uApproveJP-4.0.0 のインストールおよび設定方法

この文書にはuApprove Jet Pack 4.0 (以下、「uApprove JP」) のインストールガイドと総合マニュアルが記されています。

uApprove JPはShibboleth Identity Provider 4を拡張するプラグインです。uApprove Jet Pack 3.4で提供していた機能をShibboleth Identity Provider 4でも利用できるようにすることを目的としています。これを利用することにより、利用者はIdentity Providerで認証する際に、属性を選択的に送信することができます。 uApprove JPのコンセプトに関するより詳細な情報はこちらを参照してください。

本ガイドに関する注意事項:

- このガイドでは、uApprove JPはLinuxシステムにインストールされると仮定しています。Windows等の他のオペレーティングシステムにインストールすることも可能です。その場合は、いくつかのパスやコマンドを適切なものに置き換えてください。
- このガイドでは、パスやコマンドはIDP_HOME、$UAPPROVE_INSTALL$といった変数で示されます。明示的に置き換えが不要と書かれていない限りは、これらの変数は実際のパスに置き換えてください。

目次

- 目次
- 想定
- 前提条件
- 1 基本的なデプロイ
  - 1.1 ライブラリのインストール
  - 1.2 Velocity テンプレートファイル
  - 1.3 CSS ファイル
  - 1.4 メッセージファイル
  - 1.5 設定のカスタマイズ
  - 1.6 カスタムテンプレート
  - 1.7 ログの設定
  - 1.8 デプロイ
- 2 高度なデプロイ
  - 2.1 リレーショナルデータベースを用いたユーザ同意情報の保存
  - 2.2 テンプレート
  - 2.3 ローカライズ
- 3 AttributeInMetadata
  - 3.1 AttributeInMetadataマッチングルールの設定
- 4 トラブルシューティング
  - 4.1 トラブルシューティング
  - 4.2 詳細なログ設定
- A SPでの属性使用用途通知
  - A.1 設定

想定

- Shibboleth Identity Providerは、 `$IDP_HOME$` (例: `/opt/shibboleth-idp`) にインストールされているものとします。
- Tomcatではなく、Jettyがインストールされているものとします。
- uApprove JP は、`$UAPPROVE_INSTALL$` (例: `/usr/local/src/uApproveJP-#version#`) にダウンロード、展開されているものとします。

前提条件

- Shibboleth Identity Provider 4.0.0以降がインストールされている必要があります。

> Shibboleth Identity Provider 4.0.0未満のバージョンでは動作しません。

1 基本的なデプロイ

1.1 ライブラリのインストール

ライブラリをIdPのライブラリディレクトリにコピーします:

```
# cp $UAPPROVE_INSTALL$/lib/*.jar $IDP_HOME$/edit-webapp/WEB-INF/lib/
```

⚠ IDP_HOME/edit-webapp/WEB-INF/libにはそれぞれのライブラリの単一のバージョンのみが存在するようにしてください。

1.2 Velocity テンプレートファイル

属性選択画面用のVelocityテンプレートファイルをIdPのviewsディレクトリに上書きコピーします:

```
# cp $UAPPROVE_INSTALL$/manual/examples/views/intercept/* $IDP_HOME$/views/intercept/
以下のメッセージが表示される場合がありますが、y を入力してください。
cp: `/opt/shibboleth-idp/views/intercept/attribute-release.vm' を上書きしますか?
```

1.3 CSS ファイル

CSSファイルをIdPのedit-webappディレクトリにコピーします:

```
# cp $UAPPROVE_INSTALL$/manual/examples/edit-webapp/css/* $IDP_HOME$/edit-webapp/css/
```

1.4 メッセージファイル

メッセージファイルをIdPのメッセージディレクトリにコピーします:

```
# cp $UAPPROVE_INSTALL$/manual/examples/messages/* $IDP_HOME$/messages/
```

