記規則を説明します。

表 4　デフォルトのパスとファイル名

用語	説明
AccessManager-base	Access Manager のベースインストールディレクトリを表します。Access Manager のデフォルトのベースインストールと製品ディレクトリは、使用するプラットフォームにより異なります。 Solaris™ システム : `/opt/SUNWam` Linux システム : `/opt/sun/identity`
DirectoryServer-base	Sun Java System Directory Server のベースインストールディレクトリを表します。特定のパス名については、製品のマニュアルを参照してください。
ApplicationServer-base	Sun Java System Application Server のベースインストールディレクトリを表します。特定のパス名については、製品のマニュアルを参照してください。
WebServer-base	Sun Java System Web Server のベースインストールディレクトリを表します。特定のパス名については、製品のマニュアルを参照してください。

シェルプロンプト

次の表に、本書で使用されているシェルプロンプトを説明します。

表 5　シェルプロンプト

シェル	プロンプト
UMIX または Linux の C シェル	*machine-name*%
UMIX または Linux の C シェルスーパーユーザー	*machine-name*#
UMIX または Linux の Bourne シェルおよび Korn シェル	$
UMIX または Linux の Bourne シェルおよび Korn シェルスーパーユーザー	#
Windows コマンド行	C:¥

関連マニュアル

Sun のテクニカルマニュアルにオンラインでアクセスするには、http://docs.sun.com を参照してください。

マニュアルのアーカイブを参照したり、特定のマニュアルのタイトル、パーツ番号、サブジェクトで検索したりすることができます。

このマニュアルセットのマニュアル

表 6　Access Manager 6 2005Q1 のマニュアルセット

マニュアルのタイトル	説明
『Technical Overview』 http://docs.sun.com/doc/817-7643	Access Manager コンポーネントの高レベルの概要。コンポーネントを統合してアイデンティティ管理を行い、企業資産と Web ベースアプリケーションを保護する方法について説明しています。Access Manager の基本的な概念と用語についても説明しています。
『配備計画ガイド』(本書) http://docs.sun.com/doc/819-1942?l=ja	既存の情報技術インフラストラクチャ内に配備する方法について説明します。
『管理ガイド』 http://docs.sun.com/doc/819-1938?l=ja	Access Manager コンソールの使用方法と、コマンド行によるユーザー管理およびデータサービスの方法について説明します。
『Migration Guide』 http://docs.sun.com/doc/817-7645	既存のデータおよび Sun Java System 製品の配備を最新バージョンの Access Manager に移行する方法について説明します。Access Manager とその他の製品のインストールおよびアップグレード方法については、『Sun Java Enterprise System 2005Q1 インストールガイド』を参照してください。
『Performance Tuning Guide』 http://docs.sun.com/doc/817-7646	Access Manager とその関連コンポーネントのチューニング方法について説明します。
『Federation Management Guide』 http://docs.sun.com/doc/817-7648	Liberty Alliance Project に基づく、連携管理について説明します。
『Developer's Guide』 http://docs.sun.com/doc/817-7649	Access Manager のカスタマイズ方法とその機能を組織の現在の技術インフラストラクチャに統合する方法について説明します。製品とその API のプログラム的側面に関する詳細も含まれています。

表 6　Access Manager 6 2005Q1 のマニュアルセット (続き)

マニュアルのタイトル	説明
『Developer's Reference』 http://docs.sun.com/doc/817-7650	Access Manager 公開 C API を構成するデータタイプ、構造、および機能について説明します。
『リリースノート』 http://docs.sun.com/doc/819-1946?l=ja	製品のリリース後、ご利用になれます。このリリースの最新情報、既知の問題、制限事項、インストールに関する注意事項、ソフトウェアまたはマニュアルに関する問題の報告方法などを提供します。

Access Manager ポリシーエージェントのマニュアル

Access Manager のポリシーエージェントのマニュアルは、次のマニュアル Web サイトにあります。

http://docs.sun.com/coll/S1_IdServPolicyAgent_21

Access Manager のポリシーエージェントは、サーバー製品と異なるスケジュールで提供されます。このため、ポリシーエージェントのマニュアルセットは、Access Manager のマニュアルのコアセットとは別個に提供されます。マニュアルセットに含まれるタイトルは、次のとおりです。

- 『Policy Agents For Web and Proxy Servers Guide』: さまざまな Web およびプロキシサーバーで Access Manager ポリシーエージェントをインストールおよび設定する方法を説明します。また、トラブルシューティングや、各エージェントに固有の情報についても説明します。
- 『J2EE Policy Agents Guide』: さまざまなホスト J2EE アプリケーションを保護可能な Access Manager ポリシーエージェントのインストールおよび設定方法について説明します。また、トラブルシューティングや、各エージェントに固有の情報についても説明します。
- 『リリースノート』は、エージェントセットのリリース後、オンラインでご利用になれます。『リリースノート』には、このリリースの最新情報、既知の問題、制限事項、インストールに関する注意事項、ソフトウェアまたはマニュアルに関する問題の報告方法などを提供します。

その他のサーバーのマニュアル

その他のサーバーのマニュアルについては、次を参照してください。

- Directory Server のマニュアル
  http://docs.sun.com/coll/DirectoryServer_05q1
- Web Server のマニュアル
  http://docs.sun.com/coll/WebServer_05q1
- Application Server のマニュアル
  http://docs.sun.com/coll/ApplicationServer8_ee_04q4
- Web Proxy Server のマニュアル
  http://docs.sun.com/prod/s1.webproxys#hic

Sun リソースへのオンライン アクセス

製品のダウンロード、プロフェッショナルサービス、パッチとサポート、追加の開発者向け情報については、次を参照してください。

ダウンロードセンター
http://wwws.sun.com/software/download/

Sun Java System サービススイート
http://www.sun.com/service/sunjavasystem/sjsservicessuite.html

Sun エンタープライズサービスによる Solaris のパッチとサポート
http://sunsolve.sun.com/

開発者用情報
http://developers.sun.com/prodtech/index.html

Sun テクニカルサポートへのお問い合わせ

製品マニュアルで解決しない本製品に関する技術的な質問がある場合は、次を参照してください。

http://www.sun.com/service/contacting

サードパーティーの関連 Web サイト

Sun は、本書に記載されたサードパーティの 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。

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 6 2005Q1 配備計画ガイド』で、パーツ番号は 819-1942 です。

ご意見、ご要望をお寄せください

第 1 章

はじめに

Sun Java™ System Access Manager (以前の Sun™ Java System Identity Server) は、組織に対し、Web ベースのサービスや Web ベース以外のアプリケーションを利用する顧客、従業員、およびパートナーのデジタルアイデンティティの運営プロセスを管理するインフラストラクチャを提供します。これらのリソースは、内部および外部の広範なコンピューティングネットワーク上に分散する可能性があるため、アクセスを管理する属性、ポリシー、および制御が定義されています。

この章では、Access Manager を導入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。

アイデンティティ管理とは

現在、各企業では、日々の作業の管理の効率化のため、高度な情報技術のインフラストラクチャを運用しています。このインフラストラクチャに不可欠な要素には、次のようなものがあります。

- 異なるオペレーティングシステムを実行するネットワークサーバー
- 情報データストア
- 人事、給与、および契約管理システム
- アカウンティング、サプライチェーン管理、およびリソース計画に対応した事業分野別アプリケーション
- 販売、マーケティング、顧客サービス、現場サポート、その他のクライアント関連機能を統合した顧客関係管理 (CRM) システム

- ショッピングおよびセキュリティ保護されたクレジットカード取引に対応した電子商取引用アプリケーション

これらの各要素は個別に配備されるため、システムごとにユーザーを追跡して、実行可能および実行不可能な操作を制御します。通常、この追跡処理には、個人のプロファイル、認証情報、およびアクセス制御などのアイデンティティデータの管理が含まれます。アイデンティティ管理を利用することで、この複製データ (矛盾していることが多い) の管理が簡単になります。

アイデンティティ管理のインフラストラクチャ

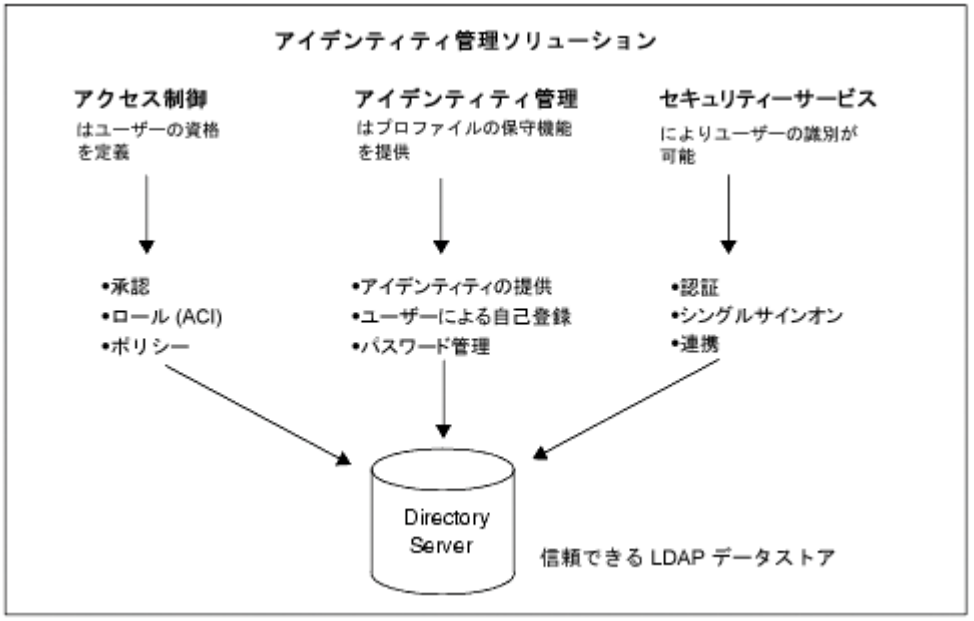
アイデンティティ管理システムの目的は、企業内のすべてのユーザーを単一のインフラストラクチャにより管理することです。単一のアイデンティティ管理システムを使用することで、繰り返しを避けて一貫性を維持することができるため、ユーザープロファイルの管理が簡単になります。さらに単一のシステムにより、アイデンティティ管理プロセスの合理化、簡略化、および自動化が可能になります。以下に、アイデンティティ管理システムの構成要素を示します。

- アイデンティティ管理の提供するインフラストラクチャは、アイデンティティおよび対応する属性、証明情報、資格の作成および管理をサポートします。この機能には、以下が含まれます。
  - アイデンティティの提供 (プロファイルの作成、変更、および削除)
  - ユーザーの自己登録
  - パスワード管理およびパスワードリセット
- セキュリティーサービスを使用することで、ユーザーのアイデンティティがネットワーク上で一貫したものになります。この機能には、以下が含まれます。
  - 認証 (当人のアイデンティティを証明する)
  - シングルサインオン機能 (一度の認証で多数のリソースへのアクセスを可能にする)
- アクセス制御は、ユーザーの資格 (さまざまなリソースをどのようにまたいつ使用できるか) を定義します。多くの場合、これらの制御によりユーザーの組織内でのロールが指定されます。集中化されたポリシーおよびロールを作成して適用することにより、組織は顧客、従業員、およびパートナーの責任を委譲できます。
  - 認証 (アイデンティティが要求されたアクションを実行できるかどうかを決定する)
  - ポリシー (保護されたリソースへのアクセス承認を定義する)
  - アクセス制御命令 (アイデンティティデータへのアクセス承認を定義する)
- 連携サービスは、独立した情報システム間の認証および承認を提供します。

- 企業の LDAP ディレクトリは、すべてのアイデンティティ情報およびアイデンティティ管理システム自体の設定情報の信頼できるデータストアとして動作します。

これらの構成要素の詳細については、図 1-1 を参照してください。

図 1-1 アイデンティティ管理ソリューションの構成要素



アイデンティティプロファイルのライフサイクル

一般的なアイデンティティプロファイルのライフサイクルについて考慮すると、アイデンティティ管理を行う際の課題を概観できます。組織内のアイデンティティには、次の 3 つの段階が存在します。

1. プロファイルの作成

   ユーザーが組織に参加すると、アイデンティティプロファイルが作成されます。プロファイルには、個人情報、雇用データ、パスワード情報、および定義済みのアクセス権が含まれます。

2. プロファイルの管理

   セットアップが終了したら、プロファイルを管理する必要があります。これには、プロファイルデータの変更、リソースアクセスポリシーの管理、アクセス制御命令の更新が含まれます。

3. プロファイルの無効化

   ユーザーがログアウトすると、プロファイルにそのことを示すフラグを付け、システムリソースへのアクセスを無効にする必要があります。

Sun Java System Access Manager

Sun Java System Access Manager は、アクセス管理、連携、およびアイデンティティ管理をサポートする Web サービスを統合した標準ベースのミドルウェアのパッケージです。このため、Access Manager は総合的なアイデンティティ管理ソリューションとして、ユーザープロファイルの作成および管理機能を、セキュリティプロセス、アクセス管理ツールおよびデータ格納用ディレクトリに統合します。これらの機能を使用すると、リソースおよび情報が保護された総合的なシステムを組織に配備するとともに、Web ベースのアプリケーションをセキュリティ保護された方法で配信できます。

アクセス管理

アクセス管理は、独自およびアプリケーション固有の認証および承認方法に代わる、共通の認証および承認インフラストラクチャを提供します。組織では、中心となる 1 箇所の管理機能から、複数のサービスに対してポリシーベースのアクセス制御を提供できます。Access Manager の一連のアクセス管理サービスは、次の機能を提供します。

シングルサインオン (SSO)

シングルサインオン機能は、一度のユーザー認証で、複数のリソースへのアクセスを可能にします。Access Manager は、Web ベースのアプリケーションで SSO をサポートします。また、Web ベース以外のアプリケーションに SSO 機能を統合するためのプログラミング用インタフェースも提供します。

プラグイン可能な認証

JAAS (Javaa Authentication and Authorization Services) ベースの認証フレームワークは、LDAP、RADIUS (Remote Authentication Dial-In User Service)、X.509 デジタル証明書、SecureID®、SafeWord®、UNIX® (PAM ベース)、Windows® NT、HTTP 基本認証、匿名、および自己登録を含む、さまざまなプラグイン可能認証モジュールをサポートします。

このフレームワークを利用することで、提供される認証サービスプロバイダインタフェース (SPI) を使用して、カスタム認証モジュールも開発できます。認証を設定することで、同一システム内の多様な組織、ロール、またはユーザーのニーズを同時にサポートしたり、複数の要因が連鎖した設定をサポートしたりできます。

マルチレベルの認証を使用すると、データやサービスの特性に基づいて求められるさまざまな認証レベルにリソースを割り当てることができます。認証サービスには、Web ベース、Java、C、および XML インタフェースからアクセスできます。

ポリシーの評価

ポリシーサービスを使用すると、さまざまなロールやグループ化メカニズムにマッピング可能なアクセス管理ルールを集中的に設定および評価できます。IP アドレス、日付と時刻、カスタム条件などのポリシー制約は、実行時にポリシーに適用して評価できます。

連携管理

インターネットは、ビジネス、コミュニティ、および個人のやり取りのための主要な手段に急速になりつつあるため、ユーザーがさまざまなアカウントアイデンティティを集約させるシステムを構築し、単一のネットワークアイデンティティを使用できるようにする必要が生じてきました。このシステムをアイデンティティ連携といいます。アイデンティティ連携を使用すると、複数のインターネットサービスプロバイダのローカルアイデンティティを関連付けたり、連結したり、またはバインドしたりすることができます。ネットワークアイデンティティを使用すると、あるサービスプロバイダのサイトに一度ログインすれば、アイデンティティを再認証または再確立しなくても提携先のサイトに移動できます。

Access Manager は、Liberty 1.1 および SAML 1.0 の完全な実装を提供します。これには、完全なプロファイル実装および、カスタム統合化に対応した SDK サポートが含まれます。Liberty アイデンティティおよびサービスプロバイダの複数ホスティングも含みます。

Liberty Alliance Project

連携管理は、認証ドメインおよびプロバイダに関するメタデータを表示、管理、および設定する方法を備えています。アイデンティティ連携を実現するために策定された Liberty Alliance Project は、20 億を超える顧客と広範な業種から参加した 138 のメンバー企業により構成されています。その使命は、消費者とビジネスユーザーのシングルサインオンを可能にする、連携されたネットワークアイデンティティソリューションを提供およびサポートすることにより、断片化するアイデンティティの問題を解決することです。

このため、Liberty 対応のアプリケーションは、ユーザーアカウントを別の Liberty 対応アプリケーションのユーザーアカウントと連携 (リンク) させて、2 つのアプリケーション間のシングルサインオンを実現できます。Access Manager は、Liberty Alliance Project の仕様を実装しています。

SAML (Security Assertion Markup Language)

SAML は、ビジネス間インフラストラクチャを実現するのに重要です。アプリケーションは、Access Manager に統合された SAML API を使用して、セキュリティ情報を交換し、信頼される他のアプリケーションとのビジネス取引を実行できます。エンドユーザーは、Web ブラウザを使用して Access Manager への認証を行い、サイト内転送 URL 経由で信頼できるサイトの外部 URL にシームレスにアクセスできます。開発者は、アプリケーション内で SAML API を使用し、信頼できる外部アプリケーション間で認証、承認、および属性情報を交換できます。

アイデンティティ管理

アイデンティティ管理は、ユーザー提供、ポリシー設定、サービス管理を可能にする拡張可能なブラウザベースのインタフェースを提供します。Access Manager コンソールを使用すると、単一のインタフェースからアイデンティティ管理を集中的に実行できます。また、ローカルグループマネージャ、外部パートナー、さらにエンドユーザーにまでも管理を委任できます。

ユーザープロファイル管理

簡潔にまとめると、ユーザープロファイル管理では、アイデンティティプロファイルの作成と削除を行います。ただし、これには、プロファイルの管理を事情に通じた管理者に委任することや、セルフサービスコンポーネント (ユーザーはこれを使ってサービスやアプリケーションを利用できる) を提供すること、新規ユーザーアカウントを作成してそのプロファイルを管理すること (パスワードの変更、自宅住所の更新など) が含まれます。

ポリシー設定

ポリシー設定は、アクセス承認時に評価されるルールの定義です。委任を実行すると、最上位レベルの管理者がポリシーの設定および管理を組織のあらゆるレベルの個人に分散できます。これにより、ポリシーの設定および管理を、リソースに対する権限を保持するユーザーに確実に託すことができます。

サービス管理

サービス管理を使用すると、Web サービスおよび対応する属性を設定、登録、および管理できます。Access Manager では、独自の管理に使用するサービス用インタフェースも提供されます。

監査

管理者は、高度な設定が可能なログ機能を使用して、ユーザーのアクティビティ、トラフィックパターン、認証および承認違反に関する詳細なレポートを作成できます。これらの機能を使用して、リソースアクセスに対するセキュリティレベルの監査も実行できます。MAC (Message Authentication Code) およびデジタル署名ベースのログセキュリティは、ログまたは監査記録に対するいかなる改ざんも検出します。デバッグ機能も有効にできます。

ポリシーエージェント

Access Manager 内のアクセス制御は、ポリシーエージェントを使用して行われます。ポリシーエージェントは、指定された Web サーバー、アプリケーションサーバー、およびプロキシサーバー上のコンテンツを不正な侵入から保護します。Access Manager では、Web およびプロキシサーバーを URL レベルで保護するポリシーエージェント、および Java テクノロジに対応したアプリケーションサーバーへのアクセスを行う Java™ 2 Platform, Enterprise Edition (J2EE) ポリシーエージェントがサポートされます。詳細については、35 ページの「ポリシーエージェントを使用した Access Manager の統合」を参照してください。

Access Manager コンソール

Access Manager コンソールは、Access Manager の配備全体で設定された、アイデンティティ、サービス、およびポリシーの作成、管理、および監視用のブラウザベースインフラストラクチャです。これは、機能的な Web アプリケーションを作成する開発者を支援する J2EE フレームワークである、Sun Java System Application Framework を使用して構築されます。HTML ページの外観を定義するために、XML ファイル、JSP (JavaServer Pages™)、CSS (Cascading Style Sheets) が使用されます。

プログラミング用インタフェース

非グラフィカルインタフェースには、Access Manager の拡張およびカスタマイズに使用される API、SPI、およびコマンド行ツールが含まれます。このインタフェースを使用すると、その他のアプリケーションから各機能にアクセスできるようになります。API および SPI の詳細は、61 ページの「Access Manager SDK」および『Access Manager Developer's Guide』を参照してください。コマンド行ツールの詳細は、『Access Manager 管理ガイド』を参照してください。

Sun Java System Directory Server

Java System Directory Server は、アイデンティティ、ポリシー、設定およびサービス情報を格納する統合化されたデータリポジトリとして機能します。

Access Manager の配備

Access Manager は、オープンな標準に準拠したプラットフォームに合わせて設計されています。このプラットフォームは、認証、承認、シングルサインオン、ポリシー、アイデンティティ、および管理機能を既存のインフラストラクチャに統合する際に使用できます。その機能は、Web コンテナの Java 仮想マシン (JVM) 内部で動作し、API およびさまざまなサーバーフレームワークにアクセス可能な Java サーブレット、JavaBeans™、および JSP の集合体として提供されます。Access Manager を企業のインフラストラクチャに統合することで、以下のタスクを達成できます。

- 柔軟性に欠ける独自ユーティリティを排除する
- 複数の Web およびアプリケーションサービス間のセキュリティ保護された認証、アクセス制御、および監査を実現する
- アクセス制御命令 (ACI) のレベルを設定して、集中化されたアイデンティティ管理を委任機能とともに実装する。これにより、以下の処理が組織内で可能になる

  - 内部アイデンティティ管理を IT 担当以外の従業員に委任する
  - 外部アイデンティティ管理をパートナーまたはサプライヤーの従業員に委任する
- 集中化されたポリシーフレームワークを設定して、実行中のアプリケーションや新たに配備されたサービスに認証機能を提供する
- 連携管理を Liberty Alliance 仕様 v.1.1 および SAML のサポートに統合する

ポリシーエージェントを使用した Access Manager の統合

Access Manager は総合的なアイデンティティ管理システムですが、それぞれの組織では何らかのアイデンティティ管理システムをすでに実装している場合があります。たとえば、ディレクトリサーバーや Web コンテナをすでに配備している場合が考えられます。Access Manager と他のシステムとの相互運用を可能にするには、組織の要件を満たすエージェントを利用できる場合は、ポリシーエージェントをダウンロードして保護されたサーバーにインストールできます。

Access Manager の各リリースと並行して、新規ポリシーエージェントの開発およびリリースが行われています。たとえば、Apache Webserver、BEA WebLogic、IBM HTTP Server、IBM WebSphere、Lotus Domino、Microsoft IIS、および PeopleSoft のさまざまなリリースに対応したポリシーエージェントが利用できます。

ポリシーエージェントに対応したオペレーティングシステムには、Solaris™ Operating System (SPARC® Platform Edition および x86 Platform Edition)、Red Hat™ Linux、HP-UX 11.x、および Windows 2000 があります。

注　上記の各製品用のポリシーエージェントが、これらのすべてのオペレーティングシステムに用意されているわけではありません。現在利用可能なポリシーエージェントの一覧については、次の Web サイトで Sun の「Web, Portal & Directory Servers Download Center」を調べて、要件を満たすポリシーエージェントが利用できるかどうかを確認してください。

`http://www.sun.com/software/download/inter_ecom.html`

配備ロードマップ

Access Manager の統合の成功には詳細な計画が重要です。これには、ハードウェア、配備中のアプリケーション、アイデンティティ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 の配備作業は、次の段階に分けることができます。

1. ビジネスの目的を識別
   例 :
   - 業務効率を改善
   - データのセキュリティを確保
   - 次の方法で生産性を保証
     - 組織内の範囲および関係を理解
     - ビジネスの目的のサポートに必要な行動変化を分析
2. 次の方法で高度なテクノロジ分析を開発し、ビジネスの目的に適用
   - テクノロジサービスを列挙
   - ビジネスの目的の達成に必要なツールを列挙
3. 次に例示する、テクノロジサービスの具体策を定義
   例 :
   - 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
   - アイデンティティ管理を使用して、パスワード同期およびアイデンティティ管理を実行
   - 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
4. 以下の要素に基づいて、イニシアチブに優先順位を設定
   - 統計的正確性
   - 予測可能性
   - 範囲
   - 費用
   - 影響
   - 複雑性
   - 動作
   - インフラストラクチャ
   - 利点
   - サポート

- 依存関係

配備計画ガイドの各章

「配備ロードマップ」で説明した各段階については、このマニュアルの以下の章で詳しく説明されています。

- 第 2 章「配備の計画」では、組織のアイデンティティ管理ソリューションの現状を評価し、将来の必要を明確にする方法 (目標および課題を含む) を定義します。
- 第 3 章「Access Manager のアーキテクチャ」では、Access Manager 製品の全コンポーネントのアーキテクチャに関する高度な概要を示します。
- 第 4 章「配備前の考慮事項」では、ハードウェア、データソース、および専門知識を含む、特定の要件を分析する方法を示します。
- 第 5 章「配備シナリオ」では、トポロジを計画し、アプリケーションを配備する簡単なシナリオについて説明します。

『Access Manager 配備計画ガイド』では、付録が追加されて説明が補足されました。以下にその内容を示します。

- 付録 A「インストールされる製品のレイアウト」では、Access Manager のインストール時に作成されるディレクトリおよびファイルについて説明します。
- 付録 B「ユーザーセッションのライフサイクル」では、複数の HTTP (HyperText Transfer Protocol) 要求間で行われる、ユーザーと Web アプリケーションとのやり取りの追跡に使用するセッションオブジェクトについて説明します。
- 付録 C「Active Directory に対する認証」では、Microsoft Active Directory でユーザーを認証する方法を説明します。
- 付録 F「RADIUS サーバーに対する認証」では、RADIUS (Remote Authentication Dial-In User Service) サーバーでユーザーを認証する方法について説明します。
- 付録 D「chroot 環境のインストール」では、chroot 環境に Access Manager をインストールして、悪意のあるプログラムが実際のルートファイルシステムにアクセスすることを防ぐ方法について説明します。

Access Manager に関連するマニュアル

Access Manager に関する追加情報については、以下のマニュアルを参照してください。

- インストール ― インストール方法については、『Sun Java Enterprise System 2005Q1 インストールガイド』を参照してください。
- 移行 ― 既存のデータの移行と以前のバージョンの Access Manager の更新についての詳細は、『Sun Java Enterprise System 2005Q1 インストールガイド』および『Access Manager Migration Guide』を参照してください。
- 管理 ― コンソールの使用方法および Access Manager の配備を管理する方法については、『Access Manager 管理ガイド』を参照してください。
- カスタマイズ ― アプリケーションのカスタマイズ方法については、『Access Manager Developer's Guide』を参照してください。

第 2 章

配備の計画

Sun Java™ System Access Manager は、複雑な分散型アイデンティティ管理システムであり、適切な配備によって、企業の組織境界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

アイデンティティ管理ソリューションには組織全体の多種多様なシステムが関係するため、Access Manager を適正に配備するには広範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枠組みに精通しておくことは重要です。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および 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。彼らは、チームリーダーおよび多

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。チームは、Sun プロフェッショナルサービスリソースおよび必要に応じて段階的に投入および廃止される LOB アプリケーション管理者で構成される仮想チームメンバーにより、しばしば増強されます。これは、現実のニーズを満たすかどうかは別として、ほぼ一般的な配備チームモデルを表したものです。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、典型的な Access Manager 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

通常、アイデンティティ管理を成功させるには、組織および行政上の境界を超えて配備を行う必要があります。これには、企業の決定権を持つ人物の承認および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。具体的な統計を入手することで、配備チームが経営陣のサポートを得るのに役立つ明確な ROI を定義できます。関連する他の問題には、以下が含まれます。

