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
インストールマニュアル

２０２０年３月２４日版

□ 目次

□ 履歴

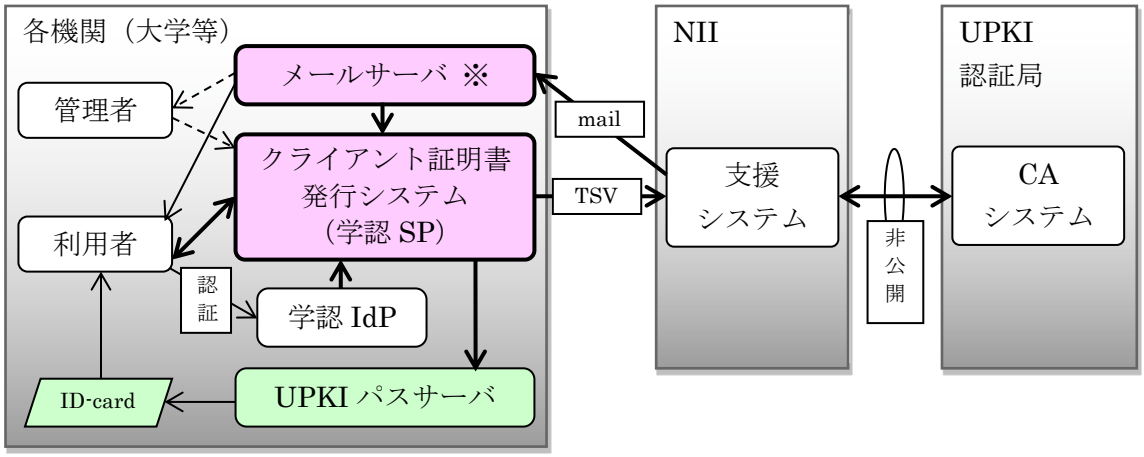
日付	改定内容	作成者
2020/03/24	2019 年度の機能更新を反映 ➢ ２．６．クライアント証明書発行システムの設定 その他ファイル名等の更新	宮地
2019/03/22	学認連携クライアント証明書発行システム改修 に伴う更新 以下新機能の追加： ➢ ２．７．クライアント証明書発行システムの管理者インタフェイス設定 ➢ ２．９．UPKI パス証明書サーバの連携設定 その他更新	宮地
2018/03/29	初版	宮地

１．概要

本インストールマニュアルは、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）を新規サーバにインストール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する。

また 2019 年 3 月の改修により UPKI パスの発行の為に UPKI パス証明書サーバへの接続機能がサポートされた。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する。これにより 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。



全システム概要と関連図（※別メールサーバはオプション）

１．１．動作確認済み環境

URL	: https://g-cert-dev.gakunin.nii.ac.jp/ （プロトタイプサーバ）
OS	: CentOS 7.1
Apache	: 2.4.6（CentOS にインストール済み、TLS 接続用のサーバ証明書付）
Shibboleth	: 2.6.1（テストフェデレーション SP として登録済み）
Ruby	: 2.3.6（rbenv 2.4.3 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	: 5.2.1（ruby gem により個別インストール）
MySQL	: 5.5.56（yum でインストール・Rails の production 環境用）
Sqlite3	: 3.7.17（CentOS にインストール済み・Rails の development 環境用）

