旧: FPSP（Filter Per SP, ユーザに対する特定SPへのアクセス制限）プラグイン

標準的なIdPの設定では、ユーザの認証が成功すれば、SPにアサーションが送られます。認証されたIdPのentityIDしか参照しないなど、ユーザの属性情報を一切要求しないSPであれば、IdPでの認証が成功しさえすれば、SPにアクセスできてしまいます。このため、各SPへのアクセスに対 する、IdP側のポリシーの反映が困難です。

IdPのプラグイン機能を利用することにより、IdP側において、ユーザの属性情報に基づいた特定SPへのアクセス制御を実現することが可能です。任意の条件を持つプラグインが作成可能ですが、ここでは、簡易な条件判定機能を持つものとしてFPSP(Filter Per SP)プラグインを紹介します。

お知らせ

本プラグインはIdPv2向けに開発されたものです。ご注意ください。IdPv3については、Shibboleth IdPによるアクセス制限 を参照してください。

現行版のFPSPについて不具合や機能拡張の議論が情報交換MLで行われております。ご一読ください。

「[upki-fed:00792] Filter Per SP (FPSP) の動作に関する質問」から始まるスレッド
1 2 3 4 5 6 7

「[upki-fed:00805] SPに送信されない属性でFilter Per SP(FPSP)を制御する」から始まるスレッド
1 2 3 4

慶應義塾の細川さんが上記議論をまとめられていました（ありがとうございます）。ご参照ください。
https://memo.itc.keio.ac.jp/blog/?p=23

- 本ページに掲載している最新版（201303版）では、エラーページをJSPファイルとして設定できるようになりました。Shibboleth標準の error.jsp をハードコードしておりますが、このファイル名を変更すれば独自のページを表示させることも可能です。
- 名称をFPSPに変更しました。

FPSPプラグインの概要

FPSPプラグインは、設定ファイルに記述されているSP毎に特定の属性値を持つユーザーについてのみアサーション（認証結果・属性情報）を送信するためのIdPプラグインです。 一つのSPに対して、送信を許可する複数の属性値が設定可能です。

FPSPプラグインのインストール手順

1. 設定ファイルの解析に commons-configurationライブラリを使用しているため、下記URLより commons-configuration-1.X-bin.zip（執筆時点の最新版は1.10）を入手して、解凍したJARファイル commons-configuration-1.X.jar を /opt/shibboleth-idp/lib/ および ${CATALINA_HOME}/webapps/idp/WEB-INF/lib/ に配置する。
   http://commons.apache.org/configuration/

2. SampleFilterPerSP.java をコンパイルする。

```
$ for jar in /opt/shibboleth-idp/lib/servlet-api-*.jar ¥
    /opt/shibboleth-idp/lib/opensaml-*.jar ¥
    /opt/shibboleth-idp/lib/openws-*.jar ¥
    /opt/shibboleth-idp/lib/shibboleth-common-*.jar ¥
    /opt/shibboleth-idp/lib/shibboleth-identityprovider-*.jar ¥
    /opt/shibboleth-idp/lib/commons-collections-*.jar ¥
    /opt/shibboleth-idp/lib/commons-lang-*.jar ¥
    /opt/shibboleth-idp/lib/commons-configuration-*.jar
> do
> export CLASSPATH=${CLASSPATH:+${CLASSPATH}:}$jar
> done

$ javac -encoding shift_jis -Xlint:unchecked SampleFilterPerSP.java
```

3. コンパイルされた SampleFilterPerSP.classを ${CATALINA_HOME}/webapps/idp/WEB-INF/classes/ 以下に配置する。

```
$ mkdir -p ${CATALINA_HOME}/webapps/idp/WEB-INF/classes/plugin/idp
$ cp -p SampleFilterPerSP.class ${CATALINA_HOME}/webapps/idp/WEB-INF/classes/plugin/idp/
```

