Oracle® OpenSSO Update 2 リリースノート

Beta

Part No: 821–2838
2010 年 7 月

101014@24999

目次

はじめに

OpenSSO 8.0 Update 2 リリースノートは、OpenSSO 8.0 Update 2 ソフトウェアのダウンロードおよびインストールに関する情報を記載しています。また、OpenSSO 8.0 Update 1 リリース以降のソフトウェアに対する変更点についての情報も記載しています。

対象読者

これらのリリースノートは、Oracle OpenSSO 8.0 をすでにインストールおよび配備している、企業の管理者および開発者を対象としています。読者は、主要な製品マニュアルに記載されている概念や手順をすでに理解している必要があります。

関連マニュアル

これらのリリースノートは、次の URL にある Oracle OpenSSO 8.0 の主要な製品マニュアルを補完するものです: http://docs.sun.com/app/docs/coll/1767.1

関連するサードパーティー **Web** サイト

このドキュメントでは、サードパーティー URL を参照して、追加の関連情報を提供します。

注 – Oracle は、このマニュアルに記載されているサードパーティー Web サイトの利用について責任を負いません。Oracle は、このようなサイトまたはリソースで得られるあらゆる内容、広告、製品、およびその他素材を保証するものではなく、責任または義務を負いません。Oracle は、これらのサイトまたはリソースから利用できるこのようなコンテンツ、商品、またはサービスを使用または信用した結果、またはそれに関連して生じた、または生じたと主張する損害または損失に対して、実際のものか主張されたものかにかかわらず、責任も義務も負わないものとします。

ドキュメント、サポート、トレーニング

その他のリソースについては次の Web サイトを参照してください。

- ドキュメント (http://docs.sun.com)
- サポート (http://www.oracle.com/us/support/systems/index.html)
- トレーニング (http://education.oracle.com) – 左のナビゲーションバーにある Sun リンクをクリックしてください。

ご意見をお寄せください

Oracle では、マニュアルの品質や有用性についてお客様からのコメントやご提案をお待ちしております。内容に誤りを見つけた場合や改善のためのその他のご提案がある場合は、http://docs.sun.com にアクセスして「Feedback」をクリックしてください。マニュアルのタイトルとパート番号のほかに、該当する場合は章、節、ページ番号もお書き添えください。お返事をご希望の場合はその旨お知らせください。

Oracle Technology Network (http://www.oracle.com/technetwork/index.html) では、Oracle ソフトウェアに関する様々なリソースを提供しています。

- 技術的な問題やその解決方法については、Discussion Forums (http://forums.oracle.com) でディスカッションできます。
- 段階を追って進められる実践的なチュートリアルは、Oracle By Example (http://www.oracle.com/technology/obe/start/index.html) に用意されています。
- サンプルコード? (http://www.oracle.com/technology/sample_code/index.html)をダウンロードします。

書体の表記規則

次の表は、本書で使用する表記上の規則について説明しています。

表 P–1 書体の表記規則

字体または記号	意味	例
AaBbCc123	コマンド名、ファイル名、ディレクトリ名、画面上のコンピュータ出力を示します。	.login ファイルを編集します。 すべてのファイルを一覧表示するには、ls -a を使用します。 machine_name% you have mail.
AaBbCc123	ユーザーが入力する文字を、画面上のコンピュータ出力とは区別して示します。	machine_name% **su** Password:

表 P–1 書体の表記規則 (続き)

字体または記号	意味	例
aabbcc123	変数を示します。実際の名前または値で置き換えます。	ファイルを削除するコマンドは、`rm` *filename* です。
AaBbCc123	書名、新しい用語、強調する語句を示します。	『ユーザーズガイド』の第 6 章を参照してください。 キャッシュは、ローカルに保存されたコピーです。 ファイルを保存しないでください。 注意: 一部の強調語句は、オンラインでは太字で示されます。

コマンドのシェルプロンプトの例

次の表は、Oracle Solaris OS に含まれているシェルのデフォルトの UNIX システムプロンプトおよびスーパーユーザープロンプトを示しています。コマンドの例に表示されているデフォルトのシステムプロンプトが Oracle Solaris リリースによって異なることに注意してください。

表 P–2 シェルプロンプト

シェル	プロンプト
Bash シェル、Korn シェル、および Bourne シェル	`$`
Bash シェル、Korn シェル、および Bourne シェル (スーパーユーザーの場合)	`#`
C シェル	`machine_name%`
C シェル (スーパーユーザーの場合)	`machine_name#`

◆ ◆ ◆ 第 1 章

1 OpenSSO 8.0 Update 2 について

この章は次のトピックで構成されています。

OpenSSO 8.0 Update 2 の新機能

OpenSSO 8.0 Update 2 には、セキュリティートークンサービスおよび OpenSSO Fedlet の機能拡張が含まれています。

セキュリティトークンサービスの機能拡張

セキュリティートークンサービスに次の新機能が追加されました。

- 特定の Web サービスプロバイダのセキュリティートークンを生成するため、TokenType をサポートします。
- 要求者としての X509 およびユーザー名のセキュリティートークンに、非対称バインディングとトランスポートバインディングの両方をサポートします。
- OpenSSO STS が SSL 経由のユーザー名で設定された場合の、ユーザー名セキュリティートークンを使用した SSL/トランスポートバインディングを強化します。
- useKey を使用して、非対称 KeyType の SAML Holder-of-Key セキュリティートークンを、Web サービスクライアントの公開鍵および Web サービスクライアントの X509 セキュリティートークンとして発行します。
- WSDL は、セキュリティートークンの設定に基づいて動的に更新されます。
- Web サービスプロバイダの公開鍵による暗号化をサポートします。

- 静的なユーザー名およびパスワードを設定ストアに格納する前に暗号化します。
- WS-Trust 要求を通じて、代理のセキュリティートークンとしての UserName トークンをサポートします。
- SAML ベアラートークンの発行をサポートします。
- 新しい Web サービスセキュリティー認証モジュール WSSAuth がダイジェストパスワードの検証をサポートします。
- 新しい OAMAuth 認証モジュールは、Oracle Access Manager と OpenSSO を使用してシングルサインオンを実現します。

詳細については、第 3 章「セキュリティートークンサービスの使用」を参照してください。

Fedlet の機能拡張

Fedlet に次の新機能が追加されました。

- .NET Fedlet での暗号化をサポートします
- .NET Fedlet へのサインインをサポートします
- .NET Fedlet は新たにシングルログアウトをサポートします
- .NET Fedlet は、サービスプロバイダ開始のシングルサインオンおよびアーティファクトをサポートします
- .NET Fedlet において複数のアイデンティティープロバイダおよびアイデンティティープロバイダのディスカバリサービスをサポートします
- Fedlet のプロパティーおよび設定ファイルの中でバージョン情報を表示します
- 新しいパスワード SPI の実装
- 属性クエリーをサポートします
- シングルログアウトをサポートします

詳細については、第 4 章「Oracle OpenSSO Fedlet の使用」を参照してください。

OpenSSO 8.0 Update 2 のハードウェアおよびソフトウェア要件

『Sun OpenSSO Enterprise 8.0 Update 1 Release Notes』の「Hardware and Software Requirements For OpenSSO Enterprise 8.0 Update 1」を参照してください

新しい Web コンテナのサポート

OpenSSO 8.0 Update 2 は、『Sun OpenSSO Enterprise 8.0 Update 1 Release Notes』の「Support for New Web Containers」で説明されている Web コンテナと、次の新しい Web コンテナをサポートします。

- Oracle WebLogic Server 10g リリース 3 (10.3)

OpenSSO 8.0 Update 2 の問題点と回避方法

CR 6959610: OpenSSO 8.0 Update 2 のサンプルを本稼働環境から削除しなければならない

OpenSSO 8.0 Update 2 のサンプルは潜在的なセキュリティー問題を引き起こす可能性があります。

回避方法。OpenSSO 8.0 Update 2 を本稼働環境に配備する場合は、潜在的なセキュリティー問題を防止するためにサンプルを削除します。

CR 6964648: WebLogic Server 10.3.3 に新しい Java セキュリティー権限が必要

OpenSSO 8.0 Update 2 を、セキュリティーマネージャーを有効にした Oracle WebLogic Server 10.3.3 に配備する場合、追加の Java セキュリティー権限が必要です。

回避方法。WebLogic Server 10.3.3 の `weblogic.policy` ファイルに次の権限を追加します。

```
permission java.lang.RuntimePermission "getClassLoader";
```

CR 6939443: WebLogic Server 10.3.x で、LDAP チェックまたは OCSP チェックを使用した証明書認証が失敗する

Oracle WebLogic Server 10.3.0 や 10.3.1 などの前バージョンにあった問題のため、LDAP チェックまたは OSCP チェックのいずれかを有効にした証明書認証が失敗します。

回避方法。この問題は WebLogic Server 10.3.3 で修正されました。LDAP チェックまたは OSCP チェックのいずれかを使用した証明書認証を利用するには、OpenSSO Update 2 と WebLogic Server 10.3.3 を使用します。

CR 6967026: コンフィギュレータが GlassFish 2.1.x から LDAPS を有効にしたディレクトリサーバーインスタンスに接続できない

GlassFish Enterprise Server v2.1.1 または v2.1.2 が OpenSSO 8.0 Update 2 の Web コンテナとして配備されている場合、コンフィギュレータは LDAPS を有効にしたディレクトリサーバーインスタンスに接続できません。

回避方法。GlassFish を Web コンテナとして LDAPS を有効にしたディレクトリサーバーを利用するには、GlassFish Enterprise Server v2.1 を配備します。

CR 6948937: WebLogic Server 10.3.3 の管理コンソールで OpenSSO 8.0 Update 2 を有効にすると例外が発生する

OpenSSO 8.0 Update 2 (`opensso.war`) を WebLogic Server 10.3.3 の管理コンソールに配備し、「Start」をクリックして OpenSSO 8.0 Update 2 で要求の受信を開始すると、WebLogic Server ドメインが起動したコンソールで例外が発生します。

注: OpenSSO 8.0 Update 2 を起動したあとは、ふたたび停止して再起動するまでは起動状態が保たれ、例外が発生することもありません。

回避方法。OpenSSO 8 Update 2 の `opensso-client-jdk15.war` ファイルから `saaj-impl.jar` ファイルを WebLogic Server 1.0.3.3 の設定 `endorsed` ディレクトリにコピーします。次にその手順を示します。

1. Oracle WebLogic Server 10.3.3 ドメインを停止します。

2. 必要であれば、OpenSSO 8.0 Update 2 の opensso.zip ファイルを解凍します。
3. 一時的なディレクトリを作成して、*zip-root*/opensso/samples/opensso-client.zip ファイルをそのディレクトリに解凍します。*zip-root* は opensso.zip ファイルを解凍した場所です。例:

```
cd zip-root/opensso/samples
mkdir ziptmp
cd ziptmp
unzip ../opensso-client.zip
```

4. 一時的なディレクトリを作成して、opensso-client-jdk15.war から saaj-impl.jar ファイルを抽出します。例:

```
cd zip-root/opensso/samples/ziptmp/war
mkdir wartmp
cd wartmp
jar xvf ../opensso-client-jdk15.war WEB-INF/lib/saaj-impl.jar
```

5. WEBLOGIC_JAVA_HOME/jre/lib ディレクトリの下に endorsed という名前のディレクトリを作成します (endorsed がまだ存在していない場合)。WEBLOGIC_JAVA_HOME は、WebLogic Server が使用するように設定した JDK です。
6. saaj-impl.jar ファイルを WEBLOGIC_JAVA_HOME/jre/lib/endorsed ディレクトリにコピーします。
7. WebLogic Server ドメインを起動します。

CR 6959373: updateschema スクリプトを実行したあと、Web コンテナの再起動が必要になる

updateschema.sh スクリプトまたは updateschema.bat スクリプトを実行したあと、OpenSSO 8.0 Update 2 の Web コンテナを再起動する必要があります。

CR 6961419: updateschema.bat スクリプトの実行にパスワードファイルが必要になる

updateschema.bat スクリプトは、複数の ssoadm コマンドを実行します。したがって、Windows システムで updateschema.bat を実行する前に、amadmin ユーザーのパスワードユーザーを平文で記載したパスワードファイルを作成しておく必要があります。 updateschema.bat スクリプトを実行すると、パスワードファイルへのパスを入力するよう求められます。 スクリプトは終了する前にパスワードファイルを削除します。

OpenSSO 8.0 Update 2 のマニュアル

このマニュアル以外にも、次のコレクションからその他の OpenSSO 8.0 マニュアルを入手できます。

http://docs.sun.com/coll/1767.1

ドキュメントに関する問題

OpenSSO 8.0 Update 2 には、次に示すドキュメントに関する問題が含まれています。

CR 6958580: コンソールのオンラインヘルプにサポートされていない Discovery Agent が記載されている

OpenSSO 8.0 Update 2 の管理コンソールのオンラインヘルプには、サポートされてないディスカバリエージェントについて記載されています。

回避方法。なし。オンラインヘルプに記載されているディスカバリエージェントに関する情報を無視します。

CR 6967006 コンソールのオンラインヘルプでは、OAMAuth および WSSAuth の認証モジュールについて記載されていない

OpenSSO 8.0 Update 1 の管理コンソールのオンラインヘルプでは、Oracle Access Manager (OAM) および Web Service Security (WSS) の認証モジュールについて記載されていません。

回避方法。これらの認証モジュールについての詳細は、第 3 章「セキュリティートークンサービスの使用」を参照してください

CR 6953582: Fedlet の Java API リファレンスは公開されなければならない

Fedlet の Java API 公開リファレンスは、『Oracle OpenSSO 8.0 Update 2 Java API Reference』の一部として利用可能です。次のドキュメントコレクションから入手可能です。http://docs.sun.com/coll/1767.1。

注: この Java API リファレンスには getPolicyDecisionForFedlet メソッドが含まれていますが、OpenSSO 8.0 Update 2 はこのメソッドをサポートしていません。

CR 6953579: OpenSSO Fedlet の README ファイルには、シングルログアウト機能について記載がない

Fedlet の README ファイルには、シングルログアウト機能について記載がありません。

回避方法。Oracle OpenSSO 8.0 Update 2 の場合、Fedlet のシングルログアウト機能は第 4 章「Oracle OpenSSO Fedlet の使用」に記載されています。

その他の情報とリソース

次の場所から、その他の有用な情報やリソースを確認できます。

- 17 ページの「非推奨事項の通知」
- 18 ページの「問題の報告とフィードバックの提供方法」
- 18 ページの「障害を持つ方々向けのアクセシビリティー情報」
- 18 ページの「関連するサードパーティーの Web サイト」

