Orchestrating a brighter world

NEC

SecureWare/Credential Lifecycle Manager

利用の手引

2019 年 10 月

日本電気株式会社

はしがき

本書は、SecureWare/Credential Lifecycle Manager(以下、CLM と称します)の利用方法について説明したものです。CLM は、クライアントへの電子証明書配布や自動更新を行うための製品です。CLM の概念および操作方法について順を追って説明します。

また、本書では、CLM のクライアントである SecureWare/Credential Lifecycle Agent (以下、CLA と称します)の操作方法についても説明します。

本書の構成は以下のとおりです。

章	タイトル	内容
1	はじめに	CLM 概要の説明
2	起動と停止	CLM、CLA の起動と停止の手順
3	設定ファイル	CLM の設定ファイルについての説明
4	ログ出力	CLM、CLA が出力するログについての説明
5	ID・パスワード管理	ID・パスワード管理機能についての説明
6	証明書管理	証明書管理機能についての説明
7	共通鍵管理	共通鍵管理機能についての説明
8	デバイス管理	デバイス管理機能についての説明
9	鍵自動更新・統計情報表示	証明書・共通鍵の自動更新機能についての説明
10	AWS IoT 接続設定	AWS IoT との接続自動設定機能についての説明
11	エッジ/デバイス自動検出・登録	エッジ/デバイスの自動検出・登録機能についての説明

◎付録

付録	タイトル	内容
1	ログメッセージ一覧	ログメッセージの一覧
2	バックアップ・リストア	CLM・CLA のバックアップとリストア手順
3	CLM が連携する認証局	CLM が連携する認証局の仕様と設定変更手順
4	エッジゲートウェイのキッティングに関する留意事項	NEC 製エッジゲートウェイのキッティングを行う際の留意事項
5	証跡データのエクスポート･クリーンアップ	証跡データのエクスポート・クリーンアップについての説明

2019 年 10 月　第五版

備考

本書に説明しているすべての機能はプログラムプロダクトであり、次のプロダクト名、およびプロダクトリリースに対応しています。

プロダクト名	プロダクトリリース
SecureWare/Credential Lifecycle Manager	V1.1.1
SecureWare/Credential Lifecycle Agent	V1.1.1

本書は、以下のオペレーティングシステムに対応しています。

【Linux】

RedHat Enterprise Linux 6、7

CentOS 6、7

Debian 8.6、8.8

Automotive Grade Linux 5.0.3

【Windows】

Windows Server 2012 R2、2016

Winows 10 IoT

・ Linux は Linus Torvalds 氏の米国およびその他の国における登録商標あるいは商標です。

・ Red Hat は米国 Red Hat ,Inc.の登録商標です。

・ Microsoft、Windows、は、米国マイクロソフト社の米国およびその他の国における登録商標です。

・ Oracle と Java は、Oracle Corporation 及びその子会社、関連会社の米国及びその他の国における登録商標です。文中の社名、商品名等は各社の商標または登録商標である場合があります。

・ その他、記載されている会社名、製品名は、各社の商標および登録商標です。

・ 各ソフトウェアで独自のライセンスが指定されているソフトウェアに関しては、そのライセンスが優先されます。

・ 本書の記述には提供していない機能も含まれています。ご利用に当たっては、製品に添付されているリリースメモまたは、セットアップカードをご覧ください。

目次

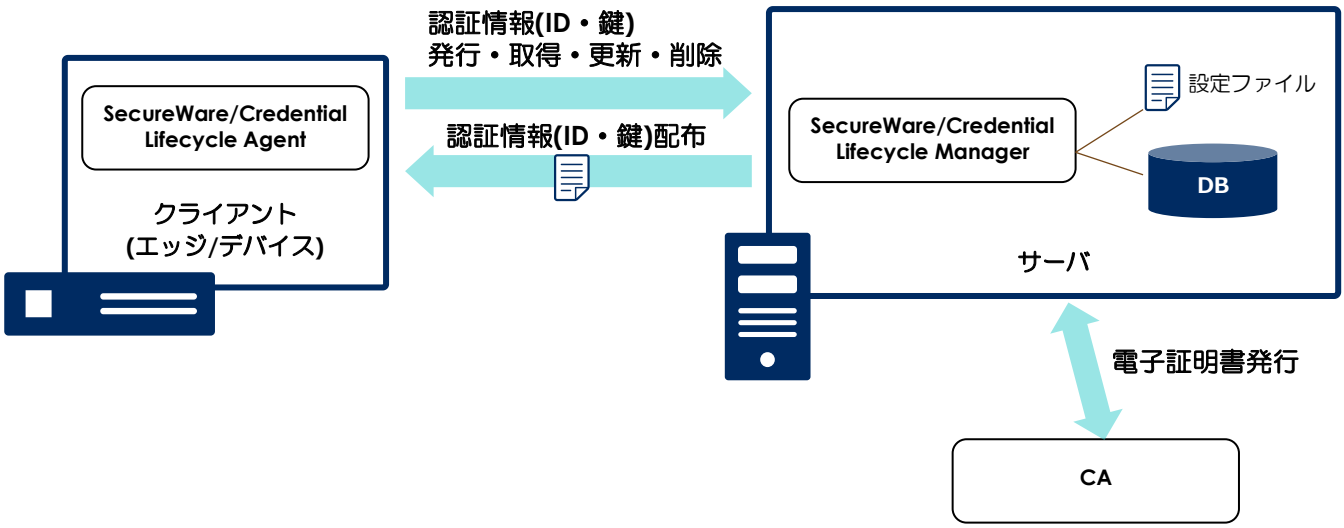
1 はじめに

1.1 SecureWare/Credential Lifecycle Manager とは

SecureWare/Credential Lifecycle Manager は、IoT 環境での相互認証/暗号化通信に必要となる認証情報(ID・鍵)を生成・管理する製品です。

本製品は、認証局(以降、CA と呼称します)と連携し、クライアント(エッジ/デバイス)が使用可能な電子証明書の発行や管理を行うことができるほか、ID・パスワードや共通鍵の生成・管理を行うことができます。本製品で発行した電子証明書、ID・パスワード、共通鍵は、製品内でクライアント(エッジ/デバイス)と紐づけて管理します。

本製品は、認証情報とデバイス情報を管理する Manager（CLM）と、クライアントにインストールし、クライアント上で認証情報の管理を行う Agent から構成されます。Agent 上で動作する製品は SecureWare/Credential Lifecycle Agent です(以下、CLA と称します) 。



1.2 主な機能

CLM が提供する主な機能概要を以下に記載します。

- ID・パスワードの発行・取得・削除

  認証に利用可能な ID とパスワードを発行・取得・削除する機能。

- ID・パスワードの照合(認証)

  ID・パスワードの発行・取得で得られた ID・パスワードを照合することで、ID・パスワードによる認証を行う機能。

- 電子証明書・共通鍵の発行機能

  電子証明書・共有鍵を発行する機能。

- 電子証明書・共通鍵の更新機能

  発行済みの電子証明書・共通鍵を新しい電子証明書・共通鍵に更新する機能。

- 電子証明書・共通鍵の取得機能

  発行済みの電子証明書・共通鍵を再度取得する機能。

- 電子証明書・共通鍵の失効/削除機能

  発行済みの電子証明書を失効する/共通鍵を削除する機能。

- 電子証明書・共通鍵の改ざん検知機能

  CLM が発行した電子証明書・共通鍵が、エッジ/デバイス上で改ざんされた場合に CLM へ改ざんを通知する機能。

- 共通鍵の照合機能

  共通鍵発行・取得・更新で得られた共通鍵を照合することで、共通鍵による認証を行う機能。

- TWINE 連携機能

  軽量暗号 開発キット(TWINE)と連携し、CLM が発行した共通鍵を TWINE 用暗号鍵に変換して保存する機能や、TWINE 用暗号鍵を用いた共通鍵の照合を行う機能。

- 鍵自動更新・統計情報表示

  エッジ/デバイス上にインストールした CLA と連携し、エッジ/デバイス上の電子証明書・共通鍵を定期的に自動更新する機能や、リモートから鍵を管理する機能。

  また、エッジ/デバイス上の鍵更新状況を統計情報として収集・グラフ化して表示する機能。

- AWS IoT 接続設定機能

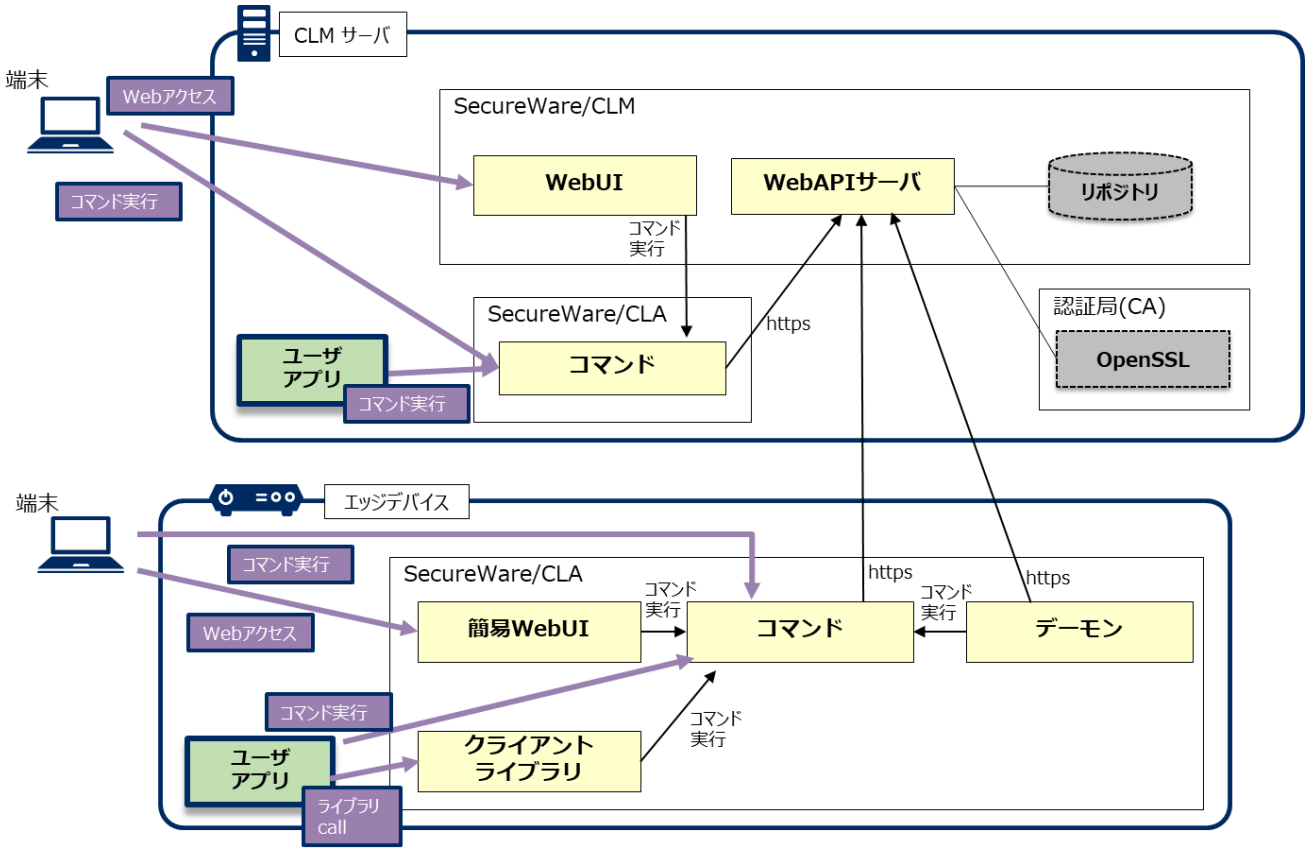
  エッジ/デバイスが CLM で発行した電子証明書(デバイス証明書)を使用して、AWS IoT へ接続するために必要な設定をワンクリックで自動設定する機能。

- エッジ/デバイス自動検出機能

  InfoCage 不正接続防止と連携し、InfoCage 不正接続防止で監視しているネットワークに接続したエッジ/デバイスを自動検出し、検出したエッジ/デバイスとその情報を自動登録する機能。

1.3 主なコンポーネント

CLM・CLA を構成する主なコンポーネントについて、以下に説明します。



- WebAPI サーバ

  認証情報とデバイス情報の管理を行うコンポーネントです。

  認証局と連携した電子証明書の管理や、共通鍵、ID・パスワードの管理など、認証情報の管理を行います。また、認証情報は、認証情報を保有するデバイスと紐づけして管理します。

  CLM の機能をクライアントから使うためのインターフェースとして、WebAPI を提供します。

- リポジトリ

  認証情報やデバイス情報、証跡などを管理するデータベースです。

- 認証局(CA)

  電子証明書を発行する機関です。本バージョンでは、OpenSSL の認証局を利用可能です。

- WebUI

  CLM に付属する管理画面です。

  コマンドが提供する機能のうち、ID・パスワードの管理とデバイス管理(EDMS オプション導入時)を Web ブラウザから簡単に利用できるように UI を提供しています。

  また、デバイスと AWS IoT との連携設定を行うための UI や、エッジ/デバイスの鍵更新状況を

グラフィカルに表示し、更新が行われていないエッジ/デバイスのリスト化や、リモートから鍵管理を行うための UI も提供しています。

- コマンド

  CLM の提供する機能を使うためのコマンドです。

  コマンドを使うことで、CLM の提供する機能を容易に、且つ便利に利用できます。

- デーモン

  CLM の提供機能を使うためのデーモンです。

  証明書・共通鍵の定期自動更新やリモート更新、証明書・共通鍵の改ざんチェック・チェック結果通知機能を提供しています。

- 簡易 WebUI

  CLA に付属する管理画面です。

  コマンドが提供する機能のうち、証明書、共通鍵の管理やデバイスと AWS IoT との接続設定を Web ブラウザから簡単に利用できるように UI を提供しています。

- クライアントライブラリ

  CLM の機能をユーザアプリケーションから利用するためのライブラリです。

  本バージョンでは、Node-RED のフローを提供しています。

1.4 動作環境

SecureWare/Credential Lifecycle Manager (Linux 版)

OS	RedHat Enterprise Linux 6.8 以降 RedHat Enterprise Linux 7.0 以降 CentOS 6.8 以降 CentOS 7.0 以降
必須ソフトウェア	[WebAPI サーバ] OpenSSL 1.1.0j (製品に同梱) PostgreSQL 9.6.11 OpenJDK 1.8.0 update 202 Apache Tomcat 8.5.37 軽量暗号 開発キット (製品に同梱・ライセンスは有償) [WebUI] OpenJDK 1.8.0 update 202 Apache Tomcat 8.5.37 Perl sudo ・RHEL/CentOS 6.8 以降の場合 1.8.6 p 3 以降 ・RHEL/CentOS 7.0 以降の場合 1.8.19 p 2 以降 [コマンド] Boost C++ Libraries(製品に同梱) OpenSSL (製品に同梱)
推奨空ディスク容量	2GByte 以上
推奨空きメモリサイズ	4GByte 以上
Web ブラウザ	Internet Explorer 11 Firefox 55.0

SecureWare/Credential Lifecycle Manager (Windows 版)

OS	Windows Server 2012 R2、2016 Windows 10 IoT Enterprise

必須ソフトウェア	[WebAPI サーバ] OpenSSL 1.1.0j (製品に同梱) PostgreSQL 9.6.11 Oracle Java SE Runtime Environment 8u202 Apache Tomcat 8.5.37 軽量暗号 開発キット (製品に同梱・ライセンスは有償) [WebUI] Oracle Java SE Runtime Environment 8u202 Apache Tomcat 8.5.37 Strawberry Perl [コマンド] Boost C++ Libraries(製品に同梱) OpenSSL (製品に同梱)
推奨空ディスク容量	2GByte 以上
推奨空きメモリサイズ	4GByte 以上
Web ブラウザ	Internet Explorer 11 Firefox 55.0

SecureWare/Credential Lifecycle Agent (エッジゲートウェイ (Linux 版))

必須ソフトウェア	[簡易 WebUI] lighttpd Perl sudo [コマンド・デーモン] Boost C++ Libraries(製品に同梱) OpenSSL (製品に同梱)
ディスク容量	25MByte 以上 簡易 WebUI : 10MB コマンド・デーモン: 15MB
メモリサイズ	13MByte 以上 簡易 WebUI : 3MB コマンド・デーモン: 10MB
Web ブラウザ	Internet Explorer 11 Firefox 55.0 以降

SecureWare/Credential Lifecycle Agent (Debian)

OS	Debian Linux 8.6 (x86/x64) Debian Linux 8.8 (x64)
必須ソフトウェア	[簡易 WebUI] lighttpd Perl sudo [コマンド・デーモン] Boost C++ Libraries(製品に同梱) OpenSSL (製品に同梱)
ディスク容量	25MByte 以上 簡易 WebUI : 10MB コマンド・デーモン: 15MB
メモリサイズ	13MByte 以上 簡易 WebUI : 3MB コマンド・デーモン: 10MB
Web ブラウザ	Internet Explorer 11 Firefox 55.0 以降

SecureWare/Credential Lifecycle Agent (RHEL/CentOS)

OS	RedHat Enterprise Linux 6.8 以降 RedHat Enterprise Linux 7.0 以降 CentOS 6.8 以降 CentOS 7.0 以降
必須ソフトウェア	[簡易 WebUI] Oracle Java SE Runtime Environment 8u202 Apache Tomcat 8.5.37 Perl sudo ・RHEL/CentOS 6.8 以降の場合 1.8.6 p 3 以降 ・RHEL/CentOS 7.0 以降の場合 1.8.19 p 2 以降 [コマンド・デーモン] Boost C++ Libraries(製品に同梱) OpenSSL (製品に同梱)
ディスク容量	515MByte 以上 簡易 WebUI : 500MB コマンド・デーモン: 15MB

メモリサイズ	13MByte 以上 簡易 WebUI　　: 100MB コマンド・デーモン:　10MB
Web ブラウザ	Internet Explorer 11 Firefox 55.0 以降

SecureWare/Credential Lifecycle Agent (AGL)

OS	Automotive Grade Linux 5.0.3
HW	Raspberry Pi 3 model B
必須ソフトウェア	[簡易 WebUI] lighttpd Perl sudo [コマンド・デーモン] Boost C++ Libraries(製品に同梱) OpenSSL (製品に同梱)
ディスク容量	25MByte 以上 簡易 WebUI　　: 10MB コマンド・デーモン: 15MB
メモリサイズ	13MByte 以上 簡易 WebUI　　:　3MB コマンド・デーモン: 10MB
Web ブラウザ	Internet Explorer 11 Firefox 55.0 以降

SecureWare/Credential Lifecycle Agent (エッジゲートウェイ(Windows 版))

OS	Windows10 IoT Enterprise
必須ソフトウェア	[簡易 WebUI] Oracle Java SE Runtime Environment 8u202 Apache Tomcat 8.5.37 Strawberry Perl [コマンド・サービス] Boost C++ Libraries(製品に同梱) OpenSSL (製品に同梱)

ディスク容量	515MByte 以上 簡易 WebUI　　：500MB コマンド・サービス:　15MB
メモリサイズ	13MByte 以上 簡易 WebUI　　：100MB コマンド・サービス:　10MB
Web ブラウザ	Internet Explorer 11 Firefox 55.0 以降

SecureWare/Credential Lifecycle Agent (Windows)

OS	Windows Server 2012R2、2016 Windows 10 IoT (x86/x64)
必須ソフトウェア	[簡易 WebUI] Oracle Java SE Runtime Environment 8u202 Apache Tomcat 8.5.37 Strawberry Perl [コマンド・サービス] Boost C++ Libraries(製品に同梱) OpenSSL (製品に同梱)
ディスク容量	515MByte 以上 簡易 WebUI　　：500MB コマンド・サービス:　15MB
メモリサイズ	13MByte 以上 簡易 WebUI　　：100MB コマンド・サービス:　10MB
Web ブラウザ	Internet Explorer 11 Firefox 55.0 以降

1.5 用語説明

本書で使用している用語について説明します。

表 1-1　用語説明

No.	用語	説明
1	認証局	電子商取引事業者などに、暗号通信などで必要となる電子証明書を発行する機関です。 CLM では、OpenSSL で構築した認証局と連携できます。
2	エッジ ID	エッジゲートウェイを識別するための ID です。
3	CLM(WebAPI) CLM(WebUI)	CLM の WebAPI サーバ、WebUI の略称です。
4	CLM(cmd) CLA(cmd) CLA(デーモン)	CLM サーバにインストールする CLA のコマンド、およびエッジデバイスにインストールする CLA のコマンド、デーモンの略称です。
5	CLA(簡易 WebUI)	CLA の簡易 WebUI の略称です。

2 起動と停止

本章では、CLM、CLA(簡易 WebUI、デーモン)の起動と停止、再起動の手順を説明します。

2.1 CLM の起動

CLM を起動するには、Tomcat と、CLM のリポジトリである PostgreSQL を起動します。

Linux (RHEL/CentOS 6.8)

1. PostgreSQL を起動します。

   # service postgresql-9.6 start

2. Tomcat を起動します。

   # /usr/local/tomcat/bin/startup.sh

Linux (RHEL/CentOS 7)

1. PostgreSQL を起動します。

   # systemctl start postgresql-9.6

2. Tomcat を起動します。

   # /usr/local/tomcat/bin/startup.sh

Windows

tomcat8-clm サービスを起動することで、PostgreSQL も起動します。

1. スタートメニューから「Windows 管理ツール」-「サービス」をクリックします。
2. 「サービス」画面が表示されます。サービス一覧から「tomcat8-clm」を選択し、「起動」ボタンをクリックします。

2.2 CLM の停止

CLM を停止するには、Tomcat と、CLM のリポジトリである PostgreSQL を停止します。

Linux (RHEL/CentOS 6.8)

1. Tomcat を停止します。

   # /usr/local/tomcat/bin/shutdown.sh

2. PostgreSQL を停止します。

   # service postgresql-9.6 stop

Linux (RHEL/CentOS 7)

1. Tomcat を停止します。

   # /usr/local/tomcat/bin/shutdown.sh

2. PostgreSQL を停止します。

   # systemctl stop postgresql-9.6

Windows

tomcat8-clm サービスを停止することで、PostgreSQL も停止します。

1. スタートメニューから「Windows 管理ツール」-「サービス」をクリックします。
2. 「サービス」画面が表示されます。サービス一覧から「tomcat8-clm」を選択し、「停止」ボタンをクリックします。

2.3 CLM の再起動

CLM を再起動するには、Tomcat を再起動します。

Linux (RHEL/CentOS 6.8)

1. Tomcat を停止します。

   # /usr/local/tomcat/bin/shutdown.sh

2. Tomcat を起動します。

   # /usr/local/tomcat/bin/startup.sh

Linux (RHEL/CentOS 7)

1. Tomcat を停止します。

   # /usr/local/tomcat/bin/shutdown.sh

2. Tomcat を起動します。

   # /usr/local/tomcat/bin/startup.sh

Windows

1. スタートメニューから「Windows 管理ツール」-「サービス」をクリックします。
2. 「サービス」画面が表示されます。サービス一覧から「tomcat8-clm」を選択し、「再起動」ボタンをクリックします。

2.4 CLA(簡易 WebUI)の起動

CLA(簡易 WebUI)を起動するには、lighttpd または Tomcat を起動します。

Linux (エッジゲートウェイ)

lighttpd を起動します。

/etc/init.d/lighttpd start

Linux (Debian、AGL)

lighttpd を起動します。

/etc/init.d/lighttpd start

Linux (RHEL/CentOS 6.8、RHEL/CentOS 7)

Tomcat を起動します。

/usr/local/tomcat/bin/startup.sh

Windows (エッジゲートウェイ、WindowsOS)

Tomcat を起動します。

1. スタートメニューから「Windows 管理ツール」-「サービス」をクリックします。
2. 「サービス」画面が表示されます。サービス一覧から「tomcat8-cla」を選択し、「再起動」ボタンをクリックします。

2.5 CLA(簡易 WebUI)の停止

CLA(簡易 WebUI)を停止するには、lighttpd または Tomcat を停止します。

Linux (エッジゲートウェイ)

lighttpd を停止します。

/etc/init.d/lighttpd stop

Linux (Debian、AGL)

lighttpd を停止します。

/etc/init.d/lighttpd stop

Linux (RHEL/CentOS 6.8、RHEL/CentOS 7)

Tomcat を停止します。

/usr/local/tomcat/bin/shutdown.sh

Windows (エッジゲートウェイ、WindowsOS)

Tomcat を停止します。

1. スタートメニューから「Windows 管理ツール」-「サービス」をクリックします。
2. 「サービス」画面が表示されます。サービス一覧から「tomcat8-cla」を選択し、「再起動」ボタンをクリックします

2.6 CLA(簡易 WebUI)の再起動

CLA(簡易 WebUI)を再起動するには、lighttpd または Tomcat を再起動します。

Linux (エッジゲートウェイ)

lighttpd を再起動します。

/etc/init.d/lighttpd restart

Linux (Debian、AGL)

lighttpd を再起動します。

/etc/init.d/lighttpd restart

Linux (RHEL/CentOS 6.8、RHEL/CentOS 7)

Tomcat を再起動します。

1. Tomcat を停止します。

   # /usr/local/tomcat/bin/shutdown.sh

2. Tomcat を起動します。

   # /usr/local/tomcat/bin/startup.sh

Windows (エッジゲートウェイ、WindowsOS)

Tomcat を再起動します。

1. スタートメニューから「Windows 管理ツール」-「サービス」をクリックします。
2. 「サービス」画面が表示されます。サービス一覧から「tomcat8-cla」を選択し、「再起動」ボタンをクリックします

2.7 CLA(デーモン)の起動

CLA(デーモン)を起動するには、swcsecd を実行、または swcsecd サービスを起動します。
起動に成功すると、swcagentd プロセスが常駐します。

なお、CLA(デーモン)の自動起動・停止については、2.10 章をご覧ください。

Linux (エッジゲートウェイ、Debian)

swcsecd を実行します。

/etc/init.d/swcsecd start

Linux (RHEL/CentOS 6.8)

swcsecd サービスを起動します。

service swcsecd start

Linux (RHEL/CentOS 7、AGL)

swcsecd サービスを起動します。

systemctl start swcsecd

Windows (エッジゲートウェイ、WindowsOS)

swcsecd サービスを起動します。

1. スタートメニューから「Windows 管理ツール」-「サービス」をクリックします。
2. 「サービス」画面が表示されます。サービス一覧から「SecureWare/Credential Lifecycle Agent daemon」を選択し、「起動」ボタンをクリックします

コマンドでサービスを起動する場合は、管理者権限でコマンドプロンプトを起動し、次のコマンドを実行します。

> net start swcsecd

2.8 CLA(デーモン)の停止

CLA(デーモン)を停止するには、swcsecd を実行、または swcsecd サービスを停止します。

なお。CLA(デーモン)の自動起動・停止については、2.10 章をご覧ください。

Linux (エッジゲートウェイ、Debian)

swcsecd を実行します。

/etc/init.d/swcsecd stop -n

Linux (RHEL/CentOS 6.8)

swcsecd サービスを停止します。

service swcsecd stop -n

Linux (RHEL/CentOS 7、AGL)

swcsecd サービスを停止します。

systemctl stop swcsecd

Windows (エッジゲートウェイ、WindowsOS)

swcsecd サービスを停止します。

1. スタートメニューから「Windows 管理ツール」-「サービス」をクリックします。
2. 「サービス」画面が表示されます。サービス一覧から「SecureWare/Credential Lifecycle Agent daemon」を選択し、「停止」ボタンをクリックします

コマンドでサービスを起動する場合は、管理者権限でコマンドプロンプトを起動し、次のコマンドを実行します。

> net stop swcsecd

2.9 CLA(デーモン)の再起動

CLA(デーモン)を再起動するには、swcsecd を実行、または swcsecd サービスを再起動します。

Linux (エッジゲートウェイ、Debian)

swcsecd を実行します。

/etc/init.d/swcsecd restart -n

Linux (RHEL/CentOS 6.8)

swcsecd サービスを再起動します。

service swcsecd restart -n

Linux (RHEL/CentOS 7、AGL)

swcsecd サービスを再起動します。

systemctl restart swcsecd

Windows (エッジゲートウェイ、WindowsOS)

swcsecd サービスを再起動します。

1. スタートメニューから「Windows 管理ツール」-「サービス」をクリックします。
2. 「サービス」画面が表示されます。サービス一覧から「SecureWare/Credential Lifecycle Agent daemon」を選択し、「再起動」ボタンをクリックします

コマンドでサービスを再起動する場合は、管理者権限でコマンドプロンプトを起動し、次のコマンドを実行します。

> net stop swcsecd

> net start swcsecd

2.10　CLA(デーモン)の自動起動・停止

CLA(デーモン)は、一部の OS を除き、OS の起動・停止に合わせて自動起動・停止します。

OS 起動・停止時に自動起動・停止しないようにするには、手動で設定を削除します。

また、設定削除後、再度 OS 起動・停止時に自動起動・停止するようにする場合は、自動起動・停止を再設定します。

Linux (エッジゲートウェイ)

自動起動・停止設定の変更はできません。これは、エッジゲートウェイの仕様です。

Linux (Debian)

自動起動・停止しないようにするには、insserv コマンドで自動起動・停止設定を削除します。

```
# insserv -r swcsecd
```

自動起動・停止を再開するには、insserv コマンドで自動起動・停止設定を行います。

```
# insserv -d swcsecd
```

Linux (RHEL/CentOS 6.8)

自動起動・停止しないようにするには、chkconfig コマンドで自動起動・停止設定を削除します。

```
# chkconfig --del swcsecd
```

自動起動・停止を再開するには、chkconfig コマンドで自動起動・停止設定を行います。

```
# chkconfig --add swcsecd
```

Linux (RHEL/CentOS 7、AGL)

自動起動・停止しないようにするには、systemctl コマンドで自動起動・停止設定を無効化します。

```
# systemctl disable swcsecd
```

自動起動・停止を再開するには、systemctl コマンドで自動起動・停止設定を有効化します。

```
# systemctl daemon-reload
# systemctl enable swcsecd
```

Windows (エッジゲートウェイ、WindowsOS)

自動起動・停止しないようにするには、swcsecd サービスの設定を変更します。

1. スタートメニューから「Windows 管理ツール」-「サービス」をクリックします。
2. 「サービス」画面が表示されます。サービス一覧から「SecureWare/Credential Lifecycle Agent daemon」を選択し、右クリックします。
3. ポップアップメニューが表示されます。「プロパティ」をクリックします。
4. 「(ローカル コンピューター) SecureWare/Credential Lifecycle Agent daemon のプロパティ」ダイアログが表示されます。「全般」タブの「スタートアップの種類」を「自動」から「手動」に変更します。
5. ダイアログ下部の「OK」ボタンをクリックします。

コマンドで設定を変更する場合は、管理者権限でコマンドプロンプトを起動し、次のコマンドを実行します。

```
> sc config swcsecd start=demand
```

自動起動・停止を再開するには、swcsecd サービスの設定を変更します。

1. スタートメニューから「Windows 管理ツール」-「サービス」をクリックします。
2. 「サービス」画面が表示されます。サービス一覧から「SecureWare/Credential Lifecycle Agent daemon」を選択し、右クリックします。
3. ポップアップメニューが表示されます。「プロパティ」をクリックします。
4. 「(ローカル コンピューター) SecureWare/Credential Lifecycle Agent daemon のプロパティ」ダイアログが表示されます。「全般」タブの「スタートアップの種類」を「手動」から「自動」に変更します。
5. ダイアログ下部の「OK」ボタンをクリックします。

コマンドで設定を変更する場合は、管理者権限でコマンドプロンプトを起動し、次のコマンドを実行します。

```
> sc config swcsecd start=auto
```

3　設定ファイル

本章では、CLM、CLA の設定情報を保持するファイルについて説明します。

3.1　CLM の設定ファイル

3.1.1 WebAPI システム設定ファイル

WebAPI システム設定ファイルは、以下のファイルです。

[Linux]

