SAML
設定ガイド(PAN-OS 9.1) V1.0
～Shibboleth IdP との連携～

Palo Alto Networks K.K.
2020/8

目次

1. はじめに 3
2. SAML 動作イメージ 4
2.1. ネットワーク構成 4
2.2. SAML 認証フロー 5
3. Shibboleth-IdP の構築 6
3.1. CentOS7 の準備 6
3.2. Shibboleth に必要なソフトウェアのインストール 6
3.3. Tomcat の有効化 6
3.4. Tomcat の設定 6
3.5. Apache の設定 7
3.6. Shibboleth の設定 8
3.7. LDAP サーバとの連携 11
4. SAML 認証の設定 (Captive Portal) 13
4.1. IdP のサーバ証明書と秘密鍵の設定 14
4.2. IdP メタデータの編集 15
4.3. SP (PA Firewall) の設定 16
4.4. SP メタデータを IdP へインポート 24
4.5. SAML 動作確認 26
4.6. User-ID の連携 27
5. SAML 認証のセキュリティ強化設定 31
5.1. SP の署名を IdP が検証する設定 31
5.2. IdP のサーバ証明書の信頼性を検証する設定 35
6. SAML 認証の設定 (GlobalProtect) 40
6.1. 設定ステップ 40
6.2. SP(PA Firewall)の設定 41
6.3. IdP(Shibboleth)側の設定 42
6.4. 動作確認 44
7. その他［参考情報］ 45
7.1. LDAP アトリビュートを SAML Response で取得したい場合 45
7.2. 留意事項 46
8. 参考文献(URL) 48
9. Appendix 48
9.1. LDAP の新規インストール 48
9.2. JavaScript の確認方法 49

1. はじめに

本ガイドにて、PA Firewall での SAML の設定方法をご紹介します。

PA Firewall では、以下の機能を利用する際に SAML 認証を行うことができます。

① Captive Portal 認証
② GlobalProtect の認証
③ マネージメントアクセス時の認証

PA Firewall は SAML SP として動作しますので、SAML IdP は別途準備する必要があります。

本ガイドでは、オープンソースの Shibboleth3 を IdP として構築し、PA Firewall の SAML SP 動作(上記①および②)を確認できるようにしています。

本ガイドを弊社提供の正式ドキュメントと併用して頂き、新規設置作業や日々の運用時の設定変更作業時の参考ドキュメントとしてご活用ください。

※) 本ガイドは、以下の OS バージョンを利用しています。

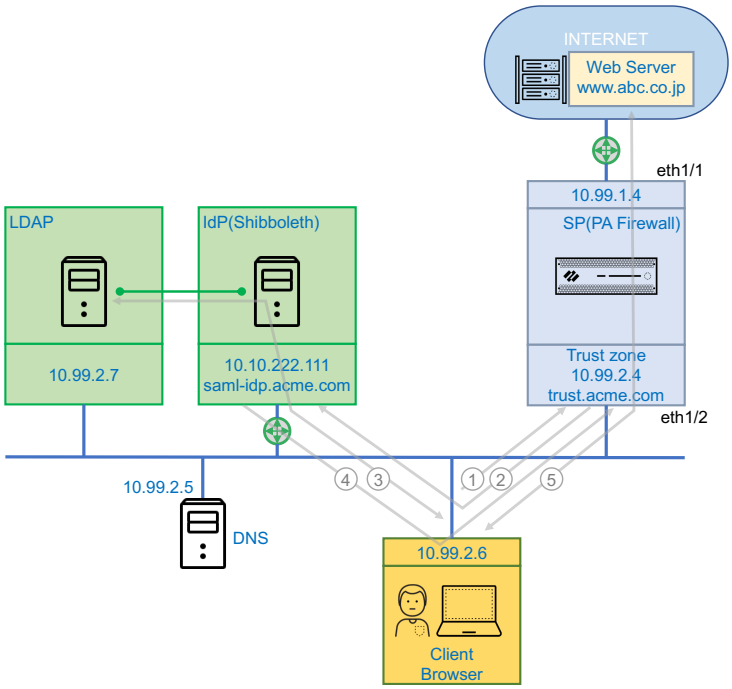
PAN-OS:	9.1.3
GlobalProtect Agent:	5.1.1

適用するバージョンが異なる場合は、該当するバージョンのドキュメントを参照してください。

2. SAML 動作イメージ

2.1. ネットワーク構成

本ガイドでは、以下のようなネットワーク構成で、SAML 認証の動作を確認します。



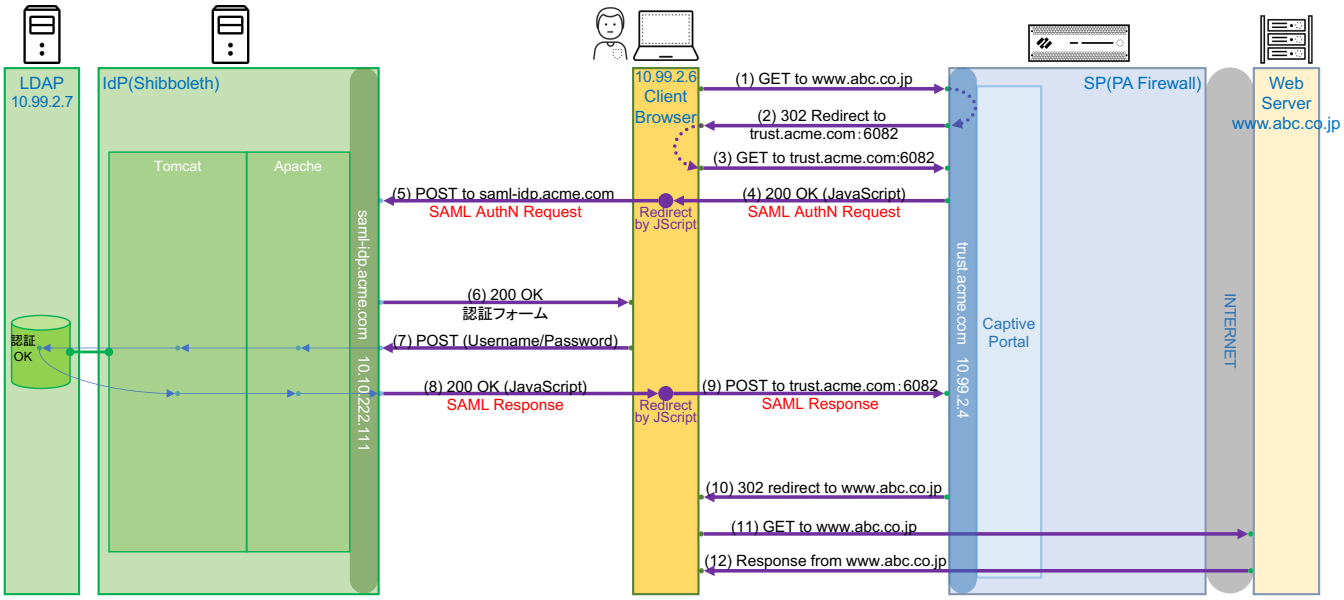
動作概要:

① クライアントが Web ブラウザで「www.abc.co.jp」へアクセスを試みる。
② SAML の認証リクエスト(AuthN Request)が、Web ブラウザを経由して IdP へ送られる。
③ IdP から認証フォームが送られてくるので、クライアントは Username と Password を入力する。
④ IdP での認証が成功すると、SAML Response(Assertion)が Web ブラウザを経由して SP へ送られる。
⑤ 「www.abc.co.jp」へアクセスできる。

- IdP(Shibboleth)のバージョンは、「3.4.6」を利用します。
- クライアントの Web ブラウザは、「Chrome」を利用します。

2.2. SAML 認証フロー

以下は、フローの詳細です(Captive Portal の場合)。



(1) クライアントの Web ブラウザで、「www.abc.co.jp」へアクセス。
(2) PA Firewall の Captive Portal 設定により、「https://trust.acme.com:6082」への 302 リダイレクトが返される。
(3) クライアント PC の Web ブラウザは、「https://trust.acme.com:6082」へリダイレクトアクセス。
(4) SAML 認証の設定が行われた Captive Portal が、SAML 認証リクエスト(AuthN Request)を返答する。
(5) それを受信した Web ブラウザでは、JavaScript により、その SAML AuthN Request を「saml-idp.acme.com」(＝Shibboleth-IdP)へ POST する。
(6) Shibboleth-IdP は、クライアントの Web ブラウザへ認証フォームを返答する。
(7) クライアントはその認証フォームに Username と Password を入力する。
(8) 認証が成功すれば、Shibboleth-IdP は、SAML Response を返答する。
(9) それを受信した Web ブラウザは、JavaScript により、その SAML Response を「trust.acme.com:6082」へ送信する。
(10) SAML Response を受信した SP である PA Firewall は、(1)で受信した「www.abc.co.jp」を、302 リダイレクトで返答する。
(11) Web ブラウザは「www.abc.co.jp」へアクセス。
(12) 「www.abc.co.jp」からの HTTP レスポンスが得られる。

3. Shibboleth-IdP の構築

Shibboleth を SAML IdP として利用しますので、そのインストールを行います。

3.1. CentOS7 の準備

Shibboleth をインストールするサーバーとして、CentOS7 を準備します。

- メモリ:3GB 以上が推奨とされています。
- LDAP サーバは、別途構築済みの前提です。

3.2. Shibboleth に必要なソフトウェアのインストール

CentOS7 に以下をインストールします。

```
# yum -y install httpd
# yum -y install mod_ssl

# yum -y install java-1.8.0-openjdk

# yum -y install tomcat
```

3.3. Tomcat の有効化

```
# systemctl enable tomcat
```

3.4. Tomcat の設定

3.4.1. Tomcat の環境変数設定

(1) 以下の内容で、Tomcat の環境変数設定ファイルを新規に作ります。

```
# vi /etc/profile.d/java-tomcat.sh
```

```
# /etc/profile.d/java-tomcat.sh
JAVA_HOME=/usr/lib/jvm/jre
#export MANPATH=$MANPATH:/usr/java/default/man
CATALINA_HOME=/usr/share/tomcat
CATALINA_BASE=$CATALINA_HOME
PATH=$JAVA_HOME/bin:$CATALINA_BASE/bin:$CATALINA_HOME/bin:$PATH
export PATH JAVA_HOME CATALINA_HOME CATALINA_BASE
```

(2) 環境変数を読み込みます。

```
# source /etc/profile
```

3.4.2. Tomcat の起動オプションの変更

以下のように編集します。

```
# vi /etc/sysconfig/tomcat
```

```
#JAVA_OPTS="-Xminf0.1 -Xmaxf0.3"
JAVA_OPTS="-server -Xmx1500m -XX:MaxPermSize=256m -XX:+UseG1GC "
```

3.4.3. Tomcat の server.xml 設定

```
# vi /usr/share/tomcat/conf/server.xml
```

以下をコメントアウトします。

```
<!--      コメントアウト
 <Connector port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443" />
-->       コメントアウト
```

以下の行をコメントアウトし、コピーして、以下のように書き換えます。

```
<!--      コメントアウト
<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />
-->       コメントアウト


<Connector port="8009" protocol="AJP/1.3" redirectPort="8443"
           enableLookups="false" tomcatAuthentication="false" address="127.0.0.1"
maxPostSize="100000" />
```

3.5. Apache の設定

Web サーバである Apache の設定を行います。

3.5.1. httpd.conf の設定

```
# vi /etc/httpd/conf/httpd.conf
```

```
ServerName saml-idp.acme.com:80
ProxyPass /idp/ ajp://localhost:8009/idp/
```

3.5.2. ssl.conf の設定

```
# vi /etc/httpd/conf.d/ssl.conf
```

```
ServerName saml-idp.acme.com:443
ProxyPass /idp/ ajp://localhost:8009/idp/
```

3.5.3. httpd サービスの再起動

httpd を再起動します。

```
# systemctl restart httpd
```

3.6. Shibboleth の設定

3.6.1. Shibboleth3 のダウンロード

Web ブラウザで「http://shibboleth.net/downloads/identity-provider/latest3」へアクセスし、最新バージョンを確認してください。
最新版の Shibboleth3(本資料作成時点の最新：version 3.4.6)を以下のコマンドでダウンロードします。

```
# wget http://shibboleth.net/downloads/identity-provider/latest3/shibboleth-identity-provider-3.4.6.tar.gz
```

3.6.2. Shibboleth3 の展開

ダウンロードした shibboleth3 のアーカイブを展開します。

```
# tar xzvf shibboleth-identity-provider-3.4.6.tar.gz
```

3.6.3. インストールスクリプトの実行

展開したディレクトリに移動し、インストールスクリプトを実行します。

```
# cd shibboleth-identity-provider-3.4.6

# ./bin/install.sh -Didp.conf.filemode=640
```

```
Source (Distribution) Directory (press <enter> to accept default): [/home/admin-admin/shibboleth-identity-provider-3.4.6]
(Enter)

Installation Directory: [/opt/shibboleth-idp]
(Enter)

Hostname: [saml-idp.acme.com]
(ホスト名を IdP で使う FQDN に変更して Enter)

SAML EntityID: [https://saml-idp.acme.com/idp/shibboleth]
(Enter)

Attribute Scope: [acme.com]
(ドメインを指定して Enter)

Backchannel PKCS12 Password: (任意)
Re-enter password: (任意)
Cookie Encryption Key Password: (任意)
Re-enter password: (任意)
Warning: /opt/shibboleth-idp/bin does not exist.
Warning: /opt/shibboleth-idp/edit-webapp does not exist.
Warning: /opt/shibboleth-idp/dist does not exist.
Warning: /opt/shibboleth-idp/doc does not exist.
Warning: /opt/shibboleth-idp/system does not exist.
Generating Signing Key, CN = saml-idp.acme.com URI = https://saml-idp.acme.com/idp/shibboleth ...
...done
Creating Encryption Key, CN = saml-idp.acme.com URI = https://saml-idp.acme.com/idp/shibboleth ...
...done
Creating Backchannel keystore, CN = saml-idp.acme.com URI = https://saml-idp.acme.com/idp/shibboleth ...
...done
Creating cookie encryption key files...
...done
Rebuilding /opt/shibboleth-idp/war/idp.war ...
...done

BUILD SUCCESSFUL
Total time: 1 minute 0 seconds
```

3.6.4. パーミッションおよびオーナー権限の変更

以下のように、ディレクトリのパーミッションおよびオーナー権限を変更します。

```
# chown -R tomcat:tomcat /opt/shibboleth-idp/logs/
# chgrp -R tomcat /opt/shibboleth-idp/conf/
# chmod -R g+r /opt/shibboleth-idp/conf/
# find /opt/shibboleth-idp/conf -type d -exec chmod -R g+s {} ¥;
# chgrp tomcat /opt/shibboleth-idp/metadata/
# chmod g+w /opt/shibboleth-idp/metadata/
# chmod +t /opt/shibboleth-idp/metadata/
```

3.6.5. idp.war の設定

war ファイルとは Java 製の Web アプリケーションで使用する servlet や設定ファイル、JAR 形式のライブラリがまとめられているアーカイブファイルです。
Shibboleth が利用している Tomcat は Java で動いているので、この作業が必要です。

(1) idp.war ファイルのビルドに必要なパッケージをインストールします。

```
# yum -y install jakarta-taglibs-standard
```

(2) パッケージに含まれる必要な jar ファイルのシンボリックリンクを作成します。

```
# ln -s /usr/share/java/jakarta-taglibs-core.jar /opt/shibboleth-idp/edit-webapp/WEB-INF/lib/jakarta-taglibs-core.jar

# ln -s /usr/share/java/jakarta-taglibs-standard.jar /opt/shibboleth-idp/edit-webapp/WEB-INF/lib/jakarta-taglibs-standard.jar
```

(3) Idp.war の Build を行います。

```
# /opt/shibboleth-idp/bin/build.sh
```

```
Installation Directory: [/opt/shibboleth-idp]
(Enter)

Rebuilding /opt/shibboleth-idp/war/idp.war ...
...done

BUILD SUCCESSFUL
Total time: 6 seconds
```

(4) Tomcat に idp.war の定義を追加します。以下の新規ファイルを生成します。

```
# vi /usr/share/tomcat/conf/Catalina/localhost/idp.xml
```

```
<Context docBase="/opt/shibboleth-idp/war/idp.war"
        privileged="true"
        antiResourceLocking="false"
        swallowOutput="true">

    <Manager pathname="" />

    <!-- For Tomcat 8.0: work around lack of Max-Age support in IE/Edge -->
    <CookieProcessor alwaysAddExpires="true" />

</Context>
```

3.6.6. アクセス許可の設定

Shibboleth へのアクセスを許可したい送信元サブネットを指定します。
本ガイドでは、10.0.0.0/8 を許可することにします。

```
# vi /opt/shibboleth-idp/conf/access-control.xml
```

```
        <entry key="AccessByIPAddress">
            <bean id="AccessByIPAddress" parent="shibboleth.IPRangeAccessControl"
                p:allowedRanges="#{ {'127.0.0.1/32', '::1/128', '10.0.0.0/8'} }" />
        </entry>
```

3.6.7. Tomcat サービスの開始

(1) Tomcat を開始します。

```
# systemctl start tomcat
```

(2) Tomcat のログにエラーが出ていないことを確認します (ログファイルの YYYYMMDD は日付」)。

```
# tail /usr/share/tomcat/logs/catalina.YYYY-MM-DD.log
```

3.6.8. 動作確認

アクセス許可を設定した 10.0.0.0/8 サブネットの端末の Web ブラウザから、下記へアクセスします。

https://saml-idp.acme.com/idp/status

以下のようなレスポンスが得られれば、Shibboleth は動作しています。

```
### Operating Environment Information
operating_system: Linux
operating_system_version: 3.10.0-1127.10.1.el7.x86_64
operating_system_architecture: amd64
jdk_version: 1.8.0_252
available_cores: 1
used_memory: 186 MB
maximum_memory: 1500 MB

### Identity Provider Information
idp_version: 3.4.6
start_time: 2020-06-13T16:48:14+09:00
current_time: 2020-06-13T16:48:16+09:00
uptime: 1823 ms

service: shibboleth.LoggingService
last successful reload attempt: 2020-06-13T07:48:00Z
last reload attempt: 2020-06-13T07:48:00Z

service: shibboleth.ReloadableAccessControlService
last successful reload attempt: 2020-06-13T07:48:03Z
last reload attempt: 2020-06-13T07:48:03Z

～省略～
```

3.7. LDAP サーバとの連携

Shibboleth と LDAP サーバ間の連携のための設定を行います。

(LDAP 環境を新規に構築する場合には、Appendix の URL を参照して LDAP をインストールしてください。)

3.7.1. Openldap クライアントのインストール (接続テストのため)

(1) CentOS7 に Openldap クライアントをインストールします。

```
# yum install -y openldap-clients
```

(2) LDAP サーバから情報が取得できることを確認します。

```
# ldapsearch -h 10.99.2.7 -x -b 'dc=acme,dc=com' '(objectclass=*)'
```

以下は、上記のコマンドで得られた LDAP 情報のサンプルです。

```
# extended LDIF
#
# LDAPv3
# base <dc=acme,dc=com> with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#

# acme.com
dn: dc=acme,dc=com
objectClass: top
objectClass: dcObject
objectClass: organization
o: Server World
dc: Acme
(省略)
```

3.7.2. Shibboleth の LDAP との接続設定

Shibboleth が LDAP サーバと接続するための設定を行います。
(以下はサンプルです。ご自身の LDAP 環境にあった設定を行ってください。)

```
# vi /opt/shibboleth-idp/conf/ldap.properties
```

```
## Connection properties ##
#idp.authn.LDAP.ldapURL                    = ldap://localhost:10389
idp.authn.LDAP.ldapURL                     = ldap://10.99.2.7:389
#idp.authn.LDAP.useStartTLS                = true
idp.authn.LDAP.useStartTLS                 = false
#idp.authn.LDAP.useSSL                     = false
# Time in milliseconds that connects will block
#idp.authn.LDAP.connectTimeout              = PT3S
# Time in milliseconds to wait for responses
#idp.authn.LDAP.responseTimeout             = PT3S

#idp.authn.LDAP.baseDN                      = ou=people,dc=example,dc=org
idp.authn.LDAP.baseDN                      = ou=people,dc=acme,dc=com
#idp.authn.LDAP.subtreeSearch               = false
idp.authn.LDAP.subtreeSearch               = true
idp.authn.LDAP.userFilter                  = (uid={user})
# bind search configuration
# for AD: idp.authn.LDAP.bindDN=adminuser@domain.com
#idp.authn.LDAP.bindDN                      = uid=myservice,ou=system
idp.authn.LDAP.bindDN                      = cn=Manager,dc=acme,dc=com
#idp.authn.LDAP.bindDNCredential            = myServicePassword
idp.authn.LDAP.bindDNCredential            = LDAP バインド用のパスワード
```

