Red Hat JBoss Enterprise Application Platform 7.3

SAML v2 での SSO のセットアップ方法

SAML 2.0 を使用して Red Hat JBoss Enterprise Application Platform へのシングルサインオンユーザーアクセスを設定して管理するための手順

Last Updated: 2023-02-28

Red Hat JBoss Enterprise Application Platform 7.3 SAML v2 での SSO のセットアップ方法

SAML 2.0 を使用して Red Hat JBoss Enterprise Application Platform へのシングルサインオンユーザーアクセスを設定して管理するための手順

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律上の通知

概要

本書の目的は、SAML v2 によるシングルサインオンについて深掘りし、Red Hat JBoss Enterprise Application Platform (JBoss EAP) 内で設定および設定する方法を提供することです。本書をお読みいただく前に、JBoss EAP セキュリティーアーキテクチャーガイドをお読みください。また、このガイドに記載されているシングルサインオンおよび SAML v2 の情報を理解してください。本書を読み終わることで、シングルサインオンおよび SAML v2、JBoss EAP との関係、および設定方法を深く実践的に理解することができます。

目次

第1章 SAML V2 DEEPER DIVE でのシングルサインオン

シングルサインオンおよび SAML の基本については、JBoss EAP **セキュリティーアーキテクチャー** ガイドで説明されています。このセクションでは、SAML v2 およびシングルサインオンに関連するコンポーネントの詳細を説明します。

1.1. SAML V2 とは

SAML (Security Assertion Markup Language) は、通常はアイデンティティープロバイダーとサービスプロバイダーである 2 者が認証および承認情報を交換できるようにするデータ形式とプロトコルです。この情報は、アサーションを含む SAML トークンの形式で交換され、サービスプロバイダーによる認証の影響を受けるよう ID プロバイダーによって発行されます。複数のサービスプロバイダーと共にアイデンティティープロバイダーから発行される SAML トークンをサブジェクトが使用および再利用する機能により、SAML v2 はブラウザーベースのシングルサインオンを容易にします。

1.1.1. ブロックの構築

SAML に留意すべき最も重要な概念は、エンティティー間でセキュリティーアサーションを渡すことだけです。SAML には、このタスクを実行するために使用するコンポーネントが複数含まれています。

1.1.1.1. エンティティー

エンティティーは、アサーションの作成および受け渡しに関与するすべての当事者です。SAML には、以下の個別のエンティティーという概念があります。

subject
: **subject** は **principal** とも呼ばれ、SAML によってセキュア化された **service provider** のリソースへのアクセスを要求します。

service provider
: **service provider** (**SP**) には、**identity provider** から必要な **subject** のアイデンティティーのアサーションとして検証が必要です。

identity provider
: **identity provider** (**IDP**) は、**service providers** による認証および承認の決定に使用できる、**subject** に関するトークンの形式で、アサーションのセットを提供します。

つまり **subject** は発行されたアサーションを取得し 、**identity providers** はこれらのアサーションを発行して、**service providers** はこれらのアサーションを使用して **subject** を認証および承認します。

1.1.1.2. セキュリティーアサーション

セキュリティーアサーションは、サブジェクトに関するアイデンティティープロバイダーが発行する一連のステートメントです。サービスプロバイダーはこれらのアサーションを使用して、サブジェクトに関するアクセス制御の決定を行います。ステートメントは以下の形式を取ることができます。

認証
: **Authentication** アサーションは、特定の時点で指定されたメソッドを使用してサブジェクトが正常に認証されたことをアサートします。認証済みのサブジェクトに関する他の情報が含まれる認証コンテキストも authentication ステートメントで指定できます。

Attribute
: **Attribute** アサーションは、サブジェクトに特定の属性があることをアサートします。

Authorization Decision

Authorization Decision アサーションは **accept** または **deny** リソース上のサブジェクトの承認リクエストに対して応答 (承認または拒否) をアサートします。

例

> This user logged in as Sarah at 9:30 using a username and password. Sarah is a member of the Managers group. Sarah is accepted to access the Employee Information resource.

- **This user logged in as Sarah at 9:30 using a username and password** というステートメントは **Authentication** アサーションです。
- **Sarah is a member of the Managers group** というステートメントは、**Attribute** アサーションです。
- **Sarah is accepted to access the Employee Information resource** ステートメントは **Authorization Decision** アサーションです。

アサーションは SAML トークンとしてパケージ化され、SAML プロトコルを使用してトランスポートされます。

1.1.1.3. プロトコル

SAML プロトコルは、通常リクエストと応答の形式でアサーションがパッケージ化される方法と、これを処理する正しい方法のルールを記述します。これらのルールの後に、リクエストと応答のプロデューサーとコンシューマーの両方を追加する必要があります。リクエストは、認証、属性、または承認の決定について、特定の既知のアサーションまたはクエリーアイデンティティープロバイダーを要求することができます。セキュリティーアサーションを含むリクエストおよび応答メッセージは XML でフォーマットされ、指定されたスキーマに従います。

1.1.1.4. バインディング

SAML バインディングは、SAML プロトコルをトランスポートおよびメッセージングに使用される他の標準プロトコルにマップする方法を指定します。例には以下が含まれます。

- HTTP リダイレクトにマップする SAML バインディング。
- HTTP **POST** にマップする SAML バインディング。
- SAML リクエスト/応答を SOAP リクエストおよび応答にマップする SAML バインディング。

1.1.1.5. プロファイル

SAML プロファイルは、アサーション、プロトコル、バインディングを使用して、Web Browser シングルサインオン、Single Logout、および Assertion Query などの特定のユースケースに対応します。

1.2. SAML V2 とシングルサインオンが動作する仕組み

SAML v2 でのブラウザーベースのシングルサインオンの基本は、JBoss EAP **Security Architecture** ガイドの SAML を使用したブラウザーベースのシングルサインオン および SAML でブラウザーベースのシングルサインオンを使用する複数の Red Hat JBoss Enterprise Application Platform インスタンスと複数のアプリケーション セクションで説明されています。このセクションでは、SAML v2 を使用したブラウザーベースのシングルサインオンに関連する SAML プロファイルおよびバインディングについての詳細情報を提供します。

1.2.1. Web ブラウザーシングルサインオンプロファイル

Web Browser シングルサインオン プロファイルは、ブラウザーエージェントの形式で、ブラウザーベースのシングルサインオンを処理する方法で IDP、SP、およびプリンシパルを指定します。SP および IDP には、Web Browser シングルサインオンプロファイルで使用できるバインディングが複数あり、多数のフローが可能です。また、このプロファイルは IDP または SP のいずれかから開始されるメッセージフローをサポートします。このプロファイルは、SAML アサーションを SP にプッシュする IDP、または IDP からアサーションをプルする SP もサポートします。SP または IDP のいずれかから開始するフローは、JBoss EAPSecurity Architecture ガイド で詳細に説明されています。IDP からプッシュされた SAML アサーションは HTTP **POST** メッセージまたは HTTP リダイレクトを使用します。SP がプルする SAML アサーションでは、アーティファクトを受信側に送信し、次にアサーションを取得するために逆参照されます。

Web Browser Single シングルサインオンプロファイルの基本フローは以下のとおりです。

1. SP へのプリンシパル HTTP 要求。
   プリンシパルは、最初に HTTP ユーザーエージェント (例: ブラウザー) を使用して SP でセキュアなリソースにアクセスしようとします。プリンシパルが有効なセキュリティーコンテキストで SAML トークンをすでに発行している場合、SP はプリンシパルを許可または拒否します。これは最後のステップです。そうでなければ、SP は認証要求の IDP の場所を特定します。

2. SP は IDP を決定します。
   SP は、IDP と SP の推奨バインディングをサポートするエンドポイントを見つけます。これにより、SP が認証要求を IDP に送信できます。このプロセスの具体的な方法は、実装ごとに異なる場合があります。

3. プリンシパルを使用して SP から IDP に発行された認証要求。
   SP が IDP の場所とエンドポイントを判別すると、SP は **<AuthnRequest>** メッセージの形式で認証リクエストを発行します。これはユーザーエージェントによって IDP に配信されます。HTTP Redirect、HTTP **POST**、または HTTP Artifact SAML バインディングを使用すると、ユーザーエージェントを使用してメッセージを IDP に転送することができます。

4. IDP はプリンシパルを識別します。
   Authentication Request がプリンシパルによって IDP に配信されると、プリンシパルは IDP によって識別されます。識別方法は Web ブラウザーのシングルサインオンプロファイルによって明確に定義されていません。これは、**FORM** を使用した認証、既存のセッション情報、kerberos 認証など、数多くの方法で実現できます。

5. IDP は SP への応答を発行します。
   プリンシパルが特定されると、IDP は **<response>** メッセージの形式で応答を発行し、ユーザーエージェントを使用してプリンシパルによるアクセスを許可または拒否するために SP に返送します。このメッセージは、少なくとも単一の認証アサーションを含み、エラーを示すのにも使用できます。HTTP **POST** または HTTP Artifacts はこのメッセージの転送に使用できますが、ほとんどのユーザーエージェントの URL 長制約により HTTP Redirect を使用することはできません。ユーザーエージェントが、SP ではなく IDP に直接アクセスしようとするなど、IDP ベースのフローを開始した場合は、このステップでプロセスが開始されます。成功すると、HTTP **POST** または HTTP アーティファクトが IDP に事前設定されたロケーションに送信されます。

6. SP はプリンシパルへのアクセスを許可または拒否します。
   SP が Response を受信すると、セキュリティーコンテキストを作成して要求されたリソースへのアクセスをプリンシパルに付与したり、アクセスを拒否したり、独自のエラー処理を行ったりする場合があります。

注記

JBoss EAP は SAML アーティファクトバインディングをサポートしません。

HTTP REDIRECT VS。POST BINDING

HTTP Redirect バインディングは、HTTP **GET** リクエストと URL クエリーパラメーターを使用してプロトコルメッセージを送信します。この方法で送信されたメッセージは、受信側によって送信およびデコードされる前に URL と Base-64 エンコードされます。HTTP **POST** バインディングは、フォームデータを使用してメッセージを送信し、メッセージに対して base-64 エンコード/エンコードも送信します。SP および IDP の両方がリダイレクトまたは **POST** バインディングを使用してメッセージを送受信できます。特定のシナリオでの URL の長さの制限により、通常 HTTP Redirect は短いメッセージを渡す際に使用され、HTTP **POST** は長いメッセージを渡す際に使用されます。

1.2.2. Global Logout Profile

Global Logout Profile を使用すると、IDP と SP のセットで認証されたプリンシパルをログアウトでき、そのアサーションを、関連する 1 つ以上の IDP と SP に伝播できます。

プリンシパルが IDP で認証されると、プリンシパルと IDP が認証セッションを確立します。IDP は、その認証に基づいて、さまざまな SP に対してアサーションを発行したり、公開したりすることがあります。プリンシパルが SP 内のセキュアなリソースにアクセスしようとすると、SP は IDP から発行されたアサーションに基づいて、プリンシパルで追加のセッションを確立することを選択する可能性があるため、IDP に依存します。

セッションまたはセッションのセットが作成されると、さまざまな方法でプリンシパルがセッションから個別にログアウトされたり、グローバルログアウトプロファイルを使用してすべてのセッションやすべての SP および IDP から一度にログアウトしたりできます。Global Logout Profile は、フローにおいて HTTP Redirect、HTTP **POST**、または HTTP Artifact バインディングを使用できます。また、本書では扱われない特定のケースで SOAP バインディングを使用することもできます。

注記

Single Logout Profile は、Global Logout Profile の同意語として使用できます。

注記

JBoss EAP は SAML アーティファクトバインディングをサポートしません。

Web ブラウザーのシングルサインオンプロファイルフローと同様に、グローバルログアウトプロファイルフローは IDP または SP のいずれかで開始できます。

Global Logout Profile の基本的なフローは次のとおりです。

1. Session Participant によって IDP 発行されたログアウト。
   Service Providers やその他パーティーなどのセッションの参加者は、独自のセッションをプリンシパルとともに終了させ、**<LogoutRequest>** メッセージの形式で、Logout Request を、プリンシパルのセキュリティーアサーションを最初に発行した IDP に送信します。この要求は IDP と依拠当事者 (relyiing party) 間で直接送信することも、プリンシパルのユーザーエージェントをパススルーとして使用することで間接的に送信することもできます。