- Oracle Advanced Customer Services for Systems:

  http://www.oracle.com/us/support/systems/advanced-customer-services/index.html
- ソフトウェア製品: http://www.oracle.com/us/sun/sun-products-map-075562.html
- SunSolve: http://sunsolve.sun.com/
- Sun Developer Network (SDN): http://developers.sun.com/
- Sun 開発者サービス:http://developers.sun.com/services/

非推奨事項の通知

- サービス管理サービス (SMS) API (com.sun.identity.sm パッケージ) および SMS モデルは、将来のリリースの OpenSSO には含まれなくなります。
- Unix 認証モジュールおよび Unix 認証ヘルパー (amunixd) は、将来のリリースの OpenSSO には含まれなくなります。
- 『Sun Java System Access Manager 7.1 リリースノート』では、一般に Access Manager SDK (AMSDK) と呼ばれる Access Manager com.iplanet.am.sdk パッケージ、および関連するすべての API と XML テンプレートが将来のリリースの OpenSSO に含まれなくなると述べています。

  AMSDK が削除されると、その結果としてレガシーモードのオプションおよびサポートも削除されます。

移行オプションは現時点で利用できません。また、将来利用可能になる予定もありません。Oracle Identity Manager に、AMSDK の代わりに利用できるユーザープロビジョニングソリューションが用意されています。Identity Manager の詳細については、http://www.oracle.com/products/middleware/identity-management/identity-manager.html を参照してください。

問題の報告とフィードバックの提供方法

OpenSSO 8.0 Update 2 または今後のパッチに関する質問や問題は、http://sunsolve.sun.com/ のサポートリソースまでご連絡ください。

このサイトは、メンテナンスプログラムやサポートの連絡先番号だけでなく、ナレッジベース、オンラインサポートセンター、Product Tracker へのリンクもあります。サポートを依頼する場合は、次の情報をご用意ください。

- 問題の説明。 問題がいつ起きたか、状況や業務への影響など
- マシンのタイプ、オペレーティングシステムのバージョン、Web コンテナとそのバージョン、JDK のバージョン、OpenSSO のバージョンに加え、問題に関係する可能性があるすべてのパッチ、その他のソフトウェア
- 問題を再現するための手順
- エラーログまたはコアダンプ

障害を持つ方々向けのアクセシビリティー情報

このメディアの出版以降にリリースされたアクセシビリティー機能を入手するには、米国リハビリテーション法 508 条に関する製品評価資料を請求し、その内容を確認して、どのバージョンが、アクセシビリティーに対応したソリューションを配備するためにもっとも適しているかを特定してください。

アクセシビリティーに関する Oracle の取り組みについては、http://www.oracle.com/index.html をご覧ください。

関連するサードパーティーの Web サイト

このドキュメントでは、サードパーティー URL を参照して、追加の関連情報を提供します。

注 – Oracle は、このマニュアルに記載されているサードパーティー Web サイトの利用について責任を負いません。Oracle は、このようなサイトまたはリソースで得られるあらゆる内容、広告、製品、およびその他素材を保証するものではなく、責任または義務を負いません。Oracle は、このようなサイトまたはリソースで得られるあらゆるコンテンツ、製品、またはサービスによって生じる、または使用に関連して生じる、または信頼することによって生じる、いかなる損害または損失についても責任または義務を負いません。

◆ ◆ ◆ 第 2 章

2 OpenSSO 8.0 Update 2 のインストール

この章は次のトピックで構成されています。

OpenSSO 8.0 Update 2 のインストールの概要

OpenSSO 8.0 Update 2 は、パッチ **TBS** として入手可能です。

OpenSSO 8.0 Update 2 (または以降のパッチ) をインストールする前に、このマニュアルに記載されている新機能、ハードウェアとソフトウェアの要件、および問題点と回避方法についての情報を確認してください。

OpenSSO 8.0 Update 2 には、次の方法でインストールできる `opensso.war` ファイルが含まれています。

- 既存の **OpenSSO 8.0** 配備へのパッチ適用: Update 2 の `ssopatch` ユーティリティーを使用して、既存の OpenSSO 8.0 配備にパッチを適用します。手順はこの章で解説しています。

注: Oracle では OpenSSO 8.0 リリースのパッチ適用のみをサポートしています。たとえば、OpenSSO 8.0 に OpenSSO 8.0 Update 2 のパッチを適用することはサポートされています。

- 新規 **OpenSSO 8.0 Update 2** 配備のインストール**:** OpenSSO 8.0 Update 2 の `opensso.war` ファイルをインストールして設定します。手順は『Sun OpenSSO Enterprise 8.0 Installation and Configuration Guide』で解説しています。
- 特別な **WAR** ファイルの新規作成: `createwar` スクリプトを使用して、Update 2 の `opensso.war` ファイルから次の新規 WAR ファイルのいずれかを作成します。
  - OpenSSO 管理コンソール専用 WAR
  - 分散認証 UI サーバーの WAR
  - OpenSSO サーバー専用 WAR (管理コンソール以外)
  - IDP のディスカバリサービスの WAR

    詳細については、『Sun OpenSSO Enterprise 8.0 Update 1 Release Notes』の第 4 章「Creating a Specialized OpenSSO Enterprise 8.0 Update 1 WAR File」を参照してください。
- 既存の特別な **OpenSSO WAR** ファイルの適用: Update 2 の `ssopatch` ユーティリティーを使用して、既存の特別な OpenSSO 8.0 WAR ファイルのパッチを適用します。この手順は『Sun OpenSSO Enterprise 8.0 Installation and Configuration Guide』の第 23 章「Patching OpenSSO Enterprise 8.0」で解説しています

注 – Access Manager 7.1 または Access Manager 7 2005Q4 を実行していて Update 2 にアップグレードする場合、次の手順に従います。

1. 『Sun OpenSSO Enterprise 8.0 Upgrade Guide 』の手順に従って Access Manager 7.x から OpenSSO 8.0 にアップグレードします。
2. この章の手順に従って Update 2 のパッチを適用します。

OpenSSO 8.0 Update 2 のパッチ

Sun では、OpenSSO 8.0 Update 2 用のパッチを定期的にリリースしています。これらのパッチに関する情報は、定期的にここを確認してください。

パッチの運用計画

▼ OpenSSO 8.0 用のパッチを運用する計画を立てる

1 **23 ページの「ssopatch ユーティリティーの概要」**を確認します。

2. 24 ページの「ssopatch ユーティリティーのインストール」の手順に従って、使用中のプラットフォームに適したパッチユーティリティーをインストールします。
3. 26 ページの「OpenSSO WAR ファイルとその内部マニフェストとの比較」の手順に従って、既存の WAR ファイルに関する情報を取得して、既存の WAR ファイルがカスタマイズまたは変更されているかを判定します。
4. 27 ページの「2 つの OpenSSO WAR ファイルの比較」の手順に従って、既存の WAR ファイルと Update 2 の WAR ファイルを比較して、もとの WAR ファイルでカスタマイズされたファイル、新しい WAR ファイルで更新されたファイル、および 2 つの WAR バージョン間で追加または削除されたファイルを返します。
5. 25 ページの「OpenSSO WAR ファイルのバックアップ」の手順に従って、既存の OpenSSO WAR ファイルをバックアップおよびアーカイブします。
6. 28 ページの「OpenSSO WAR ファイルのパッチ適用」の手順に従って、OpenSSO WAR ファイルをパッチ適用します。
7. 31 ページの「updateschema スクリプトの実行」の手順に従って、updateschema スクリプトを実行します。

   注: OpenSSO サーバー専用、管理コンソール専用、分散認証 UI サーバー、IDP のディスカバリサービス用の WAR など、opensso.war から生成した特別な WAR ファイルにパッチ適用する場合は、30 ページの「特別な OpenSSO WAR へのパッチ適用」を参照してください。

ssopatch ユーティリティーの概要

ssopatch ユーティリティーは、Solaris および Linux では ssopatch として、Windows では ssopatch.bat として利用できる、Java のコマンド行ユーティリティーです。

注: OpenSSO 8.0 Update 2 での ssopatch の構文は、OpenSSO 8.0 のリリースから大幅に変更されています。新しい構文については、31 ページの「updateschema スクリプトの実行」を参照してください。

ssopatch パッチユーティリティーは、次の機能を実行します。

- OpenSSO WAR をもとのマニフェストと比較して、WAR ファイルがカスタマイズまたは変更されたかを判定します
- 2 つの OpenSSO WAR ファイルを比較して、もとの WAR ファイルに加えられたカスタマイズと新しい WAR ファイルに加えられた変更を含む、2 ファイル間の相違を判定します
- パッチ適用された OpenSSO WAR ファイルを新規作成するために必要なファイルのステージング領域を生成します

OpenSSO 8.0 Update 2 の ZIP ファイル (opensso_80U2.zip) をダウンロードして解凍したら、*zip-root*/opensso/tools ディレクトリの ssoPatchTools.zip ファイルでパッチユーティリティーと関連するファイルが利用できます。*zip-root* は、opensso_80U2.zip を解凍した場所です。

ssopatch ユーティリティーは、マニフェストファイルを使用して特定の OpenSSO WAR ファイルの内容を判定します。マニフェストファイルとは、次の内容を含む ASCII テキストファイルです。

- OpenSSO WAR ファイルの特定バージョンを識別する文字列
- OpenSSO WAR ファイル内にある個別のファイルすべて (各ファイルのチェックサム情報を含む)

マニフェストファイルは一般に、OpenSSO.manifest という名前で、OpenSSO WAR ファイルの META-INF ディレクトリに格納されています。

ssopatch ユーティリティーは、その結果を標準出力 (stdout) に送信します。出力をファイルにリダイレクトすれば、ssopatch の出力を取得することもできます。ssopatch が正常に終了すると、終了コードのゼロ (0) を返します。エラーが発生した場合、ssopatch はゼロ以外の終了コードを返します。

ssopatch ユーティリティーのインストール

ssopatch ユーティリティーをインストールする前に、次の手順に従います。

- OpenSSO 8.0 Update 2 の ZIP ファイル (opensso_80U2.zip) をダウンロードして解凍します。
- JAVA_HOME 環境変数を変更して JDK 1.5 以降を指定するようにします。

ssopatch ユーティリティーをインストールする

1. *zip-root*/opensso/tools ディレクトリの中にある ssoPatchTools.zip ファイルを見つけます。*zip-root* は、opensso_80U2.zip を解凍した場所です。
2. ssoPatchTools.zip ファイルを解凍するための新しいディレクトリを作成します。例: ssopatchtools
3. ssoPatchTools.zip ファイルをこの新しいディレクトリに解凍します。
4. フルパスを指定せずにカレントディレクトリ以外のディレクトリから ssopatch ユーティリティーを実行する場合は、ユーティリティーを PATH 変数に追加します。

次の表に、ssoPatchTools.zip 内のファイルを示します。

ファイルまたはディレクトリ	説明
README	ssopatch の説明を記載した Readme ファイル
/lib	必要な ssopatch JAR ファイル
/patch	updateschema スクリプトと updateschema.bat スクリプト、および関連する XML ファイル
/resources	必要なプロパティーファイル
ssopatch および ssopatch.bat	Solaris、Linux、および Windows システム用のユーティリティー

OpenSSO WAR ファイルのバックアップ

開始する前に、既存の OpenSSO WAR ファイルおよび設定データを次の手順に従ってバックアップします。

- 既存の OpenSSO WAR ファイルを安全な場所にコピーします。その後、何らかの理由で Update 2 をバックアウトする必要がある場合は、この WAR ファイルのバックアップコピーを再配備できます。
- 『Sun OpenSSO Enterprise 8.0 Administration Guide』の第 15 章「Backing Up and Restoring Configuration Data」の手順に従って、設定データをバックアップします。

ssopatch ユーティリティーの実行

ssopatch ユーティリティーを実行するには、次の使用法に従います。

```
ssopatch
--help|-?
[--locale|-l]

ssopatch
--war-file|-o
[--manifest|-m]
[--locale|-l]

ssopatch
--war-file|-o
--war-file-compare|-c
[--staging|-s]
[--locale|-l]
[--override|-r]
[--overwrite|-w]
```

各オプションは次のとおりです。

- `-war-file|-o` は、以前に配備していた WAR ファイル (`opensso.war` など) へのパスを指定します。
- `-manifest|-m` は、作成するマニフェストファイルへのパスを指定します。このオプションを指定した場合、マニフェストファイルは `-war-file|-o` で指定された WAR ファイルから生成されます。
- `-war-file-compare|-c` は、`-war-file|-o` で指定された WAR ファイルと比較する WAR ファイルへのパスを指定します。
- `-staging|-s` は、OpenSSO WAR のファイルが書き込まれるステージング領域へのパスを指定します。
- `-locale|-l` は、使用するロケールを指定します。このオプションを指定しない場合、`ssopatch` はデフォルトのシステムロケールを使用します。
- `-override|-r` は 2 つの WAR ファイルのバージョンチェックをオーバーライドします。バージョンチェックは WAR ファイルのバージョンを判定して、互換性のあるバージョンである場合にのみ続行します。このオプションを指定すると、このチェックをオーバーライドできます。

  デフォルトは false (バージョンチェックを実行する) です。
- `-overwrite|-w` は、既存のステージング領域にあるファイルを上書きします。デフォルトは false (ファイルを上書きしない) です。

OpenSSO WAR ファイルとその内部マニフェストとの比較

この手順を実行して、OpenSSO WAR ファイルがダウンロードされてからカスタマイズまたは変更されたかを判定します。

`ssopatch` ユーティリティーは新しい内部マニフェストファイルを生成して、この内部マニフェストを `META-INF` ディレクトリにあるもとの OpenSSO WAR ファイル内に格納されているマニフェストと比較します。

OpenSSO WAR ファイルをその内部マニフェストと比較する

1. `ssopatch` を実行して OpenSSO WAR ファイルをその内部マニフェストと比較します。次に例を示します。

```
./ssopatch -o /zip-root/opensso/deployable-war/opensso.war
Generating Manifest for: /zip-root/opensso/deployable-war/opensso.war
Comparing manifest of Internal (Enterprise 8.0 Build 6(200810311055))
against /zip-root/opensso/deployable-war/opensso.war (generated-200905050855)
```

```
File not in original war (images/login-origimage.jpg)
File updated in new war (images/login-backimage.jpg)
File updated in new war (WEB-INF/classes/amConfigurator.properties)
Differences: 3
```