3.7.3. Tomcat サービスの再起動

(1) Tomcat を再起動します。

```
# systemctl restart tomcat
```

(2) Tomcat のログにエラーが出ていないことを確認します。

```
# tail /usr/share/tomcat/logs/catalina.YYYY-MM-DD.log
```

(3) 以下のコマンドで、動作を確認します。

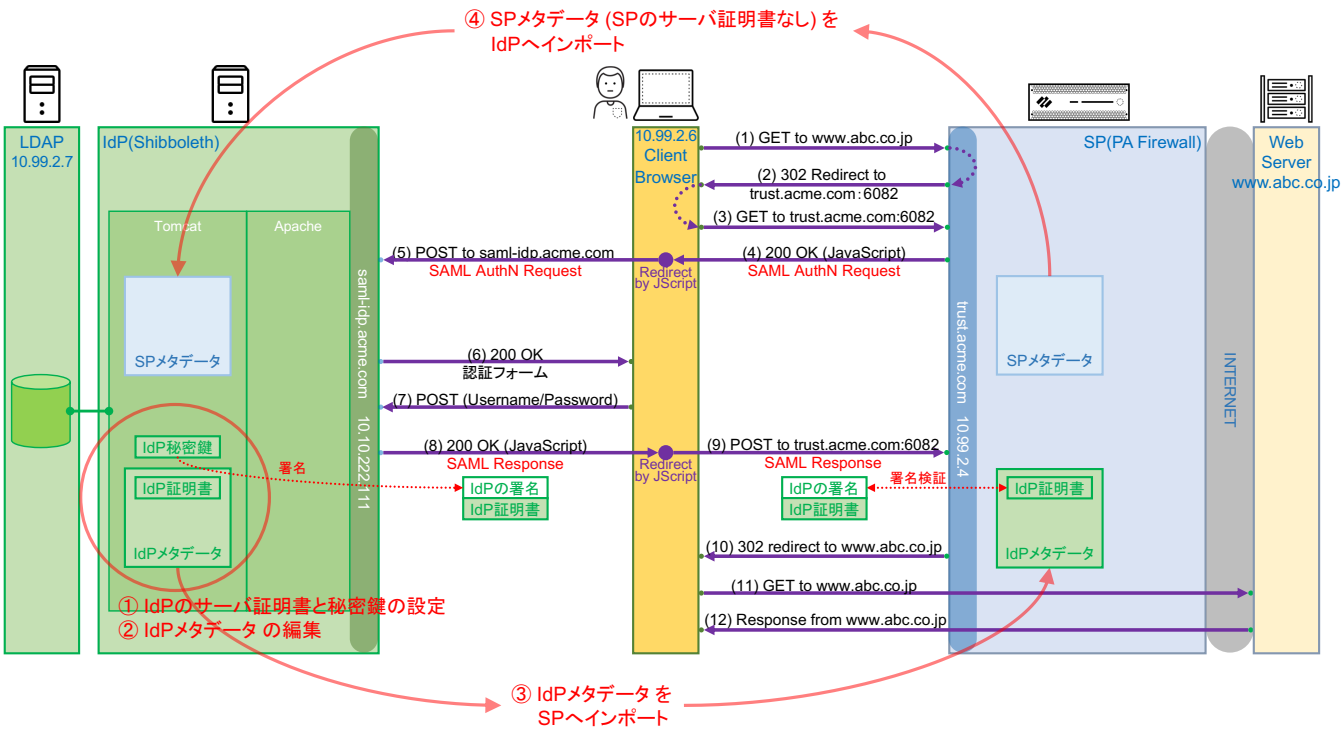
```
# wget -O - https://localhost/idp/status --no-check-certificate
```

以下のような返答が得られれば OK です。

```
(前略)
### Operating Environment Information
operating_system: Linux
operating_system_version: 3.10.0-1127.10.1.el7.x86_64
operating_system_architecture: amd64
jdk_version: 1.8.0_252
available_cores: 1
used_memory: 152 MB
maximum_memory: 1500 MB

### Identity Provider Information
idp_version: 3.4.6
start_time: 2020-06-13T17:19:47+09:00
current_time: 2020-06-13T17:25:37+09:00
uptime: 349706 ms

service: shibboleth.LoggingService
last successful reload attempt: 2020-06-13T08:19:21Z
last reload attempt: 2020-06-13T08:19:21Z

service: shibboleth.ReloadableAccessControlService
last successful reload attempt: 2020-06-13T08:19:25Z
last reload attempt: 2020-06-13T08:19:25Z

service: shibboleth.MetadataResolverService
last successful reload attempt: 2020-06-13T08:19:24Z
last reload attempt: 2020-06-13T08:19:24Z

service: shibboleth.RelyingPartyResolverService
last successful reload attempt: 2020-06-13T08:19:24Z
last reload attempt: 2020-06-13T08:19:24Z

service: shibboleth.NameIdentifierGenerationService
last successful reload attempt: 2020-06-13T08:19:24Z
last reload attempt: 2020-06-13T08:19:24Z

service: shibboleth.AttributeResolverService
last successful reload attempt: 2020-06-13T08:19:23Z
last reload attempt: 2020-06-13T08:19:23Z

      DataConnector staticAttributes: has never failed

service: shibboleth.AttributeFilterService
last successful reload attempt: 2020-06-13T08:19:23Z
last reload attempt: 2020-06-13T08:19:23Z

100%[===================================================================================>] 1,431      -
-.-K/s 時間 0s

2020-06-13 17:25:37 (111 MB/s) - stdout へ出力完了 [1431/1431]
```

4. SAML 認証の設定 (Captive Portal)

まずは、「IdP の署名」を SP が検証するだけの設定(「SP の署名」を IdP が検証しない設定)を実施していきます。



① IdP のサーバ証明書と秘密鍵の設定

どのサーバ秘密鍵で「IdP の署名」を行うのか、どの「IdP 証明書」を SAML Response で SP に送るのかを指定します。

② IdP メタデータの編集

Shibboleth インストール時に自動的に生成される IdP メタデータを、SP(PA)でインポートして利用できるように編集します。

③ IdP メタデータを SP へインポート

IdP メタデータを、SP(PA)にインポートします。(結果、SML IdP Server Profile が生成されます。)

④ SP メタデータを IdP へインポート

SP(PA Firewall)で生成されるメタデータを IdP へインポートします。
まずは SP のサーバ証明書を持たない状態(シンプルな状態)での確認を行います。

4.1. IdP のサーバ証明書と秘密鍵の設定

「IdP の署名」をどのサーバ秘密鍵で行い、どの「IdP 証明書」を SAML Response で SP へ送るのかを指定します。

4.1.1. サーバ秘密鍵のオーナー権限及びパーミッションの変更

この段階では簡易的に、デフォルトで存在するサーバ証明書と秘密鍵を使うことにします。

サーバ秘密鍵のオーナー権限を変更しておきます。主な理由は Shibboleth-IdP のエラーを回避するためです。
合わせてパーミッションも Read-Only に変更しておきます。

/opt/shibboleth-idp/credentials ディレクトリに、以下 3 種の組合せが存在します。

① idp-backchannel.crt idp-backchannel.p12
② idp-signing.crt idp-signing.key
③ idp-encryption.crt idp-encryption.key

このうち、②を使って、IdP の署名を行います。

(1) 以下のコマンドを実行し、idp-signing.key のオーナー権限とパーミッションを変更します。

```
# chown tomcat:root /opt/shibboleth-idp/credentials/idp-signing.key
# chmod 400 /opt/shibboleth-idp/credentials/idp-signing.key
```

(2) idp-encryption.key は、SAML アサーションの暗号化のために使われるものですが、PA Firewall が対応していないので、実質利用しません。
しかし、Shibboleth の設定としては必須であり、オーナー権限を変更しておかないとエラーが出るため、こちらも変更しておきます。

```
# chown tomcat:root /opt/shibboleth-idp/credentials/idp-encryption.key
# chmod 400 /opt/shibboleth-idp/credentials/idp-encryption.key
```

4.1.2. idp.propereties の設定(確認のみ)

idp.properties.xml の設定で、IdP の署名をどのサーバ秘密鍵で行うか、どのサーバ証明書を SAML Response で SP へ送るのかを指定します。

この段階では、デフォルトで存在する「idp-signing.key」で IdP の署名を行うことにします。

以下の状態になっていることを確認します。

```
# less /opt/shibboleth-idp/conf/idp.properties
```

```
# Settings for public/private signing and encryption key(s)
# During decryption key rollover, point the ".2" properties at a second
# keypair, uncomment in credentials.xml, then publish it in your metadata.
idp.signing.key = %{idp.home}/credentials/idp-signing.key
idp.signing.cert = %{idp.home}/credentials/idp-signing.crt
idp.encryption.key = %{idp.home}/credentials/idp-encryption.key
idp.encryption.cert = %{idp.home}/credentials/idp-encryption.crt
```

- idp-encryption.key は使いませんが、Shibboleth の設定としては必須となっていますので、このまま有効にしておきます。

4.2. IdP メタデータの編集

Shibboleth インストール時に生成されている IdP メタデータをコピーして編集します。

ファイル名： /opt/shibboleth-idp/metadata/idp-metadata.xml

後のステップで、この編集後のファイルを SP(PA Firewall)にインポートします。

このファイルを開くと、/opt/shibboleth-idp/credentials ディレクトリにデフォルトで存在する 3 つのサーバ証明書が以下の順番で連なっている箇所が二箇所あります。この部分を編集します。

① idp-backchannel.crt ←削除します。
② idp-signing.crt ←これを残して、これを使います。
③ idp-encryption.crt

(1) 一番上の証明書(idp-backchannel.crt)は使わないので、削除してください。
(実質、3 番目(idp-encryption.crt)も必要ありませんので、削除しても構いませんし、残っていても支障ありません。)
(2) 二箇所とも同様の編集を行なってください。

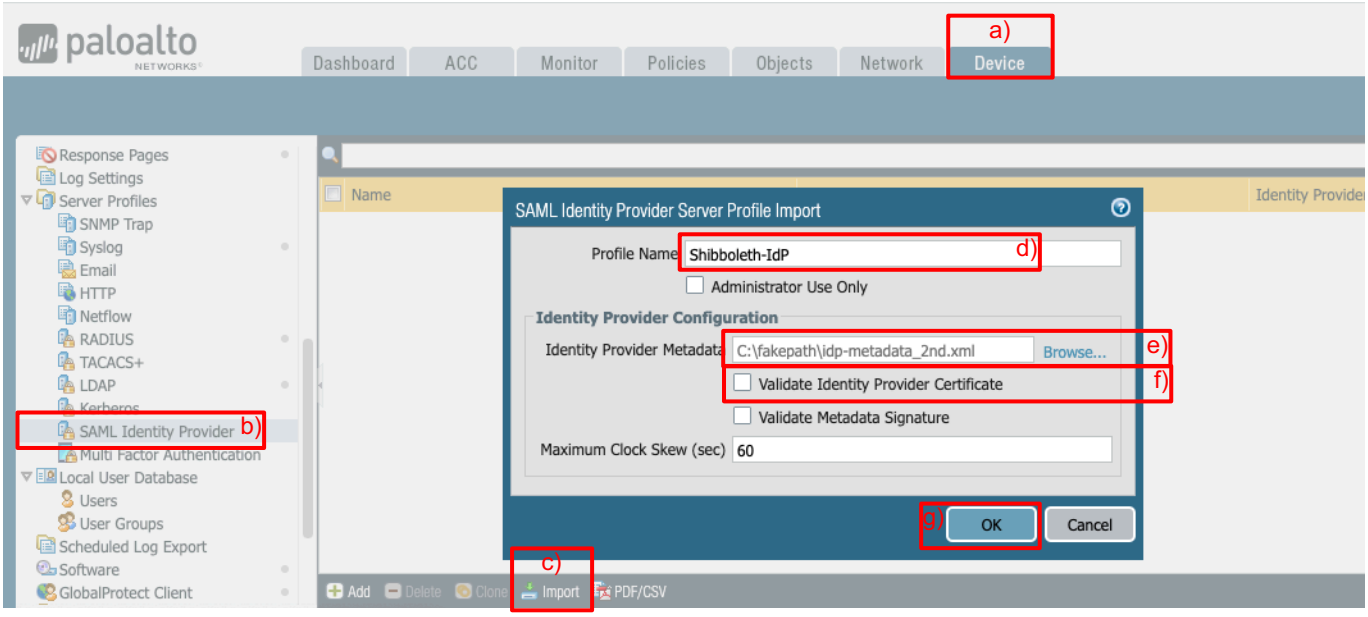
```
～略～
        <!-- First signing certificate is BackChannel, the Second is FrontChannel -->
        <KeyDescriptor use="signing">                                   ↓削除
            <ds:KeyInfo>
                <ds:X509Data>
                    <ds:X509Certificate>
MIIELzCCApegAwIBAgIUQOlwOaeZZa4CrxfVBoruoVTPuWgwDQYJKoZIhvcNAQEL
BQAwHDEaMBgGA1UEAwwRc2PtbC1pZHAuYWNtZS5jb20wHhcNMjAwNjEzMDc0MDAz
～略～
        ① idp-backchannel.crt
～略～
SPHd35zcqp3cDDVFPB/qctUPm+vDfMHsdKZmchp88UfjKCb7kFjP224veXkGrNj2
Llk4d8yQjLnHz929w7v4KsWuzA==
                    </ds:X509Certificate>
                </ds:X509Data>
            </ds:KeyInfo>

        </KeyDescriptor>                                                ↑削除
        <KeyDescriptor use="signing">
            <ds:KeyInfo>
                <ds:X509Data>
                    <ds:X509Certificate>
MIIEMDCCApigAwIBAgIVANqMoMaTeVPML6h5+erQj9Lpjxg6MA0GCSqGSIb3DQEB
CwUAMBwxGjAYBgNVBAMMEXNhbWwtaWRwLmFjbWUuY29tMB4XDTIwMDYxMzA3NDAw
～略～
        ② idp-signing.crt
～略～
DwUxLYLXjW3ZDSskYGaFDwc9hQEoh4mUerBM0GV3ZvAsvct9tDuEvbPsXO+vbqN4
2EKmR98g6kCPiEGXgABN+BDjVwE=
                    </ds:X509Certificate>
                </ds:X509Data>
            </ds:KeyInfo>

        </KeyDescriptor>
        <KeyDescriptor use="encryption">
            <ds:KeyInfo>
                <ds:X509Data>
                    <ds:X509Certificate>
MIIEMDCCApigAwIBAgIVAIynJUkg27pGN0yUyt1iOw+tXn21MA0GCSqGSIb3DQEB
CwUAMBwxGjAYBgNVBAMMEXNhbWwtaWRwLmFjbWUuY29tMB4XDTIwMDYxMzA3NDAw
～略～
        ③ idp-encryption.crt
～略～
Uee+q1x6BvqwqIEyI06RphOygZMgY1hWfgpOVKMQa3RWKHhAosdFYGFm/MWFvLdY
RZ1WOjSG/ivZ2ZwY5+/q/0WR6gw=
                    </ds:X509Certificate>
                </ds:X509Data>
            </ds:KeyInfo>

        </KeyDescriptor>
～略～
(2 箇所目も同様に修正)
```

4.3. SP (PA Firewall) の設定

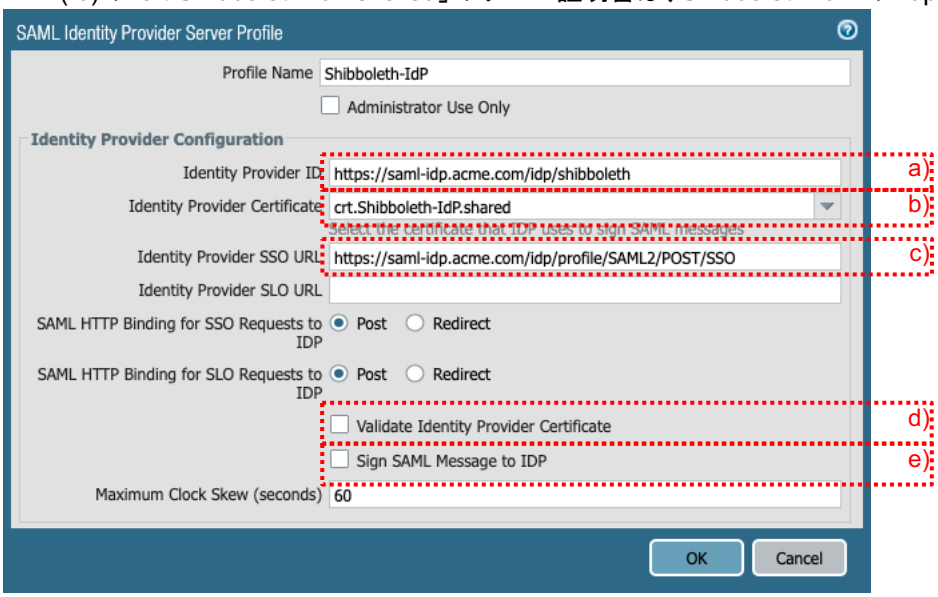
4.3.1. IdP メタデータを SP へインポート (＝SAML IdP サーバプロファイルの設定)

前のステップで編集した IdP メタデータをインポートすることで、SAML IdP サーバプロファイルが生成されます。

(1) a)「Device」 → b)「SMAL Identity Provider」 → c)「Import」をクリックします。
(2) d)Profile Name に「Shibboleth-IdP(任意)」、e)で IdP メタデータファイルを選択します。
今の段階では、f)「Validate Identity Provider Certificate」のチェックは外しておきます。
g)「OK」をクリックします。



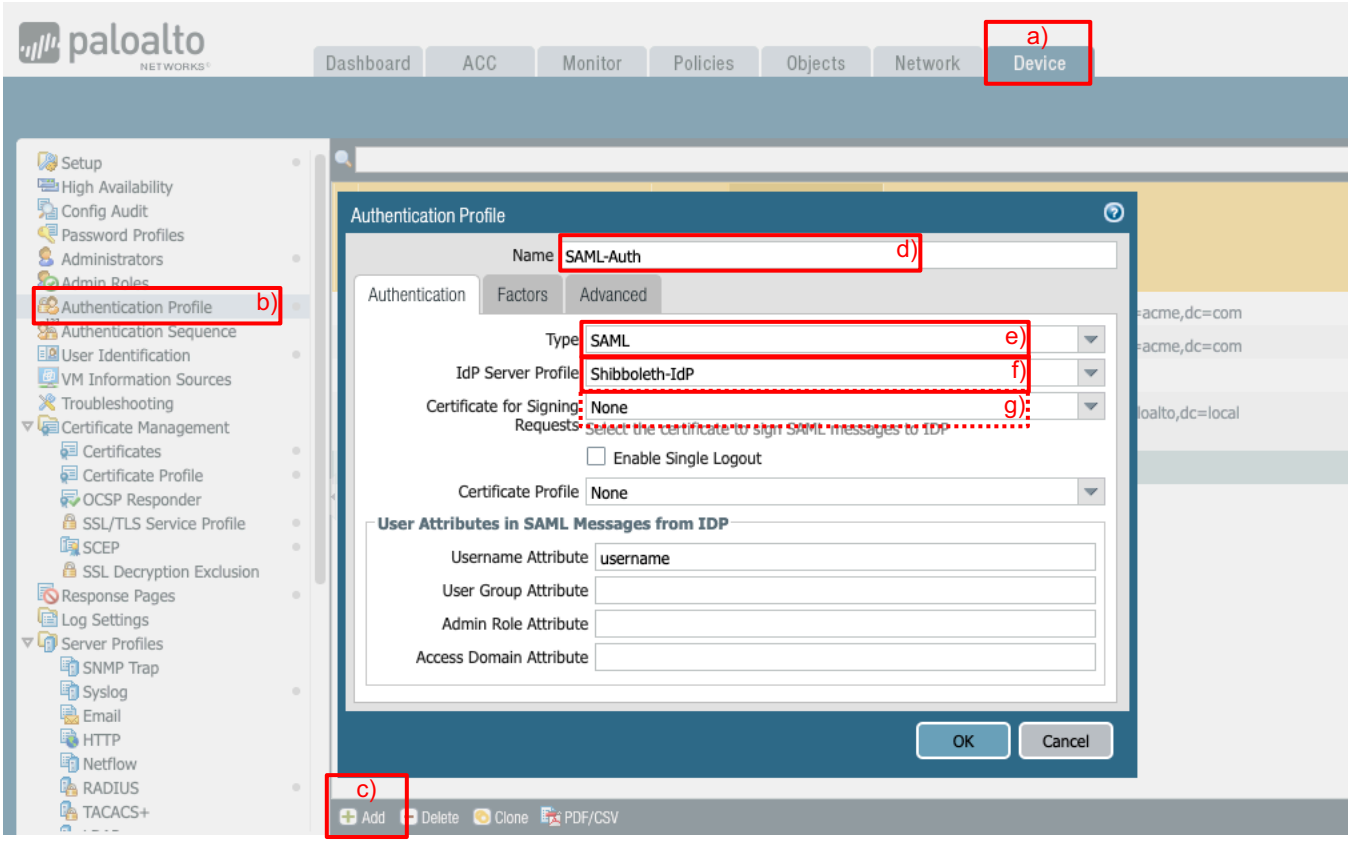
(3) (確認のみ): 「Shibboleth-IdP」を開いてみると、IdP メタデータ内の情報が適用され、以下のような状態になっているはずです。
(b)の「crt.Shibboleth-IdP.shared」のサーバ証明書は、Shibboleth-IdP の「idp-signing.crt」です。)