- 誰がアイデンティティ管理の配備からメリットを享受するか
- アイデンティティ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ゆくか

注 アイデンティティ管理の概念および Access Manager を配備することの価値を納得させることがしばしば必要になります。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の利点を経営陣に積極的に語り、統合化に関する需要を喚起したり、インフラストラクチャ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リーダーとして、1 人の人物を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。これは、テクニカルリーダー、プロジェクトマネージャ、および執行責任者などの間で論理的に分散されたロールとすることも可能です。このロールがどのように定義されるとしても、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

この担当者は、スケジュールの調整を行います。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。この担当者は、優れたコミュニケーション能力と政治的手腕を持っていなければなりません。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があります。

注 事業分野別の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

この人物は、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、および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仕方で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。Access Manager のアーキテクチャ、サービス、エージェント、および API に関する詳細な知識が求められます。

LOB アプリケーション管理者

これは、Access Manager ポリシーエージェントまたはポリシー実施ポイントをアプリケーションに統合する責務を担う人物です。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、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。通常は、最後に 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。このため、LOB アプリケーションに関する詳細な知識および制御能力を備えた、技術面の専門家でなければなりません。

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

この人物は、Access Manager の配備および保守を担当します。通常、専任のユーザーが割り当てられ、共通サービスの可用性の保証や、必要なインフラストラクチャの拡張、およびポリシーやロールの設定を行います。管理者は、ガイドラインを策定して統合化サポートを支援し、LOB アプリケーション管理者に技術サポートを提供します。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。管理者は、定義されている LDAP スキーマおよびアイデンティティデータへの追加や変更を受け入れたり統合したりするとともに、ディレクトリサービスの可用性管理を担当します。アイデンティティ管理インフラストラクチャをサポートするために、ディレクトリサービスの変更が必要になることがあります。

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、データセンターや、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、これらのさまざまな管理者間で明快なコミュニケーションを行う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することや、特定のネットワークを確立することが重要な場合があります。これらの担当者にプロジェクトの里程標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 および他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。

注	独立系ソフトウェアベンダーは、コンピュータの 1 つ以上のハードウェアタイプ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ソフトウェア製品を制作および販売します。プラットフォームを作成する企業 (IBM、Hewlett-Packard、Apple、Microsoft など) は、ISV の支援およびサポートを提供します。Sun Microsystems は、プラットフォームおよびソフトウェア製品を作成します。

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 プロフェッショナルサービスおよび他の ISV と契約を結んで、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働くことができる) と率直な討議を行うとともに、プロフェッショナルサービスを利用すると、投資を有効に活用し、配備を成功させる助けが得られます。

提携先のサードパーティ

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公平な話し合いの場を設定する助けになります。

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

目標を設定することにより、Access Manager の配備完了後のあるべき状態を定義できます。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、アイデンティティ管理ソリューションでは、セキュリティを拡張し、インフラストラクチャ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 (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を実装する
- アイデンティティ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
- アイデンティティプロファイルの迅速な終了により、セキュリティを改善する
- セキュリティ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タイムフレームを圧縮する
- 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および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、日常業務で使用す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、ビジネスプロセス、IT インフラストラクチャ、および仮想データに関する情報収集を含めることができます。

ビジネスプロセス

ビジネスプロセスとは、組織内の異なる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、次の手順を含められます。

- 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

IT インフラストラクチャには、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。

- 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、または重要性のあまり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

  これには、アプリケーションの配備先のハードウェアおよびそのオペレーティングシステムが含まれます。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 Sun Java System Access Manager アプリケーション自体が最低限含まれます。また、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。これには、セキュリティ上の理由で Access Manager のポリシーエージェントをインストール可能な企業のリソースが含まれます。

- 各部門が利用する Access Manager のサービス

  これには、Access Manager 内部に統合されたデフォルトおよびカスタムのサービスが含まれます。ロールおよびポリシーの戦略は、部門ごとに割り当ておよび定義を行う必要があります。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内に非互換性が存在するかどうか
- 現在のシステムに速度低下が見られるか。ダウンタイムはどの程度か
- アプリケーションは十分にセキュリティ保護されているか
- 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、48 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

仮想データは、Access Manager にアクセスするプロファイル用の、あらゆる状況で利用可能なデータです。このデータは、Access Manager からアクセス可能な設定でもあり、Access Manager によりセキュリティ保護されるデータでもあります。これには、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、および他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 が保護する対象

  Access Manager は、あらゆる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、外部の販売情報を閲覧するためにサインインする顧客よりも、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、機密情報および秘密データが含まれます。具体的には、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、および連携顧客プロファイルが含まれる場合があります。このデータは、Directory Server に格納される場合も格納されない場合もあります。

- 信頼すべきデータの入手元

  多くの場合、異なるデータタイプを定義する複数のスキーマが存在します。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る、または既存のデータが変更される頻度はどれほどか

詳細は、50 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

アイデンティティ管理サービスは、通常、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使用する、集中化された 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配備されています。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行う場合は、次の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、アイデンティティ管理インフラストラクチャに受け入れられるアプリケーション
- 現在、配備または再構築の初期プロセスにあるか、Access Manager 配備と一致する時系列を論理的に保持するアプリケーション

スプレッドシートを作成して、最も将来性の高いアプリケーションから取得し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。これにより、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は、メインフレームに由来する「グリーンスクリーン」アプリケーション (テキストベースのインタフェースを持つアプリケーション) です。この場合、メインフレームアプリケーションの再ファクタリング / アーキテクチャの待機中に、他のアプリケーションを統合するというメリットを享受できます。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。

注 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、以下が含まれます。

- 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する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

注 LOB アプリケーションも、サードパーティ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) を稼働させることができます。これらのアプリケーションは、Access Manager エージェントがサポートするプラットフォーム上に常に配備されるわけではありません。エージェントの可用性に基づいて、これらのアプリケーションの配備基準を評価し、組み込みのスケジュール設定を行ってください。カスタム作業が必要な場合には、通常、Java ベースのアプリケーションの方が統合がより容易です。

セキュリティモデル

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ため、配備の有力な候補になります。セキュリティ情報には、以下を含めることができます。

- 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
- 認証を外部で行うことは可能か。それは意味のあることか
- 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

アイデンティティのセッションライフサイクルは、認証アプリケーションの評価を行う上で重要な項目です。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理解しているかを確認してください。アプリケーションの統合化を行う際に参照できるよう、このプロセスを正確にドキュメント化してください。この項目の詳細については、付録 B 「ユーザーセッションのライフサイクル」を参照してください。

カスタマイズおよびブランド設定

アプリケーションの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を考慮する必要があります。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。カスタマイズおよびブランド設定要件を満たす場合、そのための時間もスケジュールに組み込む必要があります。このため、アプリケーションの評価にカスタマイズおよびブランド設定要件が含まれているかどうかを確認してください。

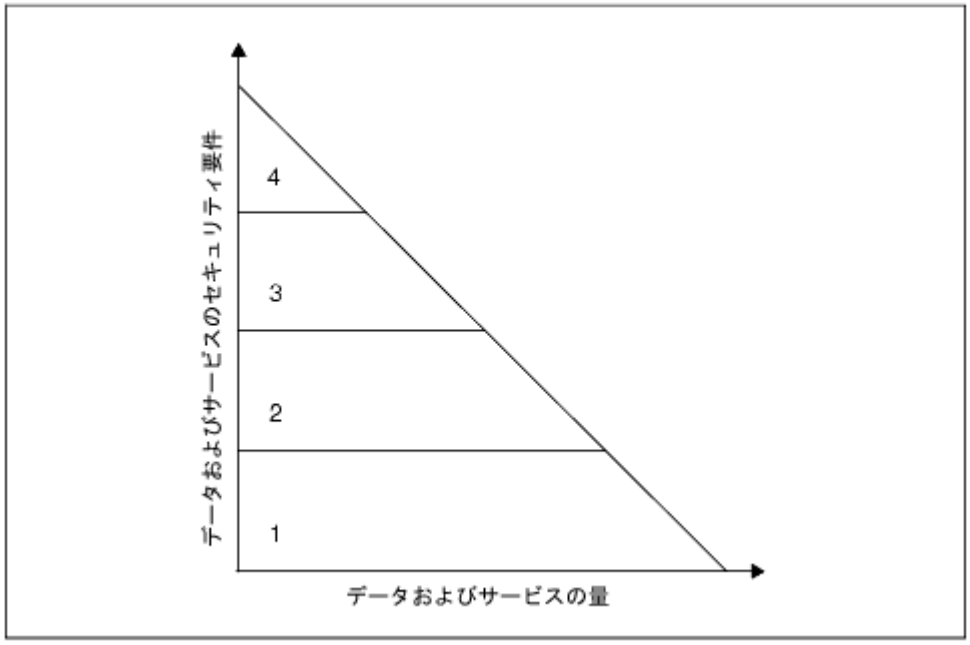
データの分類

アプリケーションを分析し、適切なレベルに分類したなら、アプリケーションにより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。アプリケーションの評価で収集された情報は、プロセスの前の部分で使用されます。収集した情報をさまざまなセキュリティ層に編成するのは、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。

図 2-1 は、一般的な組織内のデータを表現したものです。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、セキュリティが最大であることを示します。これは財務や HR データなどに適用

されます。組織により、これよりも層が多いか少ない場合がありますが、この図から、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要件も低くなることを理解できます 関連するリスクが大きくなるにつれ、セキュリティ要件も大きくなります。実際のところ、高度なセキュリティ要件を備えたデータはごくわずかであり、大量のデータはセキュリティ要件を必要としません。

図 2-1　データおよびサービスのセキュリティ要件



認証および承認機能の割り当てが可能な、データおよびサービスタイプの機能グループを構築することが目標であ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。適切であれば、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レベルに応じて分類できたなら、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層に関連付けます。たとえば、データで分類した場合、次のようにできます。

- 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設定するか、ネットワーク上に配置しないようにします。

認証要件とデータ / サービス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、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、万能であるとは言えませんが、セキュリティ要件を理解し、論理的に管理可能なグループにマッピングする助けになります。

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、アプリケーション間で使用する共通のグループ / ロールを見つけることです。これらは、組織内の機能ロールにマッピングされます。これは理想的な方法です。また、これらのロール / グループのソース (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。これは Sun Java System Directory Server に存在するのが理想です。存在しない場合には、カスタムプラグインが必要になります。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アクセスします。作業の完了時点で、以下のものを入手できているはずです。

- 既存のロールおよびグループの明快なマップ
- 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、コスト / 複雑性を低減するために作成する新規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できます。次の節では、一般的な配備スケジュールの手順を説明します。

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、大抵、期待される結果を含む設定済みのテストケースセットである、「テストベッド」データベースが存在します。このテストベッドに、コンセプト証明を適用できます。すべてが順調に進めば、新たに得られた結果がドキュメント化された結果と同じになります。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対する答えを提出し、すべてのニーズを効果的に満たすことができること、およびリスクを最小限に抑えられることを証明することです。これは通常高速に行われるため、限定さ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。コンセプト証明および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、大抵、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予算作成用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、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、重要なハブアプリケーション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、成功が迅速に立証および認識されるように、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分析 (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。これは、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、テストベッド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後で、徐々に製品環境に追加されてゆきます。

注　スケジュールはプロジェクトの複雑さに応じて変化するため、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。

配備のチューニング

Access Manager 6 2005Q1 のインストール後、amtune と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して、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、インタラクティブな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、特定の環境で実行したい amtune チューニングを指定します。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で、変更不可。

amtune 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 を除いて実際の変更を行います。

amtune 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、他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。

注 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、スクリプトファイルにより、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。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して、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と、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詳細は、『Sun Java System Access Performance Tuning Guide』を参照してください。

第 3 章

Access Manager のアーキテクチャ

Sun Java™ System Access Manager には、アイデンティティオブジェクト、ポリシー、およびサービスの管理用インタフェースが用意されています。この章では、これらのインタフェースについて、製品のコアアーキテクチャとの関連を示しながら説明します。また、必要に応じて、サービスのアーキテクチャ上の詳細についても説明します。次の節で構成されています。

概要

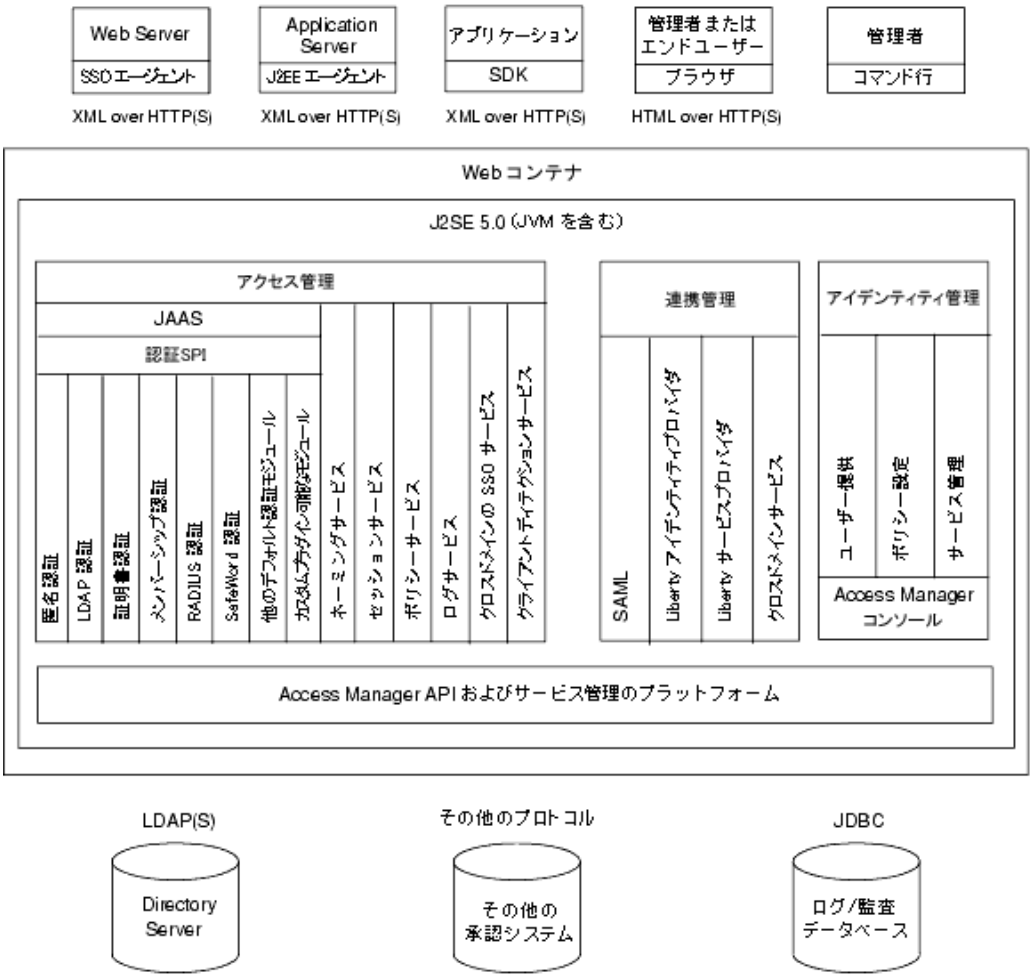
Access Manager は、Java™ 2 Platform, Enterprise Edition (J2EE) プラットフォーム上に構築されており、サービスを提供するサーブレット、Web サービスエンドポイント、および Web アプリケーションを操作するためのインフラストラクチャとして Web コンテナを使用します。提供される各サービスは、Web サービスとして実装されます。Web サービスは、現在注目を集めている標準技術セットを使用して構築され、ネットワーク上で集中的に提供されます。

これらのサービスを企業間で実装するため、Access Manager はアイデンティティ関連のオブジェクト、ポリシー、およびサービスの管理用インタフェースを提供します。主なインタフェースには、多様な Web サーバーおよびアプリケーションサーバーにセキュリティを提供するポリシーエージェント、Java™ および C アプリケーションプログラミングインタフェース (API)、XML (eXtensible Markup Language) ファイル、JavaServer Pages™、HTML (HyperText Markup Language) ページ、コマンド行インタフェース (CLI)、およびグラフィカルユーザーインタフェース (Access Manager コン

ソール) が含まれます。これらのインタフェースを使用することで、Sun Java System Directory Server 内のアイデンティティ情報の管理が可能になります。また、SSO、認証、ポリシーベースの承認、およびログ機能を企業規模のアプリケーションにまで拡張することも可能になります。

図 3-1 に、Access Manager 6 2005Q1 の機能アーキテクチャ、および異なるコンポーネントが相互にやり取りを行う方法を示します。

図 3-1　Access Manager 6 2005Q1 の機能アーキテクチャ



これらのコンポーネントに固有の個別サービスの詳細は、『Access Manager Developer's Guide』を参照してください。次の「統合化ポイント」では、一部のコンポーネントが相互にやり取りを行う方法について説明します。

統合化ポイント

Access Manager は、さまざまな統合化ポイントを利用してサービスの提供を行う分散型システムです。「統合化ポイント」とは、アプリケーションがサードパーティ製のシステムやソフトウェアとの円滑な対話処理を可能にするコネクタのことです。Access Manager の主要な統合化ポイントは、ポリシーエージェントおよび SDK (ソフトウェア開発キット) です。

ポリシーエージェント

Access Manager 内のアクセス制御は、ポリシーエージェントを使用して行われます。ポリシーエージェントは、指定された Web、アプリケーション、またはプロキシサーバーを不正な侵入から保護します。ポリシーエージェントは、Access Manager とは個別に提供されます。最新の Sun Java System Access Manager ポリシーエージェントは、Sun Microsystems Download Center からダウンロードできます。ポリシーエージェントは、Web およびプロキシサーバー上のリソースを保護するものと、さまざまな配備コンテナ内の J2EE アプリケーションを保護するものの 2 つに分類できます。

Web およびプロキシサーバーエージェント

Web サーバーおよびプロキシサーバーのポリシーエージェントは、対象のサーバーに配備されたコンテンツを不正侵入から保護します。エージェントは、管理者が設定したポリシーに基づいて、多数のサービスおよび Web リソースへのアクセスを制御します。ユーザーが、ブラウザで、保護された Web サーバー上の URL を指定すると、エージェントは要求を遮断し、ユーザーのセッショントークンが存在する場合にはそれを検証します。トークンの認証レベルが不十分であるか存在しない場合、ログインページに適切な認証サービスが呼び出されて、認証を (さらに) 行うようユーザーに求めます。認証サービスは、ユーザーの証明情報の有効性を検証します (たとえば、LDAP サービスは、ユーザー名およびパスワードが Sun Java System Directory Server に格納されているかどうかを検証する)。ユーザーの証明情報が正しく認証されている場合、ポリシーエージェントはユーザーに割り当てられているすべてのロール (ポリシーを含む) を確認します。ユーザーに割り当てられたすべてのポリシーを集約して、URL へのアクセスを許可するか拒否するかが個別に決定されます。

J2EE エージェント

Access Manager は、さまざまな配備コンテナ内の J2EE アプリケーションを保護するエージェントを提供します。これらのアプリケーションには、IBM Lotus Domino™ などのサーバーや、PeopleSoft などの企業が提供するアプリケーションが含まれます。通常、J2EE エージェントは、エージェントフィルタとエージェントレルムという 2 つのコンポーネントで構成されます (ただし、配備コンテナにより公開およびサポートされるインタフェースに部分的に左右される)。前者は認証を処理し、後者は承認を処理します。

エージェントフィルタ

エージェントフィルタは、サーバー内部に向かう要求を遮断するサーブレットフィルタです。これは、要求にセッショントークンが含まれるかどうかをチェックします。利用可能なセッショントークンが存在する場合、エージェントフィルタは Access Manager セッションサービスを使用してトークンを検証します。利用可能なトークンが存在しない場合、ユーザーは、通常の SSO 交換の場合のように認証サービスにリダイレクトされます。認証されたユーザーは、サーバーに再度導かれます。エージェントフィルタはそこで要求を再び遮断し、新たに取得したトークンを検証します。検証後に、フィルタはコンテナの主体およびレルムをインスタンス化します。要求は許可され、フィルタを通過してアプリケーションにアクセスすることが可能になります。このメカニズムを通して、エージェントフィルタは、有効な Access Manager トークンを保持する要求だけが、保護されたアプリケーションへのアクセスを許可されることを保証します。

エージェントレルム

エージェントレルムは、承認コンポーネントを提供します。「レルム」とは、J2EE 準拠のアプリケーションサーバーが、ユーザー、グループ、配備済みアプリケーションへのアクセス制御に関する情報を提供する手段です。これは、セキュリティポリシーが定義および施行される範囲を指します。リソースへのアクセスが試みられた場合、サーバーは、特定のレルムを使用してユーザーおよびそのロールを検証するように設定されます。デフォルトでは、多数のアプリケーションサーバーが、出荷時にレルムを実装しています。これには、デフォルトの File Based に加え、LDAP、NT、UNIX、および RDBMS (Relational Database Management System) が含まれます。エージェントレルムは、サーバーのレルムインタフェースを実装します。Access Manager を配備することでユーザーおよびロール情報の管理が可能になります。エージェントレルムを使用すると、エージェントフィルタにより認証されたユーザーに対し、ロールベースの J2EE リソースの承認をきめ細かく提供できます。

注　詳細は、『Sun Java System Access Manager Web Policy Agents Guide』または『Sun Java System Access Manager J2EE Policy Agents Guide』を参照してください。

Access Manager SDK

Access Manager 内部のアプリケーションを統合する別の方法は、SDK を使用してプログラム上で行う方法です。SDK アクセスは、主に Java で提供されます。いくつかの機能は C インタフェースで提供されます。XML over HTTP(s) 経由で Access Manager サービスとのインタフェースを直接提供することも可能です。ただし、ドキュメント形式は今後のリリースで変更される可能性があります。Access Manager の将来のバージョンでは、WSDL (Web Service Defintion Language) により定義されたエンドポイントとの Web サービスインタフェースが正式にサポートされる予定です。現在提供されている SDK は、次のとおりです。

アイデンティティ管理 SDK

基盤となる Directory Server で使用されるデータモデリングが記述された Access Manager テンプレートと、アイデンティティ管理 SDK を組み合わせて使用することで、ユーザー、ロール、グループ、コンテナ、組織、組織単位、およびサブ組織の作成および管理用フレームワークが提供されます。これにより、LDAP クライアント API よりもはるかに高い抽象化レベルでインタフェースが提供されるため、ディレクトリベースのアイデンティティ情報の管理が簡略化されます。ここで、コアパッケージは、Directory Server に対して直接動作する `com.iplanet.am.sdk`、および JAX-RPC 経由で Access Manager に対してリモートに動作することで Directory Server 自体に対して機能する `com.sun.identity.um` です (これは、`am_services.jar` の一部)。最後に、データの検証、パスワードポリシーの施行などのタスクに対応した、コア SDK のプラグイン作成用の SPI が提供されます。

サービス管理 SDK

開発者は、サービス管理インタフェースを使用してサービス定義を登録できます。サービス定義は、アプリケーション設定データの管理に使用されます。API パッケージ名は、`com.sun.identity.sm` です。

認証 API と認証 SPI

Access Manager では、プログラムによる Access Manager への認証を可能にする JAAS 実装が提供されます。これにより、認証サービスのすべての機能にリモートからアクセスできます。また、JAAS PAM 仕様に準拠した認証 SPI により、新規認証タイプの作成が可能になります。

ユーティリティ API

この API は、システムリソースの管理に使用できるいくつかの Java クラスを提供します。これには、スレッド管理やデバッグデータ形式などがあります。Java パッケージ名は com.iplanet.am.util です。現在 C インタフェースとの互換性はありません。

ログ API とログ SPI

ログサービスの主なタスクは、アクセス許可、アクセス拒否、およびユーザーアクティビティの記録です。ログ API を使用して、外部 Java アプリケーションのログ作成を有効にし、統合化された任意のアプリケーションに共通ログ機能を提供できます。ログ SPI を使用して、カスタマイズされた機能に対応したプラグインを開発できます。

クライアントディテクション API

Access Manager は、リソースにアクセスを試みているクライアントブラウザのタイプを検出して、適切に書式設定されたページを使って応答できます。この目的で使用する API パッケージは、`com.iplanet.services.cdm` です。このパッケージは、`am_services.jar` の一部です。

SSO API

Access Manager では、セッショントークンの検証および管理用インタフェース、およびユーザーの認証証明情報の管理用インタフェースが提供されます。SSO ソリューションへの参加を希望するアプリケーションはすべて、この API を使用できます。パッケージ名は `com.iplanet.sso` です。

ポリシー API

ポリシー API を使用することで、Access Manager ポリシーの評価と管理、およびポリシーサービスの追加機能の提供が可能になります。ポリシーストアからポリシーを直接読み込んで解釈するポリシーエバリュエータ、および JAX-RPC (Java API for XML-based Remote Procedure Calls) を使用して Access Manager のクライアントとして動作するリモートポリシーエバリュエータが提供されます。この API は、アイデンティティがアクセスするリソースを判別する機能も提供します。

SAML SDK

Access Manager は、SAML API を使用して認証動作、認証決定、および属性情報の交換を行います。API パッケージ名は、`com.sun.identity.saml` で始まります。このパッケージは、`am_services.jar` の一部です。

連携管理 API

Access Manager では、連携管理 API を使用して、Liberty Alliance Project 仕様に基づく機能が追加されます。API パッケージ名は、`com.sun.liberty` です。このパッケージは、`am_services.jar` の一部です。

詳細については、『Access Manager Federation Management Guide』を参照してください。

機能プロセス

Access Manager には、統合化された多数の機能が含まれます。以下にその内容を示します。

- 認証とユーザーセッション
- ポリシーの統合
- クライアントディテクションの統合
- CDSSO、SAML、および連携

統合化されたプロセスについては、次の節で説明します。プログラムに関連するポイントの詳細は、『Access Manager Developer's Guide』を参照してください。

認証とユーザーセッション

認証サービスは、Access Manager のエントリポイントです。ユーザーは、Access Manager コンソールおよび対応する管理機能にアクセスする前に、認証プロセスに合格しなければなりません。Access Manager により保護されたサービスまたはアプリケーションへのアクセスを試みるユーザーは、アクセス許可の前にも認証が必要になります。認証サービスは、認証モジュールを呼び出して、必要な証明情報を収集および検証します。Access Manager は、アプリケーションによるシングルサインオン機能への参加を可能にする API も提供します。シングルサインオン機能を利用すると、ユーザーは一度の認証で複数のリソースにアクセスできるようになります。

Access Manager への認証には、2 つの方法があります。1 つは、管理者またはエンドユーザーが管理目的で Access Manager コンソール自体へのアクセスを試み、認証サービスにリダイレクトされる方法です。もう 1 つは、ユーザーが Access Manager により保護されたアプリケーションへのアクセスを試みて認証サービスにリダイレクトされる方法です。前者では HTML over HTTP(S) インタフェースを使用し、後者では C ベースの API または Java API を使用します。

注 3 番目のオプションは、ユーザーがリモートサーバーマシンの保護されたリソースへのアクセスを試みる場合です。このオプションには、Access Manager ポリシーエージェントのインストールが含まれます。詳細は、65 ページの「ポリシーの統合」を参照してください。

HTML over HTTP(S) インタフェース

通常の Web ベースのシナリオでは、管理目的で Access Manager を起動する管理者またはエンドユーザーが、Web ブラウザに認証サービスの URI を入力して、認証ユーザーインタフェース Java サーブレットにアクセスします。