この例では、もとの WAR ファイルに次の変更が検出されました。

- `images/login-origimage.jpg` は `opensso.war` 内にありますが、もとのマニフェストにはありません。
- `images/login-backimage.jpg` が `opensso.war` でもとのマニフェストからカスタマイズされています。
- `WEB-INF/classes/amConfigurator.properties` ファイルが `opensso.war` でもとのマニフェストからカスタマイズされています。

2 つの OpenSSO WAR ファイルの比較

この手順を使用して 2 つの WAR ファイルを比較し、次のような変更が加えられたファイルを示します。

- もとの OpenSSO WAR 内でのカスタマイズ
- 新しい OpenSSO WAR ファイル内の更新
- 2 つの OpenSSO WAR バージョン間での追加または削除

2 つの OpenSSO WAR ファイルを比較する

1. `ssopatch` を実行して 2 つの WAR ファイルを比較します。この例では、次のように `-override` オプションを使用して、2 つの WAR ファイル間でのバージョンチェックをオーバーライドしています。

```
./ssopatch -o /zip-root/opensso/deployable-war/opensso.war
-c /u1/opensso/deployable-war/opensso.war --override
Generating Manifest for: /zip-root/opensso/deployable-war/opensso.war
Original manifest: Enterprise 8.0 Build 6(200810311055)
New manifest: Enterprise 8.0 Update 2 Build 6.1(200904300525)
Versions are compatible
Generating Manifest for: /u1/opensso/deployable-war/opensso.war
Comparing manifest of /zip-root/opensso/deployable-war/opensso.war
(generated-200905050919) against
    /u1/opensso/deployable-war/opensso.war (generated-200905050920)
File updated in new war(WEB-INF/classes/amClientDetection_en.properties)
File updated in new war(WEB-INF/classes/fmSAMLConfiguration_fr.properties)
...
Differences: 1821
Customizations: 3
```

この例では、新しい WAR ファイルでファイルの更新およびカスタマイズがあったことを示しています。

OpenSSO WAR ファイルのパッチ適用

この手順を実行して、もとの WAR ファイルを新しい WAR ファイルとマージする、新しいステージング領域を作成します。

この操作は各 WAR ファイルのマニフェストを比較して、次のファイルを表示します。

- もとの WAR ファイル内でカスタマイズされたファイル
- 新しい WAR ファイルで更新されたファイル
- 2 つの WAR ファイルのバージョン間で追加または削除されたファイル

ssopatch はその後、該当するファイルをステージング領域にコピーします。パッチ適用された WAR ファイルを新規作成して配備する前に、この領域にカスタマイズを追加する必要があります。

ステージング領域を作成して OpenSSO WAR ファイルにパッチ適用する

1. ssopatch はもとの opensso.war ファイルを変更しませんが、パッチが適用された opensso.war ファイルをバックアウトする必要がある場合に備えて、このファイルをバックアップすることをお勧めします。
2. ssopatch を実行してステージング領域を作成します。次に例を示します。

```
./ssopatch -o /zip-root/opensso/deployable-war/opensso.war
  -c /u1/opensso/deployable-war/opensso.war --override -s /tmp/staging
Generating Manifest for: /zip-root/opensso/deployable-war/opensso.war
Original manifest: Enterprise 8.0 Build 6(200810311055)
New manifest: Enterprise 8.0 Update 2 Build 6.1(200904300525)
Versions are compatible
Generating Manifest for: /u1/opensso/deployable-war/opensso.war
Comparing manifest of /zip-root/opensso/deployable-war/opensso.war
    (generated-200905051031) against /u1/opensso/deployable-war/opensso.war
    (generated-200905051032)
File was customized in original, but not found in new war.
Staging area using original war version (samples/saml2/sae/header.jsp)
File was customized in original, but not found in new war.
Staging area using original war version
    (WEB-INF/template/opends/config/upgrade/config.ldif.4517)
File was customized in original, but not found in new war.
Staging area using original war version
    (WEB-INF/template/opends/config/upgrade/schema.ldif.4517)
Differences: 1813
Customizations: 0
```

この例では /tmp/staging がステージング領域で、ssopatch がファイルをコピーする場所です。

前の手順の結果を使用して、ステージング領域のファイルを必要に応じて更新します。

パッチを適用したファイルを新規作成する前に、次の表を使って各ファイルに対して実行すべき対応を判断します。

ssopatch 結果	説明および必要な対応
`File not in original war` *filename*	表示されているファイルがもとの WAR ファイルに存在しないが、WAR ファイルの最新バージョンに存在します。 対応: なし
`File updated in new war` *filename*	表示されているファイルがもとの WAR ファイルと新しい WAR ファイルの両方に存在し、WAR ファイルの最新バージョンで更新されています。もとの WAR ファイルではカスタマイズされていません。 対応: なし
`File customized` *filename*	表示されているファイルが両方の WAR ファイルに存在し、WAR ファイルのもとのバージョンでカスタマイズされていますが、WAR ファイルの最新バージョンでは更新されていません。 対応: なし
`May require manual customization` *filename*	ファイルが両方の WAR ファイルに存在し、WAR ファイルのもとのバージョンでカスタマイズされていて、WAR ファイルの最新バージョンでも更新されています。 対応: ファイル内のカスタマイズが必要な場合は、ステージングディレクトリで新たに更新されたファイルにそのカスタマイズを手動で追加する必要があります。
`File was customized in original, but not found in new war`	ファイルがもとの WAR ファイルに存在しますが、新しい WAR ファイルには存在しません。 対応: なし。

次の手順

1. 新しい OpenSSO WAR ファイルをステージング領域のファイルから作成します。次に例を示します。

   ```
   cd /tmp/staging
   jar cvf /patched/opensso.war *
   ```

   `/patched/opensso.war` は、パッチ適用された新しい OpenSSO WAR ファイルです
2. もとの配備 URI を使用して、`/patched/opensso.war` ファイルを Web コンテナに再配備します。 たとえば、`/opensso` などです。

OpenSSO の設定変更。 新しい OpenSSO WAR ファイルは、もとの WAR ファイルにはない設定変更が加えられている可能性があります。変更された設定がある場合は、各パッチに個別に記録されます。設定変更に関する詳細については、パッチに

関するドキュメントと『Sun OpenSSO Enterprise 8.0 リリースノート』を確認してください。新しい WAR ファイルの設定が変更されない場合でも、OpenSSO マニフェストファイル内のバージョン文字列は変更されます。

パッチを適用したバージョンをバックアウトする必要がある場合、パッチを適用した WAR ファイルの配備を取り消してからもとの WAR ファイルを再配備します。

OpenSSO WAR マニフェストファイルの作成

OpenSSO マニフェストファイルは、特定リリースの WAR ファイル内にある個別のファイルすべてを識別するテキストファイルで、各ファイルのチェックサム情報が含まれています。

この手順を使用して、OpenSSO サーバー専用、管理コンソール専用、分散認証 UI サーバー、IDP のディスカバリサービス用の WAR など、特別な OpenSSO WAR を含めることができるマニフェストファイルを作成します

OpenSSO WAR マニフェストファイルを作成する

1. `ssopatch` を実行して OpenSSO マニフェストファイルを作成します。次に例を示します。

   ```
   ./ssopatch -o zip-root/opensso/deployable-war/opensso.war --manifest /tmp/manifest
   ```

   `opensso.war` は、既存の OpenSSO WAR ファイルです。

   `ssopatch` ユーティリティーは、`manifest` という名前の新しいマニフェストファイルを `/tmp` ディレクトリに作成します。

2. WAR ファイルにパッチを適用できるようにするためには、この新しいマニフェストファイルを `opensso.war` ファイル内にある `META-INF` ディレクトリにコピーします。次に例を示します。

   ```
   mkdir META-INF
   cp /tmp/manifest META-INF
   jar uf opensso.war META-INF/manifest
   ```

特別な OpenSSO WAR へのパッチ適用

OpenSSO サーバー専用、管理コンソール専用、分散認証 UI サーバー、IDP のディスカバリサービス用の WAR など、特別な OpenSSO WAR を以前に作成している場合、`ssopatch` ユーティリティーを使用してパッチを適用できます。

特別な OpenSSO WAR にパッチを適用する

1. 30 ページの「OpenSSO WAR マニフェストファイルの作成」の手順に従って、特別な OpenSSO WAR のマニフェストファイルを作成します。

   注: このファイルは、すでにカスタマイズをしている場合はそのカスタマイズ以前に Sun から提供されたもとの OpenSSO 8.0 の `opensso.war` を基にして作成します。カスタマイズしたあとにマニフェストを作成すると、`ssopatch` はカスタマイズのファイルではなく Update 2 のファイルを使用するため、パッチを適用したあとに再度カスタマイズする必要があります。
2. 『Sun OpenSSO Enterprise 8.0 Update 1 Release Notes』の第 4 章「Creating a Specialized OpenSSO Enterprise 8.0 Update 1 WAR File」の手順に従って、OpenSSO 8.0 Update 2 の `opensso.war` から特別な OpenSSO WAR を生成します。
3. `ssopatch` ユーティリティーを使用して古い WAR ファイルと新しい WAR ファイルを比較します。
4. 28 ページの「ステージング領域を作成して OpenSSO WAR ファイルにパッチ適用する」の手順に従って、新しい特別な WAR ファイル用のステージング領域を生成します。
5. 新しい特別な WAR ファイルを再配備します。

updateschema スクリプトの実行

`ssopatch` を実行したあと、Solaris または Linux の場合は `updateschema.sh` を、Windows の場合は `updateschema.bat` を実行します。このスクリプトは OpenSSO サーバーのバージョンを更新し、新しいデフォルトサーバープロパティーを追加し、Update 2 でのバグ修正や機能拡張に必要な新しい属性スキーマを追加します。サーバーのバージョンを更新するために、`updateschema` を実行する必要があります。

開始する前に

- `updateschema.sh` スクリプトまたは `updateschema.bat` スクリプトは Update 2 バージョン (またはそれ以降) の `ssoadm` コマンド行ユーティリティーが必要です。そのため、このスクリプトを実行する前に『Sun OpenSSO Enterprise 8.0 Update 1 Release Notes』の第 3 章「Installing the OpenSSO Enterprise 8.0 Update 1 Admin Tools」の手順に従って Update 2 の管理ツールをダウンロードします。
- `updateschema.bat` スクリプトは、複数の `ssoadm` コマンドを実行します。したがって、Windows システムで `updateschema.bat` を実行する前に、`amadmin` ユーザーのパスワードユーザーを平文で記載したパスワードファイルを作成しておく必要があります。 `updateschema.bat` スクリプトを実行すると、パスワードファイルへのパスを入力するよう求められます。 スクリプトは終了する前にパスワードファイルを削除します。

updateschema スクリプトを実行する

1. *patch-tools*/patch ディレクトリに移動します。*patch-tools* は ssoPatchTools.zip を解凍した場所です。
2. updateschema.sh または updateschema.bat を実行します。Solaris システムの例を次に示します。

   ```
   ./updateschema.sh
   ```
3. スクリプトから入力を求められたら、次の情報を指定します。
   - ssoadm ユーティリティーへのフルパス (ssoadm 自体は除く)。 例: /opt/ssotools/opensso/bin
   - amadmin パスワード

   updateschema.sh スクリプトまたは updateschema.bat スクリプトは、メッセージやエラーがあった場合に標準出力に書き出します。
4. OpenSSO 8.0 Update 2 の Web コンテナを再起動します。

パッチインストールのバックアウト

パッチインストールをバックアウトする必要がある場合は、もとの opensso.war ファイル (または特別な WAR ファイル) を再配備するだけです。

◆ ◆ ◆ 第 3 章

3 セキュリティートークンサービスの使用

信頼できる認証局のサービスとして、OpenSSO のセキュリティートークンサービスはセキュリティートークンを発行および検証します。Web サービスセキュリティープロバイダとして、セキュリティートークンサービスは Web サービスクライアントと OpenSSO STS サービス自身との間の通信をセキュリティー保護します。OpenSSO 8.0 Update 2 から、多くの機能拡張がセキュリティートークンサービスに追加されました。

この章は次のトピックで構成されています。

WSSAuth 認証モジュールの追加

Web サービスセキュリティーの認証モジュールを利用することで、OpenSSO は 認証トークンとして受信したダイジェストパスワード、および Web サービスクライアントから Web サービスプロバイダに送信されるサービス要求に含まれるダイジェストパスワードを使用して、UserName を検証できます。

▼ 新しい Web サービスセキュリティーの認証モジュールのインスタンスを追加する

1 「**Access Manager**」タブで、「**Authentication**」サブタブをクリックします。

2 「**Module Instances**」セクションで、「**New**」をクリックします。

3 「**Name**」フィールドで、この **WSSAuth** 認証モジュールのインスタンス名を入力します。

4 「**Type**」には「**WSSAuth**」を選択します。

5 **WWSAuth** 認証モジュールのインスタンスを設定します。

▼ WWSAuth 認証モジュールのインスタンスを設定する

1 「**Access Manager**」タブで、「**Authentication**」サブタブをクリックします。

2 「**Module Instances**」セクションで、設定する **WSSAuth** 認証モジュールのインスタンス名をクリックします。

3 **WSSAuth** 認証モジュールのインスタンスレルム属性の値を入力します。
設定できる属性のリストとその説明を次の表に示します。

ユーザー検索属性	作成予定
ユーザーレルム	作成予定
ユーザーパスワード属性	作成予定
Authentication Level	作成予定

OAMAuth 認証モジュールの追加

Oracle の認証モジュールを利用することで、OpenSSO は以前に Oracle Access Manager に対して認証されていた管理者を OpenSSO に対しても認証し、シングルサインオンを実現します。管理者は OpenSSO に資格情報を提供する必要はありません。

▼ 新しい Oracle 認証モジュールのインスタンスを追加する

1 「**Access Manager**」タブで、「**Authentication**」サブタブをクリックします。

2 「**Module Instances**」セクションで、「**New**」をクリックします。

3 「**Name**」フィールドで、この **Oracle** 認証モジュールのインスタンス名を入力します。

4 「**Type**」には「**OAMAuth**」を選択します。

5 「了解」をクリックします。

6 **OAMAuth** 認証モジュールのインスタンスを設定します。

▼ Oracle 認証モジュールのインスタンスを設定する

1 「**Access Manager**」タブで、「**Authentication**」サブタブをクリックします。

2 「**Module Instances**」セクションで、設定する **OAMAuth** 認証モジュールのインスタンス名をクリックします。

3 **Oracle** 認証モジュールのインスタンスレルム属性の値を入力します。

設定できる属性のリストとその説明を次の表に示します。

リモートユーザーの HeaderName	作成予定
許可されるヘッダーの値	「Current Values」リストを表示。作成予定 ■ ヘッダーの値をリストに追加するには、「New Value」フィールドで「To Be Developed」と入力してから「Add」をクリックします。 ■ 「Current Values」リストからエントリを削除するには、削除するエントリを選択してから「Remove」をクリックします。
認証レベル	作成予定