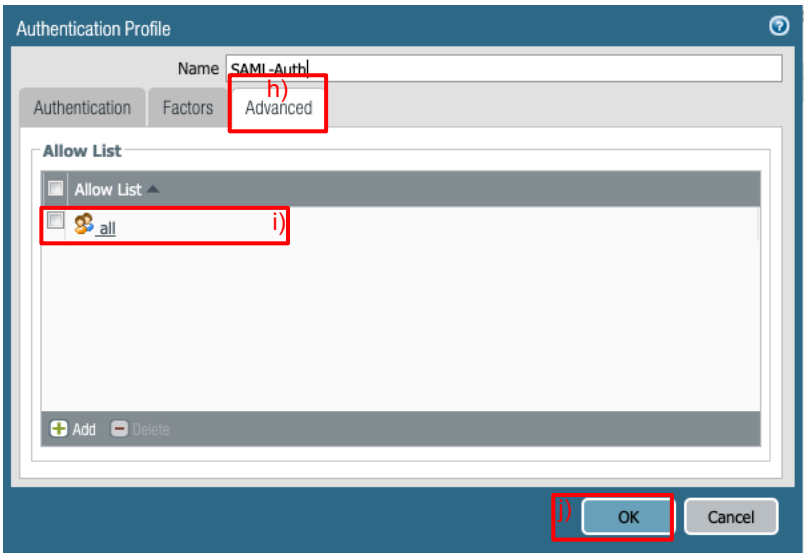
4.3.2. 認証プロファイルの設定

PA Firewall が SAML SP として動作するためには、認証プロファイルの設定が必要です。

(1) a)「Device」 → b)「Authentication Profile」 → c)「Add」をクリックします。
(2) d)Name に「SAML-Auth(任意)」、e)Type で「SAML」を選択し、f)IdP Server Profile には、設定した「Shibboleth-IdP」を選択します。
今の段階では、g)Certificate for signing Requests(「SP の署名」)は「None」のままにしておきます。
(後の章で有効にして、動作を確認します。)



(3) h)「Advanced」タブの Allow List では、i)「all」を選択し、j)「OK」をクリックします。



4.3.3.　Captive Portal の設定

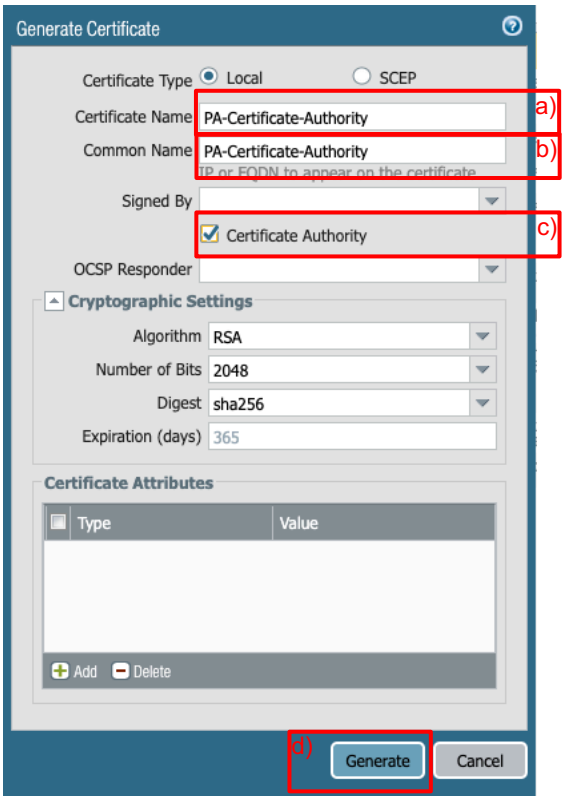
SAML に対応した認証は、Management、Captive Portal、GlobalProtect の 3 つがありますが、ここでは Captive Portal の設定を行っていきます。

4.3.3.1.　CA 証明書の生成

Captive Portal 認証では SSL が利用されるため、サーバ証明書が必要となります。
外部の証明書発行機関から提供されるサーバ証明書でも良いですが、本ガイドでは簡易的に PA で生成したものを利用することにします。

(1)　「Device」 →「Certificate Management」の下の「Certificates」 →「Device Certificates」タブ→「Generate」をクリックします。

(2)　a)Certificate Name に「PA-Certificate-Authority(任意)」と入力します。
b)Common Name にも「PA-Certificate-Authority(任意)」と入力します。
c)Certificate Authority にチェックを入れます。
d)「Generate」をクリックします。

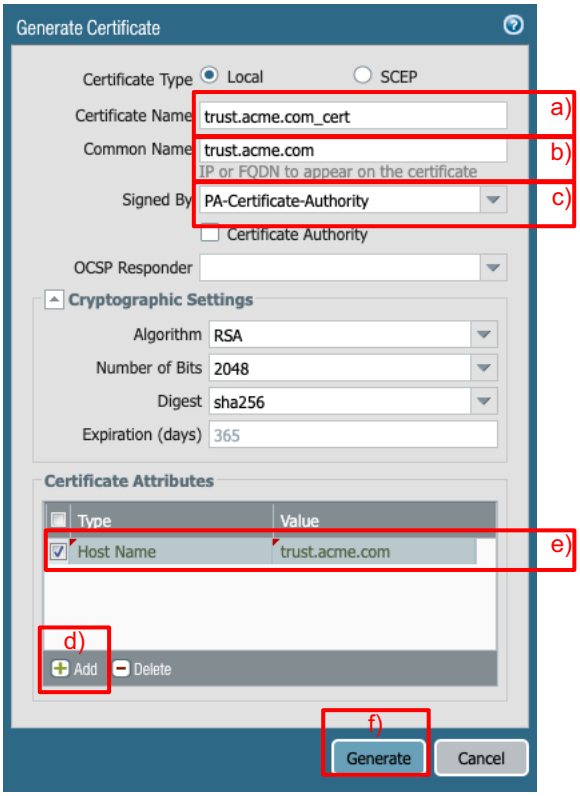


(3)　生成が成功したことを示すメッセージが出力されたら、「OK」をクリックします。

4.3.3.2. Captive Portal 用のサーバー証明書の生成

生成した認証局証明書を使って、Captive Portal が使うサーバー証明書を生成します。

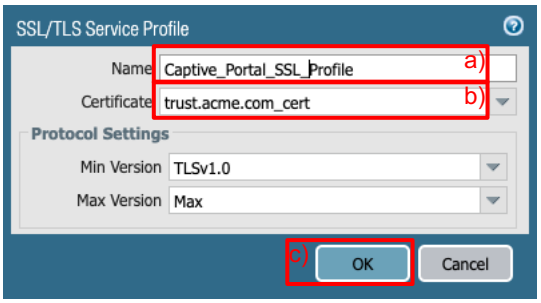
(1) a) Certificate Name に「trust.acme.com_cert (任意)」と入力します。
b) Common Name は、ネットワーク構成に従い、「trust.acme.com」と入力します。
c) Signed by は生成した認証局「PA-Certificate-Authority」を選択します。
d) 「add」をクリックして、e)「Host Name」を選択し、「trust.acme.com」を入力します。
f) 「Generate」をクリックします。



4.3.3.3. SSL/TLS サービスプロファイルの設定

SSL / TLS サービスプロファイルを使用して、Captive Portal で使用するサーバー証明書の指定と、許可する SSL/TLS のプロトコルバージョンを指定することができます。
本ガイドでは、SSL/TLS プロトコルバージョンはデフォルトのままとします。

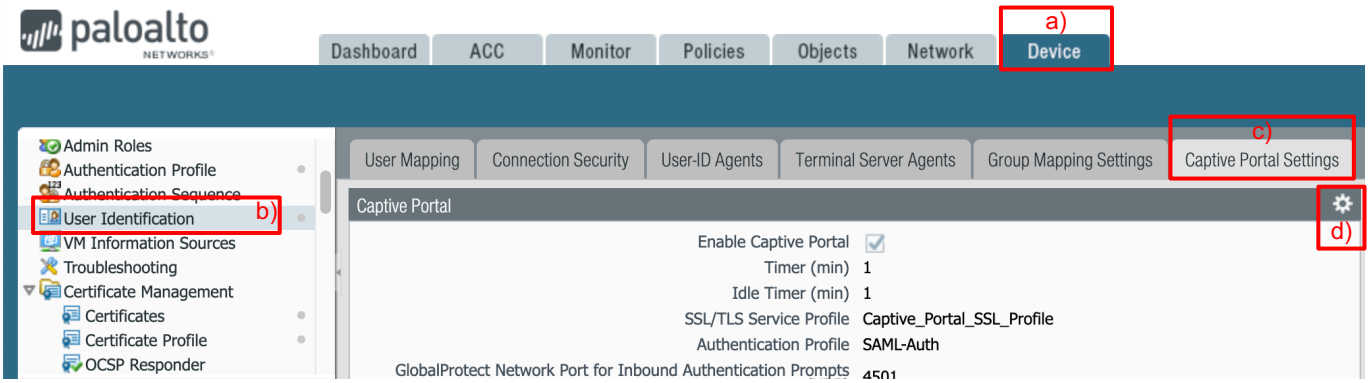
(1) 「Device」 → 「Certificate Management」の下の「SSL/TLS Service Profile」 → 「Add」をクリックします。
(2) Captive Portal 用の SSL/TLS サービスプロファイルを生成します。
a)Name に「Captive_Portal_SSL _Profile(任意)」と入力します。
b)Certificate は作成済みの「trust.acme.com_cert」を選択します。
c)「OK」をクリックします。



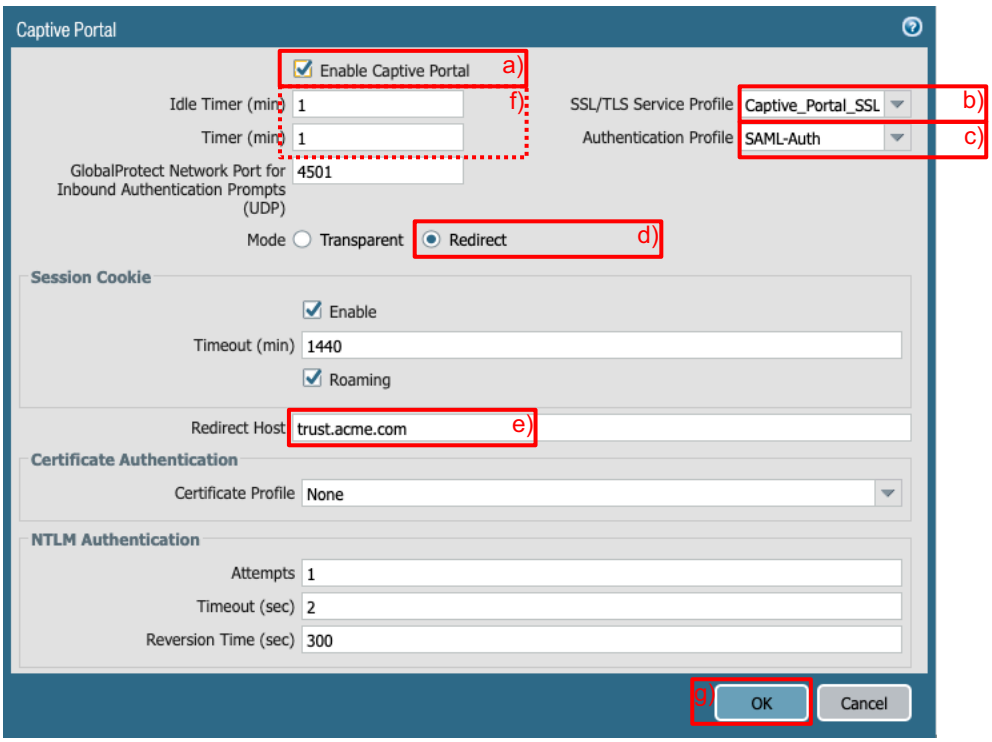
4.3.3.4. Captive Portal 動作の設定

設定した証明書や認証プロファイルを使って、Captive Portal が使えるように設定します。

(1) a)「Device」 → b)「User identification」 → c)「Captive Portal Settings」 → d)「歯車」をクリックします。



(2) a)「Enable Captive Portal」にチェックを入れます。
b)設定した SSL/TLS Service Profile: 「Captive_Portal_SSL_Profile」を選択します。
c)設定した Authentication Profile: 「SAML-Auth」を選択します。
d)Mode は「Redirect」を選択します。
e)の Redirect Host は、PA の Trust 側 IP に紐づいた「trust.acme.com」を入力します。
f)は任意ですが、最小の 1 分にしておくことで、認証を繰り返すテストをする場合には便利です。
g)「OK」をクリックします。

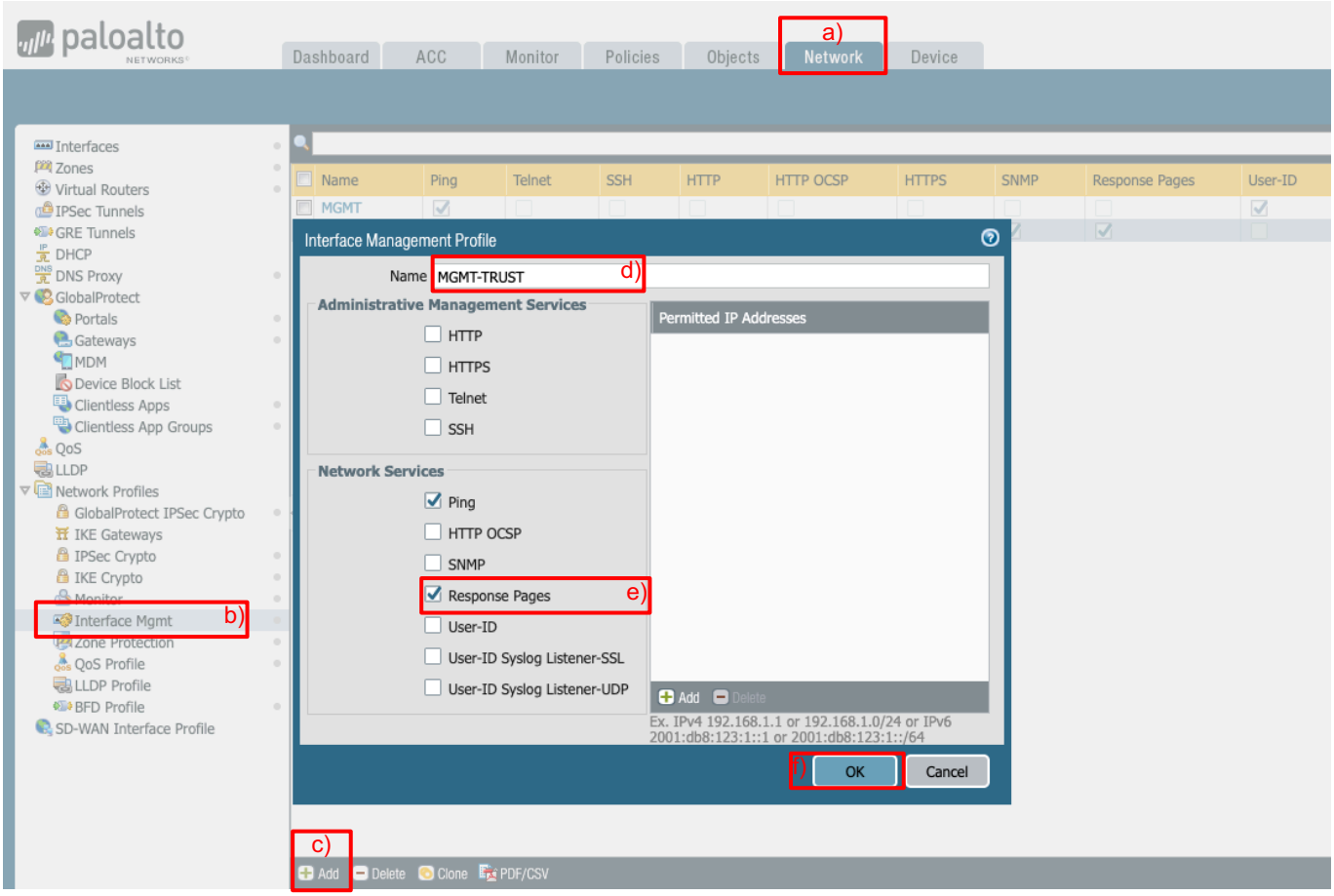


4.3.3.5. インターフェース管理プロファイルの設定

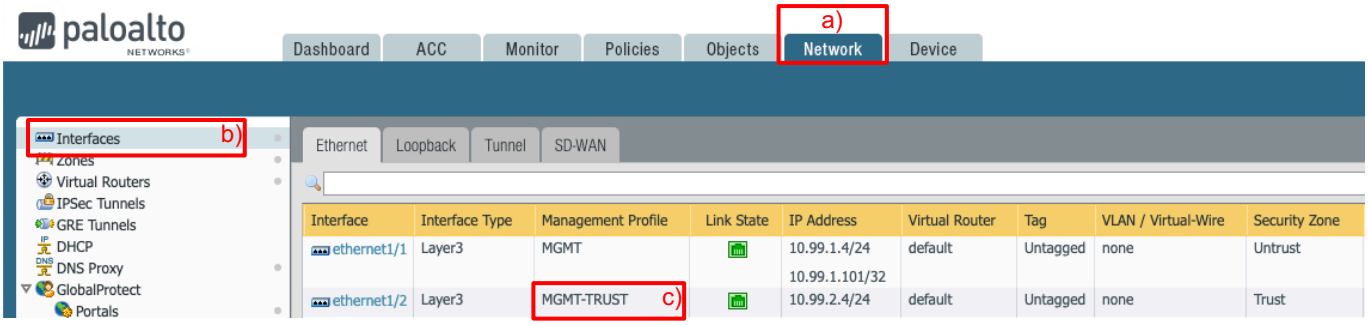
PA Firewall が Captive Portal の Web 画面を返答するためには、インターフェース管理プロファイルの設定が必要です。

Trust ゾーンに属するインタフェースだけに Captive Portal を適用したいので、新規にインターフェース管理プロファイルを生成して、それを Trust ゾーンに適用します。

(1) a)「Network」 → b)「Interface Mgmt」 → c)「Add」をクリックします。
(2) Name に d)「MGMT-TRUST(任意)」、e)「Response Pages」にチェックを入れます。
(3) f)「OK」をクリックします。