2. IDP は、Session Participant を特定します。
   IDP が Logout Request を受信すると、そのリクエストを使用して、IDP がセッション認証またはセッション参加者として所有するセッションを含め、どの依拠当事者に依存しているセッ

ションを終了するかを決定します。IDP は各セッションに対して、依存側に対してログアウトリクエストを発行し、各社からの Logout Response を待ってから、新しい Logout Response を元のセッション参加者に発行します。IDP で Global Logout Profile フローが開始された場合、フローはこのステップで開始し、その他のメカニズムを使用してセッションと SP が決定されます。

3. IDP が発行するログアウト。
   IDP がすべてのセッションと関連する当事者を判別すると、ログアウトリクエストを **<LogoutRequest>** メッセージの形式でログアウト要求を送信し、依拠当事者に依存してログアウト応答を待機します。これらの要求は IDP と依存側間で直接送信することも、プリンシパルのユーザーエージェントを介して間接的に送信することもできます。

4. Session Participant または Authority.が発行するログアウト応答をログアウトします。
   IDP 自体を含む各依存側は、Logout Request の IDP によってダイレクトされたセッションの終了を試み、**<LogoutResponse>** メッセージの形式で Logout Response を返すことがあります。Logout Request と同様に、応答は依存側と IDP 間で直接発行することも、プリンシパルのユーザーエージェントから間接的に発行することもできます。

5. IDP は元の Session Participant への Logout 応答を発行します。
   すべての Logout Response が依存側から受信されると、IDP は **<LogoutResponse>** メッセージの形式で新しいログアウト応答を送信し、ログアウトを要求した元のセッション参加者に戻ります。このフローの他の部分と同様に、この応答は IDP とセッションの参加者間で直接渡されるか、プリンシパルのユーザーエージェントによって間接的に渡すことができます。ログアウト要求が IDP で開始された場合は、この手順は省略されます。

注記

JBoss EAP では、Global Logout Profile の IDP 部分と SP 部分との間の直接の通信はサポートされません。

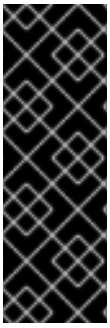
1.2.3. 複数の IDP および Identity Discovery プロファイル

SAML v2 を使用するブラウザーベースのシングルサインオンは、複数の IDP を持つこともサポートし、Web Browser シングルサインオンプロファイルと Global Logout プロファイルの両方で使用できます。複数の IDP が設定されている場合には、Identity Discovery SAML プロファイルを使用してプリンシパルが使用する IDP が決定されます。そのためには、ドメイン情報のあるクッキーと IDP の一覧の読み取りと書き込みを行います。

1.3. 詳細情報

SAML v2 の詳細は、公式 SAML 2.0 仕様 を参照してください。

第2章 SAML V2 での SSO のセットアップ方法



重要

Picketlink を使用して SAML v2 でシングルサインオンを設定することは非推奨となりました。これで、OAuth および OpenID Connect および SAML をサポートする Red Hat シングルサインオン 製品ページで Product Documentation を使用する必要があります。

JBoss EAP でアプリケーションをセキュアにし、OIDC や SAML v2 などのサポート対象のプロトコルでシングルサインオンを Red Hat シングルサインオンと使用する場合は、**Securing Applications and Services Guide** を参照してください。

このセクションでは、JBoss EAP を使用して SAML v2 でシングルサインオンを設定する実際の手順について詳しく説明します。

2.1. コンポーネント

Entities section や JBoss EAP **Security Architecture ガイド** で説明されているように、SAML v2 を使用するブラウザーベースのシングルサインオンにおいては、3 つの **entities** またはパーティーがあります。

- セキュアなリソースへのアクセスを要求するユーザーエージェント (ブラウザー) を使用するプリンシパル。
- セキュリティー保護されたリソースを格納しているサービスプロバイダー。
- プリンシパルにセキュリティーアサーションを発行し、サービスプロバイダーのセキュアなリソースへのアクセスを許可するアイデンティティープロバイダー。

さらに、SAML v2 を使用してブラウザーベースのシングルサインオンをサポートするには、以下が必要です。

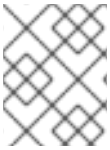
- SP および IDP としてサービスされる個別の Web アプリケーション
- SP および IDP をホストする JBoss EAP インスタンス
- SP および IDP をサポートするセキュリティードメイン

2.2. IDP と SP の設定と設定

ここでは、アプリケーションを SP または IDP のいずれかになるように設定する方法と、それらのアプリケーションをホストする JBoss EAP インスタンスをセットアップする方法について説明します。

2.2.1. IDP の設定

アプリケーションを IDP として動作するように設定するには、以下の手順を実行する必要があります。



注記

アプリケーションを作成およびデプロイする前に、セキュリティードメインを作成および設定する必要があります。

1. IDP のセキュリティードメインを作成します。

IDP はクレデンシャルのプリンシパルを処理し、そのプリンシパルの認証および承認を処理し、その結果に基づいて適切な SAML v2 セキュリティーアサーションを発行します。これには、セキュリティードメインを使用してアイデンティティーストアを設定する必要があります。このセキュリティードメインおよびアイデンティティーストアを作成する唯一の要件は、認証および承認メカニズムが適切に定義されていることです。これは、プロパティーファイル、データベース、LDAP など、さまざまな ID ストアと関連のログインモジュールを使用して IDP アプリケーションをサポートすることができることを意味します。セキュリティードメインの詳細は、JBoss EAP **Security Architecture** ガイドの Security Domains を参照してください。

以下の例では、プロパティーファイルを使用して簡単な **UsersRoles** ログインモジュールを使用します。

セキュリティードメインを作成する管理 CLI コマンド

```
/subsystem=security/security-domain=idp:add(cache-type=default)

/subsystem=security/security-domain=idp/authentication=classic:add

/subsystem=security/security-domain=idp/authentication=classic/login-
module=UsersRoles:add(code=UsersRoles,flag=required,module-options=
[usersProperties=${jboss.server.config.dir}/idp-
users.properties,rolesProperties=${jboss.server.config.dir}/idp-roles.properties])

reload
```

UsersRoles ログインモジュールは、プロパティーファイルを使用して、ユーザー/パスワードとユーザー/ロール情報を保存します。**UsersRoles** モジュールの詳細は、JBoss EAP Login Module Reference を参照してください。この例では、プロパティーファイルには以下が含まれます。

idp-users.properties

```
Eric=samplePass
Alan=samplePass
```

idp-roles.properties

```
Eric=All
Alan=
```

2. IDP の **web.xml** ファイルを設定します。
IDP の **web.xml** ファイルには以下が含まれている必要があります。

- セキュアな領域の URL パターンにマップする **<url-pattern>** が含まれる **<web-resource-collection>** を持つ **<security-constraint>**。任意で、**<security-constraint>** に許可されるロールを明記する **<auth-constraint>** を含めることもできます。
- **FORM** 認証用に設定された **<login-config>**。
- **<auth-constraint>** にロールが指定されている場合、これらのロールを **<security-role>** で定義する必要があります。

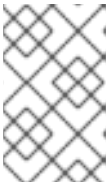
- 必要に応じて、イメージやスタイルなどのログインフォームで使用されるリソースは、追加のセキュリティー制約で、認証前にアクセスできるように、セキュアでない追加のセキュリティー制約で指定できます (例: ログインページ)。

<security-constraint> および **<security-role>** 要素を使用すると、管理者は URL パターンおよびロールを基に制限された領域または無制限の領域をセットアップできます。これにより、リソースをセキュリティーで保護することができ、保護しないこともできます。

<login-config> タグは、ユーザーの認証時に IDP が使用するログインおよびエラーページを定義します。

例: web.xml ファイル

```
<web-app>
  <display-name>IDP</display-name>
  <description>IDP</description>
  <!-- Define a security constraint that gives unlimited access to images -->
  <security-constraint>
    <web-resource-collection>
      <web-resource-name>Images</web-resource-name>
      <url-pattern>/images/*</url-pattern>
    </web-resource-collection>
  </security-constraint>
  <!-- Define a security constraint that requires the All role to access resources -->
  <security-constraint>
    <web-resource-collection>
      <web-resource-name>IDP</web-resource-name>
      <url-pattern>/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
      <role-name>All</role-name>
    </auth-constraint>
  </security-constraint>
  <!-- Define the Login Configuration for this Application -->
  <login-config>
    <auth-method>FORM</auth-method>
    <realm-name>IDP Application</realm-name>
    <form-login-config>
      <form-login-page>/jsp/login.jsp</form-login-page>
      <form-error-page>/jsp/error.jsp</form-error-page>
    </form-login-config>
  </login-config>
  <!-- Security roles referenced by this web application -->
  <security-role>
    <description>The role that is required to log in to the IDP Application</description>
    <role-name>All</role-name>
  </security-role>
</web-app>
```



注記

Welcome ページはアプリケーション内で定義することが推奨されます。デフォルトでは、JBoss EAP は **index.jsp** という名前のファイルを検索しますが、**web.xml** で **<ogg-file-list>** を使用してこれを設定できます。

例: login.jsp ファイル

```
<html>
  <head></head>
  <body>
    <form id="login_form" name="login_form" method="post" action="j_security_check"
enctype="application/x-www-form-urlencoded">
      <center> <p>Welcome to the <b>IDP</b></p> <p>Please login to proceed.</p> </center>
      <div style="margin-left: 15px;">
        <p> <label for="username">Username</label> <br /> <input id="username" type="text"
name="j_username"/> </p>
        <p> <label for="password">Password</label> <br /> <input id="password"
type="password" name="j_password" value=""/> </p>
        <center> <input id="submit" type="submit" name="submit" value="Login"/> </center>
      </div>
    </form>
  </body>
</html>
```

例: error.jsp ファイル

```
<html>
  <head></head>
  <body>
    <p>Login failed, please go back and try again.</p>
  </body>
</html>
```

3. IDP の認証システムを設定します。
この認証システムは、ユーザーの認証と SAML v2 セキュリティーアサーションを発行および検証します。オーセンティケーターの一部は、プリンシパルの認証および承認に使用するセキュリティードメインを定義することで **jboss-web.xml** ファイルに設定されます (手順 1 を参照)。また、**<login-config>** が **web.xml** で指定され、必要な依存関係が宣言されるようにする必要があります。

jboss-web.xml ファイルには以下が必要です。

- 認証または承認に使用されるセキュリティードメインを指定する **<security-domain>**。

例: jboss-web.xml ファイル

```
<jboss-web>
  <security-domain>idp</security-domain>
  <context-root>identity</context-root>
</jboss-web>
```

4. IDP に必要な依存関係を宣言します。
IDP として機能する web アプリケーションは、**org.picketlink** クラスを見つけることができるように依存関係を **jboss-deployment-structure.xml** に定義する必要があります。JBoss EAP は必要なすべての **org.picketlink** と関連クラスを提供し、アプリケーションはそれらを依存関係として宣言して使用することのみが必要になります。

jboss-deployment-structure.xml を使用した依存関係の宣言

```
<jboss-deployment-structure>
  <deployment>
    <dependencies>
      <module name="org.picketlink" services="import"/>
    </dependencies>
  </deployment>
</jboss-deployment-structure>
```

注記

これまでのバージョンの JBoss EAP では、同じ依存関係を宣言していましたが、SAML 認証システムをインストールするためのバルブが宣言されていました。JBoss EAP 7 で Undertow が導入され、**services="import"** を使用して SAML 認証システムをインストールするようになりました。

