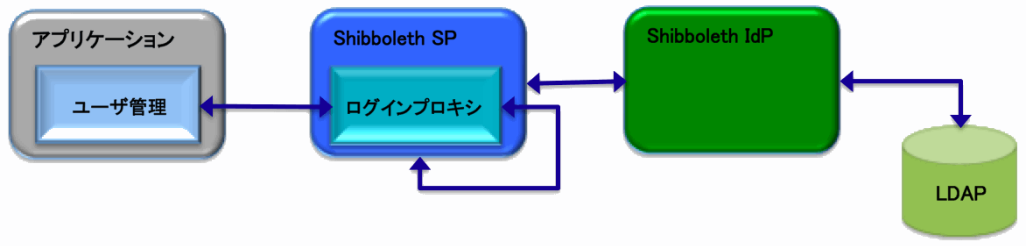
既存のアプリケーションのユーザ管理へのProxy（代行システム）を用意する

既存のアプリケーションのユーザ管理へのProxy（代行システム）を用意する

既存のアプリケーションにユーザ管理機能が存在する場合、ユーザ認証をShibbolethに代行させる機能（ログインプロキシ）を構築するパターンです。

概要図



ユーザ管理には、Shibbolethのセッション情報からユーザ管理機能のセッション情報を作成する機能を組み込む必要があります。

Shibbolethにユーザ認証させるため、ログインプロキシをShibboleth SP配下に配備します。

ユーザ管理では、ユーザ認証後にShibbolethから返されるセッション情報をもとにユーザ管理機能のセッション情報を作成します。

Shibboleth SP側の設定

Apacheの設定ファイルhttpd.conf 、.htaccessあるいは、shib.conf（rpmでインストールした場合のみ）の何れかにLocationを追加することで行います。

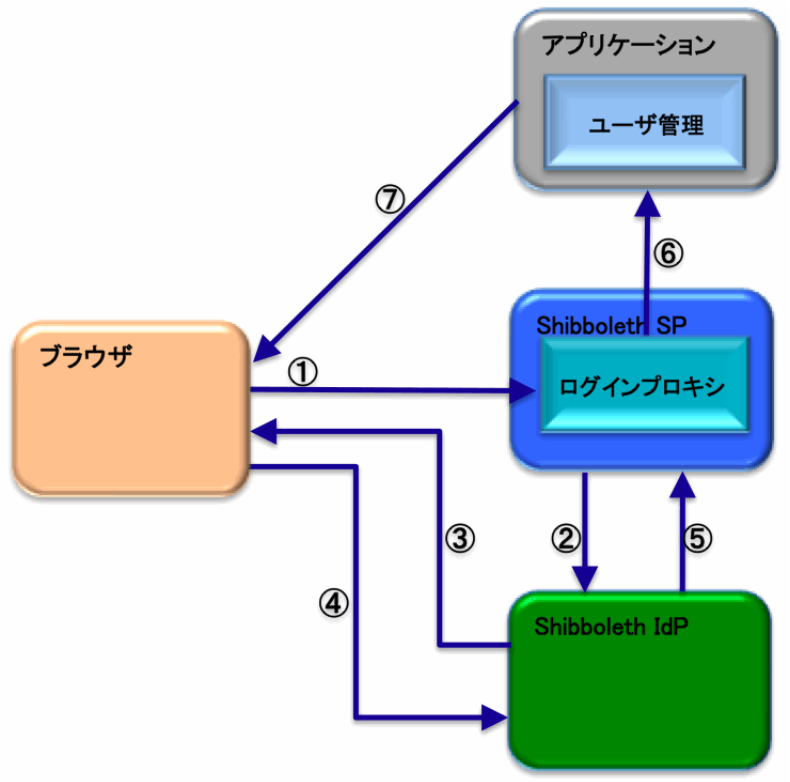
※/etc/shibboleth/shibboleth2.xmlファイルのRequestMapper要素にtype="Native"が設定されている場合に有効です。