注	*URI* は、よく知られた URL を含む、インターネットオブジェクトの記述法の総称です。

URI は、1 つ以上の認証モジュールを定義する認証プロセスに対して事前に設定されます。サーブレットは URI をパースし、セッションサービスと通信を行い (セッションサービスは無効な状態のセッショントークンを作成する)、トークンを Cookie に埋め込みます。認証 SPI (`com.sun.identity.authentication.spi`) は特定の認証モジュールを呼び出して、ログインページおよびトークン /Cookie を要求元のブラウザに渡します。ユーザーは証明情報を入力し、認証 SPI は情報を認証サービスに返します。認証サービスは、情報を対応するデータストア内の情報と比較して検証します。要求された認証プロセスがすべて完了および成功すると、ユーザーのポリシー (URL アクセスおよび拒否リストを含む)、認証レベル、および設定パラメータがセッションサービスに渡されます。セッションサービスは、トークンを有効な状態に設定します。(セッション情報はトークンに埋め込まれず、サーバーに格納される。トークンは情報へのポインタに過ぎない)。最後に、URL パラメータを使用してユーザーが適切なリソースに導かれます。

注	認証サービスの一般的な情報、および URL パラメータの詳細情報については、『Access Manager Developer's Guide』を参照してください。

XML over HTTP(S) インタフェース

Java と C アプリケーションの両方で、認証 API を使用して Access Manager からユーザー認証を要求できます。Java パッケージ `com.sun.identity.authentication` は、認証プロセスを開始し、未検証の認証証明情報をアプリケーションから認証サービスに送り返すためのインタフェースを提供します。

注	`com.sun.identity.authentication` は、ローカルとリモートのどちらでも実装できます。リモートの場合、開発者はこのパッケージから取得したクラスおよびメソッドを Java アプリケーションに統合します。

Java コールは XML メッセージに変換されて、HTTP(S) 経由で Access Manager に渡されます。受信後に、XML メッセージは Java に再度変換され、認証サービスにより解釈されます。C アプリケーション開発者も同じ手順に従いますが、最初に Access Manager との通信を行います。

注 `http://server_name.domain_name:port/service_deploy_uri/authservice` は、XML over HTTP プロトコルを使用して通信を行うサーブレットです。関連する DTD の `remote-auth.dtd` は、Solaris システムでは *AccessManager-base*`/SUNWam/dtd` ディレクトリ、Linux システムでは *AccessManager-base*`/identity/dtd` ディレクトリに存在します。

接続のオープン後に C アプリケーションが有効になり、認証要求を XML メッセージとして HTTP(S) 経由で Access Manager に送信します。受信時に、XML メッセージが再度 Java に変換され、認証サービスによりパースされます。認証サービスは、認証メソッドを判別して、セッションサービスと通信を行います。セッションサービスは、無効な状態のセッショントークンを作成し、トークンを Cookie に埋め込みます。認証 SPI (`com.sun.identity. authentication.spi`) は特定の認証モジュールを呼び出して、ログインページおよびトークン /Cookie を要求元のアプリケーションに渡します。ユーザーは証明情報を入力し、認証 SPI は情報を Access Manager に返します。Access Manager は、情報を対応する認証データストア内の情報と比較して検証します。要求された認証プロセスがすべて完了および成功すると、ユーザーのポリシー (URL アクセスおよび拒否リストを含む)、認証レベル、および設定パラメータがセッションサービスに渡されます。セッションサービスは、情報をセッショントークンに埋め込んで、有効な状態に設定します。最後に、要求元にアプリケーションへのアクセスが許可されます。

ポリシーの統合

Access Manager 配備内部の認証とシングルサインオンはどちらも相互依存の関係にあります。このため、認証が成功した後、ユーザーがセッショントークンを保持していない場合には、ユーザーができることはあまりありません。ポリシーサービスには、同様の相互依存性が存在します。ユーザーのポリシーにより配備内部の保護されたリソースに関する権限が定義されるため、Access Manager なしでは、定義済みのポリシーに関して多くを行うことはできません。この機能は、Access Manager により保護されたリソースを格納するリモートマシンの 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管理されます。

注 最新のポリシーエージェントは、Sun Microsystems Download Center の Identity Management ページからダウンロードできます。`http://wwws.sun.com/software/download/`

ユーザーがポリシーエージェントにより保護された Web サーバーに接続すると、エージェントは要求を遮断して Cookie に埋め込まれたセッショントークンをチェックします。トークン /Cookie が見つからない場合、エージェントはユーザーをこのエージェント用に設定されたログイン URL にリダイレクトして認証プロセスを開始し、セッショントークンを作成します。ここから、63 ページの「認証とユーザーセッション」に記載された手順に従って処理が行われます。ユーザーは、有効なセッショントークンを受信すると、最初に要求したリソースに再び導かれます。そこで、エージェントにより要求が再度遮断され、セッショントークンがチェックされます。セッショントークンを検出したエージェントは、新規トークンを検証し、要求がネーミングサービスに渡されてトークンが復号化され、特定のセッションサービスが転送されて検証が行われるようにする必要があります。エージェントは、応答を受信してセッションサービスの URL を抽出し、その URL にアクセスしてトークンの検証を行います。セッションサービスは、応答をエージェントに返してトークンを検証します。エージェントは、何らかの理由でセッションがタイムアウトするか無効になった場合、セッションサービスが通知を受け取ることを要求します。セッションサービスは、肯定的な応答を返します。ユーザーの定義したポリシーを取得するため、エージェントはポリシーサービスと通信できるようになります。ポリシーサービスにより、決定が返されます。次に、エージェントはこの決定を使用して、要求されたリソースへのアクセスを許可または拒否します。また、ログメッセージがログサービスに送信されます。

クライアントディテクションの統合

ユーザーを認証する最初のステップは、要求元のクライアントタイプを識別することです。認証サービスは、URL 情報を使用してブラウザタイプの特性を取得します。これらの特性に基づいて、適正な認証ページ (HTML ページなど) がクライアントブラウザに送信されます。ユーザーの検証後に、クライアントタイプがセッショントークンに追加されます。セッショントークンを使用することで、他のサービスからもクライアントタイプを取得できるようになります。クライアントディテクションの詳細は、『Sun Java System Access Manager Developer's Guide』のクライアントディレクションに関する章を参照してください。

CDSSO、SAML、および連携

CDSSO (Cross-Domain Single Sign-on)、SAML (Security Assertion Markup Language) サービス、および連携管理は、Access Manager の別個の機能です。どの機能も、手法は異なりますが提供する機能は同じです。CDSSO は、複数の DNS ドメイン間でシングルサインオンを可能にする Access Manager 独自の機能です。既存の配備に SAML 仕様が実装されている組織では、同様の目的 (クロスドメインのシングルサインオン)

で SAML サービスを使用できます。連携管理では、連携アイデンティティ管理に Liberty Alliance Project バージョン 1.1 仕様が使用されます。連携管理を使用することで、接続されたパートナーのネットワーク経由でのサインオンの簡略化や、個人のアイデンティティの使用および公開などのユーザー制御が可能になります。

注 Liberty 仕様は、SAML 仕様を統合化して作成されています。ただし、この仕様で、SAML は Access Manager SAML サービス本体の一部ではありません。

CDSSO

CDSSO は、Access Manager 認証および承認プロセスの設定可能な部分です。これには、63 ページの「認証とユーザーセッション」で説明したポリシーエージェントのインストールおよび設定が含まれます。詳細は、『Sun Java System Access Manager Developer's Guide』、『Sun Java System Access Manager Web Policy Agents Guide』、または『Sun Java System Access Manager J2EE Policy Agents Guide』のシングルサインオンに関する章を参照してください。

SAML

SAML サービスは、セキュリティ当局と信頼されるパートナー間のセキュリティ情報の交換に XML フレームワークを使用する、オープンな標準プロトコルに基づいています。Access Manager の内部アーキテクチャの詳細は、『Sun Java System Access Manager Developer's Guide』の SAML サービスに関する章を参照してください。

連携

連携管理を使用すると、認証ドメイン、ホストプロバイダ、およびリモートプロバイダを Access Manager コンソールから設定および管理できます。これにより、ユーザーが一度サインオンするだけで、他の任意の連携リソースにアクセスできるようになります。連携アーキテクチャの詳細は、『Sun Java System Access Manager Developer's Guide』の連携管理に関する章を参照してください。

Access Manager の拡張

Access Manager は、リモートの Web またはアプリケーションサーバー、Sun Java System Directory Proxy Server などの LDAP ロードバランサ、およびマルチマスターレプリケーション内に配備できます。インストールする前に、以下の製品が配備に適しているかどうかを確認してください。Access Manager をインストールする前に、これらの製品をインストールおよび設定することが必要な場合があります。

Web コンテナ

一部の Web コンテナは、Access Manager の配備先の Sun Java System Web Server (または他の Web コンテナ) に対してリモートの位置に存在します。リモートの Web コンテナが、組織のコンテンツページ用として配備済みである場合には、Web コンテナを追加する必要があります。コンテンツを保護するためにポリシーエージェントをインストールすると、リモートの Web コンテナは Access Manager 環境に統合されます。Web コンテナのインストールおよび管理の詳細は、サーバーに付属のマニュアルまたは Sun Java System Web Server のマニュアルを参照してください。

複数の Directory Server インスタンス

アップグレード、フェイルオーバーディレクトリの設定、マルチマスターの配備を行うために、Directory Server の複数インスタンスをインストールできます。Access Manager の配備を成功させるためには、Directory Server のインストール、設定、および配備を適切に行う必要があります。Access Manager をインストールする前に、データベースレプリケーションアグリーメントを定義する必要があります。Directory Server のインストールおよび配備に関する詳細は、『Directory Server Installation and Migration Guide』および『Directory Server 配備計画ガイド』を参照してください。

LDAP ロードバランサ

Sun Java System Directory Proxy Server などのロードバランサと連携して動作するように、Access Manager を設定できます。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サーバー間でロードバランスを行う方法と、ユーザー付近に配置されているレプリケートされたサーバーの位置を検出する方法の 2 つの方法があります。ロードバランサを使用することで、Access Manager の基本機能に、高可用性レイヤーおよびディレクトリフェイルオーバー保護を追加できます。バックエンドの LDAP サーバーが一切利用できない場合、ロードバランサは要求トラフィックを管理し、クライアントのクエリを拒否します。インストールおよび管理の詳細は、Sun Java System Directory Proxy Server 5.2 のマニュアルを参照してください。ロードバランサに関するその他の情報については、製品に付属のマニュアルを参照してください。

第 4 章

配備前の考慮事項

Sun Java™ Access Manager 6 2005Q1 を使用すると、異機種のハードウェア、ソフトウェア、およびアプリケーションインフラストラクチャを抱える大規模な組織で、従業員、受託業者、顧客、およびサプライヤのアイデンティティ管理ソリューションを適切に配備できます。この章では、このプロセスに関連する高度な技術的概要を説明します。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、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。例を示します。

- 現在サポートが予定されている環境にどれくらいのユーザーが存在するか。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将来の計画はあるか。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

セキュリティの確保された内部および外部ネットワーク環境を提供する際、考慮する必要のあるオプションが多数存在します。以下にその内容を示します。

- 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、追加セキュリティレイヤーを提供するサーバーベースのファイアウォール。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する
- 最小化とは、システムの脆弱性が利用される機会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指す
- 分割 DNS インフラストラクチャは、1 つのドメイン内に 2 つのゾーンが作成されるインフラストラクチャ。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る。この手法は、より高度なセキュリティレベルを実現するために推奨されている。DNS サーバーも、ロードバランスを行うことで、パフォーマンスを向上させることができる

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期待どおりに長期間、連続的に使用可能であることです。このシステムは一般的に、複数のサーバーマシンで構成されますが、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、次の SPOF が含まれます。

- 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

これらの問題のうち、その大半は事前に予測できるので、高可用性の実現手法は、データストレージのバックアップやアクセスに対するフェイルオーバーの処理を中心に計画することが可能です。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。

注 高可用性シナリオでは、Access Manager のインストールすべてで同一の暗号化キーを指定する必要があります。

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。クラスタリングは、Sun Java System Access Manager では使用できないものの、システムの基盤である Sun Java System Directory Server のデータストアでは、極めて重要な手法です。たとえば、クラスタ化された 1 組の MMR サーバーでは、クラスタでの可用性を保証することにより、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

「水平スケーリング」は、複数のサーバーマシンを接続して 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バーは、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 台のサーバーマシン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、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、協調動作が可能です。通常、環境内のサーバーのインストールにより能力の限界まで使用されることはないため、垂直スケーリングはパフォーマンスを改善するために使用されます。また、特定のマシンが処理能力の限界に近づいた場合も、水平スケーリングは、多数のサーバーによって負荷を分散します。

ハードウェア要件

Access Manager をインストールするハードウェアは、一定の要件を満たす必要があります。Access Manager は、最低限、データストアとして使用する Sun Java System Directory Server および配備先の Web コンテナとともにインストールする必要があります。さらに、Directory Server と Access Manager は、異なるマシンにインストールすることが推奨されています。

詳細な要件は、コンポーネントのデフォルト設定 (Web Server によって配備される 1 つの Access Manager インスタンスと、1 つの Directory Server インスタンス) に基づく高度な内容です。Access Manager をインストールする前に、『Directory Server Installation and Migration Guide』、『Directory Server 配備計画ガイド』、および選択した Web コンテナのマニュアルを参照してください。

注 推奨される手順は、Sun Java System Access Manager を設計および配備する前に Sun Java System プロフェッショナルサービスまたは Sun Java System 認定システムインテグレータに相談することです。

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 配備の最小構成は、Access Manager と Sun Java System Web Server の両方がインストールされたマシンです。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 を搭載しても効果はあまり期待できません。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、256M バイト以上の RAM が必要です。

現実の配備シナリオでは、スレッド、SDK、HTTP サーバー、および他の内部処理用に 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います。その後、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 バイトメモリ制限を前提とする)。

注 ディレクトリリソース要件は、顧客固有のデータおよび使用方法に応じて変化しますが、通常は高度な要件が求められます。

ソフトウェア要件

Access Manager のインストール先システムは、最小のソフトウェアおよびオペレーティングシステム要件を満たす必要があります。

オペレーティングシステム要件

Access Manager 6 2005Q1 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® Platform Edition、バージョン 8、9 および 10
- Solaris™ OS、x86 Platform Edition、バージョン 9 および 10
- Red Hat™ Linux、Advanced Server

サポートされる特定のバージョンの詳細については、『Sun Java Enterprise System 2005Q1 リリースノート』および『Access Manager リリースノート』を参照してください。

Solaris 用パッチクラスタ

パッチクラスタは、108827-15 のように 2 つの番号で識別されます。最初の番号はパッチ自体を識別するためのもので、2 番目の番号はパッチのバージョンを識別するためのものです。全般的なパッチ情報および推奨されるパッチは、次の SunSolve Patch Portal からダウンロードできます。

`http://sunsolve.sun.com/`

現在 Solaris マシン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。

必要なパッチのリストについては、『Access Manager 2005Q1 リリースノート』および『Java Enterprise System 2005Q1 リリースノート』を参照してください。

JDK ソフトウェア要件

Access Manager 6 2005Q1 は、次の JDK ソフトウェアをサポートしています。

- JDK 1.5.0_01
- JDK 1.4.2_05 以上

Web コンテナ要件

Access Manager 6 2005Q1 は、完全インストールまたは SDK のみのインストール用に次の 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

サポートされる特定のバージョンについては、『Access Manager 6 2005Q1 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 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、この容量を考慮に入れる必要があります。詳細は、『Sun Java System Access Manager Web Policy Agents Guide』または『Sun Java System Access Manager J2EE Policy Agents Guide』を参照してください。

Directory Server 要件

Access Manager 6 2005Q1 には Sun Java System Directory Server 5 2005Q1 が必要です。要件の最新のリストについては、『Access Manager 6 2005Q1 リリースノート』を参照してください。

Web ブラウザ要件

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。このリリースのサポートされる Web ブラウザについては、『Access Manager 6 2005Q1 リリースノート』を参照してください。

Access Manager スキーマの理解

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。Directory Server には、データの格納方法を定義する LDAP (Lightweight Directory Access Protocol) スキーマが含まれます。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、Directory Server を、アイデンティティプロファイル、資格定義、および配備設定情報すべてのデータリポジトリとして使用します。このために、Access Manager は、Directory Server スキーマを拡張する独自のスキーマを保持します。Access Manager のインストール時に、`ds_remote_schema.ldif` および `sunone_schema2.ldif` に記述された Access Manager スキーマは、Directory Server スキーマと統合されます。`ds_remote_schema.ldif` には、特に Access Manager により使用される LDAP オブジェクトクラスおよび属性が記述されます。一般に、これらのオブジェクトクラスおよび属性は、以前のバージョンの Access Manager から受け継いだものです。`sunone_schema2.ldif` は、Sun Microsystems の新しい内部スキーマドキュメントで定義された Access Manager 固有の LDAP 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。参照用に、`ds_remote_schema.ldif` および `sunone_schema2.ldif` の内容を、それぞれ 77 ページのコード例 4-1 および 81 ページのコード例 4-2 に示します。

コード例 4-1 ds_remote_schema.ldif

```
add: objectClasses

objectClasses: ( 2.16.840.1.113730.3.2.175 NAME
'iplanet-am-session-service' DESC 'Session Service OC' SUP top
AUXILIARY MAY ( iplanet-am-session-max-session-time $
iplanet-am-session-max-idle-time $
iplanet-am-session-max-caching-time $
iplanet-am-session-get-valid-sessions $
iplanet-am-session-destroy-sessions $
iplanet-am-session-add-session-listener-on-all-sessions $
iplanet-am-session-service-status ) X-ORIGIN 'Sun Java System
Identity Management' )
```

コード例 4-1　ds_remote_schema.ldif (続き)

```
objectClasses: ( 2.16.840.1.113730.3.2.176 NAME
'iplanet-am-user-service' DESC 'User Service OC' SUP top
AUXILIARY MAY ( iplanet-am-user-auth-modules $
iplanet-am-user-login-status $ iplanet-am-user-admin-start-dn $
iplanet-am-user-auth-config $ iplanet-am-user-alias-list $
iplanet-am-user-success-url $ iplanet-am-user-failure-url $
iplanet-am-user-federation-info-key $
iplanet-am-user-federation-info $
iplanet-am-user-password-reset-options $
iplanet-am-user-password-reset-question-answer $
iplanet-am-user-password-reset-force-reset $
sunIdentityServerDiscoEntries ) X-ORIGIN 'Sun Java System
Identity Management' )
objectClasses: ( 2.16.840.1.113730.3.2.177 NAME
'iplanet-am-web-agent-service' DESC 'Web Agent Service OC' SUP
top AUXILIARY MAY ( iplanet-am-web-agent-access-allow-list $
iplanet-am-web-agent-access-deny-list $
iplanet-am-web-agent-access-not-enforced-list $
iplanet-am-web-agent-service-status ) X-ORIGIN 'Sun Java System
Identity Management' )
objectClasses: ( 2.16.840.1.113730.3.2.179 NAME
'iplanet-am-managed-role' DESC 'Managed Role OC' SUP top
AUXILIARY MAY ( iplanet-am-role-type $
iplanet-am-role-description $ iplanet-am-role-aci-description $
iplanet-am-role-aci-list $ iplanet-am-role-service-options $
iplanet-am-role-any-options $
iplanet-am-role-managed-container-dn $
iplanet-am-role-display-options) X-ORIGIN 'Sun Java System
Identity Management' )
objectClasses: ( 2.16.840.1.113730.3.2.180 NAME
'iplanet-am-managed-group' DESC 'Managed Group OC' SUP top
AUXILIARY MAY ( iplanet-am-group-subscribable $ inetgroupstatus )
X-ORIGIN 'Sun Java System Identity Management' )
objectClasses: ( 2.16.840.1.113730.3.2.181 NAME
'iplanet-am-managed-filtered-group' DESC 'Managed Filter Group
OC' SUP iplanet-am-managed-group AUXILIARY X-ORIGIN 'Sun Java
System Identity Management' )
objectClasses: ( 2.16.840.1.113730.3.2.182 NAME
'iplanet-am-managed-assignable-group' DESC 'Managed Assignable
Group OC' SUP iplanet-am-managed-group AUXILIARY X-ORIGIN 'Sun
Java System Identity Management' )
objectClasses: ( 2.16.840.1.113730.3.2.183 NAME
'iplanet-am-managed-static-group' DESC 'Managed Static Group OC'
SUP iplanet-am-managed-group AUXILIARY X-ORIGIN 'Sun Java System
Identity Management' )
```

コード例 4-1　ds_remote_schema.ldif (続き)

```
objectClasses: ( 2.16.840.1.113730.3.2.184 NAME
'iplanet-am-managed-person' DESC 'Managed Person OC' SUP top
AUXILIARY MAY ( iplanet-am-modifiable-by $
iplanet-am-static-group-dn $ iplanet-am-user-account-life )
X-ORIGIN 'Sun Java System Identity Management' )
objectClasses: ( 2.16.840.1.113730.3.2.186 NAME
'iplanet-am-managed-org-unit' DESC 'Managed OrganizationalUnit
OC' SUP top AUXILIARY MAY ( sunPreferredDomain $ associatedDomain
$ sunPreferredOrganization $ sunAdditionalTemplates $
sunOverrideTemplates $ iplanet-am-service-status ) X-ORIGIN 'Sun
Java System Identity Management' )
objectClasses: ( 2.16.840.1.113730.3.2.187 NAME
'iplanet-am-managed-people-container' DESC 'Managed People
Container OC' SUP top AUXILIARY X-ORIGIN 'Sun Java System
Identity Management' )
objectClasses: ( 2.16.840.1.113730.3.2.189 NAME
'iplanet-am-managed-group-container' DESC 'Managed Group
Container OC' SUP top AUXILIARY X-ORIGIN 'Sun Java System
Identity Management' )
objectClasses: ( 2.16.840.1.113730.3.2.166 NAME
'iplanet-am-managed-policy' DESC 'Managed Name Policy OC' SUP top
AUXILIARY MAY iplanet-am-named-policy-dn X-ORIGIN 'Sun Java
System Identity Management' )
objectClasses: ( 2.16.840.1.113730.3.2.167 NAME
'iplanet-am-domain-url-access-service' DESC 'Domain URL Access
Service OC' SUP top AUXILIARY MAY
iplanet-am-domain-url-access-allow X-ORIGIN 'Sun Java System
Identity Management' )
objectClasses: ( 1.3.6.1.4.1.42.2.27.9.2.22 NAME
'iplanet-am-saml-service' DESC 'SAML Service OC' SUP top
AUXILIARY MAY ( iplanet-am-saml-user $ iplanet-am-saml-password )
X-ORIGIN 'Sun Java System Identity Management' )
objectClasses: ( 1.3.6.1.4.1.42.2.27.9.2.23 NAME
'iplanet-am-auth-configuration-service' DESC 'Authentication
Configuration Service OC' SUP top AUXILIARY MAY (
iplanet-am-auth-configuration $ iplanet-am-auth-login-success-url
$ iplanet-am-auth-login-failure-url $
iplanet-am-auth-post-login-process-class ) X-ORIGIN 'Sun Java
System Identity Management' )
objectClasses: ( 1.3.6.1.4.1.42.2.27.9.2.25 NAME 'sunservice'
DESC 'object containing service information' SUP top MUST ou MAY
( labeleduri $ sunserviceschema $ sunkeyvalue $ sunxmlkeyvalue $
sunpluginschema $ description ) X-ORIGIN 'Sun Java System
Identity Management' )
```

コード例 4-1　　ds_remote_schema.ldif (続き)

```
objectClasses: ( 1.3.6.1.4.1.42.2.27.9.2.26 NAME 'sunorgservice'
DESC 'Service information specific to organizations' SUP top MUST
ou MAY ( sunkeyvalue $ sunxmlkeyvalue $ description ) X-ORIGIN
'Sun Java System Identity Management' )
objectClasses: ( 1.3.6.1.4.1.42.2.27.9.2.27 NAME
'sunservicecomponent' DESC 'Sub-components of the service' SUP
top MUST ou MAY ( sunserviceid $ sunsmspriority $ sunkeyvalue $
sunxmlkeyvalue $ description ) X-ORIGIN 'Sun Java System Identity
Management' )
objectClasses: ( 1.3.6.1.4.1.42.2.27.9.2.28 NAME
'sunserviceplugin' DESC 'Object that stores information specific
to plugins' SUP top MUST ou MAY ( sunpluginid $ sunkeyvalue $
sunxmlkeyvalue $ sunsmspriority ) X-ORIGIN 'Sun Java System
Identity Management' )
objectClasses: ( 1.3.6.1.4.1.42.2.27.9.2.74 NAME
'iplanet-am-managed-filtered-role' DESC 'Managed Filtered Role
OC' SUP iplanet-am-managed-role AUXILIARY X-ORIGIN 'Sun Java
System Identity Management' )
objectClasses: ( sunISManagedOrganization-oid NAME
'sunISManagedOrganization' DESC 'Sun Java System objectclass to
identify organizations' SUP top AUXILIARY MAY (
sunOrganizationAlias ) X-ORIGIN 'Sun Java System Identity
Management' )
objectClasses: ( sunIdentityServerDiscoveryService-OID NAME
'sunIdentityServerDiscoveryService' DESC 'Discovery Service OC'
SUP top AUXILIARY MAY ( sunIdentityServerDynamicDiscoEntries )
X-ORIGIN 'Sun Java System Identity Management' )
```

コード例 4-1　ds_remote_schema.ldif (続き)

```
objectClasses: ( sunIdentityServerLibertyPPService-oid NAME
'sunIdentityServerLibertyPPService' DESC
'sunIdentityServerLibertyPPService OC' SUP top AUXILIARY MAY (
sunIdentityServerPPCommonNameCN $
sunIdentityServerPPCommonNameALTCN $
sunIdentityServerPPCommonNameFN $ sunIdentityServerPPCommonNameSN
$ sunIdentityServerPPCommonNamePT $
sunIdentityServerPPCommonNameMN $ sunIdentityServerPPInformalName
$ sunIdentityServerPPLegalIdentityLegalName $
sunIdentityServerPPLegalIdentityDOB $
sunIdentityServerPPLegalIdentityMaritalStatus $
sunIdentityServerPPLegalIdentityGender $
sunIdentityServerPPLegalIdentityAltIDType $
sunIdentityServerPPLegalIdentityAltIDValue $
sunIdentityServerPPLegalIdentityVATIDType $
sunIdentityServerPPLegalIdentityVATIDValue
$sunIdentityServerPPEmploymentIdentityJobTitle
$sunIdentityServerPPEmploymentIdentityOrg $
sunIdentityServerPPEmploymentIdentityAltO ) X-ORIGIN 'Sun Java
System Identity Management' )
objectClasses: ( sunIdentityServerDevice-OID NAME
'sunIdentityServerDevice' DESC 'Device OC' SUP top AUXILIARY MAY
( cn $ uid $ sunIdentityServerDeviceVersion $
sunIdentityServerDeviceType $ userpassword $
sunIdentityServerDeviceKeyValue $ sunxmlkeyvalue $ description $
sunIdentityServerDeviceStatus ) X-ORIGIN 'Sun Java System
Identity Management' )
```

コード例 4-2　sunone_schema2.ldif

```
add: objectClasses

objectClasses: ( 2.16.840.1.113730.3.2.185 NAME
'sunManagedOrganization' DESC 'Auxiliary class which must be
present in an organization entry' SUP top AUXILIARY MAY (
inetDomainStatus $ sunPreferredDomain $ associatedDomain $
sunPreferredOrganization $ sunAdditionalTemplates $
sunOverrideTemplates $ sunRegisteredServiceName $
organizationName  ) X-ORIGIN 'Sun Java System Identity
Management' )
```