5. IDP の **picketlink.xml** ファイルを作成および設定します。
picketlink.xml ファイルは authenticator の動作を管理し、アプリケーションの起動時にロードされます。

   ファイルには、少なくとも以下の要素が含まれている必要があります。

   - **<IdentityURL>** を使用した IDP の URL と、IDP で信頼されるホストを定義する **<PicketLinkIDP>**。
   - SAML リクエストおよび応答の処理に必要なハンドラーのセットを定義する **<Handlers>**。

   picketlink.xml ファイルの例

```
<PicketLink xmlns="urn:picketlink:identity-federation:config:2.1">
  <PicketLinkIDP xmlns="urn:picketlink:identity-federation:config:2.1">
    <IdentityURL>${idp.url::http://localhost:8080/identity/}</IdentityURL>
    <Trust>
      <Domains>localhost,example.com</Domains>
    </Trust>
  </PicketLinkIDP>
  <Handlers xmlns="urn:picketlink:identity-federation:handler:config:2.1">
    <Handler
class="org.picketlink.identity.federation.web.handlers.saml2.SAML2IssuerTrustHandler" />
    <Handler
class="org.picketlink.identity.federation.web.handlers.saml2.SAML2LogOutHandler" />
    <Handler
class="org.picketlink.identity.federation.web.handlers.saml2.SAML2AuthenticationHandler" />
    <Handler
class="org.picketlink.identity.federation.web.handlers.saml2.RolesGenerationHandler" />
  </Handlers>
</PicketLink>
```

警告

ハンドラーは、Chain of Responsibility を使用して実装されます。各ハンドラーは、**picketlink.xml** で定義された順序で要求および応答でロジックを実行します。ハンドラーが設定される順序に注意することが非常に重要です。

デフォルトでは、**picketlink.xml** は IDP web アプリケーションの **WEB-INF** ディレクトリーにあります。ただし、アプリケーション外の **picketlink.xml** へのカスタムパスを設定できます。これは、複数の JBoss EAP インスタンス間で複数のアプリケーションが同じ **picketlink.xml** 設定を共有している場合に便利です。

6. **オプション: picketlink.xml** のカスタムの場所の設定
CONFIG_FILE パラメーターを使用して **picketlink.xml** のカスタムの場所を指定できます。これは、**web.xml** に **<context-param>** 要素を追加して行います。

CONFIG_FILE パラメーターの使用

```
<context-param>
  <param-name>CONFIG_FILE</param-name>
  <param-value>/path/to/picketlink.xml</param-value>
</context-param>
```

org.picketlink.federation.saml.CONFIG_PROVIDER パラメーターを使用してカスタム設定プロバイダーを指定することもできます。これにより、**org.picketlink.identity.federation.web.util.SAMLConfigurationProvider** を拡張するカスタム実装を作成して、独自の設定ロジックを提供することができます。

org.picketlink.federation.saml.CONFIG_PROVIDER パラメーターの使用

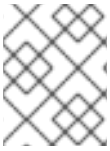
```
<context-param>
  <param-name>org.picketlink.federation.saml.CONFIG_PROVIDER</param-name>
  <param-value>MyConfigurationProvider</param-value>
</context-param>
```

注記

ここで使用する管理 CLI コマンドは、JBoss EAP スタンドアロンサーバーを実行していることを仮定しています。JBoss EAP 管理対象ドメインの管理 CLI を使用する場合の詳細は JBoss EAP Management CLI Guide を参照してください。

2.2.2. SP の設定

アプリケーションを SP として動作するように設定するには、以下の手順を実行する必要があります。



注記

アプリケーションを作成およびデプロイする前に、セキュリティードメインを作成および設定する必要があります。

1. SP のセキュリティードメインを設定します。
   IDP はユーザーのクレデンシャルを処理し、SAML v2 セキュリティーアサーションを発行するため、SP はこれらのアサーションを検証するロールを担います。この検証を実行するにはセキュリティードメインが必要になりますが、アイデンティティーストアは必要ありません。この場合、SP のセキュリティードメインは **SAML2LoginModule** を使用する必要があります。

   セキュリティードメイン追加の管理 CLI コマンド

```
/subsystem=security/security-domain=sp:add(cache-type=default)

/subsystem=security/security-domain=sp/authentication=classic:add

/subsystem=security/security-domain=sp/authentication=classic/login-
module=org.picketlink.identity.federation.bindings.jboss.auth.SAML2LoginModule:add(code=org
.picketlink.identity.federation.bindings.jboss.auth.SAML2LoginModule,flag=required)

reload
```

警告

SAML2LoginModule は SAML で PicketLink を使用するアプリケーションでのみ使用を目的としており、PicketLink Service Provider Undertow ServletExtension (**org.picketlink.identity.federation.bindings.wildfly.sp.SPServletExtension**) なしでは使用しないでください。これを行うことで、**SAML2LoginModule** または **SAML2CommonLoginModule** が常に **RELPTY_STR** のデフォルトパスワードを受け入れるため、セキュリティーリスクが発生する可能性があります。たとえば、PicketLink Service Provider Undertow ServletExtension が SP アプリケーションにインストールされていない場合でも発生する可能性があります。PicketLink Service Provider Undertow ServletExtension は、JBoss EAP の SP アプリケーションを設定する 際に自動的にインストールされます。**SAML2LoginModule** が他のログインモジュールとスタックされている場合にも、これが発生します。

```
<security-domain name="sp" cache-type="default">
  <authentication>
    <login-module
code="org.picketlink.identity.federation.bindings.jboss.auth.SAML2LoginMo
dule" flag="optional">
      <module-option name="password-stacking" value="useFirstPass"/>
    </login-module>
    <login-module code="UsersRoles" flag="required">
      <module-option name="usersProperties" value="users.properties"/>
      <module-option name="rolesProperties" value="roles.properties"/>
      <module-option name="password-stacking" value="useFirstPass"/>
    </login-module>
  </authentication>
</security-domain>
```

SAML2LoginModule は、アサーションに基づいてユーザーのセキュリティーコンテキストを構築するために使用されます。PicketLink SAML Authenticator (PicketLink Service Provider Undertow ServletExtension (**org.picketlink.identity.federation.bindings.wildfly.sp.SPServletExtension**) によってインストールされる PicketLink **SAML Authenticator は SAML2LoginModule** を使用し、SP の **picketlink.xml** で設定される IDP の認証決定を保留できるようにします。

オーセンティケーターは、セキュリティーアサーション (この場合は SAML アサーション) に基づいてプリンシパルの認証を行います.この場合、IDP によって発行されます。アプリケーションに追加された各リクエストをインターセプトし、SAML アサーションがリクエストに存在するかどうかを確認し、アサーションを検証し、プリンシパルの SAML 固有の検証を実行し、要求されたアプリケーションでプリンシパルのセキュリティーコンテキストを作成します。

オーセンティケーターの一部は、プリンシパルの認証および承認に使用するセキュリティードメインを定義することで **jboss-web.xml** ファイルに設定されます。また、 **web.xml で <login-config>** が指定され、必要な依存関係 が宣言されていることを確認する必要があります.これは後のステップで実行します。

SP の **jboss-web.xml** ファイルには以下が必要です。

- 認証または承認に使用されるセキュリティードメインを指定する **<security-domain>**。

例: jboss-web.xml ファイル

```
<jboss-web>
  <security-domain>sp</security-domain>
  <context-root>sales-post</context-root>
</jboss-web>
```

2. SP の **web.xml** ファイルを設定します。
SP の **web.xml** ファイルには以下が含まれている必要があります。

- セキュアな領域の URL パターンにマップする **<url-pattern>** が含まれる **<web-resource-collection>** を持つ **<security-constraint>**。任意で、**<security-constraint>** に許可されるロールを明記する **<auth-constraint>** を含めることもできます。

- **<auth-constraint>** にロールが指定されている場合、これらのロールを **<security-role>** で定義する必要があります。

- **FORM** 認証を指定する **<auth-method>** が含まれる **<login-config>**。これは SAML オーセンティケーターに必要です。

例: web.xml ファイル

```
<web-app>
  <display-name>SP</display-name>
  <description>SP</description>
  <!-- Define a Security Constraint on this Application -->
  <security-constraint>
    <web-resource-collection>
      <web-resource-name>SP</web-resource-name>
      <url-pattern>/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
      <role-name>All</role-name>
    </auth-constraint>
  </security-constraint>
  <!-- Security roles referenced by this web application -->
  <security-role>
    <description> The role that is required to log in to the SP Application </description>
    <role-name>All</role-name>
  </security-role>
  <!-- Define the Login Configuration for this Application -->
  <login-config>
    <auth-method>FORM</auth-method>
  </login-config>
</web-app>
```

注記

アプリケーションのウェルカムページを定義することが推奨されます。デフォルトでは、JBoss EAP は **index.jsp** という名前のファイルを検索しますが、**web.xml** の **<ogg-file-list>** を使用してこれを設定できます。

ログアウトプロセスでは、正常にログアウトされた時にプリンシパルを **logout.jsp** にリダイレクトしようとします。このファイルがアプリケーションの root ディレクトリーに定義されていることを確認してください。

3. SP に必要な依存関係を宣言します。
SP として機能する web アプリケーションは、**org.picketlink** クラスを見つけることができるように依存関係を **jboss-deployment-structure.xml** に定義する必要があります。JBoss EAP は必要なすべての **org.picketlink** と関連クラスを提供し、アプリケーションはそれらを依存関係として宣言して使用することのみが必要になります。

jboss-deployment-structure.xml を使用した依存関係の宣言

```
<jboss-deployment-structure>
  <deployment>
    <dependencies>
      <module name="org.picketlink" services="import"/>
    </dependencies>
  </deployment>
</jboss-deployment-structure>
```

注記

これまでのバージョンの JBoss EAP では、同じ依存関係を宣言していましたが、SAML 認証システムをインストールするためのバルブが宣言されていました。JBoss EAP 7 で Undertow が導入されたため、PicketLink SAML オーセンティケーターを使用してデプロイメントを手動で設定する必要はありません。**org.picketlink** 依存関係で use **services="import"** を定義すると、これらのサービスは自動的に設定されるようになりました。デプロイメントに対して SAML を有効にするには、この設定を宣言する必要があります。

4. SP 用の picketlink.xml ファイルを作成および設定します。
picketlink.xml ファイルは Authenticator の動作を管理し、アプリケーションの起動時にロードされます。

ファイルには、少なくとも以下の要素が含まれている必要があります。

- **<PicketLinkSP>** IDP (<IdentityURL>**)** の URL と SP の URL (<ServiceURL> **)** を定義します。
- SAML リクエストおよび応答の処理に必要なハンドラーのセットを定義する **<Handlers>**。

picketlink.xml ファイルの例

```
<PicketLink xmlns="urn:picketlink:identity-federation:config:2.1">
  <PicketLinkSP xmlns="urn:picketlink:identity-federation:config:2.1" BindingType="POST">
    <IdentityURL>${idp.url::http://localhost:8080/identity/}</IdentityURL>
    <ServiceURL>${sales-post.url::http://localhost:8080/sales-post/}</ServiceURL>
  </PicketLinkSP>
```

```
  <Handlers xmlns="urn:picketlink:identity-federation:handler:config:2.1">
    <Handler
class="org.picketlink.identity.federation.web.handlers.saml2.SAML2LogOutHandler" />
    <Handler
class="org.picketlink.identity.federation.web.handlers.saml2.SAML2AuthenticationHandler" />
    <Handler
class="org.picketlink.identity.federation.web.handlers.saml2.RolesGenerationHandler" />
  </Handlers>
</PicketLink>
```

注記

推奨されませんが、**BindingType="POST"** を **BindingType="REDIRECT"** に変更して SP が HTTP/REDIRECT を使用するように設定することも可能です。

警告

ハンドラーは、Chain of Responsibility を使用して実装されます。各ハンドラーは、**picketlink.xml** で定義された順序で要求および応答でロジックを実行します。ハンドラーが設定される順序に注意することが非常に重要です。

デフォルトでは、**picketlink.xml** は SP Web アプリケーションの **WEB-INF** ディレクトリーにありますが、アプリケーション外の **picketlink.xml** へのカスタムパスを設定できます。これは、複数の JBoss EAP インスタンス間で複数のアプリケーションが同じ **picketlink.xml** 設定を共有している場合に便利です。

5. **オプション: picketlink.xml** にカスタムの場所を設定します。
CONFIG_FILE パラメーターを使用して **picketlink.xml** のカスタムの場所を指定できます。これは、**web.xml** に **<context-param>** 要素を追加して行います。