設定例）　「AppLoginProxy」をShibboleth化するための設定例

```
<Location /AppLoginProxy>
  AuthType shibboleth
  ShibCompatWith24 On
  ShibRequestSetting requireSession true
  Require shib-session
</Location>
```

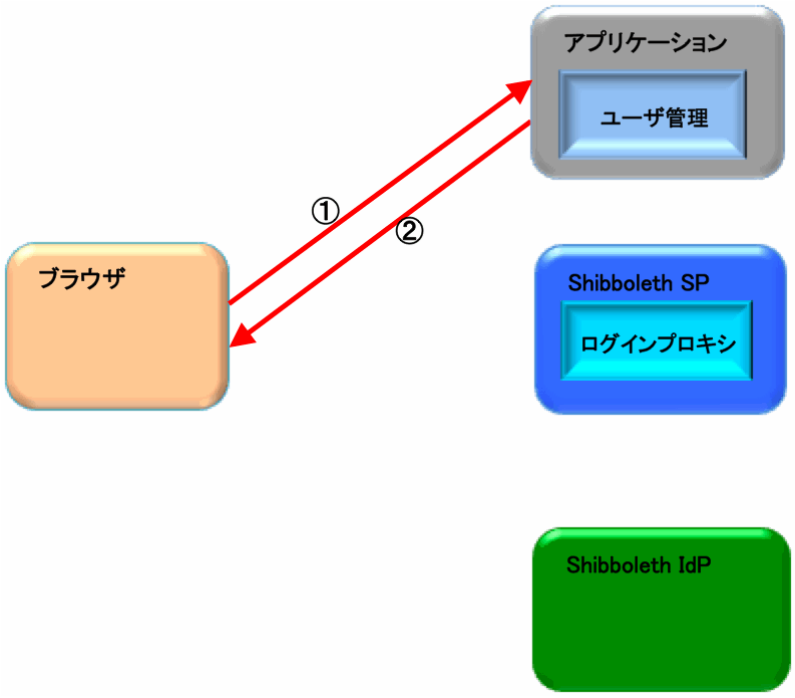
この設定により、/AppLoginProxy下の全リソースは、Shibbolethにより認証されます。

処理イメージ図：初回起動時



①ユーザ認証を受けた権限で、アプリケーションを利用するために、ログインプロキシにアクセスします。
②ログインプロキシによりShibboleth SPからShibboleth IdPの認証画面にリダイレクトされます。
③ブラウザに認証画面を表示します。
④認証画面にユーザ／パスワードを入力し、Shibboleth IdPで認証を行います。
⑤認証結果をShibboleth SPに返す。
⑥ログインプロキシは、Shibbolethのセッション情報と共にアプリケーションのトップページにリダイレクトします。
⑦認証が成功した場合は、アプリケーションのトップページでは、ユーザ管理機能により、Shibbolethのセッション情報をもとにユーザ管理機能のセッション情報を作成し、アプリケーションのトップページを表示します。
認証が失敗した場合は、認証失敗画面を表示します。

処理イメージ図：初回起動後にアプリケーションセッションが存在する場合



①トップページから目的のアプリケーションにアクセスします。
②既にShibboleth認証されているユーザからのアクセスであるため、Shibboleth認証は行わずに
　アプリケーションを実行し、結果をブラウザに返します。

サンプルコード

サンプルコードとして、既存のアプリケーションがユーザー管理を実装している場合の対応例を示します。

「ログインプロキシ」アプリケーションでは、Shibbolethのセッション情報から、既存アプリケーションのセッション情報を生成します。
またこのサンプルでは以下を前提します。

既存のアプリケーションでは認証ずみのセッション情報をクライアントのブラウザのCookieに保存しているものとします。
また、セッション情報のCookie名称は 'session_cookie'

ログインプロキシは、既存アプリと同じドメインに配置される必要があります。
上記Cookieを既存アプリと共有するためです。
Cookieの保存ドメインが共通化できれば、FQDNレベルでの一致は必要ありません。
ここでは、'application.example.net' とします。