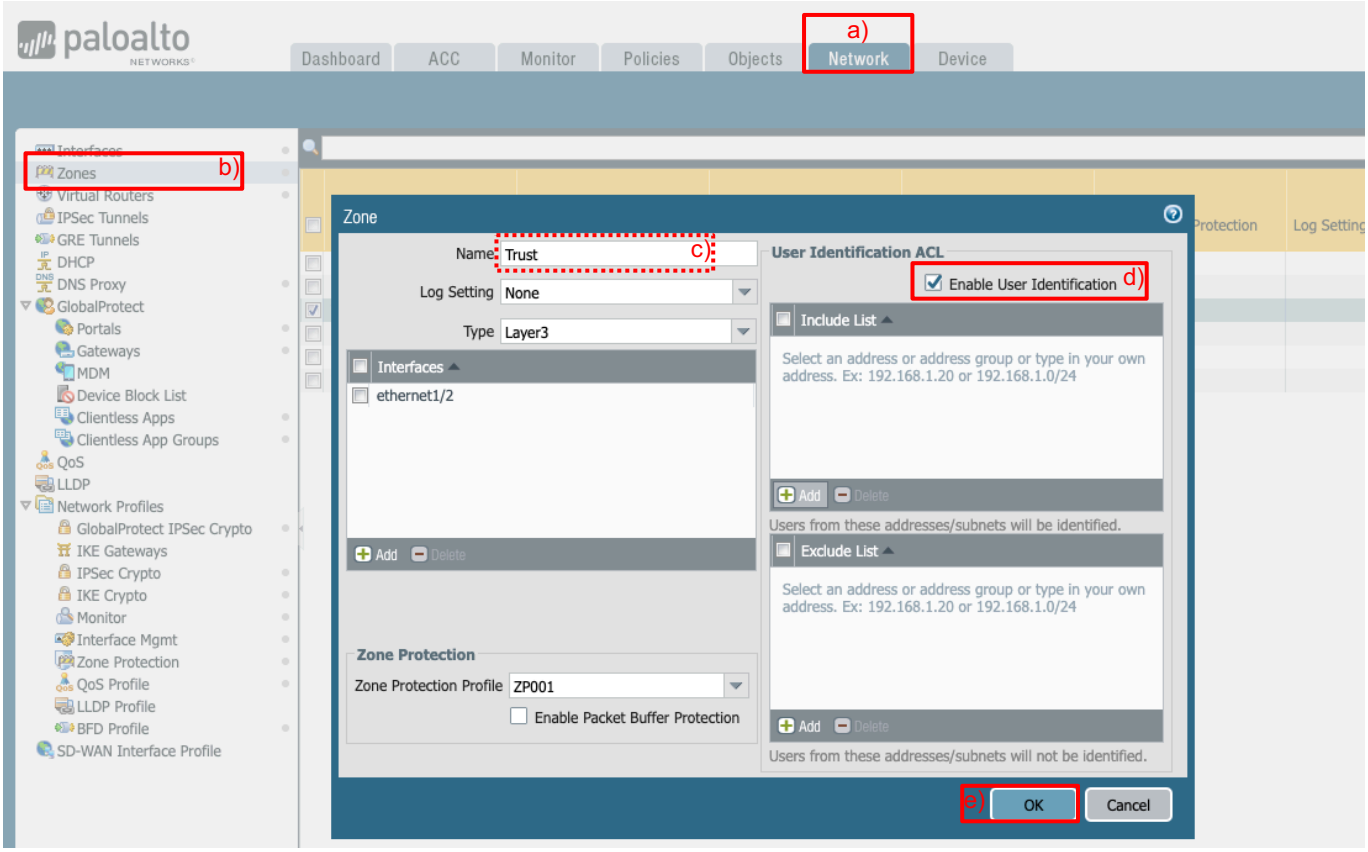
(4) a)「Network」 → b)「Interface」の ethernet1/2 に、設定したインタフェース管理プロファイル「MGMT-TRUST」を適用します。



4.3.3.6. Trust ゾーンでの User-ID の有効化

User-ID 連携を行うためには、ゾーンでユーザ識別機能を有効にする必要があります。

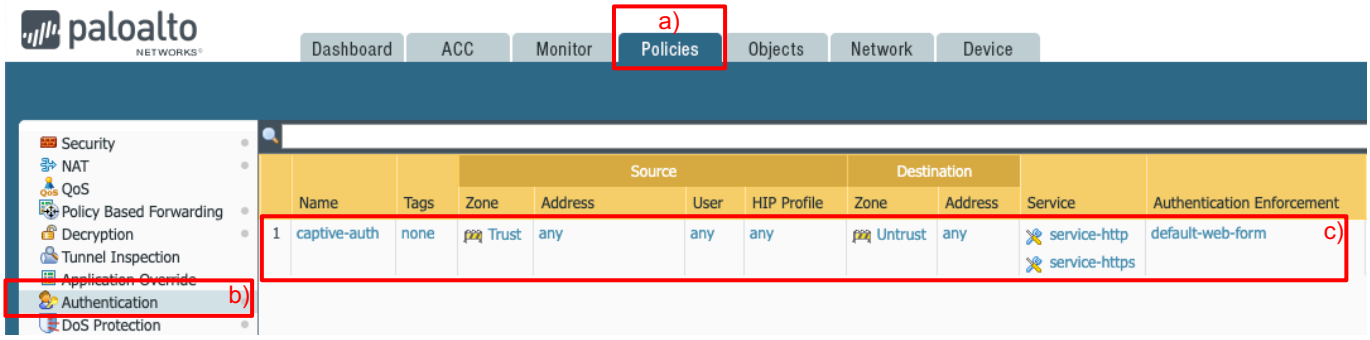
(1) a)「Network」 → b)「Zone」 で、設定済みの c)「Trust」ゾーンを開きます。
(2) d)「Enable User Identification」にチェックを入れ、e)「OK」をクリックします。



4.3.3.7.　認証ポリシーの設定

Captive Portal を動作させるには、認証(Authentication)ポリシーの設定が必要です。

Trust から Untrust への通信の際に Captive Portal が動作するよう、以下のように設定します。



4.3.3.8.　SSL 復号の設定

クライアントが HTTPS サイトへ向かう際に Captive Portal の認証フォームを返すには、SSL 復号が必須となります。
理由は、Captive Portal は、HTTP 302 リダイレクトを使うためです。
SSL 暗号された状態では HTTP ヘッダも暗号化されているため、HTTP 302 リダイレクトが動作しません。

常時 SSL 復号することを避けたい場合、送信元ユーザに「Unknown」を指定することで、SAML によるユーザ識別が完了するまでの間だけ、一時的に SSL 復号を適用することができます。

以下のように設定します。

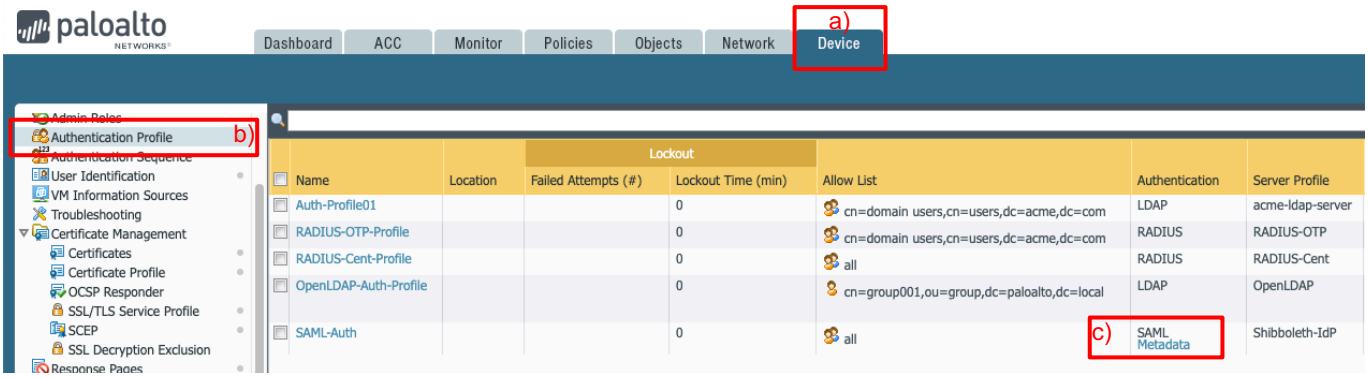
SSL 復号設定に関するその他の詳細は、下記 Link の「PA Firewall 設定ガイド」をご参照ください。
https://live.paloaltonetworks.com/t5/ナレッジドキュメント/pa-series-firewall-設定ガイド-pan-os-8-1/ta-p/209905

Commit を実施します。

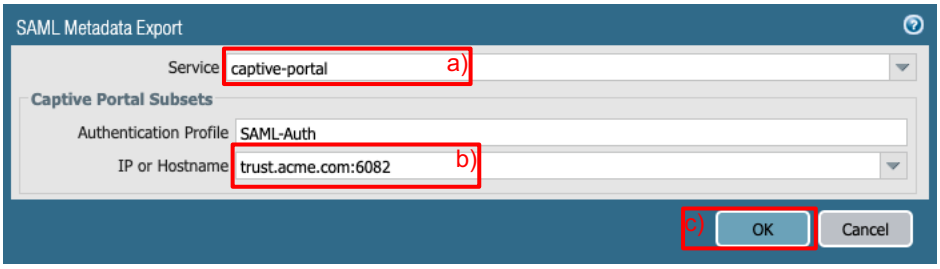
4.4. SP メタデータを IdP へインポート

4.4.1. SP メタデータのエクスポート

(1) a)「Device」 → b)「Authentication Profile」で表示された「SAML-Auth」の c)「Metadata」をクリックします。



(2) メタデータのダウンロード画面が表示されます。
a)「captive-portal」を選択し、b)「trust.acme.com:6082」を選択します。
c)「OK」を押すと、SP メタデータが XML 形式でダウンロードされます。



(3) ファイル名を「saml-auth-simple.xml (任意)」として、一旦コンソール PC へ保存します。

この SP メタデータファイルを IdP へインポートすることによって、IdP はこの情報にしたがって動作します。

このファイルを開いてみると、以下のように、SP が IdP へ送る AuthN Request 内の「SP の署名」を、IdP 側では検証しない設定(無効化)になっています。(確認のみ)

```
<md:EntityDescriptor xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
                     entityID="https://trust.acme.com:6082/SAML20/SP">
  <md:SPSSODescriptor AuthnRequestsSigned="false"
                      WantAssertionsSigned="false"
                      protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
    <md:KeyDescriptor>
      <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
      </ds:KeyInfo>
    </md:KeyDescriptor>
    <md:AssertionConsumerService index="0"
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
Location="https://trust.acme.com:6082/SAML20/SP/ACS"/>
  </md:SPSSODescriptor>
</md:EntityDescriptor>
```

4.4.2. SP メタデータを IdP へ保存

この SP メタデータファイル「saml-auth-simple.xml」を、Shibboleth-IdP の下記ディレクトリへ保存します。

ディレクトリ:/opt/shibboleth-idp/metadata/

4.4.3. Metadata-providers の設定

metadata-providers.xml ファイルで、どの SP メタデータを使うかを指定します。

```
# vi /opt/shibboleth-idp/conf/metadata-providers.xml
```

```
    <!--
    <MetadataProvider id="LocalMetadata"  xsi:type="FilesystemMetadataProvider"
metadataFile="PATH_TO_YOUR_METADATA"/>
    -->
```

(上記をコピー(または直接編集)して、下記の形に変更。)

```
    <!--  コメントアウトの削除
    <MetadataProvider id="LocalMetadata"  xsi:type="FilesystemMetadataProvider"
metadataFile="/opt/shibboleth-idp/metadata/saml-auth-simple.xml"/>
    -->  コメントアウトの削除
```

4.4.4. Relying-party の設定

relying-pary.xml ファイルで、信頼できる SP として、PA Firewall を追加します。
SP メタデータ内の entity ID の値を使って、以下を設定のように設定します。

```
# vi /opt/shibboleth-idp/conf/relying-party.xml
```

```
        <!--
        <bean parent="RelyingPartyByName" c:relyingPartyIds="https://sp.example.org">
            <property name="profileConfigurations">
                <list>
                    <bean parent="SAML2.SSO" p:encryptAssertions="false" />
                </list>
            </property>
        </bean>
        -->
```

(上記をコピー(または直接編集)して、下記の形に変更。)

```
        <!--  コメントアウトの削除
        <bean parent="RelyingPartyByName"
c:relyingPartyIds="https://trust.acme.com:6082/SAML20/SP">
            <property name="profileConfigurations">
                <list>
                    <bean parent="SAML2.SSO" p:encryptAssertions="false" />
                </list>
            </property>
        </bean>
        -->  コメントアウトの削除
```

4.4.5. Tomcat サービスの再起動

(1) Tomcat を再起動します。

```
# systemctl restart tomcat
```

(2) Tomcat のログにエラーが出ていないことを確認します。

```
# tail /usr/share/tomcat/logs/catalina.YYYY-MM-DD.log
```

(3) 以下のファイルにもエラーがないことを確認します。

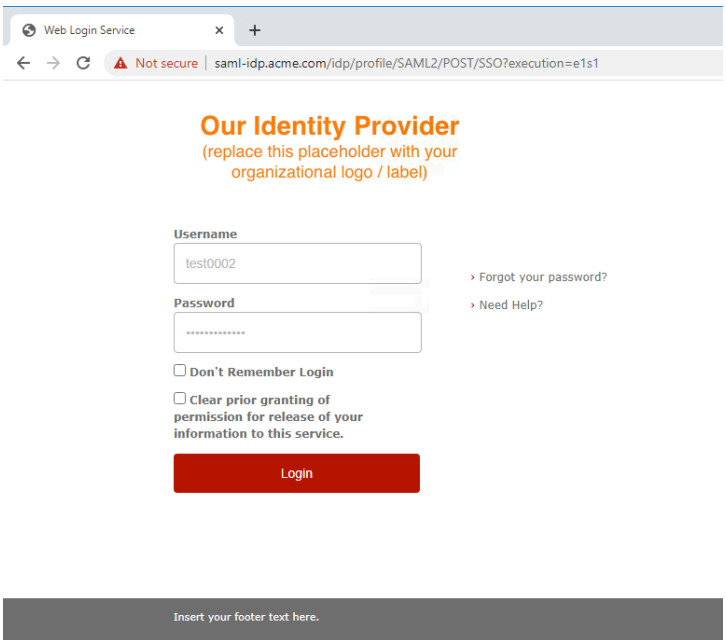
```
# tail /opt/shibboleth-idp/logs/idp-process.log
# tail /opt/shibboleth-idp/logs/idp-warn.log
```

4.5. SAML 動作確認

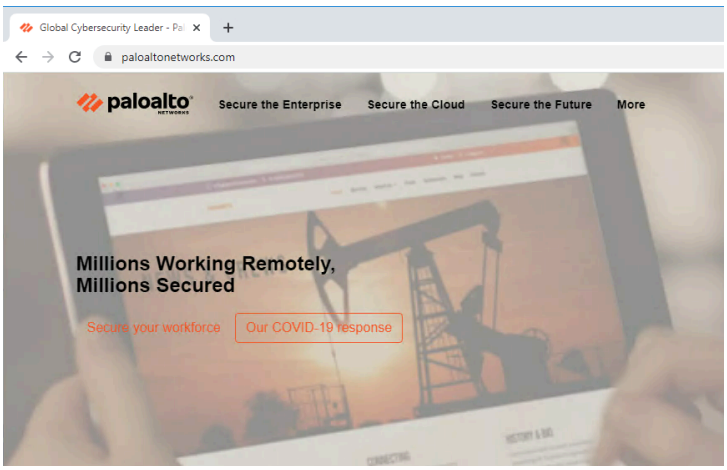
SAML 認証が動作するかを、クライアント PC から確認します。

(1) クライアントの Web ブラウザで、www.paloaltonetworks.com へアクセスします。

(2) IdP(Shibboleth)へリダイレクトされますので、ID とパスワードを入力し「Login」をクリックします。



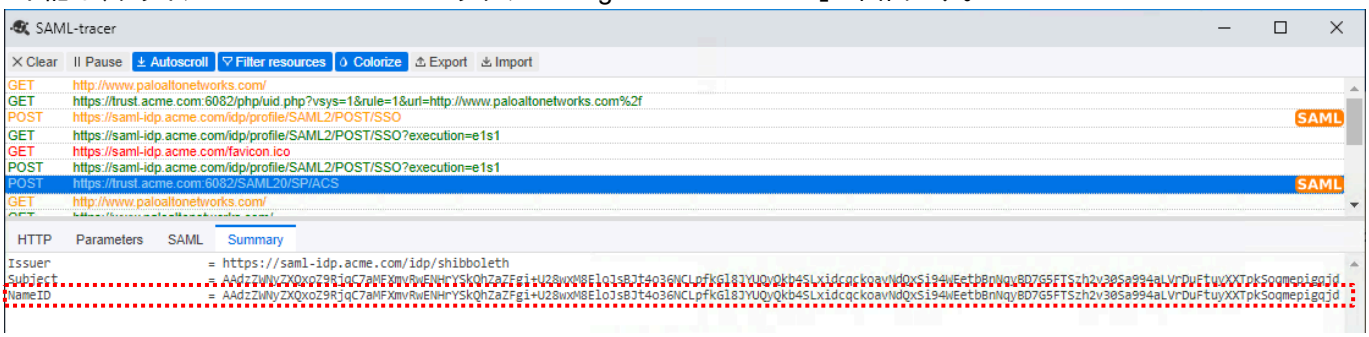
(3) 認証が完了すると、www.paloaltonetworks.com の Web 画面が表示されます。



4.6. User-ID の連携

ここまでの設定では、PA Firewall 上の User-ID には、SAML Response 内の NameID:「AAdzZWNyZXQxkkqH6y7～略～ShYH6wvBiQZRZ」のような長い文字列が使われた状態です。

下記は、クライアント PC の Chrome ブラウザの Plugin:「SAML-Tracer」の画面です。



IdP からの SAML Response で、これらパラメータを受け取っているだけで、クライアントが認証時に利用した username を受け取っていません。
PA Firewall は(指定されたパラメータが存在しない場合)デフォルトでは、NameID を User-ID に利用しています。

以降、SAML 認証を実施した username で、User-ID 連携できるように設定します。

4.6.1. Username を SP へ送るための設定

4.6.1.1. attribute-filter の設定

Shibboleth-IdP の attribute-filter.xml を編集することで、クライアントが認証時に利用した username が渡されるようになります。

(1) ファイルを編集します。

```
# vi /opt/shibboleth-idp/conf/attribute-filter.xml
```

```
    <!-- Release an additional attribute to an SP. -->
    <!--     コメントアウト
    <AttributeFilterPolicy id="example1">
        <PolicyRequirementRule xsi:type="Requester" value="https://sp.example.org" />

        <AttributeRule attributeID="uid" permitAny="true" />
    </AttributeFilterPolicy>
    -->     コメントアウト

上記をコピーして、以下のように編集

    <AttributeFilterPolicy id="example1">
        <PolicyRequirementRule xsi:type="Requester"
value="https://trust.acme.com:6082/SAML20/SP" />

        <AttributeRule attributeID="uid" permitAny="true" />
    </AttributeFilterPolicy>
```

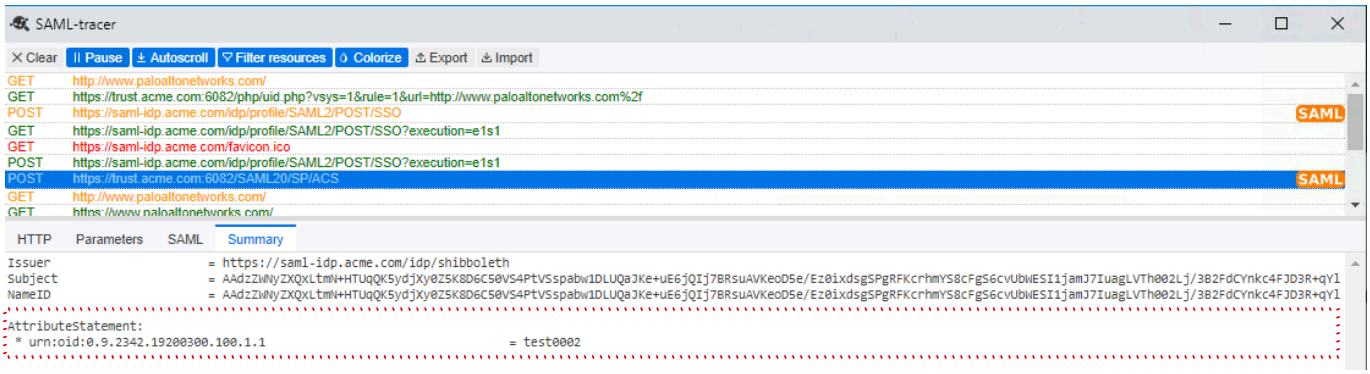
(2) Tomcat を再起動します。

```
# systemctl restart tomcat
```

4.6.1.2.　確認(1)

(1)　クライアント PC の Chrome ブラウザで SAML-Tracer を有効にして、もう一度、SAML 認証を行います。

(2)　認証時に利用した username が、Attribute Statement として、IdP から渡されるようになります。

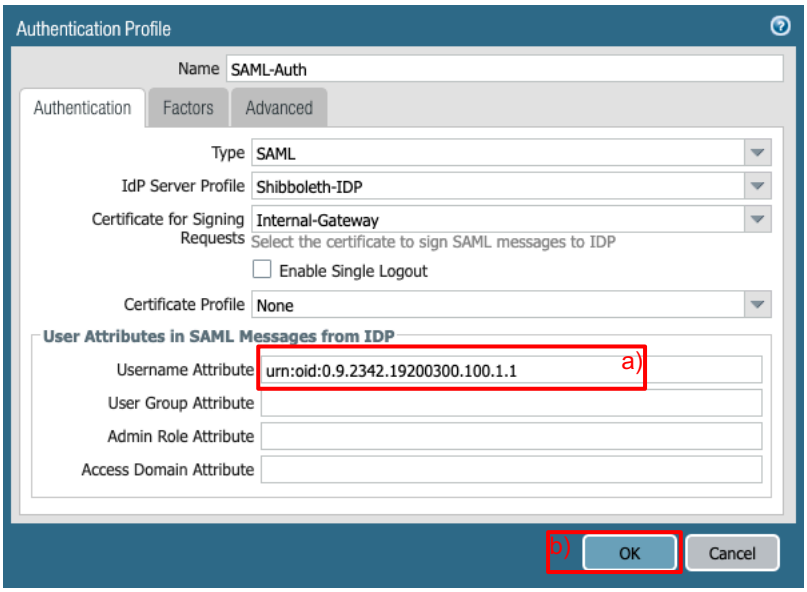


4.6.1.3.　SP(PA Firewall)側の設定変更

IdP から渡される username の変数名が「urn:oid:0.9.2342.19200300.100.1.1」なので、それに合わせて Firewall 側も設定を変更します。