/etc/swclm/swclm.properties

[Windows]

c:¥swclm¥conf¥swclm¥swclm.properties

本設定ファイルには、WebAPI の設定を記述します。ファイルの文字コードは UTF-8 です。

WebAPI システム設定ファイルの変更を反映するには、CLM の再起動が必要です。

書式

「パラメータ名=設定値」の形式で指定します。

「#」で始まる行は、コメント行と扱います。

設定項目

WebAPI システム設定ファイルの設定項目を下表に示します。

WebAPI システム設定ファイル上に記載があり下表に記載のない項目は、WebAPI が内部で使用する設定であり、通常は変更できません。

表 3-1 WebAPI 環境設定ファイル

項目名	必須/任意	説明
KeyLength	任意	パスワードの長さを byte 単位で指定します。1～128 の範囲で指定可能です。既定値は「8」です。
keyUseNumber	任意	パスワードに数字を利用するかどうかを指定します。true または false を指定します。既定値は「true」です。
keyUseUppercharacter	任意	パスワードに英大文字を利用するかどうかを指定します。true または false を指定します。既定値は「true」です。

keyUseLowercharacter	任意	パスワードに英大文字を利用するかどうかを指定します。 true または false を指定します。既定値は「true」です。
keyIDLength	任意	ID(プレフィックス＋数値(16 進数))の数値部分の長さを byte 単位で指定します。 1～16 の範囲で指定可能です。既定値は「10」です。 生成した ID の数値部分の桁数が keyIDLength の設定値に満たない場合は、0 でパディングを行います。
keyIDPrefix	任意	ID(プレフィックス＋数値(16 進数))のプレフィックス部分を指定します。プレフィックス長 0(設定なし)～16 の範囲で指定可能です。指定可能文字種は、英数字、記号(-_@.)です。既定値はありません。
keyPrefix	任意	パスワードのプレフィックスを指定します。 使用可能文字種は、ASCII コードで 0x21 から 0x7e です。 既定値はありません。
keyMode	必須	パスワード生成モードを指定します。 「sequential」(連番)「random」(ランダム)から選択できます。既定値は「random」です。
keyExpiration	必須	パスワードの有効期限を指定します。 単位は、「日」です。0 を設定した場合、有効期限は無期限となります。既定値は「36500」です。
certificateAuthority	必須	どの証明機関を使用するかを指定します。 「OpenSSL」固定です。
mode	必須	発行・取得する証明書の種別を指定します。 以下のいずれかを指定します。既定値は「caonly」です。 caonly： CA 証明書を取得 client ： サーバ証明書、クライアント証明書を取得 「caonly」「client」から選択できます。 本設定値は、WebAPI や WebAPI 実行コマンドで mode の指定がない場合に利用されます。
cacerttype	必須	CA 証明書の種別を指定します。 以下のいずれかを指定します。既定値は「pem」です。 pem: pem 形式の証明書 der：der 形式の証明書 本設定値は、WebAPI や WebAPI 実行コマンドで cacerttype の指定がない場合に利用されます。

clientcerttype	必須	クライアント証明書の種別を指定します。 以下のいずれかを指定します。既定値は「pem」です。 pem: pem 形式の証明書 der：der 形式の証明書 p12：PKCS#12 形式の証明書(chain なし) 本設定値は、WebAPI や WebAPI 実行コマンドで clientcerttype の指定がない場合に、利用されます。
cacertfile	必須	CA 証明書のパスを絶対パス、または CLM インストール PATH からの相対パスで指定します。 指定する CA 証明書は、PEM 形式である必要があります。 既定値は次の通りです。 [Linux] /usr/local/myopenssl/ca/ca1/cacert.pem [Windows] c:¥myopenssl¥ca¥ca1¥cacert.pem
clientcertpath	必須	発行したクライアント証明書、秘密鍵、CSR の格納ディレクトリを絶対パス、または CLM インストール PATH からの相対パスで指定します。 既定値は、次の通りです。 [Linux] /opt/nec/pf/swclm/SWCLM/cert [Windows] c:¥swclm¥SWCLM¥cert
opensslpath	必須	OpenSSL コマンド格納ディレクトリの PATH を指定します。 既定値は次の通りです。 [Linux] /usr/local/myopenssl/bin [Windows] c:¥myopenssl¥bin
opensslconfpath	必須	OpenSSL 設定ファイルまでの PATH を指定します。 既定値は次の通りです。 [Linux] /usr/local/myopenssl/ca/openssl.cnf [Windows] c:¥myopenssl¥ca¥openssl.cnf

capassphrase	必須	CA 証明書の秘密鍵のパスフレーズを指定します。 CA 証明書の秘密鍵として入力したパスフレーズと本項目に指定するパスフレーズは一致している必要があります。 通常は、本設定を変更する必要はありません。
opensslLockTimeout	任意	openssl コマンド呼出し時のロック解放待ちタイムアウト値を指定します。単位は、「秒」です。既定値は「10」です。
commonKeyExpiration	必須	共通鍵の有効期限を指定します。 単位は、「日」です。0 を設定した場合、有効期限は無期限となります。既定値は「36500」です。
commonKeyLength	必須	共通鍵の長さを指定します。 単位は、「bit」です。1～1024 の範囲で指定可能です。 既定値は「128」です。
dataBase	必須	どの DB を利用するかを指定します。「PostgreSQL」固定です。
dbHost	必須	データベースインストールサーバのホスト名または IP アドレスを指定します。既定値は「localhost」です。
dbPost	必須	データベースの待ち受けポート番号を指定します。既定値は「5432」です。
dbName	必須	データベース名を指定します。既定値は「swclm」です。
dbUser	必須	データベース接続ユーザ名を指定します。既定値は「swclm」です。
dbPassword	必須	データベース接続ユーザのパスワードを指定します。 通常は、本設定を変更する必要はありません。
dbSSL	任意	データベースアクセス時に SSL 通信を行うかどうかを指定します。true または false を指定します。既定値は「false」です。
systemStatusInterval	任意	稼働状態確認 API でデータベース確認を行う最小インターバルを指定します。単位は、「秒」です。既定値は「60」です。
deviceKeyPrefix	任意	デバイスキーのプレフィックスを指定します。 使用可能文字種は、ASCII コードで 0x21 から 0x7e です。 既定値はありません。

checkDeviceDuplication	任意	(EDMS オプション導入時) デバイス登録・デバイス一括登録実行時に、デバイス名の一意性チェックを行うかどうかを指定します。 登録するデバイス名の一意性が保証されている場合に、本機能を無効にすることでデバイス登録・デバイス一括登録の性能が向上します。 次のいずれかを指定します。既定値は「true」です。 true: 一意性チェックを有効化 false: 一意性チェックを無効化
maxResponseRecord	任意	参照系 WebAPI 使用時、CLM から返却するデータの最大件数を指定します。既定値は「10000(件)」です。
keyCheckTerm	任意	(EDMS オプション導入時) デバイス ID 鍵情報取得 WebAPI で返却する鍵の有効期限チェックの閾値を指定します。 単位は、「日数」です。既定値は「30(日)」です。
statisticsHistory	任意	統計情報世代管理数を指定します。 1 以上の値を指定します。1 を指定した場合は、最新の統計情報のみを CLM で管理します。 既定値は「1」です。
statisticsCollectMode	任意	統計情報収集タイミングを設定します。 次のいずれかの値を指定します。既定値は「0」です。 0：統計情報を収集しない 1：定期間隔で収集する 2：曜日、時刻指定で収集する
statisticsCollectInterval	※1	統計情報収集間隔を設定します。 統計情報収集間隔とは、統計情報収集完了から、次回統計情報収集開始までの時間をさします。 単位は、「時間」です。 ※1 statisticsCollectMode で 1 を設定した場合、本設定は必須です。

statisticsCollectDayofWeek	※2	統計情報収集を行う曜日を設定します。 曜日は、次の形式で指定します。 Sun: 日曜日 Mon: 月曜日 Tue: 火曜日 Wed: 水曜日 Thu: 木曜日 Fri: 金曜日 Sat: 土曜日 曜日は、複数指定可能です。複数指定する場合は、カンマで区切って指定します。 例) 月、水、金に実行する場合、「Mon,Wed,Fri」と指定します。 ※2 statisticsCollectMode で 2 を設定した場合、本設定は必須です。
statisticsCollectTime	※3	統計情報収集を行う時刻を設定します。 時刻は、24 時間表記形式(hh:mm)で指定します。 ※3 statisticsCollectMode で 2 を設定した場合、本設定は必須です。

3.1.2 WebAPI ログ設定ファイル

WebAPI ログ設定ファイルは、以下のファイルです。

[Linux 版]

/etc/swclm/log4j2.xml

[Windows 版]

c:¥swclm¥conf¥swclm¥log4j2.xml

本設定ファイルには、WebAPI のログ出力に関する設定を記述します。ファイルの文字コードは UTF-8 です。

ログ設定ファイルの変更を反映させるためには、CLM の再起動が必要です。

書式

XML の仕様に従います。

設定項目

WebAPI ログ設定ファイルの設定項目を下表に示します。

WebAPI ログ設定ファイル上に記載があり下表に記載のない項目は、WebAPI が内部で使用する設定であり、通常は変更できません。

表 3-2 WebAPI ログ設定ファイル（operate.log、error.log）

項目名	説明
[Linux]	
<Syslog name="syslog_operate"> <Syslog name="syslog_error">	
Host	ログを出力する syslog サーバを指定します。既定値は「localhost」です。
Facility	syslog の facility を指定します。 ログ出力先 syslog サーバで未使用の facility を指定します。 既定値は、次の通りです。 operate.log: LOCAL5 error.log : LOCAL6
<Logger name=“OPERATE”> <Logger name=“ERROR”>	
Level	ログレベルを指定します。 指定可能なレベルは、「error」、「warn」、「info」、「debug」、「fatal」です。
[Windows]	
<RollingFile name="file_operate”> <RollingFile name="file_error">	
<SizeBasedTriggeringPolicy>	
Size	ログファイルの最大ファイルサイズを指定します。
<DefaultRolloverStrategy>	
max	ローテーションしたファイルのバックアップ数を指定します。 ログローテート時、カレントのログファイルの他に、指定した数のバックアップファイルを保持します。
<Logger name=“OPERATE”> <Logger name=“ERROR”>	

	Level	ログレベルを指定します。 指定可能なレベルは、「error」、「warn」、「info」、「debug」、「fatal」です。

表 3-3 WebAPI ログ設定ファイル（trace.log、notice.log）

項目名			説明
<RollingFile name="file_trace"> <RollingFile name="file_notice">			
	<SizeBasedTriggeringPolicy>		
		Size	ログファイルの最大ファイルサイズを指定します。
	<DefaultRolloverStrategy>		
		max	ローテーションしたファイルのバックアップ数を指定します。 ログローテート時、カレントのログファイルの他に、指定した数のバックアップファイルを保持します。
<Logger name="TRACE"> <Logger name="NOTICE">			
	Level		ログレベルを指定します。 指定可能なレベルは、「error」、「warn」、「info」、「debug」、「fatal」です。 なお、トレースログを出力したい場合は、「debug」を設定します。

3.1.3 WebUI 設定ファイル

WebUI 設定ファイルは、以下のファイルです。

[Linux]

/etc/swclm/swclm.conf

[Windows]

c:¥swclm¥conf¥swclm¥swclm.conf

本設定ファイルには、WebUI の設定を記述します。ファイルの文字コードは UTF-8 です。

WebUI 設定ファイルの変更を反映するには、CLM の再起動が必要です。

書式

「パラメータ名=設定値」の形式で指定します。

「#」で始まる行は、コメント行と扱います。

設定項目

WebUI 設定ファイルの設定項目を下表に示します。

WebUI 設定ファイル上に記載があり下表に記載のない項目は、WebUI が内部で使用する設定であり、通常は変更できません。

表 3-4 WebUI 環境設定ファイル

項目名	必須/任意	説明
logfacility	任意	syslog の facility を指定します。 ログ出力先 syslog サーバで未使用の facility を指定します。 既定値は、「local4」です。
loglevel	任意	ログレベルを指定します。 指定可能なレベルは、「NONE」、「DEBUG」、「ERROR」、「WARNING」、「INFO」です。 「NONE」を指定した場合、ログを出力しません。 既定値は「ERROR」です。
AWS_PATH	任意	AWS CLI(aws コマンド)のインストール PATH を指定します。 既定値は「/usr/bin」です。
NOTICE_FILE	任意	ID・鍵状態取得で監視する Notice.log までの PATH を指定します。 既定値は、「/var/log/swclm/notice.log」です。
CA_BUNDLE	任意	CA 証明書までの PATH を指定します。 AWS IoT 接続設定(10 章)で、HTTPS 通信に使用する CA 証明書が問題で HTTPS 通信がエラーとなる場合に指定します。 既定値は、「/etc/pki/tls/certs/ca-bundle.crt」です。

3.2 CLA の設定ファイル

3.2.1 WebUI 設定ファイル

CLM の WebUI 設定ファイルと同内容を設定可能です。3.1.3 章をご覧ください。

3.2.1 daemon 設定ファイル

daemon 設定ファイルは、以下のファイルです。

[Linux]

/etc/swcagent/swclmclient.conf

[Windows]

c:¥swclm¥conf¥swcagent¥swclmclient.conf

本設定ファイルには、CLA(デーモン)の設定を記述します。ファイルの文字コードは UTF-8 です。

daemon 設定ファイルの変更を反映するには、CLA(デーモン)の再起動が必要です。

書式

「パラメータ名=設定値」の形式で指定します。

「#」で始まる行は、コメント行と扱います。

設定項目

daemon 設定ファイルの設定項目を下表に示します。

daemon 設定ファイル上に記載があり下表に記載のない項目は、daemon が内部で使用する設定であり、通常は変更できません。

表 3-5 daemon 環境設定ファイル

項目名	必須/任意	説明
host	必須	CLM のホスト名または IP アドレスを指定します。
port	任意	CLM の待ち受けポート番号を指定します。 既定値は「8443」です。
p-host	任意	プロキシサーバの IP アドレスを指定します。 CLA(デーモン)から CLM への通信時に Web プロキシサーバを経由しなければ通信できないネットワーク構成である場合は、本項目を設定してください。

p-port	任意	プロキシサーバのポート番号を指定します。 CLA(デーモン)から CLM への通信時に Web プロキシサーバを経由しなければ通信できないネットワーク構成である場合は、本項目を設定してください。
timeout	任意	タイムアウト時間を指定します。単位は秒です。既定値は 60 秒です。指定可能な値は 1～0x7FFFFFFF です。
debug	任意	CLA(デーモン)のデバッグログを出力する場合に「1」を指定します。
ext-data	必須	拡張情報を指定します。 key=value の形式で指定してください。 value は、クォートで囲む必要があります。また、複数指定する場合はカンマで区切る必要があります。 指定する拡張情報は、後述の「拡張情報」を参照してください。

表 3-6 daemon 環境設定ファイル（拡張情報）

項目名	必須/任意	説明
interval-c	任意	CLM から配布された証明書の改竄を検出する間隔を指定します。単位は、「秒」です。 既定値は「86400」(24 時間)です。
interval-k	任意	CLM から配布された共通鍵の改竄を検出する間隔を指定します。単位は、「秒」です。 既定値は「86400」(24 時間)です。
interval-p	任意	鍵自動更新・統計情報表示(9 章)で、CLM に操作問合せを行う間隔を指定します。単位は、「秒」です。 既定値は「86400」(24 時間)です。 なお、本項目は、CLM からポーリング間隔(表 9-5)を取得できない場合に使用します。
waittime	任意	CLA(デーモン)が CLM に接続しようとしてサーバ Busy が発生した場合等に、CLA(デーモン)が再接続を待ち合わせる時間を指定します。 単位は「秒」です。既定値は「3」です。
retry	任意	CLA(デーモン)が CLM に接続しようとしてサーバ Busy が発生した場合等に、CLA(デーモン)が再接続を何回試みるか指定します。 既定値は「3」です。

4 ログ出力

本章では、CLM、CLA が出力するログについて説明します。

4.1 ログの種類

CLM、CLA では以下のログを出力します。各ログの詳細は、4.2 章以降で説明します。

表 4-1 ログ種別一覧

ログ種別	説明	出力先
操作ログ	ID 鍵の管理等、CLM(WebAPI)が行った操作に関するログを出力する。	[Linux] /var/log/swclm/operate.log [Windows] c:¥swclm¥logs¥operate.log
エラーログ	CLM(WebAPI)で発生したエラーに関するログを出力する。	[Linux] /var/log/swclm/error.log [Windows] c:¥swclm¥logs¥error.log
トレースログ	CLM(WebAPI)の処理詳細(デバッグログ)を出力する。通常はログ出力なし。出力するには、設定を変更する必要がある。	[Linux] /var/log/swclm/trace.log [Windows] c:¥swclm¥logs¥trace.log
コマンドログ	CLM(cmd・daemon)、CLA(cmd・daemon)に関するログを出力する。	[Linux] /var/log/swcagent.log [Windows] イベントログ(Application)
WebUI ログ	CLM(WebUI)、CLA(簡易 WebUI)に関するログを出力する。	[Linux] /var/log/swclm/swclmweb.log [Windows] イベントログ(Application)

4.2 操作ログ

ID・パスワードに関する操作や証明書に関する操作など、CLM がクライアントからの操作要求とその結果を出力するファイルです。

4.2.1 ログ出力フォーマット

操作ログのログ出力フォーマットは、次の通りです。出力内容は、クライアントから要求された操作により異なります。

- ID パスワード発行/ID パスワード取得/ID パスワード照合/ID パスワード削除

<日時>, <レベル>, <スレッド名>, <操作>, <結果>, <鍵 ID>, <IP アドレス>, <接続元 IP アドレス>, <処理時間>

- 証明書発行

<日時>, <レベル>, <スレッド名>, <操作>, <結果>, <鍵 ID>, <モード>, <デバイス ID>, <CA 名称>, <サブジェクト>, <IP アドレス>, <接続元 IP アドレス>, <処理時間>

- 証明書取得/証明書失効

<日時>, <レベル>, <スレッド名>, <操作>, <結果>, <鍵 ID>, <鍵番号>, <デバイス ID>, <CA 名称>, <シリアル番号>, <ファイルパス>, <IP アドレス>, <接続元 IP アドレス>, <処理時間>

- 証明書更新/共通鍵更新

<日時>, <レベル>, <スレッド名>, <操作>, <結果>, <鍵 ID>, <デバイス ID>, <更新元の鍵番号>, <更新元のファイルパス>, <更新後の鍵番号>, <更新後のファイルパス>, <IP アドレス>, <接続元 IP アドレス>, <処理時間>

- 共通鍵発行/共通鍵取得/共通鍵削除

<日時>, <レベル>, <スレッド名>, <操作>, <結果>, <鍵 ID>, <デバイス ID>, <鍵番号>, <ファイルパス>, <IP アドレス>, <接続元 IP アドレス>, <処理時間>

- 共通鍵照合

<日時>, <レベル>, <スレッド名>, <操作>, <結果>, <鍵 ID>, <デバイス ID>, <照合した共通鍵の鍵番号>, <ファイルパス>, <IP アドレス>, <接続元 IP アドレス>, <処理時間>

- グループ化規則名登録/取得/変更・グループ化規則変更／削除／有効化／無効化

<日時>, <レベル>, <スレッド名>, <操作>, <結果>, <鍵 ID>, <グループ化規則番号>, <IP アドレス>, <接続元 IP アドレス>, <処理時間>

- グループ化規則取得

<日時>, <レベル>, <スレッド名>, <操作>, <結果>, <鍵 ID>, <グループ化規則番号>, <グループ番号>, <IP アドレス>, <接続元 IP アドレス>, <処理時間>

- 操作問合せ

<日時>, <レベル>, <スレッド名>, <操作>, <結果>, <鍵 ID>, <デバイス ID>, <IP アドレス>, <接続元 IP アドレス>, <処理時間>

- リモート実行設定

<日時>, <レベル>, <スレッド名>, <操作>, <結果>, <鍵 ID>, <デバイス ID>, <操作内容>, <IP アドレス>, <接続元 IP アドレス>, <処理時間>

4.2.2 ログ出力内容詳細

操作ログに出力される内容は、次の通りです。

表 4-2 操作ログ 出力内容

項目	説明
日時	ログ出力日時。 フォーマットは/etc/rsyslog.conf の設定「$ActionFileDefaultTemplate」の設定に従う。 デフォルトは、「Mon DD hh:mm:ss」。
レベル	ログ出力レベル。「INFO」固定。
操作	表 4-3 を参照。
結果	表 4-4 を参照。
鍵 ID	ID パスワード発行 API で発行した ID
鍵番号	処理対象の証明書または共通鍵の鍵番号。 共通鍵一括取得で複数の鍵番号が要求された場合には、スペースで区切って鍵番号を出力。
デバイス ID	処理対象のデバイスの ID
CA 名称	処理対象の証明書の CA 名称
シリアル番号	処理対象の証明書のシリアル番号
モード	証明書発行モード（CAOnly/Client）
サブジェクト	発行する証明書のサブジェクト
ファイルパス	証明書または共通鍵の保存ファイルパス
IP アドレス	HTTP Request から取得した接続元の IP アドレス
照合した共通鍵の鍵番号	照合対象の共通鍵の鍵番号
グループ化規則番号	処理対象のグループ化規則の番号（登録の場合は、自動生成した番号）
グループ番号	処理対象のグループの番号
操作内容	リモート実行設定で設定された操作内容
接続元 IP アドレス	HTTP ヘッダから取得した送信元の IP アドレス (X-Forwarded-For ヘッダの値。ヘッダがなければ IP アドレスと同じ値)

処理時間	各操作の処理時間

表 4-3 操作ログ 出力内容 (操作種別)

種別	説明
CREATE	ID パスワード発行
GET	ID パスワード取得
VERIFY	ID パスワード照合
DELETE	ID パスワード削除
CERT	証明書発行
CERTGET	証明書取得
CERTUPDATE	証明書更新
CERTREVOKE	証明書失効
KEYCREATE	共通鍵発行
KEYGET	共通鍵取得
KEYUPDATE	共通鍵更新
KEYDELETE	共通鍵削除
KEYVERIFY	共通鍵照合
KEYGETLIST	共通鍵一括取得
RULECREATE	グループ化規則名登録
RULEGET	グループ化規則名取得
RULEUPDATE	グループ化規則名更変更
RULEDELETE	グループ化規則削除
RULEACTIVATE	グループ化規則有効化
RULEINACTIVATE	グループ化規則無効化
GROUPSUPDATE	グループ規則変更
GROUPGET	グループ規則取得
RULECREATE	グループ化規則名登録
RULEGET	グループ化規則名取得
QUERUDEVOPE	操作問合せ
SETDEVOPE	リモート実行設定

表 4-4 操作ログ 出力内容 (実行結果)

実行結果	説明
SUCCESS	成功
BADREQT	リクエスト情報エラー

DIFFKEY	鍵情報が一致しない
KEYINVL	鍵のステータスが有効ではない
KEYEXPR	鍵が有効期限切れ
CERTEXPR	証明書が有効期限切れ
KEYOVER	鍵 ID がオーバーフロー
NOKEYID	鍵が登録されていない
NOCERT	証明書が登録されていない
NOCOMKEY	共通鍵が登録されていない
NOLINKAGE	紐付情報が登録されていない
INVCOMKEY	共通鍵情報不正
INVCERT	証明書情報不正
INVDEVICE	デバイス情報不正
CERTERR	証明書ファイルエラー
CONFERR	設定エラー
SSLERR	OpenSSL エラー
CRYPTERR	暗号化エラー
LOGERR	ログ出力エラー
DBERR	データベースエラー
DIFFDEVKEY	デバイスキーが一致しない
DEVKEYINV	デバイスのステータスが有効でない
DEVKEYEXPR	デバイスが有効期限切れ(発生しない)
NODEVICE	デバイスが登録されていない
DIFFCOMKEY	共通鍵が一致しない
COMKEYERR	共通鍵ファイルアクセスエラー
ERROR(エラーコード)	その他エラー（システムエラー）

4.3 エラーログ

CLM の運用中にエラーが発生した場合にエラー内容を出力するファイルです。

4.3.1 ログ出力フォーマット

エラーログのログ出力フォーマットは、次の通りです。

<日時>, <レベル>, <スレッド名>, <メッセージ>

4.3.2 ログ出力内容詳細

エラーログに出力される内容は、次の通りです。

表 4-5 エラーログ 出力内容

項目	説明
日時	ログ出力日時。 フォーマットは次の通り。 [Linux 版] /etc/rsyslog.conf の設定「$ActionFileDefaultTemplate」の設定に従う。 デフォルトは、「Mon DD hh:mm:ss」。 [Windows 版] 「yyyy-MM-dd hh:mm:ss.sss」固定。
レベル	ログ出力レベル。表 4-6 を参照。
メッセージ	エラー内容詳細。

表 4-6 エラーログ 出力内容（ログ出力レベル）

種別	説明
DEBUG	デバッグ用のログレベル
INFO	運用の際に、最低限必要なレベルのログを出力
WARN	何らかの問題が発生したが、不完全ながらもリクエストは完結している状態のログを出力
ERROR	何らかの問題が発生し、運用に際し大きな問題がある状態を示すログを出力

4.4 トレースログ

CLM の処理詳細(デバッグログ)を出力します。

通常は無効化しており、ログ出力しません。障害調査のために有効化してログ採取する場合があります。

4.4.1 ログ出力フォーマット

「エラーログ」の[ログ出力フォーマット]と同様です。「エラーログ」の[ログ出力フォーマット]をご覧ください。

4.4.2 ログ出力内容詳細

トレースログに出力される内容は、次の通りです。

表 4-7 トレースログ 出力内容

項目	説明
日時	ログ出力日時。フォーマットは、「yyyy-MM-dd hh:mm:ss.sss」。
レベル	ログ出力レベル。表 4-6 を参照。
メッセージ	エラー内容詳細。

4.5 コマンドログ

CLM(cmd・daemon)、CLA(cmd・daemon)を実行した際の終了コードやエラーメッセージを出力します。

4.5.1 ログ出力フォーマット

[Linux 版]

コマンドログのログ出力フォーマットは、次の通りです。

```
<日時> <ホスト名> swcagent[プロセス ID]: <メッセージ>
```

[Windows 版]

イベントログ(Application)の出力フォーマットに従います。

4.5.2 ログ出力内容詳細

コマンドログに出力される内容は、次の通りです。

表 4-8 コマンドログ 出力内容 (Linux 版)

項目	説明
日時	ログ出力日時。 フォーマットは/etc/rsyslog.conf の設定「$ActionFileDefaultTemplate」

	の設定に従う。 デフォルトは、「Mon DD hh:mm:ss」。
プロセス ID	CLA(cmd・daemon)のプロセス ID。
メッセージ	CLA(cmd・daemon)が出力するメッセージ。

表 4-9 コマンドログ 出力内容 (Windows 版)

項目	説明
日時	ログ出力日時。
ログレベル	Informational または Error。
ソース	CLA(cmd・daemon)のソース名。詳細は表 4-10 を参照。
イベント ID	CLA(cmd・daemon)が出力するメッセージに対応するイベント ID。 詳細は表 4-10 を参照。
メッセージ	CLA(cmd・daemon)が出力するメッセージ。 イベントログの全般タブには、イベントメッセージを出力する。表 4-10 を参照。 イベントログの詳細タブには、詳細なメッセージを出力する。

表 4-10 コマンドログ イベント ID 一覧

イベント ID	イベントメッセージ		ログレベル	ソース
	日本語	英語		
1001	SecureWare/CLA コマンドは正常に終了しました。	SecureWare/CLA command completed successfully.	Informational	swclmclient
2001	SecureWare/CLA デーモンが開始されました。	SecureWare/CLA daemon is started.	Informational	swcagentd
2002	SecureWare/CLA デーモンは動作中です。	SecureWare/CLA daemon is running.	Informational	swcagentd
2003	SecureWare/CLA デーモンが停止されました。	SecureWare/CLA daemon is stopped.	Informational	swcagentd
7001	SecureWare/CLA デーモンが不正なパラメータを検出しました。	SecureWare/CLA daemon detected an invalid parameter.	Warning	swcagentd
8001	SecureWare/CLA コマンドがエラーを返しました。	SecureWare/CLA command returned an error.	Error	swclmclient
9001	SecureWare/CLA デーモン	An error occurred in	Error	swcagentd

	でエラーが発生しました。	SecureWare/CLA daemon.		

4.6 WebUI ログ

CLM(WebUI)、CLA(簡易 WebUI)操作時に、ログインユーザや処理に関するログを出力します。

4.6.1 ログ出力フォーマット

[Linux 版]

WebUI ログのログ出力フォーマットは、次の通りです。

```
<日時> <ホスト名> SWCLMWEB[プロセス ID]: <メッセージ>
```

[Windows 版]

イベントログ(Application)の出力フォーマットに従います。

4.6.2 ログ出力内容詳細

WebUI ログに出力される内容は、次の通りです。

表 4-11 WebUI ログ 出力内容 (Linux 版)

項目	説明
日時	ログ出力日時。 フォーマットは/etc/rsyslog.conf の設定 「$ActionFileDefaultTemplate」の設定に従う。 デフォルトは、「Mon DD hh:mm:ss」。
プロセス ID	CLM(WebUI)(または CLA(簡易 WebUI))のプロセス ID。
メッセージ	CLM(WebUI)(または CLA(簡易 WebUI))が出力するメッセージ。

表 4-12 コマンドログ 出力内容 (Windows 版)

項目	説明
日時	ログ出力日時。
ログレベル	Informational または Error。
ソース	以下のいずれか。 awsagent.cgi [SSW:1.0.1] kitting_proc.cgi [SSW:1.0.1] login.cgi [SSW:1.0.1] swcagent.cgi [SSW:1.0.1] unload.cgi [SSW:1.0.1]

	upload.cgi [SSW:1.0.1]
イベント ID	「144」固定。
メッセージ	CLM(WebUI)(または CLA(簡易 WebUI))が出力するメッセージ。

5 ID・パスワード管理

CLM は、エッジ-クラウド間の通信を安全に行うための ID・パスワードを発行することが可能です。また、発行済み ID・パスワードの取得、削除も可能です。

発行した ID・パスワードを使用して照合(認証)を行うことができ、認証機構を持たないアプリに認証機能を提供することも可能です。

本章では、ID・パスワード管理方法について説明します。

5.1 API による ID・パスワードの管理

ID・パスワードの管理を行うためのインターフェースとして、CLM では WebAPI を提供しています。WebAPI を使用することにより、HTTPS プロトコルを使用して ID・パスワードの発行・照合・取得・削除を行うことができます。

WebAPI の詳細については、別紙「WebAPI リファレンス」をご覧ください。

5.2 コマンドによる ID・パスワードの管理

ID・パスワードの管理を行うためのインターフェースとして、CLM ではコマンド(WebAPI 実行コマンド)を提供しています。WebAPI 実行コマンドを実行することで、容易に ID・パスワードの発行・照合・取得・削除を行うことができます。

WebAPI 実行コマンドの詳細については、別紙「コマンドリファレンス」をご覧ください。

5.3 WebUI による ID・パスワードの管理

ID・パスワードの管理を行うためのインターフェースとして、CLM では WebUI を提供しています。WebUI を使用して、ID・パスワードの発行・取得・削除を行うことが可能です。照合を行うことはできませんのでご注意ください。

本章では、WebUI による ID・パスワードの発行・取得・削除について説明します。ID・パスワードの発行・取得・削除の各機能詳細については、別紙「WebAPI リファレンス」の「3. ID・パスワード管理 API」をご覧ください。

5.3.1 WebUI 初回アクセス時の事前準備

WebUI へアクセスする端末・ブラウザに、以下の設定を行います。本設定は、初回アクセス前に行います。また、本設定を行った端末・ブラウザと異なる端末・ブラウザで WebUI へアクセスする場合は、新たに当該端末で本設定を行う必要があります。

- WebUI との通信には https 通信を使用するので、CLM が提供するビルトイン CA 証明書を OS またはブラウザへインポートします。

ビルトイン CA 証明書は、以下に格納しています。端末の OS またはブラウザの証明書インポート手順に従ってビルトイン CA 証明書をインポートしてください。

[ビルトイン CA 証明書格納先]

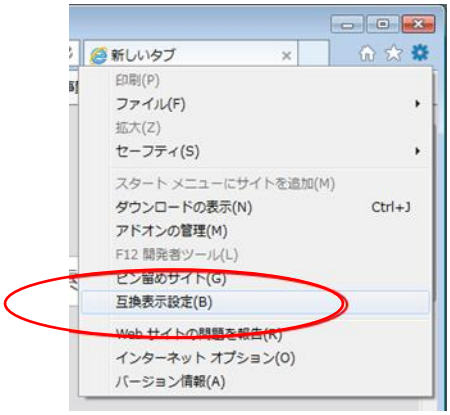
CLM インストールサーバ /opt/nec/pf/swcagent/bin/built-in-ca.pem

- WebUI アクセス時に警告を出力しないように、端末の OS の hosts ファイルに CLM インストールサーバの IP アドレスとエイリアス「swclmweb」を記載し、保存します。

