国立情報学研究所

学認連携
クライアント証明書発行システム
Docker コンテナ構築マニュアル

２０２０年３月２４日版

□ 目次

□ 履歴

日付	改定内容	作成者
2020/03/24	初版	宮地/村尾

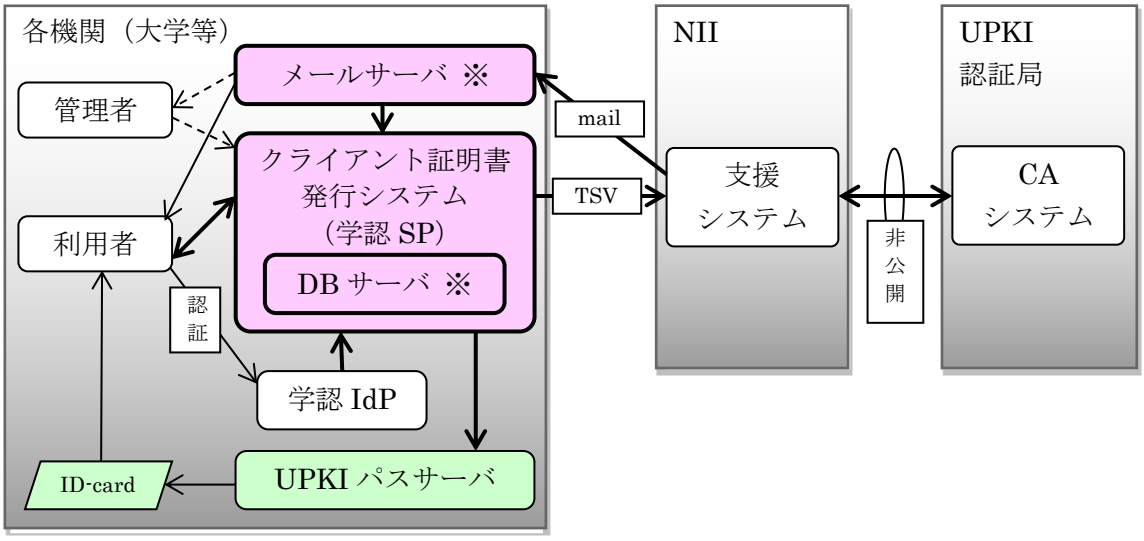
１．概要

本 Docker コンテナ構築マニュアルは、学認連携クライアント証明書発行システム（以後、クライアント証明書発行システム）を新規サーバに Docker コンテナとして構築する手順を示すものである。クライアント証明書発行システムは認証部に学術認証フェデレーション（以後、学認）を利用して認証時に必要な属性情報（メールアドレスや名前等）を取得する。認証後は、既存の UPKI 電子証明書自動発行支援システム（以後、支援システム）に対してクライアント証明書の発行依頼を行い、結果を取得して利用者サービスを提供する。これにより証明書発行申請作業を管理者が行わずに済むシステムである。

前提として既存の学認 IdP（ID プロバイダー）と支援システムへの接続が可能である必要がある。以下にシステム概要図を示す。メールサーバは支援システムからの結果取得の為に必要となる。メール受信はクライアント証明書発行システムが兼用することが望ましいが、外部のメールサーバから REST API で連携する仕組みも提供している。

Docker を構築する場合にはデータベースとして MySQL を利用していると、クライアント証明書発行システムと同じコンテナでの稼働が難しい。この場合には別途外部に存在する MySQL サーバ（DB サーバ）か別コンテナとして MySQL を稼働させる必要がある。

UPKI パスの発行を利用するには UPKI パスサーバが必要となる。UPKI パス用の証明書の発行依頼をすることにより、自動的に UPKI パス証明書サーバへの登録が実行される仕組みを提供され、UPKI パスの IC カード発行の手順のうち登録までの自動化が実現できる。UPKI パスの利用はオプションであり使わない場合には考える必要はない。本 Docker コンテナ構築マニュアルでは UPKI パスの説明は省略する。



全システム概要と関連図（※別メールサーバと DB サーバはオプション）

１．１．動作確認済み環境

クライアント証明書発行システムのホストサーバで利用

URL	: https://g-cert-dev.gakunin.nii.ac.jp/ （プロトタイプサーバ）
OS	: CentOS 7.5
Docker-CE	: 19.03.6
MySQL	: 5.5.60（yum でインストール・Rails の production 環境用）

クライアント証明書発行システム（Docker コンテナ）内部で利用

OS	: centos7
Apache	: 2.4.6（TLS 接続用のサーバ証明書付）
Shibboleth	: 3.0.4（テストフェデレーション SP として登録済み）
Ruby	: 2.5.7（rbenv 2.5.7 により個別インストール）
Rails	: 4.2.10（ruby gem により個別インストール） （Gemfile 指定により実際には Rails 4.0.5 を利用、Shibcert と同じ設定）
Passenger	: 6.0.4（ruby gem により個別インストール）
Sqlite3	: 3.7.17（Rails の development 環境用）

１．２．動作条件

１）学術認証フェデレーション（学認）
学術認証フェデレーションの SP として登録済みであり設定が可能であること。また必要となる TLS 証明書を取得済みであること。

２）UPKI 電子証明書自動発行支援システム
支援システムに接続する為の管理者用クライアント証明書が発行済みであること。

３）メールサーバ（別サーバも可）
登録メールアドレスにて.forward が使える環境を用意できること。
※1 クライアント証明書発行システムがメール受信機能設定済みであること。
※2 別途メールサーバにて.forward が使える環境でも REST API で利用可能。

４）データベースサーバ（MySQL：別サーバが必要）
クライアント証明書発行システムは Rails の Development 環境では SQLite を利用しているのでデータベースサーバは不要となる。Production 環境では MySQL を利用している為に別途データベースサーバが必要となる。

Rails 環境	利用 DB	DB サーバ	補足
Development	SQLite	不要	SQLite はファイルベースである為に簡易に利用が可能。別 DB サーバも不要であり試験用に向いている。
Production	MySQL	必要	MySQL は接続する為の別 DB サーバが必要となる。本稼働時に向いている。