CONFIG_FILE パラメーターの使用

```
<context-param>
  <param-name>CONFIG_FILE</param-name>
  <param-value>/path/to/picketlink.xml</param-value>
</context-param>
```

org.picketlink.federation.saml.CONFIG_PROVIDER パラメーターを使用してカスタム設定プロバイダーを指定することもできます。これにより、**org.picketlink.identity.federation.web.util.SAMLConfigurationProvider** を拡張するカスタム実装を作成して、独自の設定ロジックを提供することができます。

org.picketlink.federation.saml.CONFIG_PROVIDER パラメーターの使用

```
<context-param>
  <param-name>org.picketlink.federation.saml.CONFIG_PROVIDER</param-name>
  <param-value>MyConfigurationProvider</param-value>
</context-param>
```

注記

ここで使用する管理 CLI コマンドは、JBoss EAP スタンドアロンサーバーを実行していることを仮定しています。JBoss EAP 管理対象ドメインの管理 CLI を使用する場合の詳細は JBoss EAP **Management CLI Guide** を参照してください。

2.2.3. SP 開始フローの使用

SP-Inited Flow は、ブラウザーベースのシングルサインオンを記述する際に発生する一般的なユースケースで、JBoss EAP **Security Architecture** の Browser-Based SSO Using SAML および Multiple Red Hat JBoss Enterprise Application Platform Instances and Multiple Applications Using Browser-Based SSO with SAML で説明されています。つまり、プリンシパルは SP 内のセキュアなリソースにアクセスしようとします。SP は、プリンシパルのセキュリティーアサーションをチェックし、認証されていないプリンシパルを IDP にリダイレクトしてフローを開始します。IDP で認証に成功すると、プリンシパルは、リクエストされた元のリソースの検証/ 拒否アクセスを検証および許可するためのセキュリティーアサーションを使用して、初期 SP にリダイレクトされます。

手順

1. プリンシパルは SP 上のセキュアなリソースへのアクセスを試行します。
2. SP はプリンシパルでチェックを実行します。プリンシパルが認証されていない場合は、IDP にリダイレクトする必要があります。すでに認証されている場合は、その他のステップはすべてスキップされ、最後のステップが実行されます。
3. SP は IDP を見つけ、プリンシパルのブラウザーを使用して IDP に対する認証要求を発行します。
4. IDP は、設定されたアイデンティティーストアを使用してプリンシパルをログインページなどで認証しようとします。
5. プリンシパルが IDP で特定された後 (ログイン成功後など)、IDP はプリンシパルのセキュリティー関連情報が含まれる SAML v2 アサーションを、プリンシパルのブラウザーを使用して SP に発行します。
6. SP はプリンシパルのセキュリティーアサーションに対してチェックを実行し、これらのアサーションに含まれる情報に基づいて、SP は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。

このフローでは、追加のステップや設定は必要ありません。すべてのリダイレクトは設定済みの SP および IDP によって処理され、セキュアなリソースとセキュアでないリソースの両方へのリンクに必要な追加の変更は必要ありません。

2.2.4. IDP 関係フローの使用

SAML v2 を使用したブラウザーベースのシングルサインオンのほとんどの例では、前項 で説明したように SP 開始フローを使用します。ただし、SAML v2 は、IDP 開始フローまたは未承認応答フローなどの追加のフローをサポートします。このシナリオでは、SP は認証フローを開始せずに IDP から SAML 応答を受信します。代わりに、フローは IDP 側で開始され、認証されるとプリンシパルはリストから特定の SP を選択し、その URL にリダイレクトできます。

2.2.4.1. 手順

1. プリンシパルは IDP にアクセスします。

2. IDP は SAML 要求や応答がないことを確認し、SAML を使用する IDP-first シナリオを想定します。

3. IDP はプリンシパルの認証を行います。

4. 認証後、IDP は、すべての SP アプリケーションにリンクするページで、プリンシパルが表示されている、ホスト済みセクションを表示します。

5. プリンシパルは SP アプリケーションを選択します。

6. IDP は、クエリーパラメーターに SAML アサーションを持つ SP へユーザーをリダイレクトします。**POST** バインディングが使用される場合、IDP は HTTP **POST** を使用して SAML アサーションをサービスプロバイダーに送信します。

7. SP は SAML アサーションをチェックし、アクセスを提供します。

2.2.4.2. Hosted Section

hosted section は、IDP 開始フローの認証に成功した後、またはすでに認証済みのプリンシパルが IDP のルートに直接アクセスしようとする場合に、ユーザーを指示する場所です。デフォルトでは、ホストされるセクションは **/hosted/** にありますが、**HostedURI** 属性を **<PicketLinkIDP>** 要素に追加することで **picketlink.xml** ファイルで変更できます。

```
<PicketLink xmlns="urn:picketlink:identity-federation:config:2.1">
  <PicketLinkIDP xmlns="urn:picketlink:identity-federation:config:2.1" HostedURI="/hosted/">
    ...
  </PicketLinkIDP>
</Picketlink>
```

2.2.4.3. SP へのリンク

ユーザーが認証されると、IDP はすべてのサービスプロバイダーアプリケーションへのリンクが含まれるページを表示します。通常、リンクは以下のようになります。

```
<a href="http://localhost:8080/identity?SAML_VERSION=2.0&TARGET=http://localhost:8080/sales-post/">Sales</a>
```

上記のリンクは、ユーザーが IDP にリダイレクトされることに注意してください. **IDP** は、値がターゲット SP アプリケーションの URL になります。ユーザーが上記のリンクをクリックすると、IDP はリクエストから **TARGET** パラメーターを抽出し、SAML v2 応答を構築し、そのユーザーをターゲット URL にリダイレクトします。ユーザーが SP に到達すると、自動的に認証されます。**SAML_VERSION** クエリーパラメーターは、SAML 応答を作成するために IDP によって使用される必要がある SAML バージョンを指定するために使用されます。

2.2.5. グローバルログアウトプロファイルの設定

あるサービスプロバイダーで開始されるグローバルログアウトプロファイルで、アイデンティティープロバイダー (IDP) およびすべてのサービスプロバイダーからユーザーをログアウトします。

注記

グローバルログアウトプロファイルが適切に機能するには、最大 ▌つの SP が IDP ごとに設定されていることを確認してください。

1. **picketlink.xml** を設定します。
   picketlink.xml に **SAML2** LogOutHandler を追加します。

2. **logout.jsp** ページを作成します。
   ユーザーは、ログアウトプロセスの一部として、サービスプロバイダーアプリケーションの root ディレクトリーにある **logout.jsp** ページにリダイレクトされます。このページが作成されていることを確認します。

   logout.jsp の例

```
<html>
  <head></head>
  <body>
    <p>You have successfully logged out.</p>
  </body>
</html>
```

3. SP の Global Logout Profile リンクを設定します。
   SP リソースへのリンクの URL パラメーターに glo **GLO=true** を使用して、グローバルログアウトプロファイルプロセスを開始します。

   ログアウトリンクの例

```
<a href="?GLO=true">Click to log out</a>
```

2.2.6. ローカルログアウトの使用

グローバルログアウトプロファイルの他に、ローカルログアウトを使用することもできます。グローバルログアウトプロファイルとは対照的に、ローカルログアウトはプリンシパルを 1 つの SP からログアウトし、セッションはそのまま IDP およびその他の SP に残します。基本的に、ローカルログアウトにより、プリンシパルを単一の SP でローカルにログアウトできます。

ローカルログアウトを使用するプロセスは基本的に Global Logout Profile と同じですが、logout リンクの URL パラメーターは **LLO=true** の形式を取ります。

ログアウトリンクの例

```
<a href="?LLO=true">Click to log out</a>
```

プリンシパルが SP のローカルログアウトリンクをクリックすると、SP はセッションを無効にし、プリンシパルが設定されたログアウトページに転送します。

警告

プリンシパルが単一のサービスプロバイダーのみから切断されている場合、プリンシパルには IDP と、セキュアなリソースに引き続きアクセスできる他の SP のアクティブなセッションがあります。特定の状況ではこの動作が必要になる場合がありますが、ほとんどの場合でグローバルログアウトを使用することが強く推奨されます。

2.3. フェデレーションサブシステムでの IDP および SP の設定

IDP および SP を手動で設定する他に、JBoss EAP サブシステムを使用して SAML v2 を使用するシングルサインオンを設定することもできます。この方法は **Domain Model** と呼ばれ、すべての設定を個別のアプリケーションではなく JBoss EAP インスタンスに一元化できるようにします。これにより、管理コンソールや管理 CLI などの JBoss EAP 管理インターフェイスを使用してシングルサインオン設定の作成および更新が可能になります。

フェデレーション

JBoss EAP サブシステムを使用して IDP と SP を設定およびデプロイする場合、それらは **Federation** にまとめられます。Federation は **Circle of Trust** として理解することができます。Circle of Trust には、証明書や SAML 固有の設定を含む共通の設定を共有するアプリケーションが含まれます。また、ユーザーの特定に使用されるプロセス、使用される認証システムの種類、および生成される認証資格情報に関連するポリシーを正確に記述するために相互を信頼するドメインも含まれます。各フェデレーションには、IDP と多くの SP があります。また、フェデレーションは SP と IDP 間の信頼関係を定義し、各 SP がその情報を個別に追跡し、維持する必要性を取り除きます。

2.3.1. サブシステムの設定

サブシステムを使用してフェデレーションを設定する前に、JBoss EAP で有効化し、設定する必要があります。サブシステムを有効にして設定するには、以下の手順が必要です。

注記

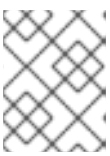
これらの手順を実行する前に、JBoss EAP インスタンスをシャットダウンすることが推奨されます。

1. エクステンションを更新します。
   JBoss EAP 設定ファイルでは、スタンドアロンインスタンスの **standalone.xml** またはドメインの **domain.xml** に **org.wildfly.extension.picketlink** 拡張を追加します。

```
<extensions>
  ...
  <extension module="org.wildfly.extension.picketlink"/>
  ...
</extensions>
```

2. サブシステムを追加します。
   JBoss EAP 設定ファイルでは、スタンドアロンインスタンスの **standalone.xml** またはドメインの **domain.xml** に **picketlink-federation** サブシステムを追加します。

```
<profile>
  ...
  <subsystem xmlns="urn:jboss:domain:picketlink-federation:2.0"/>
   ...
</profile>
```



注記

設定の例は **EAP_HOME/docs/examples/configs/standalone-picketlink.xml** で確認できます。

2.3.2. すこしのセットアップ

サブシステムのセットアップと設定が完了したら、管理インターフェイスを使用してフェデレーションを設定できます。サブシステムを使用してフェデレーションを設定する前に、IDP アプリケーションと SP アプリケーションの両方を準備する必要があります。

2.3.2.1. SP および IDP アプリケーションの準備

前のセクション で説明したように、SAMLv2 を使用してシングルサインオンを手動で設定する場合は、以下のファイルを作成または更新する必要があります。

- **web.xml**
- **jboss-web.xml**
- **picketlink.xml**
- **jboss-deployment-structure.xml**

サブシステムを使用して SAML v2 を使用してシングルサインオンのフェデレーションを設定すると、設定の大半は管理インターフェイスによって発生しますが、これらのファイルは更新されません。アプリケーションで実行する必要がある唯一の設定は、IDPs および SPs の **web.xml** ファイルで **<security-constraint>** および関連づけられた **<security-role>** を設定することです。さらに、**web.xml** の **<login-config>** と、ログインおよびエラーページも IDP に存在する必要があります。

前のセクション で説明したように IDP または SP がすでに設定されている場合は、以下のファイルを削除する必要があります。

- **jboss-web.xml**
- **picketlink.xml**
- **jboss-deployment-structure.xml**

アプリケーションの準備が完了したら、JBoss EAP インスタンスにデプロイする必要があります。

2.3.2.2. 管理 CLI を使用した子の作成

以下のコマンドは、**new-federation** というフェデレーションサンプルを以下の情報とともに追加する方法を示しています。

- アイデンティティープロバイダー **IDP.war** は **http://localhost:8080/identity/** で JBoss EAP インスタンスにデプロイされます。
- サービスプロバイダー **SP.war** は **http://localhost:8080/sales-post/** JBoss EAP インスタンスにデプロイされます。
- アイデンティティープロバイダーとサービスプロバイダーに対して **idp** と **sp** をそれぞれ適切に設定