コード例 4-2　　sunone_schema2.ldif (続き)

```
objectClasses: ( 1.3.6.1.4.1.42.2.27.9.2.75 NAME
'sunManagedSubOrganization' DESC 'Auxiliary class which must be
present in an sub organization entry' SUP top AUXILIARY MAY (
inetDomainStatus $ parentOrganization ) X-ORIGIN 'Sun Java System
Identity Management' )
objectClasses: ( 1.3.6.1.4.1.42.2.27.9.2.29 NAME 'sunNameSpace'
DESC 'Auxiliary class which must be present at the root of a
subtree representing a namespace' AUXILIARY MAY
sunNameSpaceUniqueAttrs X-ORIGIN 'Sun Java System Identity
Management' )
objectClasses: ( 1.3.6.1.4.1.42.2.27.9.2.27 NAME
'sunservicecomponent' DESC 'Sub-components of the service' SUP
top MUST ou MAY ( sunserviceid $ sunsmspriority $ sunkeyvalue $
sunxmlkeyvalue $ description ) X-ORIGIN 'Sun Java System Identity
Management' )
```

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他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の詳細は、『Access Manager Developer's Guide』の第 5 章「Identity Management」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、『Access Manager 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 のアイデンティティ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。その後、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、そのロールに適したすべてのアクセス権がユーザーに与えられます。表 4-1 に、Access Manager 管理ロールおよび、各ロールに対応する書き込み権限の範囲を示します。

表 4-1 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	最上位組織内で作成された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。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。

表 4-1　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
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。
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。

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 の詳細は、『Sun Java System Access Manager 管理ガイド』の第 16 章「管理サービス属性」を参照してください。

管理者パスワード

Access Manager のインストール中には、管理者ユーザー ID (`amadmin`) と LDAP ユーザー ID (`amldapuser`) のパスワードを入力する必要があります。

`serverconfig.xml` ファイルには、Identity SDK が Directory Server への LDAP 接続プールを確立するために使用するパラメータが含まれます。たとえば、次のユーザーです。

- プロキシユーザー (ユーザー名 ="User 1"、cn=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。

- 管理ユーザー (ユーザー名 ="User 2"、cn=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 上で操作を実行するとき (たとえば、サービス設定情報の取得)、バインドするために使用される。

これらのユーザーにはそれぞれのパスワードが伴い、serverconfig.xml ファイルに暗号化された形式で格納されています。もちろん、これらのパスワード (特に puser パスワード) を常に保護する必要がありますが、次の点にも注意してください。

- Access Manager をインストールした後に puser と dsameuser のパスワードを変更する。
- puser および dsameuser には、amadmin または amldapuser に使用したパスワードと同じパスワードを使用しない。

puser および dsameuser のパスワードの変更については、『Access Manager 管理ガイド』の `amPassword` コマンド行ツール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現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。

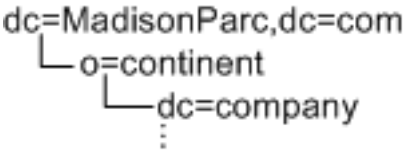
- 85 ページの「組織としてマークできるエントリのタイプは 1 つに限られる」
- 87 ページの「ピープルコンテナをユーザーの親エントリにする必要がある」
- 87 ページの「Access Manager XML で可能な組織の説明は 1 つに限られる」

この節の最後には、88 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。

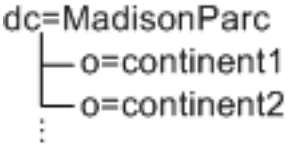
組織としてマークできるエントリのタイプは 1 つに限られる

Access Manager `iplanet-am-managed-org` 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 は、このエントリを組織であるように管理します。ただし、Access Manager では、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。

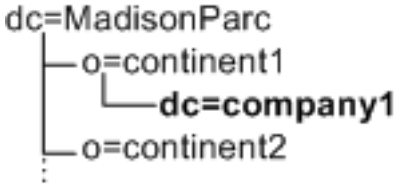
次の例では、`dc` と `o` の両方のエントリを組織にしようとしていますが、この DIT 構造は Access Manager で管理できません。



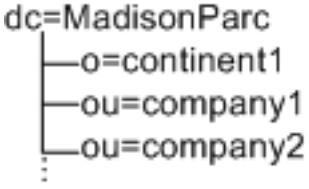
この規則には 1 つの例外があります。Access Manager ルートサフィックスでのエントリは、1 つのエントリには数えられません。したがって、次の例の DIT 構造は、Access Manager で管理できます。



`o=continent1` の下に `dc=company1` を追加したとすると、`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 は管理できます。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。UI 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。

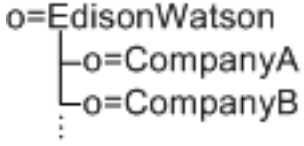
ピープルコンテナをユーザーの親エントリにする必要がある

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。

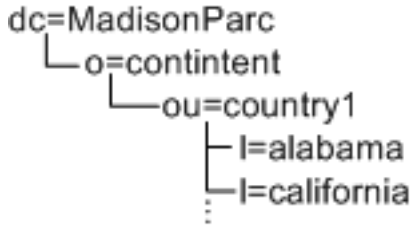


組織の説明は、組織レベルではなくルートレベル (o=EdisonWatson) で定義されます。デフォルトでは、企業 A と 企業 B の両方の UI を同一にする必要があります。また、すべてのサービスは、下位スキーマタイプユーザーの属性になるように、属性をグローバルに定義します。したがって、企業 A が、予備クラス CompanyA-user にそのユーザー用の属性を保持し、企業 B が、CompanyB-user に属性を保持している場合、企業 B の属性は上書きされ、表示されません。

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。

第 5 章

配備シナリオ

この章では、Sun Java™ System Access Manager 6 2005Q1 のさまざまな配備シナリオについて説明します。以下の節で構成されています。

複数サーバーのシナリオ

一般的な配備シナリオは 2 台のサーバーから構成され、各サーバーには、Access Manager と Directory Server の両方がインストールされています。Directory Server のインスタンスは、マルチマスター設定で設定されます。

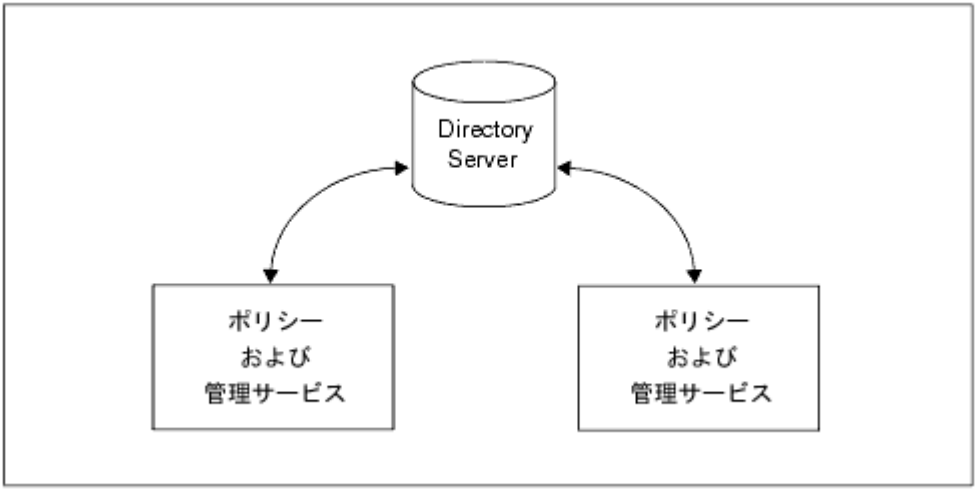
1 番目のサーバーにインストールされた Access Manager は、Directory Server のいずれかのインスタンスを指し示します。Java Enterprise System インストーラを使用したインストールでは、ユーザー自身の設定に応じて既存の DIT を使用するかまたは使用せずに、既存の Directory Server を選択できます。

Access Manager インストールスクリプトを実行して、2 番目のサーバーに Access Manager の 2 番目のインスタンスをインストールし、既存の DIT を使用して、このインスタンスが Directory Server を指し示すようにします。Access Manager は、すでに存在していることを認識した情報は、Directory Server に一切書き込みません。したがって、既存のデータが上書きされる危険はありません。

パフォーマンスの拡張、ディレクトリのレプリケーションのサポート、またはエージェントのフェイルオーバーのために、Access Manager の複数インスタンスを Directory Server に対してインストールできます。同じ Directory Server に対して、Access Manager の複数インスタンスを別々のサーバーにインストールするには、91 ページの「複数の Access Manager インスタンスのインストール」を参照してください。

図 5-1 には、1 つの Directory Server に対する 2 つの Access Manager インスタンスを示しています。

図 5-1 1 つの Directory Server に対する複数の Access Manager インスタンス



複数の Access Manager インスタンスのインストール

Access Manager の複数のインスタンスをインストールするには、次の手順に従います。

1. Java Enterprise System インストーラを実行して、Access Manager の 1 番目のインスタンスをインストールします。このインストーラの詳細については、『Sun Java Enterprise System 2005Q1 インストールガイド』を参照してください。
2. 作成しようとする Access Manager インスタンスごとに、新しい Web コンテナインスタンスを、Web コンテナ用の管理ツールを使用して作成および起動します。

   Web Server のマニュアルについては、以下を参照してください。
   http://docs.sun.com/coll/WebServer_05q1

   Application Server のマニュアルについては、以下を参照してください。
   http://docs.sun.com/coll/ApplicationServer8_ee_04q4
3. *AccessManager-base*/SUNWam/bin/amsamplesilent ファイルを、書き込み可能なディレクトリにコピーし、このディレクトリを現在のディレクトリにします。Linux システムでは、このファイルは *AccessManager-base*/identity/bin ディレクトリにあります。たとえば、/newinstances というディレクトリを作成します。
4. 配備する新しいインスタンスがわかるように、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 amnewws6instance に変更します。

   後でこのインスタンスをアンインストールする必要が生じた場合は、新しい amnewws6instance ファイルを保存します。
5. amnewws6instance ファイルで、次の変数を設定します。

   ```
   DEPLOY_LEVEL=1
   NEW_INSTANCE=true
   WEB_CONTAINER=WS6
   ```

   amnewws6instance ファイルで、作成する新しいインスタンスの必要に応じて他の変数を設定します。これらの変数の説明については、『Access Manager 2005Q1 管理ガイド』を参照してください。

   重要 : すべての Access Manager インスタンスでは、パスワード暗号化キーに同じ値を使用する必要があります。新しいインスタンスについて AM_ENC_PWD 変数を設定するには、最初のインスタンスの AMConfig.properties ファイルの am.encryption.pwd 属性から値をコピーします。異なるキー値で新しいインスタンスを配備している場合は、92 ページの「インストールのパスワード暗号化キーの変更」を参照してください。
6. 新しい amnewws6instance ファイルを入力値に指定して、amconfig スクリプトを実行します。Solaris システムの場合は次のようになります。

```
cd AccessManager-base/SUNWam/bin/
./amconfig -s /newinstances/amnewws6instance
```

-s オプションを付けると、スクリプトはサイレントモードで実行します。

スクリプトは、amnewws6instance ファイルの変数を読み込み、サイレントモードで実行され、新しい Web コンテナを設定します。

新しい Access Manager インスタンスの作成の詳細については、『Access Manager 2005Q1 管理ガイド』を参照してください。

インストールのパスワード暗号化キーの変更

パスワード暗号化キーは、Access Manager がユーザーパスワードの暗号化に使用する文字列です。すべての Access Manager サブコンポーネントは、アイデンティティ管理とポリシーサービスコアが使用する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 の複数インスタンスを別々のサーバーに配備し、管理コンソールまたは連携管理の共有ドメインサービスをインストールする予定がある場合は、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。

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。

詳細については、『Access Manager Developer's Guide』の付録 A 「AMConfig.properties File」を参照してください。

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、amconfig スクリプトを実行して追加の各インスタンスを配備する際に、このキーを使用して、サイレントインストールファイル (amsamplesilent) に AM_ENC_PWD 変数を設定します。

パスワード暗号化キーを変更する理由

次のシナリオでは、「パスワード暗号化キーを変更する」の説明のとおりに、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を説明します。

- それぞれ同じ Directory Server を指し示している Access Manager の複数サーバーインストールを実行しようとしており、1 番目の Access Manager のインストール時に使用したパスワード暗号化キーを保存していなかった場合、追加のインスタンスを配備する際に、使用するキーを取得する必要があります。

- 1 番目の Access Manager インスタンスと同じ Directory Server を指し示すが、パスワード暗号化キーが異なる追加の Access Manager インスタンスを作成した場合、その暗号化キーを 1 番目のインスタンスに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他の変更点

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、他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。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 ユーティリティについては、『Access Manager 管理ガイド』を参照してください。

パスワード暗号化キーを変更する

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、Web Server が Web コンテナの場合、次のように実行します。

   ```
   cd AccessManager-base/SUNWwbsvr/https-host2-name
   ./stop
   ```

6. AMConfig.properties ファイルを編集し、am.encryption.pwd および com.iplanet.am.service.secret の値を、手順 2 で 1 番目のインスタンスから保存しておいた値に交換します。

7. am.encryption.pwd に定義されている暗号化キーを変更したため、ampassword ユーティリティを実行して、serverconfig.xml のパスワードを再暗号化し、交換する必要があります。serverconfig.xml のパスワードは、<DirPassword> 要素によって識別できます。次の場合を考慮します。

   パスワードが同じ：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 AccessManager-base/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 つのパスワードを交換します。

   パスワードが異なる：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" および 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 のパスワードを交換します。

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、Web Server が Web コンテナの場合、次のように実行します。

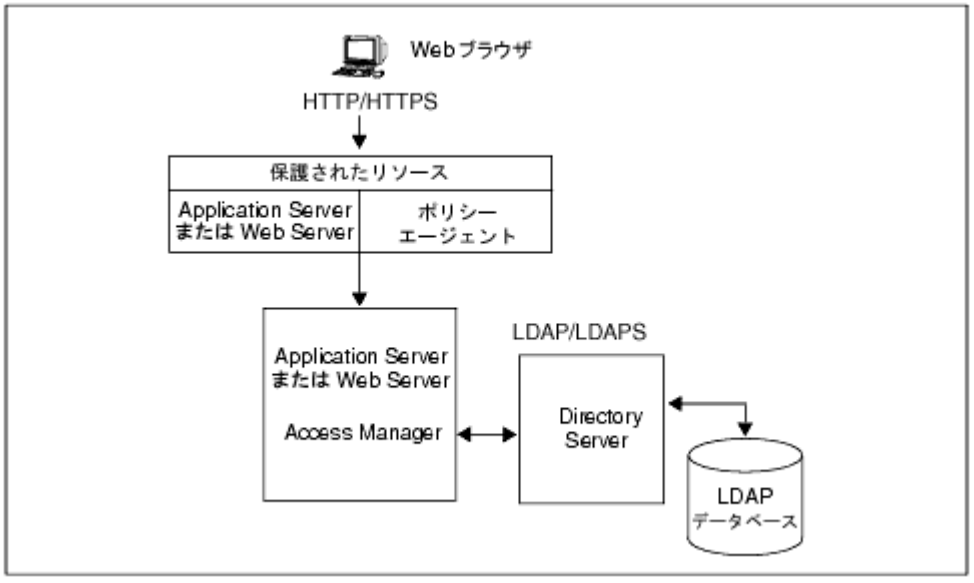
   ```
   cd AccessManager-base/SUNWwbsvr/https-host2-name
   ./start
   ```

9. 配備の中の Access Manager の他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。

Web 配備

Access Manager 配備の最も一般的な使用法は、Web ブラウザが 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する場合です。アプリケーションおよびリソースは 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 を配備できます。このアーキテクチャにより、マシン内の Web サーバーの数や、複数マシンにまたがる Access Manager のインスタンスに関して制限が課されることはありません。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 を配備できます。同様に、複数の Web サーバーを異なるマシン上で稼働させ、それぞれに Access Manager を配備することもできます。図 5-2 に、この簡単な配備シナリオを示します。

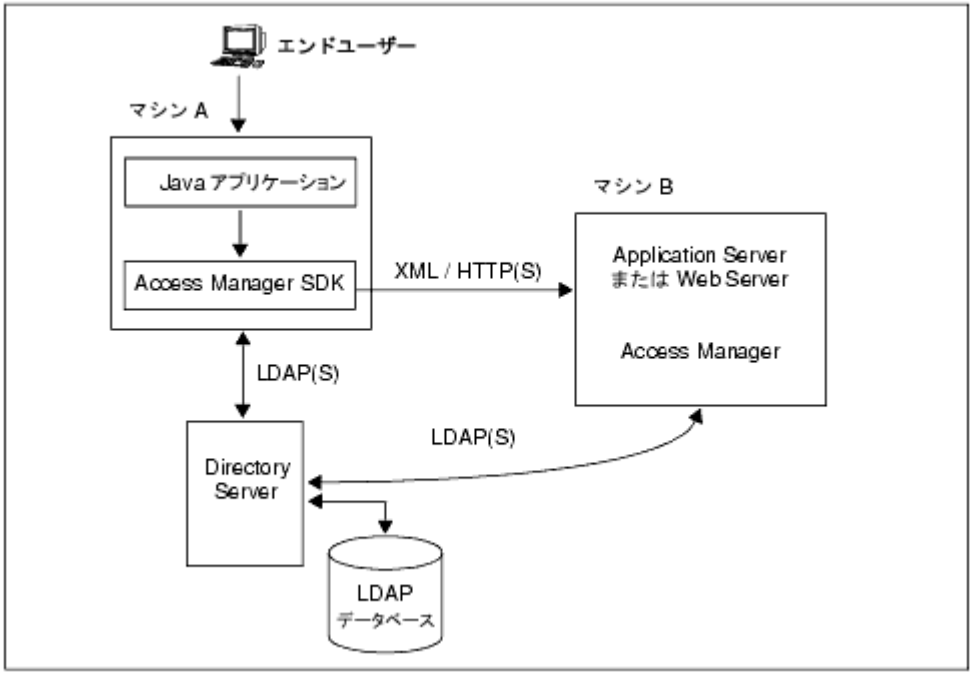
図 5-2　簡単な Web 配備シナリオ



Java アプリケーションの配備

Access Manager の別の一般的なシナリオは、Java™ アプリケーションが配備先のマシンに直接インストールされた Access Manager SDK にアクセスする場合です。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 のインスタンスが存在する追加マシンが必要になります。このマシンは、状態情報を管理してシングルサインオンを提供します。96 ページの図 5-3 に、この Java アプリケーションの配備シナリオを示します。

図 5-3　Java アプリケーションの配備



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 のアーキテクチャは、マシン内の 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 の数について制限を課しません。複数の JVM 環境の詳細については、Java System Application Server のマニュアルを参照してください。

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行う方法と、ユーザー付近に配置されているレプリケートされたサーバーの位置を検出する方法の 2 つの方法があります。Directory Server は、シングルサプライヤ設定またはマルチサプライヤ設定を行うことができます。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 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。

注 Access Manager をインストールする前に、ディレクトリレプリケーションアグリーメントを定義する必要があります。これにより、参照を検証する時間が許可されて、サプライヤデータベースとコンシューマデータベースが適正に同期するようになるため、更新の同期が正しく実行されます。

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
- ディレクトリマネージャのパスワード
- 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複数サプライヤレプリケーションで動作するように設定できます。図 5-4 に、コンシューマが読み取り専用データベースであるシングルサプライヤ設定を示します。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 5-4 シングルサプライヤレプリケーション

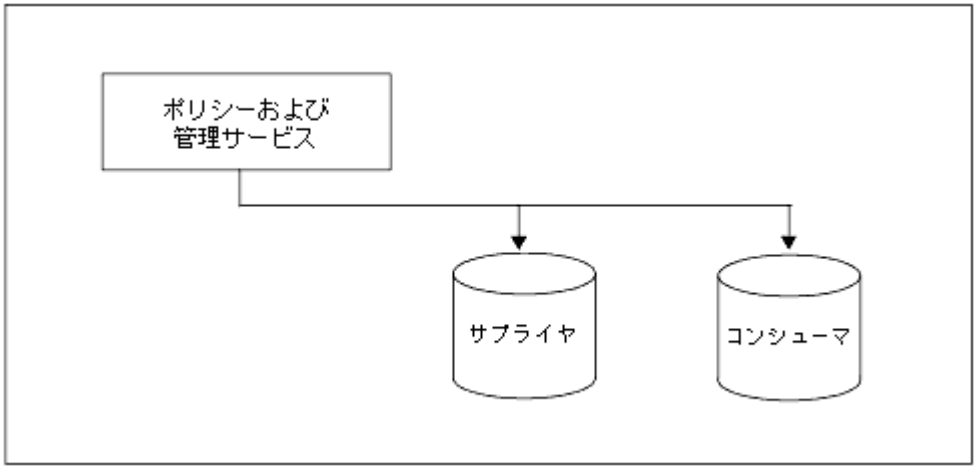
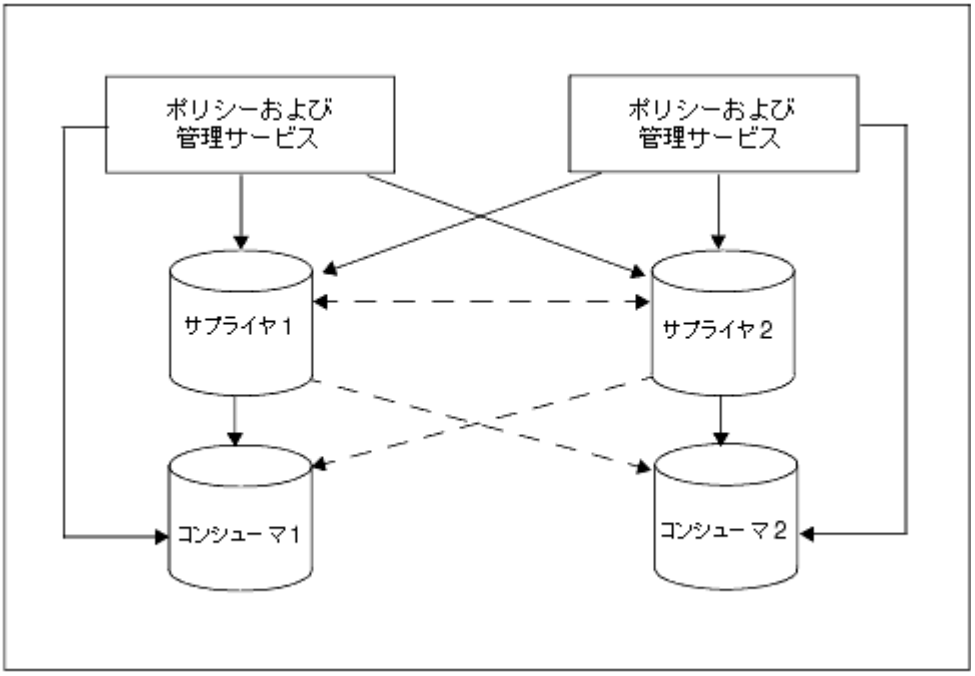


図 5-5 に、Access Manager の複数インスタンスを使用した複数サプライヤ設定を示します。この設定によりフェイルオーバー保護および高可用性が提供されるため、サーバーのパフォーマンスはさらに向上します。

図 5-5 複数サプライヤ設定 (マルチマスターレプリケーションとも呼ばれる)



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できます。また、デフォルトの組織レベルでレプリケーションを設定する場合にも、以下の手順を使用できます。

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

   手順の詳細については、『Sun Java Enterprise System 2005Q1 インストールガイド』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。

   手順の詳細については、『Directory Server 管理ガイド』を参照してください。

注 このバージョンの Access Manager で機能するように、既存の Directory Server データを移行することが必要な場合があります。手順の詳細については、『Access Manager 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、または既存のユーザーデータの使用を計画していない場合は、Sun Java Enterprise System 2005Q1 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、手順 1 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 管理ポリシーサービスをインストールしたサーバーで、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`

   a. 次のプロパティを修正して、手順 1 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`

   b. 次のプロパティを修正して、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`

   c. 次のプロパティを修正して、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`

5. 有効な Access Manager 認証モジュールごとに、手順 1 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。次の手順では、例として LDAP 認証モジュールを使用します。

   a. Access Manager コンソールで、「サービス設定」を選択します。

   b. 再設定する Access Manager サービスを見つけて、「プロパティ」の矢印をクリックします。

   c. 右側の区画で、以下の操作を行います。

      - 「LDAP サーバーとポート」という名前の最初のフィールドに、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入力します (例 : `consumer1.example.com:389`)。
      - 「LDAP サーバーとポート」という名前の 2 番目のフィールドに、セカンダリ (サプライヤ) Directory Server のホスト名とポート番号を入力します (例 : `supplier1.example.com:389`)。

   d. 「実行」をクリックします。

6. 次のファイルで、以下の操作を行います。
   AccessManager-base/SUNWam/config/ums/serverconfig.xml ファイルで、コード例 5-1 に示すように、手順 1 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。
7. Access Manager を再起動します。詳細については、『Sun Java System Access Manager 管理ガイド』(http://docs.sun.com/doc/819-1938?l=ja) を参照してください。

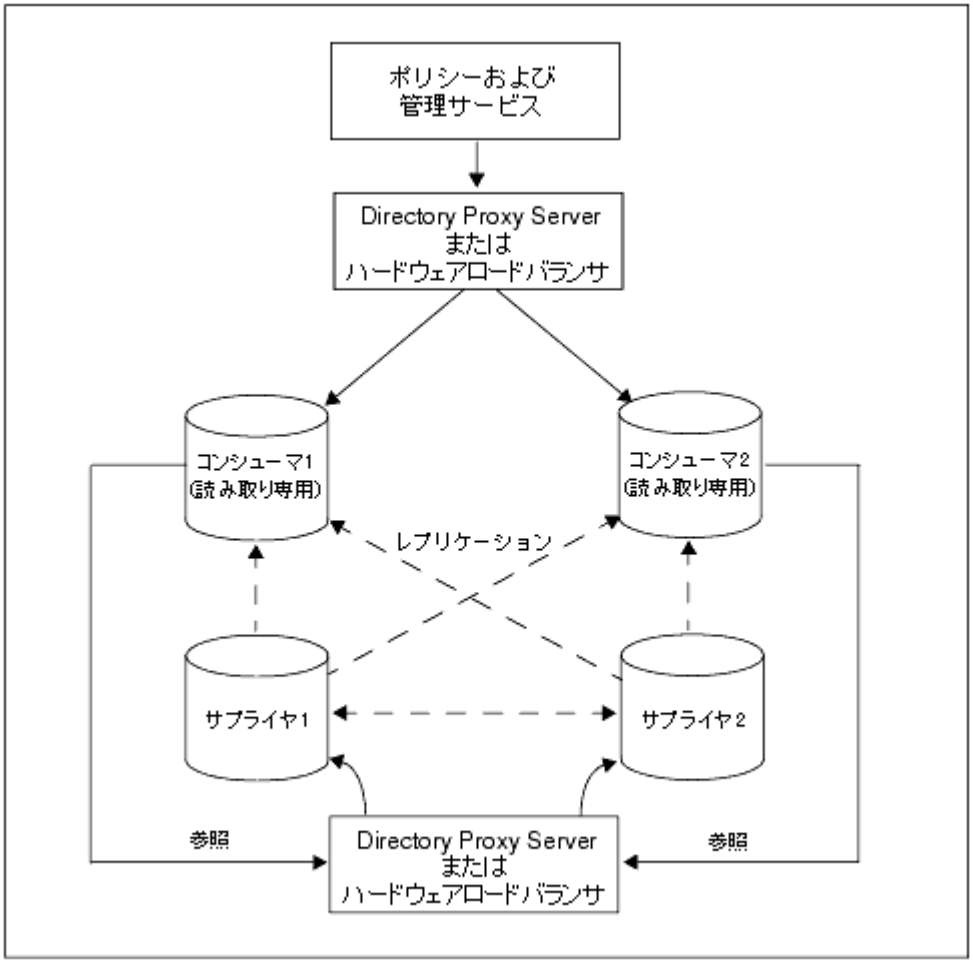
コード例 5-1 serverconfig.xml レプリケーションの修正

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

ロードバランサを使用する場合の設定

図 5-6 に、Directory Proxy Server を含む複数サプライヤ設定を示します。この設定により、Access Manager によ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のサポートが最大限に活用されます。