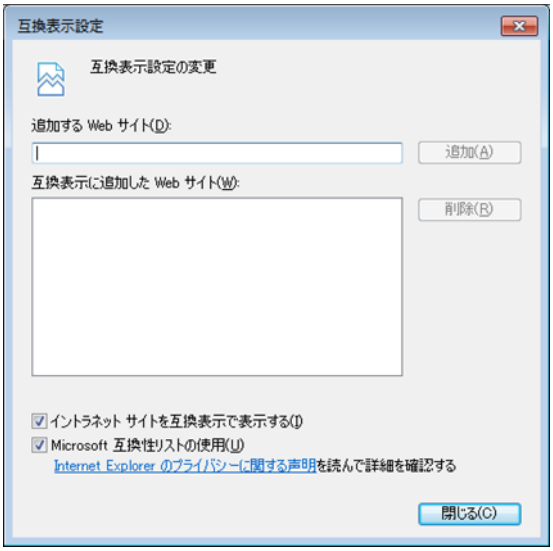
CLM インストールサーバの IP アドレス　　swclmweb

- 社内 LAN から Internet に出る際にプロキシを通す必要がある場合、ブラウザでプロキシの設定をします。
- WebUI のコンテンツの表示不正が発生しないように、ブラウザのキャッシュをクリアします。使用するブラウザのキャッシュクリア手順に従い、クリアしてください。
- 使用ブラウザが Internet Explorer の場合、互換表示設定を OFF にします。

1. Internet Explorer を起動し、互換性表示設定を開きます。



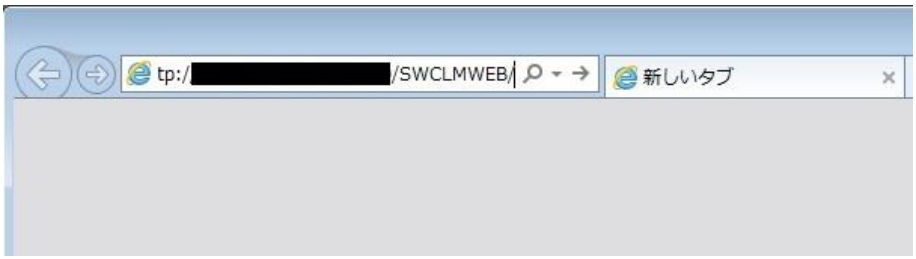
2. 互換性表示設定ダイアログが表示されます。「追加する Web サイト」に簡易 Web【サンプル】のアクセス URL を入力し、「追加」ボタンをクリックします。



5.3.2 WebUI へのログイン

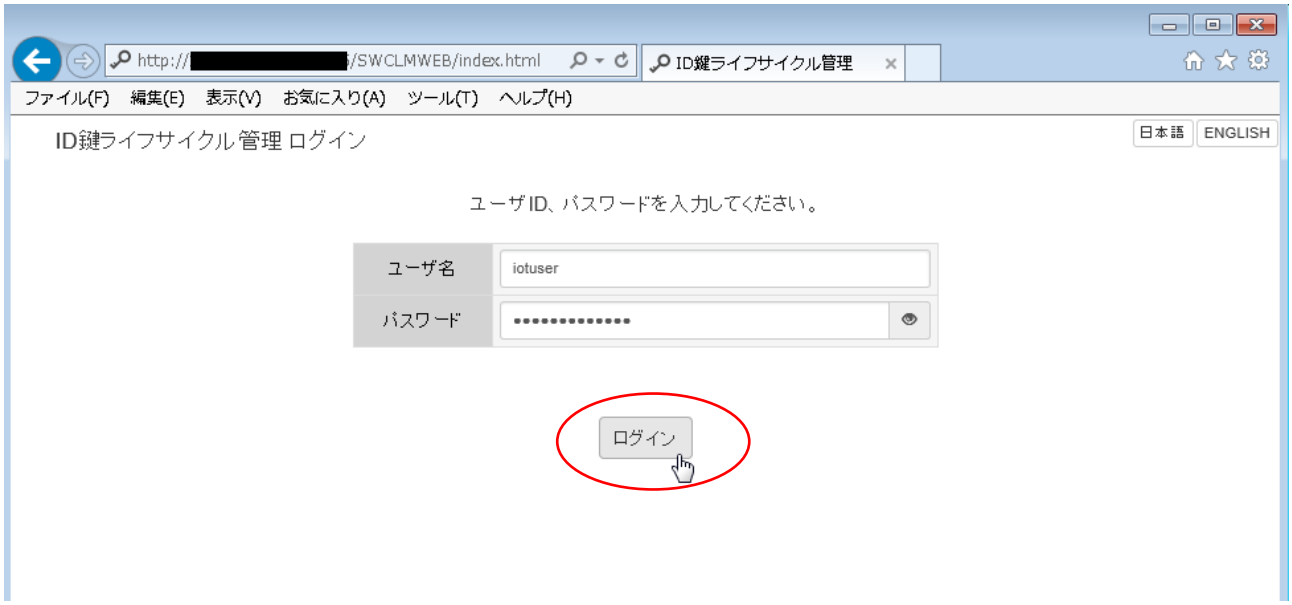
1. ブラウザで、CLM の WebUI にアクセスします。

   URL: https://swclmweb:8443/SWCLMWEB/



2. WebUI のログイン画面が表示されます。

   ユーザ名、パスワードを入力し、「ログイン」ボタンをクリックします。



◆ ユーザ名・パスワード

初期ユーザとして以下を用意しています。

ユーザ名　：iotuser パスワード：別紙「SecureWare/Credential Lifecycle Manager セットアップカード」を ご覧ください

3.　ログインに成功すると、メニューが表示されます。

5.3.3 クラウド接続設定

WebUI 初回ログイン時に、WebUI が CLM(WebAPI サーバ)へ接続するために必要な設定を WebUI 上で行う必要があります。

初回ログイン時に設定した内容は 2 回目以降のログインでも引き継がれるため、通常は 2 回目以降のログインで設定を行う必要はありません。ただし、CLM インストールサーバの IP アドレスが変更になったなど、設定した内容を変更する必要が生じた場合は、ログインの後のメニューから「クラウド接続設定」を選択し、再設定する必要があります。

メニューの「クラウド接続設定」をクリックし、接続先を設定します。

CLM インストールサーバの情報を入力し、「接続設定」ボタンをクリックします。

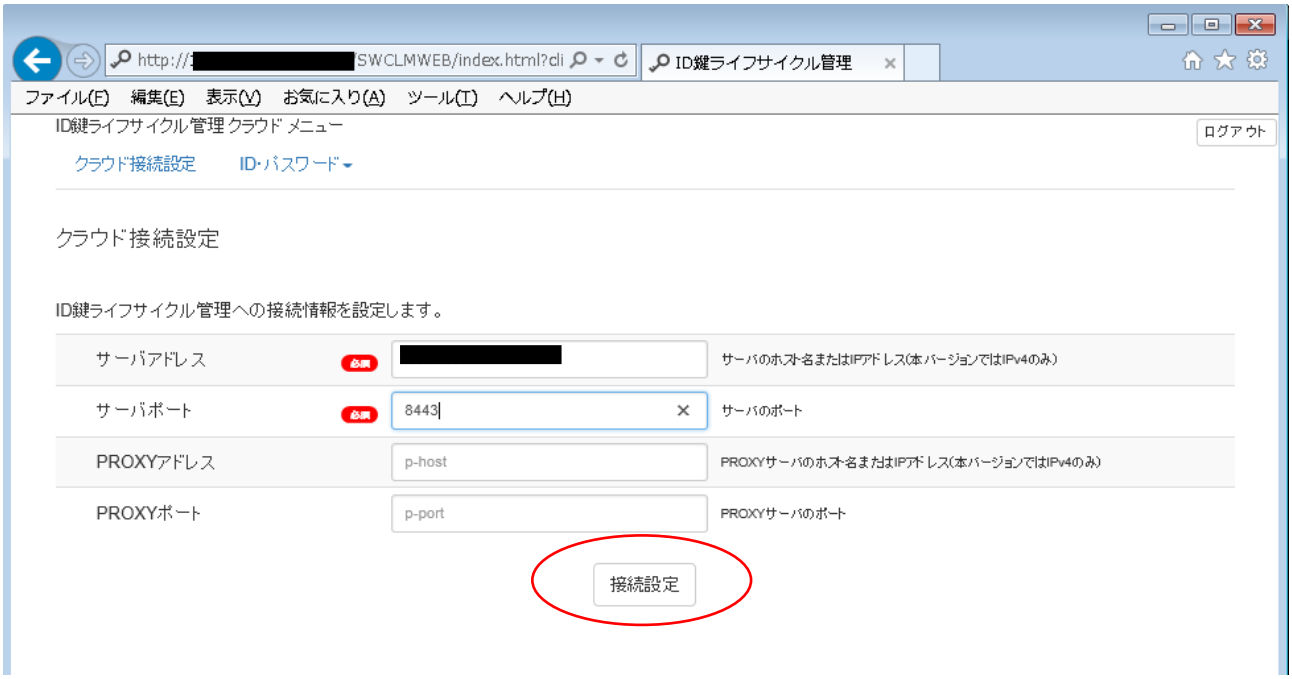


表 5-1 クラウド接続設定 入力項目

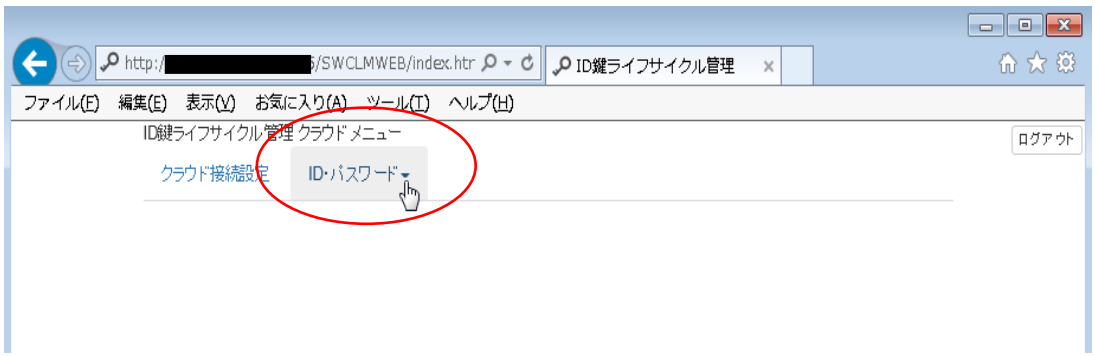
項目	説明
サーバアドレス	CLM のホスト名または IP アドレスを指定します。
サーバポート	CLM の待ち受けポート番号を指定します。
PROXY アドレス	プロキシサーバの IP アドレスを指定します。 WebUI から CLM までの通信時に Web プロキシサーバを経由しなければ通信できないネットワーク構成である場合は、本項目を設定してください。
PROXY ポート	プロキシサーバのポート番号を指定します。 WebUI から CLM までの通信時に Web プロキシサーバを経由しなければ通信できないネットワーク構成である場合は、本項目を設定してください。

接続設定が成功すると「接続設定が完了しました。」のメッセージを表示します。

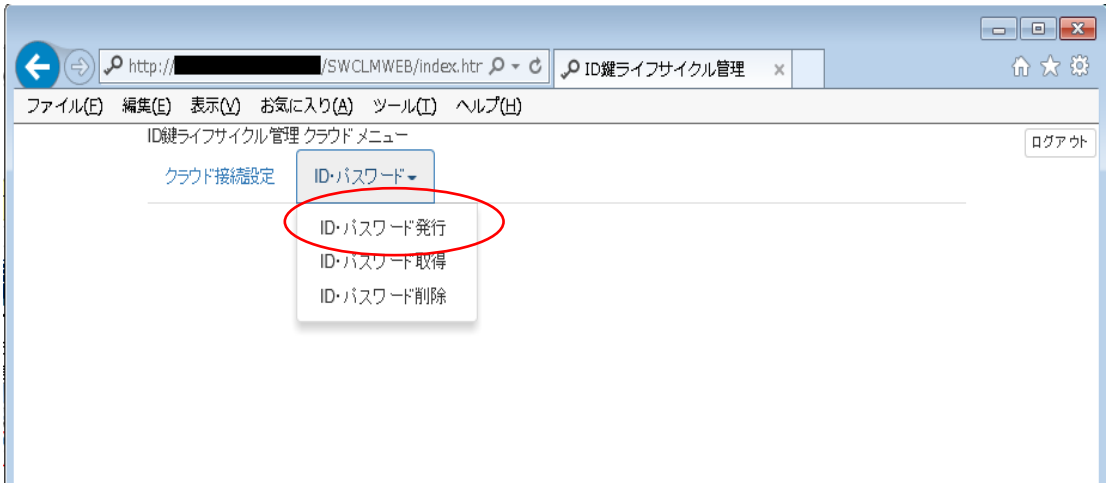
5.3.4 ID・パスワード発行

1. WebUI にログインしていない場合は、WebUI へログインします。
   ログイン手順は、5.3.2 章をご覧ください。

2. メニューの「ID・パスワード」をクリックします。



3. 「ID・パスワード」のメニュー一覧が表示されます。「ID・パスワード発行」をクリックします。



4. 「ID・パスワード発行」画面が表示されます。必要事項を入力し、「ID 発行」ボタンをクリックします。

表 5-2 ID・パスワード発行 入力項目

項目	説明
テナント ID	将来のための予約項目です。 プルダウンから「TenantA」または「TenantB」のいずれかを選択してください。
エイリアス	発行する ID・パスワードに付与するエイリアスを指定します。 最大文字列長は 64byte です。 使用可能文字種は、英数記号(但し、「;」「|」「&」「`」「(」「)」「$」「<」「>」「*」「?」「{」「}」「[」「]」「!」を除く)です。

5. ID 発行に成功すると、発行された ID の「エイリアス」、「ID」、「パスワード」が表示されます。

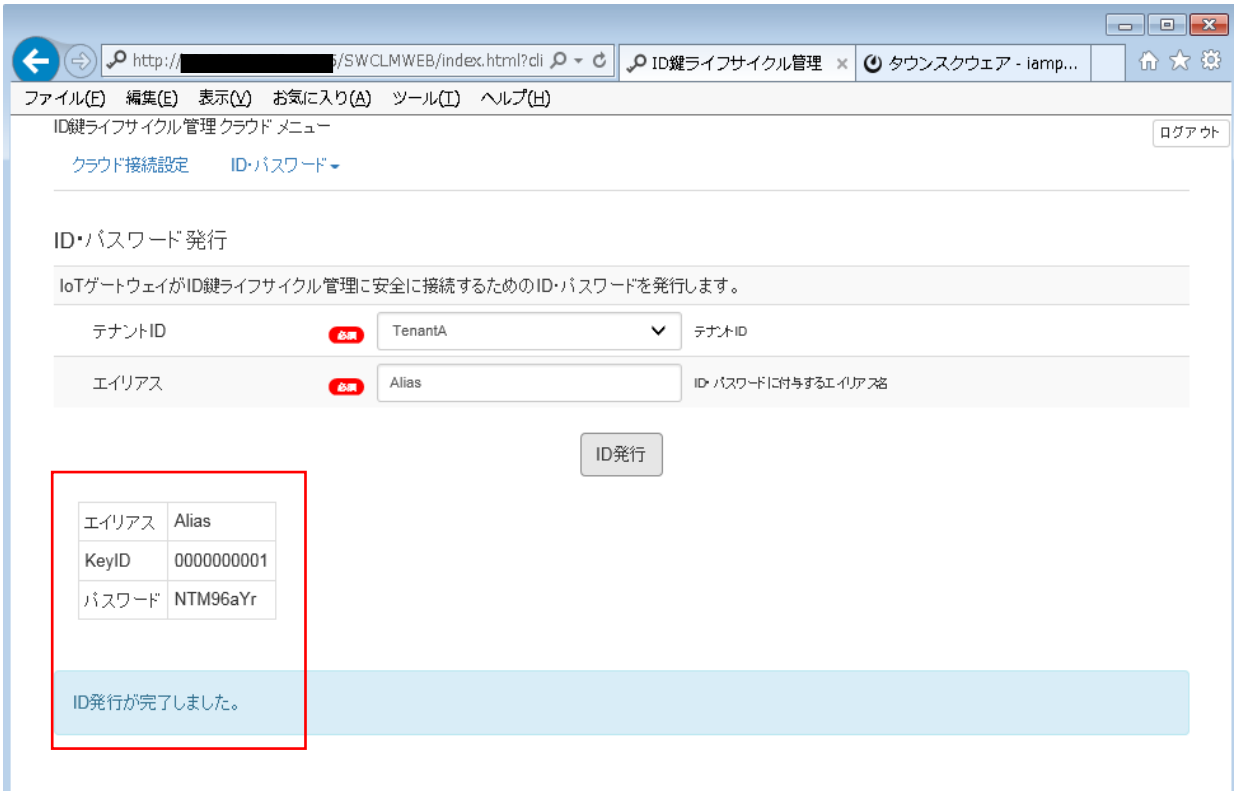


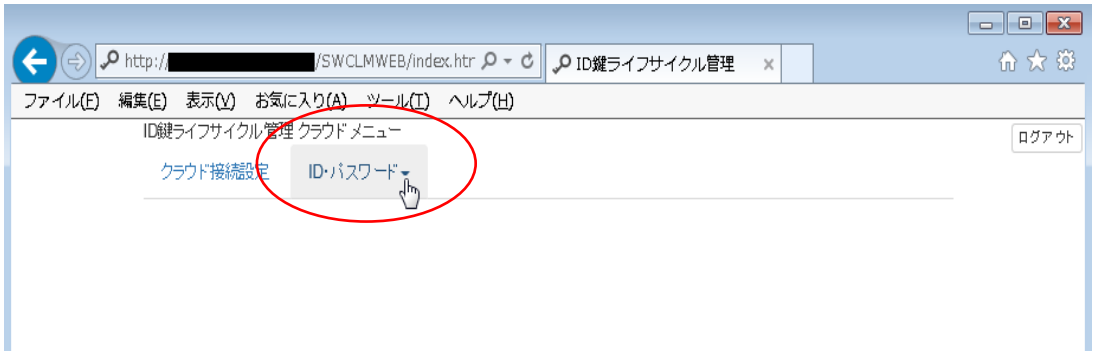
表 5-3 ID・パスワード発行 発行結果

項目	説明
エイリアス	ID・パスワードに付与するエイリアス名。
KeyID	発行された ID。
パスワード	発行された ID のパスワード。

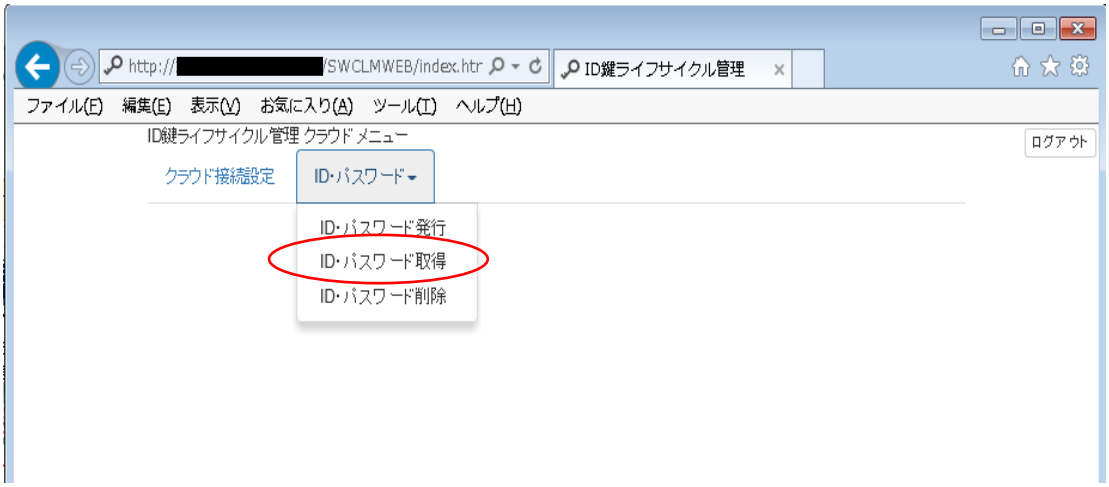
5.3.5 ID・パスワード取得

1. WebUI にログインしていない場合は、WebUI へログインします。
   ログイン手順は、5.3.2 章をご覧ください。

2. メニューの「ID・パスワード」をクリックします。



3. 「ID・パスワード」のメニュー一覧が表示されます。「ID・パスワード取得」をクリックします。



4. 「ID・パスワード取得」画面が表示されます。必要事項を入力し、「ID 取得」ボタンをクリックします。

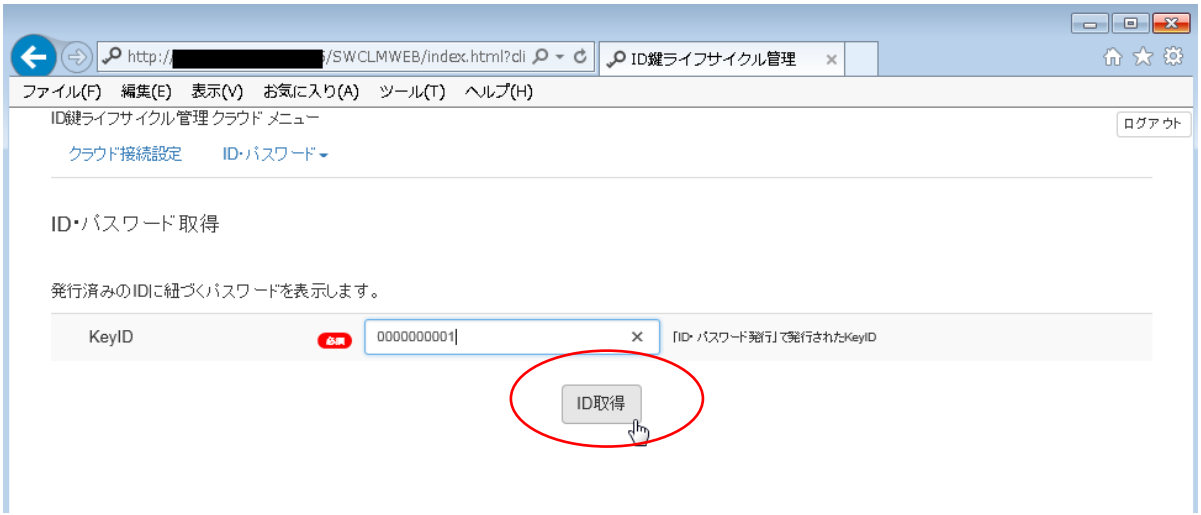


表 5-4 ID・パスワード取得 入力項目

項目	説明
KeyID	ID・パスワード発行で発行した ID を指定します。

5. ID 取得に成功すると、指定した ID とそのパスワードが表示されます。

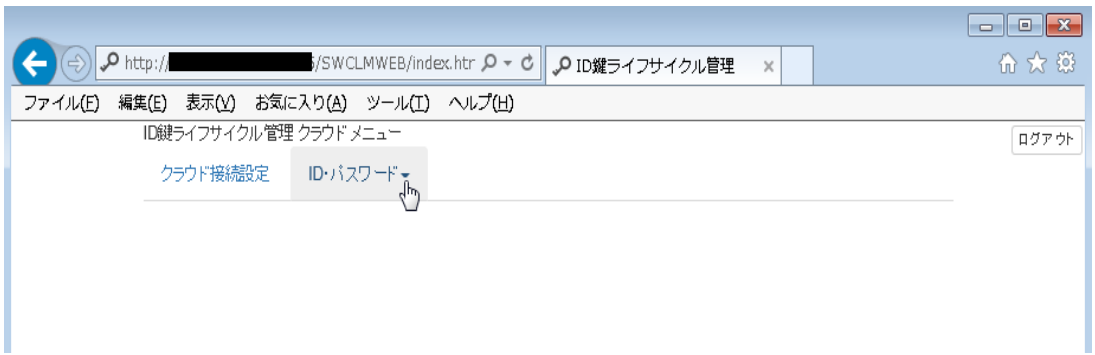
表 5-5 ID・パスワード取得 取得結果

項目	説明
KeyID	指定した ID。
パスワード	指定した ID のパスワード。

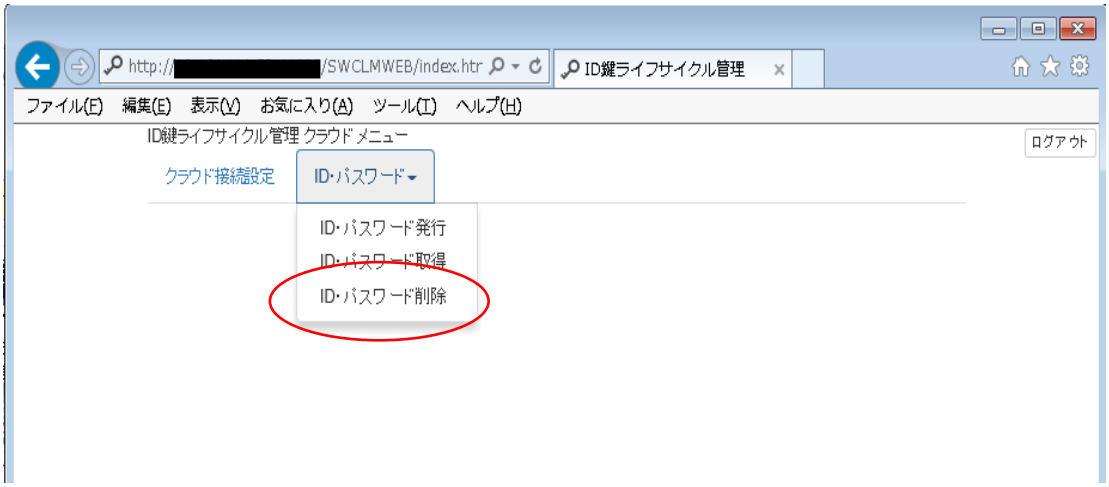
5.3.6 ID・パスワード削除

1. WebUI にログインしていない場合は、WebUI へログインします。
   ログイン手順は、5.3.2 章をご覧ください。

2. メニューの「ID・パスワード」をクリックします。



3. 「ID・パスワード」のメニュー一覧が表示されます。「ID・パスワード削除」をクリックします。



4. 「ID・パスワード削除」画面が表示されます。必要事項を入力し、「ID 削除」ボタンをクリックします。

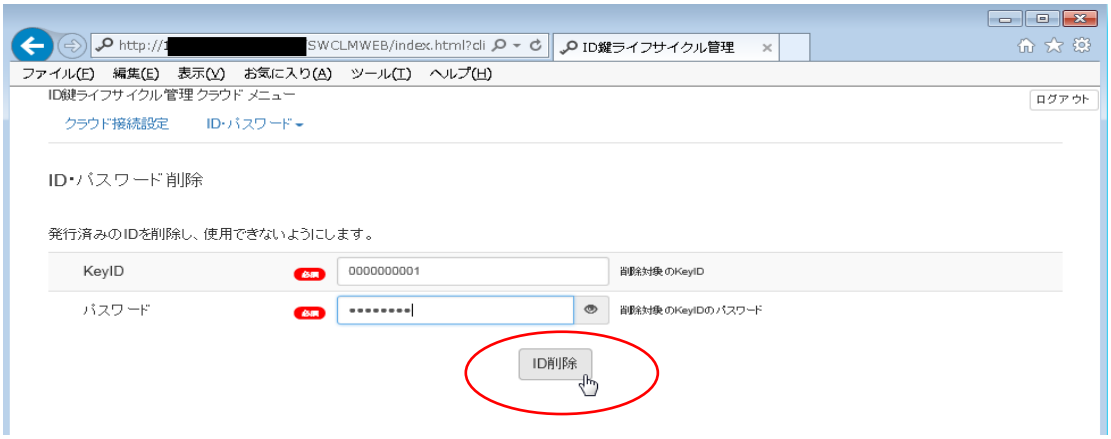


表 5-6 ID・パスワード削除 入力項目

項目	説明
KeyID	ID・パスワード発行で発行した ID を指定します。
パスワード	KeyID に指定した ID のパスワードを指定します。

5. 「ID 削除確認」ダイアログが表示されます。削除してよい場合は、「削除」ボタンをクリックします。

6. ID 削除に成功すると「ID 削除が完了しました。」のメッセージが表示されます。

6 証明書管理

CLM は、認証局(OpenSSL)と連携し、デバイスを一意に特定可能な証明書を発行できます。また、発行した、証明書を取得、更新、失効することも可能です。

発行した証明書は、SSL/TLS の相互認証・暗号化通信で使用可能です。エッジ/デバイスがクラウドとの通信に CLM で発行した証明書を使用することで、エッジ/デバイスとクラウド間の通信のセキュリティレベルを高めることが可能です。

本章では、証明書の発行・取得・更新・失効方法について説明します。

6.1 API による証明書の管理

証明書の管理を行うためのインターフェースとして、CLM では WebAPI を提供しています。WebAPI を使用することにより、HTTPS プロトコルを使用して証明書の発行・取得・更新・失効を行うことができます。

WebAPI の詳細については、別紙「WebAPI リファレンス」をご覧ください。

6.2 コマンドによる証明書の管理

証明書の管理を行うためのインターフェースとして、CLM/CLA ではコマンド(WebAPI 実行コマンド)を提供しています。WebAPI 実行コマンドを実行することで、容易に証明書の発行・取得・更新・失効を行うことができます。

WebAPI 実行コマンドの詳細については、別紙「コマンドリファレンス」をご覧ください。

6.3 WebUI による証明書の管理

証明書の管理を行うためのインターフェースとして、次の 2 つのインターフェースを提供しています。WebUI を使用することにより証明書の発行・取得・更新・失効が可能です。

- CLA 簡易 WebUI (エッジ/デバイス上で証明書を直接管理)
- CLM WebUI (エッジ/デバイス上の証明書をリモートから管理)

本章では、CLA の簡易 WebUI による証明書の発行・取得・更新・失効について説明します。証明書の発行・取得・更新・失効の各機能詳細については、別紙「WebAPI リファレンス」の「4.証明書管理 API」をご覧ください。

CLM の WebUI によるリモートからの証明書管理については、8 章をご覧ください。

6.3.1 簡易 WebUI 初回アクセス時の事前準備

簡易 WebUI へアクセスする端末・ブラウザに、以下の設定を行います。本設定は、初回アクセス前に行

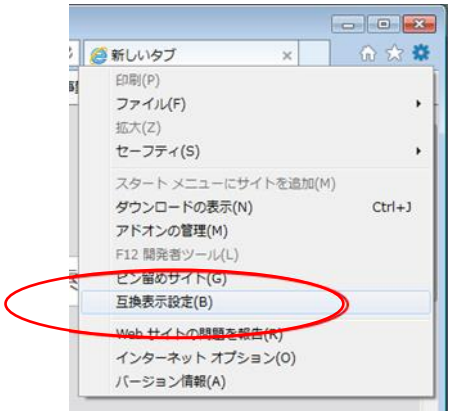
います。また、本設定を行った端末・ブラウザと異なる端末・ブラウザで簡易 WebUI へアクセスする場合は、新たに当該端末で本設定を行う必要があります。

- 簡易 WebUI との通信には https 通信を使用するので、CLM が提供するビルトイン CA 証明書を OS またはブラウザへインポートします。
  ビルトイン CA 証明書は、以下に格納しています。端末の OS またはブラウザの証明書インポート手順に従ってビルトイン CA 証明書をインポートしてください。
  [ビルトイン CA 証明書格納先]
  CLM インストールサーバ /opt/nec/pf/swcagent/bin/ built-in-ca.pem
- WebUI アクセス時に警告を出力しないように、端末の OS の hosts ファイルにエッジ/デバイスの IP アドレスとエイリアス「swclmweb」を記載し、保存します。

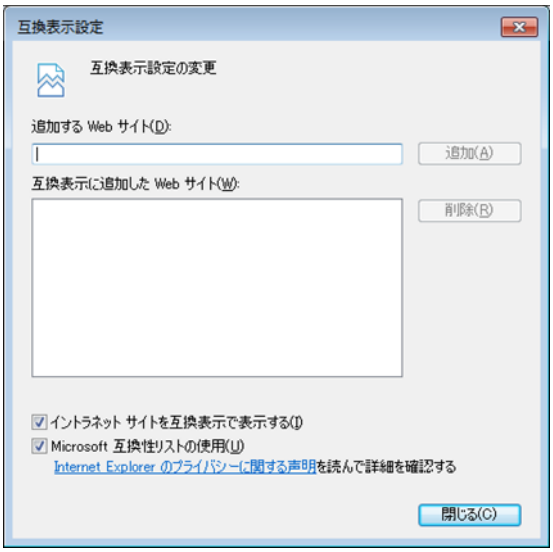
エッジ/デバイスの IP アドレス　swclmweb

- WebUI のコンテンツの表示不正が発生しないように、ブラウザのキャッシュをクリアします。使用するブラウザのキャッシュクリア手順に従い、クリアしてください。
- 使用ブラウザが Internet Explorer の場合、互換表示設定を OFF にします。