(1)　「Device」 → 「Authentication Profile」で表示された「SAML-Auth」を開きます。

(2)　a)Username Attribute に、「urn:oid:0.9.2342.19200300.100.1.1」を入力し、c)「OK」をクリックします。

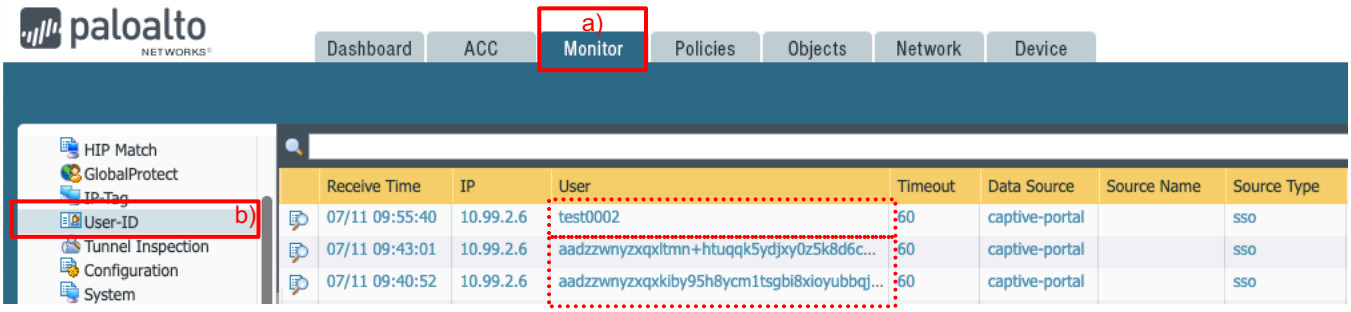


(3)　Commit します。

4.6.1.4. 確認(2)

(1) もう一度、クライアント PC で SAML 認証を行います。

(2) 以下のように、長い文字列だったユーザ名が、認証時に利用した username=「test0002」になります。



4.6.2. ［参考］変数名を「username」にするための設定

4.6.2.1. attribute-resolver の設定

Shibboleth-IdP の attribute-resolver.xml を編集することで、変数名を username に変えることができます。

(1) ファイルを編集します。

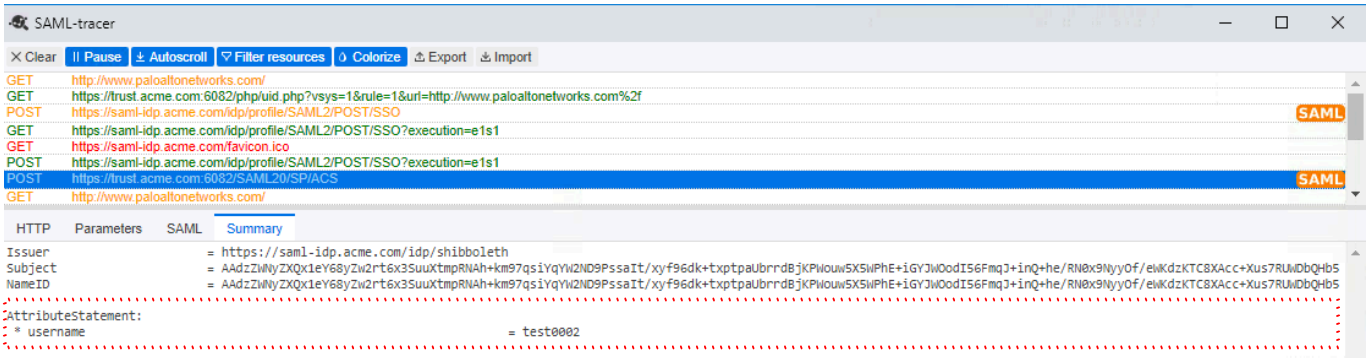
```
# vi /opt/shibboleth-idp/conf/attribute-resolver.xml
```

```
    <!--      コメントアウト
    <AttributeDefinition id="uid" xsi:type="PrincipalName">
        <AttributeEncoder xsi:type="SAML1String" name="urn:mace:dir:attribute-def:uid"
encodeType="false" />
        <AttributeEncoder xsi:type="SAML2String" name="urn:oid:0.9.2342.19200300.100.1.1"
friendlyName="uid" encodeType="false" />
    </AttributeDefinition>
    -->      コメントアウト

上記をコピーして、以下のように編集

    <AttributeDefinition id="uid" xsi:type="PrincipalName">
        <AttributeEncoder xsi:type="SAML1String" name="urn:mace:dir:attribute-def:uid"
encodeType="false" />
        <AttributeEncoder xsi:type="SAML2String" name="username" friendlyName="uid"
encodeType="false" />
    </AttributeDefinition>
```

(2) Tomcat を再起動します。

```
# systemctl restart tomcat
```

(3) 以下のように、変数名が「username」に変わります。

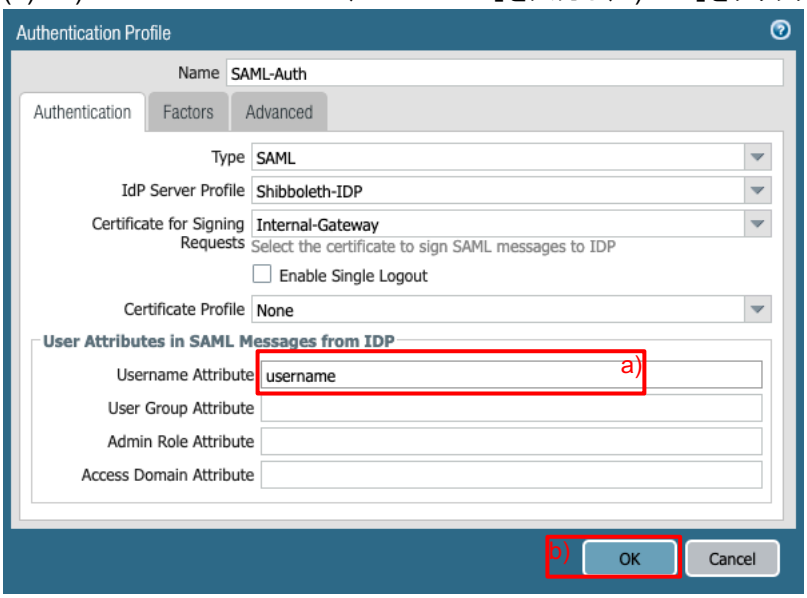


4.6.2.2.　SP(PA Firewall)側の設定変更

IdP から渡される username の変数名が、上記の設定によって「username」になるので、PA Firewall 側もそれに合わせて設定変更します。

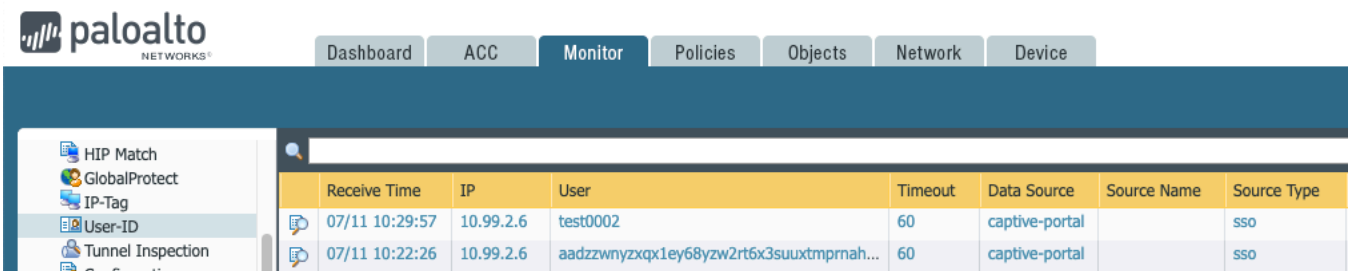
(1)　「Device」 → 「Authentication Profile」で表示された「SAML-Auth」を開きます。

(2)　a)Username Attribute に、「username」を入力し、c)「OK」をクリックします。



(3)　Commit します。

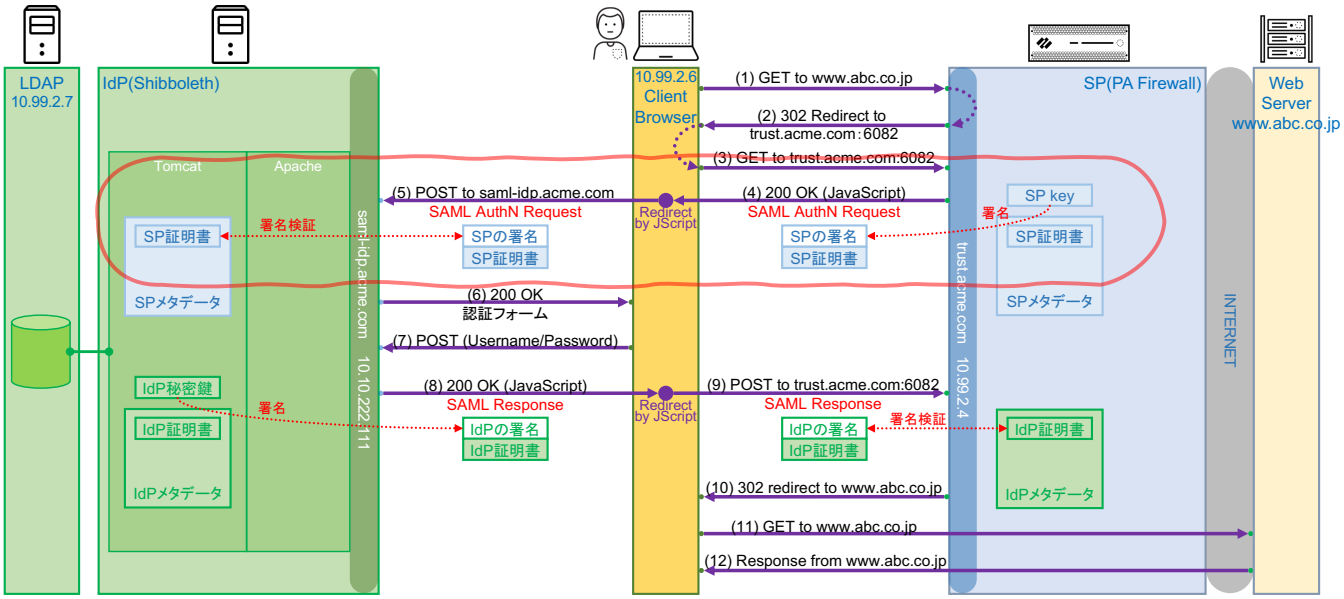
(4)　この場合も同様に、User-ID は、SAML 認証の username:「test0002」になります。



5. SAML 認証のセキュリティ強化設定

5.1. SP の署名を IdP が検証する設定

SP から IdP へ送る SAML 認証リクエスト(AuthN Request)に SP が署名し、IdP がその署名を検証することで、SP のなりすましを防ぐことができます(下図の(4)～(5)の部分)。その設定を行います。

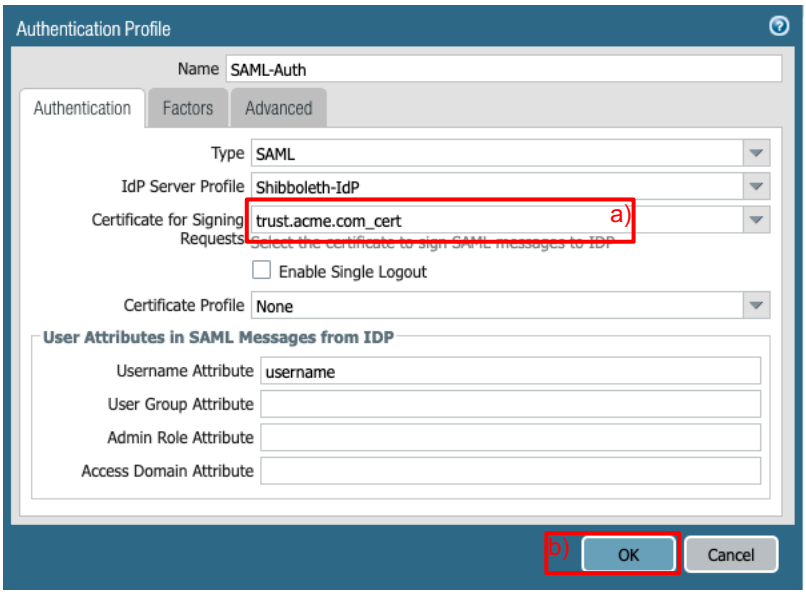


5.1.1. SP(PA Firewall)側の認証プロファイルの設定変更

どのサーバ証明書と対になった秘密鍵で「SP の署名」を行うのかを指定します。
また、このサーバ署名書が SP からの SAML 認証リクエスト(AuthN Request)に加えられます。

(1) 「Device」 → 「Authentication Profile」で表示された「SAML-Auth」を開きます。

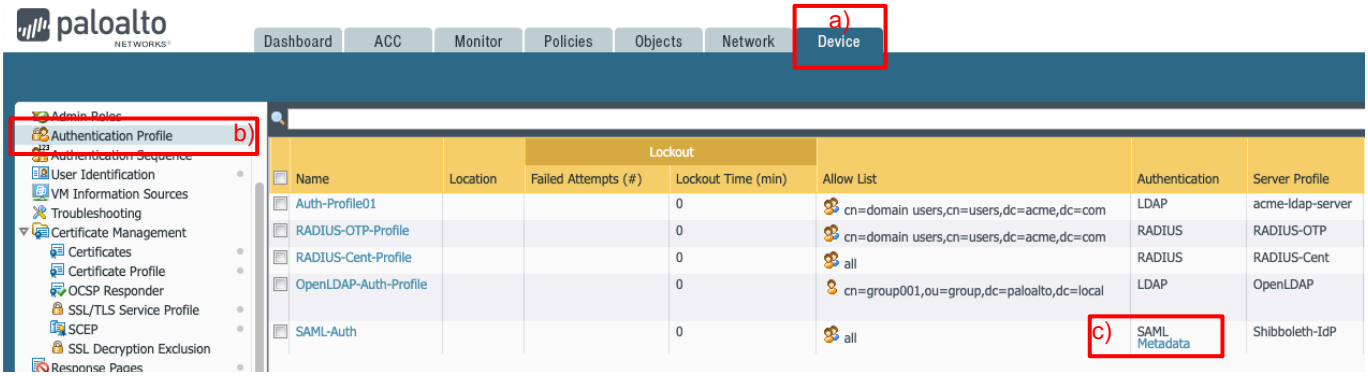
(2) a)Certificate for Singing Requests で、「trust.acme.com_cert」を選択し、b)「OK」をクリックします。



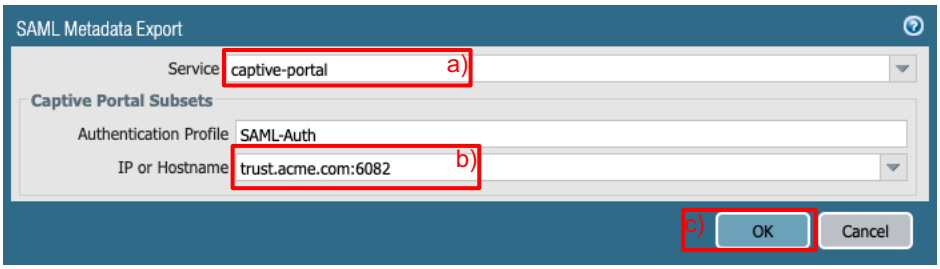
5.1.2. SP メタデータを IdP へインポート

5.1.2.1. SP メタデータのエクスポート

(1) a)「Device」 → b)「Authentication Profile」で表示された「SAML-Auth」の c)「Metadata」をクリックします。



(2) メタデータのダウンロード画面が表示されます。
a)「captive-portal」を選択します。
b)「trust.acme.com:6082」を選択します。
c)「OK」を押すと、SP メタデータが XML 形式でエクスポートされます。



(3) ファイル名を「saml-auth.xml (任意)」として、一旦コンソール PC へ保存します。

(4) このファイルを開いて、以前のメタデータと比較して、ファイル内に証明書の部分が増えていることを確認してください(それが SP のサーバ証明書です)。

5.1.2.2. SP メタデータを IdP へ保存

この SP メタデータファイル「saml-auth.xml」を、Shibboleth-IdP の下記ディレクトリへ保存します。

ディレクトリ:/opt/shibboleth-idp/metadata/

5.1.2.3. Metadata-providers の設定

metadata-providers.xml ファイルを編集して、この SP メタデータを参照するようにします。

```
# vi /opt/shibboleth-idp/conf/metadata-providers.xml
```

(本ガイドで設定した、現在の状態)

```
    <MetadataProvider id="LocalMetadata"  xsi:type="FilesystemMetadataProvider"
metadataFile="/opt/shibboleth-idp/metadata/saml-auth-simple.xml"/>
```

(以下の形に変更)

```
    <MetadataProvider id="LocalMetadata"  xsi:type="FilesystemMetadataProvider"
metadataFile="/opt/shibboleth-idp/metadata/saml-auth.xml"/>
```

5.1.3.　SP の署名検証に失敗する動作の確認

まず、SP(PA)は「SP の署名」を送らないが、IdP 側は「SP の署名」の検証は必須とする設定を行い、SP の署名検証に失敗する状態を確認します。

5.1.3.1.　SP メタデータの変更

(1)　IdP に保存した SP メタデータ (saml-auth.xml) の「AuthnRequestsSigned=」を「”true”」に変更します。

```
# vi /opt/shibboleth-idp/metadata/saml-auth.xml
```

```
<md:EntityDescriptor xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
                     entityID="https://trust.acme.com:6082/SAML20/SP">
  <md:SPSSODescriptor AuthnRequestsSigned="true"
                      WantAssertionsSigned="false"
                      protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
    <md:KeyDescriptor>
～省略～
```

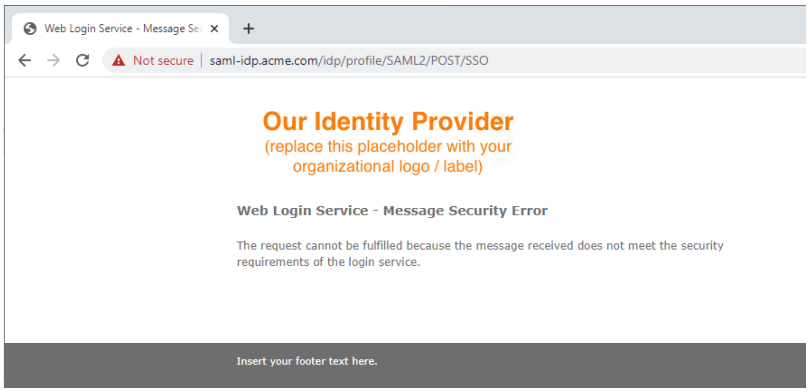
(2)　Tomcat を再起動します。

```
# systemctl restart tomcat
```

5.1.3.2.　確認

(1)　クライアント PC の Chrome ブラウザで SAML 認証を行います。

(2)　SP(PA Firewall)が「SP の署名」を行っていないので、IdP は、エラーを返答します。

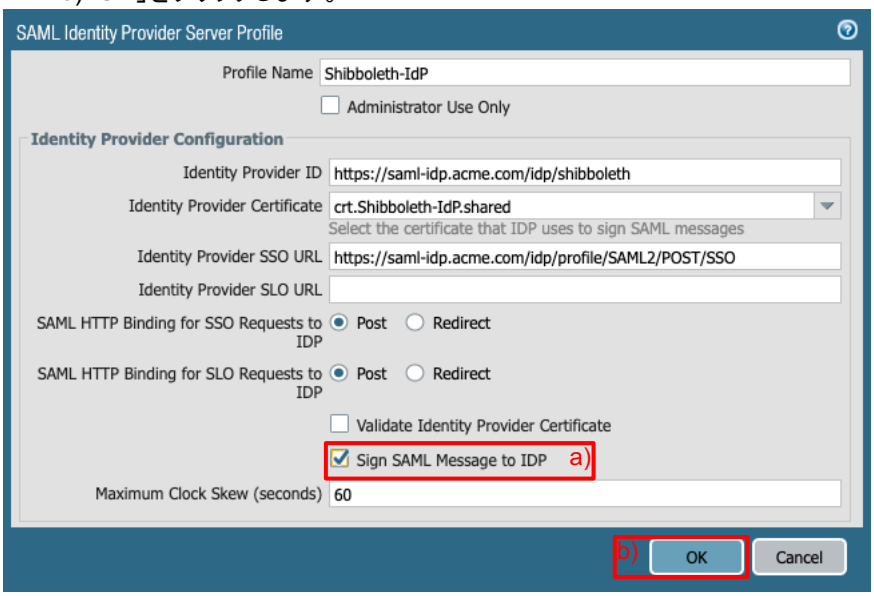


5.1.4. SP の署名検証に成功する動作の確認

SP(PA Firewall)が「SP の署名」を行うように設定して、SP の署名検証が成功する状態を確認します。

5.1.4.1. SAML IdP Server Profile の設定変更

(1) 「Device」 →「SMAL Identity Provider」 →「Shibboleth-IdP」を開きます。
(2) a)「Sign SAML Message to IDP」にチェックを入れます。
b)「OK」をクリックします。