MySQL の環境をクライアント証明書発行システムの Docker コンテナ内に含めることにも挑戦してみたが、仕組みが複雑になり過ぎる問題と共に起動と同期の問題を生じた。一般的に Docker でシステムを構築する場合には、MySQL コンテナを別に用意して連携して利用することが推奨されている。今回は提供中システムのディスク容量が不足した為に Docker ホスト側で稼働している MySQL をそのまま利用して動作確認を行った。

Production 環境で稼働させる場合には別途 MySQL コンテナとの接続か別途用意したデータベースサーバとの接続が必要となる点に注意が必要である。Development 環境を利用するとデータベースサーバが不要になるので、簡易に試験を行いたい場合には Development 環境での利用を推奨する。

１．３．ファイル構成

圧縮ファイル : gakunin-cert-docker-all.20200323.tar.gz （以下、展開後のファイル構成）

フォルダ名／ファイル名	内容	補足
gakunin-cert-docker	ルートフォルダ	
Dockerfile_shib20200319-pro.txt	Production 環境用の Dockerfile	
Dockerfile_shib20200319-dev.txt	Development 環境用の Dockerfile	
database.yml	Rails/DB 用の設定ファイル	
gakunin-cert-docker.20200320.tar.gz	UPKI 証明書発行システム本体部	
gakunin-cert-top.add.txt	httpd.conf 共通設定用	
gakunin-cert-top.pro.conf	httpd.conf の Production 環境設定用	
gakunin-cert-top.dev.conf	httpd.conf の Development 環境設定用	
httpd-shibd-foreground	httpd および shibd サービス起動用	
mycerts	各種証明書用フォルダ	
certificates	支援システム用の証明書フォルダ	3.3 章
client.cer	支援システム用の証明書（公開鍵）	
client.key	支援システム用の証明書（秘密鍵）	
private	TLS 秘密鍵用フォルダ	
server.key	サーバ TLS 証明書（秘密鍵）	
certs	TLS 公開鍵用フォルダ	
server.crt	サーバ TLS 証明書（公開鍵）	
server-chain.crt	サーバ TLS 証明書（上位証明書）	
omniauth.rb	認証部の設定用	3.7 章,
passenger.conf	Passenger の設定用	
shibd.conf	httpd.conf の Shibboleth 設定用	3.1 章
shibboleth	Shibboleth 設定用フォルダ	3.1 章
attribute-map.xml	attribute-map.xml	
attribute-policy.xml	attribute-policy.xml	
cert	Shibboleth 用証明書フォルダ	
gakunin-test-signer-2011.cer	テストフェデレーション用証明書	
server.crt	サーバ TLS 証明書（公開鍵）	
server.key	サーバ TLS 証明書（秘密鍵）	
native.logger	native.logger ファイル	
shibboleth2.xml	Shibboleth 用設定ファイル	
shibd.logger	shibd.logger ファイル	
shibcert.yml	UPKI 証明書発行システム本体の設定ファイル	3.2-3.6, 3.8 章
ssl.conf	Apache 設定ファイル(TLS 設定用)	
※1 ピンク の項目は利用する環境に合わせて設定が必要となるファイルです。		
※2 グリーン の項目は事前に準備（入手）が必要となるファイルです。		

２．インストール手順

※ CentOS 下で sudo が使えるシェルであることを前提とする。

２．１．Docker のインストール

Docker を system-wide にインストール

```
// DockerCE のインストール
$ sudo yum remove docker docker-common docker-selinux docker-engine
$ sudo yum install -y yum-utils device-mapper-persistent-data lvm2
$ sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
$ sudo yum makecache fast
$ sudo yum install docker-ce docker-ce-cli

// Docker の起動・サービス登録
$ sudo systemctl start docker
$ sudo systemctl enable docker
```

２．２．各種ファイルの準備

２．２．１．コンテナ対応・クライアント証明書発行システムのソース展開

事前にクライアント証明書発行システム gakunin-cert-docker.20200324.tar.gz をインストール先のサーバに配置します。

```
// インストール先に移動（任意で良いがここでは /home/docker の下としている）
$ mkdir /home/docker
$ cd /home/docker

// 圧縮ファイルの解凍
$ cp /<任意の場所>/gakunin-cert-docker.20200324.tar.gz .
$ tar xvfz gakunin-cert-docker.20200324.tar.gz
$ cd gakunin-cert-docker
```

２．２．２．構築するコンテナ環境（Production/Development）の選択

```
// Dockerfile の編集
$ cd /home/docker/gakunin-cert-docker/

【Production 環境を構築する場合】
$ cp Dockerfile_pro Dockerfile

【Development 環境を構築する場合】
$ cp Dockerfile_dev Dockerfile
```

２．２．３．Apache（httpd）設定ファイルの配置

```
// ssl.conf 設定ファイルの配置
$ cd /home/docker/gakunin-cert-docker/
$ vi ssl.conf
★ここで TLS に必要な設定を行う

$ cd /home/docker/gakunin-cert-docker/mycerts/certs
★ここに ssl.conf で設定したサーバ証明書(公開鍵)・中間証明書を配置する
例)
・server.crt
・server-chain.crt

$ cd /home/docker/gakunin-cert-docker/mycerts/private
★ここに ssl.conf で設定したサーバ証明書(秘密鍵)を配置する
例)
・server.key
```

２．２．４．Shibboleth（shibd）設定ファイルの配置

```
// Shibboleth 設定ファイルの配置
$ cd /home/docker/gakunin-cert-docker/shibboleth
★ここに shibboleth の基本設定※１として必要な設定済みの設定ファイルを配置
・attribute-map.xml
・attribute-policy.xml
・native.logger
・shibboleth2.xml
・shibd.logger

$ cd /home/docker/gakunin-cert-docker/shibboleth/cert
★ここに shibboleth の基本設定※１として必要な証明書ファイルを
  それぞれ下記ファイル名（固定）で配置
・gakunin-test-signer-2011.cer
・server.crt
・server.key

$ vi shibd.conf
★３．１．章を参照し適切に編集もしくは変更済み shibd.conf をここに配置
```