IDP_HOME/conf/services.xmlに、以下の変更を行います。id="shibboleth.MessageSourceResources"に<value>%{idp.home}/messages/uApproveJP</value>を追加してください:

IDP_HOME/conf/services.xml

```
...

<!--
This collection of resources differs slightly in that it should not include the file extension.
Message sources are internationalized, and Spring will search for a compatible language extension
and fall back to one with only a .properties extension.
-->
<util:list id="shibboleth.MessageSourceResources">
    <value>%{idp.home}/messages/uApproveJP</value>
    <value>%{idp.home}/messages/messages</value>

...
```

1.5 設定のカスタマイズ

IDP_HOME/conf/idp.propertiesに、以下の変更を行います。idp.consent.allowPerAttributeとidp.consent.compareValuesの値をtrueに設定してください:

IDP_HOME/conf/idp.properties

```
...

# Flags controlling how built-in attribute consent feature operates
#idp.consent.allowDoNotRemember = true
#idp.consent.allowGlobal = true
idp.consent.allowPerAttribute = true

# Whether attribute values and terms of use text are compared
idp.consent.compareValues = true

...
```

IDP_HOME/conf/global.xmlに、以下の属性選択画面で使用するbean定義を追加します:

挿入する場所に制限はありませんが、よく分からなければ末尾の行に</beans>という閉じタグがあると思いますので、その直前に挿入してください。

IDP_HOME/conf/global.xml

```
...

<bean id="shibboleth.FallbackLanguages" parent="shibboleth.CommaDelimStringArray" c:_0="#{'%{idp.ui.fallbackLanguages:}'.trim()}" />
<util:map id="shibboleth.CustomViewContext">
    <entry key="OptionalAttributeFunction">
        <bean class="jp.gakunin.idp.consent.logic.impl.OptionalAttributeFunction" />
    </entry>
    <entry key="AttributeIntendedUseFunction">
        <bean class="jp.gakunin.idp.consent.logic.impl.AttributeIntendedUseFunction" p:defaultLanguages-ref="shibboleth.
FallbackLanguages" />
    </entry>
</util:map>

...
```

Shibboleth IdP 4.1以降では、以下の2つのファイルの変更（services-system.xmlおよびattribute-release-beans.xml）についてファイルの場所が変更になっております。以下の手順で idp-conf-impl-4.1.x.jar ファイルを上書きしてください。

/opt/shibboleth-idp/dist/webapp/WEB-INF/lib/idp-conf-impl-4.?.?.jar
を適当な空のディレクトリでunzipしまして、中の
net/shibboleth/idp/conf/services-system.xml
および
net/shibboleth/idp/flows/intercept/attribute-release-beans.xml
につきまして、下記記載の通り修正を行ってください。
再度全体をzipしてidp-conf-impl-4.?.?.jar（?の部分はバージョン番号で置き換え）というファイル名にして元のファイルを上書きしてください。

IDP_HOME/system/conf/services-system.xmlに、以下の変更を行います。id="shibboleth.AttributeFilterService"のbean定義の<constructor-arg name="strategy">を以下のように変更してください:

この変更は、Shibboleth Identity Providerを再インストール(アップグレード等)する際に上書きされるため、再インストールを行った際には、再度変更を行う必要があります。

IDP_HOME/system/conf/services-system.xml

```
...

<bean id="shibboleth.AttributeFilterService" class="net.shibboleth.ext.spring.service.ReloadableSpringService"
        depends-on="shibboleth.VelocityEngine"
        p:serviceConfigurations-ref="#{'%{idp.service.attribute.filter.resources:shibboleth.AttributeFilterResources}'.trim()}"
        p:failFast="%{idp.service.attribute.filter.failFast:%{idp.service.failFast:false}}"
        p:reloadCheckDelay="%{idp.service.attribute.filter.checkInterval:PT0S}"
        p:beanPostProcessors-ref="shibboleth.IdentifiableBeanPostProcessor"
        p:beanFactoryPostProcessors-ref="shibboleth.PropertySourcesPlaceholderConfigurer">
    <constructor-arg name="claz" value="net.shibboleth.idp.attribute.filter.AttributeFilter" />
    <constructor-arg name="strategy">
        <bean class="jp.gakunin.idp.attribute.filter.spring.impl.AttributeFilterServiceStrategy"
            depends-on="shibboleth.AttributeRegistryService"
            p:transcoderRegistry-ref="shibboleth.AttributeRegistryService"
            id="ShibbolethAttributeFilter"/>
    </constructor-arg>
</bean>

...
```