※ Shibboleth IdPをアップデートする場合には配置したファイルが失われる可能性があるので注意すること。

FPSPプラグインの設定手順

1. ${CATALINA_HOME}/webapps/idp/WEB-INF/web.xml に下記を追記する。パラメータ名 Config の値は後述する FPSP の設定ファイルのパスとなる。

```
<!-- SampleFilterPerSP -->
<filter>
  <filter-name>SampleFilterPerSP</filter-name>
  <filter-class>plugin.idp.SampleFilterPerSP</filter-class>
  <init-param>
    <param-name>Config</param-name>
    <param-value>
        /opt/shibboleth-idp/conf/SampleFilterPerSP_allow.xml
    </param-value>
  </init-param>
</filter>

<filter-mapping>
  <filter-name>SampleFilterPerSP</filter-name>
  <url-pattern>/profile/*</url-pattern>
  <dispatcher>REQUEST</dispatcher>
  <dispatcher>FORWARD</dispatcher>
</filter-mapping>
```

※ すでに uApprove.jp がインストールされているIdPにインストールする場合は、uApprove.jp の filter および filter-mapping よりも上に挿入してください。

2. 上記1のパラメータ名 Config で設定したファイル（SampleFilterPerSP_allow.xml）を /opt/shibboleth-idp/conf/ に配置する。