1. Internet Explorer を起動し、互換性表示設定を開きます。



2. 互換性表示設定ダイアログが表示されます。「追加する Web サイト」に簡易 Web【サンプル】のアクセス URL を入力し、「追加」ボタンをクリックします。

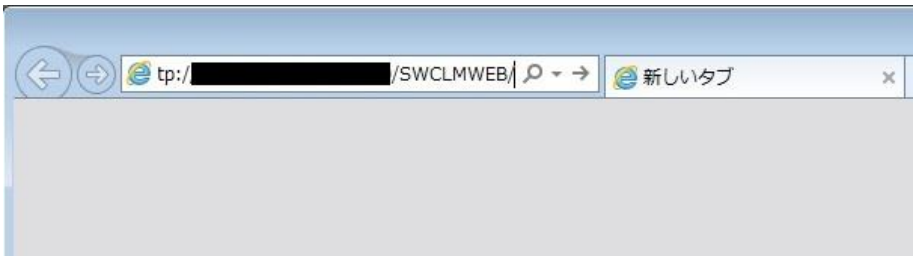


6.3.2 簡易 WebUI へのログイン・接続設定

1. ブラウザで、CLA の簡易 WebUI にアクセスします。

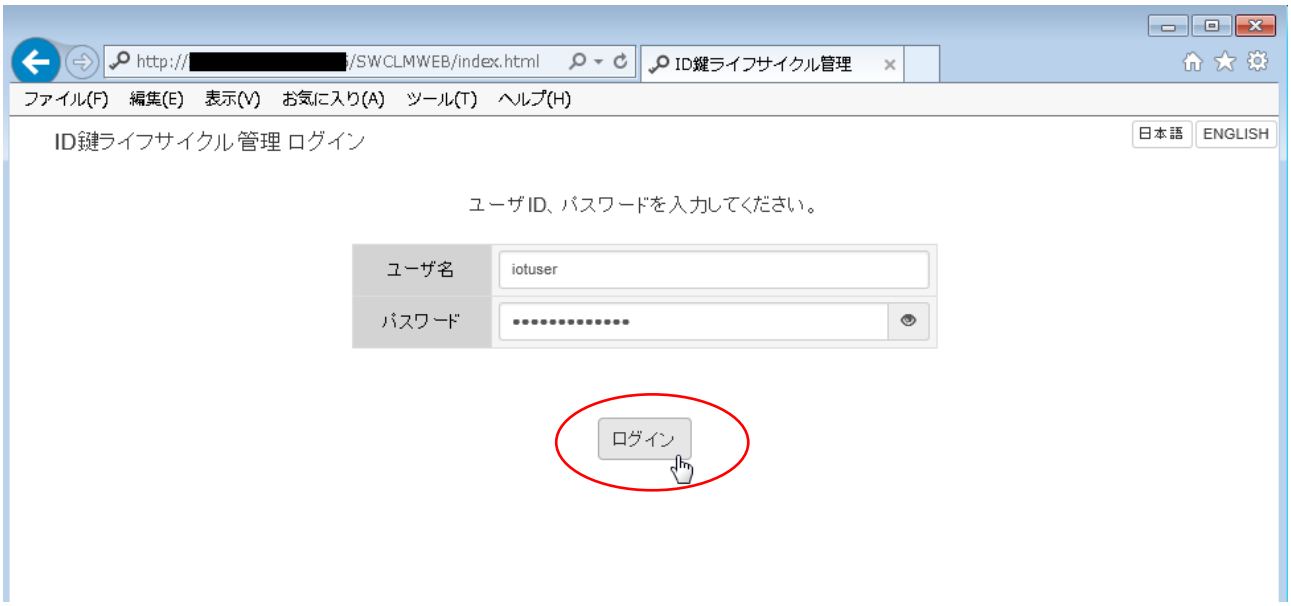
   URL: エッジゲートウェイ、Debian の場合　https://swclmweb:443/SWCLMWEB/

   RHEL/Cent の場合　https://swclmweb:8443/SWCLMWEB/



2. WebUI のログイン画面が表示されます。

   ユーザ名、パスワードを入力し、「ログイン」ボタンをクリックします。



◆ ユーザ名・パスワード

初期ユーザとして以下を用意しています。

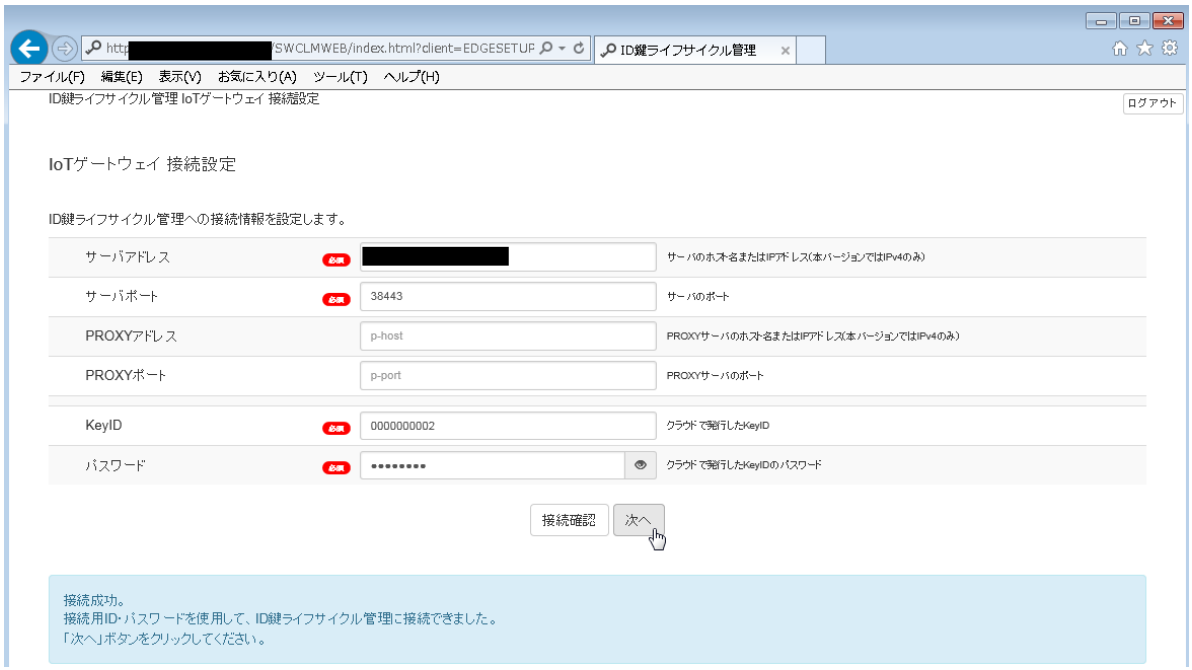
ユーザ名　：iotuser パスワード：別紙「SecureWare/Credential Lifecycle Manager セットアップカード」を ご覧ください

3. ログインに成功すると、「IoT ゲートウェイ接続設定」画面が表示されます。
   初回ログイン時は、CLM インストールサーバの情報と、4 章で発行・取得した ID・パスワードを入力し、「接続確認」ボタンをクリックします。
   なお、本画面で入力・設定した内容は、次回ログイン以降にも引き継がれます。2 回目以降のログイン時は、前回ログイン時に入力した内容が画面に自動表示されます。

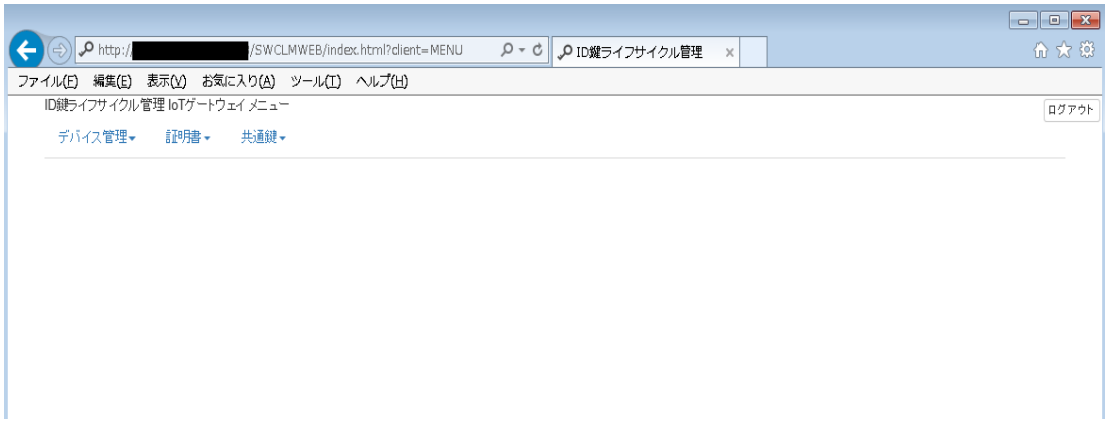
表 6-1 IoT ゲートウェイ接続設定 入力項目

項目	説明
サーバアドレス	CLM のホスト名または IP アドレスを指定します。
サーバポート	CLM の待ち受けポート番号を指定します。
PROXY アドレス	プロキシサーバの IP アドレスを指定します。 CLA から CLM までの通信時に Web プロキシサーバを経由しなければ通信できないネットワーク構成である場合は、本項目を設定してください。
PROXY ポート	プロキシサーバのポート番号を指定します。 CLA から CLM までの通信時に Web プロキシサーバを経由しなければ通信できないネットワーク構成である場合は、本項目を設定してください。
KeyID	ID・パスワード発行で発行した ID を指定します。
パスワード	KeyID に指定した ID のパスワードを指定します。

4. 「接続確認」ボタンをクリックすると、CLM への接続確認と ID・パスワード照合が行われます。照合に成功すると、CLA インストールエッジ/デバイスに ID・パスワードが設定され、「次へ」ボタンがクリック可能になります。「次へ」ボタンをクリックします
なお、CLM への接続確認と ID・パスワード照合に失敗した場合は、エラーメッセージを表示します。エラーメッセージを参考に原因を取り除き、再度「接続確認」をクリックしてください。



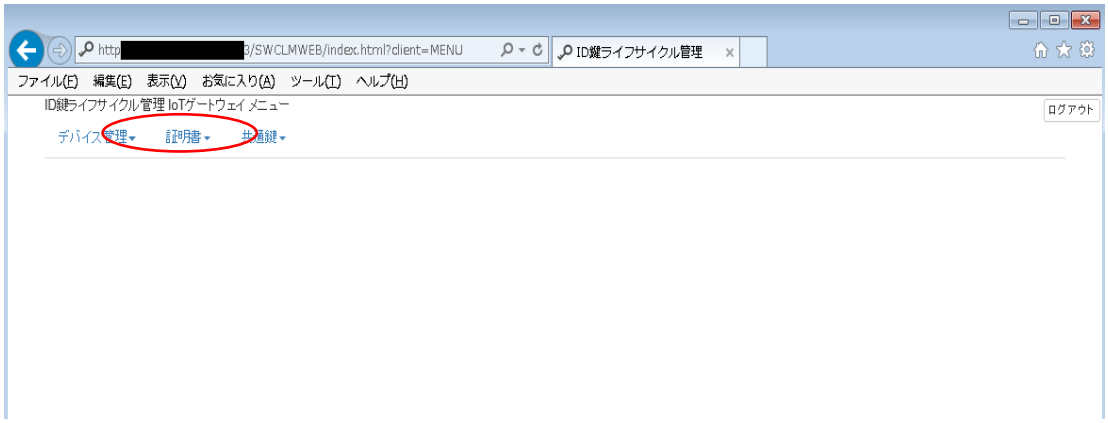
5. メニューが表示されます。



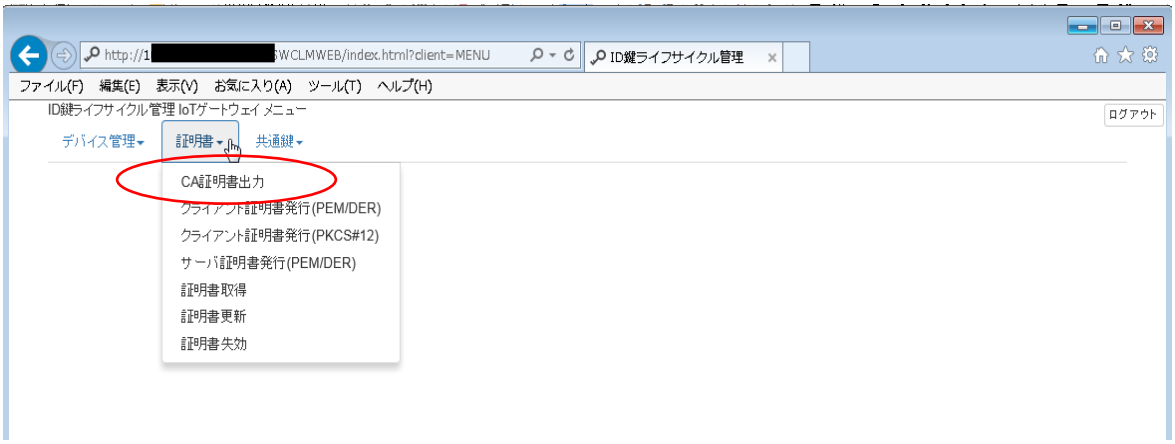
6.3.3 CA 証明書出力

1. 簡易 WebUI にログインしていない場合は、簡易 WebUI へログインします。
   ログイン手順は、6.3.2 章をご覧ください。

2. メニューの「証明書」をクリックします。



3. 「証明書」メニューが表示されます。「CA 証明書出力」をクリックします。



4. 「CA 証明書出力」画面が表示されます。必要事項を入力し、「CA 証明書出力」ボタンをクリックします。

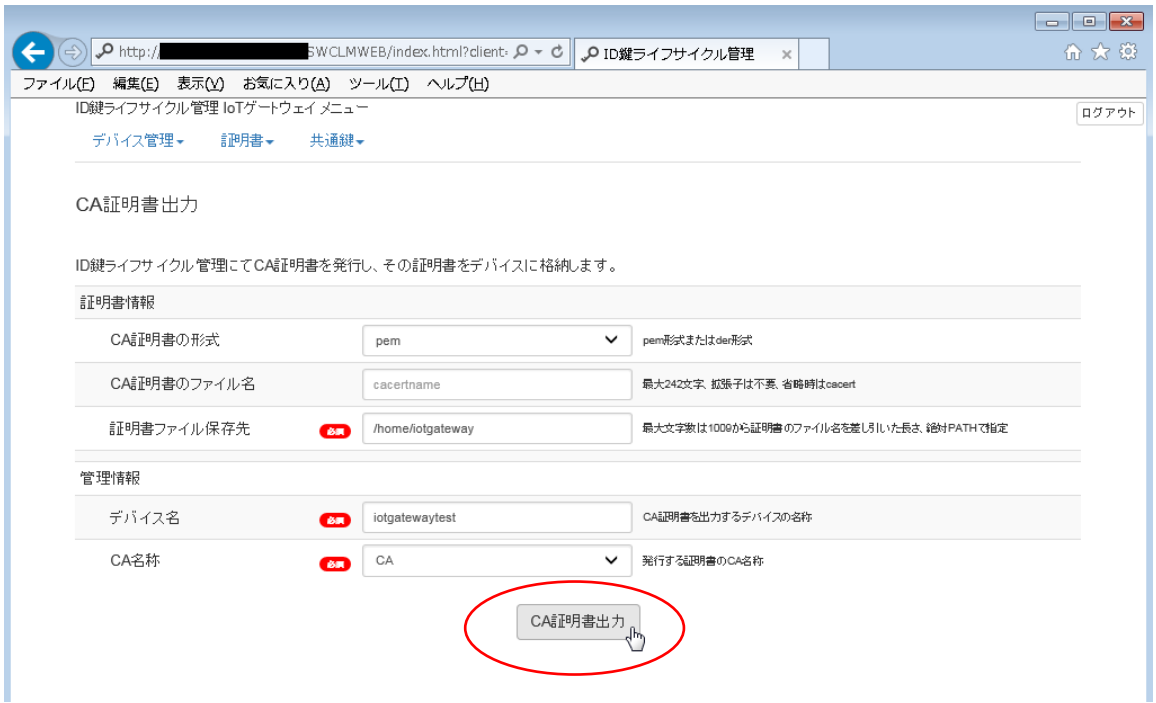


表 6-2 CA 証明書出力 入力項目

項目	説明
CA 証明書の形式	出力したい CA 証明書の種別を指定します。 以下のいずれかを選択してください。 pem: pem 形式の証明書を出力 der：der 形式の証明書を出力
CA 証明書のファイル名	出力した CA 証明書を保存するファイル名を指定します。 拡張子は不要です。既定値は「cacert」です。 最大文字列長は、242byte です。 使用可能文字種は、ASCII 0x21～0x7E(但し、「/」を除く。「`」「"」を指定する場合は、バックスラッシュでエスケープすること) です。
証明書ファイル保存先	CA 証明書を出力するディレクトリを絶対パスで指定します。 ディレクトリは、エッジデバイス上のディレクトリを指定します。 最大文字列長は、「1009 － 「CA 証明書のファイル名」に指定したファイル名の文字列長」byte です。 使用可能文字種は、ASCII 0x21～0x7E(但し、「;」「|」

	「&」「`」「(」「)」「$」「<」「>」「*」「?」「{」「}」「[」「]」「!」を除く)です。
デバイス名	どの機器(デバイス・エッジ・サーバなど)に発行するかを CLM が管理・判別するために指定します。 キッティング済みエッジ GW では、既にデバイス名は CLM にエッジ ID で登録されていますので、エッジ ID を指定します。 最大文字列長は、239byte です。 使用可能文字種は、英数記号(但し、「;」「|」「&」「`」「(」「)」「$」「<」「>」「*」「?」「{」「}」「[」「]」「!」を除く)です。
CA 名称	証明書の認証局名称を指定します。「CA」を選択します。

CA 証明書の出力に成功すると、CLA インストールエッジ/デバイス上に出力した CA 証明書の情報が表示されます。

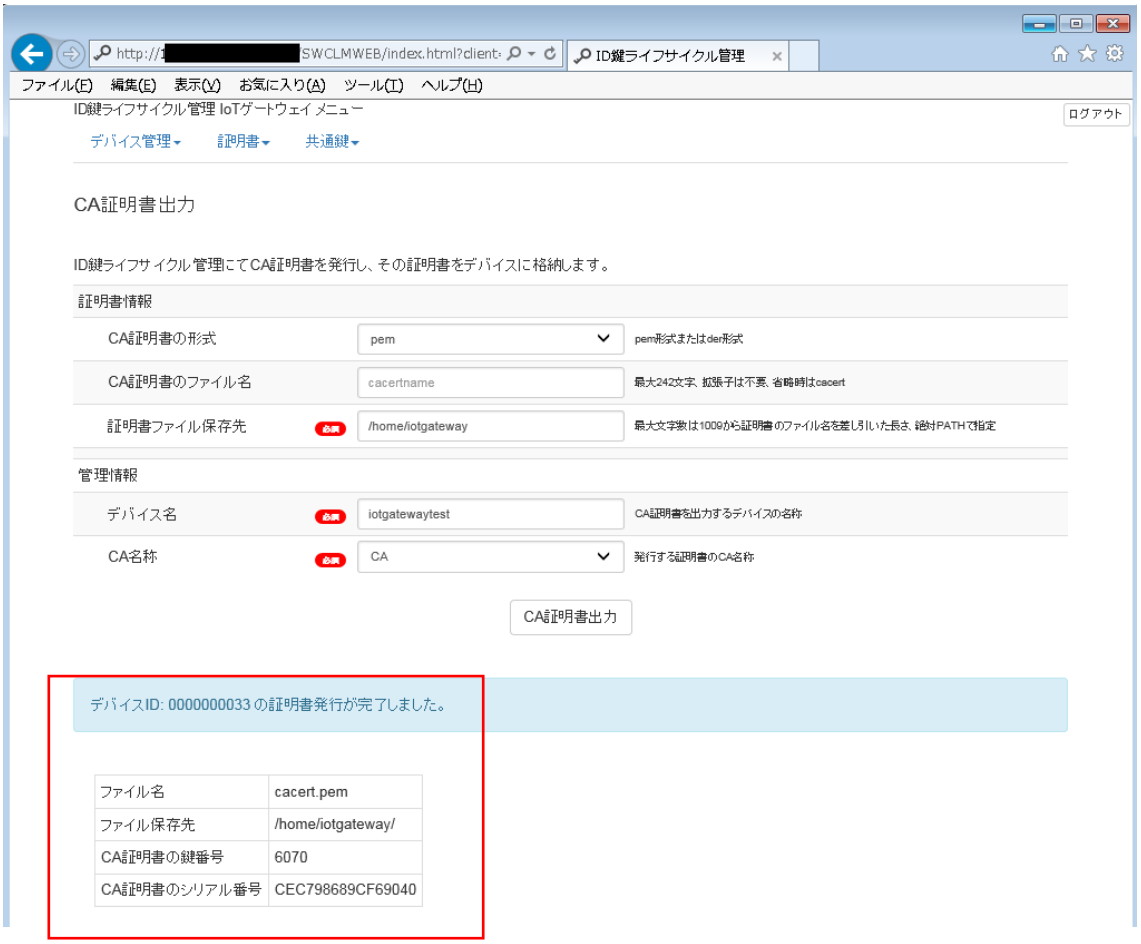


表 6-3 CA 証明書出力 出力結果

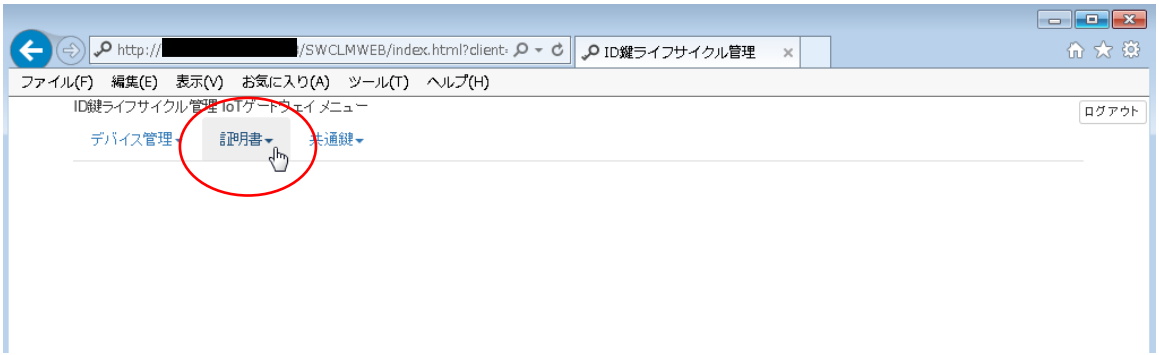
項目	説明
ファイル名	CA 証明書のファイル名。

ファイル保存先	CA 証明書を保存したディレクトリの PATH。
CA 証明書の鍵番号	証明書の鍵番号。
CA 証明書のシリアル番号	証明書のシリアル番号。

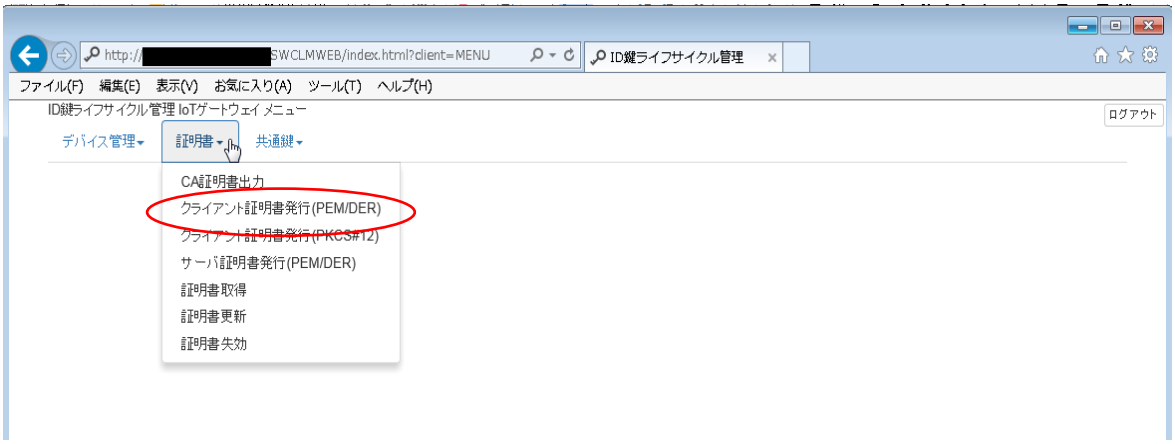
6.3.4 クライアント証明書発行(PEM/DER)

1. 簡易 WebUI にログインしていない場合は、簡易 WebUI へログインします。
   ログイン手順は、6.3.2 章をご覧ください。

2. メニューの「証明書」をクリックします。



3. 「証明書」メニューが表示されます。「クライアント証明書発行(PEM/DER)」をクリックします。



4. 「クライアント証明書発行(PEM/DER)」画面が表示されます。必要事項を入力し、「証明書発行」ボタンをクリックします

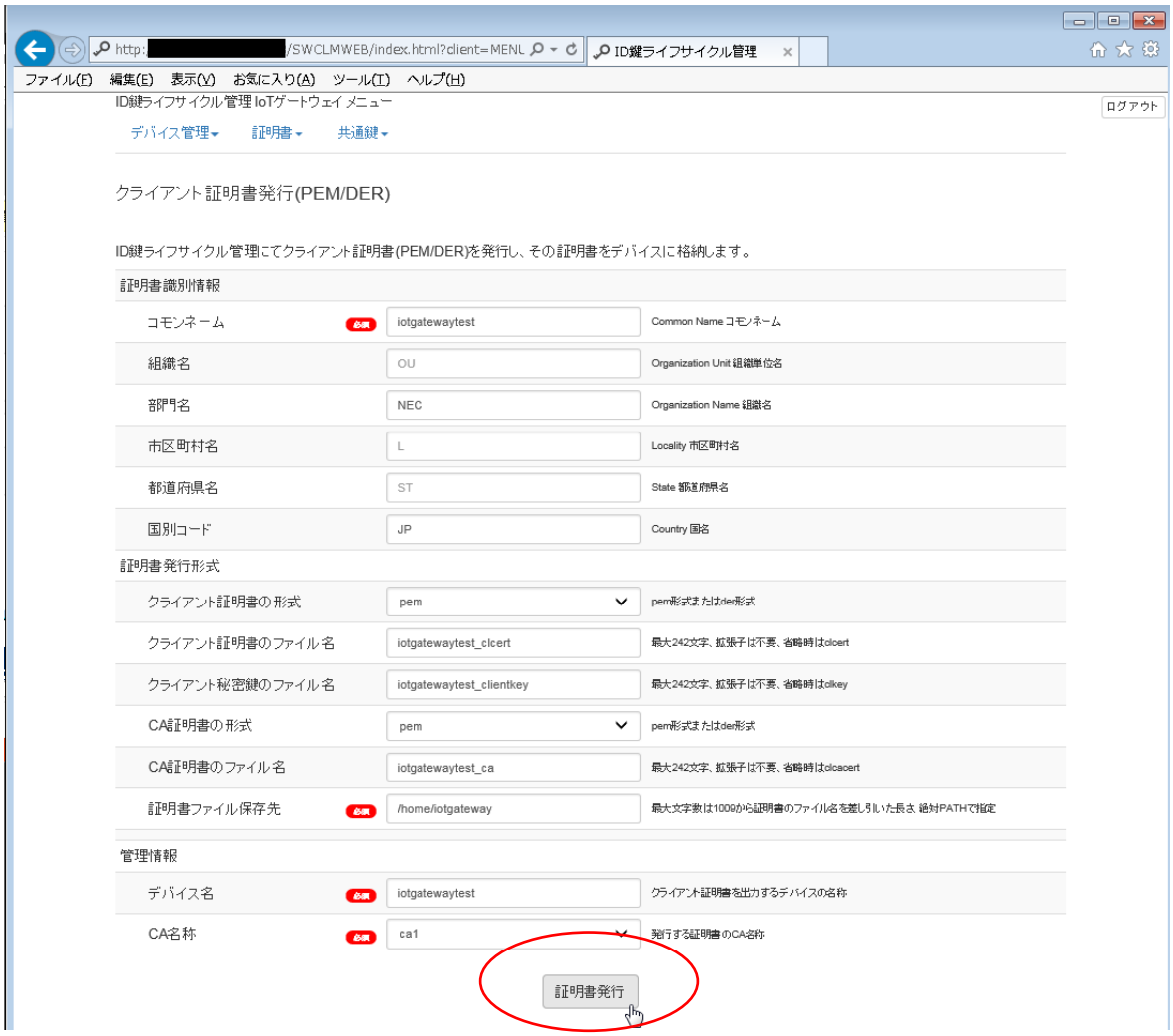


表 6-4 クライアント証明書発行(PEM/DER) 入力項目

項目	説明
コモンネーム 組織名 部門名 市町村名 都道府県名 国別コード	証明書のサブジェクトとする値を指定します。 コモンネーム以外は省略可能です。
クライアント証明書の形式	クライアント証明書の種別を指定します。 以下のいずれかを指定してください。 pem: pem 形式の証明書 der：der 形式の証明書
クライアント証明書のファイル名	クライアント証明書を保存するファイル名を指定しま

	す。 拡張子は不要です。既定値は「clcert」です。 最大文字列長は、242byte です。 使用可能文字種は、ASCII 0x21～0x7E(但し、「/」を除く。「`」「"」を指定する場合は、バックスラッシュでエスケープすること) です。
クライアント秘密鍵のファイル名	クライアント証明書の秘密鍵を保存するファイル名を指定します。 拡張子は不要です。既定値は「clkey」です。 最大文字列長は、242byte です。 使用可能文字種は、ASCII 0x21～0x7E(但し、「/」を除く。「`」「"」を指定する場合は、バックスラッシュでエスケープすること)です。
CA 証明書の形式	クライアント証明書を発行したCAのCA証明書の種別を指定します。 以下のいずれかを選択してください。 pem: pem 形式の証明書を出力 der：der 形式の証明書を出力
CA 証明書のファイル名	クライアント証明書を発行したCAのCA証明書を保存するファイル名を指定します。 拡張子は不要です。既定値は「cacert」です。 最大文字列長は、242byte です。 使用可能文字種は、ASCII 0x21～0x7E(但し、「/」を除く。「`」「"」を指定する場合は、バックスラッシュでエスケープすること) です。
証明書ファイル保存先	クライアント証明書を出力するディレクトリを絶対パスで指定します。 ディレクトリは、エッジデバイス上のディレクトリを指定します。 最大文字列長は、「1009 – 「クライアント証明書のファイル名」、「クライアント秘密鍵のファイル名」、「CA 証明書のファイル名」のファイル名の最大文字列長」byte です。 使用可能文字種は、ASCII 0x21～0x7E(但し、「;」「|」「&」「`」「(」「)」「$」「<」「>」「*」「?」「{」「}」「[」「]」「!」を除く)です。
デバイス名	どの機器(デバイス・エッジ・サーバなど)に発行するか

	を CLM が管理・判別するために指定します。 キッティング済みエッジ GW では、既にデバイス名は CLM にエッジ ID で登録されていますので、エッジ ID を指定します。 最大文字列長は、239byte です。 使用可能文字種は、英数記号(但し、「;」「|」「&」「`」「(」「)」「$」「<」「>」「*」「?」「{」「}」「[」「]」「!」を除く)です。
CA 名称	証明書の認証局名称を指定します。「ca1」を選択します。