IDP_HOME/system/flows/intercept/attribute-release-beans.xmlに、以下の変更を行います。id="IsConsentRequiredPredicate"のbean定義のclassを変更してください:

この変更は、Shibboleth Identity Providerを再インストール(アップグレード等)する際に上書きされるため、再インストールを行った際には、再度変更を行う必要があります。

IDP_HOME/system/flows/intercept/attribute-release-beans.xml

```
...

<bean id="IsConsentRequiredPredicate"
    class="jp.gakunin.idp.consent.logic.impl.IsConsentRequiredPredicate" />

...
```

1.6 カスタムテンプレート

テンプレートをカスタマイズしたい場合は、 テンプレートのカスタマイズ を参照してください。

少なくとも、所属機関のロゴを変更する必要があります。変更方法は以下のリンク先を参照してください。

GakuNinShare:Shibboleth IdP 3 - ロゴの変更

1.7 ログの設定

uApprove JPのログを出力するには、IDP_HOME/conf/logback.xmlに以下を追加します:

IDP_HOME/conf/logback.xml

```
...

<!-- Logging level shortcuts. -->
<variable name="idp.loglevel.uApproveJP" value="${idp.loglevel.uApproveJP:-INFO}" />

...

<!-- ========================================================= -->
<!-- ============== Logging Categories and Levels ============== -->
<!-- ========================================================= -->
<logger name="jp.gakunin.idp" level="${idp.loglevel.uApproveJP}"/>

...
```

1.8 デプロイ

IdP で uApprove JP を有効にするには IdP を再デプロイする必要があります:

```
# cd $IDP_HOME$
# ./bin/build.sh
Installation Directory: [/opt/shibboleth-idp]
[Enter]  ←入力なし
Rebuilding /opt/shibboleth-idp/war/idp.war ...
...done

BUILD SUCCESSFUL
Total time: 16 seconds
```

$CATALINA_BASE$/conf/Catalina/localhost/idp.xmlが存在しない場合は、追加の作業としてidp.warを$CATALINA_BASE$/webappsにコピーします:

```
# ls $CATALINA_BASE$/conf/Catalina/localhost/idp.xml
ls: cannot access /usr/java/tomcat/conf/Catalina/localhost/idp.xml: そのようなファイルやディレクトリはありません
# cp $IDP_HOME$/war/idp.war $CATALINA_BASE$/webapps/
```

Jettyを再起動します:

```
# systemctl restart jetty
```

2 高度なデプロイ

この節では高度な設定についてのトピックを取り上げます。

2.1 リレーショナルデータベースを用いたユーザ同意情報の保存

リレーショナルデータベース(以下、「RDB」とします)を用いて、ユーザ同意情報の保存を行うことができます。

2.1.1. MySQLの設定

⚠ 以下のデータベースパラメータは一例です。実際の値は必要に応じて変更してください。特にパスワードは安全なものを用意してください。

MySQLの設定を行います。

1. データベース shibbolethの作成
   Shibboleth IdPで使用するデータベース shibbolethを作成します:

   ℹ rootにパスワードが設定してあってコマンドが以下のエラーで失敗する場合は、-pオプションを追加してください。

   ```
   ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: NO)
   ```

   ```
   db$ mysql -u root
   mysql>
   CREATE DATABASE shibboleth;
   ```

2. ユーザ作成
   IdPからデータベース shibbolethにアクセスするためのユーザ shibbolethを作成し、データベース shibbolethへの権限を付与します:

   ```
   db$ mysql -u root
   mysql>
   CREATE USER 'shibboleth'@'localhost' IDENTIFIED BY 'shibpassword';          #←任意のパスワード
   GRANT INSERT, SELECT, UPDATE, DELETE ON shibboleth.* TO 'shibboleth'@'localhost';
   ```

3. StorageRecordsテーブル作成

   JPAStorageServiceが使用するテーブル StorageRecordsを作成します:

   ```
   db$ mysql -u root
   mysql>
   use shibboleth;
   CREATE TABLE `StorageRecords` (
       `context` varchar(255) NOT NULL,
       `id` varchar(255) NOT NULL,
       `expires` bigint(20) DEFAULT NULL,
       `value` longtext NOT NULL,
       `version` bigint(20) NOT NULL,
       PRIMARY KEY (`context`,`id`)
   ) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin;
   ```

2.1.3. MySQL Connector/Jのインストール

1. MySQLへのアクセスに必要なMySQL Connector/J(mysql-connector-java.jar)をインストールします:

   ```
   # yum install mysql-connector-java
   ```

2. /usr/share/java配下にインストールされているので、edit-webapp/ 配下のlibディレクトリにシンボリックリンクを作成します:

```
# rpm -ql mysql-connector-java
 （省略）
/usr/share/java/mysql-connector-java.jar
 （省略）
# ln -s /usr/share/java/mysql-connector-java.jar $IDP_HOME$/edit-webapp/WEB-INF/lib/
```

3. 1.8 デプロイの手順に従って再デプロイします。

2.1.4. idp.consent.StorageServiceの設定変更

idp.consent.StorageServiceの設定をshibboleth.JPAStorageServiceに変更します:

IDP_HOME/conf/idp.properties

```
# Set to "shibboleth.StorageService" or custom bean for alternate storage of consent
idp.consent.StorageService = shibboleth.JPAStorageService
```

2.1.5. shibboleth.JPAStorageServiceの設定

2.1.4. idp.consent.StorageServiceの設定変更でidp.consent.StorageServiceに設定したshibboleth.JPAStorageServiceを定義します。

id="Shibboleth.MySQLDataSource"のbean定義のp:url, p:username, p:passwordは、2.1.1. MySQLの設定に合わせて設定します:

IDP_HOME/conf/global.xml

```
<!-- Use this file to define any custom beans needed globally. -->
<bean id="shibboleth.JPAStorageService"
      class="org.opensaml.storage.impl.JPAStorageService"
      p:cleanupInterval="%{idp.storage.cleanupInterval:PT10M}"
      c:factory-ref="shibboleth.JPAStorageService.entityManagerFactory" />

<bean id="shibboleth.JPAStorageService.entityManagerFactory"
      class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">
     <property name="packagesToScan" value="org.opensaml.storage.impl" />
     <property name="dataSource" ref="shibboleth.MySQLDataSource" />
     <property name="jpaVendorAdapter" ref="shibboleth.JPAStorageService.JPAVendorAdapter" />
     <property name="jpaDialect">
          <bean class="org.springframework.orm.jpa.vendor.HibernateJpaDialect" />
     </property>
</bean>

<bean id="shibboleth.JPAStorageService.JPAVendorAdapter"
      class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter"
      p:database="MYSQL" />

<bean id="shibboleth.MySQLDataSource"
      class="org.apache.commons.dbcp2.BasicDataSource"
      p:driverClassName="com.mysql.jdbc.Driver"
      p:url="jdbc:mysql://localhost:3306/shibboleth"
      p:username="shibboleth"
      p:password="shibpassword"
      p:maxTotal="10"
      p:maxIdle="5"
      p:maxWaitMillis="15000"
      p:testOnBorrow="true"
      p:validationQuery="select 1"
      p:validationQueryTimeout="5" />
```