※１参考 URL：https://meatwiki.nii.ac.jp/confluence/display/GakuNinShibInstall/SP

２．２．５．UPKI 証明書発行システム本体（rails）設定ファイルの編集/配置

```
$ cd /home/docker/gakunin-cert-docker/
$ vi shibcert.yml
★３．３．～３．６．章と３．８．章を参照し適切に編集もしくは変更済み shibcert.yml を配置
```

２．２．６．支援システム用の証明書と秘密鍵の設定

```
// 取得済みの支援システム用の証明書と秘密鍵の配置
$ cd /home/docker/gakunin-cert-docker/mycerts/certificates
★ここに３．３．章に記載の手順で作成し、それぞれ client.cer と client.key というファイル名(固定)で mycerts/certificates フォルダに配置
・client.key
・client.cer
```

２．２．７．Rails/DB 用の設定（Production 環境用）

※ Development 環境構築の場合は不要

```
// DB 設定ファイルの編集
$ cd /home/docker/gakunin-cert-docker
$ vi database.yml
★ここで production の項目を接続する MYSQL サーバ上のデータベース・ユーザに合わせて編集
production: 以下の主な変更点
 database: ＜データベース名＞
 username: ＜ユーザアカウント＞
 password: ＜パスワード＞
 host: ＜MYSQL のサーバ IP アドレス＞
```

２．２．８．認証関係の設定

```
// DB 設定ファイルの編集
$ cd /home/docker/gakunin-cert-docker
$ vi omniauth.rb
★３．２．章と３．７．章を参照し適切に編集もしくは変更済み omniauth.rb をここに配置
```

※ そのままでも構わない。

２．３．Docker コンテナの構築・起動

２．３．１．Docker コンテナのビルドと起動

```
// インストール先に移動
$ cd /home/docker/gakunin-cert-docker

// Docker コンテナのビルド【コンテナイメージ名：gakunindocker】
$ sudo docker build -t gakunindocker .

// Docker コンテナの起動【コンテナ名：g-cert-docker】
$ sudo docker run -d -it -p 443:443 -p 80:80 --name g-cert-docker gakunindocker

// Docker コンテナのシェルへログイン
$ sudo docker exec -it g-cert-docker bash

//  （以下、Docker コンテナ内のシェル操作が可能となる）
[root@<container_id>]#

// ログアウトする場合
[root@<container_id>]# exit

// コンテナ内の設定を変更してサービスを再起動する場合
[root@<container_id>]# /usr/local/bin/httpd-shibd-foreground
```

２．３．２．Docker コンテナの停止と再開

```
// Docker コンテナの停止
$ sudo docker stop g-cert-docker

// Docker コンテナの開始
$ sudo docker start g-cert-docker

// Docker コンテナの削除
$ sudo docker stop g-cert-docker
$ sudo docker rm g-cert-docker

// Docker コンテナイメージの削除
$ sudo docker stop g-cert-docker
$ sudo docker rm g-cert-docker
$ sudo docker rmi gakunindocker
```

２．３．３．Docker コンテナのログの確認

```
// Docker コンテナのログの確認
$ sudo docker logs g-cert-docker

// Docker コンテナのログの確認（継続）
$ sudo docker logs g-cert-docker -f
```

３．設定情報

３．１．学認 Shibboleth（開発用 GitHub）認証の設定

※ 学認 SP が設定済みであり、必要な属性取得が許可されていること。
最低必要な属性：ePPN, displayName, mail

```
// Shibboleth 用の設定作成（既にある場合は編集）
$ sudo vi shib.conf
---(以下追加)---
# Load the Shibboleth module.
LoadModule mod_shib /usr/lib64/shibboleth/mod_shib_24.so

# Turn this on to support "require valid-user" rules from other
# mod_authn_* modules, and use "require shib-session" for anonymous
# session-based authorization in mod_shib.
ShibCompatValidUser Off

# Ensures handler will be accessible.
<Location /Shibboleth.sso>
  SetHandler shib
  AuthType None
  Require all granted
</Location>

# Used for example style sheet in error templates.
<IfModule mod_alias.c>
  <Location /shibboleth-sp>
    AuthType None
    Require all granted
  </Location>
  Alias /shibboleth-sp/main.css /usr/share/shibboleth/main.css
</IfModule>

# Configure the module for content.
<Location /auth/shibboleth>
  AuthType shibboleth
  ShibRequestSetting requireSession 1
  require shib-session
</Location>
---(ここまで)---
```

※ Shibboleth SP 用の属性設定サンプルが以下にあるので参考にする。
gakunin-cert/sys_sample/shibboleth/attribute-map.xml

※ Shibboleth SP の動作確認（以下 URL にアクセス）
https://(あなたサーバ名)/auth/shibboleth/index.php

３．２．開発用 GitHub 認証の設定（オプション）

※ GitHub は開発用に必要であるが運用時には利用しない。

※ 以下より GibHub のクライアント登録を行う。
https://github.com/settings/developers

参考：GitHub への登録内容（g-cert-dev.gakunin.nii.ac.jp）

Application name
gakunin-cert

Homepage URL
https://g-cert-dev.gakunin.nii.ac.jp/

Application description
GakuNin Cert Enrollement.