3. Shibbolet IdPを再起動する。

```
# /sbin/service tomcat6 restart
```

SampleFilterPerSP.java

下記内容のファイルをダウンロードする（リンク上で右クリックしてリンク先を保存してください）

```
/*
 * Copyright (c) 2011, National Institute of Informatics
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
```

```
 */
package plugin.idp;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.Collection;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Map;
import java.util.Set;

import org.apache.commons.configuration.Configuration;
import org.apache.commons.configuration.ConfigurationException;
import org.apache.commons.configuration.XMLConfiguration;

import edu.internet2.middleware.shibboleth.common.attribute.AttributeRequestException;
import edu.internet2.middleware.shibboleth.common.attribute.BaseAttribute;
import edu.internet2.middleware.shibboleth.common.attribute.provider.SAML2AttributeAuthority;
import edu.internet2.middleware.shibboleth.common.profile.AbstractErrorHandler;
import edu.internet2.middleware.shibboleth.common.profile.provider.BaseSAMLProfileRequestContext;
import edu.internet2.middleware.shibboleth.common.relyingparty.RelyingPartyConfiguration;
import edu.internet2.middleware.shibboleth.common.relyingparty.provider.SAMLMDRelyingPartyConfigurationManager;
import edu.internet2.middleware.shibboleth.idp.authn.AuthenticationException;
import edu.internet2.middleware.shibboleth.idp.authn.LoginContext;
import edu.internet2.middleware.shibboleth.idp.util.HttpServletHelper;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;

import org.opensaml.saml2.metadata.provider.MetadataProvider;
import org.opensaml.saml2.metadata.provider.MetadataProviderException;

public class SampleFilterPerSP implements Filter {
    /** web.xmlのパラメータキー */
    private static String CONFIGFILES = "Config";
    /** ServletContext  */
    private ServletContext m_servletContext = null;
    /** SP毎ののattributeフィルタの値 */
    private Map<String, Map> allow_config;
    /** 属性のContext   */
    private SAML2AttributeAuthority m_saml2AttrAuth;
    /** RelyingPartyのContext    */
    private SAMLMDRelyingPartyConfigurationManager m_samlRelyingPartyConfMan;


    @Override
    public void destroy() {
        // TODO 自動生成されたメソッド・スタブ
    }

    @Override
    public void doFilter(
                        ServletRequest servletRequest, ServletResponse servletResponse,
                        FilterChain filterChain) throws IOException, ServletException {
        HttpServletRequest httpServletRequest = (HttpServletRequest) servletRequest;

        // フィルタ条件がない場合は次のフィルタへ
        if (allow_config.isEmpty()) {
            filterChain.doFilter(servletRequest, servletResponse);
            return;
        }

        // Shibboleth のログイン情報
        // see also: https://wiki.shibboleth.net/confluence/display/SHIB2/IdPAuthUserPass
        LoginContext loginCtx = HttpServletHelper.getLoginContext(
                HttpServletHelper.getStorageService(m_servletContext), m_servletContext, httpServletRequest);
```

```
        if (loginCtx == null) {
            // まだログインしていない場合はログイン処理へ
            filterChain.doFilter(servletRequest, servletResponse);
            return;
        }
        // ログイン結果判定
        AuthenticationException authenticationFailure = loginCtx.getAuthenticationFailure();
        if (authenticationFailure != null) {
            filterChain.doFilter(servletRequest, servletResponse);
            return;
        }
        // ログインユーザとSPのエンティティID
        String spEntityId = loginCtx.getRelyingPartyId();
        String userName = loginCtx.getPrincipalName();

        // 属性情報リクエスト用の情報
        BaseSAMLProfileRequestContext requestCtx = new BaseSAMLProfileRequestContext();
        // SPに対応するRelyingParty
        RelyingPartyConfiguration relyingPartyConfiguration =
            m_samlRelyingPartyConfMan.getRelyingPartyConfiguration(spEntityId);
        // メタデータプロバイダ情報
        MetadataProvider metadataProvider = m_samlRelyingPartyConfMan.getMetadataProvider();

        // IdPのエンティティID
        String idpEntityId = relyingPartyConfiguration.getProviderId();

        // リクエスト情報設定
        try {
            requestCtx.setRelyingPartyConfiguration(relyingPartyConfiguration);
            requestCtx.setInboundMessageIssuer(spEntityId);
            requestCtx.setOutboundMessageIssuer(idpEntityId);
            requestCtx.setPrincipalName(userName);
            requestCtx.setLocalEntityId(idpEntityId);
            requestCtx.setPeerEntityId(spEntityId);
            requestCtx.setLocalEntityMetadata(metadataProvider.getEntityDescriptor(idpEntityId));
            requestCtx.setPeerEntityMetadata(metadataProvider.getEntityDescriptor(spEntityId));
            requestCtx.setMetadataProvider(metadataProvider);
        } catch (MetadataProviderException e) {
            e.printStackTrace();
            filterChain.doFilter(servletRequest, servletResponse);
            return;
        }

        // 属性を取得
        Map<String, BaseAttribute> attributes = null;
        try {
            attributes = m_saml2AttrAuth.getAttributes(requestCtx);
        } catch (AttributeRequestException e) {
            e.printStackTrace();
        }
        if (attributes == null) {
            filterChain.doFilter(servletRequest, servletResponse);
            return;
        }
        // コンフィグの設定と比較
        System.out.println("SampleFilterPerSP spEntityId = " + spEntityId);
        if (allow_config.containsKey(spEntityId)) {
            Map filterpersp = allow_config.get(spEntityId);
            for (BaseAttribute baseattrval : attributes.values()) {
                if (filterpersp.containsKey(baseattrval.getId())) {
                    Set val = (Set)filterpersp.get(baseattrval.getId());
                    for (Object obj: baseattrval.getValues()) {
                        if (obj != null && !obj.toString().trim().equals("")) {
                            String attrval = obj.toString();
                            if (val.contains(attrval)) {
                                // for debug
                                System.out.println("SampleFilterPerSP checked " + baseattrval.getId() + "=" + attrval);
                                filterChain.doFilter(servletRequest, servletResponse);
                                return;
                            }
                        }
                    }
                }
            }
            // for debug
            System.out.println("SampleFilterPerSP don't match any attributes.");
```

```
            for (BaseAttribute baseattrval : attributes.values()) {
                for (Object obj: baseattrval.getValues()) {
                    if (obj != null && !obj.toString().trim().equals("")) {
                        String attrval = obj.toString();
                        System.out.println("SampleFilterPerSP   " + baseattrval.getId() + "=" + attrval);
                    }
                }
            }

            doError(servletRequest, servletResponse, spEntityId, userName);
            return;
        }

        filterChain.doFilter(servletRequest, servletResponse);
    }

    /**
     * エラーの場合エラー内容表示
     * @param servletResponse
     * @param spEntityId
     * @param userName
     * @throws IOException
     */
    private void doError(ServletRequest servletRequest, ServletResponse servletResponse,
                         String spEntityId, String userName) throws IOException, ServletException {
        final String PAGE_ERROR = "/error.jsp";
        final String errmsg = "Deny you(" + userName + ") login to Service Provider " + spEntityId;

        // エラーメッセージの表示など
        Throwable error = new Exception(errmsg);

        servletRequest.setAttribute(AbstractErrorHandler.ERROR_KEY, error);
        m_servletContext.getRequestDispatcher(PAGE_ERROR).forward(servletRequest, servletResponse);
    }

    @Override
    public void init(FilterConfig filterConfig) throws ServletException {
        // web.xmlのコンフィグなど
        m_servletContext = filterConfig.getServletContext();
        allow_config = new HashMap<String, Map>();

        String configfilename = filterConfig.getInitParameter(CONFIGFILES);
        if (configfilename != null) {
            File conffile = new File(configfilename);
            if (conffile.exists()) {
                // ファイルの操作など
                try {
                    XMLConfiguration xml_config = new XMLConfiguration(conffile);

                    for (int i = 0; i < xml_config.getList("EntityDescriptor[@entityID]").size(); i++) {
                        String spEntityId = xml_config.getString("EntityDescriptor("+i+")[@entityID]");

                        if (allow_config.containsKey(spEntityId) == true) {
                            System.out.println("SampleFilterPerSP WARNING: ignore duplicated " + spEntityId
                                               + " because it is already configured");
                            continue;
                        }

                        Map<String, Set> submap = new HashMap<String, Set>();
                        String[] attributeID = xml_config.getStringArray("EntityDescriptor("+i+").Attribute[@attributeID]");
                        String[] attribute   = xml_config.getStringArray("EntityDescriptor("+i+").Attribute");
                        for (int j = 0; j < attributeID.length; j++) {
                            if (submap.containsKey(attributeID[j]) == false) {
                                submap.put(attributeID[j], new HashSet<String>());
                            }
                            submap.get(attributeID[j]).add(attribute[j]);
                        }
                        allow_config.put(spEntityId, submap);
                    }
                } catch (ConfigurationException e) {
                    throw new ServletException(e);
                }
            }

System.out.println("SampleFilterPerSP config-key-value:");
            for(Map.Entry<String, Map> e : allow_config.entrySet()) {
```

```
                    System.out.println("SampleFilterPerSP   "
                                    + e.getKey() + " : " + e.getValue());
                }
            }

// 取得した属性情報
            m_saml2AttrAuth =
                (SAML2AttributeAuthority) filterConfig.getServletContext().getAttribute("shibboleth.SAML2AttributeAuthority");
            // relying-partyの情報
            m_samlRelyingPartyConfMan =
                (SAMLMDRelyingPartyConfigurationManager) filterConfig.getServletContext().getAttribute(
                    "shibboleth.RelyingPartyConfigurationManager");
            }

}
```