2.1.6. Jettyの再起動

Jettyを再起動します:

```
# systemctl restart jetty
```

2.2 テンプレート

テンプレートのカスタマイズ

IDP_HOME/views/にある Velocityテンプレートファイル 、 IDP_HOME/edit-webapp/にあるCSS や画像ファイルは自由にカスタマイズすることができます。 Velocity を用いているので容易にカスタマイズ出来るようになっています。

Velocity については Velocity User Guide を参照してください。

2.3 ローカライズ

Relying Partyの名前と説明

現状では、ローカライズされた Relying Party の名前と説明を取得する際には、メタデータのうち<AttributeConsumingService>要素および<mdui:UIInfo>要素がサポートされています。

この名前と説明を使用する場合は、SPのメタデータを以下のように記述します:

```
<EntityDescriptor entityID="https://sp.example.org/shibboleth">

    <!-- ... -->

    <SPSSODescriptor>
        <Extensions>
            <mdui:UIInfo xmlns:mdui="urn:oasis:names:tc:SAML:metadata:ui">
                <mdui:DisplayName xml:lang="en">Example SP</mdui:DisplayName>
                <!-- Service names in other languages -->
                <mdui:Description xml:lang="en">Some description of Example SP</mdui:Description>
                <!-- Service descriptions in other languages -->
            </mdui:UIInfo>
        </Extensions>

        <!-- ... -->

        <AttributeConsumingService index="1">
            <ServiceName xml:lang="en">Example SP</ServiceName>
            <!-- Service names in other languages -->
            <ServiceDescription xml:lang="en">Some description of Example SP</ServiceDescription>
            <!-- Service descriptions in other languages -->
        </AttributeConsumingService>

        <!-- ... -->

    </SPSSODescriptor>
</EntityDescriptor>
```

両方記載されている場合には<mdui:UIInfo>要素が優先されます。

学認のSPについても海外SPの一部を除いてこの情報が含まれています。

3 AttributeInMetadata

3.1 AttributeInMetadataマッチングルールの設定

このルールは、SP がその属性を必要とした場合に、そのメタデータにより属性の送信を許可します。属性は<SPSSODescriptor>中の<AttributeConsumingService>によって示されます。 <RequestedAttribute>でisRequired="true"を記述した属性は必須とマークされ、 isRequired="false"を記述した属性はオプションとマークされます。詳細は SAMLメタデータを参照してください。

以下の点に注意してください:

- このフィルタの利用には属性の要求者のメタデータがロードされ利用可能である必要があります。
- 要求者のメタデータは<SPSSODescriptor>ロールを持っている必要があります。このロールがリストされた属性を持っているためです。
- AttributeInMetadataマッチングルールは値のルールとしてのみ働き、<PermitValueRule>の場合のみ意味をなします。

名前空間の定義

属性フィルタのポリシーにおいて、このプラグイン用に名前空間の定義を加える必要があります。以下のように行います:

- ルート <AttributeFilterPolicyGroup>のxmlns:xsi属性の前に xmlns:uajpmf="http://www.gakunin.jp/ns/uapprove-jp/afp/mf" 属性を追加します。
- xsi:schemaLocation属性のホワイトスペースで区切られた値のリストの最後に以下を追加します: http://www.gakunin.jp/ns/uapprove-jp/afp/mf http://www.gakunin.jp/schema/idp/gakunin-afp-mf-uapprovejp.xsd

ルールの定義

このルールは<PermitValueRule xsi:type="uajpmf:AttributeInMetadata">のように記述します。以下のオプショナルな属性を使用できます:

onlyIfRequired	必須とマークされた属性のみ送信を許可し、オプションとマークされたものは送信しないブーリアンフラグです。 デフォルト値は **true** です。
matchIfMetadataSilent	メタデータに <AttributeConsumingService> がない場合にオプションとマークするかどうかを決定するブーリアンフラグです。 デフォルト値は **false** です。
onlyIfChecked	オプションとマークされた属性の送信を利用者が許可/拒否できるかどうかを示すブーリアンフラグです。 false の時の動作は Shibboleth IdP 3.2.0 以降の AttributeInMetadata と同一になり、オプションとマークされた属性も必須とマークされた属性と同様にチェックボックスなしで表示されます。 デフォルト値は **false** です。