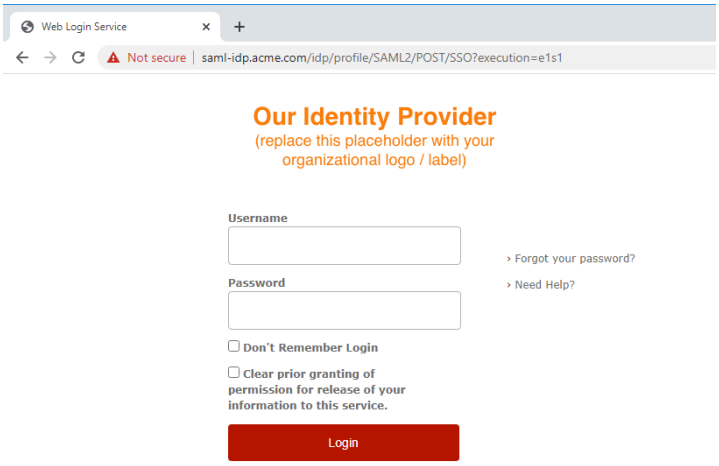


(3) Commit します。

5.1.4.2. 確認

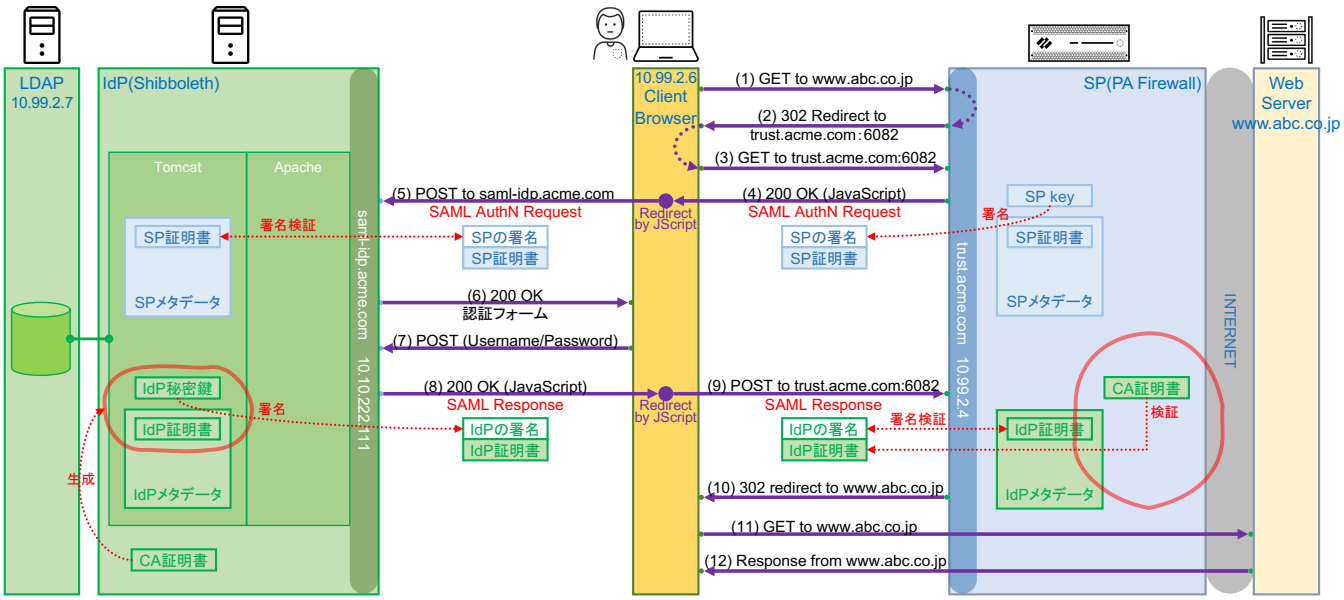
(1) クライアント PC の Chrome ブラウザで SAML 認証を行います。

(2) SP(PA Firewall)が「SP の署名」を行ったので、署名検証が行われ、IdP は正常に認証フォームを返答します。



5.2. IdP のサーバ証明書の信頼性を検証する設定

SP(PA Firewall)では、既に「IdP の署名」の検証は動作しています。
その検証に加えて、「IdP のサーバ証明書は、SP(PA Firewall)が持つ CA(認証局)から発行されたものか?」を確認することができるようになっています。その設定を行います。



5.2.1. IdP のサーバ証明書の作成

IdP のサーバ証明書の信頼性を検証するには、その IdP サーバ証明書を発行した CA の証明書が必要です。

Shibboleth がデフォルトで生成したサーバ証明書は、CA を持たない自己署名証明書であるため、別途、CA とサーバ証明書を生成します。

以下は、CentOS7 の OpenSSL で作成する手順です。

- OpenSSL を使わず、PA で生成して、Shibboleth へインポートしても構いません。
- OpenSSL はデフォルトでインストールされていると思いますが、なければ yum などでインストールしてください。

5.2.1.1. CA (ルート認証局) の作成

(1) CA 用のフォルダへ移動します。

```
# cd /etc/pki/CA/
```

(2) ファイルの生成

自己署名による証明書発行に必要なシリアル番号およびインデックス管理のためのファイルを生成します。
以下のコマンドを実行します。

```
# echo "01" > ./serial     :openssl が利用する、シリアル番号管理のためのファイル
# touch ./index.txt        :openssl が利用する、インデックス管理のためのファイル
```

(3)　CA の秘密鍵と署名リクエスト(CSR)の生成

CA 自身が、秘密鍵と署名リクエスト(CSR)を生成します。
以下のコマンドを実行します(「-subj」以降は任意です)。

```
# openssl req -new -newkey rsa:2048 -nodes -sha256 -keyout ./private/cakey.pem ¥
-out ./careq.pem ¥
-subj "/C=JP/ST=Tokyo/L=Minato-ku/O=PANJ-CA, Ltd./OU=CA/CN=panjca.panjp.local"
```

(4)　CA による自己署名

CA の証明書に自己署名します。
以下のコマンドを実行します。

```
# openssl ca -md sha256 -in ./careq.pem -keyfile ./private/cakey.pem ¥
-out ./cacert.pem ¥
-selfsign -extensions v3_ca -batch -days 3650
```

CA の生成は完了です。

5.2.1.2.　サーバ証明書の作成

(1)　サーバ証明書を保存するディレクトリを作ります。

```
# mkdir SERVER
```

(2)　saml-idp.acme.com の、秘密鍵と署名リクエスト(CSR)を生成します。
以下のコマンドを実行します(「-subj」以降は任意です)。

```
# openssl req -new -newkey rsa:2048 -nodes -sha256 -keyout ./SERVER/samlidp-key.pem ¥
-out ./SERVER/samlidp-req.pem ¥
-subj "/C=JP/ST=Kanagawa/L=Yokohama/O=ABC-Company, Ltd./OU=IT/CN=saml-idp.acme.com"
```

(3)　上記で生成した署名リクエスト(CSR)へ、CA が署名します。
以下のコマンドを実行します。

```
# openssl ca -md sha256 -in ./SERVER/samlidp-req.pem -out ./SERVER/samlidp-crt.pem ¥
-keyfile ./private/cakey.pem -batch -policy policy_anything ¥
-config /etc/pki/tls/openssl.cnf -days 3650
```

これで、saml-idp.acme.com のサーバ証明書の生成は完了です。

(4)　SERVER ディレクトリに移動して、生成されたサーバ証明書と秘密鍵を確認します。

```
# cd SERVER/
# ls -l
-rw-r--r-- 1 root root 4515  6月 13 11:32 samlidp-crt.pem
-rw-r--r-- 1 root root 1704  6月 13 11:31 samlidp-key.pem
-rw-r--r-- 1 root root 1025  6月 13 11:31 samlidp-req.pem
```

5.2.2.　生成したサーバ秘密鍵で IdP の署名を行う設定

生成した IdP 用サーバ秘密鍵で、IdP が SAML Response 内の署名を行えるように設定します。

5.2.2.1.　オーナー権限とパーミッションの変更

(1)　生成した IdP 用サーバ証明書と鍵を、/opt/shibboleth-idp/credentials/ へコピーします。

```
# cp /etc/pki/CA/SERVER/samlidp-crt.pem /opt/shibboleth-idp/credentials/
# cp /etc/pki/CA/SERVER/samlidp-key.pem /opt/shibboleth-idp/credentials/
```

(2)　IdP 用サーバ秘密鍵のオーナー権限とパーミッションを変更します。

```
# chown tomcat:root /opt/shibboleth-idp/credentials/samlidp-key.pem
# chmod 400 /opt/shibboleth-idp/credentials/samlidp-key.pem
```

5.2.2.2.　idp.properties の変更

idp.properties を以下のように変更することで、生成した IdP サーバ秘密鍵で署名され、生成した IdP サーバ証明書が SAML Response で SP へ送られるようになります。

(1)　ファイルを編集します。

```
# vi /opt/shibboleth-idp/conf/idp.properties
```

```
#idp.signing.key=%{idp.home}/credentials/idp-signing.key        ←コメントアウト
#idp.signing.cert=%{idp.home}/credentials/idp-signing.crt       ←コメントアウト

idp.signing.key=%{idp.home}/credentials/samlidp-key.pem
idp.signing.cert=%{idp.home}/credentials/samlidp-crt.pem
```

(2)　Tomcat を再起動します。

```
# systemctl restart tomcat
```

5.2.3. SP (PA Firewall) の設定

5.2.3.1. IdP サーバ証明書と CA 証明書を SP へインポート

OpenSSL で生成した IdP のサーバ証明書と CA 証明書を SP(PA Firewall)へインポートします。

- CA 証明書のファイル: `/etc/pki/CA/cacert.pem`
- サーバ証明書のファイル: `/etc/pki/CA/SERVER/samlidp-crt.pem`

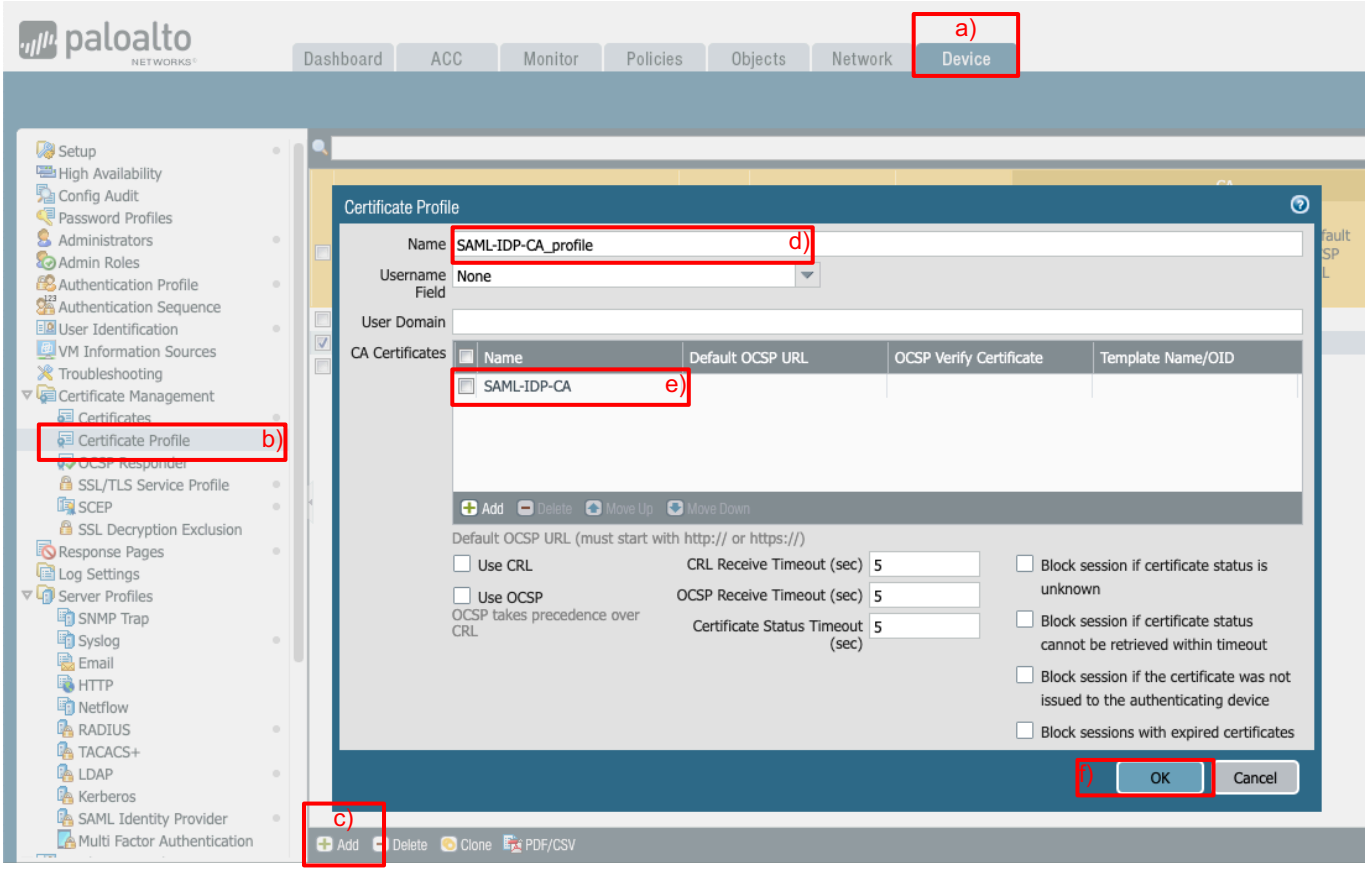
インポートした証明書は、以下の名前にしています(任意)。

- CA 証明書: SAML-IDP-CA
- IdP のサーバ証明書: SAML-IDP-Cert

5.2.3.2. 証明書プロファイルの設定

証明書プロファイルで、どの CA 証明書を使って IdP のサーバ証明書の検証を行うのかを指定します。

(1) a)「Device」 → b)「Certificate Profile」 → c)「Add」をクリックします。
d)Name に「SAML-IDP-CA_profile(任意)」、e)CA Certificates で「SAML-IDP-CA」を選択します。
f)「OK」をクリックします。

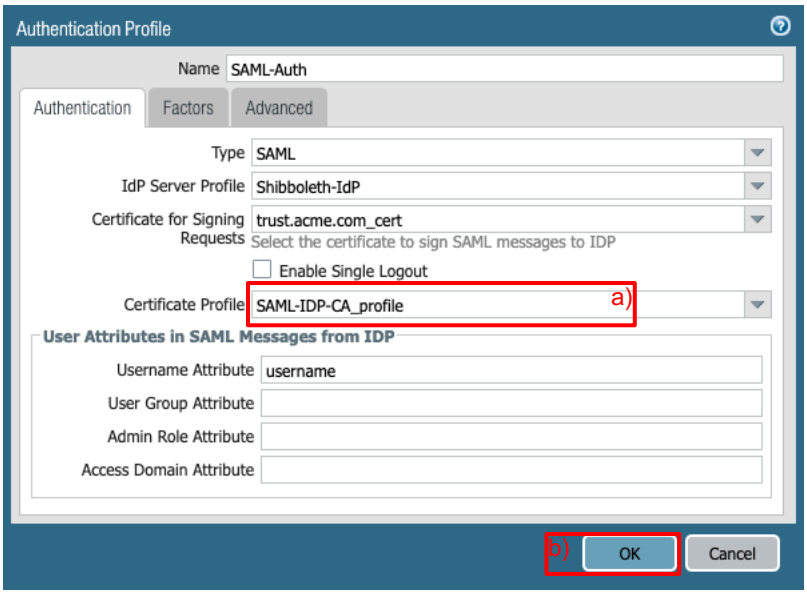


5.2.3.3.　認証プロファイルの設定

設定した証明書プロファイルを選択して、IdP のサーバ証明書の検証が行えるようにします。

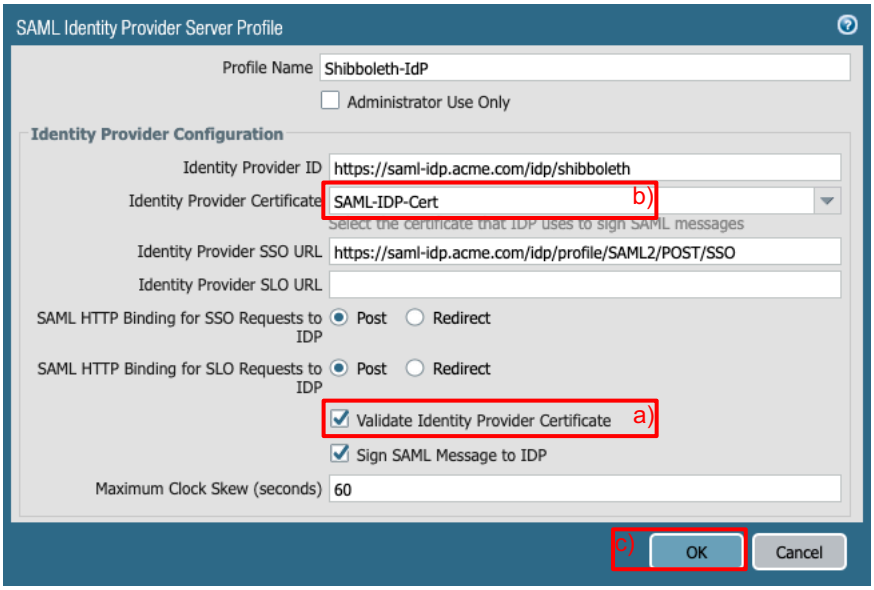
(1)　「Device」 → 「Authentication Profile」で表示された「SAML-Auth」を開きます。

(2)　a)Certificate Profile で、「SAML-IDP-CA_profile」を選択し、b)「OK」をクリックします。



5.2.3.4.　SAML IdP Server Profile の設定変更

(1)　「Device」 → 「SMAL Identity Provider」 → 「Shibboleth-IdP」を開きます。
(2)　a)「Validate Identity Provider Certificate」にチェックを入れ、b)Identity Provider Certificate で「SAML-IDP-Cert」を選択します。
　　c)「OK」をクリックします。



(3)　Commit を実施します。

5.2.4.　動作確認

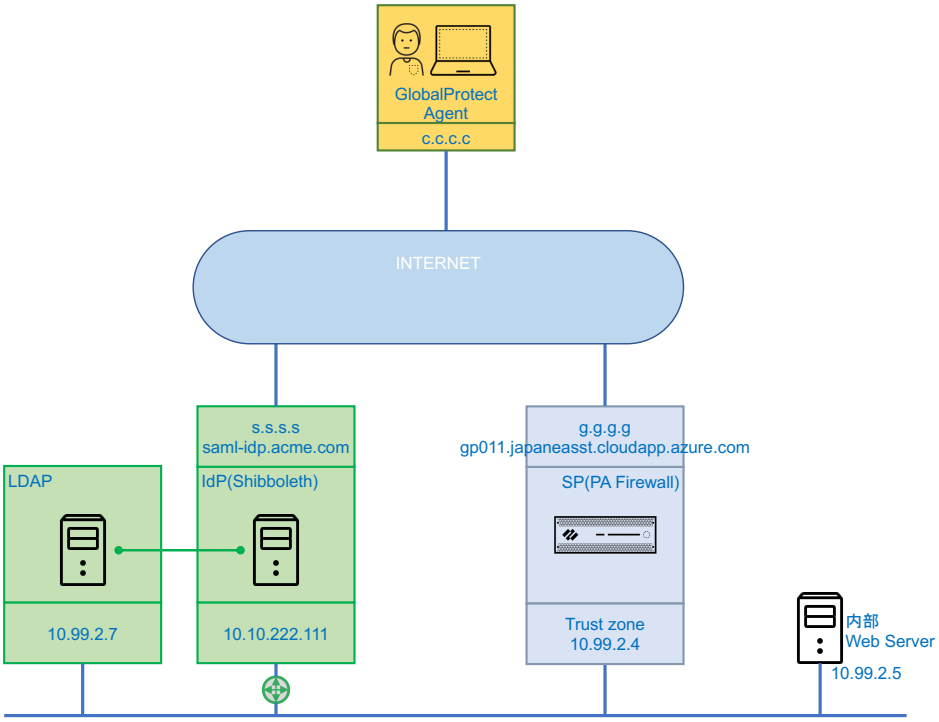
(1)　クライアント PC の Chrome ブラウザで SAML 認証を行います。
(2)　正しく認証が完了して、Web アクセスができることを確認します。