管理 CLI を使用してフェデレーションを設定するには、以下を行う必要があります。

1. 新しいフェデレーションを追加します。

```
/subsystem=picketlink-federation/federation=new-federation:add
```

2. フェデレーションにアイデンティティープロバイダーを追加します。

```
/subsystem=picketlink-federation/federation=new-federation/identity-provider=IDP.war:add(url="http://localhost:8080/identity/",security-domain=idp)
```

3. サービスプロバイダーをフェデレーションに追加します。

```
/subsystem=picketlink-federation/federation=new-federation/service-provider=SP.war:add(url="http://localhost:8080/sales-post/",security-domain=sp)
```

4. 信頼ドメインをフェデレーションに追加します。

```
/subsystem=picketlink-federation/federation=new-federation/identity-provider=IDP.war/trust-domain="localhost:8080":add
```

2.3.2.3. フェデレーションサブシステムの属性のリファレンス

picketlink-federation サブシステムの構造は次のとおりです。

フェデレーション

- saml
- key-store
  - キー
    - key
- identity-provider
  - trust
    - trust-domain
  - role-generator
  - attribute-manager
  - handlers
    - handler
      - handler-parameter
- service-providers
  - service-provider
    - handlers
      - handler
        - handler-parameter

フェデレーションサブシステムの例

```
<subsystem xmlns="urn:jboss:domain:picketlink-federation:2.0">
  <federation name="federation-redirect-with-signatures">
    <key-store file="/jbid_test_keystore.jks" password="store123" sign-key-alias="servercert" sign-key-password="test123">
      <keys>
        <key name="servercert" host="${jboss.bind.address:localhost},127.0.0.1"/>
      </keys>
    </key-store>
    <identity-provider name="idp-redirect-sig.war" url="http://${jboss.bind.address:127.0.0.1}:8080/idp-redirect-sig/" security-domain="idp" support-signatures="true" strict-post-binding="false">
      <trust>
        <trust-domain name="${jboss.bind.address:127.0.0.1}"/>
      </trust>
      <handlers>
        <handler class-name="com.mycompany.CustomHandler">
          <handler-parameter name="param1" value="paramValue1"/>
          <handler-parameter name="param2" value="paramValue2"/>
          <handler-parameter name="param3" value="paramValue3"/>
        </handler>
      </handlers>
    </identity-provider>
    <service-providers>
      <service-provider name="sp-redirect-sig1.war" security-domain="sp" url="http://${jboss.bind.address:127.0.0.1}:8080/sp-redirect-sig1/" post-binding="false" support-signatures="true">
        <handlers>
          <handler class-name="com.mycompany.CustomHandler">
            <handler-parameter name="param1" value="paramValue1"/>
            <handler-parameter name="param2" value="paramValue2"/>
            <handler-parameter name="param3" value="paramValue3"/>
          </handler>
        </handlers>
      </service-provider>
      <service-provider name="sp-redirect-sig2.war" security-domain="sp" url="http://${jboss.bind.address:127.0.0.1}:8080/sp-redirect-sig2/" post-binding="false" support-signatures="true"/>
    </service-providers>
  </federation>
</subsystem>
```

表2.1 フェデレーション

Attribute	デフォルト	説明
name		フェデレーション名。

saml

SAML タイプを定義します。このタイプは、SAML アサーションの処理および作成方法に関するすべての設定を定義します。

Attribute	デフォルト	説明
clock-skew	0	SAML アサーションのクロックスキューを定義します。値はミリ秒単位で指定する必要があります。
token-timeout	5000	SAML アサーションのタイムアウトを定義します。値はミリ秒単位で指定する必要があります。

key-store

KeyStore タイプを定義します。このタイプはキーストアの設定方法を定義します。

Attribute	デフォルト	説明
password		キーストアのパスワードを定義します。
sign-key-alias		ドキュメントの署名時に使用されるエイリアスを定義します。
sign-key-password		Sign-key-alias のパスワードを定義します。
file		ファイルの場所を定義します。
relative-to		**jboss.home.dir**、**user.home**、**user.dir** などのシステムが提供する名前付きパスのいずれか。絶対パスが、file 属性で指定されたパスに対して計算されます。

キー

鍵の設定。

key

鍵を定義します。

Attribute	デフォルト	説明
name		指定のキーストアのキーの名前またはエイリアスを定義します。
host		指定されたキーで検証されるホスト名を表す単一またはコンマ区切りの文字列リスト。

identity-provider

Identity Provider のタイプを定義します。

Attribute	デフォルト	説明
name		アイデンティティープロバイダーの一意の名前。名前はデプロイメントユニット名である必要があります。たとえば、**idp.war** です。
url		このアイデンティティープロバイダーの URL。
support-signatures	false	署名がサポートされているかどうかを示します。
encrypt	false	暗号化がサポートされているかどうかを示します。
security-domain		ユーザーの認証および承認に使用される security-domain の名前。IDP が外部ではない場合、この属性は必須です。詳細は、**external** 属性を参照してください。
strict-post-binding	true	IDP が常に HTTP **POST** バインディングを使用して応答すべきかどうかを示します。
EXTERNAL	false	設定が外部 IDP への参照であることを示します。
support-metadata	false	SAML メタデータサポートの有効化/無効化。
ssl-authentication	false	アイデンティティープロバイダーが HTTP **CLIENT_CERT** 認証もサポートすべきかどうかを示します。

trust

信頼されるドメインタイプをグループします。

trust-domain

信頼されるドメインタイプを定義します。

Attribute	デフォルト	説明
name		ドメイン名を定義します。
cert-alias		このドメインの証明書エイリアスを定義します。

role-generator

ロールをロードし、それらを SAML アサーションにプッシュするために使用される **RoleGenerator** 実装。

Attribute	デフォルト	説明
name		ロールジェネレーター名を定義します。
class-name		**RoleGenerator** タイプの完全修飾名。
code		組み込み型にマップするエイリアスを定義します。
module		**class-name** の読み込み時に使用されるモジュールを定義します。

attribute-manager

ロールをロードし、SAML アサーションにプッシュするために使用される **AttributeManager** 実装。

Attribute	デフォルト	説明
name		属性マネージャー名を定義します。
class-name		**AttributeManager** タイプの完全修飾名。
code		組み込み型にマップするエイリアスを定義します。
module		**class-name** の読み込み時に使用されるモジュールを定義します。

handlers

グループハンドラータイプ。

handler

ハンドラータイプを定義します。

Attribute	デフォルト	説明
name		ハンドラー名を定義します。
class-name		ハンドラークラス名を定義します。
code		組み込み型にマップするエイリアスを定義します。

handler-parameter

ハンドラーパラメータータイプを定義します。

Attribute	デフォルト	説明
name		パラメーター名を定義します。
value		パラメーターの値を定義します。

service-providers

サービスプロバイダーのタイプをグループ化します。

service-provider

サービスプロバイダーのタイプを定義します。

Attribute	デフォルト	説明
name		このインスタンスの名前。この名前はデプロイメントユニット名である必要があります。
url		このサービスプロバイダーの URL。
post-binding	true	使用する SAML Binding を示します。True の場合、HTTP **POST** バインディングが使用されます。それ以外の場合は、HTTP REDIRECT バインディングが使用されます。
strict-post-binding	true	使用する SAML Binding を示します。True の場合、HTTP **POST** バインディングが使用されます。それ以外の場合は、HTTP REDIRECT バインディングが使用されます。
support-signatures	false	署名がサポートされているかどうかを示します。
support-metadata	false	SAML メタデータサポートの有効化/無効化。
security-domain		ユーザーの認証に使用されるセキュリティードメイン名。
error-page	**/error.jsp**	カスタムエラーページの場所を定義します。
logout-page	**/logout.jsp**	カスタムのログアウトページの場所を定義します。

2.4. IDP のアイデンティティーストアの設定

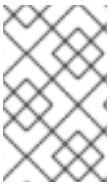
IDP はセキュリティードメインを使用するため、IDP の機能は、これをサポートする実際の ID ストアから独立しています。これにより、管理者は IDP のセキュリティードメインを設定する際に多くのオプションを利用できます。セキュリティードメインおよびログインモジュールの詳細は、JBoss EAP **セ**

キュリティーアーキテクチャーの Security Subsystem を参照してください。セキュリティードメインのログインモジュールを設定するのと同様に、異なる ID ストアがさまざまな機能とパフォーマンスのトレードオフを提供することに注意してください。

アイデンティティーストアを使用するセキュリティードメインを設定するには、以下の手順が必要です。

注記

本書の目的上、データベースログインモジュールおよび LDAP ログインモジュールは例として示されていますが、他のアイデンティティーストアやログインモジュールは IDP と使用するように設定することもできます。



注記

ここで使用する管理 CLI コマンドは、JBoss EAP スタンドアロンサーバーを実行していることを仮定しています。JBoss EAP 管理対象ドメインの管理 CLI を使用する場合の詳細は JBoss EAP **Management CLI Guide** を参照してください。

1. アイデンティティーストアを設定します。
   セキュリティードメインおよびログインモジュールを設定して、アイデンティティープロバイダー、アイデンティティープロバイダー、およびそのアイデンティティープロバイダーへの接続を使用するには、設定が必要です。

   a. Database Login Module のアイデンティティーストアを設定します。
      データベースベースのアイデンティティーストアに必要な最初の項目は、ログインモジュールが使用するデータベースです。

      以下のデータポイントが必要です。

      - ユーザー名
      - パスワード
      - ロール
      - ロールグループ

      Database Login モジュールでは、ユーザー名をパスワードにマップするクエリーと、ユーザー名をロールおよびロールグループにマップするクエリーを作成する必要があります。この情報はさまざまな方法でデータベースに保存できますが、テーブルを含むデータベースの作成については本ガイドの範囲外となります。この例では、以下の表が作成されていることを前提としています。

      表2.2 sso-users

username	passwd
Sarah	Testing123!

      表2.3 SSO-roles

username	role
role-group	Sarah
sample	SSO-Users

データソースの作成については、本書では扱いません。データソースの設定に関する詳細は、JBoss EAP **設定ガイド**の データソース管理 を参照してください。

この例では、**idpDS** という名前のデータソースが作成され、適切に設定され 、JBoss EAP インスタンスにデプロイされていることを前提とします。このデータソースは、**sso-users** および **sso-roles** テーブルを格納するデータベースに接続します。

b. LDAP ログインモジュールのアイデンティティーストアを設定します。
LDAP ログインモジュールの設定前に、適切に設定された LDAP サーバーが必要です。Database ログインモジュールとは異なり、LDAP ログインモジュールの設定にデータソースは必要ありません。LDAP の基本および JBoss EAP セキュリティーとは、JBoss EAP **セキュリティーアーキテクチャー** ガイドで説明されています。

   i. LDAP サーバーを設定します。
   LDAP サーバーの設定は本ガイドの範囲外です。この例では、**http://ldaphost.example.com:1389/** で LDAP サーバーにアクセスできます。

   ii. ディレクトリー情報の例
   LDAP サーバーのディレクトリー構造と編成は、ユースケースや組織のニーズによって大幅に異なる場合があります。この例では、以下のエントリーが LDIF 形式で作成されています。

```
dn: dc=example,dc=com
objectclass: top
objectclass: dcObject
objectclass: organization
dc: example
o:  Example
#============================
dn: ou=People,dc=example,dc=com
objectclass: top
objectclass: organizationalUnit
ou: People
#============================
dn: uid=jsmith,ou=People,dc=example,dc=com
objectclass: top
objectclass: uidObject
objectclass: person
uid: jsmith
cn: John
sn: Smith
userPassword: theduke
#============================
dn: ou=Roles,dc=example,dc=com
objectclass: top
objectclass: organizationalUnit
ou: Roles
#============================
```

```
dn: cn=Sample,ou=Roles,dc=example,dc=com
objectclass: top
objectclass: groupOfNames
cn: Sample
member: uid=jsmith,ou=People,dc=example,dc=com
description: the Sample group
```

2. セキュリティードメインを追加します。
アイデンティティーストア自体が設定され、JBoss EAP インスタンスとアイデンティティーストア間の接続が設定されたら、セキュリティードメインを作成および設定できます。以下のコマンドは、空のセキュリティードメインを作成する方法を示しています。**MY-DOMAIN** を、使用するセキュリティードメインの名前に置き換えます。