セキュリティートークンの作成

Oracle OpenSSO セキュリティートークンサービス (OpenSSO STS) は、Web サービスクライアントと Web サービスプロバイダの間で信頼関係を確立したり、その信頼関係を破棄したりします。Web サービスは、複数のクライアントと通信をする必要なく、1 つのエンティティー「OpenSSO STS」のみが発行したトークンを信頼できます。このようにして、OpenSSO STS はトラストポイントを管理するためのオーバーヘッドを大幅に削減します。

次にセクションでは、セキュリティートークンのニーズを判定する手順、およびそのニーズを満たすようセキュリティートークンサービスを設定してセキュリティートークンの生成と検証を行う手順を解説します。

Web サービスプロバイダの OpenSSO STS への登録

新しい Web サービスプロバイダのセキュリティーエージェントプロファイルを追加する場合、Web サービスプロバイダは OpenSSO STS に自動で登録されます。詳細については以降の節を参照してください。

Web サービスプロバイダを OpenSSO STS に登録したら、OpenSSO STS を設定して Web サービスプロバイダが受け入れられる Web クライアントのセキュリティートークンを生成できます。

OpenSSO STS からの Web サービスクライアントのセキュリティートークン要求

Web クライアントセキュリティートークンを生成するには、セキュリティートークンサービスを設定する前に、Web サービスプロバイダが必要としているセキュリティートークンの種類を調べる必要があります。OpenSSO STS は、Liberty Alliance Project のセキュリティートークンと Web Services Interoperability の Basic Security Profile セキュリティートークンをサポートしています。

セキュリティートークンの生成プロセスの流れ

Liberty Alliance Project のトークンを使用してセキュリティーを有効にしている場合、HTTP クライアント (ブラウザ) は Web サービスクライアントを通じて Web サービスプロバイダにアクセス要求を送信します。Web サービスのセキュリティーエージェントは、OpenSSO STS の認証サービスにその要求をリダイレクトします。Liberty Alliance Project のセキュリティー機構が機能している場合は、HTTP セキュリティーエージェントがリダイレクトを発行します。WS-IBS のセキュリティーを使用している場合は、SOAP セキュリティーエージェントがリダイレクトを発行します。

OpenSSO STS の認証サービスは、Web サービスプロバイダによって登録されたセキュリティー機構を判定し、適切なセキュリティートークンを取得します。認証成功後、Web サービスクライアント側の SOAP セキュリティーエージェントがセキュリティーヘッダーとトークンを挿入している間、Web サービスクライアントは SOAP メッセージ本文を提供します。その後、要求が WSP に送信される前にメッセージは署名されます。

Web サービスプロバイダ側の SOAP セキュリティーエージェントは、要求を Web サービスプロバイダ自身に転送する前に、SOAP 要求の中の署名とセキュリティートークンを検証します。Web サービスプロバイダはこの要求を処理して、SOAP セキュリティーエージェントによって署名された応答を Web サービスクライアントに返します。Web サービスクライアント側の SOAP セキュリティーエージェントは、応答を Web サービスクライアントに転送する前に、その署名を検証します。

Liberty Alliance Project のトランザクションでサポートされているトークンのリストと簡単な説明を次の表に示します。

表 3–1　要求者のトークン - Liberty Alliance Project

トークン	これらの要件を満たします
X.509	■ 安全な Web サービスは公開鍵基盤 (PKI) を使用します。PKI では、要求元の特定手段として、および Web サービスプロバイダに対する認証手段として、Web サービスクライアントが公開鍵を提供します。 ■ 安全な Web サービスは公開鍵基盤 (PKI) を使用します。PKI では、要求元の特定手段として、および Web サービスプロバイダに対する認証手段として、Web サービスクライアントが公開鍵を提供します。
BearerToken	■ 安全な Web サービスは Security Assertion Markup Language (SAML) SAML Bearer トークンによる確認手法を利用します。 ■ WSC は、要求元を Web サービスプロバイダに対して認証する手段として SAML 表明を公開鍵情報とともに提供します。 ■ 2 つ目の署名は表明を SOAP メッセージにバインドします。 ■ 2 つ目の署名バインディングは、Liberty Alliance Project で定義されたルールを使用します。
SAML トークン	■ 安全な Web サービスは、SAML Holder-of-Key の確認手法を利用します。 ■ WSC は、SAML 表明およびデジタル署名を SOAP ヘッダーに追加します。 ■ 送信側の証明書または公開鍵も署名とともに提供されます。 ■ この送信は、Liberty Alliance Project で定義されたルールを使用して処理されます。

WS-IBS のトランザクションでサポートされているトークンのリストと簡単な説明を次の表に示します。

表 3–2 要求者のトークン - WS-IBS

トークン	これらの要件を満たします
ユーザー名	■ 安全な Web サービスはユーザー名、パスワード、およびオプションで署名された要求を必要とします。 ■ Web サービスコンシューマは、要求元を特定する手段としてユーザー名トークンを提供します ■ Web サービスコンシューマは、パスワード、共有シークレット、またはパスワードに相当するものを提供して、Web サービスプロバイダに対して身元を認証します。
X.509	安全な Web サービスでは PKI (公開鍵基盤) を使用しています。PKI では、要求元を特定する手段、および Web サービスプロバイダに対する認証を実現する手段として、Web サービスコンシューマが公開鍵を提供します。
SAML-Holder-Of-Key	■ 安全な Web サービスは、SAML Holder-of-Key の確認手法を利用します。 ■ Web サービスコンシューマは、要求元を Web サービスプロバイダに対して認証する手段として SAML 表明を公開鍵情報とともに提供します。 ■ 2 つ目の署名は表明を SOAP ペイロードにバインドします。
SAML-SenderVouches	■ 安全な Web サービスは、SAML Sender-Vouches の確認手法を利用します。 ■ Web サービスコンシューマは、SAML 表明およびデジタル署名を SOAP ヘッダーに追加します。送信側の証明書または公開鍵も署名とともに提供されます。

セキュリティートークン生成表の使用

セキュリティートークン生成表を使用すれば、OpenSSO STS を設定して、Web サービスプロバイダが必要とする Web サービスクライアントのセキュリティートークンを生成できるようになります。まず、「OpenSSO STS 出力トークン」という名前の最後の列で、Web サービスプロバイダトークンの要件を満たす説明文を探します。そして、セキュリティートークンサービスを設定する場合は同じ行にあるパラメータの値を使用します。「トークン生成表の凡例」では、表の見出しと利用可能なオプションについての情報を記載しています。設定手順についての詳細は、5.2.3 節「To Configure the Security Token Service」を参照してください。Web サービスのセキュリティーと関連技術についての一般的な情報については、次を参照してください。

- http://www.oracle.com/technology/tech/standards/pdf/security.pdf
- http://download.oracle.com/docs/cd/E15523_01/web.1111/b32511/intro_security.htm#CDDHHGEE

セキュリティートークン生成表では、頻繁に使用するセキュリティートークンサービスのパラメータ設定と、それらの設定に基づいて OpenSSO STS が生成するセキュリティートークンの種類についてまとめています。

表 3–3 セキュリティートークン生成表

行	メッセージレベルのセキュリティーバインディング	**Web** サービスクライアントのトークン	**KeyType**	**OnBehalfOf** トークン	**Use Key**	**OpenSSO STS** 出力トークン
1	非対称	X509	ベアラー	はい	いいえ	SAML ベアラー (証明鍵なし)
2	非対称	ユーザー名	ベアラー	はい	いいえ	SAML ベアラー (証明鍵なし)
3	非対称	X509	ベアラー	いいえ	いいえ	SAML ベアラー (証明鍵なし)
4	非対称	ユーザー名	ベアラー	いいえ	いいえ	SAML ベアラー (証明鍵なし)
5	非対称	X509	対称	はい	いいえ	SAML Holder-of-Key (対称証明鍵)
6	非対称	ユーザー名	対称	はい	いいえ	SAML Holder-of-Key (対称証明鍵)
7	非対称	X509	対称	いいえ	いいえ	SAML Holder-of-Key (対称)
8	非対称	ユーザー名	対称	いいえ	いいえ	SAML Holder-of-Key (対称証明鍵)
9	非対称	X509	非対称	いいえ	Web サービスクライアントの公開鍵	SAML Holder-of-Key (非対称証明鍵)

表 3–3 セキュリティートークン生成表 (続き)

10	非対称	X509	SAML Sender-Vouches	はい	いいえ	SAML Sender-Vouches (証明鍵なし)
11	非対称	ユーザー名	SAML Sender-Vouches	はい	いいえ	SAML Sender-Vouches (証明鍵なし)
12	非対称	X509	SAML Sender-Vouches	いいえ	いいえ	エラー
13	非対称	ユーザー名	SAML Sender-Vouches	いいえ	いいえ	エラー
14	トランスポート	ユーザー名	ベアラー	はい	いいえ	SAML ベアラー (証明鍵なし)
15	トランスポート	ユーザー名	ベアラー	いいえ	いいえ	SAML ベアラー (証明鍵なし)
16	トランスポート	ユーザー名	対称	はい	いいえ	SAML Holder-of-Key (対称)
17	トランスポート	ユーザー名	対称	いいえ	いいえ	SAML Holder-of-Key (対称証明鍵)
18	トランスポート	ユーザー名	SAML Sender-Vouches	はい	いいえ	SAML Sender-Vouches (証明鍵なし)
19	トランスポート	ユーザー名	SAML Sender-Vouches	いいえ	いいえ	エラー
20	非対称	ユーザー名	非対称	いいえ	Web サービスクライアントの公開鍵	エラー
21	トランスポート	ユーザー名	非対称	いいえ	Web サービスクライアントの公開鍵	エラー
22	非対称	X509	非対称	はい	いいえ	エラー

表 3–3 セキュリティートークン生成表 (続き)

23	非対称	ユーザー名	非対称	はい	いいえ	エラー
24	トランスポート	ユーザー名	非対称	はい	いいえ	エラー
25	非対称	X509	非対称	いいえ	いいえ	SAML Holder-of-Key (非対称証明鍵)
26	非対称	X509	いいえ	いいえ	いいえ	SAML Holder-of-Key (非対称証明鍵)
27	非対称	ユーザー名	いいえ	いいえ	いいえ	SAML Holder-of-Key (対称証明鍵)
28	トランスポート	ユーザー名	いいえ	いいえ	いいえ	SAML Holder-of-Key (対称証明鍵)

セキュリティートークンサービスの問題点と回避方法

作成予定

設定に関する問題点と回避方法

作成予定

マニュアルの正誤表

作成予定

◆ ◆ ◆ 第 4 章

Oracle OpenSSO Fedlet の使用

この節では、Oracle OpenSSO Fedlet についての次の情報を記載しています。

Oracle OpenSSO Fedlet について

Oracle OpenSSO Fedlet は、Java または .NET のサービスプロバイダアプリケーションに配備できる計量なサービスプロバイダ (SP) 実装で、アプリケーションは SAMLv2 プロトコルを使用して Oracle OpenSSO 8.0 Update 2 などのアイデンティティープロバイダ (IDP) とやりとりできるようになります。Fedlet には、お使いのプラットフォームに合わせて 2 つのバージョンが用意されています。

- Java Fedlet は OpenSSO 8.0 で最初にリリースされました。詳しくは、『Sun OpenSSO Enterprise 8.0 Deployment Planning Guide』の第 5 章「Using the OpenSSO Enterprise Fedlet to Enable Identity Federation」を参照してください。
- .NET Fedlet は OpenSSO 8.0 Update 1 でリリースされました。詳しくは、『Sun OpenSSO Enterprise 8.0 Update 1 Release Notes』の第 10 章「Using the ASP.NET Fedlet with OpenSSO Enterprise 8.0 Update 1」を参照してください。

Oracle OpenSSO 8.0 Update 2 では、次の Fedlet が利用できます。

- Oracle OpenSSO 8.0 Update 2 の ZIP ファイルを解凍したあと、Java Fedlet および .NET Fedlet の両方が次のファイルから取得できます。

  zip-root`/opensso/fedlet/fedlet-unconfigured.zip`。*zip-root* は、Oracle OpenSSO 8.0 Update 2 の ZIP ファイルを解凍した場所です。
- Oracle OpenSSO 8.0 Update 2 をインストールしたあと、OpenSSO 8.0 管理コンソールで「Common Tasks」の下にある「Create Fedlet」ワークフローを使用して Java Fedlet を作成できます。

Oracle OpenSSO Fedlet の要件

Fedlet には次の要件があります。

- `fedlet.war` または、Fedlet と統合された Java サービスプロバイダアプリケーションを配備する場合は、Oracle OpenSSO 8.0 Update 2 がサポートする Web コンテナ。12 ページの「OpenSSO 8.0 Update 2 のハードウェアおよびソフトウェア要件」を参照してください。
- .NET Fedlet を配備する場合は、Microsoft Internet Information Server (IIS) 7.0 以降
- JDK 1.6.x 以降

Oracle OpenSSO Fedlet の設定

この節では、サービスプロバイダアプリケーションで Fedlet を最初に設定する方法を説明します。

- 44 ページの「Java Fedlet を設定する」
- 46 ページの「.NET Fedlet を設定する」

Fedlet の初回設定が終わったあと、ほかに実施する設定があれば続けます。いくつかの留意点を次に示します。

- Fedlet の `sp.xml` ファイルを編集する場合は、このファイルをアイデンティティープロバイダに再インポートする必要があります。
- サービスプロバイダ側でほかの Fedlet 設定も変更する場合は、その情報をアイデンティティープロバイダの管理者に伝えて、アイデンティティープロバイダ側で必要な設定を変更できるようにします。

▼ Java Fedlet を設定する

1. アイデンティティープロバイダ側で、アイデンティティープロバイダの **XML** メタデータを生成し、**`idp.xml`** という名前のファイルにそのメタデータを保存します。

   Oracle OpenSSO 8.0 Update 2 の場合、`exportmetadata.jsp` を使用します。例:

   ```
   http://opensso-idp.example.com:8080/opensso/saml2/jsp/exportmetadata.jsp
   ```

2. 必要な場合は、サービスプロバイダ側で **Fedlet** の **ZIP** ファイルを解凍します。

3. **Fedlet** のホームディレクトリを作成します。**Fedlet** は、メタデータ、トラストサークル、設定プロパティーファイルをこのディレクトリから読み取ります。