Authorization callback URL
https://g-cert-dev.gakunin.nii.ac.jp/auth/github/callback

参考：GitHub 登録後に発行される ID とシークレット（g-cert-dev.gakunin.nii.ac.jp）

Client ID
f59271dcf3ffe5e54931

Client Secret
c0c06d04e96064823a648d9370d3f8da6a6468e8

設定ファイル：omniauth.rb

項目	GitHub 設定	補足
provider :github	開発（GitHub）用設定	
最初の項目	GitHub の OAuth 認証のクライアント ID	上例 Client ID
2 番目の項目	GitHub の OAuth 認証のシークレット	上例 Client Secret

３．３．UPKI 電子証明書自動発行支援システムの設定

※ 支援システム用の証明書と秘密鍵を PKCS#12 形式で取得済みであること。
※ 支援システム用証明書発行時のドメインと S/MIME 証明書のドメインは一致している必要があるので注意が必要。

1）OpenSSL を使って P12 ファイルを証明書と鍵ファイルに変換

```
// 取得した PKCS#12 ファイル名が getcert.p12 の場合

// 秘密鍵ファイルの作成
# openssl pkcs12 -in getcert.p12 -nocerts -nodes -out client.key

// 証明書ファイルの作成
# openssl pkcs12 -in getcert.p12 -clcerts -nokeys -out client.cer
```

2）Docker のクライアント証明書発行システムへのセット

```
// 秘密鍵ファイルの配置
# cp client.key gakunin-cert-docker/mycerts/certificates

// 証明書ファイルの配置
# cp client.cer gakunin-cert-docker/mycerts/certificates/certificates
```

設定ファイル：shibcert.yml

項目		内容	補足
production:		リリース運用設定	Rails 起動オプションで指定
development:		開発用設定	Rails 起動オプションで指定
	certificate_file:	支援システム接続用証明書ファイル	config/certificates/client.cer
	certificate_key_file:	支援システム接続用秘密鍵ファイル	config/certificates/client.key

３．４．支援システムからのメール受信の設定

A) リモート設定の場合

クライアント証明書発行システムでは指定されたメールアドレスが取得できない場合は、メールアドレス取得可能なサーバにてリモート設定を行う。Basic 認証用の設定は config/shibcert.yml にある。

```
// .forward ファイルのセット（事前に_forward.remote-sample を転送しておくこと）
$ cp _forward.remote-sample .forward

// .forward 中の Basic 認証のユーザ ID とパスワードを修正する
$ vi .forward
```

参考：.forward 例

```
| "base64" | "curl -u gcertsys:8aBEpQ7r31
 https://g-cert-dev.gakunin.nii.ac.jp/mail/processor -X POST
 -H 'Content-Type: text/plain' -k -d @-"
```

設定ファイル：shibcert.yml

項目		内容	設定例
production:		リリース運用設定	
development:		開発用設定	
	mail_basic_name:	メール連携時 Basic 認証のユーザ名	'gcertsys'
	mail_basic_pswd:	メール連携時 Basic 認証のパスワードの SHA-1 ハッシュ値の HEX 化文字列	

※ パスワードから SHA-1 ハッシュの HEX 文字列を取得する方法は以下。

```
$ echo -n "password" | openssl sha1
(stdin)= 5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8
```

３．５．クライアント証明書発行システムの設定

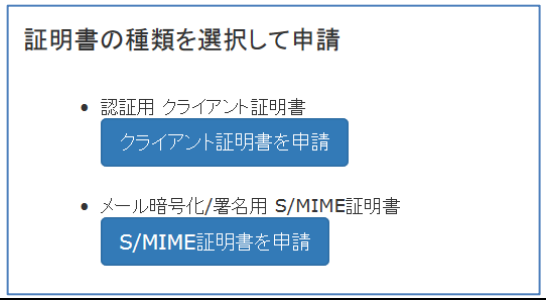
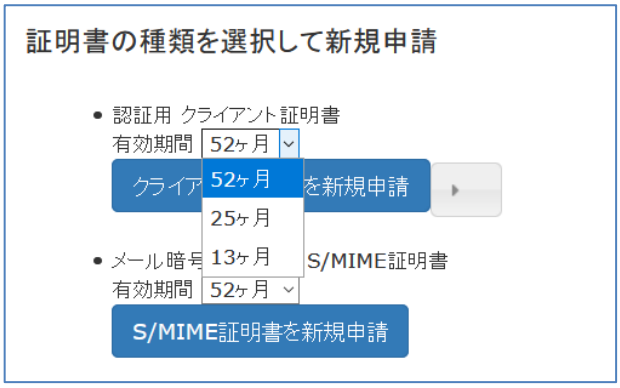
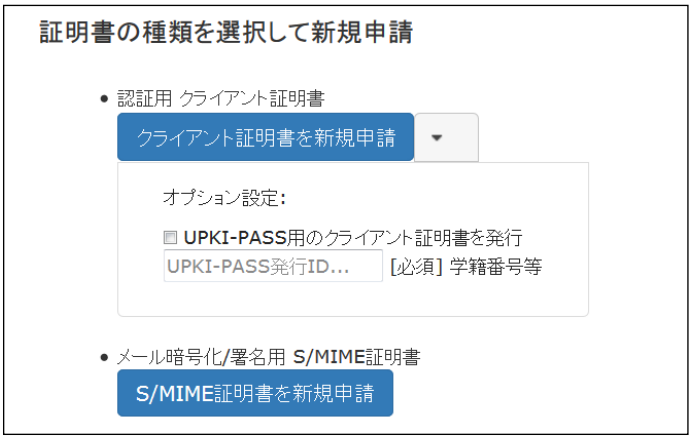
設定ファイル shibcert.yml には機関毎の設定項目がある。以下の項目を自機関用に修正する必要がある。