ログインが成功した後に表示される、サービスのトップページを'http://application.example.net/' とします。

PHPのサンプルでは、既存アプリとの連携部にPEARのHTTP_Requestクラスライブラリを利用しています。

```
<?php
  require_once "HTTP/Request.php";

  /* Shibbolethの認証状態の確認
  #  このアクションが実行された時点で、Shibbolethのセッションは確立されています。
  #  必要に応じてSPの返却したパラメータを参照して権限の確認等を行います。
  #  SPの返却したパラメータは、HTTP環境変数に追加されています。 */
  if(!$_SERVER['persistent-id']){
    // eduPersonTargetedID が存在しない場合は認証エラー
    header("HTTP/1.1 403 Forbidden");
    exit;
  }

  /* 既存アプリケーションのセッション情報を生成する
  #  この部分の実装は既存アプリケーションの構成に依存します。
  #  ここでは、'http://application.example.net/login' に諸元をPOSTすると、セッションが生成され
  #  クライアントに返却すべきCookieが得られるものとします。 */
  $http =& new HTTP_Request("http://application.example.net/login");
  $http->setMethod(HTTP_REQUEST_METHOD_POST);
  $http->addPostData("user_id", "shibboleth_user");
  $http->addPostData("password", "shibboleth_password");
  $app_response = $http->sendRequest();
  $app_cookies = $http->getResponseCookies();

  // 取得されたアプリケーションのセッションCookieをレスポンスに設定する
  setcookie('session_cookie', $app_cookies["session_cookie"], 0, "/", "application.example.net");

  // ログイン後のトップページへリダイレクトする
  header("HTTP/1.1 302 Found");
  header("Location: http://application.example.net");
?>
```

```
#!/bin/env ruby
# encoding : utf-8
require 'webrick/cgi'
require 'net/http'
Net::HTTP.version_1_2

class LoginProxy < WEBrick::CGI
  # GET request
  def do_GET(request, response)
    # Shibbolethの認証状態の確認
    #  このアクションが実行された時点で、Shibbolethのセッションは確立されています。
    #  必要に応じてSPの返却したパラメータを参照して権限の確認等を行います。
    #  SPの返却したパラメータは、HTTP環境変数に追加されています。
    unless ENV.key?('persistent-id')
      # eduPersonTargetedID が存在しない場合は認証エラー
      response.staus = 403 # 403 Forbidden
      return
    end

    # 既存アプリケーションのセッション情報を生成する
    #  この部分の実装は既存アプリケーションの構成に依存します。
    #  ここでは、'http://application.example.net/login' に諸元をPOSTすると、セッションが生成され
    #  クライアントに返却すべきCookieが得られるものとします。
    http = Net::HTTP.start('application.example.net', 80)
    app_response = http.post('/login', 'user_id=shibboleth_user&password=shibboleth_password')
    app_cookies = app_response.get_fields('Set-Cookie').inject({}) do |cookies, cookie_string|
      key, value = cookie_string[/[^;]*/].split('=')
      cookies[key] = value
    end

    # 取得されたアプリケーションのセッションCookieをレスポンスに設定する
    session_cookie = WEBrick::Cookie.new('session_cookie', app_cookies['session_cookie'])
    session_cookie.path = '/'
    session_cookie.domain = 'application.example.net'
    response.cookies << session_cookie

    # ログイン後のトップページへリダイレクトする
    redirect_url = 'http://application.example.net'
    response.set_redirect(WEBrick::HTTPStatus::Found, redirect_url)
  end

  alias do_POST do_GET
end

LoginProxy.new.start
```

Apache設定ファイルの編集（httpd.confに設定する場合）

/etc/httpd/conf.d/httpd.conf に下記コードを追加します。

```
<Location /proxyApp>
  AuthType shibboleth
  ShibCompatWith24 On
  ShibRequestSetting requireSession true
  Require shib-session
</Location>
```