6. SAML 認証の設定 (GlobalProtect)

GlobalProtect で SAML 認証を行う場合の設定です。

本ガイドでは、GlobalProtect の設定は完了しているものとして、SAML 認証に関わる部分のみ記載します。

GlobalProtect の設定に関しては、「GlobalProtect 設定ガイド」(下記 Link)をご参照ください。
https://live.paloaltonetworks.com/t5/ナレッジドキュメント/globalprotect-設定ガイド-pan-os-8-1/ta-p/280308



6.1. 設定ステップ

GlobalProtect の場合の基本的な設定ステップは以下です (Captive Portal の場合と概ね同じです)。

(1) SP(PA Firewall)側
① Portal および Gateway の認証方式を SAML に設定。
② GlobalProtect の SP メタデータをエクスポート。

(2) IdP(Shibboleth)側
① SP メタデータをディレクトリに保存。
② Metadata-providers.xml への設定追加。
③ Relying-party.xml への設定追加。
④ Attribute-filter.xml への設定追加。

(3) GlobalProtect の場合の注意点
IdP の HTTPS サーバ (本ガイドでは Apache) のサーバ証明書を、GlobalProtect Agent(以降、GP Agent)が信頼できる状態になっていることが必須です。そうなっていなければエラーとなり、繋がりません。
(検証環境で有りがちなトラブルポイントです。商用 IdP サーバでは基本的に発生しません。)

具体的には以下の作業が必要です。
① CA が発行したサーバ証明書であること。
② その CA の証明書が、GlobalProtect Agent を持つクライアント PC の証明書ストアの「信頼された証明書発行機関」のフォルダに保存されていること。

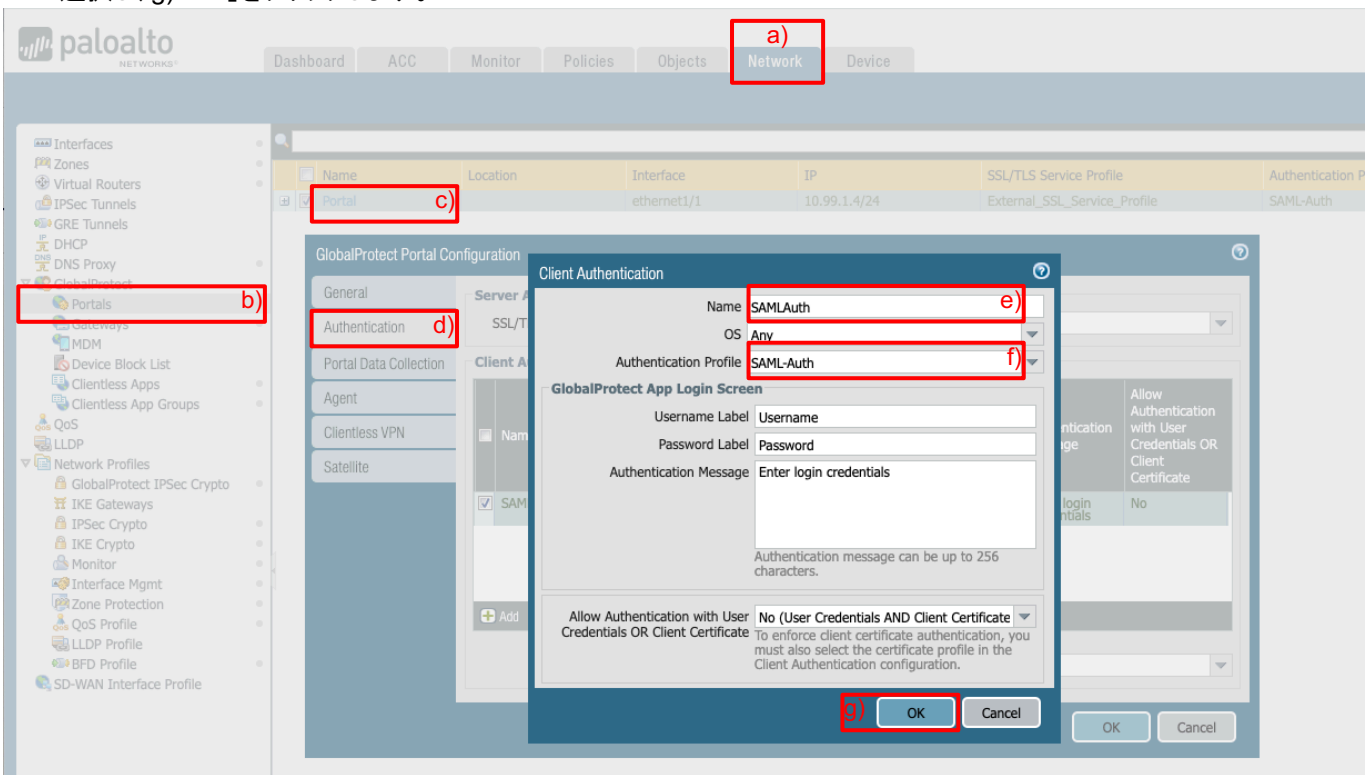
6.2. SP(PA Firewall)の設定

6.2.1. Portal 及び Gateway の認証方式を SAML に設定

Portal 及び外部 Gateway で SAML 認証が行われるように設定します。

6.2.1.1. Portal

(1) a)「Network」 → Globaprotet 下の b)「Portals」 → c)設定済みのポータルをクリックします。
(2) d)「Authentication」 → e)Name に「SAMLAuth(任意)」 → f)Authentication Profile に設定済みの「SAML-Auth」を選択し、g)「OK」をクリックします。

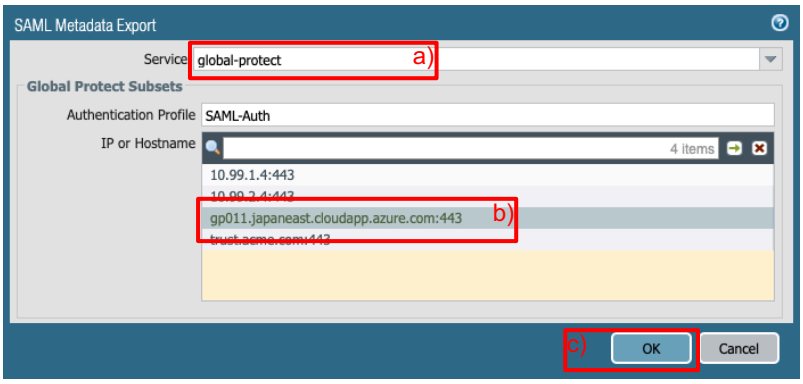


6.2.1.2. 外部 Gateway

SAML 認証を設定する Gateway にも、Portal と同様の設定を行います。

6.2.2. SP メタデータのエクスポート

(1) 「Device」 → 「Authentication Profile」で表示された「SAML-Auth」の「Metadata」をクリックします。
(2) a)「global-protect」を選択し、b)で SAML 認証を行う Gateway を選択します。
c)「OK」を押すと、SP メタデータが XML 形式でエクスポートされます。



ファイル名を「saml-auth_for_GP.xml(任意)」として保存します。

6.3. IdP(Shibboleth)側の設定

6.3.1. SP メタデータを IdP へ保存

この SP メタデータファイル「saml-auth_for_GP.xml」を、Shibboleth-IdP の下記ディレクトリへ保存します。

ディレクトリ:/opt/shibboleth-idp/metadata/

6.3.2. Metadata-providers の設定

metadata-providers.xml ファイルで、どの SP メタデータを使うかを指定します。

```
# vi /opt/shibboleth-idp/conf/metadata-providers.xml
```

```
   <MetadataProvider id="LocalMetadata"  xsi:type="FilesystemMetadataProvider"
metadataFile="/opt/shibboleth-idp/metadata/saml-auth.xml"/>
```

上記をコピーして追加し、下記の形に変更。

```
   <MetadataProvider id="LocalMetadata2"  xsi:type="FilesystemMetadataProvider"
metadataFile="/opt/shibboleth-idp/metadata/saml-auth_for_GP.xml"/>
```

6.3.3. Relying-party の設定

relying-pary.xml ファイルで、信頼できる SP として、PA Firewall を追加します。
SP メタデータ内の entity ID の値を使って、以下を設定のように設定します。

```
# vi /opt/shibboleth-idp/conf/relying-party.xml
```

```
      <bean parent="RelyingPartyByName"
c:relyingPartyIds="https://trust.acme.com:6082/SAML20/SP">
         <property name="profileConfigurations">
            <list>
               <bean parent="SAML2.SSO" p:encryptAssertions="false" />
            </list>
         </property>
      </bean>
```

上記をコピーして追加し、下記の形に変更。

```
      <bean parent="RelyingPartyByName"
c:relyingPartyIds="https://gp011.japaneast.cloudapp.azure.com:443/SAML20/SP">
         <property name="profileConfigurations">
            <list>
               <bean parent="SAML2.SSO" p:encryptAssertions="false" />
            </list>
         </property>
      </bean>
```

6.3.4. attribute-filter の設定

Shibboleth-IdP の attribute-filter.xml を編集することで、クライアントが認証時に利用した username が渡されるようになります。
GlobalProtect の場合、Portal で認証した username を、Gateway での認証時に受け渡す必要があるので、デフォルトの「NameID」値のままでは認証が完了しません。よって、GlobalProtect の場合は、この設定は必須です。

(1) ファイルを編集します。

```
# vi /opt/shibboleth-idp/conf/attribute-filter.xml
```

```
   <AttributeFilterPolicy id="example1">
      <PolicyRequirementRule xsi:type="Requester"
value="https://trust.acme.com:6082/SAML20/SP" />

      <AttributeRule attributeID="uid" permitAny="true" />
   </AttributeFilterPolicy>

上記をコピーして追加し、以下のように編集

   <AttributeFilterPolicy id="example3">
      <PolicyRequirementRule xsi:type="Requester"
value="https://gp011.japaneast.cloudapp.azure.com:443/SAML20/SP" />

      <AttributeRule attributeID="uid" permitAny="true" />
   </AttributeFilterPolicy>
```

6.3.5. HTTPS サーバ(Apache)の設定

(商用のサーバ証明書を利用しているのであれば、このステップは不要です。)

IdP の HTTPS サーバ (本ガイドでは Apache) のサーバ証明書を、GlobalProtect Agent が信頼できる状態になっていなければ、接続ができません。GlobalProtect Agent のサーバ証明書の検証を無効化できないためです。
それを回避する設定を行います。

6.3.5.1. サーバ証明書の生成

既述のステップで生成済みのサーバ証明書及び秘密鍵を利用することにします。

```
# cd /etc/pki/CA/SERVER/
# ls -l
-rw-r--r-- 1 root root 4515  6 月 13 16:51 samlidp-crt.pem
-r-------- 1 root root 1704  6 月 13 16:51 samlidp-key.pem
-rw-r--r-- 1 root root 1025  6 月 13 16:51 samlidp-req.pem
```

6.3.5.2. ssl.conf の設定

以下のファイルを編集して、HTTPS サーバが上記のサーバ証明書及び秘密鍵を利用できるようにします。

```
# vi /etc/httpd/conf.d/ssl.conf
```

```
(省略)
SSLCertificateFile /etc/pki/tls/certs/samlidp-crt.pem          ←サーバ証明書の格納先
(省略)
SSLCertificateKeyFile /etc/pki/tls/private/samlidp-key.pem     ←サーバ秘密鍵の格納先
(省略)
```

6.3.5.3. httpd 及び Tomcat サービスの再起動

(1) httpd/Tomcat を再起動します。

```
# systemctl stop httpd
# systemctl restart tomcat
# systemctl start httpd
```

(2) Tomcat のログにエラーが出ていないことを確認します。

```
# tail /usr/share/tomcat/logs/catalina.YYYY-MM-DD.log
```

(3) 以下のファイルにもエラーがないことを確認します。

```
# tail /opt/shibboleth-idp/logs/idp-process.log
# tail /opt/shibboleth-idp/logs/idp-warn.log
```

6.3.5.4. GP Agent へ CA 証明書をインポート

① OpenSSL で生成済みの CA 証明書を、GP Agent を持つクライアント PC へコピーします。

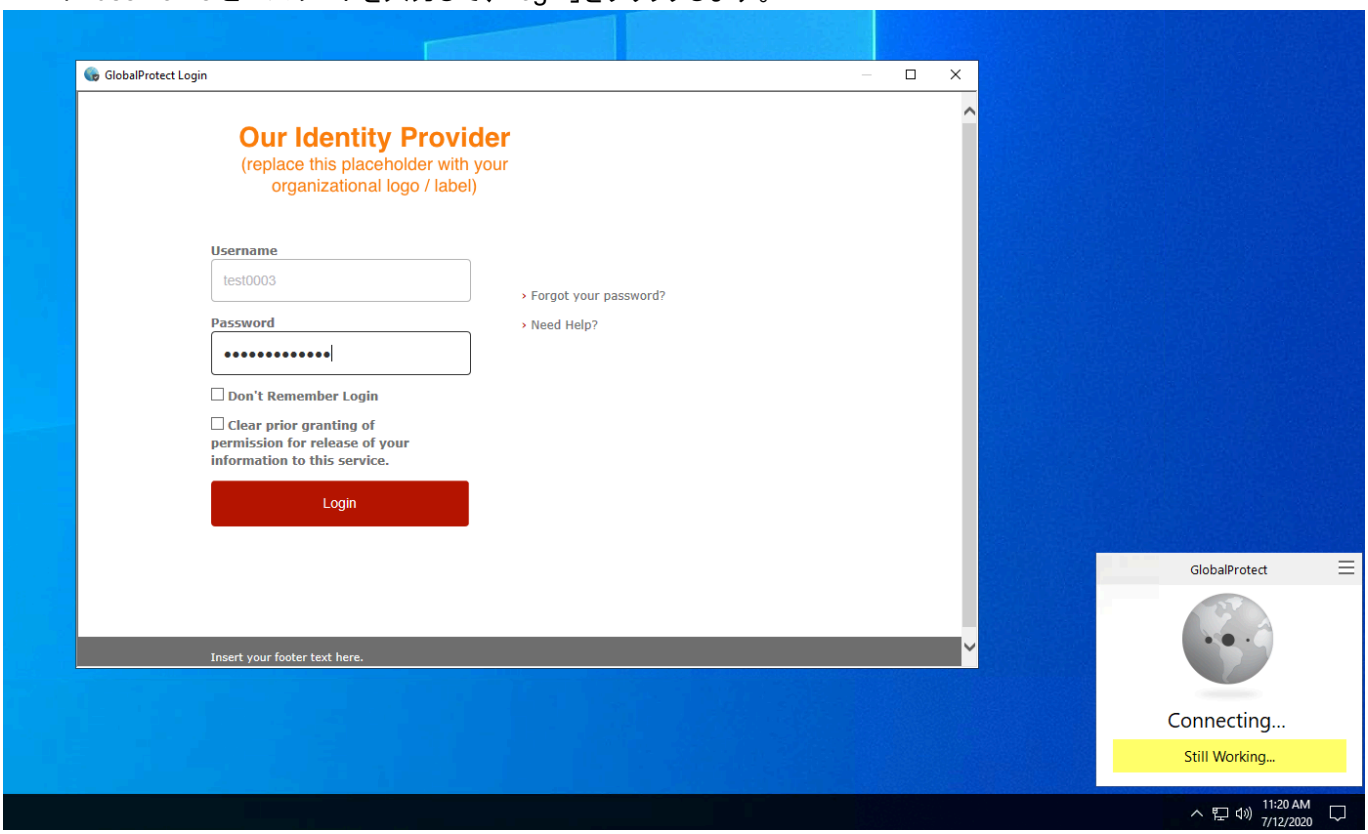
ファイル名：`/etc/pki/CA/cacert.pem`

② この CA 証明書を、クライアント PC の証明書ストア（Windows の例：「信頼できる証明書発行機関」）へインポートします。

6.4. 動作確認

6.4.1. GP Agent からのアクセス

GP Agent を持つクライアント PC から Portal へアクセスすると、以下のような認証フォームが表示されます。
ここに username とパスワードを入力して、「login」をクリックします。



認証が成功すると、接続できます。

7. その他[参考情報]

7.1. LDAP アトリビュートを SAML Response で取得したい場合

LDAP がもつユーザの属性情報(アトリビュート)を、SAML Response で取得したい場合があります。
例えば、Management インタフェースで SAML 認証を行って、アトリビュートで何らかの制御を行いたい場合に有効です。
その設定を行います。

7.1.1. attribute-resolver.xml

(1) デフォルトで存在する attribute-resolver-full.xml を、attribute-resolver.xml に置き換えます。

(2) そのファイルを以下のように書き換えます。

```
# vi  /opt/shibboleth-idp/conf/attribute-resolver.xml
```

```
    <!-- Schema: Core schema attributes-->
<!--    コメントアウトを削除

    <AttributeDefinition xsi:type="Simple" id="uid">
        <InputDataConnector ref="myLDAP" attributeNames="uid"/>
        <AttributeEncoder xsi:type="SAML1String" name="urn:mace:dir:attribute-def:uid" encodeType="false"
/>
        <AttributeEncoder xsi:type="SAML2String" name="urn:oid:0.9.2342.19200300.100.1.1"
friendlyName="uid" encodeType="false" />
    </AttributeDefinition>

    <AttributeDefinition xsi:type="Simple" id="mail">
        <InputDataConnector ref="myLDAP" attributeNames="mail"/>
        <AttributeEncoder xsi:type="SAML1String" name="urn:mace:dir:attribute-def:mail"
encodeType="false" />
        <AttributeEncoder xsi:type="SAML2String" name="urn:oid:0.9.2342.19200300.100.1.3"
friendlyName="mail" encodeType="false" />
    </AttributeDefinition>

    <AttributeDefinition xsi:type="Simple" id="homePhone">
        <InputDataConnector ref="myLDAP" attributeNames="homePhone"/>
        <AttributeEncoder xsi:type="SAML1String" name="urn:mace:dir:attribute-def:homePhone"
encodeType="false" />
        <AttributeEncoder xsi:type="SAML2String" name="urn:oid:0.9.2342.19200300.100.1.20"
friendlyName="homePhone" encodeType="false" />
    </AttributeDefinition>
(省略)
    <AttributeDefinition xsi:type="Simple" id="organizationName">
        <InputDataConnector ref="myLDAP" attributeNames="o"/>
        <AttributeEncoder xsi:type="SAML1String" name="urn:mace:dir:attribute-def:o" encodeType="false"
/>
        <AttributeEncoder xsi:type="SAML2String" name="urn:oid:2.5.4.10" friendlyName="o"
encodeType="false" />
    </AttributeDefinition>
(省略)
-->    コメントアウトを削除

(省略)
    <!-- Example LDAP Connector -->
<!--    コメントアウトを削除
    <DataConnector id="myLDAP" xsi:type="LDAPDirectory"
        ldapURL="%{idp.attribute.resolver.LDAP.ldapURL}"
        baseDN="%{idp.attribute.resolver.LDAP.baseDN}"
        principal="%{idp.attribute.resolver.LDAP.bindDN}"
        principalCredential="%{idp.attribute.resolver.LDAP.bindDNCredential}"
        useStartTLS="%{idp.attribute.resolver.LDAP.useStartTLS:true}"
        connectTimeout="%{idp.attribute.resolver.LDAP.connectTimeout}"
               trustFile="%{idp.attribute.resolver.LDAP.trustCertificates}"          ←削除
        responseTimeout="%{idp.attribute.resolver.LDAP.responseTimeout}">
        <FilterTemplate>
            <![CDATA[
                %{idp.attribute.resolver.LDAP.searchFilter}
            ]]>
        </FilterTemplate>
            <ConnectionPool
            minPoolSize="%{idp.pool.LDAP.minSize:3}"
            maxPoolSize="%{idp.pool.LDAP.maxSize:10}"
            blockWaitTime="%{idp.pool.LDAP.blockWaitTime:PT3S}"
            validatePeriodically="%{idp.pool.LDAP.validatePeriodically:true}"
```

```
        validateTimerPeriod="%{idp.pool.LDAP.validatePeriod:PT5M}"
        expirationTime="%{idp.pool.LDAP.idleTime:PT10M}"
        failFastInitialize="%{idp.pool.LDAP.failFastInitialize:false}" />
    </DataConnector>
→   コメントアウトを削除
```