設定ファイル：shibcert.yml

項目		内容	設定例
flag:		共通設定情報	
	use_pin_generate:	PIN ローカル生成の有無	true : UPKI パスの PIN 情報ローカル生成がオン
	use_upki_pass:	UPKI パス証明書発行有無	true : UPKI パス設定のオプション画面が利用可能
	use_vlan:	VLAN 利用有無	true : VLAN 設定のオプションが利用可能、use_pki_pass が false の時のみ有効、
	use_client_valid:	クライアント証明書の期間指定有無	true：52/25/13 ヶ月の指定が可能、false なら 52 ヶ月標準は true
	use_smime_valid:	S/MIME 証明書の期間指定有無	true：52/25/13 ヶ月の指定が可能、false なら 52 ヶ月標準は true
	admin_page_num:	管理者ページ画面のユーザページングサイズ	標準では 20 が設定され指定ユーザ数毎にページング
label: ja:		日本語表示 設定	
label: en:		英語表示 設定	
	site_title:	サイト名	
	purpose:	証明書表示名	
production:		リリース運用設定	
	(development 共通)	--	--
development:		開発用設定	
	cert_download_type:	デフォルトダウンロード種別	1 固定で良い
	admin_name:	申請 TSV 記載の管理者名	'g-cert-dev.gakunin.nii.ac.jp'
	admin_ou:	申請 TSV 記載の管理者所属名	'UPKI Client Cert Enrollment System Develop'
	admin_mail:	申請 TSV 記載の管理者メール（発行システム受付アドレス）	'gcertsys@langedge.jp'
	admin_pass_mail:	申請 TSV 記載の UPKI パス用管理者メール	'contact@langedge.jp' 受取可能任意メールで良い
	user_ou:	申請 TSV 記載のユーザ所属名 管理者所属と同じでも良い	'UPKI Client Cert Enrollment System Develop'
	base_dn_dev:	開発時に重複を避ける為の OU を	OU=dev05

		指定	
	base_dn_auth:	クライアント証明書用の DN ベース（CN 以外）	OU=UPKI Client Cert Enrollment System Develop, O=National Institute Of Informatics, L=Tokyo,C=JP
	base_dn_smime:	S/MIME 証明書用の DN ベース（CN 以外）	
	access_reject_to:	IP アクセス制限による拒否時のリダイレクト先	標準では "/403.html"
	access_white_ip:	IP アクセス制限ホワイトリスト	カンマ区切りで複数 IP の指定可能、範囲指定可能
	access_black_ip:	IP アクセス制限ブラックリスト	

※ flag 指定によるダッシュボード画面の表示例を以下に示す。

オプション指定	ダッシュボード画面表示
機能オプション無し： use_upki_pass: false use_vlan: false use_client_valid: false use_smime_valid: false	
期間オプション指定： use_upki_pass: false use_vlan: false use_client_valid: true use_smime_valid: true	
UPKI パス利用： use_upki_pass: true use_vlan: false use_client_valid: false use_smime_valid: false	

VLAN 利用： use_upki_pass: false use_vlan: true use_client_valid: false use_smime_valid: false	

３．６．クライアント証明書発行システムの管理者インタフェイス設定

クライアント証明書発行システムの管理者インタフェイスを利用するには 2 つの設定が必要となる。1 つは uid（ePPN）の指定（最大 2 アカウント）であり、もう 1 つは管理者インタフェイス利用時に入力が必要となる Basic 認証の設定である。

設定ファイル：shibcert.yml

項目	内容
production:	リリース運用設定
development:	開発用設定
admin_uid1:	1 人目の管理者ユーザ ID（ePPN）指定 ユーザ ID（uid）に何を使うかは「２．９．連携する属性を変更する場合」を参照、標準では ePPN になっている
admin_uid2:	2 人目の管理者ユーザ ID（ePPN）指定（未設定時は空文字列）
admin_basic_name:	管理者インタフェイス利用時 Basic 認証のユーザ名
admin_basic_pswd:	管理者インタフェイス利用時 Basic 認証のパスワードの SHA-1 ハッシュ値の HEX 化文字列

３．７．連携する属性を変更する場合

設定ファイル omniauth.rb にて連携する Shibboleth の属性との関連付けが行える。例えば学籍番号を利用するのであれば number 属性用に gakuninScopedPersonalUniqueCode 属性を取得できるようにしておく必要がある。

設定ファイル：omniauth.rb

項目	Shibboleth 属性	補足
provider :shibboleth	学認（Shibboleth）用設定	
uid_field	'eppn'	uid として利用
name_field	'displayName'	name として利用
info_fields	---	オプション情報群
email	'mail'	mil として利用
number	'gakuninScopedPersonalUniqueCode'	number として利用 無い場合 eppn を利用
location	'contactAddress'	※未使用
image	'photo_url'	※未使用
phone	'contactPhone'	※未使用

３．８．UPKI パス証明書サーバの連携設定

設定ファイル：shibcert.yml

項目		内容
production:		リリース運用設定
development:		開発用設定
	upki_pass_key1:	UPKI パス証明書サーバ連携時に必要となる設定用キーワード 証明書サーバ運用者に確認して取得してセットする
	upki_pass_key2:	UPKI パス証明書サーバ連携時に必要となる失効用キーワード 証明書サーバ運用者に確認して取得してセットする
	upki_pass_url:	UPKI パス証明書サーバ連携時のサーバ URL をセットする

なお UPKI パス証明書（P12 一括）発行申請時には PIN 発行通知メールは事前に NII にて設定された登録担当者のメールアドレス宛に届く。このままではメールがクライアント証明書発行システム届かないので「[UPKI] アクセス PIN 発行通知」メールをクライアント証明書発行システムの受付メールアドレス（例：'gcertsys@langedge.jp'）に転送する必要がある。以下に qmail を使っている場合の転送設定例を示す。

○ 転送対象の「[UPKI] アクセス PIN 発行通知」メールチェック用 .upki.sh

```
#!/bin/sh
if [ $SENDER = "ca-support@ml.secom-sts.co.jp" ]; then                  ←判定①
 if [ $RECIPIENT = "miyachi@langedge.jp" ]; then                        ←判定②
  if grep "Subject: =?iso-2022-jp?B?W1VQSOldIBskQiUiJS8lOyU5GyhC¥       継続行
          UElOGyRCSC85VERMQO4bKEI=?="; then                             ←判定③
   exit 0
  fi
 fi
fi
exit 99
```

判定①	送信元が支援システム（ca-support@ml.secom-sts.co.jp）宛からかチェック
判定②	送信先が登録担当者（上例では miyachi@langedge.jp）宛かチェック
判定③	サブジェクトが「[UPKI] アクセス PIN 発行通知」かをチェック ※ 7bit になるように ISO-2022 エンコードされている

○ 登録担当者のメール設定ファイル .qmail

```
./Maildir/                          ← 自身のメールボックスへ保存（標準設定）
| /home/miyachi/.upki.sh            ← .upki.sh により PIN 発行通知メールのチェック
gcertsys@langedge.jp                ← .upki.sh が exit 0 の場合のみ転送する受付メール
```