AttributeInMetadataマッチファンクションを使用した<PermitValueRule>の書き方は下記のようになります:

```
<PermitValueRule xsi:type="uajpmf:AttributeInMetadata" onlyIfRequired="false"
                 onlyIfChecked="true"/>
```

オプションとマークされた属性をチェックボックスつきで表示します。チェックボックスをチェックしたときだけ送信します。

AttributeInMetadataマッチファンクションを使用した<PermitValueRule>の設定例を示します:

```
<!--    ==============================================================================
case 1: mail 属性、eduPersonPrincipalName属性、eduPersonAffiliation属性を、メタデータの
        定義と照合するルールです。

        メタデータでisRequired="true"が指定されている属性は、すべて必須情報になり常に
        送信されます。

        メタデータでisRequired="false"が指定されている属性は以下の通りです。
        * mail属性は必須情報となり常に送信されます。
        * eduPersonPrincipalName属性はオプション情報となります。属性選択画面ではチェック
          ボックスつきで表示されます。利用者がチェックボックスをチェックした場合に限り送信
          されます。
        * eduPersonAffiliation属性は送信されません。

        メタデータにAttributeConsumingServiceをもたないSPの場合はどの属性も送信しません。
        ============================================================================== -->
<AttributeFilterPolicy id="PolicyforSPwithAttributeConsumingService">
    <PolicyRequirementRule xsi:type="ANY" />

    <AttributeRule attributeID="mail">
        <PermitValueRule xsi:type="uajpmf:AttributeInMetadata"
                         onlyIfRequired="false" />
    </AttributeRule>

    <AttributeRule attributeID="eduPersonPrincipalName">
        <PermitValueRule xsi:type="uajpmf:AttributeInMetadata"
                         onlyIfRequired="false"
                         onlyIfChecked="true" />
    </AttributeRule>

    <AttributeRule attributeID="eduPersonAffiliation">
        <PermitValueRule xsi:type="uajpmf:AttributeInMetadata" />
    </AttributeRule>

</AttributeFilterPolicy>

<!--    ==============================================================================
case 2: メタデータに AttributeConsumingServiceがないSPに対するルールを追加したルール
        です。

        AttributeConsumingService要素を持たないSPの場合は以下の通りです。
        * mail属性は必須情報となり常に送信されます。
        * eduPersonPrincipalName属性はオプション情報となります。属性選択画面ではチェック
          ボックスつきで表示されます。利用者がチェックボックスをチェックした場合に限り送信
          されます。
        * eduPersonAffiliation属性は送信されません。

        AttributeConsumingService要素を持つSPの場合はcase 1と同じです。
        ============================================================================== -->
<AttributeFilterPolicy id="PolicyforSPwithoutAttributeConsumingService">
    <PolicyRequirementRule xsi:type="ANY" />

    <AttributeRule attributeID="mail">
        <PermitValueRule xsi:type="uajpmf:AttributeInMetadata"
                         matchIfMetadataSilent="true"
                         onlyIfRequired="false" />
    </AttributeRule>

    <AttributeRule attributeID="eduPersonPrincipalName">
        <PermitValueRule xsi:type="uajpmf:AttributeInMetadata"
                         matchIfMetadataSilent="true"
                         onlyIfRequired="false"
                         onlyIfChecked="true" />
    </AttributeRule>

    <AttributeRule attributeID="eduPersonAffiliation">
        <PermitValueRule xsi:type="uajpmf:AttributeInMetadata" />
    </AttributeRule>

</AttributeFilterPolicy>
```