5. クライアント証明書の発行に成功すると、CLA インストールエッジ/デバイス上に発行・保存した証明書の情報が表示されます。

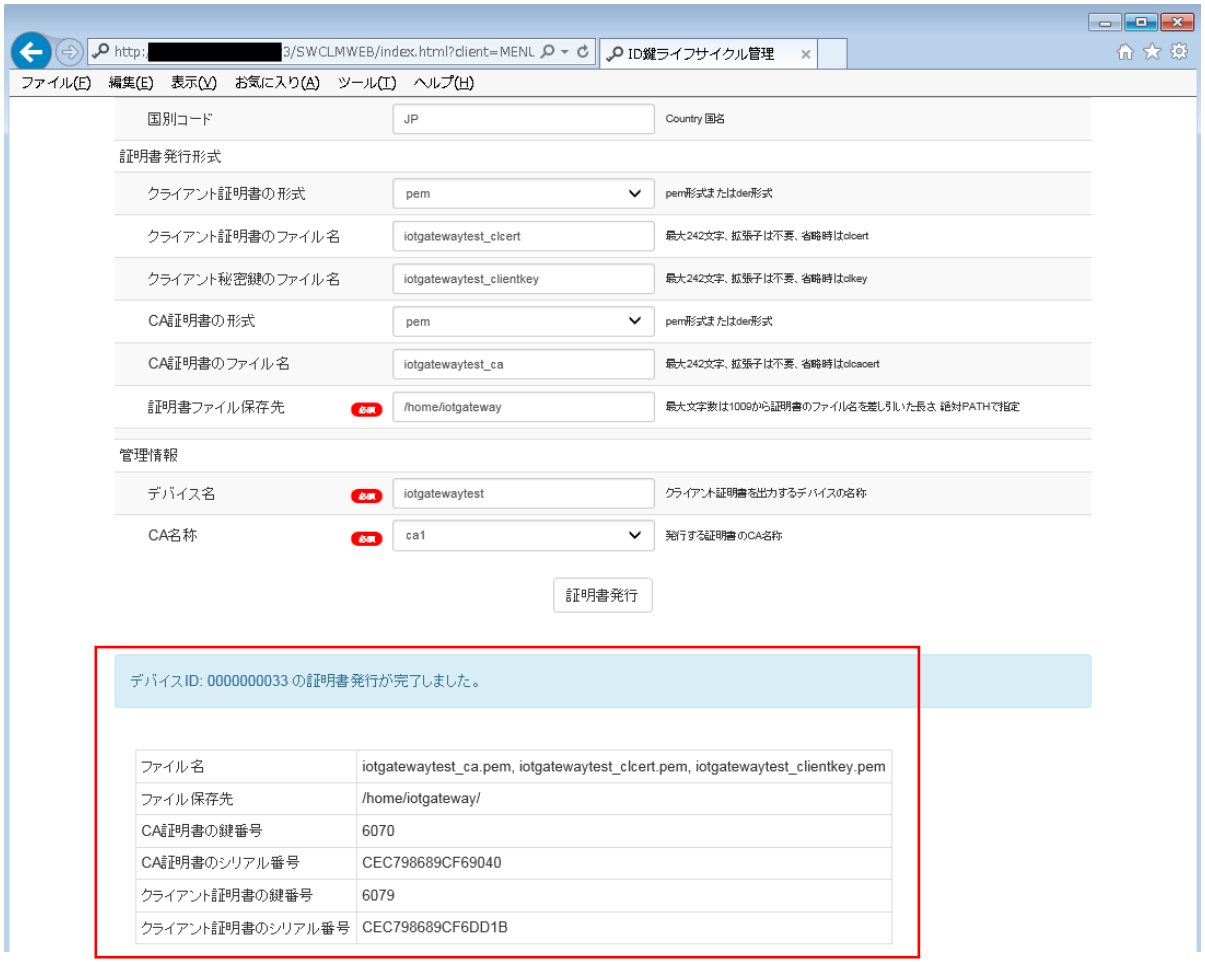


表 6-5 クライアント証明書発行(PEM/DER) 発行結果

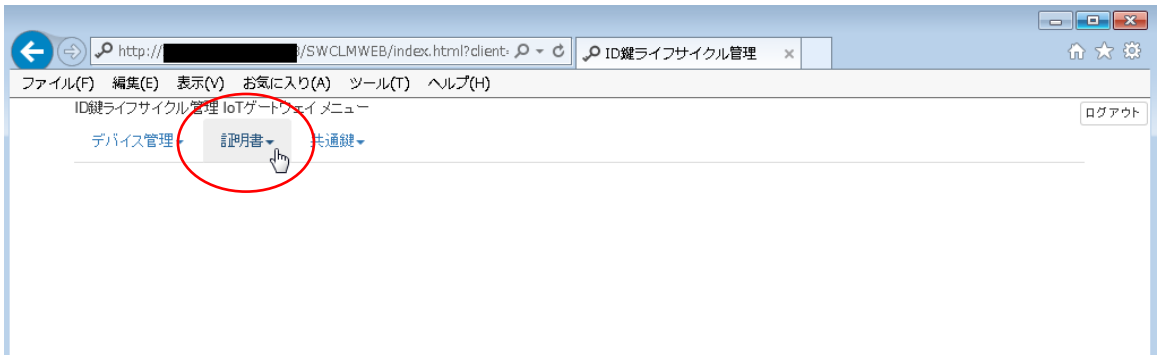
項目	説明
ファイル名	CLM から発行・出力された証明書のファイル名。 順に、クライアント証明書を発行した CA の CA 証明書のファイル名、クライアント証明書のファイル名、クライアント証明

	書の秘密鍵のファイル名。
ファイル保存先	証明書を保存したディレクトリの PATH。
CA 証明書の鍵番号	CA 証明書の鍵番号。
CA 証明書のシリアル番号	CA 証明書のシリアル番号。
クライアント証明書の鍵番号	クライアント証明書の鍵番号。
クライアント証明書のシリアル番号	クライアント証明書のシリアル番号。

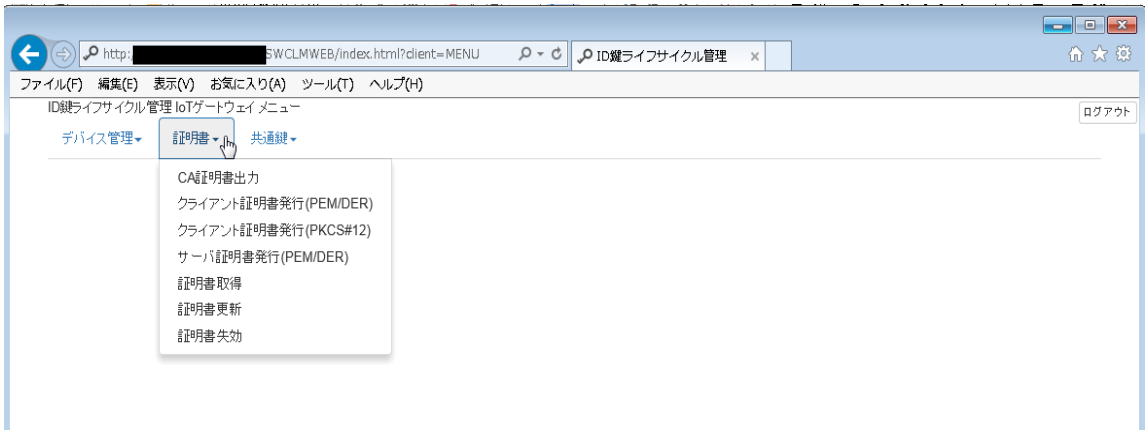
6.3.5 クライアント証明書発行(PKCS#12)

1. 簡易 WebUI にログインしていない場合は、簡易 WebUI へログインします。
   ログイン手順は、6.3.2 章をご覧ください。

2. メニューの「証明書」をクリックします。



3. 「証明書」メニューが表示されます。「クライアント証明書発行(PKCS#12)」をクリックします。



4. 「クライアント証明書発行(PKCS#12)」画面が表示されます。必要事項を入力し、「証明書発行」ボタンをクリックします

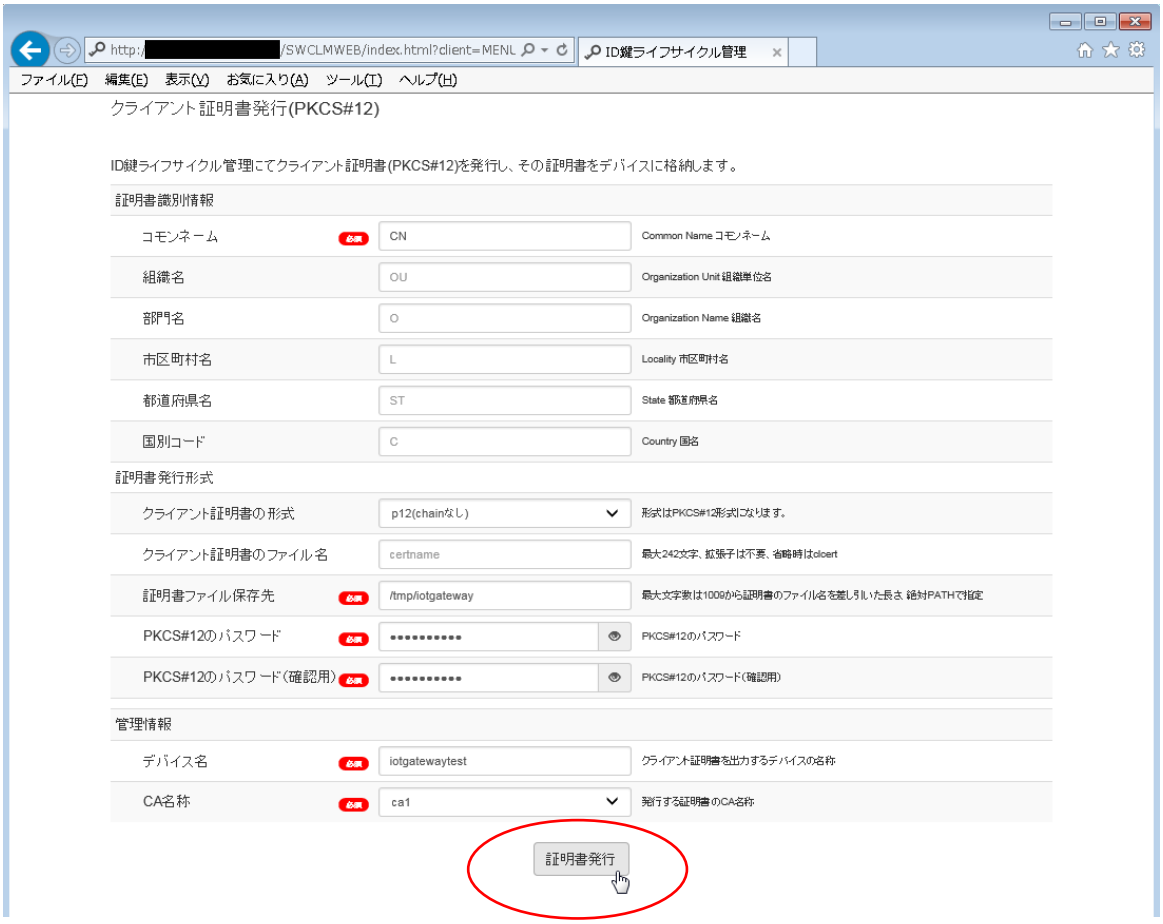


表 6-6 クライアント証明書発行(PKCS#12) 入力項目

項目	説明
コモンネーム 組織名 部門名 市町村名 都道府県名 国別コード	証明書のサブジェクトとする値を指定します。 コモンネーム以外は省略可能です。
クライアント証明書の形式	クライアント証明書の種別を指定します。 「p12(chain なし)」を選択します。
クライアント証明書のファイル名	クライアント証明書を保存するファイル名を指定します。 拡張子は不要です。既定値は「clcert」です。 最大文字列長は、242byte です。 使用可能文字種は、ASCII 0x21～0x7E(但し、「/」を除く。「`」「"」を指定する場合は、バックスラッシュでエ

	スケープすること) です。
証明書ファイル保存先	クライアント証明書を出力するディレクトリを絶対パスで指定します。 ディレクトリは、エッジデバイス上のディレクトリを指定します。 最大文字列長は、「1009 – 「クライアント証明書のファイル名」に指定したファイル名の文字列長」byte です。 使用可能文字種は、ASCII 0x21～0x7E(但し、「;」「|」「&」「`」「(」「)」「$」「<」「>」「*」「?」「{」「}」「[」「]」「!」を除く)です。
PKCS#12 のパスワード PKCS#12 のパスワード(確認用)	クライアント証明書の秘密鍵のパスフレーズを指定します。PKCS#12 形式に変換する際のパスフレーズにも利用します。 最大文字列長は、50byte です。 使用可能文字種は、ASCII 0x21～0x7E(但し、「;」「|」「&」「`」「(」「)」「$」「<」「>」「*」「?」「{」「}」「[」「]」「!」を除く)です。
デバイス名	どの機器(デバイス・エッジ・サーバなど)に発行するかを CLM が管理・判別するために指定します。 キッティング済みエッジ GW では、既にデバイス名は CLM にエッジ ID で登録されていますので、エッジ ID を指定します。 最大文字列長は、239byte です。 使用可能文字種は、英数記号(但し、「;」「|」「&」「`」「(」「)」「$」「<」「>」「*」「?」「{」「}」「[」「]」「!」を除く)です。
CA 名称	証明書の認証局名称を指定します。「ca1」を選択します。

5. 証明書の発行に成功すると、CLA インストールエッジ/デバイス上に発行・保存した証明書の情報が表示されます。

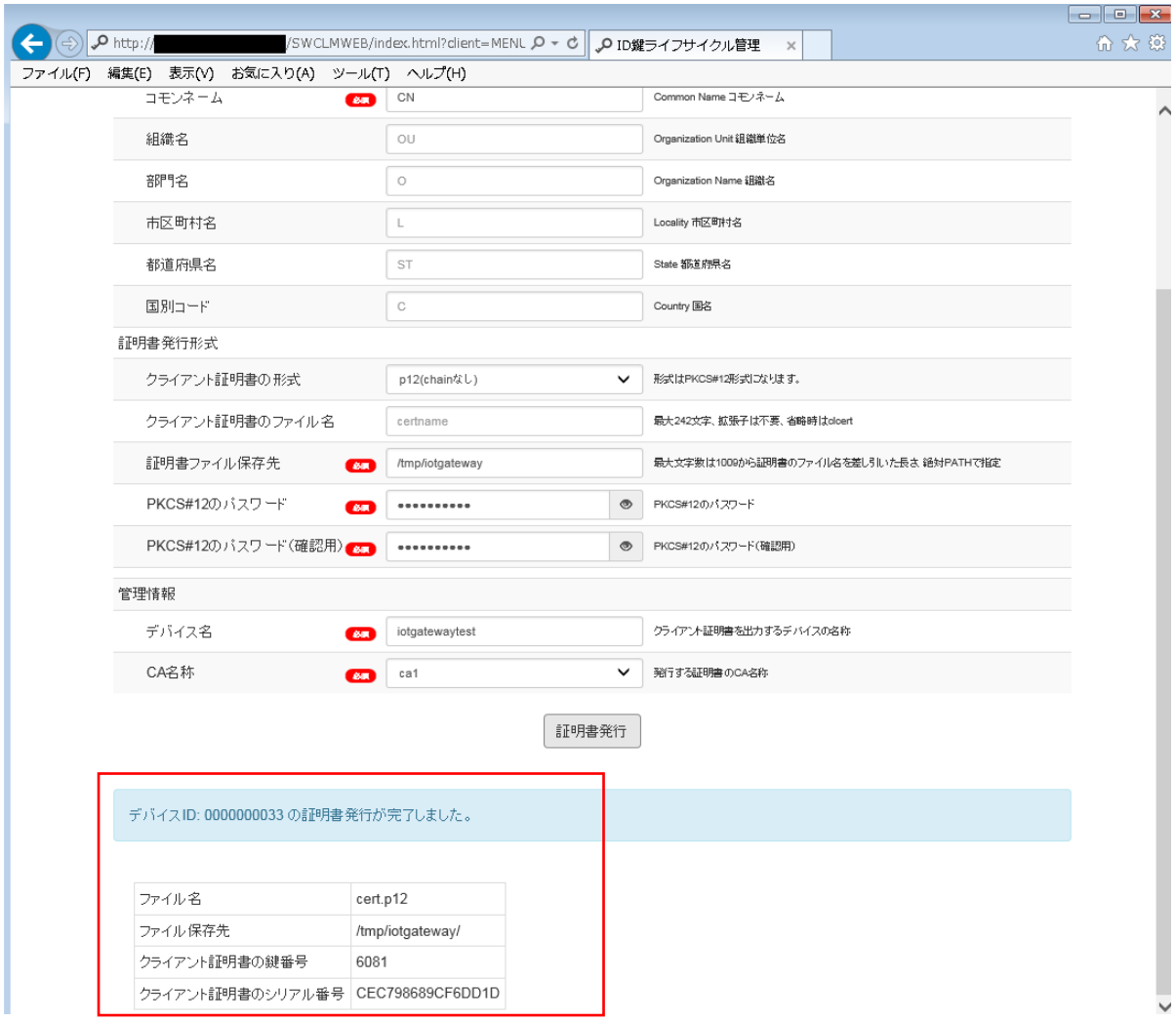


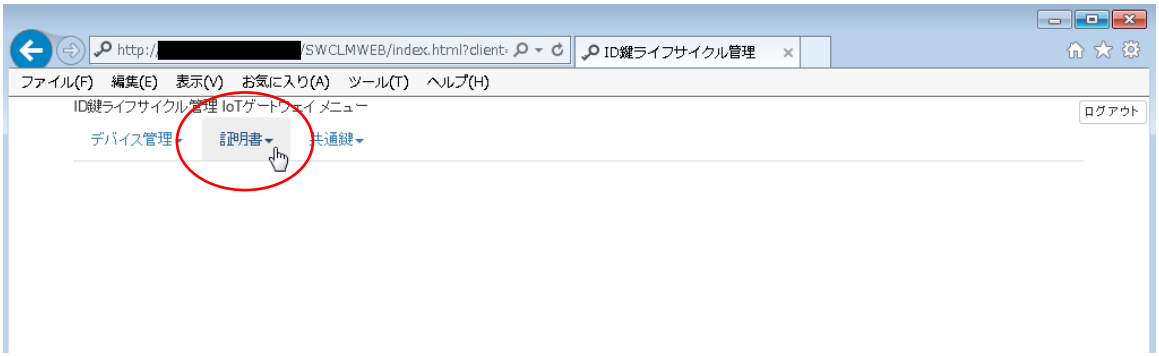
表 6-7 クライアント証明書発行(PKCS#12) 発行結果

項目	説明
ファイル名	CLM から発行・出力された証明書のファイル名。
ファイル保存先	証明書を保存したディレクトリの PATH。
クライアント証明書の鍵番号	クライアント証明書の鍵番号。
クライアント証明書のシリアル番号	クライアント証明書のシリアル番号。

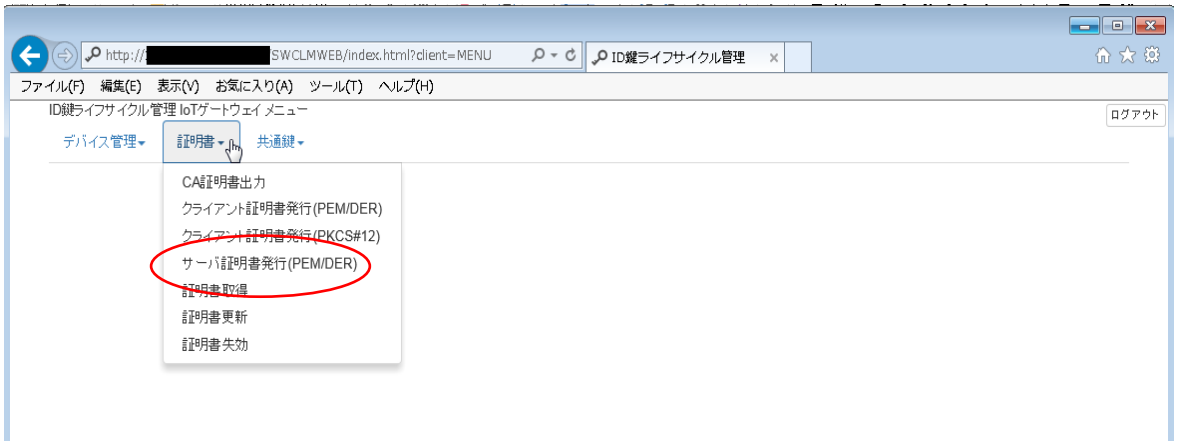
6.3.6 サーバ証明書発行(PEM/DER)

1. 簡易 WebUI にログインしていない場合は、簡易 WebUI へログインします。
   ログイン手順は、6.3.2 章をご覧ください。

1. メニューの「証明書」をクリックします。



2. 「証明書」メニューが表示されます。「サーバ証明書発行(PEM/DER)」をクリックします。



3. 「サーバ証明書発行(PEM/DER)」画面が表示されます。必要事項を入力し、「証明書発行」ボタンをクリックします

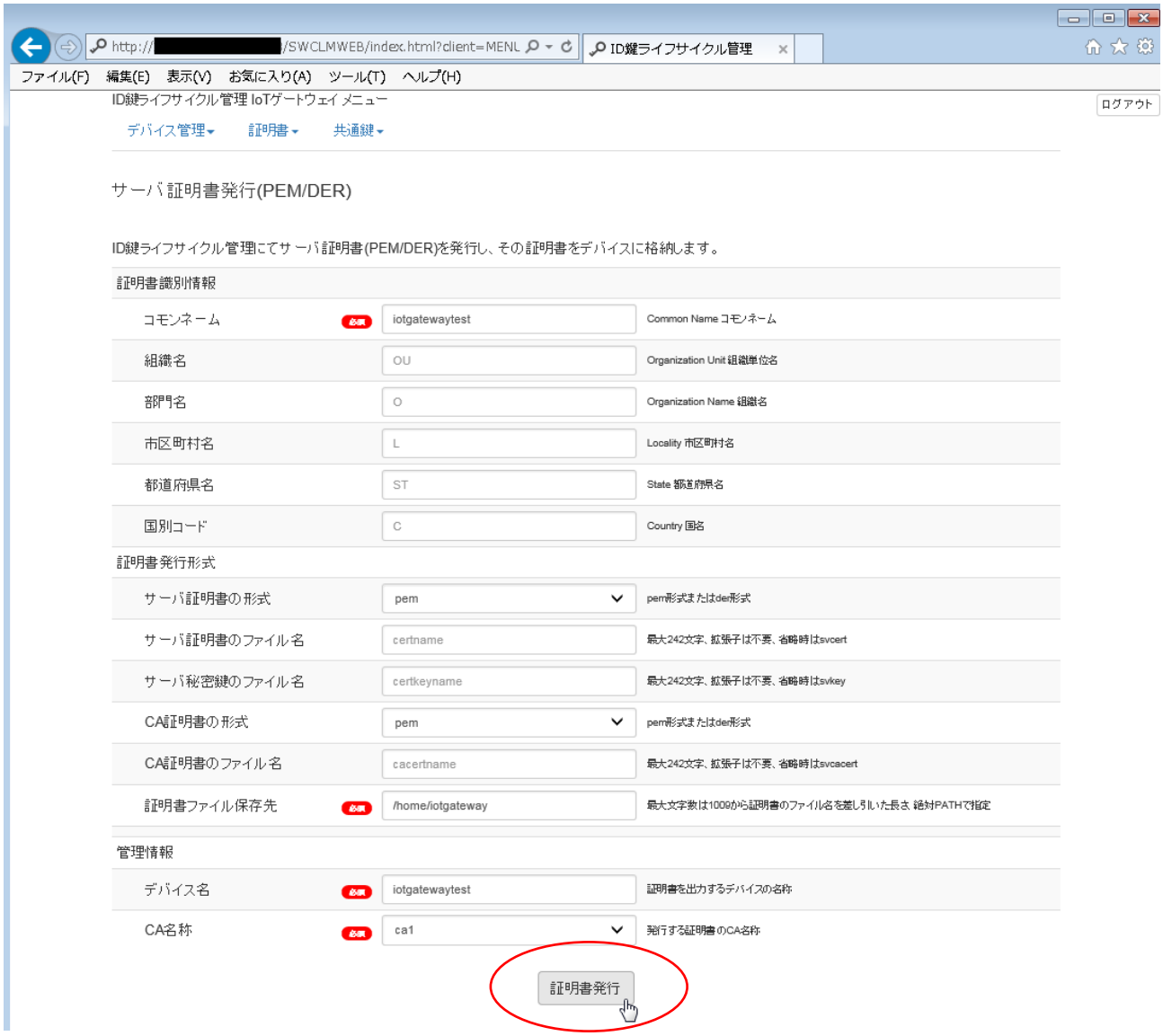


表 6-8 サーバ証明書発行(PEM/DER) 入力項目

項目	説明
コモンネーム 組織名 部門名 市町村名 都道府県名 国別コード	証明書のサブジェクトとする値を指定します。 コモンネーム以外は省略可能です。
サーバ証明書の形式	サーバ証明書の種別を指定します。 以下のいずれかを指定してください。 pem: pem 形式の証明書 der：der 形式の証明書
サーバ証明書のファイル名	サーバ証明書を保存するファイル名を指定します。 拡張子は不要です。既定値は「svcert」です。 最大文字列長は、242byte です。

	使用可能文字種は、ASCII 0x21～0x7E(但し、「/」を除く。「`」「"」を指定する場合は、バックスラッシュでエスケープすること) です。
サーバ秘密鍵のファイル名	サーバ証明書の秘密鍵を保存するファイル名を指定します。 拡張子は不要です。既定値は「svkey」です。 最大文字列長は、242byte です。 使用可能文字種は、ASCII 0x21～0x7E(但し、「/」を除く。「`」「"」を指定する場合は、バックスラッシュでエスケープすること)です。
CA 証明書の形式	クライアント証明書を発行したCAのCA証明書の種別を指定します。 以下のいずれかを選択してください。 pem: pem 形式の証明書を出力 der：der 形式の証明書を出力
CA 証明書のファイル名	クライアント証明書を発行したCAのCA証明書を保存するファイル名を指定します。 拡張子は不要です。既定値は「cacert」です。 最大文字列長は、242byte です。 使用可能文字種は、ASCII 0x21～0x7E(但し、「/」を除く。「`」「"」を指定する場合は、バックスラッシュでエスケープすること) です。
証明書ファイル保存先	サーバ証明書を出力するディレクトリを絶対パスで指定します。 ディレクトリは、エッジデバイス上のディレクトリを指定します。 最大文字列長は、「1009 – 「サーバ証明書のファイル名」、「サーバ秘密鍵のファイル名」、「CA 証明書のファイル名」に指定したファイル名の最大文字列長」byte です。 使用可能文字種は、ASCII 0x21～0x7E(但し、「;」「|」「&」「`」「(」「)」「$」「<」「>」「*」「?」「{」「}」「[」「]」「!」を除く)です。
デバイス名	どの機器(デバイス・エッジ・サーバなど)に発行するかを CLM が管理・判別するために指定します。 キッティング済みエッジ GW では、既にデバイス名は CLM にエッジ ID で登録されていますので、エッジ ID を指定します。

	最大文字列長は、239byte です。 使用可能文字種は、英数記号(但し、「;」「|」「&」「`」「(」「)」「$」「<」「>」「*」「?」「{」「}」「[」「]」「!」を除く)です。
CA 名称	証明書の認証局名称を指定します。「ca1」を選択します。

4. 証明書の発行に成功すると、CLA インストールエッジ/デバイス上に発行・保存した証明書の情報が表示されます。

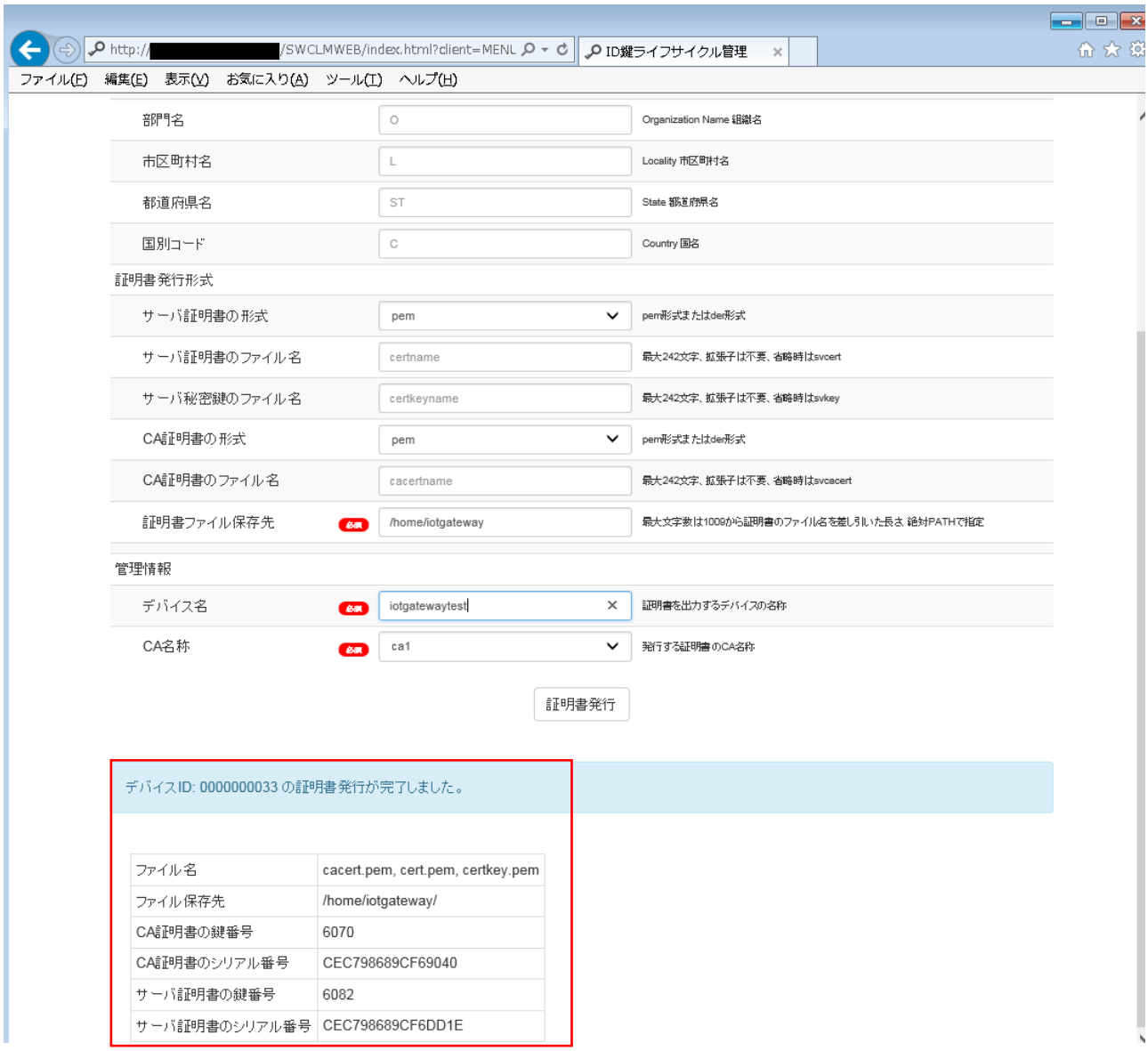


表 6-9 サーバ証明書発行(PEM/DER) 発行結果

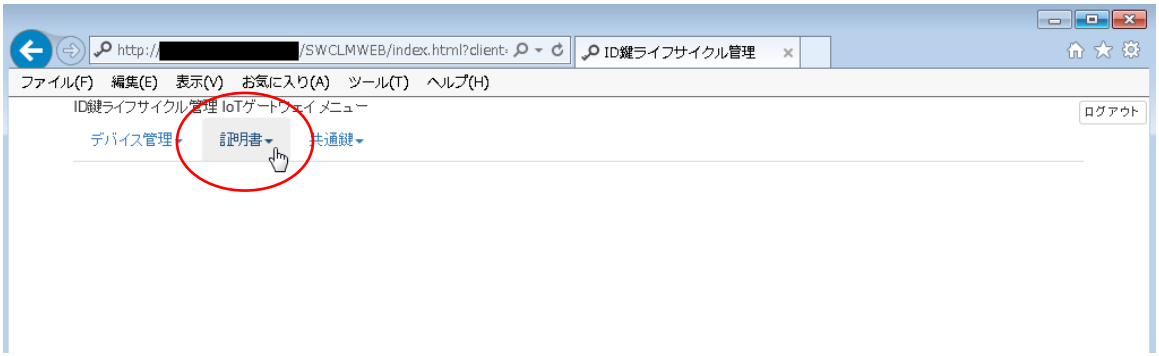
項目	説明
ファイル名	CLM から発行・出力された証明書のファイル名。 順に、サーバ証明書を発行した CA の CA 証明書のファイル名、サーバ

	証明書のファイル名、サーバ証明書の秘密鍵のファイル名。
ファイル保存先	証明書を保存したディレクトリの PATH。
CA 証明書の鍵番号	CA 証明書の鍵番号。
CA 証明書のシリアル番号	CA 証明書のシリアル番号。
サーバ証明書の鍵番号	サーバ証明書の鍵番号。
サーバ証明書のシリアル番号	サーバ証明書のシリアル番号。