Appendix-A． Dockerfile の解説

A． 1．【Production 環境用の Dockerfile】

```
# ベースイメージ
FROM centos:centos7

########################################
# Production Dockerfile
########################################

# 作成したユーザ情報
LABEL maintainer="Admin <admin@admin.com>"

# ruby と rails のバージョンを指定
ENV ruby_ver="2.5.7"
ENV rails_ver="4.2.10"

# 必要なパッケージをインストール
RUN yum -y update && ¥
#    yum -y install epel-release && ¥
    yum -y install git make autoconf curl wget && ¥
    yum -y install gcc-c++ glibc-headers openssl-devel readline libyaml-devel readline-deve
l zlib zlib-devel sqlite-devel bzip2 && ¥
    yum -y install  sqlite-devel && ¥
    yum -y install openssl && ¥
    yum clean all

RUN yum -y remove mariadb* && ¥
    rm -rf /var/lib/mysql

RUN yum -y install localinstall http://dev.mysql.com/get/mysql57-community-release-el7-7.no
arch.rpm && ¥
    yum -y install mysql-community-client mysql-community-devel && ¥
    yum clean all

RUN cp -p /usr/lib64/mysql/libmysqlclient.so /usr/lib64/

# ruby と bundle をダウンロード
RUN git clone https://github.com/sstephenson/rbenv.git /usr/local/rbenv && ¥
    git clone https://github.com/sstephenson/ruby-build.git /usr/local/rbenv/plugins/ruby-b
uild

# コマンドで rbenv が使えるように設定
RUN echo 'export RBENV_ROOT="/usr/local/rbenv"' >> /etc/profile.d/rbenv.sh && ¥
    echo 'export PATH="${RBENV_ROOT}/bin:${PATH}"' >> /etc/profile.d/rbenv.sh && ¥
    echo 'eval "$(rbenv init --no-rehash -)"' >> /etc/profile.d/rbenv.sh

# ruby と rails をインストール
RUN source /etc/profile.d/rbenv.sh; rbenv install ${ruby_ver}; rbenv global ${ruby_ver}
RUN source /etc/profile.d/rbenv.sh; gem install --version ${rails_ver} rails;

# クライアント証明書発行システムのソース展開
RUN mkdir /home/rails
```

```
ADD gakunin-cert-docker.20200320.tar.gz /home/rails

WORKDIR /home/rails/gakunin-cert

COPY database.yml /home/rails/gakunin-cert/config
COPY omniauth.rb /home/rails/gakunin-cert/config/initializers

RUN /bin/bash -l -c 'bundle install --path vendor/bundle' && ¥
    /bin/bash -l -c 'bundle exec rake db:migrate:reset' && ¥
#    /bin/bash -l -c 'bundle exec rake db:migrate RAILS_ENV=development'
    /bin/bash -l -c 'bundle exec rake db:migrate RAILS_ENV=production' && ¥
    /bin/bash -l -c 'bundle exec rake assets:precompile RAILS_ENV=production'

# Passenger のインストールと Apache の設定
RUN source /etc/profile.d/rbenv.sh; gem install passenger --no-ri --no-rdoc && ¥
    yum install -y curl-devel httpd-devel apr-devel apr-util-devel && ¥
    /bin/bash -l -c 'passenger-install-apache2-module --auto --languages ruby'

WORKDIR /etc/httpd/conf.d
COPY passenger.conf .

# Apache 設定
WORKDIR /home/rails/gakunin-cert
COPY gakunin-cert-top.pro.conf .
#COPY gakunin-cert-top.dev.conf .
COPY gakunin-cert-top.add.txt .
RUN echo setEnv SECRET_KEY_BASE `/bin/bash -l -c 'bundle exec rake secret'` >> gakunin-cert
-top.conf && ¥
    cat gakunin-cert-top.add.txt >> gakunin-cert-top.pro.conf && ¥
#    cat gakunin-cert-top.add.txt >> gakunin-cert-top.dev.conf && ¥
    mv gakunin-cert-top.pro.conf gakunin-cert-top.conf && ¥
#    mv gakunin-cert-top.dev.conf gakunin-cert-top.conf && ¥
    cp gakunin-cert-top.conf /etc/httpd/conf.d

# UPKI 電子証明書自動発行支援システムの設定
COPY mycerts/certificates/ /home/rails/gakunin-cert/config/certificates

# クライアント証明書発行システムの設定

WORKDIR /home/rails/gakunin-cert/config
COPY shibcert.yml .

# Apache SSL 用設定ファイルデータ ############################
COPY mycerts/certs/ /etc/pki/tls/certs
COPY mycerts/private/ /etc/pki/tls/private

COPY ssl.conf /etc/httpd/conf.d/

# DB 関連ファイルの権限設定
WORKDIR /home/rails/gakunin-cert/
RUN chown -R nobody:nobody db log tmp config

######### Shibboleth インストール ##############################
#Workaround since OpenSUSE's provo-mirror is not working properly
#COPY security:shibboleth.repo /etc/yum.repos.d/security:shibboleth.repo
```