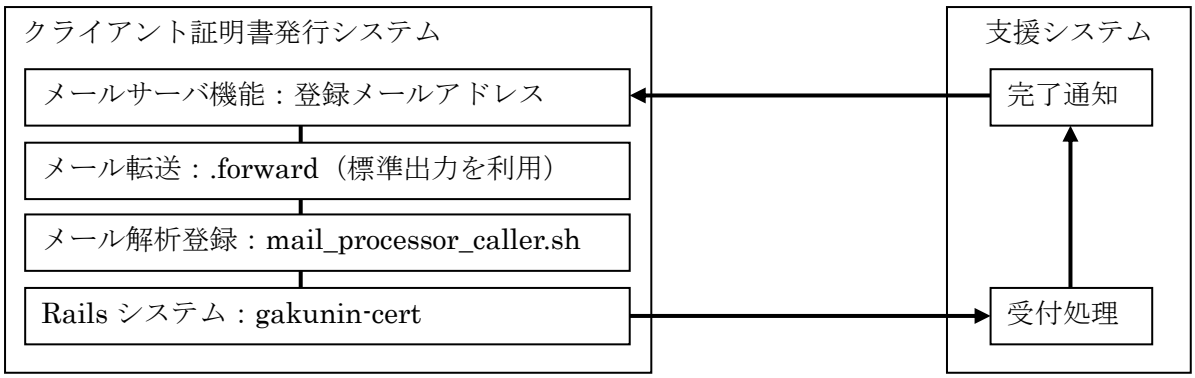
１．２．動作条件

１）学術認証フェデレーション（学認）
インストールサーバが学術認証フェデレーションの SP 設定済みであること。
※　shibd が動作済みであり、IdP にて認証が可能なこと。

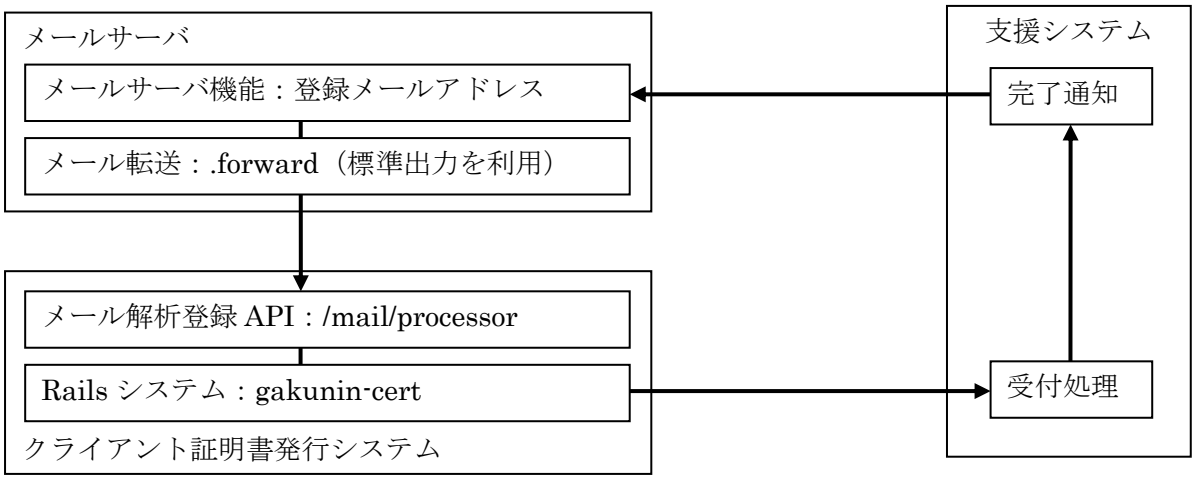
２）UPKI 電子証明書自動発行支援システム
支援システムに接続する為の管理者用クライアント証明書（登録担当者用証明書）が発行済みであること。