図 5-6　ロードバランサを使用する複数サプライヤレプリケーション



LDAP ロードバランサを使用することで、Access Manager の基本機能に、高可用性レイヤーおよびディレクトリフェイルオーバー保護を追加できます。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 サーバーが利用できなくなった場合には、要求トラフィックの管理を続行し、クライアントクエリの拒否を開始します。ロードバランサのインストールを選択した場合、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. 設定前に、以下の操作を行います。

a. Directory Server をレプリケーション用に設定します。ディレクトリのレプリケーションに関する総合的な情報、および設定方法に関する詳細情報については、『Directory Server 管理ガイド』の「レプリケーションの管理」を参照してください。

b. LDAP ロードバランサをインストールおよび設定します。製品に同梱されているマニュアルの指示に従ってください。

2. `AMConfig.properties` ファイルで、次のプロパティを修正して、手順 a でインストールしたコンシューマ Directory Server のロードバランサ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
```

注：`com.sun.am.event.connection.idle.timeout` プロパティは、持続的な検索が再開されるまでのタイムアウト値を分単位で指定します。理想的には、この値は、ロードバランサまたはファイアウォール TCP タイムアウト値より小さくして、接続が切断される前に持続的な検索が再開されるようにする必要があります。デフォルト値のゼロ (0) は接続が切断されるときに検索が再開されないことを指定します。

3. 有効な Access Manager 認証モジュールごとに、手順 a でインストールしたコンシューマ 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。

a. Access Manager コンソールで、「サービス設定」を選択します。

b. 再設定する認証モジュールを見つけて、「プロパティ」の矢印をクリックします。

c. 右側の区画で、以下の操作を行います。

- 「LDAP サーバーとポート」という名前の最初のフィールドに、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。
- 「LDAP サーバーとポート」という名前の 2 番目のフィールドには、何も入力しません。

d. 「実行」をクリックします。

4. *AccessManager-base*`/SUNWam/config/ums/serverconfig.xml` ファイル (Solaris システム) で、コード例 5-2 に示すように、手順 a 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。

5. Access Manager を再起動します。詳細については、『Sun Java System Access Manager 管理ガイド』(`http://docs.sun.com/doc/819-1938?l=ja`) を参照してください。

コード例 5-2　　serverconfig.xml ロードバランサの変更

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

注　　ロードバランサを使用する Access Manager の設定方法の詳細については、付録 E「ロードバランサの設定」を参照してください。

レプリケーションの警告

ディレクトリのレプリケーションを Access Manager 配備内に実装できない場合があります。Directory Server のレプリケーションの計画および実装に関する総合的な情報については、『Sun Java System Directory Server 配備計画ガイド』を参照してください。

ファイアウォールを使用した Directory Server

Access Manager と Directory Server 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 接続プールの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、次の解決策を参考にしてください。

- グローバルタイムアウト属性の設定
- 個々のクライアント接続のタイムアウトの設定

グローバルタイムアウト属性の設定

ファイアウォールのアイドル接続タイムアウト値を下回る値に対して、Directory Server の nsslapd-idletimeout グローバル属性を設定できます。ただし、この解決策は、nsslapd-idletimeout が、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、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個別の属性を設定できます。nsIdleTimeout 属性は、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。

amldapuser に nsIdleTimeout 属性を追加するには、Directory Server コンソールか、ldapmodify ツールを使用します。ldapmodify を使用するには、たとえば次のように指定します。

```
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。

Directory Server 属性と ldapmodify ツールの詳細については、次の Web サイトにアクセスして、『Directory Server 管理ガイド』を参照してください。

http://docs.sun.com/coll/DirectoryServer_05q1

Access Manager および Portal Server の配備

Java Enterprise System 2005Q1 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、インストールされた Directory Server のバージョン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バージョンで実行するか、以下を選択します。

- コンポーネントの選択パネルで、これら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 6 2005Q1 を選択します。
  - Directory & Identity Services で、Access Manager 6 2005Q1 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

    重要 デフォルトでは、Portal Server 6 2005Q1 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をチェックする必要があります。
- 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、インストールされた Directory Server のバージョン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。リモートサーバーのコンポーネントでインストールと設定を行う前に、このサーバーで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 をインストールする必要があります。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。

詳細については、次のマニュアルを参照してください。

- インストーラの実行 :『Sun Java Enterprise System 2005Q1 インストールガイド』(http://docs.sun.com/doc/819-0808?l=ja)
- Portal Server の配備 :『Sun Java System Portal Server 配備計画ガイド』(http://docs.sun.com/doc/819-1206?l=ja)
- Access Manager の設定 :『Sun Java System Access Manager 管理ガイド』(http://docs.sun.com/doc/819-1938?l=ja)

セッションフェイルオーバー

Sun Java™ System Access Manager 6 2005Q1 では、Sun Java System Message Queue (Message Queue) を通信ブローカーとして、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、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 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、ユーザーの再ログインを必要としたりせずに、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。

次の節があります。

- Access Manager セッションフェイルオーバーの概要
- ハードウェアおよびソフトウェア要件
- 配備シナリオ

Access Manager セッションフェイルオーバーの概要

Access Manager 6 2005Q1 セッションフェイルオーバーには、以下のコンポーネントが含まれます。

- Access Manager 6 2005Q1 の複数のインスタンス。各インスタンスは別々のホストサーバーのサポートされる Web コンテナで実行します。
- Access Manager インスタンスとセッションストアデータベース間のセッションメッセージを管理する Message Queue ブローカークラスタ。
- セッションストアデータベースとして Sleepycat Software, Inc. の Berkeley DB (http://www.sleepycat.com/)。Berkeley DB クライアントデーモンは amsessiondb です。

  デフォルトで、Access Manager は Message Queue と Berkeley DB をデフォルトの通信およびリポジトリソリューションとして使用します。必要に応じて、バックエンドセッションストアとして別の JDBC 2.0 準拠のリポジトリ (データベース) を、対応するプラグイン可能なセッションリポジトリインタフェースの適切な実装によって使用することもできます。ただし、デフォルト以外の別のセッションリポジトリを使う場合は、特別な構成 (このガイドでは取り上げていない) が必要です。

Access Manager セッションフェイルオーバーは、Message Queue 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を開始、更新、または終了すると、Access Manager はセッションの作成、更新、または削除メッセージを Message Queue ブローカー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 ブローカー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 ブローカークラスタに、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 ブローカー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共に Message Queue ブローカー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 ブローカーが失敗すると、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。後で Message Queue ブローカーが再起動すると、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の詳細については、『Sun Java System Message Queue 技術の概要』(http://docs.sun.com/doc/819-2221?l=ja) を参照してください。

ハードウェアおよびソフトウェア要件

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

- Solaris™ Operating System、SPARC® Platform Edition および x86 Platform Edition、バージョン 9 および 10
- Red Hat™ Linux、Advanced Server (サポートされる特定のバージョンについては、『Access Manager リリースノート』を参照)。

Access Manager 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必要です。

- Sun Java System Access Manager 6 2005Q1 (6.3)。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 Directory Server にアクセスする必要があります。
- Access Manager を実行するための Web コンテナ : Web Server 6.1 2005Q1 SP4、Application Server Enterprise Edition 8.1 2005Q1、IBM WebSphere Application Server、または BEA WebLogic。
- Sun Java System Message Queue 3 2005Q1 (3.6)
- Berkeley DB version 4.2.52
- JDK 1.4.x 以降

これらのプラットフォームおよびコンポーネントのサポートされるバージョンの最新情報については、『Access Manager リリースノート』(http://docs.sun.com/doc/819-1946?l=ja) を参照してください。

配備シナリオ

この節では、Access Manager セッションフェイルオーバーコンポーネントの配備のシナリオをいくつか紹介します。

図 5-7 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 (サポートされる Web コンテナ上)、Message Queue ブローカークラスタ、および Berkeley DB クライアント (amsessiondb) を実行しているシナリオを示します。両方の Access Manager インスタンスは同じ Directory Server にアクセスします。

図 5-7 Access Manager セッションフェイルオーバー配備シナリオ 1

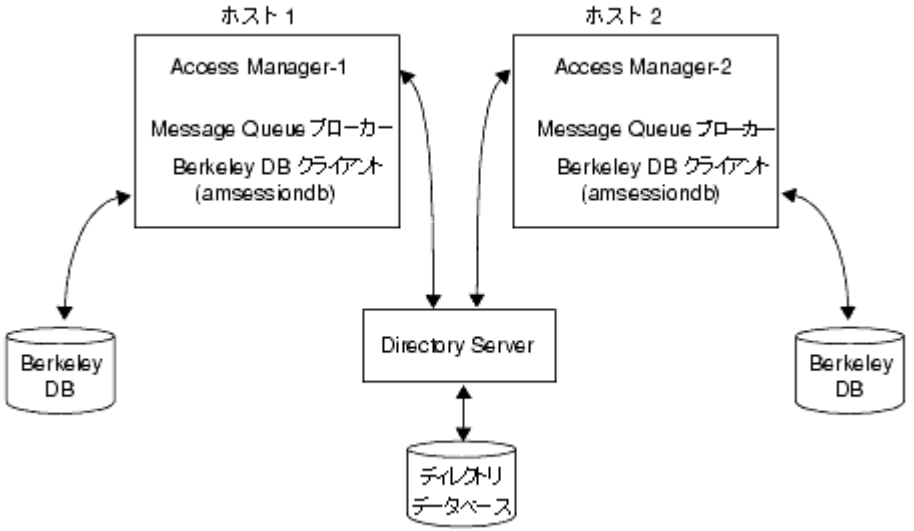


図 5-8 に、以下を含む Access Manager セッションフェイルオーバーを使用した配備シナリオを示します。

- 別々のホストサーバーのサポートされる Web コンテナで実行する 3 つの Access Manager インスタンス。
- 別々のサーバーでクラスタモードで実行する Message Queue ブローカー。
- Message Queue ブローカーと同じサーバーで実行する Berkeley DB クライアント (amsessiondb)。
- Access Manager のすべてのインスタンスからアクセスされる Directory Server。
- 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 5-8　Access Manager セッションフェイルオーバーシナリオ 2

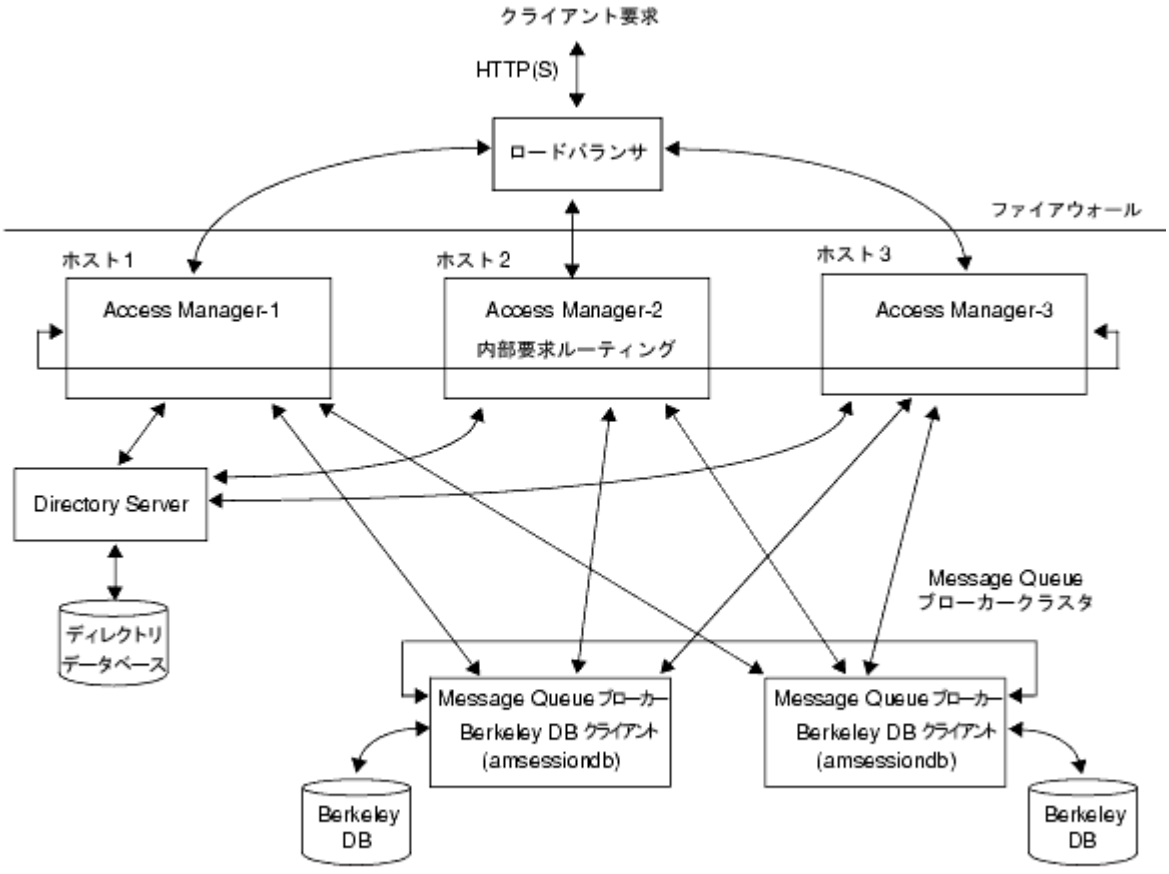


図 5-8 に示す配備のバリエーションとして、2 台のサーバーに Berkeley DB クライアント (amsessiondb) を配備して、パフォーマンスを向上することができます。

図 5-8 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。

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

表 5-1 に、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 5-1 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 6 2005Q1	Java ES 2005Q1 インストーラを使用して、Access Manager の 1 番目のインスタンスをインストールします。インストーラがセッションフェイルオーバーに必要なパッケージを追加します。 参照 :『Sun Java Enterprise System 2005Q1 インストールガイド』 http://docs.sun.com/doc/819-0808?l=ja amconfig スクリプトを実行し、以下を実行します。 • 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には、1 番目の Access Manager インスタンスを設定します。 • 他のホストサーバーに追加の Access Manager インスタンスを配備します。 参照 :『Sun Java System Access Manager 管理ガイド』 http://docs.sun.com/doc/819-1938?l=ja 図 5-7 に示す配備シナリオを参照してください。
Sun Java System 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 2005Q1 インストールガイド』 http://docs.sun.com/doc/819-0808?l=ja

表 5-1　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 (続き)

コンポーネント	インストール方法
Berkeley DB	Java ES インストーラおよび amconfig スクリプトは Access Manager パッケージまたはセッションフェイルオーバーに必要な RPM を追加します。 Access Manager のインスタンスを実行していないサーバーで、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 を追加します。 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 • SUNWbdbj 参照 : Solaris マニュアル : http://docs.sun.com/app/docs/prod/solaris 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 • sun-berkeleydatabase-java 参照 : Linux オンラインマニュアルページ。 図 5-8 に示す配備シナリオを参照してください。

ヒント　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キー値を使用する必要があります。1 番目のインスタンスをインストールする場合、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。次に、`amconfig` スクリプトを実行して、他のサーバーに Access Manager インスタンスを配備する際に、`amconfig` スクリプト入力ファイルの AM_ENC_PWD 変数に同じ値を設定します。

セッションフェイルオーバーの設定

Access Manager セッションフェイルオーバーを設定するには、以下の手順に従います。

1. AMConfig.properties ファイルの変更
2. Cookie エンコードを無効にする
3. Web コンテナ server.xml ファイルの編集
4. Message Queue サーバーの新しいユーザーの追加
5. ロードバランサ用のセカンダリ設定インスタンスの作成
6. amsessiondb スクリプトの編集 (必要な場合)

AMConfig.properties ファイルの変更

Access Manager インスタンスを実行している各ホストサーバーで、`AMConfig.properties` ファイルの次のプロパティをロードバランサを指し示すように変更します。例を示します。

```
com.iplanet.am.server.protocol=http
com.iplanet.am.server.host=lbhost.example.com
com.iplanet.am.server.port=80
com.iplanet.am.console.protocol=http
com.iplanet.am.console.host=lbhost.example.com
com.iplanet.am.console.port=80
com.iplanet.am.profile.host=lbhost.example.com
com.iplanet.am.profile.port=80
com.iplanet.am.naming.url=http://lbhost.example.com:80/amserver/nam
ingservice
```

配備で Liberty Alliance Project を実装している場合、このパラメータを設定します。

```
com.sun.identity.liberty.interaction.wspRedirectHandler=http://lbho
st.example.com:80/amserver/WSPRedirectHandler
```

Cookie エンコードを無効にする

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を false (Java ES インストーラによって設定されるデフォルト値) に設定します。

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

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

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

- Application Server 8.1、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 のエンコードまたはデコードも実行する必要がありません。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。

Web コンテナ 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、imq.jar および jms.jar の実際にインストールされている場所を Access Manager Web コンテナの server.xml (または同等の) 設定ファイルに追加します。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/htt
ps/jar/webserv-jstl.jar:/usr/share/lib/ktsearch.jar"
```

Message Queue サーバーの新しいユーザーの追加

Message Queue ユーザー名およびパスワードとして、「guest」ユーザーを使用したくない場合は、Message Queue がインストールされているサーバーで Message Queue ブローカー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とします。

```
# /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
```

ロードバランサ用のセカンダリ設定インスタンスの作成

Access Manager コンソールで、ロードバランサ用に新しいセカンダリ設定インスタンスを作成します。

1. amadmin として Access Manager コンソールにログインします。
2. 「サービス設定」、次に「セッション」をクリックします。
3. ロードバランサ用の新しい「セカンダリ設定インスタンス」を作成します。
   - インスタンス名 : ロードバランサの URL を指定します。
     例 : `http://lbhost.example.com:80`
   - セッションストアユーザーおよびセッションストアパスワード :「Message Queue サーバーの新しいユーザーの追加」で追加した名前とパスワードを使用します。
   - セッションクラスタサーバーリスト : 同じセッションフェイルオーバークラスタに参加しているプラットフォームリストからサーバー ID を指定します。ロードバランサのサーバー ID は含めないでください。例 : `02,03,04`
   - データベース URL: Message Queue ブローカーアドレスリストを指定します。
     例 : `mqsvr1.example.com:7777,mqsvr2.example.com:7777`

     注 : 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考慮してください。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. 「OK」をクリックして、変更を保存します。

amsessiondb スクリプトの編集 (必要な場合)

Access Manager では `/bin` ディレクトリに amsessiondb スクリプトが含まれます。

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

`amsessiondb` スクリプトは、「Message Queue サーバーの新しいユーザーの追加」で説明した Message Queue のユーザー名とパスワードを使用して、Message Queue ブローカーに接続します。スクリプトは Berkeley DB クライアント (amsessiondb) を起動し、データベースを作成するため、118 ページの表 5-2 に示すデータベース値を設定できます。

`amsessiondb` スクリプトには、Access Manager セッションフェイルオーバーコンポーネントの場所を指定する変数を格納しています。

```
JAVA_HOME=/usr/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て、変数に正しいパスを設定します。

Access Manager セッションフェイルオーバースクリプト

Access Manager では /bin ディレクトリに、Berkeley DB クライアント (amsessiondb) amsessiondb および amsfopasswd スクリプトを提供しています。

- 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。

amsessiondb

amsessiondb スクリプトは Berkeley DB クライアント (amsessiondb) を起動し、データベースを作成するため、表 5-2 に示すデータベース値を設定できます。Berkeley DB を作成するすべてのサーバーで、amsessiondb スクリプトを実行する必要があります。

重要 amsessiondb スクリプトを実行する前に、amsessiondb スクリプトの編集 (必要な場合) の説明のとおりに、パスが正しく設定されていることを確認します。

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 もしくは --password オプション)。しかし、ファイルに暗号化されたパスワードを使いたい場合 (-f もしくは --passwordfile オプション) は、次のようにスクリプトを実行します。

1. amsfopasswd スクリプトを実行し、ファイルへの Message Queue ブローカーのテキストパスワードを暗号化します。
2. -f または --passwordfile オプションに 手順 1 からのファイルを使用して、amsessiondb スクリプトを実行します。

amsessiondb の使用例

```
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
```

表 5-2　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。「Message Queue サーバーの新しいユーザーの追加」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 ブローカー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。「Message Queue サーバーの新しいユーザーの追加」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。
-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。 **注**：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 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ためには、100,000 セッションあたりに 1G バイト必要です。

表 5-2　amsessiondb スクリプトの引数

引数	説明
-a *MQServerAddressList* | --clusteraddress *MQServerAddressList*	次の形式の Message Queue ブローカーアドレスリスト。 *host*1:*port*[,*host*2:*port*,*host*3:*port*,...] 例: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。

amsessiondb の例

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

amsfopasswd

amsfopasswd スクリプトは、テキストの Message Queue ブローカーパスワードを受け入れ、ファイルに暗号化されたパスワードを返します。これにより、このファイルを amsessiondb スクリプトの入力として使えるようになります。

amsfopasswd の使用例

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

amsfopasswd -h | --help
```

表 5-3　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。

amsfopasswd の例

```
amsfopasswd -f /tmp/pwfile -c mypassword
```

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 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、以下の手順に従います。

1. Message Queue ブローカーの起動
2. Berkeley DB クライアント (amsessiondb) の起動
3. 各 Access Manager インスタンスの起動

Message Queue ブローカーの起動

Message Queue がインストールされている各サーバーで、`imqbrokerd` コマンドを使用して、Message Queue ブローカーを起動します。Solaris システムの場合は次のようになります。

```
# /usr/bin/imqbrokerd -vmargs "-Xms256m -Xmx2096m" -port 7777
-cluster mqsvr1:7777,mqsvr2:7777
```

ここで `mqsvr1` および `mqsvr1` は Message Queue サーバー名です。また、7777 は使用している Message Queue ブローカーポート番号です。

`imqbrokerd` コマンドの詳細については、『Sun Java System Message Queue 管理ガイド』(`http://docs.sun.com/doc/819-2217?l=ja`) を参照してください。

Berkeley DB クライアント (amsessiondb) の起動

Berkeley DB クライアント (amsessiondb) を起動する前に、データベースディレクトリが存在することを確認します。

- -b (または --dbdirectory) オプションを指定しない場合は、amsessiondb スクリプトを実行しようとしているディレクトリの下に、デフォルトの sessiondb ディレクトリが存在していることを確認します。また、sessiondb ディレクトリは空である必要があります。
- -b (または --dbdirectory) オプションを指定する場合は、指定するディレクトリが存在し、空であることを確認します。

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ためには、100,000 セッションあたりに 1G バイト必要です。

次に、データベースを作成する各サーバーで、amsessiondb スクリプトを実行します。たとえば、Solaris システムでは次のように実行します。

```
# cd /opt/SUNWam/bin
# ./amsessiondb -u amsvrusr -w password -b /opt/AMSessionDBFiles -a
mqsvr1:7777,mqsvr2:7777
```

または