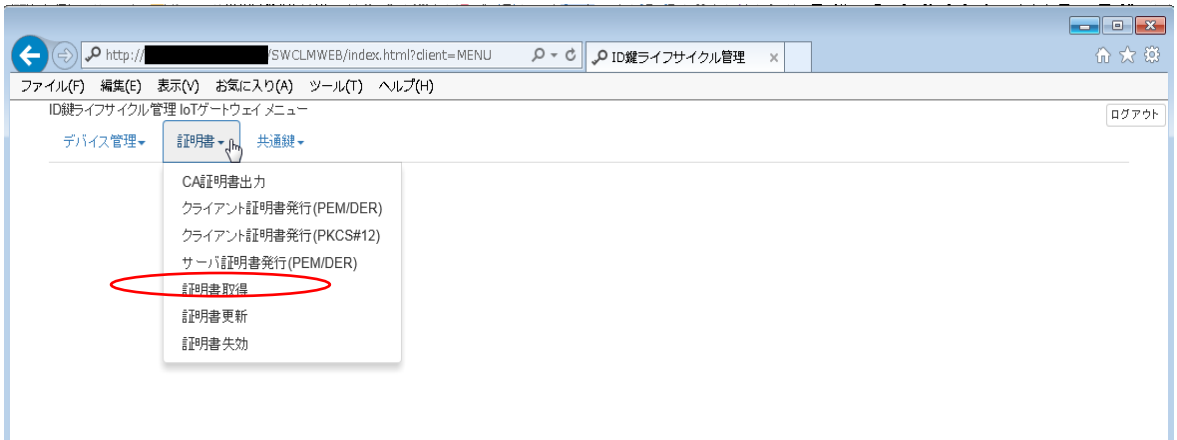
6.3.7 証明書取得

1. 簡易 WebUI にログインしていない場合は、簡易 WebUI へログインします。
   ログイン手順は、6.3.2 章をご覧ください。

2. メニューの「証明書」をクリックします。



3. 「証明書」メニューが表示されます。「証明書取得」をクリックします。



4. 「証明書取得」画面が表示されます。必要事項を入力し、「証明書取得」ボタンをクリックします

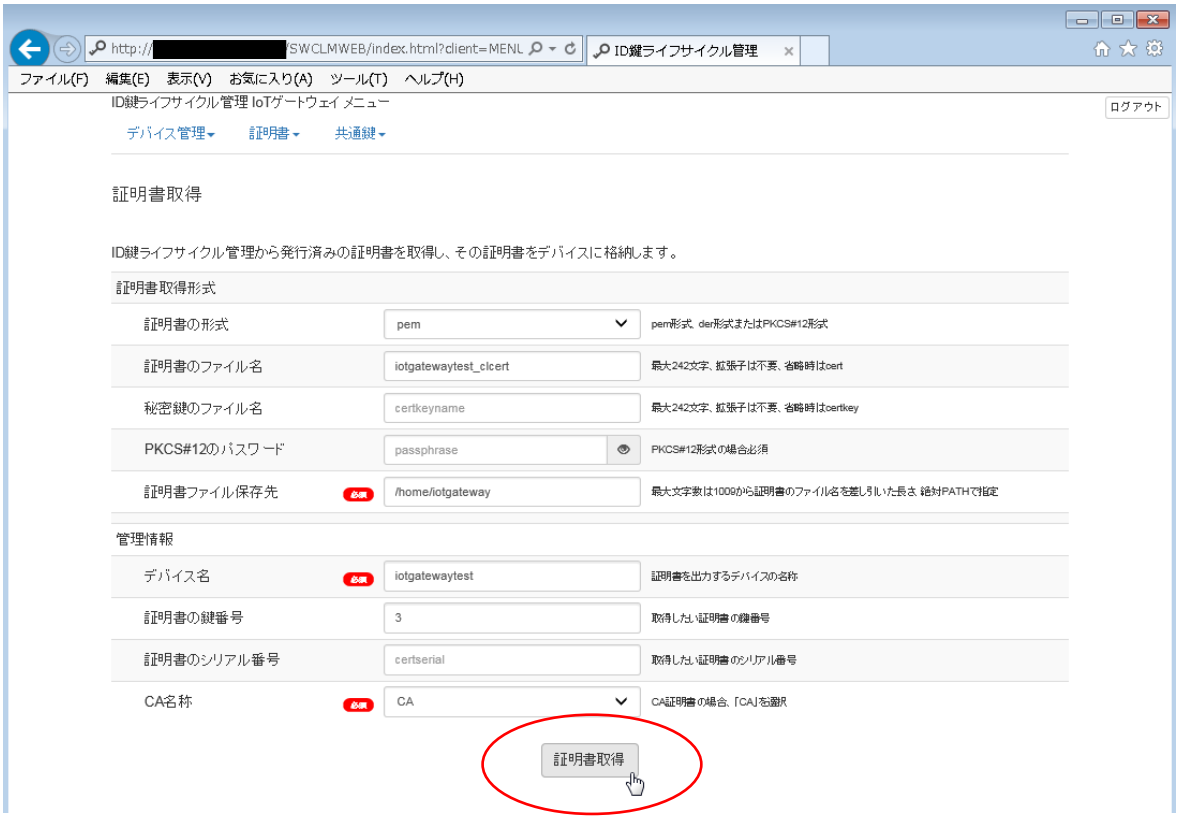


表 6-10 証明書取得 入力項目

項目	説明
証明書の形式	証明書の種別を指定します。 以下のいずれかを指定してください。 pem: pem 形式の証明書 der：der 形式の証明書 p12：PKCS#12 形式の証明書(chain なし)
証明書のファイル名	証明書を保存するファイル名を指定します。 拡張子は不要です。既定値は「cert」です。 最大文字列長は、242byte です。 使用可能文字種は、ASCII 0x21～0x7E(但し、「/」を除く。「`」「"」を指定する場合は、バックスラッシュでエスケープすること) です。
秘密鍵のファイル名	証明書の秘密鍵を保存するファイル名を指定します。 拡張子は不要です。既定値は「certkey」です。 最大文字列長は、242byte です。 使用可能文字種は、ASCII 0x21～0x7E(但し、「/」を除く。「`」「"」を指定する場合は、バックスラッシュでエスケープすること)です。

PKCS#12 のパスワード	証明書の秘密鍵のパスフレーズを指定します。PKCS#12 形式に変換する際のパスフレーズにも利用します。 最大文字列長は、50byte です。 使用可能文字種は、ASCII 0x21～0x7E(但し、「;」「|」「&」「`」「(」「)」「$」「<」「>」「*」「?」「{」「}」「[」「]」「!」を除く)です。
証明書ファイル保存先	証明書を出力するディレクトリを絶対パスで指定します。 ディレクトリは、エッジデバイス上のディレクトリを指定します。 最大文字列長は、「1009 – 「証明書のファイル名」、「秘密鍵のファイル名」のファイル名の最大文字列長」byte です。 使用可能文字種は、ASCII 0x21～0x7E(但し、「;」「|」「&」「`」「(」「)」「$」「<」「>」「*」「?」「{」「}」「[」「]」「!」を除く)です。
デバイス名	どの機器(デバイス・エッジ・サーバなど)から取得されるかを CLM が判別するために指定します。 キッティング済みエッジ GW では、既にデバイス名は CLM にエッジ ID で登録されていますので、エッジ ID を指定します。 最大文字列長は、239byte です。 使用可能文字種は、英数記号(但し、「;」「|」「&」「`」「(」「)」「$」「<」「>」「*」「?」「{」「}」「[」「]」「!」を除く)です。
証明書の鍵番号	対象証明書の鍵番号を指定します。 「証明書のシリアル番号」か本項目のどちらかを指定します。
証明書のシリアル番号	対象証明書のシリアル番号を指定します。 「証明書の鍵番号」か本項目のどちらかを指定します。
CA 名称	証明書の認証局名称を指定します。「ca1」を選択します。

5. 証明書の取得に成功すると、CLA インストールエッジ/デバイス上に取得・保存した証明書の情報が表示されます。

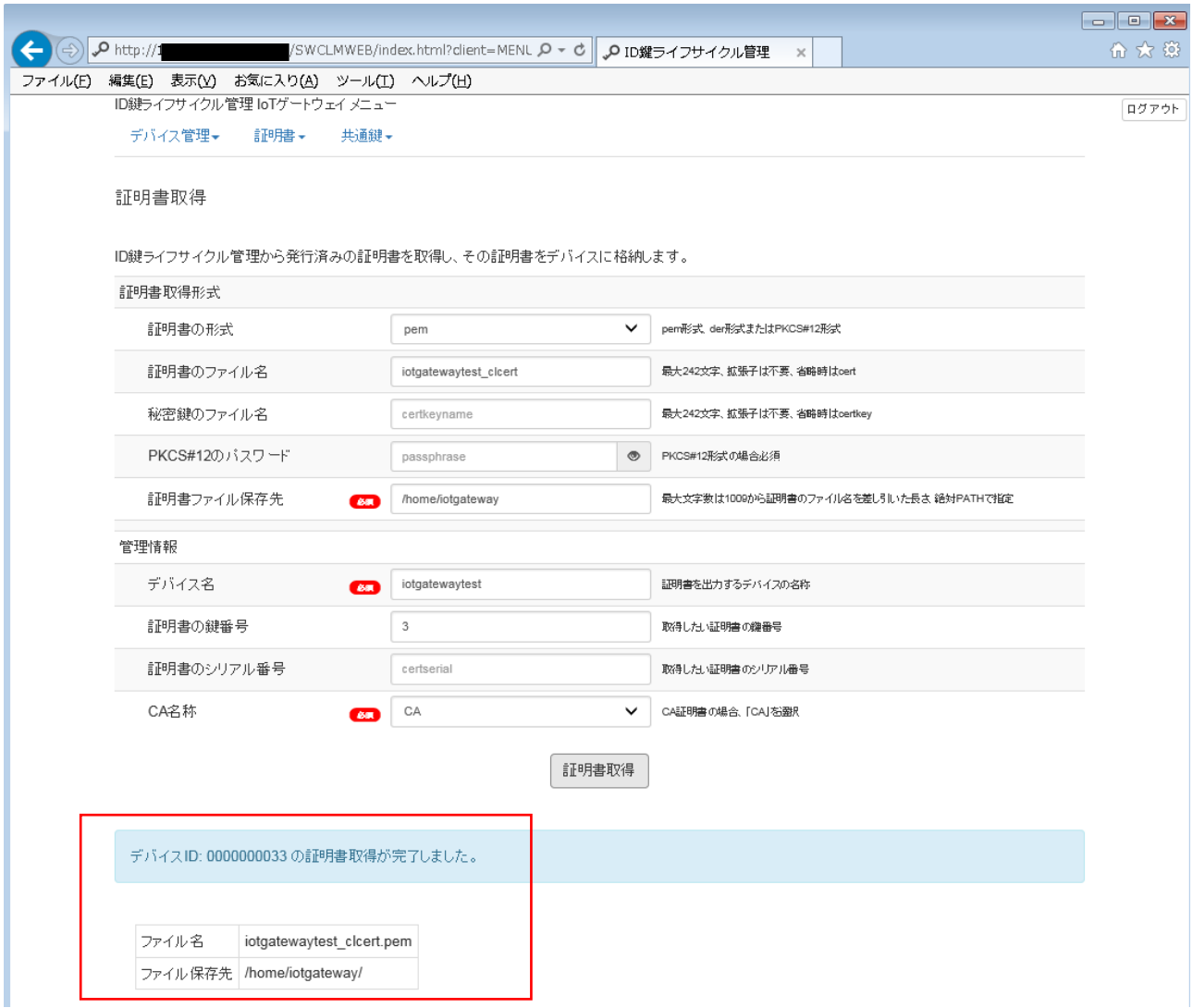


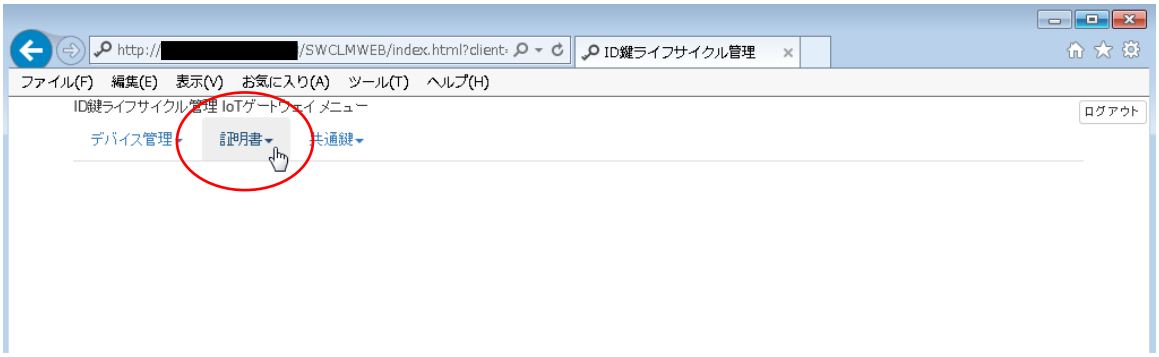
表 6-11 証明書取得 取得結果

項目	説明
ファイル名	CLM から取得した証明書のファイル名。
ファイル保存先	証明書を保存したディレクトリの PATH。

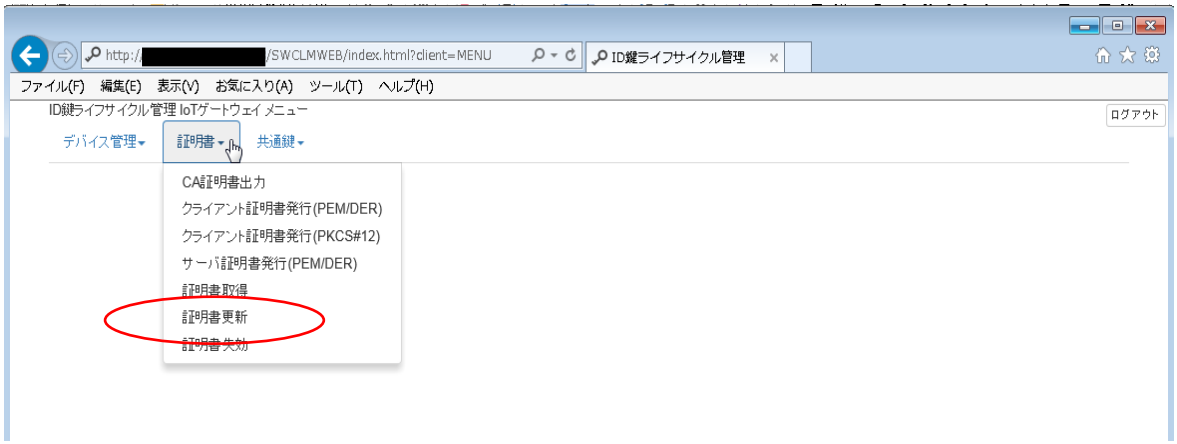
6.3.8 証明書更新

1. 簡易 WebUI にログインしていない場合は、簡易 WebUI へログインします。
   ログイン手順は、6.3.2 章をご覧ください。

2. メニューの「証明書」をクリックします。



3. 「証明書」メニューが表示されます。「証明書更新」をクリックします。



4. 「証明書更新」画面が表示されます。必要事項を入力し、「証明書更新」ボタンをクリックします

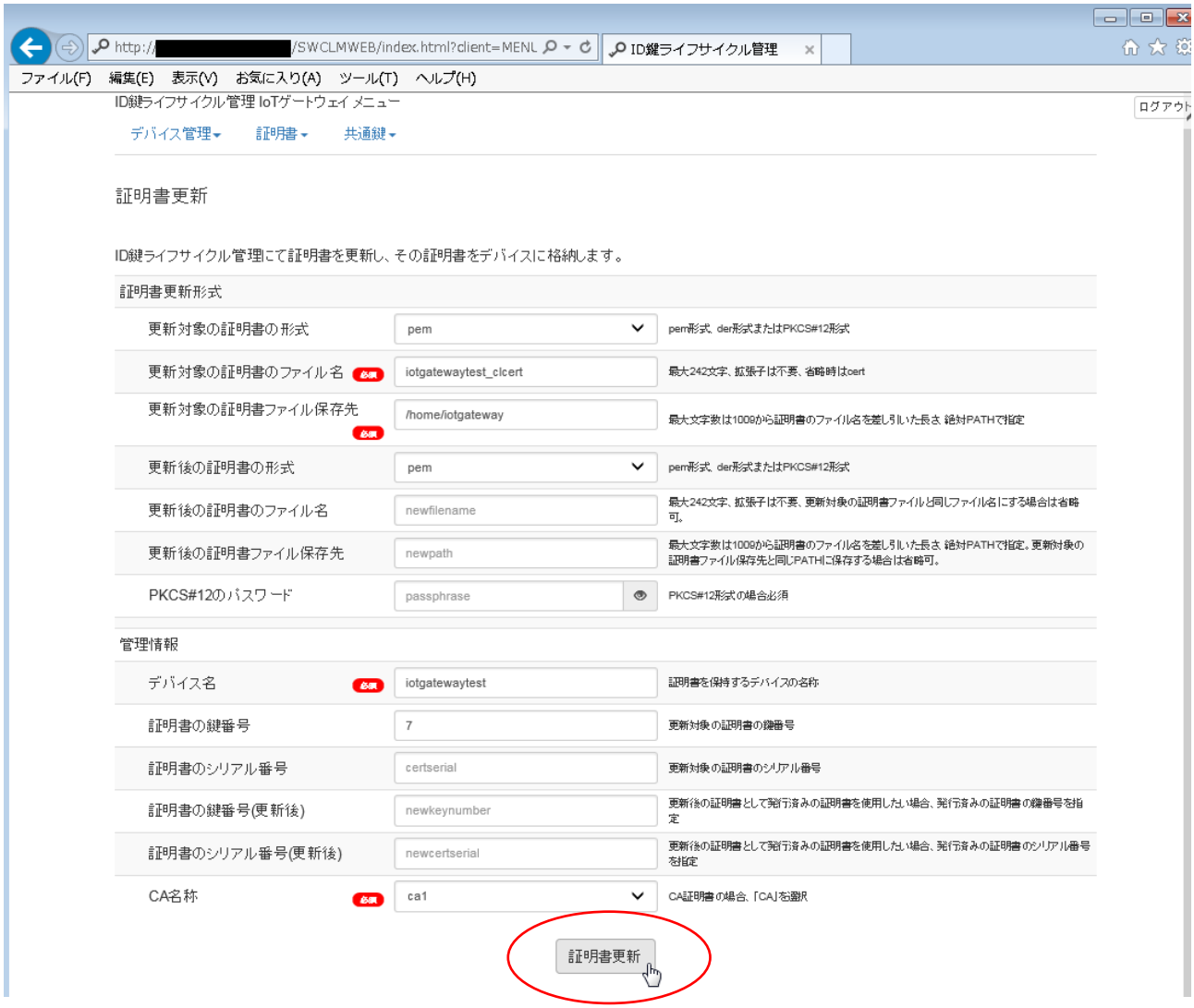


表 6-12 証明書更新 入力項目

項目	説明
更新対象の証明書の形式	更新対象証明書の種別を指定します。 以下のいずれかを指定してください。 pem: pem 形式の証明書 der：der 形式の証明書 p12：PKCS#12 形式の証明書(chain なし)
更新対象の証明書のファイル名	更新対象証明書ファイル名を指定します。 拡張子は不要です。既定値は「cert」です。 最大文字列長は、242byte です。 使用可能文字種は、ASCII 0x21～0x7E(但し、「/」を除く。「`」「"」を指定する場合は、バックスラッシュでエ

	スケープすること)です。
更新対象の証明書ファイル保存先	更新対象証明書が保存されているディレクトリを絶対パスで指定します。 最大文字列長は、1009 – (「更新対象の証明書のファイル名」に指定したファイル名の最大文字列長) byte です。 使用可能文字種は、ASCII 0x21～0x7E(但し、「;」「|」「&」「`」「(」「)」「$」「<」「>」「*」「?」「{」「}」「[」「]」「!」を除く)です。
更新後の証明書の形式	更新後の証明書の種別を指定します。 以下のいずれかを指定してください。 pem: pem 形式の証明書 der：der 形式の証明書 p12：PKCS#12 形式の証明書(chain なし)
更新後の証明書のファイル名	更新した証明書を保存するファイル名を指定します。 拡張子は不要です。 「更新対象の証明書ファイル名」に指定したファイル名と異なるファイル名を付与する場合に指定してください。 最大文字列長は、242byte です。 使用可能文字種は、ASCII 0x21～0x7E(但し、「/」を除く。「`」「"」を指定する場合は、バックスラッシュでエスケープすること)です。
更新後の証明書ファイル保存先	更新後の証明書出力ディレクトリを絶対パスで指定します。 「更新対象の証明書ファイル保存先」に指定したディレクトリと異なるディレクトリに出力する場合に指定してください。 最大文字列長は、1009 – (「更新後の証明書のファイル名」に指定したファイル名の文字列長) byte です。 使用可能文字種は、ASCII 0x21～0x7E(但し、「;」「|」「&」「`」「(」「)」「$」「<」「>」「*」「?」「{」「}」「[」「]」「!」を除く)です。
PKCS#12 のパスワード	更新後の証明書の秘密鍵のパスフレーズを指定します。 取得対象証明書の種別に「p12」が指定された場合、PKCS#12 形式に変換する際のパスフレーズにも利用します。 最大文字列長は、50byte です。

	使用可能文字種は、ASCII 0x21～0x7E(但し、「;」「|」「&」「`」「(」「)」「$」「<」「>」「*」「?」「{」「}」「[」「]」「!」を除く)です。
デバイス名	どの機器(デバイス・エッジ・サーバなど)から更新されるかを CLM が判別するために指定します。 キッティング済みエッジ GW では、既にデバイス名は CLM にエッジ ID で登録されていますので、エッジ ID を指定します。 最大文字列長は、239byte です。 使用可能文字種は、英数記号(但し、「;」「|」「&」「`」「(」「)」「$」「<」「>」「*」「?」「{」「}」「[」「]」「!」を除く)です。
証明書の鍵番号	対象証明書の鍵番号を指定します。 「証明書のシリアル番号」か本項目のどちらかを指定します。
証明書のシリアル番号	対象証明書のシリアル番号を指定します。 「証明書の鍵番号」か本項目のどちらかを指定します。
証明書の鍵番号(更新後)	更新後の証明書の鍵番号を指定します。 既存の証明書を更新後証明書としたい場合に、「証明書のシリアル番号(更新後)」か本項目のどちらかを指定してください。
証明書のシリアル番号(更新後)	更新後の証明書のシリアル番号を指定します。 既存の証明書を更新後証明書としたい場合に、「証明書の鍵番号(更新後)」か本項目のどちらかを指定してください。
CA 名称	証明書の認証局名称を指定します。「ca1」を選択します。

5. 証明書の更新に成功すると、CLA インストールエッジ/デバイス上に更新・保存した証明書の情報が表示されます。

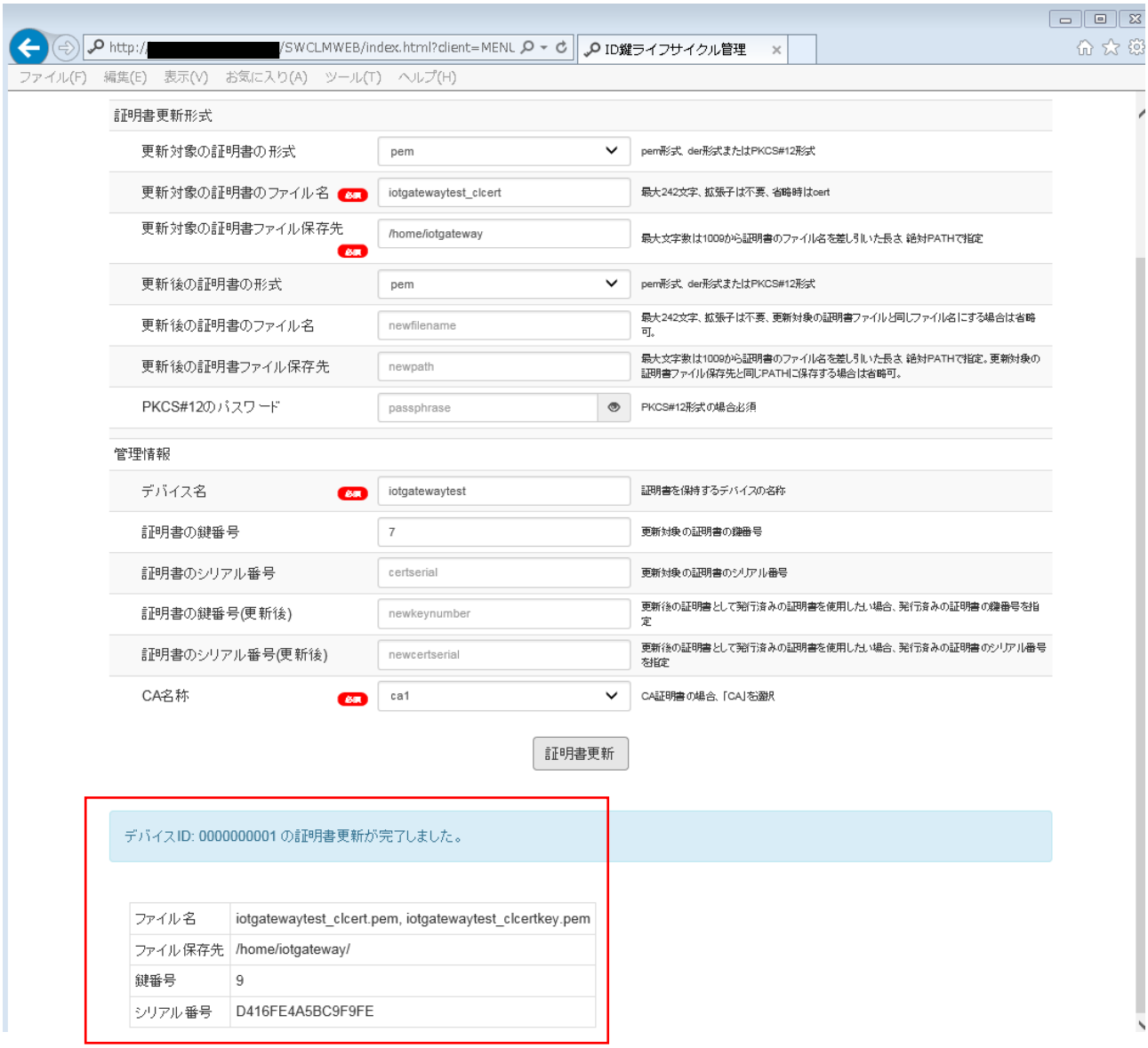


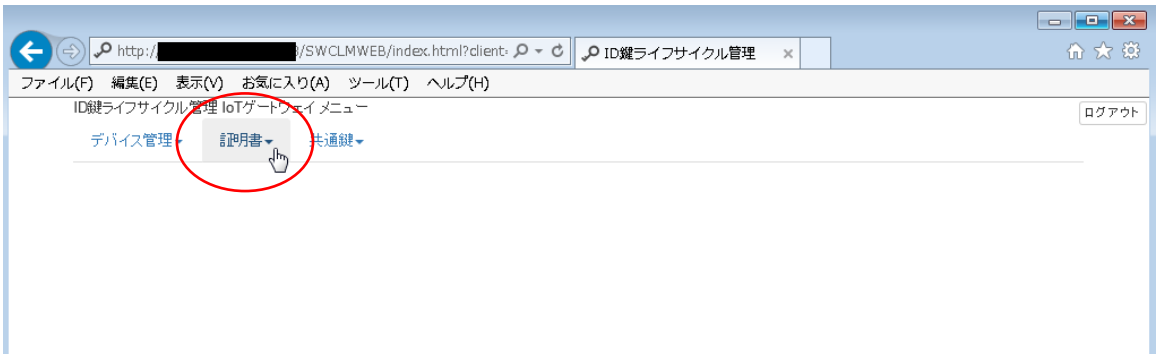
表 6-13 証明書更新 更新結果

項目	説明
ファイル名	CLM で更新した証明書のファイル名。 順に、証明書を更新した CA の CA 証明書のファイル名、証明書のファイル名、証明書の秘密鍵のファイル名。
ファイル保存先	証明書を保存したディレクトリの PATH。
証明書の鍵番号	更新後の証明書の鍵番号。
証明書のシリアル番号	更新後の証明書のシリアル番号。

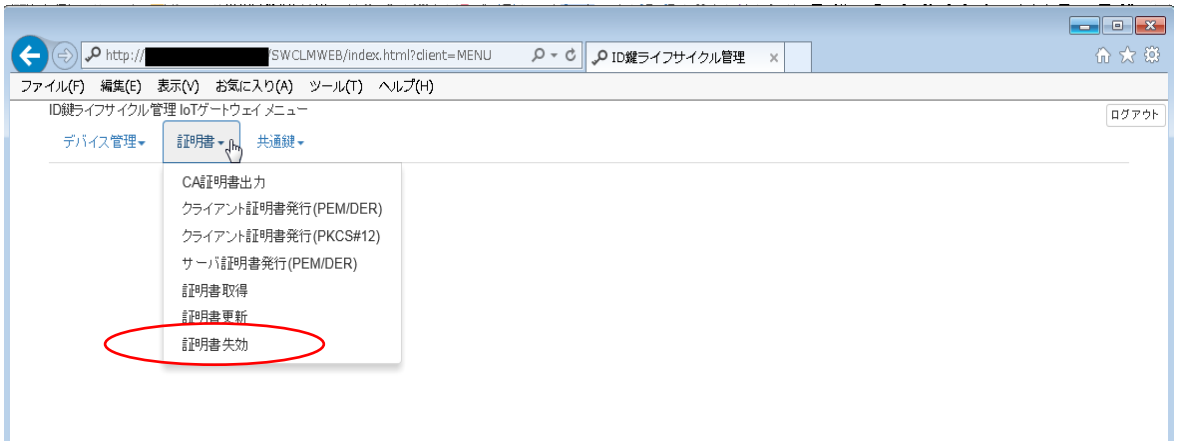
6.3.9 証明書失効

1. 簡易 WebUI にログインしていない場合は、簡易 WebUI へログインします。
   ログイン手順は、6.3.2 章をご覧ください。

2. メニューの「証明書」をクリックします。



3. 「証明書」メニューが表示されます。「証明書失効」をクリックします。



4. 「証明書失効」画面が表示されます。必要事項を入力し、「証明書失効」ボタンをクリックします。

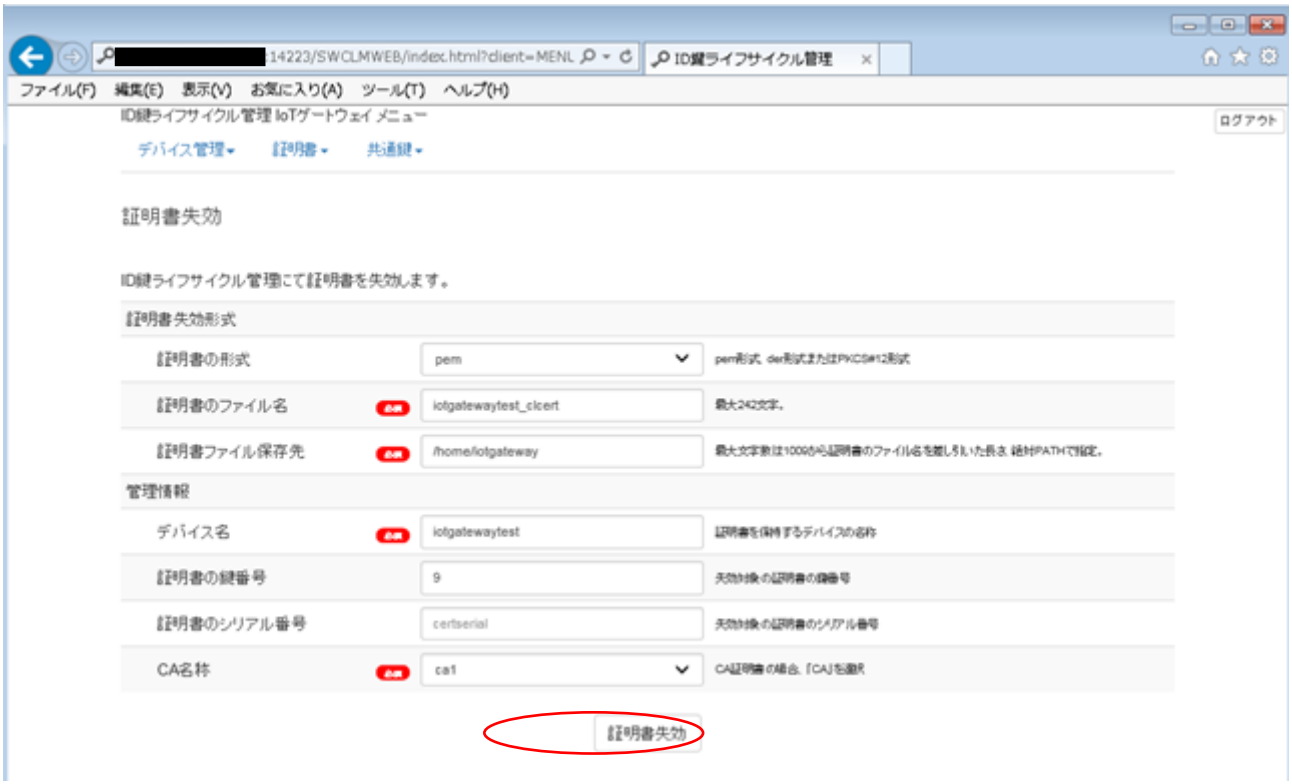


表 6-14 証明書取得 入力項目

項目	説明
証明書の形式	失効対象証明書の種別を指定します。 以下のいずれかを指定してください。 pem: pem 形式の証明書 der：der 形式の証明書 p12：PKCS#12 形式の証明書(chain なし)
証明書のファイル名	失効対象証明書ファイル名を指定します。拡張子は不要です。 最大文字列長は、242byte です。 使用可能文字種は、ASCII 0x21～0x7E(但し、「/」を除く。「`」「"」を指定する場合は、バックスラッシュでエスケープすること)です。
証明書ファイル保存先	失効対象証明書が保存されているディレクトリを絶対パスで指定します。 最大文字列長は、1009 – (「証明書のファイル名」に指定したファイル名の文字列長) byte です。 使用可能文字種は、ASCII 0x21～0x7E(但し、「;」「|」「&」「`」「(」「)」「$」「<」「>」「*」「?」「{」「}」「[」