   デフォルトの場所は、Feldet の Web コンテナを実行しているユーザーのホームディレクトリ下にある Fedlet サブディレクトリです。この場所は `user.home` JVM プロ

パティーで示されています。たとえば、このホームディレクトリが /home/webservd の場合、Fedlet のホームディレクトリは次のようになります。

```
/home/webservd/fedlet
```

Fedlet のデフォルトホームディレクトリを変更するには、JVM ランタイム com.sun.identity.fedlet.home プロパティーの値を希望する場所に設定します。例:

```
-Dcom.sun.identity.fedlet.home=/export/fedlet/conf
```

このように変更すると、Fedlet はそのメタデータ、トラストサークル、設定ファイルを /export/fedlet/conf ディレクトリから読み取るようになります。

4 **Java Fedlet の java/conf ディレクトリから次のファイルを Fedlet ホームディレクトリにコピーします。**

- sp.xml-template
- sp-extended.xml-template
- idp-extended.xml-template
- fedlet.cot-template

5 **Fedlet ホームディレクトリで、コピーしたファイルの名前を変更して、各ファイルの名前から -template を取り除きます。**

6 **Fedlet ホームディレクトリにコピーして名前を変更した各ファイルで、次の表に示すとおりにタグを置き換えます。**

タグ	置き換え後のタグ
FEDLET_COT	リモートアイデンティティープロバイダおよび Java Fedlet サービスプロバイダアプリケーションがメンバーとなっているトラストサークル (COT) の名前。
FEDLET_ENTITY_ID	Java Fedlet サービスプロバイダアプリケーションの ID (名前)。例: fedletsp
FEDLET_PROTOCOL	Java Fedlet サービスプロバイダアプリケーションの Web コンテナのプロトコル (fedlet.war など)。例: https
FEDLET_HOST	Java Fedlet サービスプロバイダアプリケーションの Web コンテナのホスト名 (fedlet.war など)。例: fedlet-host.example.com
FEDLET_PORT	Java Fedlet サービスプロバイダアプリケーションの Web コンテナのポート番号 (fedlet.war など)。例: 80
FEDLET_DEPLOY_URI	Java Fedlet サービスプロバイダアプリケーションの URL。例: http://fedletsp.example.com/myFedletApp
IDP_ENTITY_ID	リモートアイデンティティープロバイダの ID (名前)。例: openssoidp

タグ	置き換え後のタグ
注: Fedlet サービスプロバイダまたはアイデンティティープロバイダのエンティティ ID にパーセント記号 (%) またはコンマ (,) が含まれている場合は、`fedlet.cot` ファイル内で置き換える前にその文字をエスケープする必要があります。 たとえば、「%」を「%25」に、「,」を「%2C」に変更します。	

7 **Java Fedlet** の **`java/conf`** ディレクトリから、**`FedletConfiguration.properties`** ファイルを **Fedlet** のホームディレクトリにコピーします。

8 アイデンティティープロバイダの標準メタデータ **XML** ファイル (手順 **1** のもの) を **Fedlet** ホームディレクトリにコピーします。このファイルは **`idp.xml`** という名前にする必要があります。

9 **Java Fedlet XML** メタデータファイル (**`sp.xml`**) をアイデンティティープロバイダにインポートします。

Oracle OpenSSO 8.0 Update 2 の場合、OpenSSO 8.0 管理コンソールで「Common Tasks」の下にある「Register Remote Service Provider」ワークフローを使用して、Java Fedlet サービスプロバイダのメタデータをインポートしたり、Java Fedlet サービスプロバイダをトラストサークルに追加したりできます。

次の手順　要件によっては、Java Fedlet のその他の設定を続けます。

▼ .NET Fedlet を設定する

1 アイデンティティープロバイダ側で、アイデンティティープロバイダの **XML** メタデータを生成し、**`idp.xml`** という名前のファイルにそのメタデータを保存します。

Oracle OpenSSO 8.0 Update 2 の場合、`exportmetadata.jsp` を使用します。次に例を示します。

```
http://opensso-idp.example.com:8080/opensso/saml2/jsp/exportmetadata.jsp
```

2 必要な場合は、サービスプロバイダ側で **Fedlet** の **ZIP** ファイルを解凍します。

3 **.NET Fedlet** の **`asp.net/conf`** フォルダから次のファイルをアプリケーションの **`App_Data`** フォルダにコピーします。

- `sp.xml-template`
- `sp-extended.xml-template`
- `idp-extended.xml-template`
- `fedlet.cot-template`

4 **`App_Data`** フォルダで、コピーしたファイルの名前を変更して、各ファイルの名前から **`-template`** を取り除きます。

5 **App_Data** フォルダにコピーして名前を変更した各ファイルで、次の表に示すとおりにタグを置き換えます。

タグ	置き換え後のタグ
FEDLET_COT	リモートアイデンティティープロバイダおよび .NET Fedlet サービスプロバイダアプリケーションがメンバーとなっているトラストサークル (COT) の名前。
FEDLET_ENTITY_ID	.NET Fedlet サービスプロバイダアプリケーションの ID (名前)。例: `fedletsp`
FEDLET_DEPLOY_URI	.NET Fedlet サービスプロバイダアプリケーションの URL。例: `http://fedletsp.example.com/myFedletApp`
IDP_ENTITY_ID	リモートアイデンティティープロバイダの ID (名前)。例: `openssoidp`

6 アイデンティティープロバイダの標準メタデータ **XML** ファイル (手順 **1** のもの) をアプリケーションの **App_Data** フォルダにコピーします。このファイルは **idp.xml** という名前にする必要があります。

7 **.NET Fedlet asp.net/bin** フォルダの **Fedlet.dll** ファイルおよび **Fedlet.dll.config** ファイルを、アプリケーションの **bin** フォルダにコピーします。

8 **.NET Fedlet XML** メタデータファイル (**sp.xml**) をアイデンティティープロバイダにインポートします。

Oracle OpenSSO 8.0 Update 2 の場合、OpenSSO 8.0 管理コンソールで「Common Tasks」の下にある「Register Remote Service Provider」ワークフローを使用して、.NET Fedlet サービスプロバイダのメタデータをインポートしたり、Java Fedlet サービスプロバイダをトラストサークルに追加したりできます。

次の手順　要件によっては、.NET Fedlet のその他の設定を続けます。

OpenSSO 8.0 Update 2 の Fedlet の新機能

Oracle OpenSSO 8.0 Update 2 の Fedlet には、次の新機能が含まれています。

- 48 ページの「Fedlet のバージョン情報 (CR 6941387)」
- 48 ページの「Java Fedlet パスワードの暗号化および復号化 (CR 6930477)」
- 48 ページの「Java Fedlet の署名および暗号化のサポート」
- 52 ページの「Java Fedlet の属性クエリーのサポート (CR 6930476)」
- 53 ページの「.NET Fedlet での要求の暗号化および応答の復号化 (CR 6939005)」
- 55 ページの「.NET Fedlet の要求および応答の署名 (CR 6928530)」
- 57 ページの「.NET Fedlet のシングルログアウト (CR 6928528 および CR 6930472)」

- 58 ページの「.NET Fedlet サービスプロバイダ開始のシングルサインオン (CR 6928525)」
- 58 ページの「.NET Fedlet による複数のアイデンティティープロバイダとディスカバリサービスのサポート (CR 6928524)」
- 60 ページの「.NET Fedlet によるアイデンティティープロバイダのディスカバリサービスのサポート (CR 6928524)」

Fedlet のバージョン情報 (CR 6941387)

Oracle OpenSSO Fedlet にはバージョン情報が含まれます。Fedlet パッケージ (ZIP ファイル) 内のファイルを抽出したあと、次のいずれかのファイルを表示して Fedlet のバージョンを確認します。

- Java Fedlet の場合: `java/conf/FederationConfig.properties`
- .NET Fedlet の場合: `asp.net/bin/Fedlet.dll.config`

Java Fedlet パスワードの暗号化および復号化 (CR 6930477)

Java Fedlet は、`storepass` および `keypass` の各パスワードファイルを暗号化するための `fedletEncode.jsp` を `fedlet.war` ファイル内に提供しています。デフォルトでは、それぞれの Fedlet に異なる暗号化鍵が生成されます。この暗号化鍵を変更するには、Fedlet の `FederationConfig.properties` ファイルで `am.encryption.pwd` プロパティーを設定します。

Java Fedlet の署名および暗号化のサポート

Java Fedlet は XML 署名の検証と、暗号化された `assertion` と `NameID` 要素、さらにそれらの属性の復号化をサポートします。

▼ Java Fedlet で署名および暗号化を設定する

1. **`keytool`** ユーティリティーを使用して、**`keystore.jks`** という名前のキーストアファイルを作成します。
2. 署名に使用する非公開鍵 (該当する場合は公開証明書も) および暗号化に使用する非公開鍵 (該当する場合は公開証明書も) を、**`keystore.jks`** ファイルに追加します。
3. **`.storepass`** ファイルを作成します。
4. **`.storepass`** ファイルにパスワードを追加します。**`fedletEncode.jsp`** を使用してパスワードを暗号化します。

5 **.keypass** ファイルを作成します。

6 **.keypass** ファイルにパスワードを追加します。**fedletEncode.jsp** を使用してパスワードを暗号化します。

7 平文のパスワードを使用している場合は、**FederationConfig.properties** ファイル内で次の行をコメントアウトします。

```
com.sun.identity.saml.xmlsig.passwordDecoder=
    com.sun.identity.fedlet.FedletEncodeDecode
```

8 **FederationConfig.properties** ファイル内で次の属性への完全パスを設定します。*path* の部分は各ファイルへの絶対パスです。

```
com.sun.identity.saml.xmlsig.keystore=path/keystore.jks
com.sun.identity.saml.xmlsig.storepass=path/.storepass
com.sun.identity.saml.xmlsig.keypass=path/.keypass
```

9 **keytool** を使用して署名証明書をエクスポートします。次に例を示します。

```
keytool -export -keystore keystore.jks -rfc -alias test
```

このツールを実行すると、keystore.jks にアクセスするために使用するパスワードを入力するよう求められ、その後証明書が生成されます。

10 暗号化証明書が必要な場合は、前の手順で示すとおり、**keytool** を使用してエクスポートします。または、署名と暗号化の両方に同一の証明書を使用します。

11 **KeyDescriptor XML** ブロックを作成して、そこに暗号化証明書を追加します。たとえば、次のように **KeyDescriptor** 要素の **use="signing"** タグを記述します。

```
<KeyDescriptor use="signing">
      <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:X509Data>
          <ds:X509Certificate>
MIICQDCCAakCBEeNB0swDQYJKoZIhvcNAQEEBQAwZzELMAkGA1UEBhMCVVMxEzARBgNVBAgTCkNh
bGlmb3JuaWExFDASBgNVBAcTC1NhbnRhIENsYXJhMQwwCgYDVQQKEwNTdW4xEDAOBgNVBAsTB09w
ZW5TU08xDTALBgNVBAMTBHRlc3QwHhcNMDgwMTE1MTkxOTM5WhcNMTgwMTEyMTkxOTM5WjBnMQsw
CQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTEUMBIGA1UEBxMLU2FudGEgQ2xhcmExDDAK
BgNVBAoTA1N1bjEQMA4GA1UECxMHT3BlblNTTzENMAsGA1UEAxMEdGVzdDCBnzANBgkqhkiG9w0B
AQEFAAOBjQAwgYkCgYEArSQc/U75GB2AtKhbGS5piiLkmJzqEsp64rDxbMJ+xDrye0EN/q1U5Of\+
RkDsaN/igkAvV1cuXEgTL6RlafFPcUX7QxDhZBhsYF9pbwtMzi4A4su9hnxIhURebGEmxKW9qJNY
Js0Vo5+IgjxuEWnjnnVgHTs1+mq5QYTA7E6ZyL8CAwEAATANBgkqhkiG9w0BAQQFAAOBgQB3Pw/U
QzPKTPTYi9upbFXlrAKMwtFf2OW4yvGWWvlcwcNSZJmTJ8ARvVYOMEVNbsT4OFcfu2/PeYoAdiDA
cGy/F2Zuj8XJJpuQRSE6PtQqBuDEHjjmOQJ0rV/r8mO1ZCtHRhpZ5zYRjhRC9eCbjx9VrFax0JDC
/FfwWigmrW0Y0Q==
          </ds:X509Certificate>
        </ds:X509Data>
      </ds:KeyInfo>
</KeyDescriptor>
```

12 別の **KeyDescriptor XML** ブロックを作成して、そこに暗号化証明書を追加します。たとえば、次のように **KeyDescriptor** 要素の **use="encryption"** タグを記述します。

```
<KeyDescriptor use="encryption">
      <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
```

```
        <X509Data>
          <X509Certificate>
MIICQDCCAakCBEeNB0swDQYJKoZIhvcNAQEEBQAwZzELMAkGA1UEBhMCVVMxEzARBgNVBAgTCkNh
bGlmb3JuaWExFDASBgNVBAcTC1NhbnRhIENsYXJhMQwwCgYDVQQKEwNTdW4xEDAOBgNVBAsTB09w
ZW5TU08xDTALBgNVBAMTBHRlc3QwHhcNMDgwMTE1MTkxOTM5WhcNMTgwMTEyMTkxOTM5WjBnMQsw
CQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTEUMBIGA1UEBxMLU2FudGEgQ2xhcmExDDAK
BgNVBAoTA1N1bjEQMA4GA1UECxMHT3BlblNTTzENMAsGA1UEAxMEdGVzdDCBnzANBgkqhkiG9w0B
AQEFAAOBjQAwgYkCgYEArSQc/U75GB2AtKhbGS5piiLkmJzqEsp64rDxbMJ+xDrye0EN/q1U5Of\+
RkDsaN/igkAvV1cuXEgTL6RlafFPcUX7QxDhZBhsYF9pbwtMzi4A4su9hnxIhURebGEmxKW9qJNY
Js0Vo5+IgjxuEWnjnnVgHTs1+mq5QYTA7E6ZyL8CAwEAATANBgkqhkiG9w0BAQQFAAOBgQB3Pw/U
QzPKTPTYi9upbFXlrAKMwtFf2OW4yvGWWvlcwcNSZJmTJ8ARvVYOMEVNbsT4OFcfu2/PeYoAdiDA
cGy/F2Zuj8XJJpuQRSE6PtQqBuDEHjjmOQJ0rV/r8mO1ZCtHRhpZ5zYRjhRC9eCbjx9VrFax0JDC
/FfwWigmrW0Y0Q==
          </X509Certificate>
        </X509Data>
      </KeyInfo>
<EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc">
  <KeySize xmlns="http://www.w3.org/2001/04/xmlenc#">128</KeySize>
</EncryptionMethod>
</KeyDescriptor>
```