セキュリティードメイン追加の管理 CLI コマンド

```
/subsystem=security/security-domain=MY-DOMAIN:add(cache-type=default)
```

3. 認証セクションとログインモジュールをセキュリティードメインに追加します。空のセキュリティードメインを作成したら、ログインモジュールを追加して authentication セクションを作成する必要があります。以下のコマンドは、既存のセキュリティードメインに空の認証セクションを追加する方法を示します。

認証セクションをセキュリティードメインに追加する管理 CLI コマンド

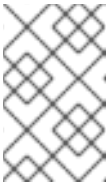
```
/subsystem=security/security-domain=MY-DOMAIN/authentication=classic:add
```

空の認証セクションが作成されると、ログインモジュールを追加して、必要なアイデンティティーストアを使用するように設定できます。ログインモジュールをセキュリティードメインに追加したら、通常、設定の再読み込みが必要になります。

以下は、ログインモジュールを追加し、設定を再読み込みするための一般的なコマンド構造です。**MY-DOMAIN**、**MY-LOGIN-MODULE**、および **MY-CONFIGURATION** を適切な値に置き換えます。

```
/subsystem=security/security-domain=MY-DOMAIN/authentication=classic/login-module=MY-LOGIN-MODULE:add(MY-CONFIGURATION)

reload
```

- Database ログインモジュール



注記

この例では、**idpDS** という名前のデータソースが 手順 1 で作成されており、**idp-db-domain** という名前のセキュリティードメインが 手順 2 で作成されていることを前提としています。

データベースログインモジュールを使用するための認証セクションを設定するための管理 CLI コマンド

```
/subsystem=security/security-domain=idp-db-domain/authentication=classic/login-module=Database:add(code=Database,flag=required,module-options=[("dsJndiName"=>"java:/idpDS"),("principalsQuery"=>"select passwd from 'sso-users'
```

```
where username=?"),("rolesQuery"=>"select role, role-group from 'sso-roles' where
username=?")])

reload
```

- LDAP ログインモジュールを追加します。
  LdapExtended ログインモジュールの設定に必要な手順は、How to Configure Identity Management を参照してください。

2.5. SP および IDP を使用した SSL/TLS の設定

SSL/TLS の基本については、JBoss EAP セキュリティーアーキテクチャー を参照してください。ブラウザーベースのシングルサインオン環境に SSL/TLS サポートを追加すると、シングルサインオン以外の環境に追加されることはあまり変わりません。IDP と SP の両方に HTTPS コネクターを追加して、トラフィックのセキュリティーを保護することができます。

まだ作成されていない場合は、SSL/TLS サーバーアイデンティティーのセキュリティーレルムを作成する必要があります。SSL/TLS 証明書でも設定する必要があります。

2.5.1. レガシーセキュリティーレルムを使用したアプリケーション用の一方向 SSL/TLS の有効化

この例では、キーストアの **identity.jks** がサーバー設定ディレクトリーにコピーされ、指定のプロパティーで設定されたことを仮定します。管理者は、example の独自の値を置き換えてください。

注記

ここで使用する管理 CLI コマンドは、JBoss EAP スタンドアロンサーバーを実行していることを仮定しています。JBoss EAP 管理対象ドメインの管理 CLI を使用する場合の詳細は 管理 CLI ガイド を参照してください。

1. 最初に HTTPS セキュリティーレルムを追加し、設定します。HTTPS セキュリティーレルムが設定されたら、セキュリティーレルムを参照する **undertow** サブシステムで **https-listener** を設定します。

```
batch

/core-service=management/security-realm=HTTPSRealm:add

/core-service=management/security-realm=HTTPSRealm/server-identity=ssl:add(keystore-
path=identity.jks, keystore-relative-to=jboss.server.config.dir, keystore-password=password1,
alias=appserver)

/subsystem=undertow/server=default-server/https-listener=https:write-
attribute(name=security-realm, value=HTTPSRealm)

run-batch
```

警告

Red Hat では、影響するすべてのパッケージで TLSv1.1 または TLSv1.2 を利用するために SSLv2、SSLv3、および TLSv1.0 を明示的に無効化することを推奨しています。

2. JBoss EAP サーバーを再起動し、変更を反映します。

2.6. 証明書ベースの認証を使用するようにアイデンティティープロバイダーを設定する

SSL/TLS を使用するように SP および IDP を設定する他に、証明書ベースの認証を使用するように IDP を設定することもできます。証明書ベースの認証を使用するように IDP を設定する前に、SSL/TLS を使用するように IDP と SP を設定する必要があります。

1. クライアント証明書とトラストストアを作成します。
   クライアントが認証に使用する証明書およびトラストストアを作成する必要があります。これらは、サーバー設定およびクライアントのブラウザーで使用する必要があります。

   クライアント証明書およびトラストストアの例

   ```
   $ keytool -genkeypair -alias client -storetype jks -keyalg RSA -keysize 2048 -keypass
   change_it -keystore client.jks -storepass change_it -dname
   "CN=client,OU=Sales,O=Systems Inc,L=Raleigh,ST=NC,C=US" -validity 730 -v

   $ keytool -export -alias client -keystore client.jks -storepass change_it -file client.cer

   $ keytool -import -file client.cer -alias client -keystore client.truststore
   ```

2. IDP のセキュリティードメインの作成
   認証に IDP が使用する証明書ベースのログインモジュールを使用するセキュリティードメインを作成する必要があります。証明書ベースのログインモジュールの詳細は、Login Module Reference を参照してください。

   CertificateRoles ログインモジュールのあるセキュリティードメインの例

   ```
   /subsystem=security/security-domain=idp-cert:add

   /subsystem=security/security-domain=idp-cert/authentication=classic:add

   /subsystem=security/security-domain=idp-cert/authentication=classic/login-
   module=CertificateRoles:add(code=CertificateRoles,flag=optional,module-options=
   [("password-stacking"=>"useFirstPass"),("securityDomain"=>"idp-cert"),
   ("verifier"=>"org.jboss.security.auth.certs.AnyCertVerifier")])

   /subsystem=security/security-domain=idp-cert/jsse=classic:add(truststore=
   {url=>"/path/to/client.jks",password=>change_it})

   reload
   ```

また、このセキュリティードメインを使用するよう IDP を設定する必要もあります。IDP の設定に関する詳細は、Setting up an IDP セクションを参照してください。

注記

また、**RegExUserNameLoginModule** を Certificate ログインモジュールとともに使用して、プリンシパル名からユーザー名、UID、またはその他の情報を抽出することもできます。**RegExUserNameLoginModule** の詳細は、Login Module Reference を参照してください。

3. クライアント証明書をクライアントのブラウザーにインポートします。IDP とサーバー設定が完了したら、クライアント証明書を使用するようにクライアントのブラウザーを設定する必要があります。この設定は、ブラウザーによって異なります。クライアントのブラウザーがクライアント証明書を使用するように設定されていると、クライアントは証明書を使用して IDP で認証できるようになります。

2.7. KERBEROS 認証を使用するようにアイデンティティープロバイダーを設定する

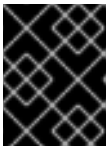
IDP は他の ID ストアの他に、Kerberos を認証メカニズムとして使用することもできます。Kerberos を使用するように IDP を設定するには、以下を行う必要があります。

注記

作業用の Kerberos 環境があることを前提とします。

1. Kerberos 認証のセキュリティードメインを設定します。
以下のコマンドを使用して、IDP で必要なセキュリティードメインを設定できます。詳細は、**Kerberos による SSO のセットアップ方法** ガイドの JBoss EAP レガシーセキュリティーサブシステム設定 セクションを参照してください。

```
/subsystem=security/security-domain=host:add(cache-type=default)

/subsystem=security/security-domain=host/authentication=classic:add

/subsystem=security/security-domain=host/authentication=classic/login-
module=Kerberos:add(code=Kerberos, flag=required, module-options=[debug=false,
storeKey=true, refreshKrb5Config=true, useKeyTab=true, doNotPrompt=true,
keyTab=/home/username/service.keytab,
principal=host/SERVER_NAME@REALM_NAME])

/subsystem=security/security-domain=app-spnego:add(cache-type=default)

/subsystem=security/security-domain=app-spnego/authentication=classic:add

/subsystem=security/security-domain=app-spnego/authentication=classic/login-
module=SPNEGO:add(code=SPNEGO, flag=required, module-options=
[serverSecurityDomain=host])
```



重要

関連するシステムプロパティーがすべて 有効にされていることも確認する必要があります。

ログインモジュールの詳細は、JBoss EAP **Login Module Reference** の Kerberos Login Module および SPNEGO Login Module を参照してください。

2. SP のセキュリティードメインを設定します。
以下のコマンドを使用して、SP が必要とするセキュリティードメインを設定できます。詳細は、SP の設定 を参照してください。

```
/subsystem=security/security-domain=sp:add(cache-type=default)

/subsystem=security/security-domain=sp/authentication=classic:add

/subsystem=security/security-domain=sp/authentication=classic/login-
module=org.picketlink.identity.federation.bindings.jboss.auth.SAML2LoginModule:add(code=org
.picketlink.identity.federation.bindings.jboss.auth.SAML2LoginModule,flag=required)
```

3. 変更を反映するためにサーバーをリロードします。

```
reload
```

4. 上記の手順を完了すると、以下の設定が作成されます。

例: IDP および SP のセキュリティードメイン

```
<security-domain name="host" cache-type="default">
  <authentication>
    <login-module code="Kerberos" flag="required">
      <module-option name="debug" value="false"/>
      <module-option name="storeKey" value="true"/>
      <module-option name="refreshKrb5Config" value="true"/>
      <module-option name="useKeyTab" value="true"/>
      <module-option name="doNotPrompt" value="true"/>
      <module-option name="keyTab" value="/home/username/service.keytab"/>
      <module-option name="principal" value="HTTP/testserver@MY_REALM"/>
    </login-module>
  </authentication>
</security-domain>
<security-domain name="app-spnego" cache-type="default">
    <authentication>
        <login-module code="SPNEGO" flag="required">
            <module-option name="serverSecurityDomain" value="host"/>
        </login-module>
    </authentication>
    <mapping>
    ...
    </mapping>
</security-domain>
<security-domain name="sp" cache-type="default">
    <authentication>
        <login-module
code="org.picketlink.identity.federation.bindings.jboss.auth.SAML2LoginModule"
flag="required"/>
    </authentication>
</security-domain>
```

5. IDP アプリケーションを設定します。

IDP 設定のプロセスは、IDP の設定 セクションで説明されているものと同じですが、以下の変更があります。

- JBoss Negotiation の追加の依存関係の宣言
- **SPNEGO** ログインモジュールでセキュリティードメインを使用するよう IDP アプリケーションを設定します。

注記

IDP の設定時に、設定で **PicketLinkSTS** 要素を指定する必要はありません。OMMITED **PicketLink** の場合は、**picketlink-core-VERSION.jar** 内の **core-sts** という名前のファイルからデフォルト設定を読み込みます。

この設定は、必要な場合にのみ上書きします。たとえば、トークンのタイムアウトを変更したり、SAML アサーションのカスタムセキュリティートークンプロバイダーを指定したりします。

例: Kerberos および picketlink 依存関係を含む jboss-deployment-structure.xml

```
<jboss-deployment-structure>
  <deployment>
    <dependencies>
      <module name="org.picketlink" services="import"/>
      <module name="org.jboss.security.negotiation"/>
    </dependencies>
  </deployment>
</jboss-deployment-structure>
```

例: IDP の jboss-web.xml

```
<jboss-web>
  <security-domain>app-spnego</security-domain>
  <context-root>identity</context-root>
</jboss-web>
```