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。

※1：ローカル接続（クライアント証明書発行システム内にメールサーバ機能を持つ）



※2：リモート接続（別メールサーバで受け取ったメールを REST API で転送する）



4）UPKI パスドライバ（サーバ）：オプション

クライアント証明書発行システムから HTTP(S)接続が可能なネットワーク上に UPKI パスドライバが稼働していること。

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．Ruby/Rails 関連のインストール

１）rbenv/ruby を system-wide にインストールして Rails をインストール

```
// git のインストール
$ sudo yum install -y git
$ sudo yum install -y openssl-devel readline-devel zlib-develb zip2
$ sudo yum install -y gcc gcc-c++

// git を使って rbenv と ruby-build の取得
$ sudo git clone git://github.com/sstephenson/rbenv.git /usr/local/rbenv
$ sudo git clone git://github.com/sstephenson/ruby-build.git /usr/local/rbenv/plugins/ruby-build

$ sudo visudo
---(Defaults をコメント化して rbenv パスと環境変数を追加)---
#Defaults secure_path = /sbin:/bin:/usr/sbin:/usr/bin
Defaults secure_path = /sbin:/bin:/usr/sbin:/usr/bin:/usr/local/rbenv/bin:/usr/local/rbenv/shims
Defaults env_keep += "RBENV_ROOT"
---(ここまで)---

$ sudo rbenv init -
$ sudo vi /etc/profile
---(以下の環境変数設定を追加)---
export RBENV_ROOT=/usr/local/rbenv
export PATH="$RBENV_ROOT/bin:$PATH"
eval "$(rbenv init -)"
---(ここまで)---
$ cd /usr/local/rbenv/
$ sudo mkdir versions
$ sudo mkdir shims
$ source /etc/prfile

// ruby のインストール
$ sudo rbenv install 2.3.6
$ sudo rbenv global 2.3.6
$ ruby -v
ruby 2.3.6p384 (2017-12-14 revision 61254) [x86_64-linux]
$

// Rails のインストール（Ruby 2.3.6 の組合せが必要）
$ sudo gem install rails --version 4.2.10
$ rails -v
Rails 4.2.10
$
```

２）クライアント証明書発行システムのソース展開

※ gakunin-cert.20200324.tar.gz が必要です（日時は最新版を利用）。

```
// インストール先に移動（任意で良いがここでは /home/rails の下としている）
$ cd /home/rails
$ tar xvfz gakunin-cert.20200324.tar.gz
$ cd gakunin-cert

// bundle install にて必要な gem をインストールする（エラーが無いことを確認する）
$ bundle install --path vendor/bundle

// データーベースの初期化（以下は開発環境用）
$ bundle exec rake db:migrate:reset
$ bundle exec rake db:migrate RAILS_ENV=development
```

３）Passenger のインストールと Apache の設定

```
// Passenger の gem インストール
$ sudo gem install passenger --no-ri --no-rdoc

// Passenger のセットに必要なプロジェクトのインストール
$ sudo yum install -y curl-devel httpd-devel apr-devel apr-util-devel

// Passenger の Apache 用モジュールのインストール ※A
$ sudo passenger-install-apache2-module

$ cd /etc/httpd/conf.d
$ sudo vi passenger.conf
---(以下追加：注 環境毎に異なる ※A の出力を利用すること)---
LoadModule passenger_module ¥
  /usr/local/rbenv/versions/2.3.6/lib/ruby/gems/2.3.0/gems/passenger-5.2.1/buildout/ ¥
  apache2/mod_passenger.so
<IfModule mod_passenger.c>
  PassengerRoot /usr/local/rbenv/versions/2.3.6/lib/ruby/gems/2.3.0/gems/passenger-5.2.1
  PassengerDefaultRuby /usr/local/rbenv/versions/2.3.6/bin/ruby
</IfModule>
---(ここまで)---
$
```

４）gakunin-cert プロジェクトの Apache 設定

```
$ cd /etc/httpd/conf.d

// Rails 用シークレットの生成 ※B
$ bundle exec rake secret

// gakunin-cert 用の設定作成（以下は development 用）
$ sudo vi gakunin-cert-top.conf
---(以下追加)---
DocumentRoot /home/rails/gakunin-cert/public
<Location />
  RailsEnv development
#  RailsEnv production
  PassengerBaseURI /
  PassengerAppRoot /home/rails/gakunin-cert
</Location>
<Directory /home/rails/gakunin-cert/public>
  SetEnv SECRET_KEY_BASE (※B の rake secret の結果 HEX 文字列をここにコピー)
  AllowOverride all
  Require all granted
  Options -MultiViews
</Directory>
---(ここまで)---

// 動作確認
$ sudo service httpd restart
```

２．２．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

```
$ cd /etc/httpd/conf.d

// 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)---

// 動作確認
$ sudo service shibd restart
```