④

```
#RUN yum -y update ¥
#    && yum -y install wget ¥
RUN wget http://download.opensuse.org/repositories/security://shibboleth/CentOS_7/security:
shibboleth.repo -P /etc/yum.repos.d ¥
#    && yum -y install httpd shibboleth-3.0.4-3.2 mod_ssl ¥
    && yum -y install shibboleth-3.0.4-3.2 mod_ssl ¥
    && yum -y clean all

COPY httpd-shibd-foreground /usr/local/bin/
COPY shibboleth/ /etc/shibboleth/
COPY shib.conf /etc/httpd/conf.d/

RUN chown shibd.shibd /etc/shibboleth/cert/server.key && ¥
    chmod 440 /etc/shibboleth/cert/server.key

RUN test -d /var/run/lock || mkdir -p /var/run/lock ¥
    && test -d /var/lock/subsys/ || mkdir -p /var/lock/subsys/ ¥
    && chmod +x /etc/shibboleth/shibd-redhat ¥
    && echo $'export LD_LIBRARY_PATH=/opt/shibboleth/lib64:$LD_LIBRARY_PATH¥n'¥
       > /etc/sysconfig/shibd ¥
    && chmod +x /etc/sysconfig/shibd /etc/shibboleth/shibd-redhat /usr/local/bin/httpd-shib
d-foreground ¥
    && sed -i 's/ErrorLog "logs¥/error_log"/ErrorLog ¥/dev¥/stdout/g' /etc/httpd/conf/http
d.conf ¥
    && echo -e "¥nErrorLogFormat ¥"httpd-error [%{u}t] [%-m:%l] [pid %P:tid %T] %7F: %E: [c
lient¥ %a] %M% ,¥ referer¥ %{Referer}i¥"" >> /etc/httpd/conf/httpd.conf ¥
    && sed -i 's/CustomLog "logs¥/access_log" combined/CustomLog ¥/dev¥/stdout ¥"httpd-comb
ined %h %l %u %t ¥¥¥"%r¥¥¥" %>s %b ¥¥¥"%{Referer}i¥¥¥" ¥¥¥"%{User-Agent}i¥¥¥"¥"/g' /etc/htt
pd/conf/httpd.conf ¥
    && sed -i 's/ErrorLog logs¥/ssl_error_log/ErrorLog ¥/dev¥/stdout/g' /etc/httpd/conf.d/s
sl.conf ¥
    && sed -i 's/<¥/VirtualHost>/ErrorLogFormat ¥"httpd-ssl-error [%{u}t] [%-m:%l] [pid %P:
tid %T] %7F: %E: [client¥¥ %a] %M% ,¥¥ referer¥¥ %{Referer}i¥"¥n<¥/VirtualHost>/g' /etc/htt
pd/conf.d/ssl.conf ¥
    && sed -i 's/CustomLog logs¥/ssl_request_log/CustomLog ¥/dev¥/stdout/g' /etc/httpd/con
f.d/ssl.conf ¥
    && sed -i 's/TransferLog logs¥/ssl_access_log/TransferLog ¥/dev¥/stdout/g' /etc/httpd/c
onf.d/ssl.conf

EXPOSE 80 443

CMD ["httpd-shibd-foreground"]
```

⑤

① ベースコンテナイメージの指定

Docker Hub で公開されている centos:centos7 がベースイメージとなります。

参考 URL：https://hub.docker.com/_/centos

② Ruby と Rails のバージョンの指定

Ruby：2.5.7 / Rails：4.2.10

③ クライアント証明書発行システムに必要なモジュール群のインストール
以下を実施しています。
・Ruby および Rails のインストール
・MySQL クライアントのインストール
・DB の初期化（migrate）
・Passeger のインストール
・各種設定ファイルの配置・生成
  ・shibcert.yml、passenger.conf、omniauth.rb、database.yml
  ・gakunin-cert-top.conf
  ・サーバ証明書 server.key、server.crt、server-chain.crt
  ・支援システム用の秘密鍵と証明書 client.cer、client.key

④ Shibboleth-SP のインストールに関連する記載
参考 URL：https://hub.docker.com/r/unicon/shibboleth-sp/tags

以下を実施しています。
・shibboleth-sp のインストール
・各種設定ファイルの配置
  ・shibd.conf
  ・attribute-map.xml、attribute-policy.xml、
  ・gakunin-test-signer-2011.cer、server.crt、server.key
  ・native.logger、shibd.logger
  ・shibboleth2.xml