13 **Java Fedlet** の **`sp.xml`** ファイルで、これらの署名証明書および暗号化証明書の各 **XML** ブロックを **`SPSSODescriptor`** 要素の下に追加します。サンプルの **`SPSSODescriptor`** 要素については、例 **4–1**を参照してください。

`AuthnRequestsSigned` 属性が `true` に設定され、Java Fedlet はすべての認証要求に署名するようになります。

14 **Java Fedlet** の **`sp-extended.xml`** ファイルで、次の要素の値を設定します。

- `signingCertAlias` には、キーストアにある XML 署名証明書のエイリアスが含まれています。
- `encryptionCertAlias` には、キーストアにある XML 暗号化証明書のエイリアスが含まれています。

15 **Java Fedlet** サービスプロバイダの暗号化内容を強化するには、**`sp-extended.xml`** ファイルで次の属性を **`true`** に設定します。

- `wantAssertionEncrypted`
- `wantNameIDEncrypted`
- `wantAttributeEncrypted`

16 **Java Fedlet** サービスプロバイダの署名内容と署名対象を強化するには、次の属性を **`true`** に設定します。

- `idp.xml` ファイルの `wantAuthnRequestsSigned` は、何に署名すべきかを Fedlet に伝えます。
- `sp.xml` ファイルの `AuthnRequestsSigned` および `WantAssertionsSigned` は、Fedlet の署名対象をアイデンティティープロバイダに伝えます。
- `sp-extended.xml` ファイルの `wantArtifactResponseSigned` は、何に署名すべきかを Fedlet に伝えます。

- sp-extended.xml ファイルの wantPOSTResponseSigned
- sp-extended.xml ファイルの wantLogoutRequestSigned
- sp-extended.xml ファイルの wantLogoutResponseSigned

アイデンティティープロバイダが特定のメッセージに署名を必要とする場合、idp-extended.xml ファイルでそれぞれの属性を true に設定します。たとえば、wantLogoutRequestSigned や wantLogoutResponseSigned などです。

注 – sp-extended.xml ファイルで属性を設定する場合は、その情報をアイデンティティープロバイダの管理者に伝えて、アイデンティティープロバイダ側で必要な設定を変更できるようにします。

---

17 **Java Fedlet の Web コンテナを再起動します。**

18 **Java Fedlet の sp.xml ファイルをアイデンティティープロバイダにインポートします。**

例 4–1 Java Fedlet のサンプル SPSSODescriptor 要素

```
<EntityDescriptor entityID="fedlet"
xmlns="urn:oasis:names:tc:SAML:2.0:metadata">

<SPSSODescriptor AuthnRequestsSigned="true" WantAssertionsSigned="false"
protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
<b><KeyDescriptor use="signing">
      <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:X509Data>
          <ds:X509Certificate>
MIICQDCCAakCBEeNB0swDQYJKoZIhvcNAQEEBQAwZzELMAkGA1UEBhMCVVMxEzARBgNVBAgTCkNh
bGlmb3JuaWExFDASBgNVBAcTC1NhbnRhIENsYXJhMQwwCgYDVQQKEwNTdW4xEDAOBgNVBAsTB09w
ZW5TU08xDTALBgNVBAMTBHRlc3QwHhcNMDgwMTE1MTkxOTM5WhcNMTgwMTEyMTkxOTM5WjBnMQsw
CQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTEUMBIGA1UEBxMLU2FudGEgQ2xhcmExDDAK
BgNVBAoTA1N1bjEQMA4GA1UECxMHT3BlblNTTzENMAsGA1UEAxMEdGVzdDCBnzANBgkqhkiG9w0B
AQEFAAOBjQAwgYkCgYEArSQc/U75GB2AtKhbGS5piiLkmJzqEsp64rDxbMJ+xDrye0EN/q1U5Of\+
RkDsaN/igkAvV1cuXEgTL6RlafFPcUX7QxDhZBhsYF9pbwtMzi4A4su9hnxIhURebGEmxKW9qJNY
Js0Vo5+IgjxuEWnjnnVgHTs1+mq5QYTA7E6ZyL8CAwEAATANBgkqhkiG9w0BAQQFAAOBgQB3Pw/U
QzPKTPTYi9upbFXlrAKMwtFf2OW4yvGWWvlcwcNSZJmTJ8ARvVYOMEVNbsT4OFcfu2/PeYoAdiDA
cGy/F2Zuj8XJJpuQRSE6PtQqBuDEHjjmOQJ0rV/r8mO1ZCtHRhpZ5zYRjhRC9eCbjx9VrFax0JDC
/FfwWigmrW0Y0Q==
          </ds:X509Certificate>
        </ds:X509Data>
      </ds:KeyInfo>

</KeyDescriptor></b>
<b><KeyDescriptor use="encryption">
      <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
        <X509Data>
          <X509Certificate>
MIICQDCCAakCBEeNB0swDQYJKoZIhvcNAQEEBQAwZzELMAkGA1UEBhMCVVMxEzARBgNVBAgTCkNh
bGlmb3JuaWExFDASBgNVBAcTC1NhbnRhIENsYXJhMQwwCgYDVQQKEwNTdW4xEDAOBgNVBAsTB09w
ZW5TU08xDTALBgNVBAMTBHRlc3QwHhcNMDgwMTE1MTkxOTM5WhcNMTgwMTEyMTkxOTM5WjBnMQsw
CQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTEUMBIGA1UEBxMLU2FudGEgQ2xhcmExDDAK
```

```
BgNVBAoTA1N1bjEQMA4GA1UECxMHT3BlblNTTzENMAsGA1UEAxMEdGVzdDCBnzANBgkqhkiG9w0B
AQEFAAOBjQAwgYkCgYEArSQc/U75GB2AtKhbGS5piiLkmJzqEsp64rDxbMJ+xDrye0EN/q1U5Of\+
RkDsaN/igkAvV1cuXEgTL6RlafFPcUX7QxDhZBhsYF9pbwtMzi4A4su9hnxIhURebGEmxKW9qJNY
Js0Vo5+IgjxuEWnjnnVgHTs1+mq5QYTA7E6ZyL8CAwEAATANBgkqhkiG9w0BAQQFAAOBgQB3Pw/U
QzPKTPTYi9upbFXlrAKMwtFf2OW4yvGWWvlcwcNSZJmTJ8ARvVYOMEVNbsT4OFcfu2/PeYoAdiDA
cGy/F2Zuj8XJJpuQRSE6PtQqBuDEHjjmOQJ0rV/r8mO1ZCtHRhpZ5zYRjhRC9eCbjx9VrFax0JDC
/FfwWigmrW0Y0Q==
          </X509Certificate>
        </X509Data>
      </KeyInfo>

<EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc">
<KeySize xmlns="http://www.w3.org/2001/04/xmlenc#">128</KeySize>
</EncryptionMethod>
</KeyDescriptor></b>
<NameIDFormat>urn:oasis:names:tc:SAML:2.0:nameid-format:transient</NameIDFormat
><AssertionConsumerService index="1"
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
Location="http://server.sun.com:7070/fedlet/fedletapplication"/>
</SPSSODescriptor>
</EntityDescriptor>
```

Java Fedlet の属性クエリーのサポート (CR 6930476)

Java Fedlet は SAMLv2 の属性クエリーをサポートし、Oracle OpenSSO 8.0 Update 2 などのアイデンティティープロバイダに特定のアイデンティティー属性の値を問い合わせます。クエリーに署名をしたり、クエリーを暗号化したりするよう Fedlet を設定できます。Fedlet クエリーを発行するには署名が必要ですが、暗号化は任意です。

▼ Java Fedlet で属性クエリーを設定する

1 48 ページの「Java Fedlet の署名および暗号化のサポート」の手順に従って、**XML** 署名を有効にして属性クエリーに署名します。

2 前の手順で生成した証明書を **Fedlet** の **sp.xml** ファイル内にある **RoleDescriptor** 要素に追加します。次の例では、ペーストした証明書の中に **2** つの **KeyDescriptor** タグがあります。**1** つは署名用で、もう **1** つは暗号化用です。暗号化を有効にしていない場合、**KeyDescriptor use="encryption"** タグは不要です。

```
<RoleDescriptor xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:query="urn:oasis:names:tc:SAML:metadata:ext:query"
      xsi:type="query:AttributeQueryDescriptorType"
      protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
  <KeyDescriptor use="signing">
      <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:X509Data>
          <ds:X509Certificate>
            --certificate--
          </ds:X509Certificate>
```

```
        </ds:X509Data>
      </ds:KeyInfo>
  </KeyDescriptor>
  <KeyDescriptor use="encryption">
      <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:X509Data>
          <ds:X509Certificate>
            --certificate--
          </ds:X509Certificate>
        </ds:X509Data>
      </ds:KeyInfo>
    <EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc">
<xenc:KeySize
xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">128</xenc:KeySize>
    </EncryptionMethod>
  </KeyDescriptor>
</RoleDescriptor>
```

3 **Java Fedlet の `sp-extended.xml` ファイルで、`signingCertAlias` 属性の値を指定して、設定している場合は `encryptionCertAlias` 属性の値も指定します。**

アイデンティティープロバイダを設定して表明を暗号化する場合は、`NameID` 要素も暗号化します。したがって、`wantNameIDEncrypted` 属性の値を `true` に設定する必要があります。`AttributeQueryConfig` 要素に XML コードを追加します。次に例を示します。

```
<Attribute name="signingCertAlias">
      <Value>test</Value>
</Attribute>
<Attribute name="encryptionCertAlias">
      <Value>test</Value>
</Attribute>
<Attribute name="wantNameIDEncrypted">
      <Value>true</Value>
</Attribute>
```

この例では、`test` はサンプル鍵のエイリアスです。

4 **Java Fedlet のメタデータファイル (`sp.xml`) をアイデンティティープロバイダにインポートします。**

また、アイデンティティープロバイダで追加の設定手順を実行して Fedlet の属性クエリーをサポートします。

.NET Fedlet での要求の暗号化および応答の復号化 (CR 6939005)

.NET Fedlet は、NameID、属性、および表明の要素に対して、送信する XML 要求を暗号化したり、受信する応答を復号化したりできます。

▼ .NET Fedlet を設定して要求を暗号化および応答を復号化する

1 Micrsoft 管理コンソールの証明書スナップインを使用して、X.509 証明書をローカルコンピュータのアカウント内にある個人フォルダにインポートします。このスナップインを使用するには、次の Microsoft の記事を参照してください。

http://msdn.microsoft.com/en-us/library/ms788967.aspx

2 「プロパティー」ダイアログを表示して値を入力することで、この証明書にわかりやすい名前を指定します (この値は手順 4 で保存)。

3 前述の Microsoft の記事に従って、Internet Information Server (IIS) によって使用されるユーザーアカウントの証明書に対して読み取りアクセスができるよう適切な権限を設定します。次に例を示します。

a. 証明書スナップインで、「Action」、「All Tasks」、「Manage Private Keys」の順に選択します。

b. IIS を実行しているユーザーアカウント (通常は NETWORK SERVICE) に「Allow Read」のアクセス権を指定します。

4 .NET Fedlet の拡張メタデータファイル (`sp-extended.xml`) で、手順 2 で `encryptionCertAlias` 属性の値に指定したわかりやすい名前を指定します。次に例を示します。

```
<Attribute name="encryptionCertAlias">
<Value>MyFedlet</Value>
```

5 .NET Fedlet のサービスプロバイダのメタデータファイル (`sp.xml`) で、暗号化鍵用の `KeyDescriptor` を追加します。

すでに使用した Microsoft 管理コンソールの証明書スナップインを使用して、`KeyDescriptor` XML ブロックに挿入する証明書の公開鍵を、Base64 エンコーディングでエクスポートします。この `KeyDescriptor` は、`SPSSODescriptor` 内の最初の子要素である必要があります。次に例を示します。

```
<KeyDescriptor use="encryption">
          <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
            <ds:X509Data>
              <ds:X509Certificate>
MIICQDCCAakCBEeNB0swDQYJKoZIhvcNAQEEBQAwZzELMAkGA1UEBhMCVVMxEzARBgNVBAgTCkNh
bGlmb3JuaWExFDASBgNVBAcTC1NhbnRhIENsYXJhMQwwCgYDVQQKEwNTdW4xEDAOBgNVBAsTB09w
ZW5TU08xDTALBgNVBAMTBHRlc3QwHhcNMDgwMTE1MTkxOTM5WhcNMTgwMTEyMTkxOTM5WjBnMQsw
CQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTEUMBIGA1UEBxMLU2FudGEgQ2xhcmExDDAK
BgNVBAoTA1N1bjEQMA4GA1UECxMHT3BlblNTTzENMAsGA1UEAxMEdGVzdDCBnzANBgkqhkiG9w0B
AQEFAAOBjQAwgYkCgYEArSQc/U75GB2AtKhbGS5piiLkmJzqEsp64rDxbMJ+xDrye0EN/q1U5Of\+
RkDsaN/igkAvV1cuXEgTL6RlafFPcUX7QxDhZBhsYF9pbwtMzi4A4su9hnxIhURebGEmxKW9qJNY
Js0Vo5+IgjxuEWnjnnVgHTs1+mq5QYTA7E6ZyL8CAwEAATANBgkqhkiG9w0BAQQFAAOBgQB3Pw/U
QzPKTPTYi9upbFXlrAKMwtFf2OW4yvGWWvlcwcNSZJmTJ8ARvVYOMEVNbsT4OFcfu2/PeYoAdiDA
cGy/F2Zuj8XJJpuQRSE6PtQqBuDEHjjmOQJ0rV/r8mO1ZCtHRhpZ5zYRjhRC9eCbjx9VrFax0JDC
/FfwWigmrW0Y0Q==
```

```
                </ds:X509Certificate>
              </ds:X509Data>
            </ds:KeyInfo>
            <EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc">
            <KeySize
xmlns="http://www.w3.org/2001/04/xmlenc#">128</KeySize>
            </EncryptionMethod>
</KeyDescriptor>
```

6 **.NET** アプリケーションに関連づけられたアプリケーションプールを再起動します。

次の手順 この設定をテストするには、サンプルアプリケーションを使用します。また、設定したメタデータに適切な変更を加えて次の属性を設定することで、アイデンティティープロバイダとやりとりする要求を暗号化したり応答を復号化したりします。