SampleFilterPerSP_allow.xml

下記内容のファイルをダウンロードする（リンク上で右クリックしてリンク先を保存してください）

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE EntitiesDescriptor [
    <!ELEMENT EntitiesDescriptor (EntityDescriptor+)>
    <!ELEMENT EntityDescriptor   (Attribute+)>
    <!ELEMENT Attribute          (#PCDATA)>

    <!ATTLIST EntityDescriptor   entityID    CDATA #REQUIRED>
    <!ATTLIST Attribute          attributeID CDATA #REQUIRED>
]>

<EntitiesDescriptor>
  <EntityDescriptor entityID="https://shiken-sp00.nii.ac.jp/shibboleth-sp" >
    <Attribute attributeID="eduPersonAffiliation">
       student
    </Attribute>
    <Attribute attributeID="eduPersonAffiliation">
       member
    </Attribute>
  </EntityDescriptor>

  <EntityDescriptor entityID="https://shiken-sp01.nii.ac.jp/shibboleth-sp" >
    <Attribute attributeID="organizationalUnitName">
       Test Unit1
    </Attribute>
    <Attribute attributeID="eduPersonAffiliation">
       student
    </Attribute>
  </EntityDescriptor>
</EntitiesDescriptor>
```

（注意１：同じ entityID を複数回定義することはできません）
（注意２：複数の Attribute が記述されている場合はOR条件になります）
（注意３：条件に使用できる Attribute は当該SPに送信されるもの（つまり attribute-filter.xml 等でフィルタされていないもの）に限ります）
（注意４：attributeID に記述する属性名は attribute-resolver.xml で定義している名称と一致させなければなりません）

この例では、以下を条件として設定したことになります。詳細はソースコードをご参照ください。

- https://shiken-sp00.nii.ac.jp/shibboleth-sp に対して、属性 eduPersonAffiliation が student もしくは member のどちらかに一致する場合にアサーションを送信します。それ以外の場合はエラーになります。
- https://shiken-sp01.nii.ac.jp/shibboleth-sp に対して、属性 organizationalUnitName が Test Unit1 もしくは属性 eduPersonAffiliation が student のどちらかに一致する場合にアサーションを送信します。それ以外の場合はエラーになります。
- その他のSPに対しては、通常通り属性情報を送信します。