例: PicketLinkSTS 要素を含む picketlink.xml

```
<PicketLink xmlns="urn:picketlink:identity-federation:config:2.1">
  <PicketLinkIDP xmlns="urn:picketlink:identity-federation:config:2.1">
  <IdentityURL>${idp.url::http://localhost:8080/idp/}</IdentityURL>
    <Trust>
     <Domains>redhat.com,localhost,amazonaws.com</Domains>
    </Trust>
  </PicketLinkIDP>
  <Handlers xmlns="urn:picketlink:identity-federation:handler:config:2.1">
      <Handler
class="org.picketlink.identity.federation.web.handlers.saml2.SAML2IssuerTrustHandler"
/>
      <Handler
class="org.picketlink.identity.federation.web.handlers.saml2.SAML2LogOutHandler" />
      <Handler
class="org.picketlink.identity.federation.web.handlers.saml2.SAML2AuthenticationHandler"
/>
```

```
    <Handler
class="org.picketlink.identity.federation.web.handlers.saml2.RolesGenerationHandler" />
  </Handlers>
  <!-- The configuration bellow defines a token timeout and a clock skew. Both
configurations will be used during the SAML Assertion creation. This configuration is
optional. It is defined only to show you how to set the token timeout and clock skew
configuration.   -->
  <PicketLinkSTS xmlns="urn:picketlink:identity-federation:config:1.0"
TokenTimeout="5000" ClockSkew="0">
    <TokenProviders>
    <TokenProvider

ProviderClass="org.picketlink.identity.federation.core.saml.v1.providers.SAML11AssertionT
okenProvider"
      TokenType="urn:oasis:names:tc:SAML:1.0:assertion"
      TokenElement="Assertion"
TokenElementNS="urn:oasis:names:tc:SAML:1.0:assertion" />
    <TokenProvider

ProviderClass="org.picketlink.identity.federation.core.saml.v2.providers.SAML20AssertionT
okenProvider"
      TokenType="urn:oasis:names:tc:SAML:2.0:assertion"
      TokenElement="Assertion"
TokenElementNS="urn:oasis:names:tc:SAML:2.0:assertion" />
    </TokenProviders>
  </PicketLinkSTS>
</PicketLink>
```

重要

IDP の **web.xml** で設定されたロールが Kerberos 環境に設定されたロールと一致することを確認する必要があります。そのためには、IDP のセキュリティードメインで別のログインモジュールを設定し、適切なロールを SPNEGO 認証の後にマップするか、IDP のセキュリティードメインでマッピングプロバイダーを使用します。

2.8. JBOSS EAP の以前のバージョンからの変更

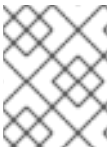
JBoss EAP 7 に Undertow サーバーの追加により、SAML SSO の設定方法に変更が加えられました。JBoss EAP 7 で IDP および SP を設定する場合や、以前のバージョンの JBoss EAP で実行されている IDP および SP を移行する場合は、これらの変更を考慮する必要があります。

- バルブは使用されなくなり、**context-param** を使用してパラメーターが設定されます。
- 依存関係宣言が変更されました。
- SAML オーセンティケーターでは、**web.xml** で **FORM** 認証を設定する必要がありました。
- 動的アカウント選択の設定が変更されました。

2.8.1. バルブおよびバルブ設定の変更

JBoss EAP 7 ではバルブは使用されなくなりましたが、Undertow ハンドラーでは同様の機能を使用できます。結果として、**jboss-web.xml** にバルブ宣言を追加する必要がなくなりました。この変更の詳細および移行への影響については、JBoss EAP **移行ガイド** Migrate Custom Application Valves セクショ

ン を参照してください。



注記

セキュリティードメインの指定などの他の設定は引き続き **jboss-web.xml** で設定されます。

バルブは使用されないようになったため、以下の項目は **web.xml** で **<context-param>** を使用して設定されるようになりました。

設定プロバイダー

```
<context-param>
  <param-name>org.picketlink.federation.saml.CONFIG_PROVIDER</param-name>
  <param-value>MyConfigurationProvider</param-value>
</context-param>
```

Audit Helper

```
<context-param>
  <param-name>org.picketlink.federation.saml.AUDIT_HELPER</param-name>
  <param-value>MyAuditHelper</param-value>
</context-param>
```

設定のリフレッシュインターバル

```
<context-param>
  <param-name>org.picketlink.federation.saml.REFRESH_CONFIG_TIMER_INTERVAL</param-
name>
  <param-value>1000</param-value>
</context-param>
```

文字エンコーディング

```
<context-param>
  <param-name>org.picketlink.federation.saml.CHARACTER_ENCODING</param-name>
  <param-value>UTF-8</param-value>
</context-param>
```

ユーザープリンシパルを属性マネージャーに渡す

```
<context-param>
  <param-
name>org.picketlink.federation.saml.PASS_USER_PRINCIPAL_TO_ATTRIBUTE_MANAGER</para
m-name>
  <param-value>true</param-value>
</context-param>
```

picketlink.xml のカスタムロケーション

```
<context-param>
  <param-name>CONFIG_FILE</param-name>
```

```
  <param-value>/path/to/picketlink.xml</param-value>
</context-param>
```

2.8.2. 依存関係宣言の変更

これまでのバージョンの JBoss EAP と同様に、**jboss-deployment-structure.xml** ファイルを使用して適切な依存関係を宣言する必要がありますが、モジュールには **services="import"** を含める必要もあります。これまでのバージョンの JBoss EAP では、同じ依存関係を宣言していましたが、SAML 認証システムをインストールするためのバルブが宣言されていました。また、**services="import"** が除外されました。JBoss EAP 7 で Undertow が導入され、**services="import"** を使用して SAML 認証システムをインストールするようになりました。

jboss-deployment-structure.xml を使用した依存関係の宣言

```
<jboss-deployment-structure>
  <deployment>
    <dependencies>
      <module name="org.picketlink" services="import"/>
    </dependencies>
  </deployment>
</jboss>
```

2.8.3. オーセンティケーターの変更

SAML オーセンティケーターは JBoss EAP の **FORM** オーセンティケーターを拡張します。SAML オーセンティケーターを有効にするには、**web.xml** で **FORM** 認証用に設定された **<login-config>** を定義する必要があります。

例: web.xml ファイル

```
<web-app>
  ...
  <login-config>
    <auth-method>FORM</auth-method>
    ...
  </login-config>
</web-app>
```

2.8.4. 動的アカウント選択の変更

以前のリリースの JBoss EAP では、バルブに渡されるパラメーターを使用して動的アカウント選択の一部が設定されていました。JBoss EAP 7 ではバルブが使用されなくなったため、この設定は **picketlink.xml** に含まれる **Provider** 要素に移動しました。これらの設定オプションの詳細は、Configuring a Dynamic Account Chooser の項を参照してください。

以前のリリースの JBoss EAP では、カスタム **IDPMapProvider** の作成時に実装された **org.picketlink.identity.federation.bindings.tomcat.sp.AbstractAccountChooserValve.AccountIDPMapProvider** インターフェイスも定義されていました。JBoss EAP 7 ではこのインターフェイスは存在しなくなり、**org.picketlink.identity.federation.web.config.IdentityURLConfigurationProvider** インターフェイスを実装する必要があります。この新しいインターフェイスのコントラクトは同じになります。

2.9. その他の機能

2.9.1. SAML Assertion Encryption

IDP と SP との間で SSL/TLS 暗号化を提供する他に、SAML アサーション自体も暗号化できます。これは、SSL/TLS を使用しないなど、セキュアでない方法で送信される SAML v2 アサーションのセキュリティーを保護する際に便利です。

IDP および SP で直接セキュリティーアサーションの暗号化を有効にするには、IDP ファイルと SP **picketlink.xml** ファイルの両方で以下の手順を実行する必要があります。

1. **Encrypt** および **SupportsSignatures** を有効化します。
   暗号化を有効にするには、**<PicketLinkIDP>** および **<PicketLinkSP>** を更新する必要があります。

   IDP の場合は、**Encrypt** および **SupportsSignatures** 属性を **<PicketLinkIDP>** に、true になるように追加または更新します。

   ```
   <PicketLink xmlns="urn:picketlink:identity-federation:config:2.1">
     ...
     <PicketLinkIDP xmlns="urn:picketlink:identity-federation:config:2.1" Encrypt="true"
   SupportsSignatures="true">
       ...
     </PicketLinkIDP>
   </PicketLink>
   ```

   SP の場合は、**<PicketLinkSP>** の **SupportsSignatures** に追加または更新します。

   ```
   <PicketLink xmlns="urn:picketlink:identity-federation:config:2.1">
     ...
     <PicketLinkSP xmlns="urn:picketlink:identity-federation:config:2.1"
   SupportsSignatures="true">
       ...
     </PicketLinkSP>
   </PicketLink>
   ```

2. ハンドラーの追加。
   さらに、ハンドラーを **<Handlers>** に追加する必要があります。

   IDP では、**SAML2EncryptionHandler** および **SAML2SignatureValidationHandler** を **picketlink.xml** ファイルに追加します。

   ```
   <PicketLink xmlns="urn:picketlink:identity-federation:config:2.1">
     ...
     <Handlers xmlns="urn:picketlink:identity-federation:handler:config:2.1">
       <Handler
   class="org.picketlink.identity.federation.web.handlers.saml2.SAML2IssuerTrustHandler" />
       <Handler
   class="org.picketlink.identity.federation.web.handlers.saml2.SAML2LogOutHandler" />
       <Handler
   class="org.picketlink.identity.federation.web.handlers.saml2.SAML2AuthenticationHandler" />
       <Handler
   class="org.picketlink.identity.federation.web.handlers.saml2.RolesGenerationHandler" />
       <Handler
   class="org.picketlink.identity.federation.web.handlers.saml2.SAML2EncryptionHandler" />
       <Handler
   class="org.picketlink.identity.federation.web.handlers.saml2.SAML2SignatureValidationHandler'
   ```

```
/>
  </Handlers>
</PicketLink>
```

SP では、**SAML2SignatureGenerationHandler** および **SAML2SignatureValidationHandler** を **picketlink.xml** ファイルに追加します。

```
<PicketLink xmlns="urn:picketlink:identity-federation:config:2.1">
  ...
  <Handlers xmlns="urn:picketlink:identity-federation:handler:config:2.1">
    <Handler
class="org.picketlink.identity.federation.web.handlers.saml2.SAML2LogOutHandler" />
    <Handler
class="org.picketlink.identity.federation.web.handlers.saml2.SAML2AuthenticationHandler" />
    <Handler
class="org.picketlink.identity.federation.web.handlers.saml2.RolesGenerationHandler" />
    <Handler
class="org.picketlink.identity.federation.web.handlers.saml2.SAML2SignatureGenerationHandle
r" />
    <Handler
class="org.picketlink.identity.federation.web.handlers.saml2.SAML2SignatureValidationHandler'
/>
  </Handlers>
</PicketLink>
```

警告

ハンドラーは、Chain of Responsibility を使用して実装されます。各ハンドラーは、**picketlink.xml** で定義された順序で要求および応答でロジックを実行します。ハンドラーが設定される順序に注意することが非常に重要です。

SAML2SignatureGenerationHandler は **SAML2EncryptoinHandler** と同じチェーンで設定することはできません。これにより、SAML メッセージが複数回署名されます。

3. キープロバイダーを設定します。
最後に、**<KeyProvider>** 要素を **両方の picketlink.xml** ファイルに追加する必要があります。この要素は、セキュリティーアサーションの暗号化および復号化に使用される Java キーストアにアクセスするための場所と認証情報を提供します。Java キーストアの生成例は、JBoss EAP **サーバーセキュリティーの設定方法** を参照してください。

IDP の要素は **<PicketLinkIDP>** に追加する必要があります。

```
<PicketLink xmlns="urn:picketlink:identity-federation:config:2.1">
  ...
  <PicketLinkIDP xmlns="urn:picketlink:identity-federation:config:2.1" Encrypt="true"
SupportsSignatures="true">
    ...
    <KeyProvider
```

```
ClassName="org.picketlink.identity.federation.core.impl.KeyStoreKeyManager">
      <Auth Key="KeyStoreURL" Value="/my_keystore.jks" />
      <Auth Key="KeyStorePass" Value="store123" />
      <Auth Key="SigningKeyPass" Value="test123" />
      <Auth Key="SigningKeyAlias" Value="servercert" />
      <ValidatingAlias Key="idp.example.com" Value="servercert" />
      <ValidatingAlias Key="localhost" Value="servercert" />
      <ValidatingAlias Key="sp1.example.com" Value="servercert" />
      <ValidatingAlias Key="sp2.example.com" Value="servercert" />
    </KeyProvider>
    ...
  </PicketLinkIDP>
  ...
<PicketLink>
```