	「]」「!」を除く)です。
デバイス名	どの機器(デバイス・エッジ・サーバなど)から失効されるかを CLM が判別するために指定します。 キッティング済みエッジ GW では、既にデバイス名は CLM にエッジ ID で登録されていますので、エッジ ID を指定します。 最大文字列長は、239byte です。 使用可能文字種は、英数記号(但し、「;」「|」「&」「`」「(」「)」「$」「<」「>」「*」「?」「{」「}」「[」「]」「!」を除く)です。
証明書の鍵番号	対象証明書の鍵番号を指定します。 「証明書のシリアル番号」か本項目のどちらかを指定します。
証明書のシリアル番号	対象証明書のシリアル番号を指定します。 「証明書の鍵番号」か本項目のどちらかを指定します。
CA 名称	証明書の認証局名称を指定します。「ca1」を選択します。

5. 「証明書失効確認」ダイアログが表示されます。証明書を失効してよい場合は、「実行」をクリックします。

6.　証明書の失効に成功すると、失効した証明書の情報が表示されます。

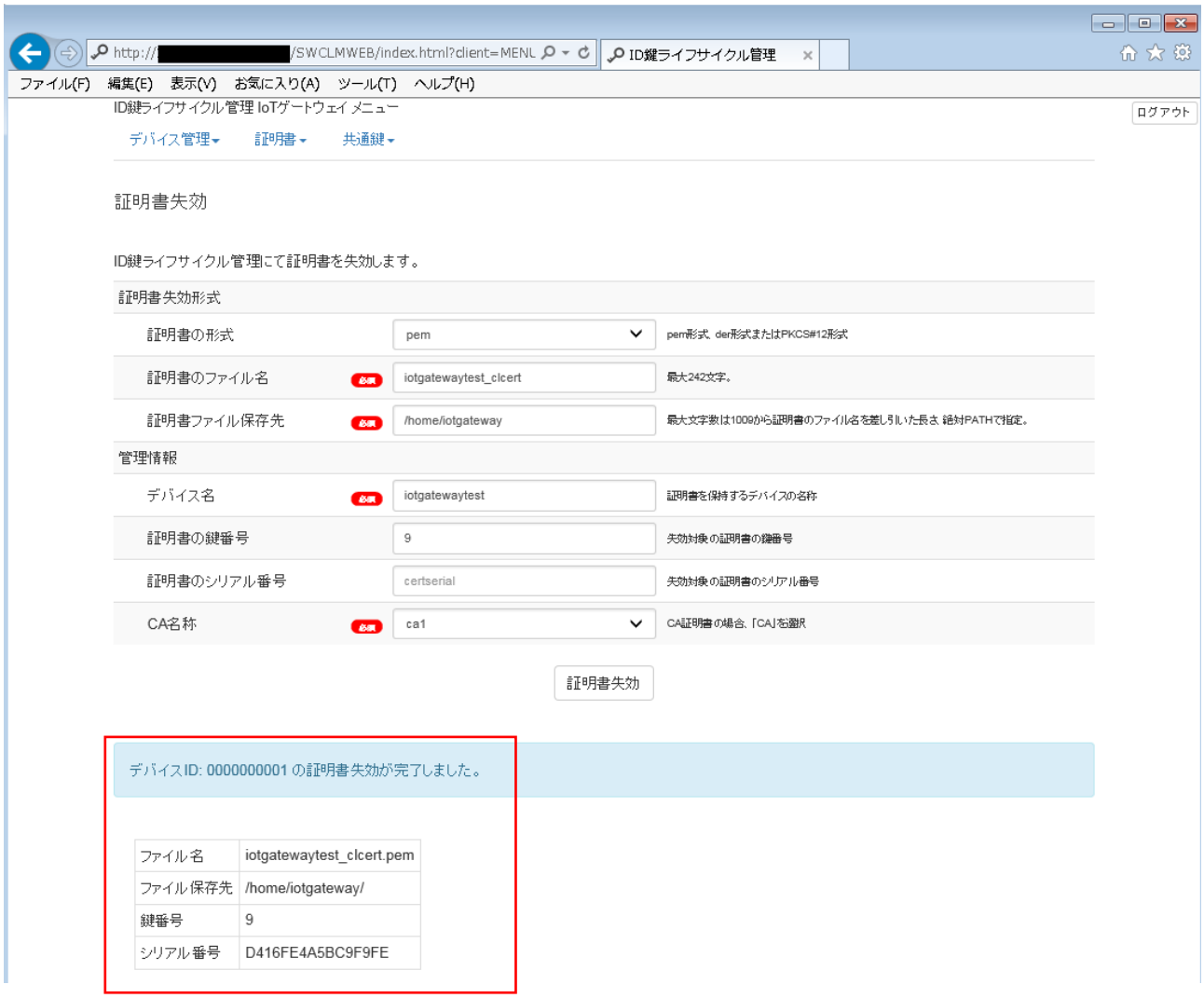


表 6-15 証明書失効 失効結果

項目	説明
ファイル名	CLM で失効した証明書のファイル名。
ファイル保存先	失効した証明書を保存している、CLA インストールエッジ/デバイス上のディレクトリの PATH。
証明書の鍵番号	失効した証明書の鍵番号。
証明書のシリアル番号	失効した証明書のシリアル番号。

7　共通鍵管理

CLM は、エッジ-クラウド間の通信を安全に行うための共通鍵を発行することが可能です。また、発行済み共通鍵の取得・更新・削除・照合も可能です。

本章では、共通鍵の管理方法について説明します。

7.1 API による共通鍵の管理

共通鍵の管理を行うためのインターフェースとして、CLM では WebAPI を提供しています。WebAPI を使用することにより、HTTPS プロトコルを使用して共通鍵の発行・取得・更新・削除・照合を行うことができます。

また、軽量暗号 開発キット(TWINE)と連携し、TWINE 用暗号鍵を用いた共通鍵の照合を行うこともできます。

WebAPI の詳細については、別紙「WebAPI リファレンス」をご覧ください。

7.2 コマンドによる共通鍵の管理

共通鍵の管理を行うためのインターフェースとして、CLM/CLA ではコマンド(WebAPI 実行コマンド)を提供しています。WebAPI 実行コマンドを実行することで、容易に共通鍵の発行・取得・更新・削除・照合を行うことができます。

また、軽量暗号 開発キット(TWINE)と連携し、CLM で発行した共通鍵を TWINE 用暗号鍵に変換して保存することや、TWINE 用暗号鍵を用いた共通鍵の照合を行うこともできます。

WebAPI 実行コマンドの詳細については、別紙「コマンドリファレンス」をご覧ください。

7.3 WebUI による共通鍵の管理

共通鍵の管理を行うためのインターフェースとして、次の 2 つのインターフェースを提供しています。WebUI を使用することにより共通鍵の発行・取得・更新・失効が可能です。また、軽量暗号 開発キット(TWINE)と連携し、CLM で発行した共通鍵を TWINE 用暗号鍵に変換して保存することもできます。

- CLA 簡易 WebUI　(エッジ/デバイス上で共通鍵を直接管理)
- CLM WebUI　(エッジ/デバイス上の共通鍵をリモートから管理)

本章では、CLA の簡易 WebUI による共通鍵の発行・取得・更新・失効について説明します。共通鍵の発行・取得・更新・削除の各機能詳細については、別紙「WebAPI リファレンス」の「5. 共通鍵管理 API」をご覧ください。

CLM の WebUI によるリモートからの共通鍵管理については、8 章をご覧ください。

7.3.1 簡易 WebUI 初回アクセス時の事前準備

簡易 WebUI へアクセスする端末・ブラウザに、設定を行います。設定は、初回アクセス前に行います。また、設定を行った端末・ブラウザと異なる端末・ブラウザで簡易 WebUI へアクセスする場合は、新たに当該端末で設定を行う必要があります。

詳細は、6.3.1 章をご覧ください。

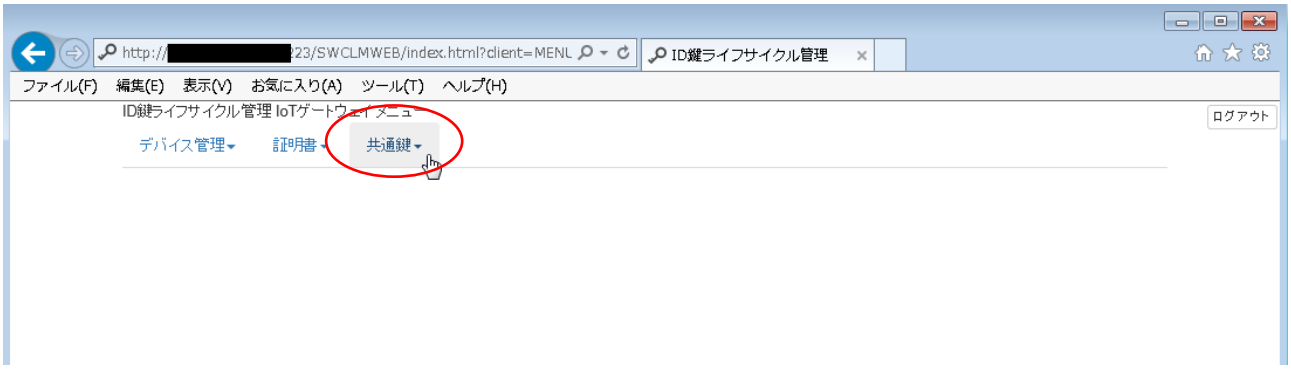
7.3.1 簡易 WebUI へのログイン・接続設定

CLA の簡易 WebUI にアクセスし、共通鍵の管理を行うには、簡易 WebUI へのログインと接続設定が必要です。詳細は、6.3.2 章をご覧ください。

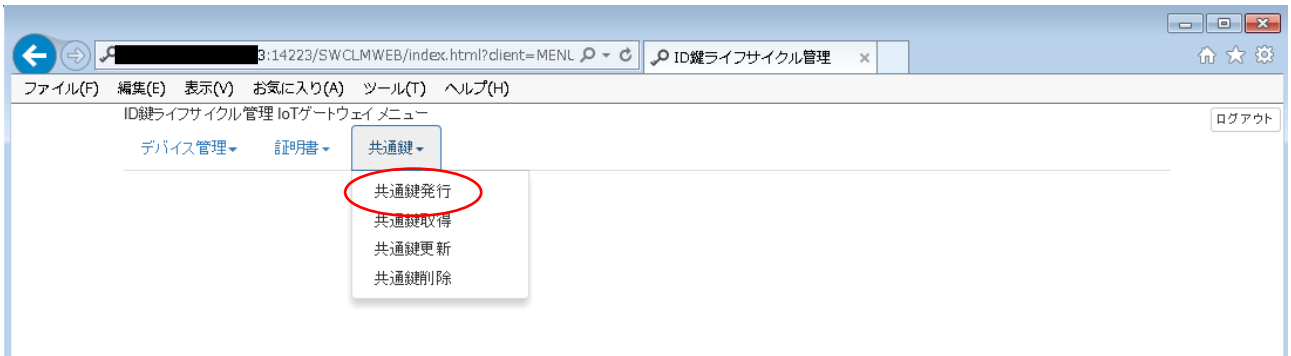
7.3.2 共通鍵発行

1. 簡易 WebUI にログインしていない場合は、簡易 WebUI へログインします。
   ログイン手順は、6.3.2 章をご覧ください。

2. メニューの「共通鍵」をクリックします。



3. 「共通鍵」メニューが表示されます。「共通鍵発行」をクリックします。



4. 「共通鍵発行」画面が表示されます。必要事項を入力し、「共通鍵発行」ボタンをクリックします

表 7-1 共通鍵発行 入力項目

項目	説明
共通鍵の形式	鍵を利用する暗号化方式を指定します。 以下のいずれかを指定します。 AES　：AES で利用する TWINE：TWINE で利用する
共通鍵長	発行する共通鍵の長さ(単位: bit)を指定します。 「共通鍵の形式」に「AES」を指定する場合、「128」または「256」のみ指定可能です。 「共通鍵の形式」に「TWINE」を指定する場合、「80」または「128」のみ指定可能です。
共通鍵ファイル名	発行した共通鍵を保存するファイル名を指定します。 省略時の既定値は「cmkey」です。 最大文字列長は、242byte です。 使用可能文字種は、ASCII 0x21～0x7E(但し、「/」を除く。「`」「"」を指定する場合は、バックスラッシュでエスケープすること。)
共通鍵ファイル保存先	共通鍵を出力するディレクトリを絶対パスで指定します。 ディレクトリは、エッジデバイス上のディレクトリを指定します。 最大文字列長は、1009 – (「共通鍵ファイル名」の文字

	列長) byte です。 使用可能文字種は、ASCII 0x21～0x7E(但し、「;」「|」「&」「`」「(」「)」「$」「<」「>」「*」「?」「{」「}」「[」「]」「!」を除く)です。
エイリアス	共通鍵に付与する Alias を指定します。 最大文字列長は、64byte です。 使用可能文字種は、英数記号(但し、「;」「|」「&」「`」「(」「)」「$」「<」「>」「*」「?」「{」「}」「[」「]」「!」を除く)です。
デバイス名	どの機器(デバイス・エッジ・サーバなど)に発行するかを CLM が管理・判別するために指定します。 キッティング済みエッジ GW では、既にデバイスは CLM にエッジ ID で登録されていますので、エッジ ID を指定します。 最大文字列長は、239byte です。 使用可能文字種は、英数記号(但し、「;」「|」「&」「`」「(」「)」「$」「<」「>」「*」「?」「{」「}」「[」「]」「!」を除く)です。

5. 共通鍵の発行に成功すると、CLA インストールエッジ/デバイス上に発行・保存した共通鍵の情報が表示されます。

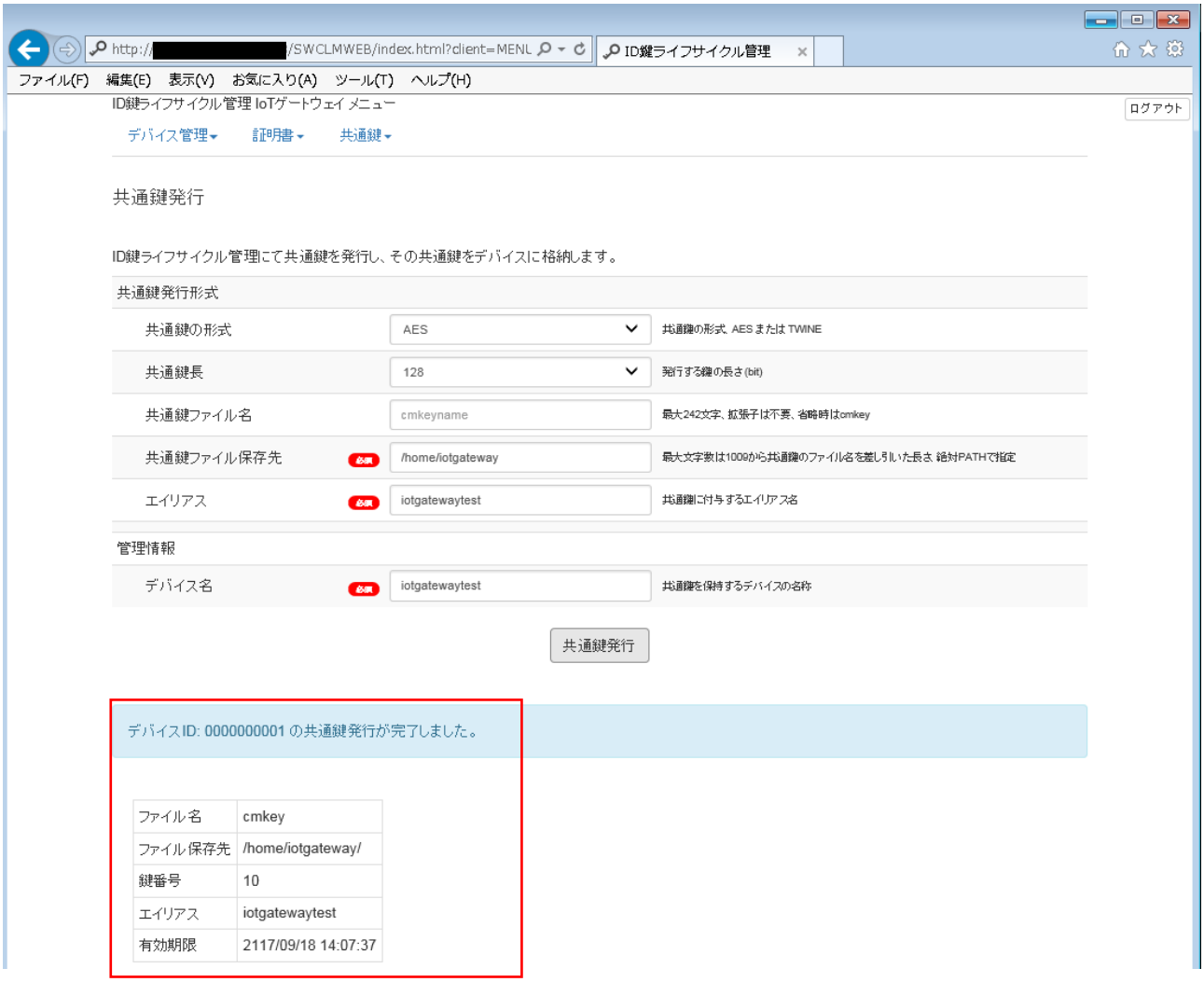


表 7-2 共通鍵発行 発行結果

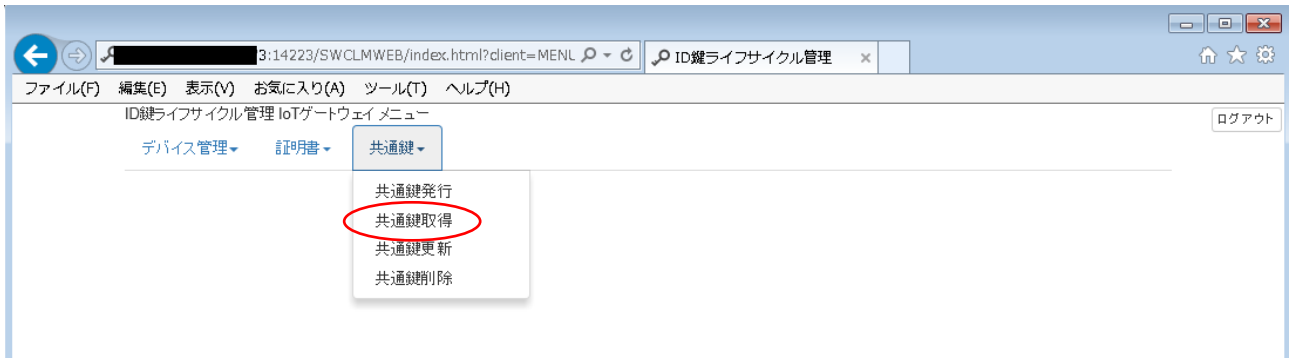
項目	説明
ファイル名	CLM から発行された共通鍵のファイル名。
ファイル保存先	共通鍵を保存したディレクトリの PATH。
鍵番号	共通鍵の鍵番号。
エイリアス	共通鍵発行時に指定した、共通鍵に付与するエイリアス。
有効期限	共通鍵の有効期限。

7.3.3 共通鍵取得

1. 簡易 WebUI にログインしていない場合は、簡易 WebUI へログインします。
   ログイン手順は、6.3.2 章をご覧ください。

2. メニューの「共通鍵」をクリックします。

3. 「共通鍵」メニューが表示されます。「共通鍵取得」をクリックします。



4. 「共通鍵取得」画面が表示されます。必要事項を入力し、「共通鍵取得」ボタンをクリックします

表 7-3 共通鍵取得 入力項目

項目	説明
共通鍵ファイル名	取得した共通鍵を保存するファイル名を指定します。 省略時の既定値は「cmkey」です。 最大文字列長は、242byte です。 使用可能文字種は、ASCII 0x21～0x7E(但し、「/」を除く。「`」「"」を指定する場合は、バックスラッシュでエスケープすること)です。
共通鍵ファイル保存先	共通鍵を出力するディレクトリを絶対パスで指定します。 ディレクトリは、エッジデバイス上のディレクトリを指定します。 最大文字列長は、1009 – (「共通鍵ファイル名」の文字列長) byte です。 使用可能文字種は、ASCII 0x21～0x7E(但し、「;」「|」「&」「`」「(」「)」「$」「<」「>」「*」「?」「{」「}」「[」「]」「!」を除く)です。
デバイス名	どの機器(デバイス・エッジ・サーバなど)から取得されるかを CLM が判別するために指定します。 キッティング済みエッジ GW では、既にデバイスは CLM にエッジ ID で登録されていますので、エッジ ID を指定

	します。 最大文字列長は、239byte です。 使用可能文字種は、英数記号(但し、「;」「|」「&」「`」「(」「)」「$」「<」「>」「*」「?」「{」「}」「[」「]」「!」を除く)です。
鍵番号	取得対象共通鍵の鍵番号を指定します。

5. 共通鍵の取得に成功すると、CLA インストールエッジ/デバイス上に取得・保存した共通鍵の情報が表示されます。

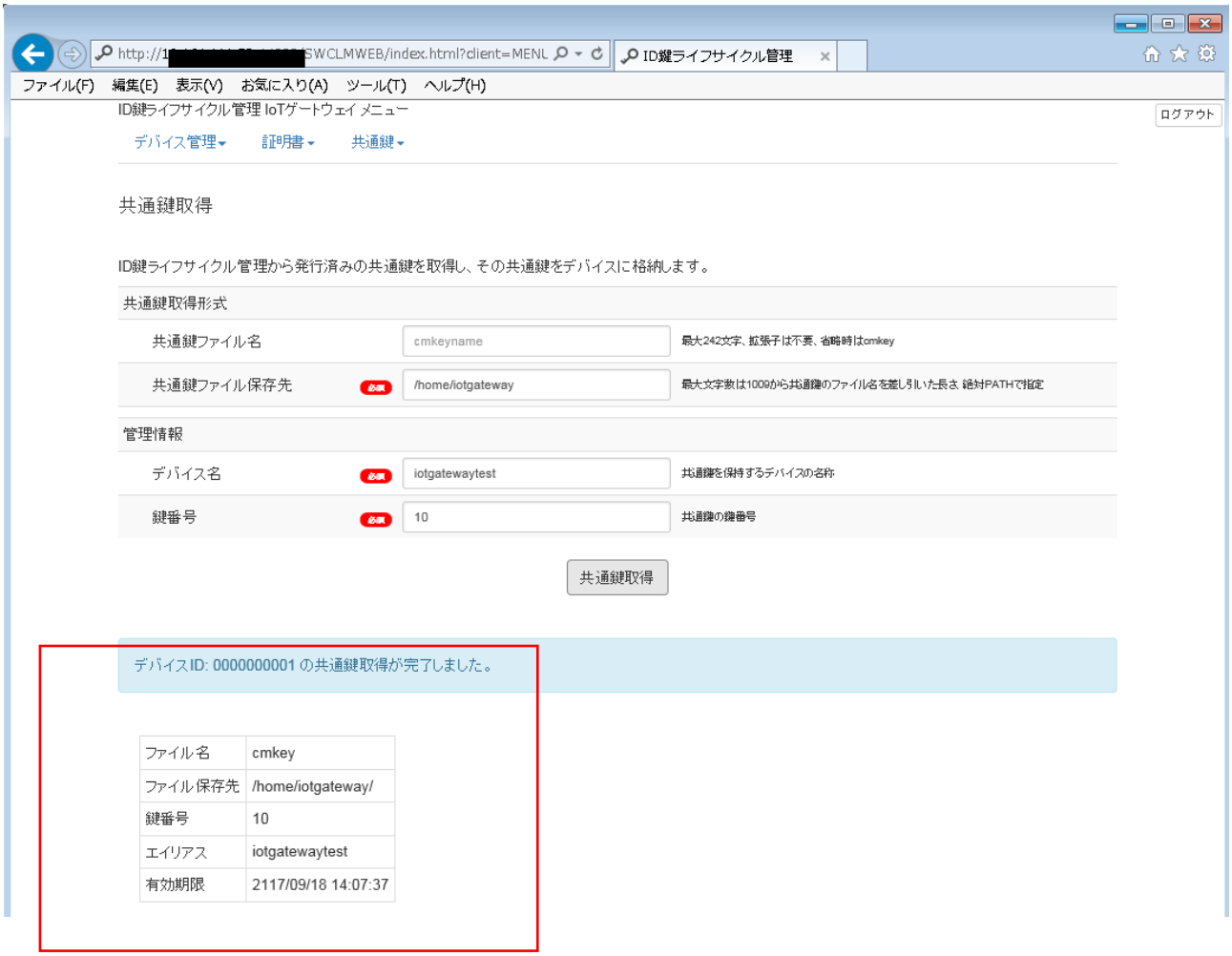


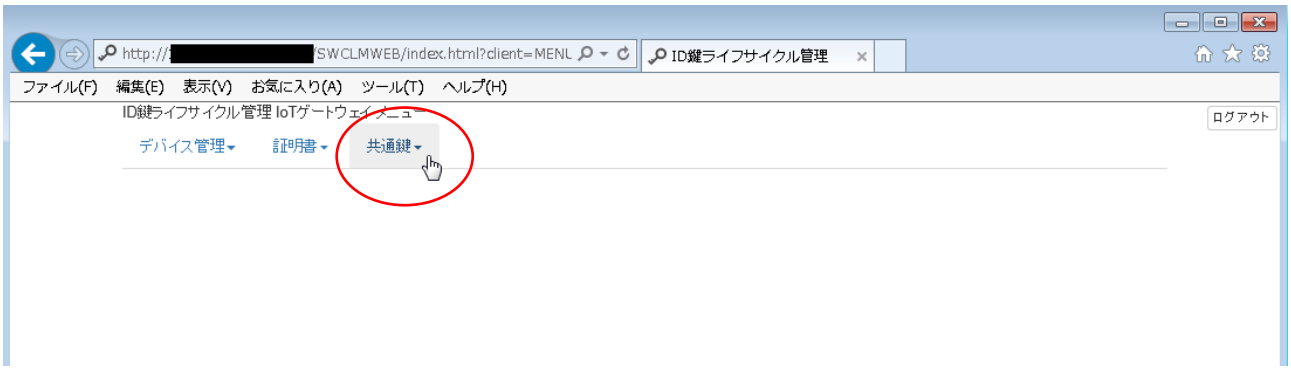
表 7-4 共通鍵取得 取得結果

項目	説明
ファイル名	CLM から取得した共通鍵のファイル名。
ファイル保存先	共通鍵を保存したディレクトリの PATH。
鍵番号	共通鍵の鍵番号。
エイリアス	共通鍵発行時に指定した、共通鍵のエイリアス。
有効期限	共通鍵の有効期限。

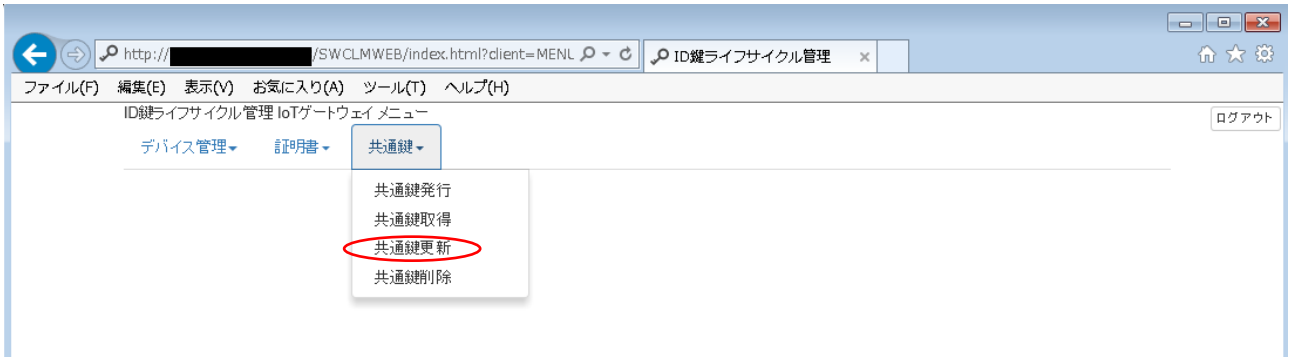
7.3.4 共通鍵更新

1. 簡易 WebUI にログインしていない場合は、簡易 WebUI へログインします。
   ログイン手順は、6.3.2 章をご覧ください。

2. メニューの「共通鍵」をクリックします。



3. 「共通鍵」メニューが表示されます。「共通鍵更新」をクリックします。



4. 「共通鍵更新」画面が表示されます。必要事項を入力し、「共通鍵更新」ボタンをクリックします

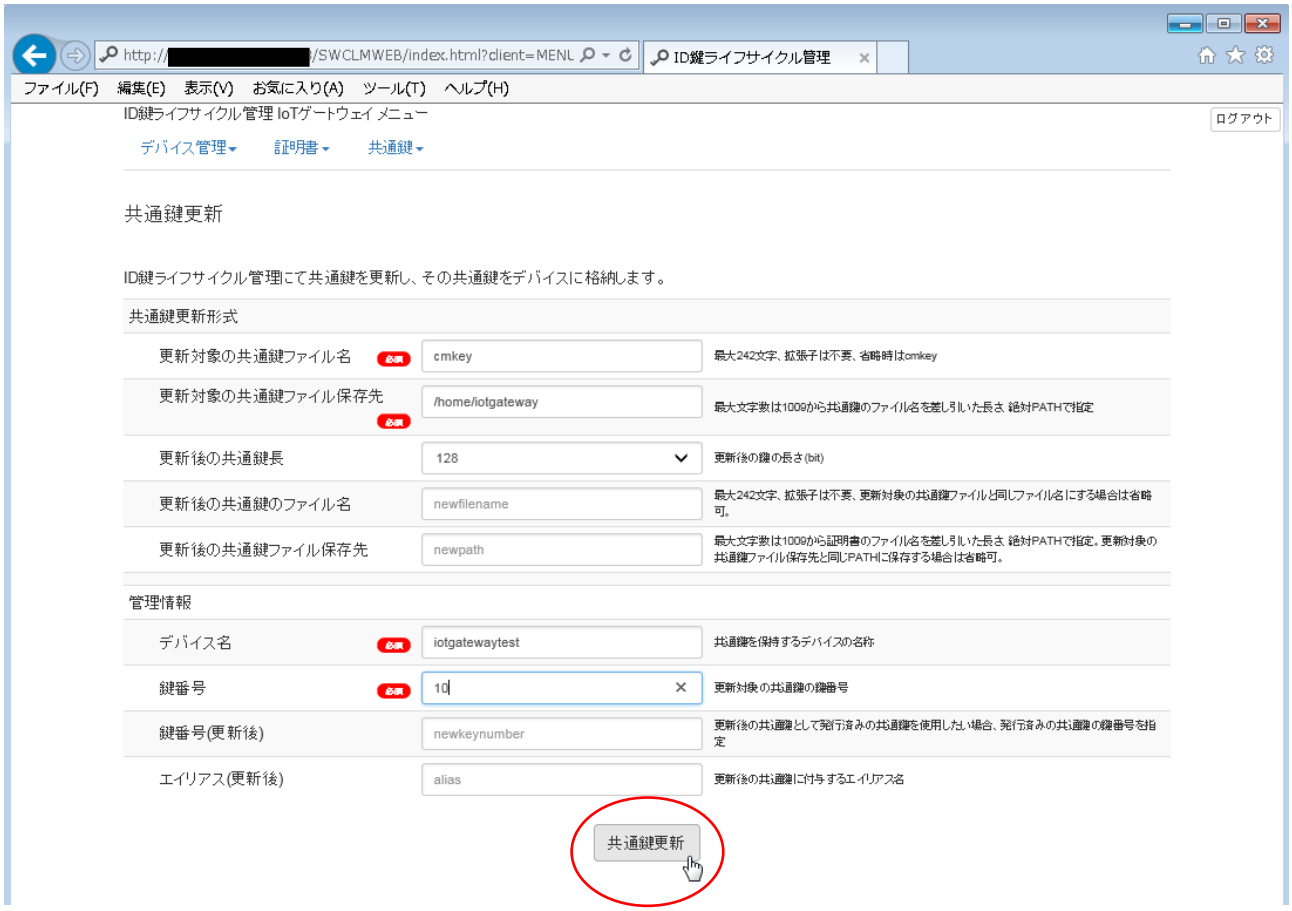


表 7-5 共通鍵更新 入力項目

項目	説明
更新対象の共通鍵ファイル名	更新対象共通鍵ファイル名を指定します。 省略時の既定値は「cmkey」です。 最大文字列長は、242byte です。 使用可能文字種は、ASCII 0x21～0x7E(但し、「/」を除く。「`」「"」を指定する場合は、バックスラッシュでエスケープすること)です。
更新対象の共通鍵ファイル保存先	更新対象共通鍵が保存されているディレクトリを絶対パスで指定します。 ディレクトリは、エッジデバイス上のディレクトリを指定します。 最大文字列長は、1009 – (「共通鍵ファイル名」の文字列長) byte です。 使用可能文字種は、ASCII 0x21～0x7E(但し、「;」「|」