4 トラブルシューティング

4.1 トラブルシューティング

- ERROR ないしWARN メッセージについては、IDP_HOME/logs/idp-process.logをチェックしてください。
- $CATALINA_BASE$/logsにあるTomcatのログファイルにエラーメッセージがないかチェックしてください。

4.2 詳細なログ設定

DEBUG ないしTRACEログレベルを有効にしたい場合は、IDP_HOME/conf/idp.propertiesにて以下を追加します:

IDP_HOME/conf/idp.properties
idp.loglevel.uApproveJP = DEBUG

A SPでの属性使用用途通知

SP管理者はSPのメタデータに属性の使用用途を記述することで、SPの利用者に属性の使用用途 (例えば、プロファイルの初期値として使用) をuApprove JPで表示することができます。

A.1 設定

属性使用用途通知機能は、<RequestedAttribute>にuajpmd:description属性を追加する、または、<SPSSODescriptor>の<Extensions>に<uajpmd:RequestedAttributeExtension>を追加することで利用できます。

<uajpmd:RequestedAttributeExtension>は多言語に対応しています。また、一つの属性に両方が設定されている場合は、<uajpmd:RequestedAttributeExtension>が優先されます。

これらを使う場合はメタデータの先頭に名前空間 xmlns:uajpmd="http://www.gakunin.jp/ns/uapprove-jp/metadata" の宣言を忘れないでください。

uajpmd:description

この属性は<ResquestedAttribute>にて定義します:

uajpmd:description	属性の使用用途の文字列です。

uajpmd:descrptionを使用した<RequestedAttribute>の設定例:

```
<md:RequestedAttribute FriendlyName="mail"
        Name="urn:oid:0.9.2342.19200300.100.1.3"
        NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
        uajpmd:description="The mail attribute is used as the initial value of the mail address field of the registration form."/>
```

<uajpmd:RequestedAttributeExtension>

<uajpmd:RequestedAttributeExtension>は以下の必須属性と一つ以上の<uajpmd:Description>と共に設定します:

uajpmd:FriendlyName	関連づけたい<RequestedAttribute>のFriendlyName属性の値です。

<uajpmd:Description>には属性の使用用途を記述します。<uajpmd:RequestedAttributeExtension>は以下の必須属性と共に設定します:

xml:lang	属性の使用用途の言語です。

<uajpmd:RequestedAttributeExtension>の設定例:

```
<md:EntitiesDescriptor Name="uapprovejp-dev-metadata.xml"
                    xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
                    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
                    xmlns:shibmd="urn:mace:shibboleth:metadata:1.0"
                    xmlns:uajpmd="http://www.gakunin.jp/ns/uapprove-jp/metadata"
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  ...
  <md:EntityDescriptor entityID="...">
    <md:SPSSODescriptor>
      ...

      <md:Extensions>
        ...
        <uajpmd:RequestedAttributeExtension FriendlyName="mail">
          <uajpmd:Description xml:lang="en">The mail attribute is used as the initial value of the mail address field of the
registration form.</uajpmd:Description>
          <uajpmd:Description xml:lang="ja">mail 属性を登録ページのメールアドレス欄の初期値として使用します</uajpmd:Description>
        </uajpmd:RequestedAttributeExtension>
        ...
      </md:Extensions>
      ...

      <md:AttributeConsumingService index="1">
              <md:ServiceName xml:lang="en">Sample Service</md:ServiceName>

        <md:RequestedAttribute FriendlyName="eduPersonPrincipalName"
                               Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.6"
                               NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
                               isRequired="true"/>
        <md:RequestedAttribute FriendlyName="mail"
                               Name="urn:oid:0.9.2342.19200300.100.1.3"
                               NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"/>
      </md:AttributeConsumingService>
      ...
    </md:SPSSODescriptor>
    ...
  </md:EntityDescriptor>
  ...
</md:EntitiesDescriptor>
```