```
# cd /opt/SUNWam/bin
# ./amsessiondb -u amsvrusr -f pwfile -b /opt/AMSessionDBFiles -a
mqsvr1:7777,mqsvr2:7777
```

ただし、

amsvrusr は「Message Queue サーバーの新しいユーザーの追加」で追加したユーザーであり、password はコマンド行に入力したパスワードです。

/opt/AMSessionDBFiles は既存のデータベースディレクトリです。

pwfile は暗号化された Message Queue パスワードを格納する amsfopasswd スクリプトが生成するファイルです。

mqsvr1:7777,mqsvr2:7777 は Message Queue ブローカーアドレスリストであり、7777 は Message Queue ブローカーポート番号です。

amsessiondb スクリプトの構文とオプションについては、amsessiondb スクリプトを参照してください。

各 Access Manager インスタンスの起動

各ホストサーバーで、Access Manager を起動します。詳細については、『Sun Java System Access Manager 管理ガイド』(http://docs.sun.com/doc/819-1938?l=ja) を参照してください。

連携管理

ここでは、Access Manager の連携管理機能を使用した配備設定について説明します。このシナリオでは、ドメイン A、ドメイン B、およびドメイン C のそれぞれに 2 つの Access Manager インスタンス、および 1 つの Directory Server インスタンスが含まれるものとします。ドメイン A およびドメイン B は、それぞれサービスプロバイダ A およびサービスプロバイダ B を保持します。ドメイン C は、アイデンティティプロバイダ C を保持します。

注 Access Manager では、1 つのサーバーインスタンス内に複数のプロバイダを受け入れることが可能です。

この連携シナリオを適正に管理するため、理解する必要のある概念が 2 つあります。

- 「ホストプロバイダ」は、特定の Access Manager インスタンスを、サービスプロバイダとして動作するものと定義します。
- 「リモートプロバイダ」には、Access Manager のインスタンスが設定されているマシンとは別のマシンをホストとする任意のタイプのプロバイダに関連するデータが含まれます。これは、Access Manager のインスタンスの場合もあれば、そうでない場合もあります。また、対応する任意のサービスプロバイダまたはアイデンティティプロバイダである可能性があります。

このため、上記の定義済みシナリオでは、サービスプロバイダ A は、自らをホストプロバイダとして、サービスプロバイダ B およびサービスプロバイダ C をリモートプロバイダとして、それぞれ設定します。サービスプロバイダ B は、自らをホストプロバイダとして、サービスプロバイダ A およびサービスプロバイダ C をリモートプロバイダとして、それぞれ設定します。サービスプロバイダ C は、自らをホストプロバイダとして、サービスプロバイダ A およびサービスプロバイダ B をリモートプロバイダとして、それぞれ設定します。これらの設定が完了したら、各ドメイン内に認証ドメインを作成できます。

連携管理の詳細については、『Access Manager Federation Management Guide』を参照してください。

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 インストーラを使用して Sun Java™ System Access Manager 6 2005Q1 をインストールした後のディレクトリレイアウトについて説明します。次のディレクトリがあります。

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 6 2005Q1 をインストールしたプラットフォームにより異なります。

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

Access Manager のマニュアルの中では、*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。

製品ディレクトリ

ベースインストールディレクトリ内で、Access Manager 6 2005Q1 パッケージ、共有バイナリファイル、コマンド行ツール、およびさまざまな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、デフォルトの製品ディレクトリは、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、`/SUNWam` または `/identity` の製品ディレクトリの名前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 :
  - amclient.war
  - amcommon.war
  - amconsole.war
  - ampassword.war
  - amserver.war.

  WAR ファイルの詳細については、『Access Manager Developer's Guide』を参照してください。

- サブディレクトリ：
  - /bin ディレクトリ
  - /docs ディレクトリ
  - /dtd ディレクトリ
  - /include ディレクトリ
  - /ldaplib/ ディレクトリ
  - /lib ディレクトリ
  - /locale ディレクトリ
  - /migration ディレクトリ
  - /public_html ディレクトリ
  - /samples ディレクトリ
  - /share ディレクトリ
  - /upgrade ディレクトリ
  - /web-src ディレクトリ

Access Manager をインストールした後、pkgchk(1M) ユーティリティを使用して、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。例を示します。

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

/bin ディレクトリ

表 A-1 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について説明しています。詳細については、『Access Manager 管理ガイド』を参照してください。

表 A-1 Access Manager のコマンド行ツールおよびユーティリティ

ユーティリティ	説明
am2bak	Access Manager コンポーネントをバックアップします。
amadmin	Directory Server への XML サービスファイルをロードし、DIT 上でのバッチ管理タスクを実行します。
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。

表 A-1　Access Manager のコマンド行ツールおよびユーティリティ (続き)

ユーティリティ	説明
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インストール設定ファイル。
amconfig、amutils、amdsconfig、amsdkconfig、amsvcconfig、amas70config、amwas51config、amwl81config、amws61config、amas81config	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の詳細については、『Access Manager 管理ガイド』を参照してください。
amserver	Access Manager インスタンスを起動および停止します。
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。
amsessiondb および amsfopassword	Access Manager セッションフェイルオーバースクリプト。

/amtune ディレクトリ

/amtune サブディレクトリには、Access Manager チューニングスクリプトが含まれます。これらのスクリプトには、amtune、amtune-as7、amtune-as8、amtune-directory.template、amtune-env、amtune-identity、amtune-os、amtune-prepareDSTuner、および amtune-ws61 があります。

これらのスクリプトでは、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 パラメータを設定できます。

/docs ディレクトリ

/docs ディレクトリには、API Javadoc に使用される HTML ファイルおよび関連ファイルが含まれます。これらのファイルには、allclasses-frame.html、am_public_javadocs.jar、com ディレクトリ、deprecated-list.html、help-doc.html、index-all.html、index.html、META-INF ディレクトリ、overview-frame.html、overview-summary.html、overview-tree.html、package-list、packages.html、serialized-form.html、および stylesheet.css があります。

/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、『Access Manager Developer's Guide』を参照してください。表 A-2 では /dtd ディレクトリの DTD ファイルについて説明しています。

表 A-2 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 (.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。

移行の詳細については、『Java System Access Manager Migration Guide』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、/admin、/appserver、/authentication、/console、/csdk、/liberty、/logging、/phase2、/policy、/saml、/sso、および /um のサブディレクトリが含まれます。

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 ディレクトリには、Access Manager 内部で使用される次の追加ユーティリティを収録した、bin/ サブディレクトリが含まれます。

- 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、以前のバージョンの Access Manager をアップグレードするためのスクリプトとその他のファイルが含まれます。

/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導入先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/` は、Access Manager Console の導入先ディレクトリ。index.html と、/META-INF、/WEB-INF、および /console のサブディレクトリが含まれる。

  /console ディレクトリには、/auth、/base、/federation、/images、/js directory、/policy directory、/service、/session、および /user のサブディレクトリが含まれる。

- `/common` は、Access Manager Liberty Common Domain コンポーネントの導入先ディレクトリ。/META-INF および /WEB-INF のサブディレクトリが含まれる。
- `/password` は、Access Manager Password Synchronization コンポーネントの導入先ディレクトリ。index.html と、/META-INF、/WEB-INF、および /password のサブディレクトリが含まれる。

- /services は、Access Manager Core Service の導入先ディレクトリ。index.html と、/META-INF、/WEB-INF、/admin、/config、/css、/docs、/fed_css、/fed_images、/images、/js、および /login_images のサブディレクトリが含まれる。

/debug、/logs、および /tmp ディレクトリ

/debug、/logs、および /tmp のデフォルトの場所は、Access Manager 2005Q1 をインストールしたプラットフォームによって異なります。

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

これらのディレクトリの詳細については、『Access Manager 管理ガイド』を参照してください。

設定 (/config) ディレクトリ

デフォルトの設定ディレクトリ (/config) の場所は、Access Manager 2005Q1 をインストールしている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 ファイルが収められています。

- .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 ファイル、AMConfig.properties.template、および SSOConfig.properties には、Access Manager 設定属性が含まれます。
- serverconfig.xml ファイルには、暗号化パスワードなどの設定情報が含まれます。
- amdrop.sql、amoraclecreate.sql、amhadbcreate.sql、amProfile.conf、および amhaenv.conf ファイルはセッションフェイルオーバのデータベース処理に使われます。
- map_mime.types ファイルは MIME 情報を識別します。
- /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。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、Sun Microsystems 内部の Schema 2 ドキュメントで定義された Access Manager 固有の LDAP 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 固有の LDAP スキーマオブジェクトクラスおよび属性を、Directory Server から削除します。

- /request サブディレクトリには、SunAMClientData.xml および mobileRequest.xml ファイルが含まれます。
- /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。
  - serverconfig.xml ファイルは、Access Manager の設定情報を Directory Server に提供します。
  - ums.xml の提供するテンプレートセットには、Access Manager を使用して管理されるアイデンティティ関連オブジェクトの LDAP 設定情報が含まれます。
- /xml サブディレクトリには、XML ファイルが含まれます。通常これらの XML ファイルは、設定には使用されません。これらを修正した場合は、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ない)。このディレクトリの XML ファイルの詳細については、『Access Manager Developer's Guide』を参照してください。

ユーザーセッションのライフサイクル

Sun Java™ System Access Manager では、アクセス管理サービスを提供する際、複数の HTTP (HyperText Transfer Protocol) 要求にまたがる、ユーザーと Web アプリケーションとの対話処理の追跡に使用するセッションオブジェクトを作成できます。この章は、Access Manager コンポーネントのプロトコルのやり取りを追跡して、セッションのライフサイクルについて説明します。次の節で構成されています。

概要

以降のセクションでは、Web ブラウザを使用して保護されたリソースへのアクセスを要求するユーザーに認証および承認サービスを提供する際の、Access Manager コンポーネントのプロトコルのやり取りを追跡します。これにより、セッションのライフサイクルの様子を理解できます。

要求

最初に、認証されていないユーザーが保護されたリソースに対する要求を作成します。この要求はサーバーに送信されます。リソースは、ポリシーエージェントにより保護されています。コード例 B-1 に、ブラウザから送信される GET 要求を示します。

コード例 B-1 GET 要求ヘッダー

```
GET / HTTP/1.1
Host: application.sun.com:8089
```

Access Manager では、すべてのアクセス要求は、有効なセッショントークン (プログラムでは SSOToken) が存在することで明示的に許可されない限り、暗黙的に拒否されます。

注 この動作は、導入時の条件に基づいて逆にできます。

この場合、セッショントークンは提供されないため、ポリシーエージェントは認証サービスに要求をリダイレクトします。コード例 B-2 に、要求元のブラウザに返されたリダイレクト情報を示します。これには、認証サービスへの URI、および元の要求の URL を含む goto パラメータが含まれます。

コード例 B-2 リダイレクト情報に対する GET 応答

```
HTTP/1.1 302 Moved Temporarily
Location:
http://identityserver.sun.com:58081/amserver/UI/Login?goto=http%
3A%2F%2Fapplication.sun.com%3A8089%2Findex.html
```

HTTP で待機中のユーザーのブラウザによりリダイレクトが許可され、認証サービス URI への要求が実行されます。コード例 B-3 に、認証サービスに送信される GET 要求を示します。

コード例 B-3　認証サービスにリダイレクトされる GET 要求

```
GET
/amserver/UI/Login?goto=http%3A%2F%2Fapplication.sun.com%3A8089%
2Findex.html HTTP/1.1
Host: identityserver.sun.com:58081
```

認証

認証要求の受信時に、認証サービスは、Access Manager の設定および要求パラメータに基づいて、ユーザーに提供する認証モジュールを決定します。新規の認証要求すべてと同様、ユーザーのやり取りを追跡するため、セッションサービスにより無効なセッショントークンが作成されます (このセッショントークンには、ユーザーを表すランダムに生成された文字列である暗号化されたセッション ID も含まれる)。セッショントークンは、Cookie (デフォルトでは `iPlanetDirectoryPro`) 内で設定されます。認証サービスは、認証要求に応答して、適切な証明情報をユーザーに求めるフォームとともにこれを送信します。

注　フォームのプロトコル (HTML、WML など) は、認証を要求するクライアントに基づいて、クライアントディテクションサービスにより決定されます。このサービスの詳細は、『Access Manager Developer's Guide』を参照してください。

コード例 B-4 に、ユーザーから認証証明情報を要求する HTML 認証フォームのヘッダーを示します。HTML 自体は、省略されています。

コード例 B-4　ユーザーに返される認証フォーム

```
HTTP/1.1 200 OK
Server: Sun-ONE-Web-Server/6.1
X-dsameversion: 6.1 6.1
Set-cookie:
JSESSIONID=DE271E3F2D52473B409DD8A7C58C24A5;Path=/amserver
Set-cookie:
iPlanetDirectoryPro=AQIC5wM2LY4SfcywFlTefDRmqlthG54qrg27LiyS8LnH
Aj4%3D;Domain=.sun.com;Path=/

<html>
Login Form...
</html>
```

ユーザーは、受信したフォームに認証証明情報を入力して、Access Manager に送信します。通常、この処理は、パスワード属性を保護するために SSL 経由で行われます。理解を容易にするため、コード例 B-5 で送信される証明情報はクリア HTTP 経由で送信されています。

コード例 B-5 Access Manager に返される POST 証明情報

```
POST /amserver/UI/Login HTTP/1.1
Host: identityserver.sun.com:58081
Cookie: JSESSIONID=DE271E3F2D52473B409DD8A7C58C24A5;
iPlanetDirectoryPro=AQIC5wM2LY4SfcywFlTefDRmqlthG54qrg27LiyS8LnH
Aj4%3D
Content-Type: application/x-www-form-urlencoded

IDToken1=user1&IDToken2=password
```

認証サービスにより受信された証明情報は、適切な認証モジュールにより検証されます。証明情報は検査に合格し、セッショントークンの状態は有効に変更され、セッション情報 (ログイン時刻、認証方式、認証レベルなど) は保存されるものとします。`iPlanetDirectoryPro` Cookie には、有効なセッショントークンが含まれるようになります。サーバーはブラウザに対し最初に要求されたリソースへのリダイレクトで応答します。コード例 B-6 に、このリダイレクト応答を示します。

コード例 B-6 最初に要求されたリソースへのリダイレクト

```
HTTP/1.1 302 Moved Temporarily
Server: Sun-ONE-Web-Server/6.1
X-autherrorcode: 0
X-dsameversion: 6.1
Location: http://application.sun.com:8089/index.html
```

HTTP で待機中のユーザーのブラウザにより、元のリソースへのリダイレクトが許可され、その後再度アクセスが要求されます。今回は、認証プロセス中に作成されたセッショントークンが、要求に含まれます。

コード例 B-7　トークンとともにリダイレクトされる GET 要求ヘッダー

```
GET /index.html HTTP/1.1
Host: application.sun.com:8089
Referer:
http://identityserver.sun.com:58081/amserver/UI/Login?goto=http%
3A%2F%2Fapplication.sun.com%3A8089%2Findex.html
Cookie:
iPlanetDirectoryPro=AQIC5wM2LY4SfcywFlTefDRmqlthG54qrg27LiyS8LnH
Aj4%3D
```

セッショントークン

ポリシーエージェントは、要求を再び遮断します。要求には Access Manager と同じ DNS ドメイン内のセッショントークンが含まれるようになったため、エージェントはこのトークンおよび関連するセッションの有効性の判定を試みます。最初に、セッションの出所を確認する必要があります。このため、ネーミングサービスとの通信が行われます。ネーミングサービスは、Access Manager が使用する内部サービスのサービス URL をクライアントが検索することを許可します。この情報は、セッションに関する通信に使用できます。ネームサービスは、セッションを暗号化して、対応する URL を返します。この URL を使用して、適用可能なサービスからセッションに関する情報が取得されます。コード例 B-8 に、ネーミング情報の POST 要求を示します。

コード例 B-8　ネーミング情報の POST 要求

```
POST /amserver/namingservice HTTP/1.0
Content-Type: text/xml; charset=UTF-8

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<RequestSet vers="1.0" svcid="com.iplanet.am.naming" reqid="9">
<Request><![CDATA[
<NamingRequest vers="1.0" reqid="2"
sessid="AQIC5wM2LY4SfcywFlTefDRmqlthG54qrg27LiyS8LnHAj4=">
<GetNamingProfile>
</GetNamingProfile>
</NamingRequest>]]>
</Request>
</RequestSet>
```

コード例 B-9 に、ネーミング情報の POST 要求に対する応答を示します。属性名および対応する URL 値に注目してください。

コード例 B-9　ネーミング情報に関する応答

```
HTTP/1.1 200 OK
Content-type: text/html

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ResponseSet vers="1.0" svcid="com.iplanet.am.naming" reqid="9">
<Response><![CDATA[<NamingResponse vers="1.0" reqid="2">
<GetNamingProfile>
<Attribute name="iplanet-am-naming-policy-url"
value="http://identityserver.sun.com:58081/amserver/policyservic
e"></Attribute>
<Attribute name="iplanet-am-naming-session-class"
value="com.iplanet.dpro.session.service.SessionRequestHandler"><
/Attribute>
<Attribute name="iplanet-am-naming-session-url"
value="http://identityserver.sun.com:58081/amserver/sessionservi
ce"></Attribute>
<Attribute name="iplanet-am-naming-samlawareservlet-url"
value="http://identityserver.sun.com:58081/amserver/SAMLAwareSer
vlet"></Attribute>
<Attribute name="serviceObjectClasses"
value="iplanet-am-naming-service"></Attribute>
<Attribute name="iplanet-am-naming-auth-url"
value="http://identityserver.sun.com:58081/amserver/authservice"
></Attribute>
<Attribute name="iplanet-am-naming-profile-class"
value="com.iplanet.dpro.profile.agent.ProfileService"></Attribut
e>
<Attribute name="iplanet-am-naming-samlassertionmanager-url"
value="http://identityserver.sun.com:58081/amserver/AssertionMan
agerServlet/AssertionManagerIF"></Attribute>
<Attribute name="iplanet-am-naming-umservice-url"
value="http://identityserver.sun.com:58081/amserver/UserManageme
ntServlet/"></Attribute>
<Attribute name="01"
value="http://identityserver.sun.com:58081"></Attribute>
<Attribute name="iplanet-am-naming-policy-class"
value="com.sun.identity.policy.remote.PolicyRequestHandler"></At
tribute>
<Attribute name="iplanet-am-naming-logging-class"
value="com.sun.identity.log.service.LogService"></Attribute>
<Attribute name="iplanet-am-naming-profile-url"
value="http://identityserver.sun.com:58081/amserver/profileservi
ce"></Attribute>
<Attribute name="iplanet-am-naming-samlsoapreceiver-url"
value="http://identityserver.sun.com:58081/amserver/SAMLSOAPRece
iver"></Attribute>
```

コード例 B-9　ネーミング情報に関する応答 (続き)

```
<Attribute name="iplanet-am-naming-logging-url"
value="http://identityserver.sun.com:58081/amserver/loggingservi
ce"></Attribute>
<Attribute name="iplanet-am-naming-fsassertionmanager-url"
value="http://identityserver.sun.com:58081/amserver/FSAssertionM
anagerServlet/FSAssertionManagerIF"></Attribute>
<Attribute name="iplanet-am-platform-server-list"
value="http://identityserver.sun.com:58081"></Attribute>
<Attribute name="iplanet-am-naming-samlpostservlet-url"
value="http://identityserver.sun.com:58081/amserver/SAMLPOSTProf
ileServlet"></Attribute>
<Attribute name="iplanet-am-naming-auth-class"
value="com.sun.identity.authentication.server.AuthXMLHandler"></
Attribute>
</GetNamingProfile>
</NamingResponse>]]></Response>
</ResponseSet>
```

ネーミングサービスにより提供される情報を使用して、ポリシーエージェントはセッションサービスへの POST 要求を作成して、含まれるセッショントークンを検証します。コード例 B-10 に、セッションサービスへの POST 要求を示します。

コード例 B-10　セッションサービスへのセッション検証用の POST 要求

```
POST /amserver/sessionservice HTTP/1.0
Host: identityserver.sun.com
Content-Type: text/xml; charset=UTF-8

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<RequestSet vers="1.0" svcid="Session" reqid="10">
<Request><![CDATA[
<SessionRequest vers="1.0" reqid="4">
<GetSession reset="true">
<SessionID>AQIC5wM2LY4SfcywFlTefDRmqlthG54qrg27LiyS8LnHAj4=</Ses
sionID>
</GetSession>
</SessionRequest>]]>
</Request>
<Request><![CDATA[
<SessionRequest vers="1.0" reqid="5">
<AddSessionListener>
<URL>http://application.sun.com:8089/amagent/UpdateAgentCacheSer
vlet?shortcircuit=false</URL>
<SessionID>AQIC5wM2LY4SfcywFlTefDRmqlthG54qrg27LiyS8LnHAj4=</Ses
sionID>
</AddSessionListener>
```

コード例 B-10　セッションサービスへのセッション検証用の POST 要求 (続き)

```
</SessionRequest>]]>
</Request>
</RequestSet>
```

セッションサービスは、要求を受信し、セッショントークンの有効性をチェックします。セッションがタイムアウトしていることも、その他の理由で無効であることもないと仮定し、セッションサービスはセッションが有効であると応答します。このアサーションは、セッション自体のサポート情報と一体化しています。

注　セッションリスナも、セッションに対して登録されます。これにより、セッションの状態または有効性が変更されていることを、ポリシーエージェントが通知することが可能になります。

コード例 B-11 に、セッションサービスの応答を示します。

コード例 B-11　セッションの有効性を示すセッションサービス応答

```
HTTP/1.1 200 OK

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ResponseSet vers="1.0" svcid="session" reqid="10">
<Response><![CDATA[<SessionResponse vers="1.0" reqid="4">
<GetSession>
<Session sid="AQIC5wM2LY4SfcywFlTefDRmqlthG54qrg27LiyS8LnHAj4="
stype="user" cid="uid=user1,ou=people,dc=sun,dc=org"
cdomain="dc=sun,dc=org" maxtime="300" maxidle="120"
maxcaching="3" timeidle="0" timeleft="17993" state="valid">
<Property name="authInstant"
value="2003-10-02T01:38:23Z"></Property>
<Property name="clientType" value="genericHTML"></Property>
<Property name="CharSet" value="UTF-8"></Property>
<Property name="Locale" value="en_US"></Property>
<Property name="UserToken" value="user1"></Property>
<Property name="loginURL"
value="http://identityserver.sun.com:58081/amserver/UI/Login"></
Property>
<Property name="SessionHandle"
value="shandle:AQIC5wM2LY4SfcxlvOiI7LJwWcusZAdkM3CHmIoeehu6urc="
></Property>
<Property name="Host" value="192.168.1.100"></Property>
<Property name="AuthType" value="LDAP"></Property>
```

コード例 B-11　セッションの有効性を示すセッションサービス応答 (続き)

```
<Property name="Principals"
value="uid=user1,ou=People,dc=sun,dc=org|AQIC5wM2LY4SfcywFlTefDR
mqlthG54qrg27LiyS8LnHAj4="></Property>
<Property name="cookieSupport" value="true"></Property>
<Property name="Organization" value="dc=sun,dc=org"></Property>
<Property name="USERS_DN"
value="uid=user1,ou=people,dc=sun,dc=org"></Property>
<Property name="AuthLevel" value="0"></Property>
<Property name="UserId" value="user1"></Property>
<Property name="HostName" value="192.168.1.100"></Property>
<Property name="Principal"
value="uid=user1,ou=people,dc=sun,dc=org"></Property>
</Session></GetSession>
</SessionResponse>]]></Response>
<Response><![CDATA[<SessionResponse vers="1.0" reqid="5">
<AddSessionListener>
<OK></OK>
</AddSessionListener>
</SessionResponse>]]></Response>
</ResponseSet>
```

ポリシー

ユーザーの提供したセッショントークンが有効であることを示す応答を受け取ると、ポリシーエージェントは要求されたリソースへのアクセスをユーザーに許可できるかどうかを判定する必要があります。ポリシーエージェントは、HTTP ネームスペース内の一部のリソースに関する決定を求める、ポリシーサービスへの要求を作成します。この要求とともに、IP アドレスや DNS 名など、設定済みポリシーの条件セットに影響を与える可能性のある追加環境情報も含まれます。コード例 B-12 に、情報を求めてポリシーサービス URI に送信される POST 要求を示します。

コード例 B-12　ポリシー情報を求める POST 要求

```
POST /amserver/policyservice HTTP/1.0
Host: identityserver.sun.com
Content-Type: text/xml; charset=UTF-8

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<RequestSet vers="1.0" svcid="Policy" reqid="11">
<Request><![CDATA[
<PolicyService version="1.0">
<PolicyRequest requestId="3"
appSSOToken="AQIC5wM2LY4SfczlhkwdQHfKgzxYu0qWY+DFCB9VGmknzvM=">
```

コード例 B-12　ポリシー情報を求める POST 要求 (続き)

```
<GetResourceResults
userSSOToken="AQIC5wM2LY4SfcywFlTefDRmqlthG54qrg27LiyS8LnHAj4="
serviceName="iPlanetAMWebAgentService"
resourceName="http://application.sun.com:8089/"
resourceScope="subtree">
<EnvParameters>
<AttributeValuePair>
<Attribute name="requestDnsName"/>
<Value>drnick</Value>
<Value>drnick.sun.com</Value>
</AttributeValuePair>
<AttributeValuePair>
<Attribute name="requestIp"/>
<Value>192.168.1.100</Value>
</AttributeValuePair>
</EnvParameters>
</GetResourceResults>
</PolicyRequest>
</PolicyService>]]>
</Request>
</RequestSet>
```

ポリシーサービスは、要求を受信した後で、要求に適用されるリソース定義を含むポリシーをチェックします。

注　ポリシーは、Access Manager 内にキャッシュされます。ポリシーがキャッシュに書き込まれていない場合は、Directory Server から読み込まれます。

要求に適用されるポリシーが検出されると、ポリシーサービスは、セッショントークンにより識別されたユーザーがいずれかのポリシーサブジェクトのメンバーであるかどうかを確認します。リソースに一致するポリシーが検出され、ユーザーが有効なサブジェクトである場合は、ポリシーの追加条件が評価されます (たとえば、正しい日時であるか、正しいネットワークからのものであるかなど)。すべての条件が満たされた場合は、ポリシーサービスがそのユーザーにアクセス許可を与えることを決定し、ポリシーエージェントに許可の決定を伝えます。コード例 B-13 に、ユーザーが保護されたリソースにアクセス可能であることを確認する、ポリシーサービスからの応答を示します。

コード例 B-13　許可ポリシー応答

```
HTTP/1.1 200 OK

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```

コード例 B-13　許可ポリシー応答 (続き)

```
<ResponseSet vers="1.0" svcid="policy" reqid="11">
<Response><![CDATA[<PolicyService version="1.0">
<PolicyResponse requestId="3">
<ResourceResult name="http://application.sun.com:8089/">
<PolicyDecision>
<ActionDecision timeToLive="1065121515571">
<AttributeValuePair>
<Attribute name="POST"/>
<Value>allow</Value>
</AttributeValuePair>
<Advices>
</Advices>
</ActionDecision>
<ActionDecision timeToLive="1065121515571">
<AttributeValuePair>
<Attribute name="GET"/>
<Value>allow</Value>
</AttributeValuePair>
<Advices>
</Advices>
</ActionDecision>
</PolicyDecision>
</ResourceResult>
</PolicyResponse>
</PolicyService>]]>
</Response>
```

要求されたページ

ポリシーエージェントは、ポリシーサービスからの決定を受信したら、このアクセス情報に基づいて動作する必要があります。この場合、GET および POST 操作に対して許可決定が発行されます。これはユーザーが要求した操作と一致するため、ポリシーエージェントはアクセスを許可します。決定はセッショントークンとともにキャッシュされるため、後続の要求はキャッシュを使用してチェックできます。Access Manager との通信は必要ありません。管理者により定義された時間を過ぎるか、ポリシーまたはセッションの状態変更が明示的に通知されると、キャッシュは期限切れになります。ただし、ユーザーにアクセスが許可される前にアクションをログに記録して、監査トレールを維持する必要があります。ポリシーエージェントは、ログ要求をログサービスに発行します。コード例 B-14 にログ要求を示します。

コード例 B-14　ポリシーエージェントからのログ要求

```
POST /amserver/loggingservice HTTP/1.0
Host: identityserver.sun.com
Content-Length: 416
Accept: text/xml
Content-Type: text/xml; charset=UTF-8

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<RequestSet vers="1.0" svcid="Logging" reqid="12">
<Request><![CDATA[
<logRecWrite reqid="2"><log logName="amAuthLog"
sid="AQIC5wM2LY4SfczlhkwdQHfKgzxYu0qWY+DFCB9VGmknzvM="></log><lo
gRecord><recType>Agent</recType><recMsg>User user1 was allowed
access to
http://application.sun.com:8089/index.html.</recMsg></logRecord>
</logRecWrite>]]></Request>
</RequestSet>
```

ログサービスは要求を受信し、設定に基づいて、要求を署名済みのファイル (オプション) または JDBC ストアに記録します。その後、応答はポリシーエージェントに返されて、ログが通知されます。コード例 B-15 に応答を示します。

コード例 B-15　エージェントにログを通知する応答

```
HTTP/1.1 200 OK

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ResponseSet vers="1.0" svcid="iplanet.webtop.service.logging"
reqid="12">
```

コード例 B-15　エージェントにログを通知する応答 (続き)

```
<Response><![CDATA[OK]]></Response>
</ResponseSet>
```

これで、要求されたリソースへのアクセスがポリシーエージェントにより許可されました。コード例 B-16 に、要求されたリソースの HTML ページを示します。HTML 自体は、省略されています。

コード例 B-16　エージェントによる要求されたページへのアクセス許可

```
HTTP/1.1 200 Ok
Content-type: text/html

<HTML>
The requested page....
</HTML>
```

シングルサインオン要求

この節では、2 つのスレッドについて説明します。最初のスレッドは、認証されたユーザーが、同じ DNS ドメイン内の異なるサーバー上の、セッショントークンにより検証済みの保護されたリソースを要求した時点で発生します。この 1 番目のスレッドはシングルサインオン機能を使用します。2 番目のスレッドは、認証されたユーザーが、異なる DNS ドメイン内の異なるサーバー上の保護されたリソースを要求した時点で発生します。2 番目のスレッドは、クロスドメインのシングルサインオン機能を使用します。

スレッド 1: シングルサインオン

要求したページの受信後に、ユーザーは異なるサーバー上の保護されたリソースを要求します。コード例 B-17 は、この 2 番目の要求を示します。セッショントークンが要求に含まれている点に注目してください。これにより、シングルサインオンが可能になります。

コード例 B-17　有効なセッショントークンを含む 2 番目の要求

```
GET / HTTP/1.1
Host: webservice.sun.com:8090
Cookie:
iPlanetDirectoryPro=AQIC5wM2LY4SfcywFlTefDRmqlthG54qrg27LiyS8LnH
Aj4%3D
```

セッショントークンが存在するため、ポリシーエージェントはユーザー認証を取得する必要はありません。このため、この場合は、135 ページの「認証」で説明した手順は省略されます。ただし、エージェントは、以前のセッショントークンを (キャッシュされたエントリが存在しないため) 表示できません。このため、137 ページの「セッショントークン」、141 ページの「ポリシー」および 144 ページの「要求されたページ」に記載された手順を実行します。

注　以下に示す手順は、137 ページの「セッショントークン」、141 ページの「ポリシー」および 144 ページの「要求されたページ」に記載されている手順を大幅に修正したものです。

1. ネーミングサービスとの通信が行われます。コード例 B-18 に、Access Manager が使用する内部サービスの URL を求めるネーミングサービスへの要求を示します。ネーミングサービスは、セッションを暗号化し、対応する URL を返します。この URL を使用して、セッションデータの取得が行われます。

コード例 B-18　ネーミングサービスの POST 要求

```
POST /amserver/namingservice HTTP/1.0
Host: identityserver.sun.com

...
<NamingRequest vers="1.0" reqid="2"
sessid="AQIC5wM2LY4SfcywFlTefDRmqlthG54qrg27LiyS8LnHAj4=">
....

HTTP/1.1 200 OK
```