⑤ 起動時に実行されるコマンド
/usr/local/bin/以下にある起動コマンド httpd-shibd-foreground を実行します。

Ａ．２．【Development 環境用の Dockerfile】

```
# ベースイメージ
FROM centos:centos7

########################################
# Development Dockerfile
########################################

# 作成したユーザ情報
LABEL maintainer="Admin <admin@admin.com>"

# ruby と rails のバージョンを指定
ENV ruby_ver="2.5.7"
ENV rails_ver="4.2.10"

# 必要なパッケージをインストール
RUN yum -y update && ¥
#    yum -y install epel-release && ¥
    yum -y install git make autoconf curl wget && ¥
    yum -y install gcc-c++ glibc-headers openssl-devel readline libyaml-devel readline-deve
l zlib zlib-devel sqlite-devel bzip2 && ¥
    yum -y install  sqlite-devel && ¥
    yum -y install openssl && ¥
    yum clean all

RUN yum -y remove mariadb* && ¥
    rm -rf /var/lib/mysql

RUN yum -y install localinstall http://dev.mysql.com/get/mysql57-community-release-el7-7.no
arch.rpm && ¥
    yum -y install mysql-community-client mysql-community-devel && ¥
    yum clean all

RUN cp -p /usr/lib64/mysql/libmysqlclient.so /usr/lib64/

# ruby と bundle をダウンロード
RUN git clone https://github.com/sstephenson/rbenv.git /usr/local/rbenv && ¥
    git clone https://github.com/sstephenson/ruby-build.git /usr/local/rbenv/plugins/ruby-b
uild

# コマンドで rbenv が使えるように設定
RUN echo 'export RBENV_ROOT="/usr/local/rbenv"' >> /etc/profile.d/rbenv.sh && ¥
    echo 'export PATH="${RBENV_ROOT}/bin:${PATH}"' >> /etc/profile.d/rbenv.sh && ¥
    echo 'eval "$(rbenv init --no-rehash -)"' >> /etc/profile.d/rbenv.sh

# ruby と rails をインストール
RUN source /etc/profile.d/rbenv.sh; rbenv install ${ruby_ver}; rbenv global ${ruby_ver}
RUN source /etc/profile.d/rbenv.sh; gem install --version ${rails_ver} rails;

# クライアント証明書発行システムのソース展開
RUN mkdir /home/rails

ADD gakunin-cert-docker.20200320.tar.gz /home/rails
```

```
WORKDIR /home/rails/gakunin-cert

COPY database.yml /home/rails/gakunin-cert/config
COPY omniauth.rb /home/rails/gakunin-cert/config/initializers

RUN /bin/bash -l -c 'bundle install --path vendor/bundle' && ¥
    /bin/bash -l -c 'bundle exec rake db:migrate:reset' && ¥
    /bin/bash -l -c 'bundle exec rake db:migrate RAILS_ENV=development'
#    /bin/bash -l -c 'bundle exec rake db:migrate RAILS_ENV=production' && ¥
#    /bin/bash -l -c 'bundle exec rake assets:precompile RAILS_ENV=production'

# Passenger のインストールと Apache の設定
RUN source /etc/profile.d/rbenv.sh; gem install passenger --no-ri --no-rdoc && ¥
    yum install -y curl-devel httpd-devel apr-devel apr-util-devel && ¥
    /bin/bash -l -c 'passenger-install-apache2-module --auto --languages ruby'

WORKDIR /etc/httpd/conf.d
COPY passenger.conf .

# Apache 設定
WORKDIR /home/rails/gakunin-cert
#COPY gakunin-cert-top.pro.conf .
COPY gakunin-cert-top.dev.conf .
COPY gakunin-cert-top.add.txt .
RUN echo setEnv SECRET_KEY_BASE `/bin/bash -l -c 'bundle exec rake secret'` >> gakunin-cert
-top.conf && ¥
#    cat gakunin-cert-top.add.txt >> gakunin-cert-top.pro.conf && ¥
    cat gakunin-cert-top.add.txt >> gakunin-cert-top.dev.conf && ¥
#    mv gakunin-cert-top.pro.conf gakunin-cert-top.conf && ¥
    mv gakunin-cert-top.dev.conf gakunin-cert-top.conf && ¥
    cp gakunin-cert-top.conf /etc/httpd/conf.d

# UPKI 電子証明書自動発行支援システムの設定
COPY mycerts/certificates/ /home/rails/gakunin-cert/config/certificates

# クライアント証明書発行システムの設定

WORKDIR /home/rails/gakunin-cert/config
COPY shibcert.yml .

# Apache SSL 用設定ファイルデータ ##########################
COPY mycerts/certs/ /etc/pki/tls/certs
COPY mycerts/private/ /etc/pki/tls/private

COPY ssl.conf /etc/httpd/conf.d/

# DB 関連ファイルの権限設定
WORKDIR /home/rails/gakunin-cert/
RUN chown -R nobody:nobody db log tmp config

######### Shibboleth インストール ############################
#Workaround since OpenSUSE's provo-mirror is not working properly
#COPY security:shibboleth.repo /etc/yum.repos.d/security:shibboleth.repo

#RUN yum -y update ¥
#    && yum -y install wget ¥
```

①

②

```
RUN wget http://download.opensuse.org/repositories/security://shibboleth/CentOS_7/security:
shibboleth.repo -P /etc/yum.repos.d ¥
#    && yum -y install httpd shibboleth-3.0.4-3.2 mod_ssl ¥
    && yum -y install shibboleth-3.0.4-3.2 mod_ssl ¥
    && yum -y clean all

COPY httpd-shibd-foreground /usr/local/bin/
COPY shibboleth/ /etc/shibboleth/
COPY shib.conf /etc/httpd/conf.d/

RUN chown shibd.shibd /etc/shibboleth/cert/server.key && ¥
    chmod 440 /etc/shibboleth/cert/server.key

RUN test -d /var/run/lock || mkdir -p /var/run/lock ¥
    && test -d /var/lock/subsys/ || mkdir -p /var/lock/subsys/ ¥
    && chmod +x /etc/shibboleth/shibd-redhat ¥
    && echo $'export LD_LIBRARY_PATH=/opt/shibboleth/lib64:$LD_LIBRARY_PATH¥n'¥
       > /etc/sysconfig/shibd ¥
    && chmod +x /etc/sysconfig/shibd /etc/shibboleth/shibd-redhat /usr/local/bin/httpd-shib
d-foreground ¥
    && sed -i 's/ErrorLog "logs¥/error_log"/ErrorLog ¥/dev¥/stdout/g' /etc/httpd/conf/http
d.conf ¥
    && echo -e "¥nErrorLogFormat ¥"httpd-error [%{u}t] [%-m:%l] [pid %P:tid %T] %7F: %E: [c
lient¥ %a] %M% ,¥ referer¥ %{Referer}i¥"" >> /etc/httpd/conf/httpd.conf ¥
    && sed -i 's/CustomLog "logs¥/access_log" combined/CustomLog ¥/dev¥/stdout ¥"httpd-comb
ined %h %l %u %t ¥¥¥"%r¥¥¥" %>s %b ¥¥¥"%{Referer}i¥¥¥" ¥¥¥"%{User-Agent}i¥¥¥"¥"/g' /etc/htt
pd/conf/httpd.conf ¥
    && sed -i 's/ErrorLog logs¥/ssl_error_log/ErrorLog ¥/dev¥/stdout/g' /etc/httpd/conf.d/s
sl.conf ¥
    && sed -i 's/<¥/VirtualHost>/ErrorLogFormat ¥"httpd-ssl-error [%{u}t] [%-m:%l] [pid %P:
tid %T] %7F: %E: [client¥¥ %a] %M% ,¥¥ referer¥¥ %{Referer}i¥"¥n<¥/VirtualHost>/g' /etc/htt
pd/conf.d/ssl.conf ¥
    && sed -i 's/CustomLog logs¥/ssl_request_log/CustomLog ¥/dev¥/stdout/g' /etc/httpd/con
f.d/ssl.conf ¥
    && sed -i 's/TransferLog logs¥/ssl_access_log/TransferLog ¥/dev¥/stdout/g' /etc/httpd/c
onf.d/ssl.conf

EXPOSE 80 443

CMD ["httpd-shibd-foreground"]
```

① Rails を Development 環境用に切り替える設定

② Development 環境用に切り替えるための設定ファイル gakunin-cert-top.conf の配置

以上