※ 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

２．３．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：config/initializers/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

２．４．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）クライアント証明書発行システムへのセット

```
// 秘密鍵ファイルのコピー
$ cp client.key /home/rails/gakunin-cert/config/certificates

// 証明書ファイルのコピー
$ cp client.cer /home/rails/gakunin-cert/config/certificates
```

設定ファイル：config/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

２．５．支援システムからのメール受信の設定

A) ローカル設定の場合

クライアント証明書発行システム自体で指定されたメールアドレスが取得できる場合は、以下のローカル設定を行う。

```
// メール取得ユーザのホームディレクトリに移動
$ cd ~

// .forward ファイルのセット
$ cp /home/rails/gakunin-cert/lib/scripts/_forward.local-sample .forward
```

参考：.forward 例

```
| /home/rails/gakunin-cert/lib/scripts/mail_processor_caller.sh
```

B) 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：config/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
```

２．６．クライアント証明書発行システムの設定

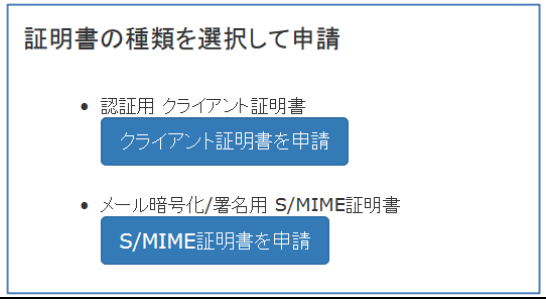
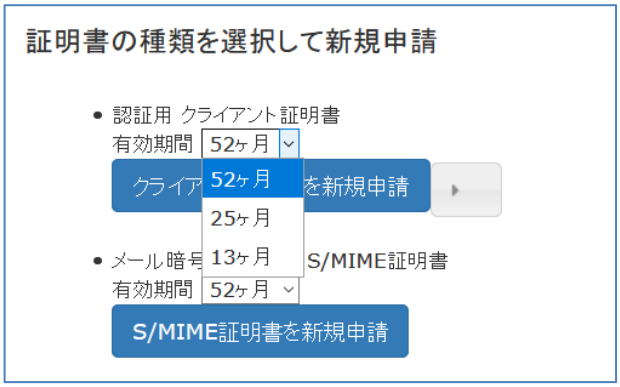
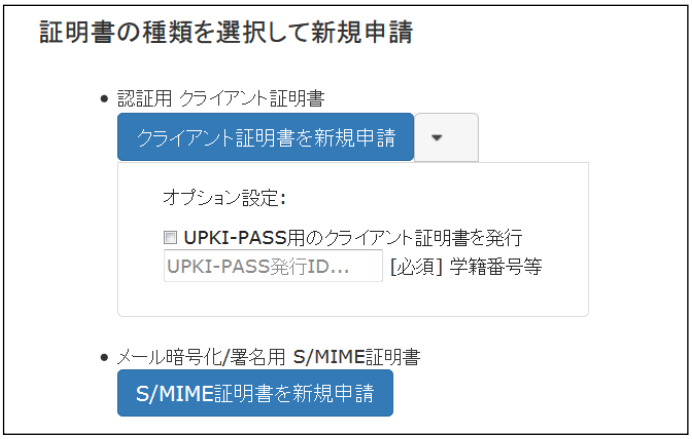
設定ファイル shibcert.yml には各機関毎の設定項目がある。以下の項目を自機関用に修正する必要がある。

設定ファイル：config/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になる
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:	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 を指定	OU=dev05
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,
base_dn_smime:	S/MIME 証明書用の DN ベー	

		ス（CN 以外）	O=National Institute Of Informatics, L=Tokyo,C=JP
	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	

２．７．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：config/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

２．８．連携する属性を変更する場合

設定ファイル config/initializers/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：config/initializers/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

２．９．UPKI パス証明書サーバの連携設定

設定ファイル：config/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

以上