SP の場合は、**<PicketLinkSP>** に要素を追加する必要があります。

```
<PicketLink xmlns="urn:picketlink:identity-federation:config:2.1">
  ...
  <PicketLinkSP xmlns="urn:picketlink:identity-federation:config:2.1"
SupportsSignatures="true">
    ...
    <KeyProvider
ClassName="org.picketlink.identity.federation.core.impl.KeyStoreKeyManager">
      <Auth Key="KeyStoreURL" Value="/my_keystore.jks" />
      <Auth Key="KeyStorePass" Value="store123" />
      <Auth Key="SigningKeyPass" Value="test123" />
      <Auth Key="SigningKeyAlias" Value="servercert" />
      <ValidatingAlias Key="idp.example.com" Value="servercert" />
      <ValidatingAlias Key="localhost" Value="servercert" />
    </KeyProvider>
    ...
  </PicketLinkSP>
</PicketLink>
```

注記

アサーションを適切に暗号化および復号化するには、IDP は署名を生成する必要があり、SP はこれらの署名を検証し、その出所を特定する必要があります。これは、**<ValidatingAlias>** 要素を使用して実行できます。IDP には、信頼される信頼されたサーバー/ドメイン (**<Trust>** 要素のすべてのエントリー) ごとに **<ValidatingAlias>** が必要です。SPS には、IDP を含むサーバー/ドメインごとに **<ValidatingAlias>** が必要です。

2.9.2. アサーションでのデジタル署名

デジタル署名により、IDP は SAML v2 セキュリティーアサーションに署名し、SP によって署名とアサーションを検証できます。これは、特に SSL/TLS を使用しないなど、セキュアでない方法で送信されるアサーションの信頼性を検証する際に便利です。

IDP および SP で直接セキュリティーアサーションのデジタル署名を有効にするには、IDP ファイルと SP **picketlink.xml** ファイルの両方で以下の手順を実行する必要があります。

1. **SupportsSignatures** を有効にします。

デジタル署名を有効にするには、**<PicketLinkIDP>** 要素および **<PicketLinkSP>** 要素を更新する必要があります。

IDP および SP の場合は、**<PicketLinkSP>** の **SupportsSignatures** 属性を true に追加または更新します。

```
<PicketLink xmlns="urn:picketlink:identity-federation:config:2.1">
  ...
  <PicketLinkIDP xmlns="urn:picketlink:identity-federation:config:2.1"
SupportsSignatures="true">
    ...
  </PicketLinkIDP>
</PicketLink>
```

```
<PicketLink xmlns="urn:picketlink:identity-federation:config:2.1">
  ...
  <PicketLinkSP xmlns="urn:picketlink:identity-federation:config:2.1"
SupportsSignatures="true">
    ...
  </PicketLinkSP>
</PicketLink>
```

2. ハンドラーの追加。
さらに、ハンドラーを **<Handlers>** に追加する必要があります。

IDP および SP では、**SAML2SignatureGenerationHandler** および **SAML2SignatureValidationHandler** を **picketlink.xml** ファイルに追加します。

IDP picketlink.xml

```
<PicketLink xmlns="urn:picketlink:identity-federation:config:2.1">
  ...
  <Handlers xmlns="urn:picketlink:identity-federation:handler:config:2.1">
    <Handler
class="org.picketlink.identity.federation.web.handlers.saml2.SAML2IssuerTrustHandler" />
    <Handler
class="org.picketlink.identity.federation.web.handlers.saml2.SAML2LogOutHandler" />
    <Handler
class="org.picketlink.identity.federation.web.handlers.saml2.SAML2AuthenticationHandler" />
    <Handler
class="org.picketlink.identity.federation.web.handlers.saml2.RolesGenerationHandler" />
    <Handler
class="org.picketlink.identity.federation.web.handlers.saml2.SAML2SignatureGenerationHandle
r" />
    <Handler
class="org.picketlink.identity.federation.web.handlers.saml2.SAML2SignatureValidationHandler'
/>
  </Handlers>
</PicketLink>
```

SP picketlink.xml

```
<PicketLink xmlns="urn:picketlink:identity-federation:config:2.1">
  ...
```

```
  <Handlers xmlns="urn:picketlink:identity-federation:handler:config:2.1">
    <Handler
class="org.picketlink.identity.federation.web.handlers.saml2.SAML2LogOutHandler" />
    <Handler
class="org.picketlink.identity.federation.web.handlers.saml2.SAML2AuthenticationHandler" />
    <Handler
class="org.picketlink.identity.federation.web.handlers.saml2.RolesGenerationHandler" />
    <Handler
class="org.picketlink.identity.federation.web.handlers.saml2.SAML2SignatureGenerationHandle
r" />
    <Handler
class="org.picketlink.identity.federation.web.handlers.saml2.SAML2SignatureValidationHandler'
/>
  </Handlers>
</PicketLink>
```

警告

ハンドラーは、Chain of Responsibility を使用して実装されます。各ハンドラーは、**picketlink.xml** で定義された順序で要求および応答でロジックを実行します。ハンドラーが設定される順序に注意することが非常に重要です。

SAML2SignatureGenerationHandler は **SAML2EncryptionHandler** と同じチェーンで設定することはできません。これにより、SAML メッセージが複数回署名されます。

3. キープロバイダーを設定します。
最後に、**<KeyProvider>** 要素を **両方の picketlink.xml** ファイルに追加する必要があります。この要素は、セキュリティーアサーションの署名に使用される Java キーストアにアクセスするための場所およびクレデンシャルを提供します。Java キーストアの生成例は、JBoss EAP **サーバーセキュリティーの設定方法** を参照してください。

   IDP の要素は **<PicketLinkIDP>** に追加する必要があります。

```
<PicketLink xmlns="urn:picketlink:identity-federation:config:2.1">
  ...
  <PicketLinkIDP xmlns="urn:picketlink:identity-federation:config:2.1"
SupportsSignatures="true">
    ...
    <KeyProvider
ClassName="org.picketlink.identity.federation.core.impl.KeyStoreKeyManager">
      <Auth Key="KeyStoreURL" Value="/my_keystore.jks" />
      <Auth Key="KeyStorePass" Value="store123" />
      <Auth Key="SigningKeyPass" Value="test123" />
      <Auth Key="SigningKeyAlias" Value="servercert" />
      <ValidatingAlias Key="idp.example.com" Value="servercert" />
      <ValidatingAlias Key="localhost" Value="servercert" />
      <ValidatingAlias Key="sp1.example.com" Value="servercert" />
      <ValidatingAlias Key="sp2.example.com" Value="servercert" />
```

```
    </KeyProvider>
    ...
  </PicketLinkIDP>
  ...
<PicketLink>
```

SP の場合は、**<PicketLinkSP>** に要素を追加する必要があります。

```
<PicketLink xmlns="urn:picketlink:identity-federation:config:2.1">
  ...
  <PicketLinkSP xmlns="urn:picketlink:identity-federation:config:2.1"
SupportsSignatures="true">
    ...
    <KeyProvider
ClassName="org.picketlink.identity.federation.core.impl.KeyStoreKeyManager">
      <Auth Key="KeyStoreURL" Value="/my_keystore.jks" />
      <Auth Key="KeyStorePass" Value="store123" />
      <Auth Key="SigningKeyPass" Value="test123" />
      <Auth Key="SigningKeyAlias" Value="servercert" />
      <ValidatingAlias Key="idp.example.com" Value="servercert" />
      <ValidatingAlias Key="localhost" Value="servercert" />
    </KeyProvider>
    ...
  </PicketLinkSP>
</PicketLink>
```

注記

アサーションを適切に暗号化および復号化するには、IDP は署名を生成する必要があり、SP はこれらの署名を検証し、その出所を特定する必要があります。これは、**<ValidatingAlias>** 要素を使用して実行できます。IDP には、信頼される信頼されたサーバー/ドメイン (**<Trust>** 要素のすべてのエントリー) ごとに **<ValidatingAlias>** が必要です。SPS には、IDP を含むサーバー/ドメインごとに **<ValidatingAlias>** が必要です。

2.9.3. 動的アカウント選択の設定

サービスプロバイダーが複数のアイデンティティープロバイダーで設定されている場合は、そのサービスプロバイダーが、クレデンシャルの認証に使用する IDP を選択するように要求するように設定できます。動的アカウント選択でサービスプロバイダーを設定するには、以下を行う必要があります。

1. すべてのアイデンティティープロバイダーを設定します。
   アイデンティティープロバイダーのセットアップに関する詳細は、IDP の設定 セクションを参照してください。

2. **WEB-INF/idpmap.properties** ファイルを設定します。
   name=url 形式を使用して利用可能なすべてのアイデンティティープロバイダーを一覧表示する **WEB-INF/idpmap.properties** ファイルを作成する必要があります。

   WEB-INF/idpmap.properties の例

```
Domain=http://localhost:8080/idp/
Domain-Alt=http://localhost:8080/idp-alt/
```

3. アカウント選択ランディングページを作成します。
ユーザーが認証するアイデンティティープロバイダーを選択できるようにするには、アカウントランディングページを作成して、サービスプロバイダーに追加する必要があります。このページには、認証を許可するすべてのアイデンティティープロバイダーへのリンクが含まれている必要があります。

accountChooser.html の例

```
<html>
  <head>...</head>
  <body>
    <h1>Account Chooser</h1>
    <ul>
      <li><a href="?idp=Domain">Domain</a>
      <li><a href="?idp=Domain-Alt">Domain Alt</a>
    </ul>
  </body>
</html>
```

4. **picketlink.xml** xml で **IdentityURL** 要素を設定します。
また、アカウント選択ランディングページを参照するように、**picketlink.xml** で **IdentityURL** 要素を設定する必要もあります。サービスプロバイダーの他の **picketlink.xml** の設定に関する詳細は、SP の設定 の項を参照してください。

picketlink.xml の例

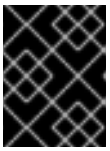
```
<PicketLink xmlns="urn:picketlink:identity-federation:config:2.1">
  <PicketLinkSP xmlns="urn:picketlink:identity-federation:config:2.1"
BindingType="REDIRECT">
    <IdentityURL>
      <Provider Page="/accountChooser.html"/>
    </IdentityURL>
    ...
  </PicketLinkSP>
  <Handlers xmlns="urn:picketlink:identity-federation:handler:config:2.1">
    ...
  </Handlers>
</PicketLink>
```

Provider 要素で利用可能な属性を使用して、動的アカウント選択の追加オプションを設定できます。

表2.4 Provider 要素属性

オプション	タイプ	デフォルト	説明
Page	String	/accountChooser.html	異なる IDP アカウントを一覧表示する HTML/JSP ページの名前。

オプション	タイプ	デフォルト	説明
expiration	Integer	-1	Cookie の有効期限 (秒単位)。デフォルトは **-1** です.これは、ブラウザーが閉じられると Cookie が期限切れすることを意味します。
DefaultURL	String		デフォルト IDP の URL。
Domain	String		ユーザーのブラウザーに送信されるクッキーに使用されるドメイン名。
Type	String		デフォルト実装を置き換える IDP マッピングの実装の完全修飾名。この実装では、**org.picketlink.identity.federation.web.config.IdentityURLConfigurationProvider** を実装する必要があります。デフォルトの実装では、SP Web アプリケーションの **WEB-INF/idpmap.properties** ファイルを使用します。



重要

picketlink-federation サブシステムを使用した動的アカウント選択の設定はサポートされません。

2.9.4. AJAX リクエストの処理

特定のケースでは、SP がセキュアなリソースに対する AJAX リクエストを受信する必要がある場合があります。これは追加設定なしに自動的に処理され、認証されたユーザーが AJAX 呼び出しを実行できるようにします。

これは、要求に **X-Requested-With** ヘッダーが存在するかどうかをチェックすることによって実行されます。REL 要求は、**X-Requested-With** ヘッダーの **XMLHttpRequest** の値で識別されます。さらに、ユーザーが認証されず、AJAX を使用して IDP と SP の両方へリクエストを送信する場合、PicketLink はログインページの代わりに **403** HTTP ステータスコードで応答します。

Revised on 2023-01-28 12:57:23 +1000