- 表明: `sp-extended.xml` メタデータファイルの `wantAssertionEncrypted` 属性を `true` に設定して、アイデンティティープロバイダから送られてくる応答に含まれている `EncryptedAssertion` 要素を .NET Fedlet が復号化できるようにします。
- 属性: `sp-extended.xml` メタデータファイルの `wantAssertionEncrypted` 属性を `true` に設定して、アイデンティティープロバイダから送られてくる応答に含まれている `EncryptedAttribute` 要素を .NET Fedlet が復号化できるようにします。
- NameID: `idp-extended.xml` メタデータファイルの `wantNameIDEncrypted` 属性を `true` に設定して、送信する要求の中にある `NameID` 要素を .NET Fedlet が暗号化できるようにします。`sp-extended.xml` で同じ属性を設定して、アイデンティティープロバイダから送られてくる応答の中の `EncryptedID` 要素を .NET Fedlet が復号化できるようにします。

.NET Fedlet の要求および応答の署名 (CR 6928530)

.NET Fedlet は `Authn` 要求やログアウト要求など、送信する XML 要求の署名をサポートします。

▼ .NET Fedlet が要求および応答に署名するよう設定するには、次の手順に従います。

1 **Micrsoft** 管理コンソールの証明書スナップインを使用して、**X.509** 証明書をローカルコンピュータのアカウント内にある個人フォルダにインポートします。このスナップインを使用するには、次の **Microsoft** の記事を参照してください。

`http://msdn.microsoft.com/en-us/library/ms788967.aspx`

2 「プロパティー」ダイアログを表示して値を入力することで、この証明書にわかりやすい名前を指定します **(**手順 **4** でこの値を保存**)**。

3 前述の **Microsoft** の記事に従って、**Internet Information Server (IIS)** によって使用されるユーザーアカウントの証明書に対して読み取りアクセスができるよう適切な権限を設定します。例:

   a. 証明書スナップインで、「**Action**」、「**All Tasks**」、「**Manage Private Keys**」の順に選択します。

   b. **IIS** を実行しているユーザーアカウント (通常は **NETWORK SERVICE**) に「**Allow Read**」のアクセス権を指定します。

4 **.NET Fedlet** の拡張メタデータファイル (**sp-extended.xml**) で、手順 **2** で **signingCertAlias** 属性の値に指定したわかりやすい名前を指定します。次に例を示します。

```
<Attribute name="signingCertAlias">
<Value>MyFedlet</Value>
```

5 **.NET Fedlet** のサービスプロバイダのメタデータファイル (**sp.xml**) で、署名鍵用の **KeyDescriptor** を追加します。

すでに使用した Microsoft 管理コンソールの証明書スナップインを使用して、KeyDescriptor XML ブロックに挿入する証明書の公開鍵を、Base64 エンコーディングでエクスポートします。この KeyDescriptor は、SPSSODescriptor 内の最初の子要素である必要があります。次に例を示します。

```
<KeyDescriptor use="signing">
             <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                <ds:X509Data>
                   <ds:X509Certificate>
MIICQDCCAakCBEeNB0swDQYJKoZIhvcNAQEEBQAwZzELMAkGA1UEBhMCVVMxEzARBgNVBAgTCkNh
bGlmb3JuaWExFDASBgNVBAcTC1NhbnRhIENsYXJhMQwwCgYDVQQKEwNTdW4xEDAOBgNVBAsTB09w
ZW5TU08xDTALBgNVBAMTBHRlc3QwHhcNMDgwMTE1MTkxOTM5WhcNMTgwMTEyMTkxOTM5WjBnMQsw
CQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTEUMBIGA1UEBxMLU2FudGEgQ2xhcmExDDAK
BgNVBAoTA1N1bjEQMA4GA1UECxMHT3BlblNTTzENMAsGA1UEAxMEdGVzdDCBnzANBgkqhkiG9w0B
AQEFAAOBjQAwgYkCgYEArSQc/U75GB2AtKhbGS5piiLkmJzqEsp64rDxbMJ+xDrye0EN/q1U5Of\+
RkDsaN/igkAvV1cuXEgTL6RlafFPcUX7QxDhZBhsYF9pbwtMzi4A4su9hnxIhURebGEmxKW9qJNY
Js0Vo5+IgjxuEWnjnnVgHTs1+mq5QYTA7E6ZyL8CAwEAATANBgkqhkiG9w0BAQQFAAOBgQB3Pw/U
QzPKTPTYi9upbFXlrAKMwtFf2OW4yvGWWvlcwcNSZJmTJ8ARvVYOMEVNbsT4OFcfu2/PeYoAdiDA
cGy/F2Zuj8XJJpuQRSE6PtQqBuDEHjjmOQJ0rV/r8mO1ZCtHRhpZ5zYRjhRC9eCbjx9VrFax0JDC
/FfwWigmrW0Y0Q==
                   </ds:X509Certificate>
                </ds:X509Data>
             </ds:KeyInfo>
</KeyDescriptor>
```

6 **.NET** アプリケーションに関連づけられたアプリケーションプールを再起動します。

.NET Fedlet のシングルログアウト (CR 6928528 および CR 6930472)

.NET Fedlet はアイデンティティープロバイダで開始されたシングルログアウトと、サービスプロバイダで開始されたシングルログアウトの両方をサポートします。シングルログアウトを実装するため、.NET Fedlet のサンプルアプリケーションには `asp.net/SampleApp` フォルダに `logout.aspx` ファイルおよび `spinitiatedslo.aspx` ファイルが含まれています。Fedlet のシングルログアウト機能のしくみを確認するには、.NET Fedlet サンプルアプリケーションを配備します。

▼ **.NET Fedlet** サービスプロバイダアプリケーションでシングルログアウトを設定するには、次の手順に従います。

1. **.NET Fedlet** をまだ設定していない場合は、**Readme** ファイルの手順に従います。
2. **.NET** アプリケーションのパブリックコンテンツ内にある **logout.aspx** ファイルおよび **spinitiatedslo.aspx** ファイルをコピーします。
3. アプリケーションの設定ファイルに次の変更を加えます。
   - `sp.xml` ファイル内で、`logout.aspx` ファイルへのパスが、アプリケーションで使用しているファイルの正しい場所を指定していることを確認します。
   - `idp.xml` ファイルで (またはアイデンティティープロバイダの設定中に) `spinitiatedslo.aspx` ファイルへのパスが、アプリケーションで使用しているファイルの正しい場所を指定していることを確認します。
4. ログアウト要求とログアウト応答に署名をする場合、**sp-extended.xml** ファイルおよび **idp-extended.xml** ファイルで、次の属性を **true** に設定します。
   - `wantLogoutRequestSigned`
   - `wantLogoutResponseSigned`
5. **Fedlet** サービスプロバイダのメタデータファイル **(sp.xml)** をアイデンティティープロバイダにインポートします。

   アイデンティティープロバイダの管理者にも Fedlet サービスプロバイダのシングルログアウトを設定したことを伝えて、アイデンティティープロバイダ側で必要な追加の設定変更ができるようにします。

.NET Fedlet サービスプロバイダ開始のシングルサインオン (CR 6928525)

.NET Fedlet は SAMLv2 のサービスプロバイダが開始するシングルサインオン (SSO) をサポートします。さらに、.NET Fedlet がアーティファクトを受信したあとで、SOAP 経由でアイデンティティープロバイダのアーティファクト解決サービスを発行してアーティファクトを解決できるようにするため、アーティファクトのサポートが必要になります。

.NET Fedlet のサンプルアプリケーションは、シングルサインオンの設定方法を示しています。アプリケーションに必要なアーティファクトをインストールしたら、アイデンティティープロバイダによる認証成功後に SAMLv2 応答を含んだ HTTP POST を受信するために、特定の URI が必要になります。次のコーディング例は、.NET アプリケーションでこの情報を取得する方法を示しています。

例 4–2 .NET Fedlet アプリケーションで AuthnResponse を取得するコーディング例

```
AuthnResponse authnResponse = null;
try
{
      ServiceProviderUtility spu = new ServiceProviderUtility(Context);
      authnResponse = spu.GetAuthnResponse(Context);
}
catch (Saml2Exception se)
{
      // invalid AuthnResponse received
}
catch (ServiceProviderUtilityException spue)
{
      // issues with deployment (reading metadata)
}
```

アプリケーションが SAMLv2 を受信したあと、authnResponse オブジェクトが表明情報とともに生成されます。サンプルアプリケーションでは、このオブジェクトから属性および被認証者の情報を取得する方法を示しています。

.NET Fedlet による複数のアイデンティティープロバイダとディスカバリサービスのサポート (CR 6928524)

.NET Fedlet は、複数のアイデンティティープロバイダとアイデンティティープロバイダのディスカバリサービスをサポートします。

配備によっては、Oracle OpenSSO 8.0 Update 2 などの複数のアイデンティティープロバイダで .NET Fedlet を設定する場合があります。追加する各アイデンティティープロバイダに対して、次の作業を実行します。

▼ 複数のアイデンティティープロバイダ用に .NET Fedlet を設定する

1 追加のアイデンティティープロバイダから **XML** メタデータファイルを取得します。

2 追加のアイデンティティープロバイダのメタデータファイルを **idp** *n* **.xml** という名前にします。*n* は追加するアイデンティティープロバイダの番号です。たとえば、**2** 番目のアイデンティティープロバイダの場合は **idp2.xml**、**3** 番目の場合は **idp3.xml** という名前になります。この手順では、**idp2.xml** というファイル名にします。

3 手順 **2** の **idp2.xml** ファイルをアプリケーションの **App_Data** フォルダにコピーします。

4 この新しいアイデンティティープロバイダを **.NET Fedlet** のトラストサークルに追加します。

新しいアイデンティティープロバイダを既存のトラストサークルに追加するには、次の手順に従います。

アプリケーションの App_Data フォルダにある fedlet.cot ファイルで、新しい IDP のエンティティ ID (idp2.xml メタデータファイルの entityID 属性で示される) を sun-fm-trusted-providers 属性の値に追加します。追加するときの区切り文字にはコンマ (,) を使用します。

新しいアイデンティティープロバイダを新しいトラストサークルに追加するには、次の手順に従います。

a. アプリケーションの **App_Data** フォルダに **fedlet2.cot** という名前の新しいファイルを作成します。既存の **fedlet.cot** をテンプレートとして使用して、**cot-name** 属性の値を新しいトラストサークルの名前 **(cot2** など**)** に変更します。 新しいアイデンティティープロバイダのエンティティ **ID** と **Fedlet** のエンティティ **ID** の両方を、**sun-fm-trusted-providers** 属性の値として含めます。**2** つのエンティティ **ID** はコンマ **(,)** で区切ります。

b. **sp-extended.xml** ファイルで、新しいトラストサークルの名前を **cotlist** 属性の値に追加します。たとえば、**cot2** という名前のトラストサークルの場合、次のようになります。

```
<Attribute name="cotlist">
<Value>saml2cot</Value>
<Value>cot2</Value>
</Attribute>
```

5 アプリケーションの **App_Data** フォルダで、新しいアイデンティティーの拡張メタデータとして、**idp2-extended.xml** ファイルを新規作成します。既存の **idp-extended.xml** ファイルをテンプレートとして使用し、**entityID** を新しいアイデンティティープロバイダのエンティティ **ID** に変更します。アイデンティティープロバイダ用に新しいトラストサークルが作成されている場合は、**cotlist** 属性の値をそ

のトラストサークルの名前に変更します。追加のアイデンティティープロバイダがリモートアイデンティティーであることを確認します。

6 **Fedlet .NET** アプリケーションに関連づけられたアプリケーションプールを再起動します。

7 **Fedlet** メタデータ **XML** ファイル **(`sp.xml`)** を追加のアイデンティティープロバイダにインポートして、アイデンティティープロバイダのエンティティと同じトラストサークルに追加する必要があります。**`sp.xml`** ファイルをアイデンティティープロバイダにインポートするか、そのファイルをアイデンティティープロバイダの管理者に渡してインポートしてもらいます。

.NET Fedlet によるアイデンティティープロバイダのディスカバリサービスのサポート (CR 6928524)

このシナリオでは、トラストサークル内の複数のアイデンティティープロバイダで .NET Fedlet が設定されているので、優先するアイデンティティープロバイダを決定するためにアイデンティティープロバイダのディスカバリサービスを使うよう Fedlet を設定します。

.NET Fedlet を使用して、ディスカバリサービスを使用中のアイデンティティープロバイダ用に設定する必要があります。Oracle OpenSSO 8.0 Update 2 でアイデンティティープロバイダのディスカバリサービスを設定するための情報については、次のドキュメントコレクションを参照してください: `http://docs.sun.com/coll/1767.1`。

▼ **.NET Fedlet** を設定してアイデンティティープロバイダのディスカバリサービスを使用するには、次の手順に従います。

1 **.NET Fedlet** の **`fedlet.cot`** ファイルで、**`sun-fm-saml2-readerservice-url`** プロパティーを **SAMLv2 Reader Service URL** に設定します。次に例を示します。

```
sun-fm-saml2-readerservice-url=http://discovery.common.com/opensso/saml2reader
```

2 **.NET Fedlet** アプリケーションに関連づけられたアプリケーションプールを再起動します。

Oracle OpenSSO Fedlet の一般的な問題と回避方法

作成予定

マニュアルの正誤表

Fedlet の Java API リファレンスは、次のドキュメントコレクションの『Oracle OpenSSO Enterprise 8.0 Java API Reference』で利用可能です。http://docs.sun.com/coll/1767.1

注 – `getPolicyDecisionForFedlet` メソッドは OpenSSO 8.0 Update 2 リリースではサポートされません。

◆ ◆ ◆ 第 5 章

OpenSSO 8.0 Update 2 と Oracle Access Manager の統合

この章では、OpenSSO 8.0 Update 2 と Oracle Access Manager 10g または 11g を使用してシングルサインオンを実装する手順を説明します。この情報は、『Sun OpenSSO Enterprise 8.0 Integration Guide』の第 3 章「Integrating Oracle Access Manager」に記載されている情報を補足するものです。 このユースケースでは、Oracle Access Manager セッションを活用することで、OpenSSO によって保護されたアプリケーションに対してシングルサインオンを実現します。設定された OpneSSO 認証モジュールは、Oracle Access Manager セッションに基づいて OpenSSO セッションを生成します。

統合手順の概要

1. 63 ページの「開始する前に」
2. 統合部分の展開
3. OpenSSO で Oracle Access Manager のソースファイルを構築する
4. 67 ページの「(省略可能) Oracle Access Manager での OpenSSO の認証スキームの構築」
5. 68 ページの「Oracle Access Manager および Oracle OpenSSO STS を使用したシングルサインオンの設定」
6. 71 ページの「シングルサインオンをテストする」
7. 71 ページの「(省略可能) Oracle Access Manager への Oblix AuthScheme のインストール」