コード例 B-18　ネーミングサービスの POST 要求 (続き)

```
POST /amserver/namingservice HTTP/1.0

<ResponseSet vers="1.0" svcid="com.iplanet.am.naming" reqid="9">
....
</ResponseSet>
```

2. ネーミングサービスからの応答に基づき、適切なセッションサービスとの通信が行われます。コード例 B-19 に、セッション ID を含むセッションサービスに送信される要求を示します。

コード例 B-19　セッションサービスへの POST 要求

```
POST /amserver/sessionservice HTTP/1.0

...
<SessionID>AQIC5wM2LY4SfcywFlTefDRmqlthG54qrg27LiyS8LnHAj4=</Ses
sionID>
....

HTTP/1.1 200 OK

....
<Session sid="AQIC5wM2LY4SfcywFlTefDRmqlthG54qrg27LiyS8LnHAj4="
stype="user" cid="uid=user1,ou=people,dc=sun,dc=org"
cdomain="dc=sun,dc=org" maxtime="300" maxidle="120"
maxcaching="3" timeidle="0" timeleft="17991" state="valid">
....
```

3. セッションサービスからの有効なセッション応答を想定し、ポリシーサービスに POST 要求が行われます。コード例 B-20 に、認証済みのユーザーに関するポリシー情報を求める、ポリシーサービスへの要求を示します。

コード例 B-20　ポリシーサービスへの POST 要求

```
POST /amserver/policyservice HTTP/1.0

...
```

コード例 B-20　ポリシーサービスへの POST 要求 (続き)

```
<GetResourceResults
userSSOToken="AQIC5wM2LY4SfcywFlTefDRmqlthG54qrg27LiyS8LnHAj4="
serviceName="iPlanetAMWebAgentService"
resourceName="http://webservice.sun.com:8090/"
resourceScope="subtree">
...
```

4. この場合、保護されたリソースへのアクセスを許可するポリシーは検出されず、ポリシーサービスは適切な否定応答を返します。コード例 B-21 に、この応答を示します。

コード例 B-21　ポリシーサービスからのアクセス否定応答

```
HTTP/1.1 200 OK

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ResponseSet vers="1.0" svcid="policy" reqid="11">
<Response><![CDATA[<PolicyService version="1.0">
<PolicyResponse requestId="3">
<ResourceResult name="http://webservice.sun.com:8090/">
<PolicyDecision>
</PolicyDecision>
</ResourceResult>
</PolicyResponse>
</PolicyService>]]>
</Response>
</ResponseSet>
```

5. ポリシーエージェントでログが設定されているため、このアクセス拒否がポリシーエージェントによりログに記録されます。コード例 B-22 に、要求したリソースへのアクセスがユーザーに許可されなかったことを示すログの記録を求める、ログサービスへの要求を示します。

コード例 B-22　ログサービスへの POST 要求

```
POST /amserver/loggingservice HTTP/1.0

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<RequestSet vers="1.0" svcid="Logging" reqid="12">
<Request><![CDATA[
```

コード例 B-22　ログサービスへの POST 要求 (続き)

```
<logRecWrite reqid="2"><log logName="amAuthLog"
sid="AQIC5wM2LY4SfcyueSeNMEFDFRRLub8BfjaxeyDvTPXlVnA="></log><lo
gRecord><recType>Agent</recType><recMsg>User user1 was denied
access to
http://webservice.sun.com:8090/index.html.</recMsg></logRecord><
/logRecWrite>]]></Request>
</RequestSet>

HTTP/1.1 200 OK

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ResponseSet vers="1.0" svcid="iplanet.webtop.service.logging"
reqid="12">
<Response><![CDATA[OK]]></Response>
</ResponseSet>
```

6. 次に、ポリシーエージェントは *Forbidden* メッセージをユーザーに発行します。コード例 B-23 に、このメッセージを表示する HTML ページを示します。ここで、アクセスが拒否されたことを示す、管理者の指定したページにユーザーをリダイレクトすることもできます。

コード例 B-23　アクセス拒否のメッセージを表示する HTML ページ

```
HTTP/1.1 403 Forbidden

<HTML><HEAD><TITLE>Forbidden</TITLE></HEAD>
<BODY><H1>Forbidden</H1>
Your client is not allowed to access the requested object.
</BODY></HTML>
```

スレッド 2: クロスドメインシングルサインオン

2 番目の要求ページへのアクセスが拒否されたため、ユーザーは別の DNS ドメイン内のサーバーに存在する保護されたリソースを要求します。Access Manager は、HTTP ドメイン Cookie (`http://www.ietf.org/rfc/rfc2965.txt`) を使用してアプリケーション間でトークンを転送するため、「スレッド 1: シングルサインオン」のように、要求を使ってポリシーエージェントにトークンが自動的に渡されることはありません。

1. トークンを新規 DNS ドメインに転送して、ドメイン内のアプリケーションから使用可能にするのは、Access Manager 内の CDSSO (Cross Domain Single Sign-On) プロトコルの役割です。コード例 B-24 に、別の DNS ドメイン内の保護されたアプリケーションへのアクセスをセッショントークンなしで求める要求を示します。

コード例 B-24　別の DNS ドメイン内の保護されたアプリケーションの GET 要求

```
GET /index.html?sunwMethod=GET HTTP/1.1
Host: webservice.java.com:8088
```

2. ポリシーエージェントは、セッショントークンが存在しないことを認識します。ただし、この場合、エージェントは CDSSO 用に設定されているため、リダイレクトは認証サービスではなく、セッションの転送に Liberty プロトコルを使用する CDSSO コントローラサービスに対して行われます。このため、リダイレクトには関連する Liberty パラメータが含まれます。コード例 B-25 に、ブラウザ経由で行われる、Liberty パラメータを含むリダイレクトを示します。

コード例 B-25　ブラウザ経由で行われる CDSSO コントローラサービスへのリダイレクト

```
HTTP/1.1 302 Moved Temporarily
Location:
http://identityserver.sun.com:58081/amserver/cdcservlet?goto=htt
p%3A%2F%2Fwebservice.java.com%3A8088%2Findex.html%3FsunwMethod%3
DGET&refererservlet=http%3A%2F%2Fwebservice.java.com%3A8088%2Fin
dex.html%3FsunwMethod%3DGET&RequestID=29980&MajorVersion=1&Minor
Version=0&ProviderID=http%3A%2F%2Fwebservice.java.com%3A8088%2Fa
magent&IssueInstant=2003-10-02T04%3A25%3A06Z&ForceAuthn=false&Is
Passive=false&Federate=false
```

3. HTTP で待機中のユーザーのブラウザにより、リダイレクトが許可されます。今回、プライマリドメイン内の Cookie であるため、要求にはセッショントークンが含まれます。コード例 B-26 に、ブラウザから CDSSO コントローラサービスへの、セッショントークンを含む HTTP リダイレクトを示します。

コード例 B-26　セッショントークンを含む、ブラウザからの HTTP リダイレクト

```
GET
/amserver/cdcservlet?goto=http%3A%2F%2Fwebservice.java.com%3A808
8%2Findex.html%3FsunwMethod%3DGET&refererservlet=http%3A%2F%2Fwe
bservice.java.com%3A8088%2Findex.html%3FsunwMethod%3DGET&Request
ID=29980&MajorVersion=1&MinorVersion=0&ProviderID=http%3A%2F%2Fw
ebservice.java.com%3A8088%2Famagent&IssueInstant=2003-10-02T04%3
A25%3A06Z&ForceAuthn=false&IsPassive=false&Federate=false
HTTP/1.1
Host: identityserver.sun.com:58081
Cookie:
iPlanetDirectoryPro=AQIC5wM2LY4SfcywFlTefDRmqlthG54qrg27LiyS8LnH
Aj4%3D
```

4. CDSSO コントローラサービス内の CDC サーブレットはセッショントークンを受け取り、セッション情報の詳細を定めた Liberty Post プロファイル応答を作成して、ブラウザに返信します。コード例 B-27 に、この返信を示します。

コード例 B-27　ブラウザへの POST 返信

```
HTTP/1.1 200 OK
Server: Sun-ONE-Web-Server/6.1
Date: Thu, 02 Oct 2003 01:38:28 GMT
Content-type: text/html
Pragma: no-cache
Transfer-encoding: chunked

<HTML>
<BODY Onload="document.Response.submit()">
<FORM NAME="Response" METHOD="POST"
ACTION="http://webservice.java.com:8088/index.html?sunwMethod=GE
T">
<INPUT TYPE="HIDDEN" NAME="LARES" VALUE="<ENCODED_LIBERTY_DOC>"/>
</FORM>
</BODY></HTML>
```

5. ユーザーのブラウザは、Body タグ onLoad に含まれる Action および Javascript に基づいて、Liberty ドキュメントを含むフォームをポリシーエージェントに自動的に送信します。コード例 B-28 に、ポリシーエージェントへのブラウザ POST を示します。

コード例 B-28 ポリシーエージェントへのブラウザ POST

```
POST /index.html?sunwMethod=GET HTTP/1.1
Host: webservice.java.com:8088
Referer:
http://identityserver.sun.com:58081/amserver/cdcservlet?goto=htt
p%3A%2F%2Fwebservice.java.com%3A8088%2Findex.html%3FsunwMethod%3
DGET&refererservlet=http%3A%2F%2Fwebservice.java.com%3A8088%2Fin
dex.html%3FsunwMethod%3DGET&RequestID=29980&MajorVersion=1&Minor
Version=0&ProviderID=http%3A%2F%2Fwebservice.java.com%3A8088%2Fa
magent&IssueInstant=2003-10-02T04%3A25%3A06Z&ForceAuthn=false&Is
Passive=false&Federate=false
Content-Type: application/x-www-form-urlencoded

LARES=<ENCODED_LIBERTY_DOC>
```

6. ポリシーエージェントは、Liberty ドキュメントを受信して、ユーザーのセッション情報を抽出します。この時点で、ポリシーエージェントが通常のエージェントと同様の方法でセッションを検証する必要があります。このため、リソースへのアクセスを許可または拒否する前に、137 ページの「セッショントークン」、141 ページの「ポリシー」、および 144 ページの「要求されたページ」に記載された手順に従って処理が実行されます。

注 説明を簡略化するため、137 ページの「セッショントークン」、141 ページの「ポリシー」および 144 ページの「要求されたページ」の内容はここでは繰り返しません。

この場合、セッショントークンは有効であると判定され、ユーザーはアクセスが許可されます。ポリシーエージェントは、要求されたドキュメントをユーザーに渡し、新規 DNS ドメインの Cookie 内にセッショントークンを設定します。これで、新規ドメイン内のすべてのエージェントが Cookie を使用できるようになります。コード例 B-29 に、新規 DNS ドメイン Cookie セットを含む、返される HTML ページを示します (HTML 自体は、簡略化のために削除されている)。

コード例 B-29　ユーザーにアクセスが許可された、新規 Cookie を含む HTML ページ

```
TTP/1.1 200 Ok
Set-cookie:
iPlanetDirectoryPro=AQIC5wM2LY4Sfcwwte3peDKZILScZ40uK%2FsU0I0WQQ
P6xXA%3D;Domain=.java.com;Path=/

<HTML>
...
</HTML>
```

セッションの終了

1. 認証、および SSO と CDSSO の実行が完了したため、ユーザーはセッションを終了できます。セッションは、アイドルや最大タイムアウト、またはユーザーの明示的なログアウトを条件に、管理者が終了できます。この場合は、ユーザーがサービスへのリンクをクリックしてログアウトします。コード例 B-30 に、ログアウトサービスへのアクセス要求を示します。

コード例 B-30　ログアウトサービスの GET 要求

```
GET /amserver/UI/Logout HTTP/1.1
Host: identityserver.sun.com:58081
Cookie:
iPlanetDirectoryPro=AQIC5wM2LY4SfcywFlTefDRmqlthG54qrg27LiyS8LnH
Aj4%3D
```

2. ログアウトサービスは、ログアウト要求を受信し、ユーザーのセッションを破棄されたものとしてマークし、セッショントークンに無効な値を新たに設定し、成功を示すログアウトページをユーザーに返します。コード例 B-31 に、ログアウト成功後にユーザーに送信される HTML ページの詳細を示します (HTML 自体は、簡略化のために削除されている)。

コード例 B-31　ユーザーに返される成功を示す HTML ページ

```
HTTP/1.1 200 OK
Server: Sun-ONE-Web-Server/6.1
Set-cookie:
iPlanetDirectoryPro=AQICv6c3Z1VfvgCgpLlaEqkqM70TLV24OTB1XkPa%2BL
tA8bXvC7c5XABU95Ta8UJi6dQZhXEUSgTRBWHXQ6kcxe8qgw%3D%3D;Domain=.s
un.com;Path=/

<title>Sun ONE Access Manager (Logout)</title>
...
```

3. ユーザーのセッション状態が変更されたため、セッションに関心を示したアプリケーションに通知するのはセッションサービスの役割になります。この場合は、各ポリシーエージェントはセッション通知用に設定されており、セッションが無効であることをエージェントに通知するドキュメントがポリシーエージェントごとに送信されます。これにより、セッションがキャッシュからフラッシュされます。

コード例 B-32　セッション通知用の POST

```
POST /amagent/UpdateAgentCacheServlet?shortcircuit=false HTTP/1.1
Host: application.sun.com:8089

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<NotificationSet vers="1.0" svcid="session" notid="8">
<Notification><![CDATA[<SessionNotification vers="1.0" notid="8">
<Session sid="AQIC5wM2LY4SfcywFlTefDRmqlthG54qrg27LiyS8LnHAj4="
stype="user" cid="uid=user1,ou=people,dc=sun,dc=org"
cdomain="dc=sun,dc=org" maxtime="300" maxidle="120"
maxcaching="3" timeidle="3" timeleft="17983" state="destroyed">
<Property name="authInstant"
value="2003-10-02T01:38:23Z"></Property>
<Property name="clientType" value="genericHTML"></Property>
<Property name="CharSet" value="UTF-8"></Property>
<Property name="Locale" value="en_US"></Property>
<Property name="UserToken" value="user1"></Property>
<Property name="loginURL"
value="http://identityserver.sun.com:58081/amserver/UI/Login"></
Property>
<Property name="SessionHandle"
value="shandle:AQIC5wM2LY4SfcxlvOiI7LJwWcusZAdkM3CHmIoeehu6urc="
></Property>
<Property name="Host" value="192.168.1.100"></Property>
<Property name="AuthType" value="LDAP"></Property>
```

コード例 B-32　セッション通知用の POST (続き)

```
<Property name="Principals"
value="uid=user1,ou=People,dc=sun,dc=org|AQIC5wM2LY4SfcywFlTefDR
mqlthG54qrg27LiyS8LnHAj4="></Property>
<Property name="cookieSupport" value="true"></Property>
<Property name="Organization" value="dc=sun,dc=org"></Property>
<Property name="USERS_DN"
value="uid=user1,ou=people,dc=sun,dc=org"></Property>
<Property name="AuthLevel" value="0"></Property>
<Property name="UserId" value="user1"></Property>
<Property name="HostName" value="192.168.1.100"></Property>
<Property name="Principal"
value="uid=user1,ou=people,dc=sun,dc=org"></Property>
</Session>
<Type>5</Type>
<Time>1065058714469</Time>
</SessionNotification>]]></Notification>
</NotificationSet>


POST /amagent/UpdateAgentCacheServlet?shortcircuit=false HTTP/1.1
Host: webservice.sun.com:8090

....
<Session sid="AQIC5wM2LY4SfcywFlTefDRmqlthG54qrg27LiyS8LnHAj4="
stype="user" cid="uid=user1,ou=people,dc=sun,dc=org"
cdomain="dc=sun,dc=org" maxtime="300" maxidle="120"
maxcaching="3" timeidle="3" timeleft="17983" state="destroyed">
...


POST /amagent/UpdateAgentCacheServlet?shortcircuit=false HTTP/1.1
Host: webservice.java.com:8088

...
<Session sid="AQIC5wM2LY4SfcywFlTefDRmqlthG54qrg27LiyS8LnHAj4="
stype="user" cid="uid=user1,ou=people,dc=sun,dc=org"
cdomain="dc=sun,dc=org" maxtime="300" maxidle="120"
maxcaching="3" timeidle="3" timeleft="17983" state="destroyed">
...
```

4. セッション通知の受信後に、ポリシーエージェントはキャッシュからセッションをフラッシュします。これで、セッションのライフサイクルは終了します。

Active Directory に対する認証

Sun Java™ System Access Manager では、さまざまなバックエンドソースに対して認証を実行することが可能です。主に使用されるのは Directory Server および LDAP ですが、LDAP 認証モジュールを設定することで Microsoft Active Directory を認証ソースとして使用できます。この付録は、次の節で構成されています。

注 Access Manager 6 2005Q1 は Active Directory を標準機能としてサポートしています。詳細については、『Sun Java System Access Manager 管理ガイド』(http://docs.sun.com/doc/819-1938?l=ja) を参照してください。

この付録の情報は、以前の Access Manager リリースでの Active Directory の実装のために含まれています。

概要

Active Directory は LDAPv3 に準拠したディレクトリサーバーであるため、Access Manager を設定するだけで Active Directory を認証ソースとして認識させることが可能です。この設定を定義する場合には、2 つの選択肢を考慮します。管理者は、この目的のために Active Directory に対するすべての LDAP 認証か、新規の認証モデルのいずれかを作成して、登録できます。

注 LDAP v3 に準拠するすべてのサーバーに対し、LDAP 認証モジュールをここで説明する方法で使用できます。

既存の LDAP 認証モジュールを指し示す

この方法では、LDAP 認証モジュールを使用して認証を試みるすべてのユーザーが Active Directory で認証されます。Sun Java System Directory Server で検証が行われるデフォルトの LDAP 認証とは、この点が異なります。この付録では、このシナリオについて主に説明します。

Active Directory 認証モジュールを新規作成する

別のオプションは、Active Directory に対して LDAP 認証を実行する、専用の認証モジュールを作成および登録する方法です。Access Manager では、1 つの認証モジュールの複数インスタンスを異なる複数のサーバ設定で使用することはできません。この手法を採用する場合は、新規クラスが必要になります。このオプションは、Active Directory サフィックスと Directory Server サフィックスが異なる場合に使用します。このオプションについて、ここでは扱いません。カスタム認証モジュールの作成方法については Access Manager Developer's Guide』を参照してください。

複数の LDAP サブ設定

管理者は、1 つの組織内に複数の LDAP 認証モジュール設定を定義できます。これらの追加設定はコンソールには表示されませんが、要求元のユーザーの認証が初期検索で見つからない場合に、プライマリ設定と連動して動作します。たとえば、ある組織で、2 つの異なるドメイン内で認証を検索できるように LDAP サーバーの検索を定義することも、1 つのドメイン内で複数のユーザーネーミング属性を設定することも可能です。後者の場合、コンソールにテキストフィールドが 1 つだけ表示されます。プライマリ検索条件を使用してユーザーが検出されない場合、LDAP モジュールは 2 番目のスコープを使用して検索を実行します。このオプションの詳細は、『Access Manager Developer's Guide』を参照してください。

Active Directory 認証の設定

以下に、LDAP 認証モジュールを使用して Active Directory に対して証明情報を確認する手順を示します。

1. Access Manager をインストールおよび設定します。

   以下に、LDAP 認証モジュールのデフォルト設定を変更する手順を示します。これは、潜在的な危険性の伴う操作であり、ロックアウトが発生する可能性があります。Access Manager へのアクセスが拒否された場合は、「トラブルシューティング」を参照してください。

2. 組織のコア認証サービスの「ユーザープロファイル」で、「ダイナミックに作成」を選択します。

   Access Manager では、外部ソースに認証が委任されていても、Directory Server 内部に認証用のアカウントが存在する必要があります。これには、次のオプションが存在します。

   - メタディレクトリを使用して、アカウントを同期させる。
   - ダイナミックプロファイル作成を有効にする。これにより、Access Manager は該当するユーザーのアカウントを検索できるようになる。該当するアカウントが存在しない場合は、Active Directory 内に同名のアカウントが自動的に作成される。

注 この属性の詳細および有効化の方法については、『Sun Java SystemAccess Manager 管理ガイド』を参照してください。

3. LDAP 認証モジュール内のプライマリ LDAP サーバーおよびポート属性を、Active Directory のホスト名およびポート番号に変更します。書式は、`hostname.domain.com:389` になります。
4. 「ユーザー検索の開始 DN」を、Active Directory 用の適切なベースに変更します。

   ベースサフィックスを指定するだけでは、この属性は機能しません。通常、`cn=Users` に `cn=Users,dc=domain,dc=com` 内のサフィックスを追加したものになります。

5. 「root ユーザーバインド DN」属性を、Active Directory の読み取り権限を持つユーザーに変更して、パスワードを更新します。

   Active Directory への匿名アクセスは許可されないため、バインドアカウントが必要になります。これは、ユーザーが認証する属性に対する読み取り機能を持つ Active Directory の単なるアカウントです。たとえば、`cn=administrator,cn=Users,dc=domain,dc=com` のようになります。このエントリのパスワードは、更新が必要です。

6. ユーザーネーミング属性を変更します。

   LDAP 認証モジュールは、この属性を Active Directory 内で検索し、検出した属性を使って Directory Server 内の UID を一致させます。たとえば、メタディレクトリにより 2 つのシステムの同期が行われ、Directory Server 内のエントリがメールを使用して RDN として格納される場合、Active Directory 内へのメールアドレスの格納時に `userPrinicipalName` をここに指定できます。この場合、`sAMAccountName` が Active Directory 内で最も UID に類似した属性であるため、これを指定するのが最善です。

7. 「ユーザーエントリ検索属性」を変更します。

   この属性は、Active Directory 内でのアカウントの検出に使用されます。これは、ユーザーがログインに使用する属性に対応している必要があります。たとえば、ユーザーが共通名を使用してログインする場合、この属性の値は `cn` になります。この場合、`sAMAccountName` が Active Directory 内で最も UID に類似した属性であるため、これを指定するのが最善です。

注 この属性には複数の値を含めることができ、それぞれの値を使用して実行が試みられます。

8. 「認証において ユーザー DN を返す」を選択解除します。

   この属性により、LDAP 認証モジュールは認証した DN を Access Manager に返すため、DN を再度検索して承認する手間を省くことができます。Active Directory の DN は、Directory Server の DN と同じではないため、オフにする必要があります。これにより、Access Manager が Active Directory から返される値を手順 6 の「ユーザーネーミング属性」で指定された属性として取得し、コア認証で指定された属性 (デフォルトでは `uid`) を使用して Directory Server を検索することが可能になります。この例では、LDAP 認証は Active Directory で `sAMAccountName=`*`UID_entered_by_user`* を検索し、認証後に Directory Server で Active Directory から返された `uid=sAMAccountName` を検索します。

9. `amAdmin` アカウントを Active Directory に追加します。

   `amAdmin` が Access Manager へのログインを続行できるように、`amAdmin` の `sAMAccountName` を使用してアカウントを作成する必要があります。このユーザーは、次のいずれかの方法で Active Directory 内で作成されます。

   a. 「スタート」-「プログラム」-「管理ツール」で、「Active Directory ユーザーとコンピュータ」を選択します。

   b. 左の区画で「ユーザー」ノードを右クリックして、「新規」-「ユーザー」を選択します。

   c. 適切な情報を入力し、アカウント名に `amAdmin` を指定します。

10. `amAdmin` が Access Manager にログインできることを確認します。

11. Active Directory で定義されたユーザーが Access Manager にログインできることを確認します。

これで、Access Manager が Active Directory での認証用に設定されました。

トラブルシューティング

LDAP 認証情報の変更により、ロックアウトが発生する可能性があります。以下に、この問題に対処する方法を示します。

Access Manager へのクイックアクセス

`AMConfig.properties` の `com.iplanet.authentication.super.user` プロパティ内で検出された DN を使用すると、`amAdmin` が LDAP 認証モジュールを介して Access Manager にログインできます。`amAdmin` は、Directory Server の Access Manager マネージャです。このプロパティの値は、`amAdmin` アカウントの完全な DN である、`uid=amAdmin,ou=People,`*root-suffix* です。完全な DN は、「ユーザー名」フィールドに入力します。この場合、`AMConfig.properties` で設定された Directory Server のインスタンスが使用されます。LDAP 認証モジュールパラメータは使用されません。

Directory Server を使用した再設定

認証設定情報は Directory Server に格納されるため、エントリは簡単に変更できます。`ou=default,ou=OrganizationConfig,ou=1.0,ou=iPlanetAMAuthLDAPService, ou=service,`*root-suffix* は、認証設定サービスを定義するオブジェクトです。iplanetKeyValue 属性を変更して、発生したすべてのエラーを訂正します。該当する値は次のとおりです。

- iplanet-am-auth-ldap-server
- iplanet-am-auth-ldap-base-dn
- iplanet-am-auth-ldap-user-naming-attribute
- iplanet-am-auth-ldap-user-search-attributes
- iplanet-am-auth-ldap-return-user-dn
- iplanet-am-auth-ldap-bind-dn
- iplanet-am-auth-ldap-bind-passwd

注 このプロパティの値は、ampassword ユーティリティを使用して暗号化されます。

付録 D

chroot 環境のインストール

Sun Java™ System Access Manager の chroot (changed root) 環境へのインストールは、Solaris™ Operating System の chroot 機能に基づいています。Access Manager を chroot 環境にインストールすると、指定されたディレクトリが新規のルートディレクトリになります。新規の chroot ディレクトリが、Access Manager のあらゆるインストールで使用されるパスの基点になります。

chroot 環境は、悪意のあるプログラムによる実際の root ファイルシステムへのアクセスを防ぎ、より安全な Access Manager 環境を実現する手段となります。

実際の配備の特別な要件によっては、chroot 環境の設定は複雑になります。chroot 環境に Access Manager をインストールして実行する必要がある場合には、Sun Microsystems の技術担当者に連絡してサポートを受けてください。

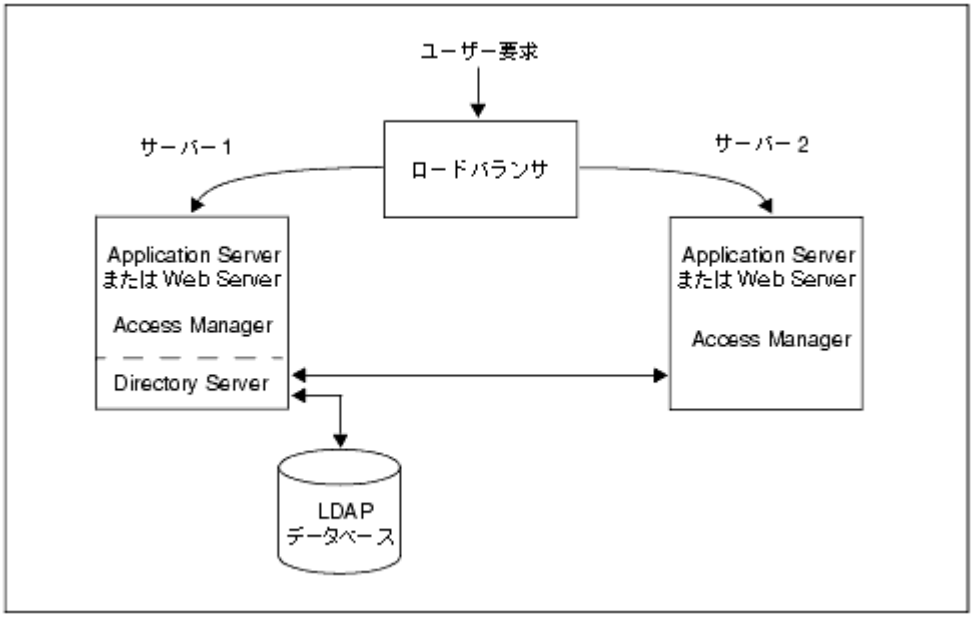
ロードバランサの設定

Sun Java™ System Access Manager は、ロードバランサを使用するように設定できます。この章では、ロードバランスの機能およびロードバランスを実現する方法について説明します。次の節で構成されています。