7.1.2. Attribute-filter.xml

SAML Response で SP へ渡したいアトリビュートを、attribute-filter.xml に追加します。

```
# vi /opt/shibboleth-idp/conf/attribute-filter.xml
```

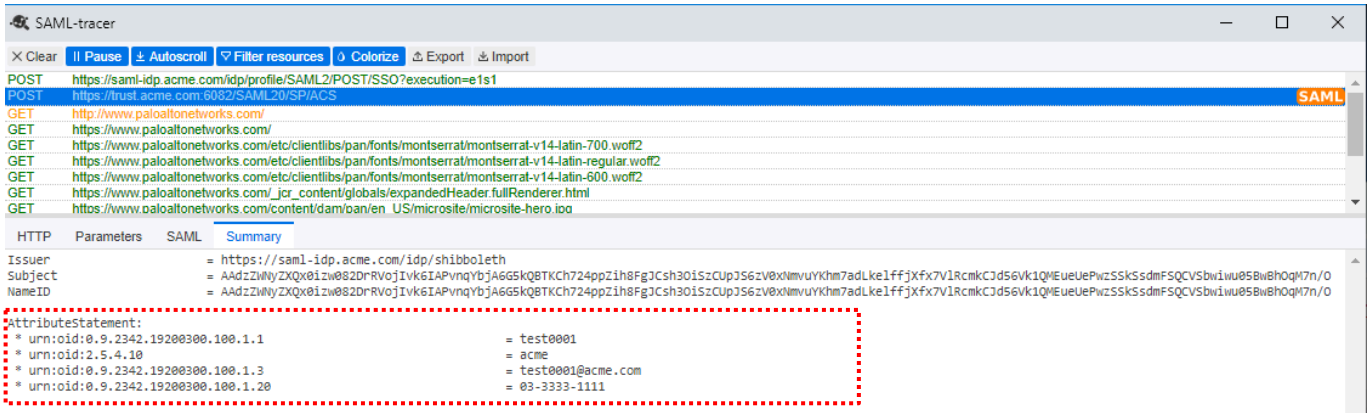
```
<AttributeFilterPolicy id="example1">
   <PolicyRequirementRule xsi:type="Requester" value="https://trust.acme.com:6082/SAML20/SP" />

   <AttributeRule attributeID="uid" permitAny="true" />
   <AttributeRule attributeID="mail" permitAny="true" />
   <AttributeRule attributeID="homePhone" permitAny="true" />
   <AttributeRule attributeID="organizationName" permitAny="true" />
</AttributeFilterPolicy>
```

Tomcat を再起動します。

```
# systemctl restart tomcat
```

7.1.3. 動作確認

(1) クライアント PC で SAML 認証を実施します。
(2) SAML-Tracer で、複数のアトリビュートが SAML Response で SP へ送られていることを確認します。



7.2. 留意事項

7.2.1. Assertion の署名は非対応

/opt/shibboleth-idp/metadata/saml-auth.xml の WantAssertionsSigned="True"にすると、IdP から SP へ渡される SAML Response 内の Assertion に署名が行われます。

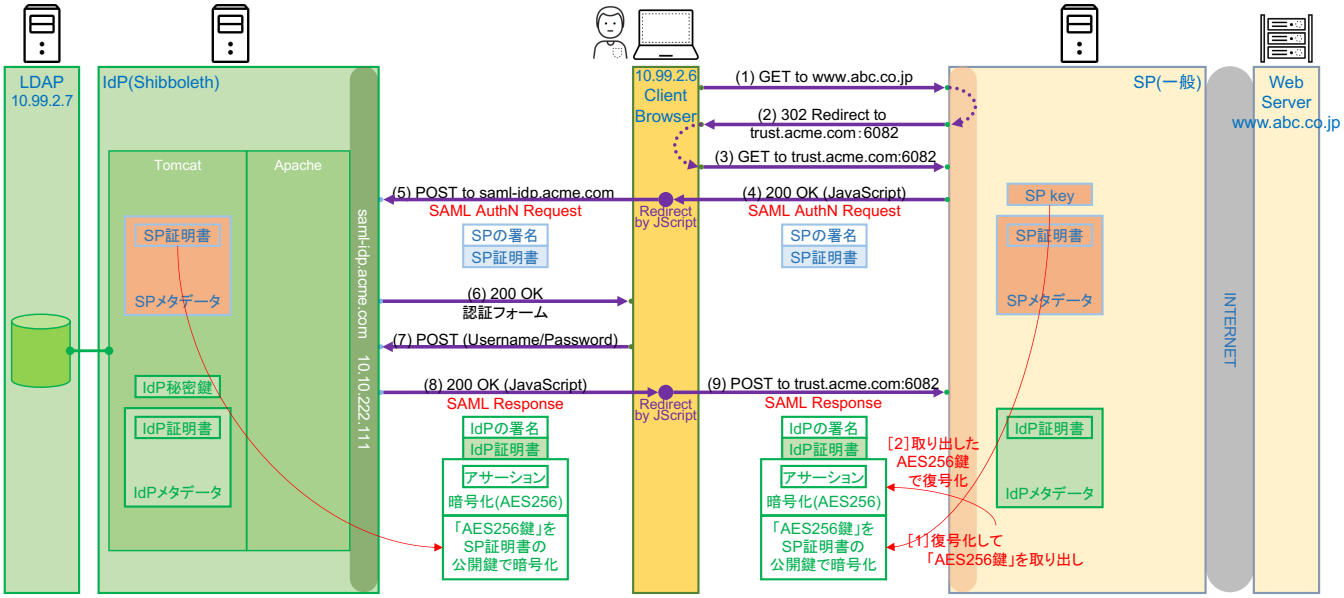
```
<md:EntityDescriptor xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
                     entityID="https://trust.acme.com:6082/SAML20/SP">
  <md:SPSSODescriptor AuthnRequestsSigned="true"
                      WantAssertionsSigned="True"
                      protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
    <md:KeyDescriptor>
```

しかし、SP である PA Firewall は、それをチェックしないので、設定する必要はありませんが、設定されていても問題ありません。
PA が検証しているのは、あくまで「IdP の署名」であり、「Assertion の署名」ではありません。

7.2.2. Assertion の暗号化は非対応

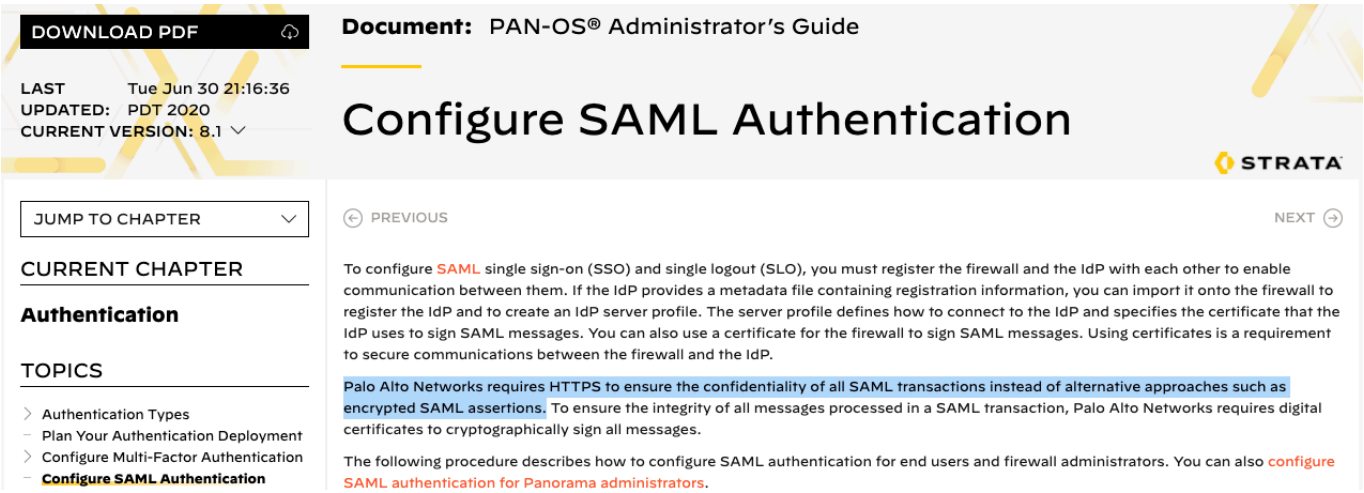
SAML には、Assertion の暗号化機能が規定されています。

Assertion 暗号のイメージ:



PA Firewall は、この Assertion 暗号には対応していませんが、その代わりに SAML トランザクションの通信自体の HTTPS 暗号を強制するアプローチをとっています。

Palo Alto Networks requires HTTPS to ensure the confidentiality of all SAML transactions instead of alternative approaches such as encrypted SAML assertions.



DOWNLOAD PDF

LAST UPDATED: Tue Jun 30 21:16:36 PDT 2020
CURRENT VERSION: 8.1

Document: PAN-OS® Administrator's Guide

Configure SAML Authentication

STRATA

JUMP TO CHAPTER

CURRENT CHAPTER

Authentication

TOPICS

- Authentication Types
- Plan Your Authentication Deployment
- Configure Multi-Factor Authentication
- **Configure SAML Authentication**

PREVIOUS NEXT

To configure SAML single sign-on (SSO) and single logout (SLO), you must register the firewall and the IdP with each other to enable communication between them. If the IdP provides a metadata file containing registration information, you can import it onto the firewall to register the IdP and to create an IdP server profile. The server profile defines how to connect to the IdP and specifies the certificate that the IdP uses to sign SAML messages. You can also use a certificate for the firewall to sign SAML messages. Using certificates is a requirement to secure communications between the firewall and the IdP.

Palo Alto Networks requires HTTPS to ensure the confidentiality of all SAML transactions instead of alternative approaches such as encrypted SAML assertions. To ensure the integrity of all messages processed in a SAML transaction, Palo Alto Networks requires digital certificates to cryptographically sign all messages.

The following procedure describes how to configure SAML authentication for end users and firewall administrators. You can also configure SAML authentication for Panorama administrators.

https://docs.paloaltonetworks.com/pan-os/8-1/pan-os-admin/authentication/configure-saml-authentication.html

8. 参考文献(URL)

学術認証フェデレーション
https://www.gakunin.jp/

Nanako-Network.Informations
https://secure.nanako-net.info/redmine/projects/know-how/wiki/Shibboleth

Qiita
https://qiita.com/Blue2012/items/45c87ac16ad8df69bd89
https://qiita.com/Blue2012/items/d45342ab17592168cd72

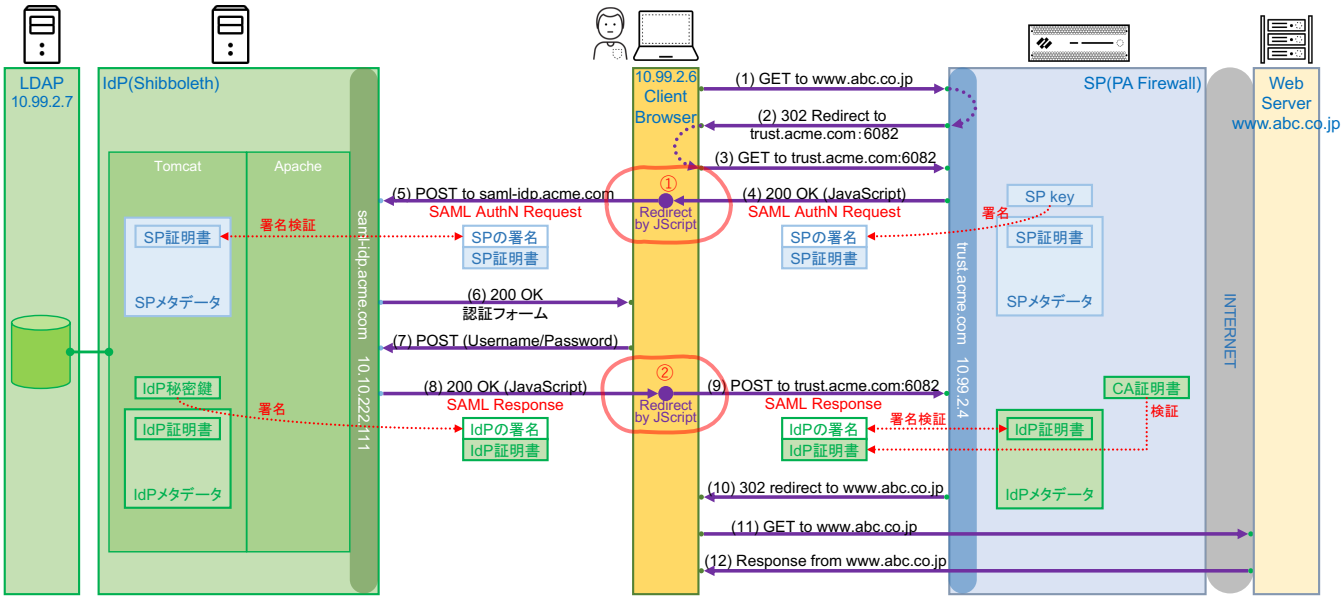
9. Appendix

9.1. LDAP の新規インストール

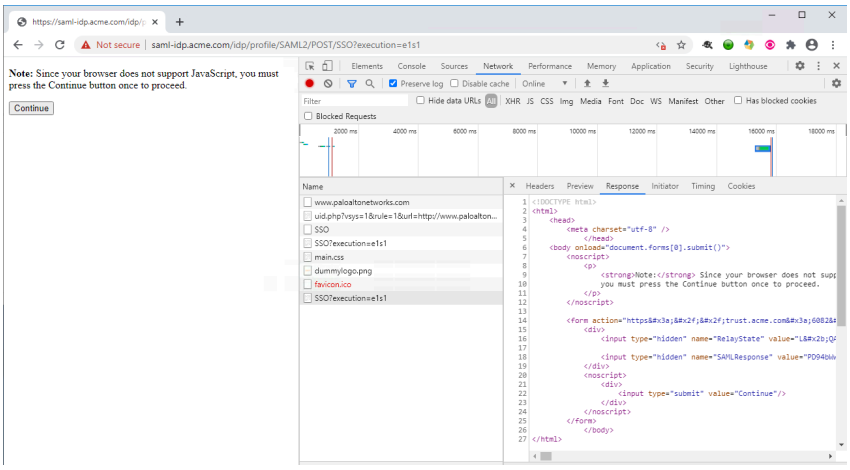
- OpenLDAP インストール
  - https://www.server-world.info/query?os=CentOS_7&p=openldap&f=1
  - 基本的に、この Link に記載されたままの方法で OK です。
- OpenLDAP の初期化 (不具合発生時に初期化したい時)
  - https://www.nomade.biz/mtarchives/2017/01/centos-7-ldap.php
- Apache のインストール (phpLDAPadmin 用)
  - https://www.server-world.info/query?os=CentOS_7&p=httpd&f=1
- PHP のインストール (phpLDAPadmin 用)
  - https://www.server-world.info/query?os=CentOS_7&p=httpd&f=3
- phpLDAPadmin のインストール
  - https://www.server-world.info/query?os=CentOS_7&p=openldap&f=7
- SELINUX の停止
  - vi /etc/selinux/config
  - SELINUX=enforce >> SELINUX=disabled

9.2. JavaScript の確認方法

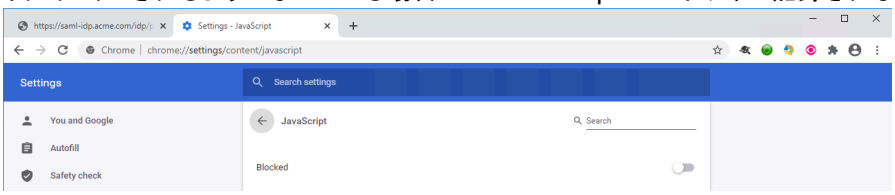
Web ブラウザで動作する JavaScript についての確認方法メモです。



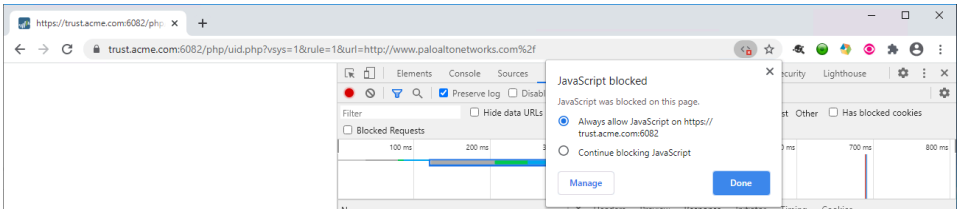
クライアント PC の Chrome ブラウザの「デベロッパーツール」を使って確認。



Chrome Settings の JavaScript を「Blocked」にしておかなければ、Chrome デベロッパーツールに JavaScript で即座にリダイレクトされるようになっている場合の HTTP Response ボディが記録されないので、その点に注意。



JavaScript の「Blocked」の影響で次に進めない場合は、JavaScript が確認できたら、以下の処理(Always allow ~)を行ってから、再度 SAML 認証を実施。



① SP→IdP (SAML AuthN Request)

```
<html>
<head>
<style>
#dError {
   color: #E10000;
   margin-top: 10px;
   padding-top: 10px;
   padding-bottom: 10px;
   margin-left: 10px;
   margin-right: 10px;
   padding-left: 10px;
   padding-right: 10px;
   font-weight: bold;
   overflow: auto;
}
</style>
</head>
<body>
<form id="myform" method="POST" action="https://saml-idp.acme.com/idp/profile/SAML2/POST/SSO">
<input type="hidden" name="SAMLRequest"
value="PHNhbWxwOkF1dGhuUmVxdWVzdCB4bWxuczpzYW1scD0idXJuOm9hc2lzOm5hbWVzOnRjOlNBTUw6Mi4wOnByb3RvY29sIiBBc3NlcnR
pb25Db25zdW1lclNlcnZpY2VVUkw9Imh0dHBzOi8vdHJ1c3QuYWNtZS5jb206NjA4Mi9TQU1MMjAvU1AvQUNTIiBEZXN0aW5hdGlvbj0iaHR0c
～略～
lFkZEEzM3N6SlorQT09CjwvZHM6WDUwOUNlcnRpZmljYXRlPjwvZHM6WDUwOURhdGE+PC9kczpLZXlJbmZvPjwvZHM6U2lnbmF0dXJlPjwvc2F
tbHA6QXV0aG5SZXF1ZXN0Pg==" />
<input type="hidden" name="RelayState" value="GuMAAG1LBV9BQUVBQUFBQkFBQUFBQUFBQUFveE1DNDVPUzR5TGpZQQ==" />
<div id="dError" class="dError" ></div>
</form>
<script>
try {
   if(typeof(Storage) != "undefined") {
      var orig_url = "http://www.paloaltonetworks.com/";
      sessionStorage.setItem ("isoffline", 0);
      if(orig_url != "")
         sessionStorage.setItem("origurl", orig_url);
   }
   document.getElementById('myform').submit();
}catch(e) {
   var privatebrowsing = "This page is not compatible in private browsing.       <br> Please turn off private
browsing and reload this page";
   derr = document.getElementById('dError');
   derr.innerHTML=privatebrowsing;
}
</script>
</body>
</html>
```

② IdP → SP (SAML Response)

```
<html>
   <head>
      <meta charset="utf-8" />
         </head>
   <body onload="document.forms[0].submit()">
      <noscript>
         <p>
            <strong>Note:</strong> Since your browser does not support JavaScript,
            you must press the Continue button once to proceed.
         </p>
      </noscript>

      <form action="https&#x3a;&#x2f;&#x2f;trust.acme.com&#x3a;6082&#x2f;SAML20&#x2f;SP&#x2f;ACS"
method="post">
         <div>
            <input type="hidden" name="RelayState"
value="L&#x2b;QAAG1LBV9BQUVBQUFBQkFBQUFBQUFBQUFveE1DNDVPUzR5TGpZQQ&#x3d;&#x3d;"/>

            <input type="hidden" name="SAMLResponse"
value="PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiPz4KPHNhbWwycDpSZXNwb25zZSBEZXN0aW5hdGlvbj0iaHR0cHM6Ly9
0cnVzdC5hY21lLmNvbTo2MDgyL1NBTUwyMC9TUC9BQ1MiIElEPSJfOGEwZmM5MTBhZTdmM2NiZjU0ZWE3YTdjYmVlY2ZmM2QiIEluUmVzcG9uc
～略～
yOkF0dHJpYnV0ZVZhbHVlPjwvc2FtbDI6QXR0cmlidXRlPjwvc2FtbDI6QXR0cmlidXRlU3RhdGVtZW50Pjwvc2FtbDI6QXNzZXJ0aW9uPjwvc
2FtbDJwOlJlc3BvbnNlPg=="/>
         </div>
         <noscript>
            <div>
               <input type="submit" value="Continue"/>
            </div>
         </noscript>
      </form>
         </body>
</html>
```