開始する前に

Oracle Access Manager との統合のために OpenSSO 8.0 Update 2 をインストールする前に、次のコンポーネントにアクセスできることを確認します。

opensso.zip	opensso.war ファイル、統合ソースコード、設定ファイル、および OpenSSO

	8.0 Update 2 のインストールと設定に必要なその他のファイルが含まれている ZIP ファイル。
OpenSSO エージェント	OpenSSO によって保護されているアプリケーションが Oracle Access Manager によって確立された認証セッションを実際に使用する際に OpenSSO エージェントが利用されます。
Oracle Access Manager 10g または 11g	Oracle の Web サイトから Oracle Access Manager をダウンロードします。Oracle Fusion Middleware 11gR1 Software ダウンロードのページを参照してください。
Oracle Web Gate 10g または 11g	OpenSSO と Oracle Webgate の両方でサポートされているコンテナに Oracle Webgate をダウンロードします。現時点では、両方の製品でサポートされているコンテナは Sun Web Server 7.x のみです。Oracle Fusion Middleware 11gR1 Software ダウンロードのページを参照してください
Oracle Access Manager SDK 10g または 11g	Oracle Access Manager をダウンロードします。Oracle Access Manager と統合する場合、OpenSSO 認証モジュールのコンパイルと構築には SDK が必要です。 Oracle Fusion Middleware 11gR1 Software ダウンロードのページを参照してください
OpenSSO C-SDK 2.2	(省略可能) Oracle Access Manager 自体で認証モジュールを作成して OAM セッションを生成するには、OpenSSO C-SDK が必要です。これは OpenSSO の観点からは一般的なユースケースではない可能性があります。『Sun OpenSSO Enterprise 8.0 C API Reference for Application and Web Policy Agent Developers』の「Where is the C SDK?」を参照してください

統合部分の展開

opensso/integrations/oracle ディレクトリには、カスタム認証モジュールやその他のプラグインをコンパイルおよび構築するためのソースおよび設定が含まれています。ユースケースのオプションおよび関連情報については、『Sun OpenSSO Enterprise 8.0 Integration Guide』の第 3 章「Integrating Oracle Access Manager」を参照してください。次の表に、opensso/integrations/oracle ディレクトリ内のファイルと、各ファイルの説明を示します。

README.html	今開いているこのファイルです。
build.xml	OpenSSO で Oracle Access Manager のカスタム認証モジュールを構築するための Ant 構築ファイル
config	OpenSSO で Oracle Access Manager の認証モジュールを作成するために必要な設定ファイル。 ■ OblixAuthService.xml Oracle Access Manager 認証モジュールの認証サービスファイル ■ OblixAuthModule.xml Oracle Access Manager の認証モジュールコールバック。 これはデフォルトで空のファイルですが、設定をするために存在しなければならないものです。 ■ OblixAuth.properties 認証のための国際化キーを格納するプロパティーファイル
lib	このディレクトリはデフォルトで空です。この lib ディレクトリには、ソースライブラリをコンパイルするために次のライブラリが含まれている必要があります。 ■ jobaccess.jar Oracle Access Manager SDK からこのファイルをコピーします。 ■ openfedlib.jar、amserver.jar、および opensso-sharedlib.jar opensso.war からこれらのファイルをコピーします ■ servlet.jar or javaee.jar

GlassFish の lib ディレクトリをコピーします。理想的には、javax.servlet.http.Cookie などの標準の Java EE クラスを持つ JAR ファイルが適当です。

source

次のソースファイルを含むディレクトリです。

- com/sun/identity/authentication/oblix/OblixAuthModule.java
- com/sun/identity/authentication/oblix/OblixAuthModule.java
- com/sun/identity/authentication/oblix/OblixPrincipal.java
- com/sun/identity/saml2/plugins/OAMAdapter.java

  このクラスは、SAML サービスプロバイダの SAML2 プラグインアダプタです。このクラスは OpenSSO のセッションサービスを使用して Oracle Access Manager に対してリモート認証を実施します。

oamauth (オプション)

このディレクトリには OpenSSO の Oblix 認証スキームのソースファイルが含まれています。これは C ベースの認証モジュールで、検証のために OpenSSO C-SDK を利用しています。

- oam/solaris/authn_api.c

  このファイルは OpenSSO の Oblix カスタム認証スキームを実装します。

- oam/solaris/include/*.h

  認証スキームをコンパイルするすのに必要なヘッダーファイルすべて。

- oam/solaris/AMAgent.properties

  OpenSSO エージェントの設定ファイルのサンプル。これは、OpenSSO セッションを検証するための認証スキームに必要です。

OpenSSO で Oracle Access Manager のソースファイルを構築する

Ant スクリプトを使用してソースファイルを構築します。対応している Ant スクリプトを設定して、PATH で設定する必要があります。

▼ Oracle Access Manager のソースファイルを構築する

1 次のコマンドを実行します。

```
cd $openssozipdir/integrations/oracle; ant -f build.xml
```

このコマンドによって、リソースファイルが構築され、`fam_oam_integration.jar` が `$openssozipdir/integrations/oracle/dist` ディレクトリに生成されます。

2 認証モジュールを **OpenSSO WAR** ファイルにバンドルします。

a. 一時的なディレクトリを作成して、**`opensso.war`** を解凍します。次に例を示します。

```
# mkdir /export/tmp
# cd /export/tmp
# jar -xvf opensso.war
```

これ以降は、`/export/tmp` を WAR のステージング領域として使用し、マクロ `$WAR_DIR` とともに表します。

b. **`$openssozipdir/integrations/oracle/dist/fam_oam_integration.jar`** を **`$WAR_DIR/WEB-INF/lib`** にコピーします。

c. **`$openssozipdir/integrations/oracle/config/OblixAuth.properties`** を **`$WAR_DIR/WEB-INF/classes`** にコピーします。

d. **`$openssozipdir/integrations/oracle/config/OblixAuthModule.xml`** を **`$WAR_DIR/config/auth/default`** および **`$WAR_DIR/config/auth/default_en`** ディレクトリにコピーします。

e. **`jar cvf opensso.war`** を使用して **`$WAR_DIR`** の **`opensso.war`** を再**WAR**化します。

例: 作成予定

(省略可能) Oracle Access Manager での OpenSSO の認証スキームの構築

注: これは一般的なユースケースではありません。SAML2 サービスプロバイダのユースケースなど、構築する必要がある場合にのみ実施します。

Oblix 認証スキームを構築するには、makefile をカスタマイズする必要があります。また、これは C ベースの認証モジュールのため、オペレーティングシステムに依存します。

▼ Oracle Access Manager で OpenSSO の認証スキームを構築する

始める前に　認証スキームファイルは $openssozipdir/integrations/oracle/oamauth/solaris ディレクトリの下にあります。

1. **OpenSSO C-SDK 2.2** バージョンをダウンロードして設定します。

   authn_api.c ファイルには、AMAgent.propeties file への参照が含まれます。ファイルを適宜編集します。

2. 環境に合わせて **makefile** をカスタマイズします。

   たとえば、gcc コンパイルの場所を指定します。また、LDFLAGS を編集して OpenSSO C-SDK の lib を指定します。

3. **make** コマンドを実行します。

   make コマンドによって authn_api.so ファイルが生成されるはずです。

Oracle Access Manager および Oracle OpenSSO STS を使用したシングルサインオンの設定

▼ Oracle Access Manager および Oracle OpenSSO 8.0 Update 2 を使用してシングルサインオンを設定する

開始する前に: Sun Java System Web Server 7.x がすでにインストールおよび設定されている必要があります。Web サーバーのインストール手順については、Sun Java System Web Server Documentation Wiki を参照してください。

1. **Sun Java System Web Server 7.x** に **OpenSSO** をインストールします。

2 **OpenSSO** のポリシーエージェントをサポートされているコンテナにインストールして、エージェントが **OpenSSO** で動作するように設定します。

手順については、『Sun OpenSSO Enterprise Policy Agent 3.0 User's Guide for J2EE Agents』または『Sun OpenSSO Enterprise Policy Agent 3.0 User's Guide for Web Agents』『Sun OpenSSO Enterprise Policy Agent 3.0 User's Guide for Web Agents』を参照してください。

3 **Oracle Access Manager** をインストールおよび設定します。

『Oracle Access Manager インストレーション・ガイド 10g (10.1.4.3)』を参照してください

4 **Oracle Access Manager** で **Oracle Access Manager SDK** をインストールおよび設定します。

『Oracle Access Manager インストレーション・ガイド 10g (10.1.4.3)』を参照してください

5 **OpenSSO** がインストールされた同一の **Web** コンテナに **Oracle Webgate** をインストールします **(Sun Web Server 7.x)**。

OpenSSO を設定して、OpenSSO Web アプリケーションの `deployURI/UI/*` のみが保護されるようにします。次に例を示します。`/opensso/UI/.../*`

Oracle Access Manager のポリシー、リソース、およびその他の設定の詳細については、『Oracle Access Manager Administration Guide』を参照してください。 OpenSSO Enterprise で、その他すべての URL の保護を解除します。これは、簡単なシングルサインオンの統合シナリオを実現するためですが、完全な統合とその他の配備の依存関係に基づいてポリシーを評価します。

6 **OpenSSO** で認証モジュールを設定します。

a. **OpenSSO** コンソールにアクセスします。

認証のため、ブラウザは Oracle Access Manager へリダイレクトします。認証成功後、ログインページが OpenSSO に表示されます。OpenSSO 管理者のユーザー名とパスワードを使用してログインします。

b. **Oracle** 認証モジュールのサービス **XML** ファイルを、**OpenSSO** の設定にインポートします。

認証モジュールサービスは、コマンド行の `ssoadm` ユーティリティーから、およびブラウザベースの ssoadm.jsp から読み込めます。

c. **`http://host:port/opensso/ssoadm.jsp`** にアクセスします。

d. 「**create-service**」オプションを選択します。

e. **`$openssozipdir/integrations/oracle/config/OblixAuthService.xml`** から **XML** ファイルをコピー＆ペーストして、「**Submit**」をクリックします。
これによって、認証モジュールのサービスが OpenSSO 設定に読み込まれます。

f. 認証モジュールを認証コアサービスに登録します。
このコアサービスにはオーセンティケータのリストが含まれています。`register-auth-module` オプションを `http://host:port/opensso/ssoadm.jsp` で選択します。認証モジュールのクラス名として、`com.sun.identity.authentication.oblix.OblixAuthModule` と入力します。

g. 認証モジュールがデフォルトのレルムに登録されていることを確認します。
URL http://host:port/opensso を使用して OpenSSO にアクセスします。OpenSSO コンソールで、デフォルトのレルムをクリックして、「Authentication」タブをクリックします。「New」をクリックして OblixAuth という名前の新しい認証モジュールを作成します。

h. 「**Authentication**」タブで、**OblixAuth** 認証モジュールを選択します。
Oblix SDK ディレクトリを設定します。「Check Remote User Header Only」を有効にして、リモートヘッダー名として `OAM_REMOTE_USER` を指定します。このパラメータは配備に合わせて設定可能です。

7 **(省略可能) OpenSSO** のコア認証サービスで「**Ignore Profile**」オプションを有効にします。
OpenSSO コンソールで、「Configuration」>「Core」>「Realm Attributes」>「User Profile」の順に選択します。「Ignored」を選択して「Save」をクリックします。
この設定により、OpenSSO が認証成功後に既存のユーザープロファイルを検索しないようにします。ただし、OpenSSO と Oracle Access Manager によって使用されるユーザーリポジトリが完全に同一である場合、この手順は不要です。「Admin Console」>「Configuration」>「Core」>「Realm Attributes」>「User Profile」の順に選択します。「Ignored」を選択して「Save」をクリックします。

8 **Web** サーバーの起動スクリプトを編集して、**Oracle Access Manager SDK** の共有ライブラリを含めます。
`startserv` スクリプト内の `LD_LIBRARY_PATH` を更新して `$ACCESSDKDIR/oblix/lib` の共有ライブラリを含めます。

9 **OpenSSO** および **Oracle Webgate** の両方を含む **Sun Web Server** を再起動します。

10 **Web Agent** のログイン **URL** を **`http://openssohost:openssoport/deployURI/UI/Login?module=OblixAuth`** に更新します。

シングルサインオンをテストする

OpenSSO によって保護されているアプリケーションの保護リソースにアクセスします。まだ認証されていない場合、ブラウザは Oracle Access Manager のログインページにユーザーをリダイレクトします。ログイン成功後、 OpenSSO セッションが作成され、最後にポリシーエージェントに保護されたアプリケーションの URL にリダイレクトされます。ポリシーに基づいて、保護されたアプリケーションへのアクセスが許可または拒否されます。

(省略可能) Oracle Access Manager への Oblix AuthScheme のインストール

この手順は、OpenSSO セッションの認証後に Oracle Access Manager セッションを生成しなければならない場合に便利です。関連するユースケースについての情報は、『Sun OpenSSO Enterprise 8.0 Integration Guide』の第 3 章「Integrating Oracle Access Manager」を参照してください。

Oblix 認証スキームは C 認証モジュールとして公開され、この認証スキームは OpenSSO C-SDK 2.2 バージョンを使用して OpenSSO セッションを検証します。Oblix の OpenSSO 認証スキームは、OpenSSO クライアントサイド設定に `AMAgent.properties` の設定を使用します。 このファイルは 認証モジュールを設定する前にカスタマイズする必要があります。構築手順でこのファイルの 場所を指定します。認証スキームを設定する前に、コンパイルされた `authn_api.so` およびその他の C-SDK ライブラリを `$OAM_INSTALL_DIR/access/oblix/lib` ディレクトリコピーする必要があります。『『Sun OpenSSO 8.0 Integration Guide』』には、Oracle 認証スキームの設定方法を図説したスクリーンショットの例が記載されています。ただしこのスクリーンショットは参照目的でのみ使用してください。詳細については、Oracle Access Manager の最新マニュアルを参照してください。

OpenSSO 8.0 Update 2 と Oracle Access Manager の統合

この節では、OpenSSO 8.0 Update 2 と Oracle Access Manager 10.1.4.0.1 または 11g を使用してシングルサインオンを実装する手順を説明します。この情報は、『Sun OpenSSO Enterprise 8.0 Integration Guide』の第 3 章「Integrating Oracle Access Manager」に記載されている情報を補足するものです。 このユースケースでは、Oracle Access Manager セッションを活用することで、OpenSSO によって保護されたアプリケーションに対してシングルサインオンを実現します。設定された OpneSSO 認証モジュールは、Oracle Access Manager セッションに基づいて OpenSSO セッションを生成します。