ロードバランサの概要

ロードバランスを使用すると、通常は 1 つのサーバーで実行される作業を複数のサーバー間で配分することで、同一時間内により多くの作業を処理できるようになります。一般に、これはすべてのユーザーの要求がより高速に処理されることを意味します。ロードバランスは、ハードウェア、ソフトウェア、またはその組み合わせで実現されます。図 E-1 に、ロードバランサで使用できるように Access Manager の配備を設定する方法を示します。この設定では、Access Manager のすべてのインスタンスが同一の Directory Server を共有することが重要です。設定が完了すると、ロードバランサ (およびすべての Access Manager サービス) が URL `http://`*loadbalancer_host.domain*`:`*port*`/amconsole` 経由でアクセスされます。

図 E-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

ロードバランサを Access Manager に配備する場合、ロードバランサが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により、指定されたサーバーによるセッションの作成後に、セッション情報を保持するため、ユーザーからの後続の要求が同一のサーバーに引き続き配信されます。Access Manager は Cookie を使用してセッション情報を中継するため、ロードバランサはセッションを作成したサーバーへのリダイレクトを実行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、すべてのサーバー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。

Resonate Central Dispatch のインストール

Resonate Central Dispatch は、ソフトウェアベースのロードバランサです。Access Manager をロードバランサで使用できるように設定する最初のステップは、ロードバランサのインストールです。2 つの物理サーバーが存在することを前提として、マシンが同一のサブネットに存在することを確認します。machine1 に Sun Java System Web Server、Sun Java System Directory Server、および Access Manager をこの順番で

インストールし、Access Manager のインスタンスが、インストール済みの Directory Server インスタンスを指し示すようにします。machine2 に Sun Java System Web Server および Access Manager をインストールし、Access Manager のインスタンスが、server1 にインストール済みの Directory Server インスタンスを指し示すようにします。Central Dispatch ソフトウェアは、次のようにインストールします。

- machine1 に CDNode、CDMaster、および CDAction が含まれる
- machine2 に CDNode、CDAdaptor、および CDAction が含まれる

インストール処理中に、Reporter Agent が両方のマシンに自動的にインストールされます。表 E-1 に、設定手順で使用される可能性のある Central Dispatch に固有の用語を示します。

表 E-1 Resonate Central Dispatch の定義済みの用語

Central Dispatch の用語	定義
CDMaster	Central Dispatch Master は、単一 (または複数) の Central Dispatch サイトの管理および監視に使用されるグラフィカルユーザーインタフェース。Central Dispatch の設定はすべて、このコンソールを使用して適用される。
ノード	CDMaster コンソールを介してノードとして設定された Access Manager のインスタンス。ノードは、スケジューラまたはサーバーとして設定可能です。
CDAdapter	Central Dispatch Adapter は、単一の Central Dispatch サイトと CDMaster 間のリンクを提供するプロキシ。
CDAction	CDAction は、Central Dispatch サイトの設定、監視、および管理に使用されるコマンド行ユーティリティ。

Central Dispatch のインストールおよび製品全般について詳しくは、ソフトウェアに同梱のドキュメントセットを参照してください。

ロードバランサの設定

「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 を使用して実装できます。以降の節で、両方のオプションに合わせてロードバランサを設定する手順を示します。説明する手順は Resonate Central Dispatch ロードバランサに適用されますが、任意のロードバランサソフトウェアで動作するように修正することも可能です。

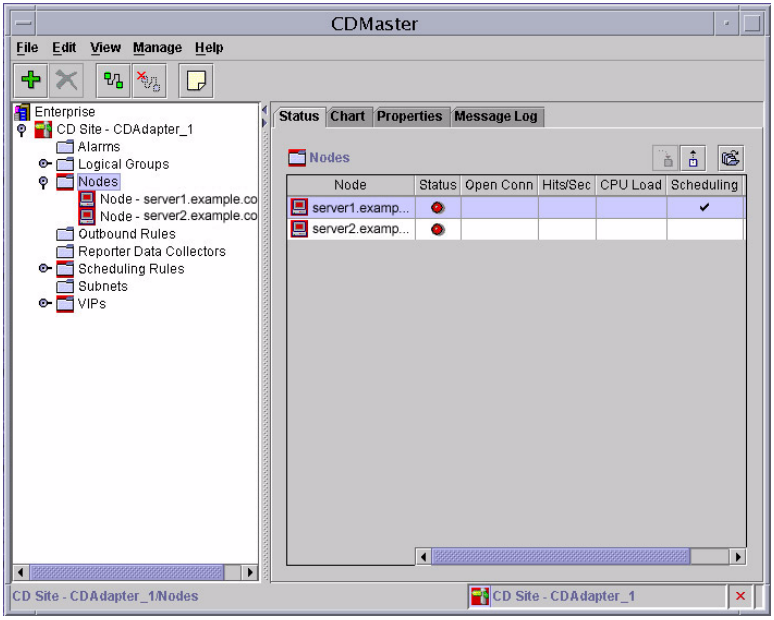
Central Dispatch を setcookie 用に設定する

1. admintool を使用して、2 つの Solaris ユーザー (cdadmin および cdmon) を作成します。
2. server1 上で CDMaster コンソールを起動します。

   デフォルトディレクトリ (/usr/local/resonate/cd/cdmaster/bin) に移動して、./cdmaster を実行します。指示に従って、machine2 にインストールされた CDAdapter に接続します。
3. CDMaster の左側のフレーム内の「Nodes」をクリックし、インストール済みの 2 つの Access Manager インスタンスのそれぞれに対して 1 つのノードを作成します。

   図 E-2 に、この手順の実行時の CDMaster コンソールのスクリーンショットを示します。

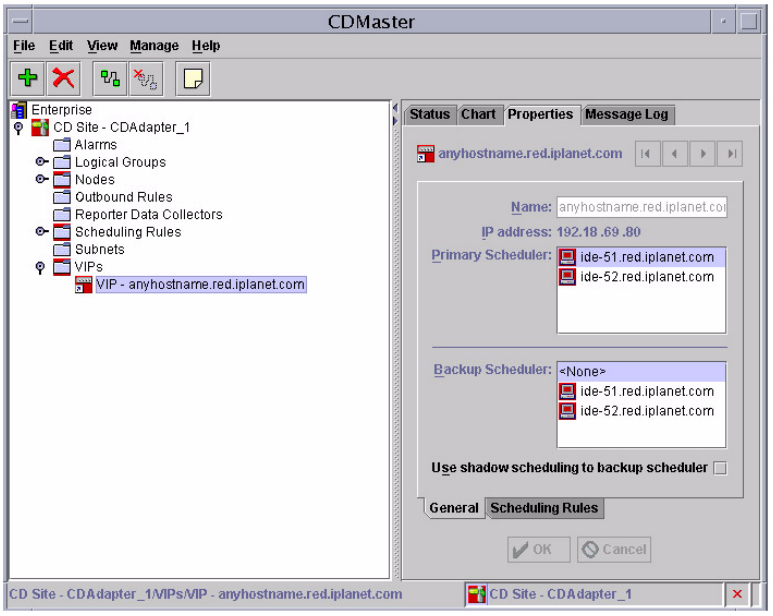
図 E-2　　Resonate を使用したノードの作成



4. CDMaster の左側のフレーム内で「VIPs」をクリックし、ロードバランサのインストール先ホスト新規仮想 IP アドレスを作成します。

   図 E-3 に、この手順および後続の手順を実行する際の CDMaster コンソールのスクリーンショットを示します。Primary Scheduler および Backup Scheduler が正しく設定されていることを確認してください。

図 E-3　　仮想 IP アドレスを新規作成する



5. 「VIPs」の右側のフレーム内の「Scheduling Rules」をクリックして「HTTP」を選択し、次のように HTTP スケジューリングルールを設定します。

   a. 「Properties」タブで、ロードバランサのインストール先ホストが仮想 IP として表示されていることを確認します。

   b. VIP ポートが、仮想 IP として定義されているものと同じであることを確認します。

   c. 「Resource」として「URL」を選択します。

6. CDMaster の左側のフレームの「Scheduling Rules」で、「HTTP」をクリックします。

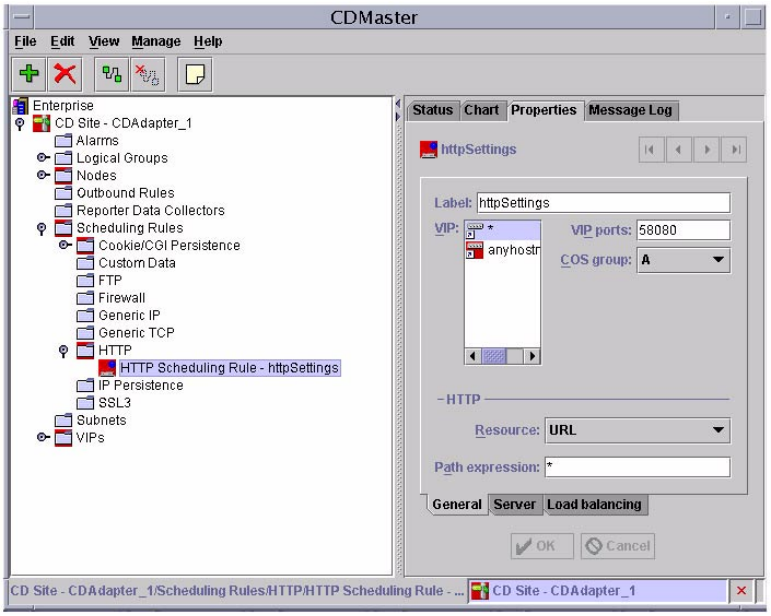
   171 ページの図 E-4 に、この手順および後続の手順を実行する際の CDMaster コンソールのスクリーンショットを示します。

7. 右側のフレーム下部の「Server」タブを選択します。

   サーバーが選択されていることを確認します。

8. 右側のフレーム下部の「Load Balancing」タブをクリックし、「Round Robin (Basic)」を選択します。

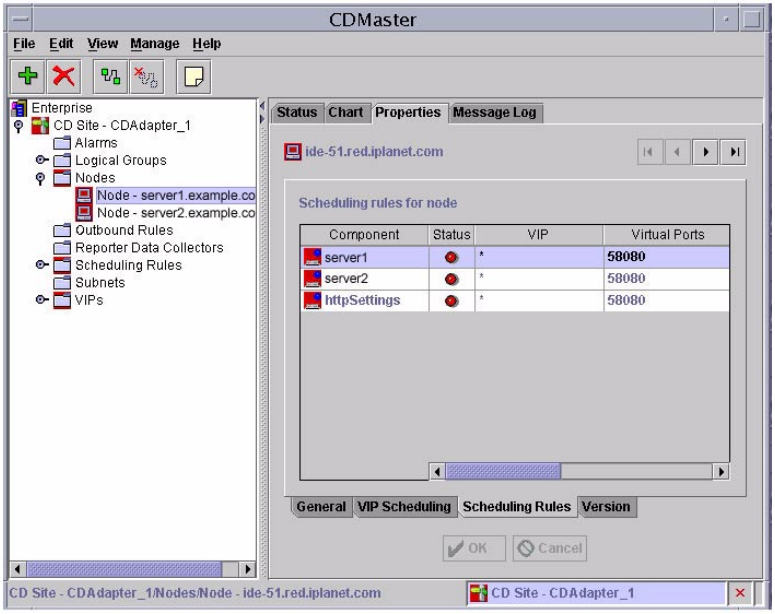
図 E-4　　HTTP スケジューリングルールを設定する



9. CDMaster の左側のフレームの「Nodes」をクリックし、Access Manager の 2 番目のインスタンスである server2 の設定済みノードを選択します。

   図 E-5 に、この手順および後続の手順を実行する際の CDMaster コンソールのスクリーンショットを示します。

図 E-5　　CDMaster でノードを設定する

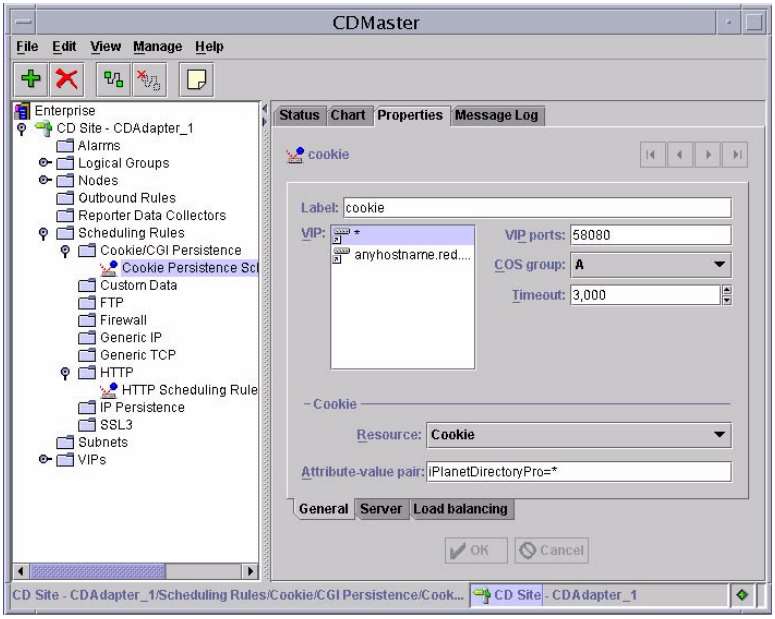


10. 右側のフレーム上部の「Properties」をクリックし、エイリアスが `server2.example.com` であり、`Server Enabled` および `Server auto enabled` が選択されていることを確認します。
11. 右側のフレーム下部の「VIP Scheduling」タブをクリックして、プライマリ仮想 IP がロードバランサのインストール先ホストに設定されていることを確認します。
12. 右側のフレーム下部の「Scheduling Rules」タブを選択して、すべてのサーバーおよびポートが「Component」に表示されることを確認します。
13. server1 に対して、171 ページの手順 9 から上記の手順 12 までを繰り返します。

    説明した手順に従って Access Manager の最初のインスタンス (server1) を設定します。ただし、手順 11 は、server2 を「Scheduling Rules」で「Primary」として設定する手順であるため、省略します。
14. CDMaster の左側のフレームの「Scheduling Rules」をクリックします。

    図 E-6 に、この手順および後続の手順を実行する際の CDMaster コンソールのスクリーンショットを示します。

図 E-6　　Cookie Persistence Scheduling Rule を設定する



15. 「Cookie/CGI Persistence」を選択して、Cookie Persistence Scheduling Rule を作成します。

    属性と値のペアが、`iPlanetDirectoryPro=*` として定義されます。

16. ルールにラベルを設定し、仮想 IP 用の適切なポートが定義されていることを確認します。

    VIP リストには、ロードバランサがインストールされた、設定済みのホストが含まれていなければなりません。

17. 右側のフレーム下部の「Server」タブを選択し、両方のサーバーにチェックが付けられていることを確認します。

18. 右側のフレーム下部の「Load Balancing」タブを選択し、「Round Robin (Basic)」を選択します。

これで、`setcookie` 用の Central Dispatch の設定が完了しました。引き続いて次の項「Access Manager を setcookie 用に設定する」に進み、配備を完成させてください。

Access Manager を setcookie 用に設定する

setcookie の使用時にロードバランサを認識できるように、server1 および server2 用の Access Manager の設定を更新する必要があります。

1. server1 にインストールされた Access Manager インスタンスに、amadmin としてログインします。
2. ロードバランサがインストールされたホストマシンの値を、「組織のエイリアス」属性に追加します。

   「アイデンティティ管理」タブで最上位の組織を表示して、「組織のエイリアス」属性を見つけます。
3. server2 を、「サービス設定」タブの「プラットフォーム」内にある「サーバーリスト」属性に追加します。
4. `AMConfig.properties` で fqdnMap プロパティを設定します。

警告 この手順を、Access Manager コンソールを使用して実行することはできません。

   デフォルトでは、fqdnMap プロパティはコメントアウトされています。# を削除して、次のようにプロパティを設定します。
   `com.sun.identity.server.fqdnMap[`*loadbalancer_host.domain*`]=`*loadbalancer_host.domain*
5. server1 および server2 を再起動します。

設定が正しく行われたことを確認する方法については、177 ページの「設定を確認する」を参照してください。

ロードバランサ Cookie の使用に合わせて Central Dispatch を設定する

1. `admintool` を使用して、2 つの Solaris ユーザー (`cdadmin` および `cdmon`) を作成します。
2. server1 上で CDMaster コンソールを起動します。

   デフォルトディレクトリ (`/usr/local/resonate/cd/cdmaster/bin`) に移動して、`./cdmaster` を実行します。指示に従って、server2 にインストールされた CDAdapter に接続します。
3. CDMaster の左側のフレーム内の「Nodes」をクリックし、インストール済みの 2 つの Access Manager インスタンスのそれぞれに対して 1 つのノードを作成します。

4. CDMaster の左側のフレーム内で「VIPs」をクリックし、ロードバランサのインストール先ホスト新規仮想 IP アドレスを作成します。

   Primary Scheduler および Backup Scheduler が設定されていることを確認してください。

5. 「VIPs」の右側のフレーム内の「Scheduling Rules」をクリックして「HTTP」を選択し、次のように HTTP スケジューリングルールを設定します。

   a. 「Properties」タブで、ロードバランサのインストール先ホストが仮想 IP として表示されていることを確認します。

   b. VIP ポートが、仮想 IP として定義されているものと同じであることを確認します。

   c. 「Resource」として「URL」を選択します。

6. CDMaster の左側のフレームの「Scheduling Rules」で、「HTTP」をクリックします。

7. 右側のフレーム下部の「Server」タブを選択します。

   サーバーが選択されていることを確認します。

8. 右側のフレーム下部の「Load Balancing」タブをクリックし、「Round Robin (Basic)」を選択します。

9. CDMaster の左側のフレームの「Nodes」をクリックし、Access Manager の 2 番目のインスタンスである server2 の設定済みノードを選択します。

10. 右側のフレーム上部の「Properties」をクリックし、エイリアスが `server2.example.com` であり、`Server Enabled` および `Server auto enabled` が選択されていることを確認します。

11. 右側のフレーム下部の「VIP Scheduling」タブをクリックして、プライマリ仮想 IP がロードバランサのインストール先ホストに設定されていることを確認します。

12. 右側のフレーム下部の「Scheduling Rules」タブを選択して、すべてのサーバーおよびポートが「Component」に表示されることを確認します。

13. server1 に対して、171 ページの手順 9 から上記の手順 12 までを繰り返します。

    説明した手順に従って Access Manager の最初のインスタンス (server1) を設定します。ただし、手順 11 は、server2 を「Scheduling Rules」で「Primary」として設定する手順であるため、省略します。

14. CDMaster の左側のフレームの「Scheduling Rules」をクリックします。

15. 「Cookie/CGI Persistence」を選択して、Cookie Persistence Scheduling Rule を server1 用に 1 つ、server2 用に 1 つ、合計 2 つ作成します。

16. server1 にラベルを設定し、仮想 IP 用の適切なポートが定義されていることを確認します。

    VIP リストには、ロードバランサがインストールされた、設定済みのホストも含まれていなければなりません。

17. 「Resource」として `cookie` を選択し、`Attribute-value pair` を `server1=server1` として定義します。

18. 右側のフレーム下部の「Server」タブを選択し、両方のサーバーが選択されていることを確認します。

19. 右側のフレーム下部の「Load Balancing」タブを選択し、「Round Robin (Basic)」を選択します。

20. server2 に対して、手順 14 から手順 19 までを繰り返します。

これで、ロードバランサ Cookie 用の Central Dispatch の設定が完了しました。引き続いて次の項「ロードバランサ Cookie に合わせて Access Manager を設定する」に進み、配備を完成させてください。

ロードバランサ Cookie に合わせて Access Manager を設定する

ロードバランサを認識できるように、server1 および server2 用の Access Manager の設定を更新する必要があります。

1. server1 にインストールされた Access Manager インスタンスに、amadmin としてログインします。
2. ロードバランサがインストールされたホストマシンの値を、「組織のエイリアス」属性に追加します。

   「アイデンティティ管理」タブで最上位の組織を表示して、「組織のエイリアス」属性を見つけます。
3. server2 を、「サービス設定」タブの「プラットフォーム」内にある「サーバーリスト」属性に追加します。
4. `AMConfig.properties` で fqdnMap プロパティを設定します。

警告 この手順を、Access Manager コンソールを使用して実行することはできません。

デフォルトでは、fqdnMap プロパティはコメントアウトされています。# を削除して、次のようにプロパティを設定します。
`com.sun.identity.server.fqdnMap[`*loadbalancer_host.domain*`]=`*loadbalancer_host.domain*

5. 次のプロパティを、server1 および server2 上の `AMConfig.properties` にそれぞれ追加します。

   a. server1 上の Cookie の名前および値を、次のように設定します。

      ```
      com.iplanet.am.lbcookie.name=server1
      com.iplanet.am.lbcookie.value=server1
      ```

   b. server2 上の Cookie の名前および値を、次のように設定します。

      ```
      com.iplanet.am.lbcookie.name=server2
      com.iplanet.am.lbcookie.value=server2
      ```

6. server1 および server2 を再起動します。

設定を確認する

次の手順で、設定が適正であることを確認します。

警告 以下の手順を実行する前に、Sun Java System Web Server の Web コンテナの `keepAliveTimeout` オプションを無効に設定してください。

1. コンソールの「Manage」タブ内の「Start」を選択して、CDMaster を起動します。
2. 複数の新規ユーザーを作成し、作成したユーザーでログインします。
3. Web ブラウザの「Location」バーに、`http://`*loadbalancer_host*`.`*domain*`:`*port*`/amconsole` と入力します。
4. Access Manager に amadmin でログインし、「現在のセッション」タブを選択します。

   amadmin として、作成したユーザーおよび対応するサーバーが表示可能になります。ユーザーは、セッションを開始したサーバーにすべてをリダイレクトする必要があります。これは、Web サーバーのアクセスログを使用しても確認できます。

ロードバランサ用の SSL ターミネーションの設定

この節では、ロードバランサを使用して、SSL 要求を処理できるように、Access Manager から返されるリダイレクト URL を設定する方法について説明します。この設定のこのシナリオには、次のコンポーネントが含まれます。

- 複数のサーバー上の、それぞれ Directory Server を指し示す Access Manager インスタンス。Access Manager 2003Q4 または 2004Q2 バージョンを使用している場合は、必要なサービスパックをインストールしていることを確認してください。
- Access Manager インスタンスを実行している各サーバーの Web コンテナ (Web Server または Application Server)。
- クライアント要求を Access Manager インスタンスに経路指定するロードバランサ。

ロードバランサ用に SSL ターミネーションを設定する

1. `amAdmin` として Access Manager にログインします。
2. Access Manager コンソールで、次のようにして、ロードバランサの DNS 名を組織のエイリアスに追加します。

   「アイデンティティ管理」タブで、「組織」を選択し、ロードバランサの DNS 名を「一般プロパティ」の下の「組織のエイリアス」に追加します。例を示します。

   ```
   lb-host.example.com
   ```

3. Access Manager コンソールで、次のようにして、ロードバランサをプラットフォームサーバーリストに追加します。

   「サービス設定」タブで、「プラットフォーム」をクリックし、「グローバル」の下の「サーバーリスト」にロードバランサを追加します。例を示します。

   ```
   lb-host.example.com
   ```

4. Access Manager インスタンスを実行している各サーバーの `AMConfig.properties` ファイルを編集します。まず、FQDN セクションを見つけて、次のように変更します。

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

   ロードバランサと Access Manager のプロトコルが異なる (http と https または https と http) 場合は、URL リダイレクト用の `com.sun.identity.url.redirect` プロパティを追加します。例を示します。

```
com.sun.identity.url.redirect=https,lb-host.example.com
```

ただし、ロードバランサと Access Manager ホストのプロトコルが同じ場合は、このプロパティを設定する必要はありません。

5. 各サーバーの Access Manager インスタンスを再起動します。

これで、ロードバランサは SSL 要求を処理し、それらを適切な Access Manager インスタンスに経路設定できるようになります。

RADIUS サーバーに対する認証

Sun Java™ System Access Manager は、RADIUS (Remote Authentication Dial-In User Service) サーバーに対してユーザーを認証できます。ここでは、その導入方法を説明します。次の節で構成されています。

概要

RADIUS は、認証および承認サービスの提供に使用される業界標準のプロトコルです。この種の認証では、クライアントである Access Manager は RADIUS 形式のメッセージを RADIUS サーバーに送信します。RADIUS サーバーは要求を認証および承認し、応答を RADIUS 形式で返します。

RADIUS サーバーの設定

管理者は、次の手順を実行し、RADIUS サーバーに対して Access Manager の認証をテストできます。

1. 認証のテストに使用するユーザーのエントリを RADIUS サーバーに追加します。

   次のユーザー情報を *`RADIUS_install`*`/etc/ raddb/users` に追加する必要があります。*Login-Host* は、Access Manager が稼働中のマシンのホストおよびドメインです。

コード例 F-1　RADIUS ユーザーのエントリ

```
"Sample_User1" Password == "Password"
User-Service-Type = Login-User,
Login-Host = access_manager_host.domain_name,
Login-Service = PortMaster
```

2. Access Manager の完全修飾ドメイン名 (FQDN) または IP アドレスを RADIUS サーバーに追加します。

   このクライアント情報は、*RADIUS_install*/etc/raddb/clients に追加されます。定義済みの共有「シークレット」も追加されることを確認してください。

コード例 F-2　RADIUS クライアントのエントリ

```
191.18.18.111           <secret>
ms.red.example.com       <secret>
```

3. *RADIUS_install/sbin* ディレクトリに移動し、次のコマンドを使用して RADIUS サーバーを再起動します。

   ```
   ./radiusd &
   ```

Access Manager の設定

1. Access Manager に amAdmin としてログインします。
2. 最上位レベルの組織に移動します。
3. ナビゲーションフレームの「表示」ドロップダウンメニューから「サービス」を選択します。
4. RADIUS が認証サービスに登録されていない場合、「追加」をクリックします。

   RADIUS が登録済みの場合、手順 6 に進みます。
5. サービスフレームから「RADIUS」を選択して、「了解」をクリックします。
6. ナビゲーションフレームで RADIUS プロパティの矢印をクリックします。

   テンプレートが作成されていない場合、作成します。

7. RADIUS サーバーの完全修飾ドメイン名または IP アドレスを、「RADIUS サーバー 1」フィールドに追加します。
8. 181 ページの「RADIUS サーバーの設定」の手順 2 で使用した共有シークレットを入力します。
9. RADIUS サーバーのポート番号を入力し、テンプレートの変更を保存します。

   デフォルトは 1645 です。
10. ナビゲーションフレームで、「コア」プロパティの矢印をクリックします。
11. 「組織認証モジュール」リストで「RADIUS」を選択し、変更を保存します。

警告 手順 11 で RADIUS を選択する際、LDAP の選択を解除しないようにしてください。

12. Access Manager コンソールからログアウトします。
13. URL `http://`*access_manager_host.domain_name:port*`/`*service_deploy_uri*`/UI/Login?module=RADIUS` で、`Sample_User1` としてログインします。

用語集

このマニュアルセットで使用する用語の一覧については、最新の『Sun Java™ Enterprise System 用語集』を参照してください。

http://docs.sun.com/doc/819-1933?l=ja

索引

A

B

C

D

I

J

L

M

N

R

S

W

あ

い

か

き

く

け

こ

さ

し

す

せ

そ

た

と

に

は

ふ

へ

ま

ら

れ

ろ