	「&」「`」「(」「)」「$」「<」「>」「*」「?」「{」「}」「[」「]」「!」を除く)です。
更新後の共通鍵長	更新後の共通鍵の長さ(単位: bit)を指定します。 更新対象共通鍵の形式が「AES」である場合、「128」または「256」を指定します。 更新対象共通鍵の形式が「TWINE」である場合、「80」または「128」を指定します。
更新後の共通鍵ファイル名	更新した共通鍵を保存するファイル名を指定します。 「更新対象の共通鍵ファイル名」に指定したファイル名と異なるファイル名を付与する場合に指定してください。 最大文字列長は、242byte です。 使用可能文字種は、ASCII 0x21～0x7E(但し、「/」を除く。「`」「"」を指定する場合は、バックスラッシュでエスケープすること。)
更新後の共通鍵ファイル保存先	更新後の共通鍵出力ディレクトリを絶対パスで指定します。 「更新対象の共通鍵ファイル保存先」に指定したディレクトリと異なるディレクトリに出力する場合に指定してください。 最大文字列長は、1009 – (「更新後の共通鍵ファイル名」に指定したファイル名の文字列長) byte です。 使用可能文字種は、ASCII 0x21～0x7E(但し、「;」「|」「&」「`」「(」「)」「$」「<」「>」「*」「?」「{」「}」「[」「]」「!」を除く)です。
デバイス名	どの機器(デバイス・エッジ・サーバなど)から更新されるかを CLM が判別するために指定します。 キッティング済みエッジ GW では、既にデバイスは CLM にエッジ ID で登録されていますので、エッジ ID を指定します。 最大文字列長は、239byte です。 使用可能文字種は、英数記号(但し、「;」「|」「&」「`」「(」「)」「$」「<」「>」「*」「?」「{」「}」「[」「]」「!」を除く)です。
鍵番号	更新対象共通鍵の鍵番号。
鍵番号(更新後)	更新後の共通鍵の鍵番号を指定します。 既存の共通鍵を更新後共通鍵としたい場合に指定してく

	ださい。
エイリアス(更新後)	更新後の共通鍵に付与する Alias を指定します。 最大文字列長は、64byte です。 使用可能文字種は、英数記号(但し、「;」「|」「&」「`」「(」「)」「$」「<」「>」「*」「?」「{」「}」「[」「]」「!」を除く)です。

5. 共通鍵の更新に成功すると、CLA インストールエッジ/デバイス上に更新・保存した共通鍵の情報が表示されます。

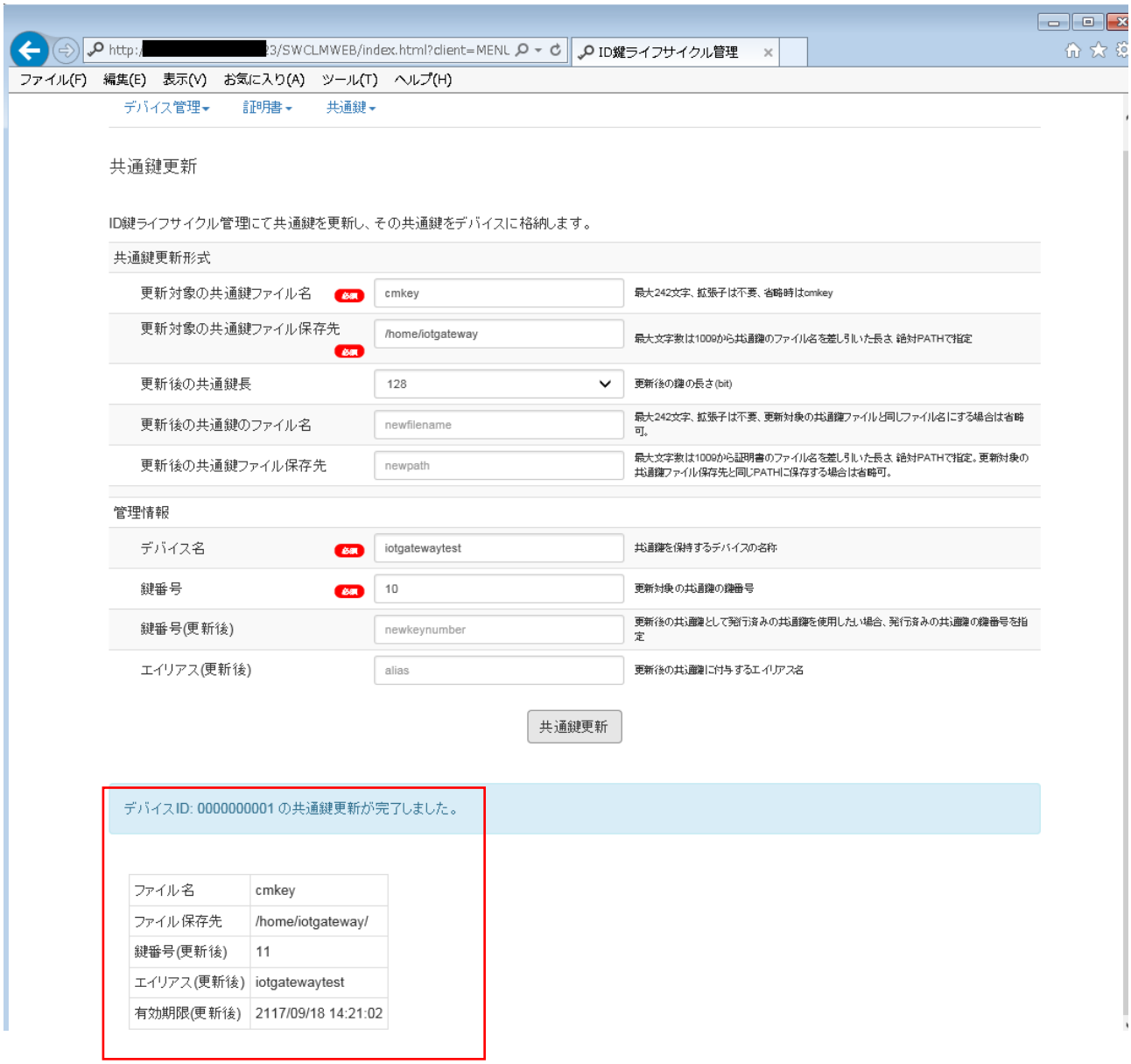


表 7-6 共通鍵更新 更新結果

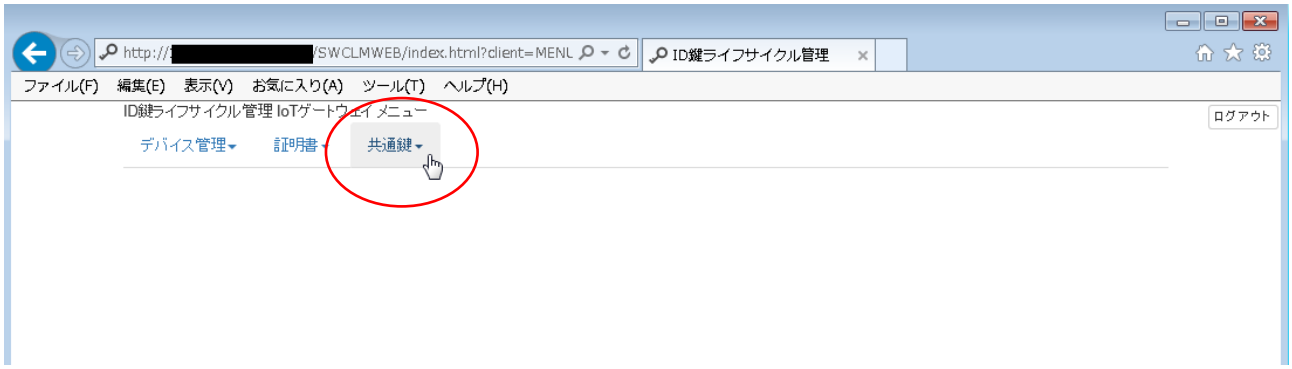
項目	説明
ファイル名	CLM で更新した共通鍵のファイル名。
ファイル保存先	共通鍵を保存したディレクトリの PATH。

鍵番号	更新後の共通鍵の鍵番号。
エイリアス	更新後の共通鍵に付与したエイリアス。
有効期限	更新後の共通鍵の有効期限。

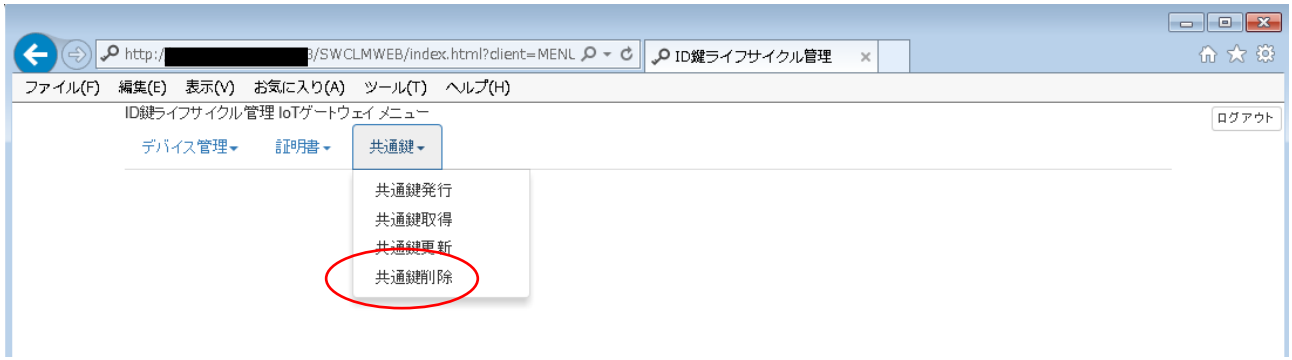
7.3.5 共通鍵削除

1. 簡易 WebUI にログインしていない場合は、簡易 WebUI へログインします。
   ログイン手順は、6.3.2 章をご覧ください。

2. メニューの「共通鍵」をクリックします。



3. 「共通鍵」メニューが表示されます。「共通鍵削除」をクリックします。



4. 「共通鍵削除」画面が表示されます。必要事項を入力し、「共通鍵削除」ボタンをクリックします。

表 7-7 共通鍵削除 入力項目

項目	説明
共通鍵ファイル名	削除対象共通鍵ファイル名を指定します。 最大文字列長は、242byte です。 使用可能文字種は、ASCII 0x21～0x7E(但し、「/」を除く。「`」「"」を指定する場合は、バックスラッシュでエスケープすること)です。
削除対象ファイルの保存先	削除対象共通鍵が保存されているディレクトリを絶対パスで指定します。 ディレクトリは、エッジデバイス上のディレクトリを指定します。 最大文字列長は、1009 – (「共通鍵ファイル名」の文字列長) byte です。 使用可能文字種は、ASCII 0x21～0x7E(但し、「;」「|」「&」「`」「(」「)」「$」「<」「>」「*」「?」「{」「}」「[」「]」「!」を除く)です。
デバイス名	どの機器(デバイス・エッジ・サーバなど)から削除されるかを CLM が判別するために指定します。 キッティング済みエッジ GW では、既にデバイスは CLM にエッジ ID で登録されていますので、エッジ ID を指定します。

	最大文字列長は、239byte です。 使用可能文字種は、英数記号(但し、「;」「|」「&」「`」「(」「)」「$」「<」「>」「*」「?」「{」「}」「[」「]」「!」を除く)です。
鍵番号	対象共通鍵の鍵番号を指定します。

5. 「共通鍵削除確認」ダイアログが表示されます。共通鍵を削除してよい場合は、「実行」をクリックします。

7. 共通鍵の削除に成功すると、削除した共通鍵の情報が表示されます。

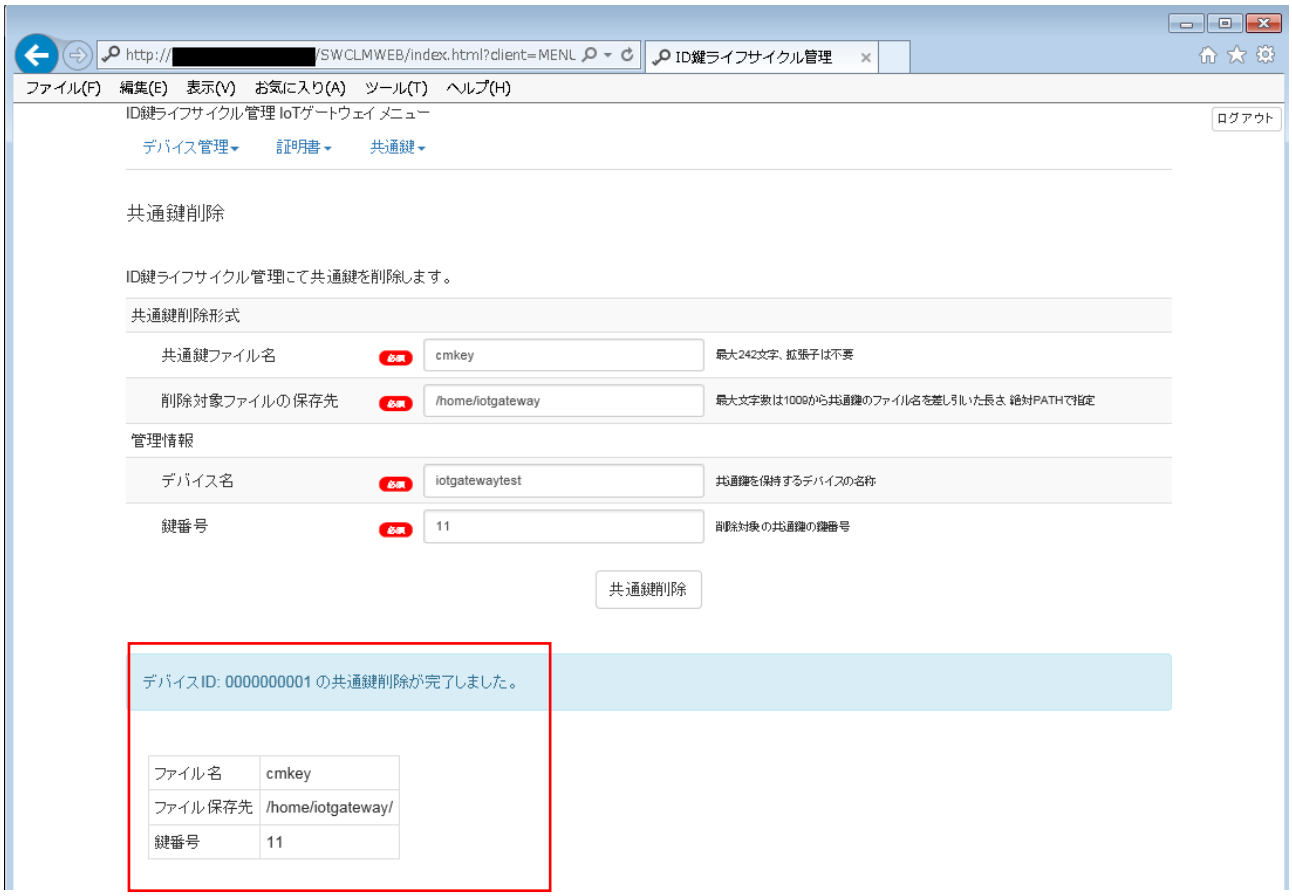


表 7-8 共通鍵削除 削除結果

項目	説明
ファイル名	CLM で削除した共通鍵のファイル名。
ファイル保存先	削除した共通鍵を保存している、CLA インストールエッジ/デバイス上のディレクトリの PATH。
鍵番号	削除した共通鍵の鍵番号。

8 デバイス管理

CLM では、認証情報(ID・鍵)と認証情報を保有するデバイスとを紐づけして管理するために、デバイスの情報を管理する機能(デバイス管理)を保有しています。

デバイス管理は、デバイス情報を管理しますが、その情報は証明書管理・共通鍵管理で自動的に情報が登録されるため、特に明示的な操作は不要です。具体的には次の通りです。

- 証明書管理
  証明書発行を行う際に指定する必須項目のデバイス名で自動的にデバイス情報が登録されます。
- 共通鍵管理
  共通鍵発行を行う際に指定する必須項目のデバイス名で自動的にデバイス情報が登録されます。

なお、デバイス情報を手動で管理したい場合は、EDMS オプションを購入ください。
利用方法は、EDMS オプションのマニュアルに記載されています。

9 鍵自動更新・統計情報表示

エッジ/デバイスやサーバに CLA をインストールし CLM の管理下に置くことで、エッジ/デバイス・サーバ上の鍵(証明書・共通鍵)を定期的に自動更新することができます。

また、鍵の更新状況はグラフ化して、CLM(WebUI)から確認することができます。

9.1 運用フロー

鍵自動更新の運用フローは、次の通りです。フローの詳細は、以降の章で説明します。

9.2 事前準備

本機能を使用するにあたり、鍵の管理方法とエッジ/デバイス・サーバの管理単位を決定します。

また、CLM(WebUI)へアクセスする端末・ブラウザに設定を行います。

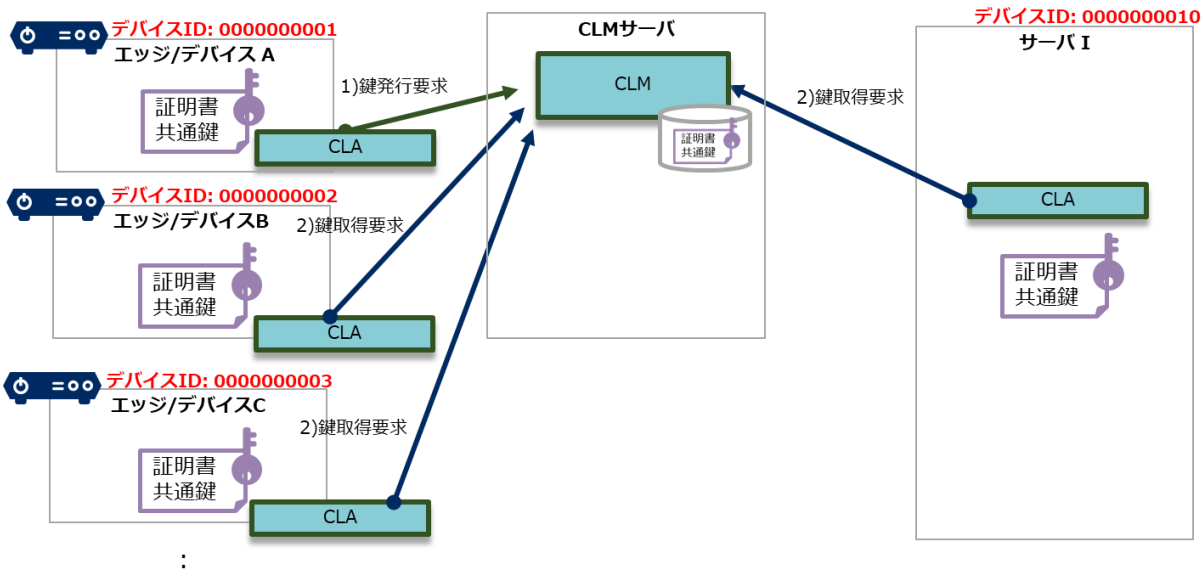
9.2.1 鍵の管理方法の決定

エッジ/デバイス・サーバが使用する鍵の管理方法を決定します。鍵の管理方法は、CLM で発行・管理している鍵をどのようにエッジ/デバイス・サーバで使用するかにより決定します。

- 管理方法 A: 全エッジ/デバイス・サーバで、同一の鍵を使用する

  鍵をシステム内のエッジ/デバイス・サーバのうち 1 か所で発行・更新します。発行・更新した鍵をシステム内のエッジ/デバイス・サーバで取得し使用します。下図では、エッジ/デバイス A で鍵を発行・更新し、エッジ/デバイス B、C、サーバ I でエッジ/デバイス A が発行・更新した鍵を取得し、使用しています。

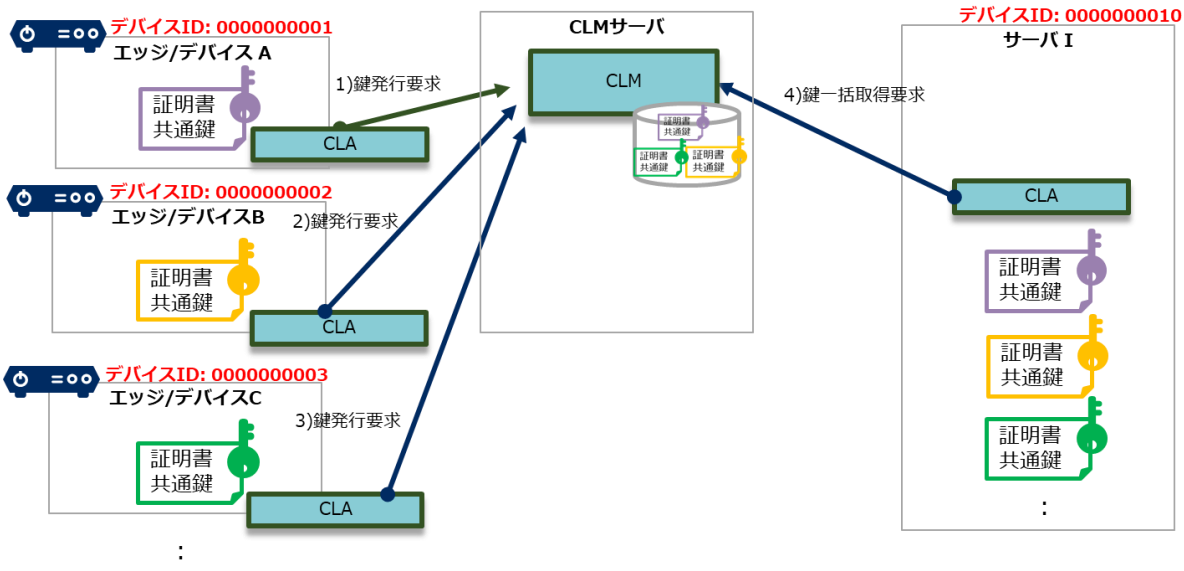
  システム全体で共通の鍵を使用したい場合に、本管理方法を選択します。



➢ 管理方法 B: エッジ/デバイス単位で異なる鍵を使用する

鍵をシステム内のエッジ/デバイスでそれぞれ発行・更新し、サーバでエッジ/デバイスが発行・更新した鍵を一括取得して使用します。下図では、エッジ/デバイス A～C で鍵を発行・更新し、サーバ I でエッジ/デバイス A～C が発行・更新した鍵を取得し、使用しています。

エッジ/デバイス毎に異なる鍵を使用したい場合に、本管理方法を選択します。



9.2.2 エッジ/デバイスの管理単位の決定

鍵の管理方法を決定後、エッジ/デバイスの管理単位を決定します。CLM では、複数台のエッジ/デバイスを条件付けしてグルーピングし、グループ単位で管理することができるようになっています。エッジ/デバイスをグループ単位で管理することにより、鍵の更新状況の可視化や鍵の更新間隔等の設定変更をまとめて実施でき、膨大な数のエッジ/デバイスを容易に管理可能です。

エッジ/デバイスをグループ化するための条件(以降、グループ化条件と表記します)には、デバイス情報を

使用することができます。グループ化条件に使用可能なデバイス情報は、次の通りです。

表 9-1 グループ化条件に使用可能なデバイス情報

デバイス情報の項目	説明
devicename	デバイスのデバイス名
ipaddress	デバイスの IP アドレス
macaddress	デバイスの MAC アドレス
拡張情報	デバイスの拡張情報に設定した項目名

9.2.3 CLM(WebUI)へアクセスする端末・ブラウザへの設定

CLM(WebUI)、CLA(簡易 WebUI)へアクセスする端末・ブラウザに設定を行います。設定は、初回アクセス前に行います。詳細は 5.3.1 章、5.3.3 章をご覧ください。

9.3 グループ化規則の作成

9.2.1 章、9.2.2 章で決定した鍵の管理方法とデバイスの管理単位を CLM に登録します。

デバイスの管理単位は、CLM へ「グループ」という単位で登録します。鍵の管理方法は、グループの付属情報として登録します。グループは、複数登録可能です。複数のグループを束ねたものを「グループ化規則」と呼びます。

グループ化規則は、CLM にサンプルとして登録されている以下の規則を更新して作成するか、新規に作成します。グループ化規則の作成・更新手順については、9.9.1 章をご覧ください。

表 9-2 グループ化規則 サンプル 一覧

グループ化規則名	説明
デフォルトグループ専用サンプル	すべてのデバイスを「デフォルトグループ」という 1 つのグループとしてグループ化するサンプル。
デバイス名用サンプル	エッジ/デバイスを、デバイスのデバイス名でグループ化するサンプル。
IP アドレス用サンプル	エッジ/デバイスを、デバイスの IP アドレスでグループ化するサンプル。
MAC アドレス用サンプル	エッジ/デバイスを、デバイスの MAC アドレスでグループ化するサンプル。
地区用サンプル	エッジ/デバイスを、デバイスの拡張情報でグループ化するサンプル。 本サンプルでは、拡張情報に「chiku」(値は「東日本」「北日本」「西日本」)を登録している場合のグループ化規則を提供。

9.4 グループ化規則の有効化

9.3 章で作成したグループ化規則を有効化します。

有効化すると、CLM はグループ化規則に設定しているグループ化条件に基づいて、CLM に登録されているデバイスのグルーピングを行い、グループ化規則に設定している鍵の更新条件に基づいて鍵の自動更新を開始します。

グループ化規則の有効化手順については、9.4 章をご覧ください。

9.5 エッジ/デバイスのデバイス ID 登録・設定

グループ化規則を有効化後、エッジ/デバイスを CLM に登録・CLA の設定を行い、CLM の管理下におきます。エッジ/デバイスの状態により、登録・設定手順が異なります。

9.5.1 エッジ/デバイスに、CLA がインストールされていない場合

1. CLM(WebUI)または ID 鍵コマンドでデバイス ID 登録を行い、デバイス ID を取得します。
   デバイス ID 登録の手順については、別紙「EDMS オプション利用の手引き」をご覧ください。
2. エッジ/デバイスに、CLA をインストールします。
   CLA インストール時に、1 で入手したデバイス ID と、CLM への接続情報を指定してインストールします。
   詳細は、別紙「SecureWare/Credential Lifecycle Agent セットアップカード」をご覧ください。

9.5.2 エッジ/デバイスに、CLA がインストール済みである場合

9.5.2.1対象エッジ/デバイスがエッジゲートウェイ(Linux 版)である場合

エッジゲートウェイ上の CLA のバージョンを確認します。「V1.1.0 (CLM V1.1)」より古いバージョンである場合、CLA をバージョンアップし、CLM への接続情報を設定します。

詳細は、開発元にお問い合わせください。

9.5.2.2対象エッジ/デバイスがエッジゲートウェイ(Linux 版)ではない場合

1. エッジ/デバイスが CLM に登録されているか(デバイス ID が発行されているか)を確認します。
   デバイスが CLM に登録されているかは、次の方法で確認可能です。詳細は、別紙「EDMS オプション利用の手引き」をご覧ください。
   - CLM(WebUI): メニュー [デバイス一覧]、[デバイス鍵情報取得]
   - ID 鍵コマンド: デバイス ID 情報検索コマンド

2. エッジ/デバイスが CLM に登録されていない場合は、CLM(WebUI)または ID 鍵コマンドでデバイス ID 登録を行い、デバイス ID を取得します。
   デバイス ID 登録の手順については、別紙「EDMS オプション利用の手引き」をご覧ください。

3. エッジ/デバイス上の CLA のバージョンを確認します。「V1.1.0 (CLM V1.1)」より古いバージョンである場合は、CLA をバージョンアップします。
   CLA のバージョンは、次のコマンドで確認可能です。バージョンアップ手順については、開発元にお問い合わせください。
   [Linux]
   # /opt/nec/pf/swcagent/bin/SWCLMCLIENT –version
   [Windows]
   > SWCLMCLIENT.cmd –version

4. CLA にデバイス ID と CLM への接続情報を設定します。
   - デバイス ID
     次のファイルが存在するか確認します。存在しない場合、ファイルを作成し、デバイス ID を記載して上書き保存します。
     [Linux]
     /opt/nec/pf/swcagent/bin/deviceid
     [Windows]
     %ProgramFiles%¥NEC¥swcagent¥deviceid
   - CLM への接続情報
     次のファイルをテキストエディタで開き、接続情報を期して上書き保存します。
     [Linux]
     /etc/swcagent/swcagentd.conf
     [Windows]
     c:¥swclm¥conf¥swcagent¥swcagentd.conf
     [設定内容]
     以下の 2 行を記載します。

host=<CLM サーバ IP アドレスまたは FQDN> port=<CLM 待ち受けポート番号> 例) host=127.0.0.1 port=18443

5. CLA(daemon)を再起動します。

9.6 運用状況の確認(統計情報の参照)

CLM では、エッジ/デバイス上の鍵更新状況を統計情報として定期的に収集しています。収集した情報は、CLM(WebUI)でエッジ/デバイスのグループ単位にグラフ化・リスト化して表示・確認することが可能です。

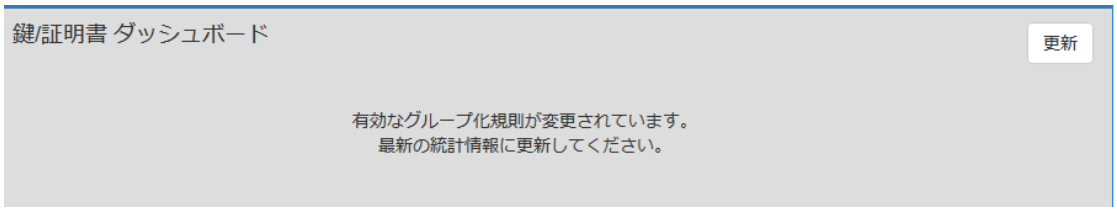
また、グラフ化・リスト化された情報から、リモート実行(リモート実行による鍵・デバイス管理 9.7 章)で鍵を管理することも可能です。

1. WebUI にログインしていない場合は、WebUI へログインします。
   ログイン手順は、5.3.2 章をご覧ください。

2. メニューの[鍵/証明書管理]-[鍵/証明書 ダッシュボード]をクリックします。

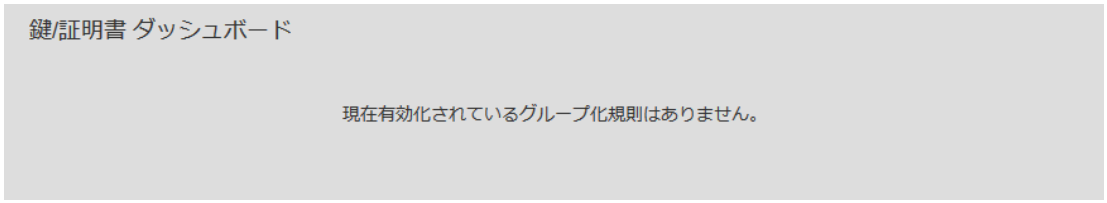
3. グループ化規則が有効になっている場合、「鍵/証明書 ダッシュボード」画面が表示されます。

次のメッセージが表示された場合は、グループ化規則は有効になっていますが、統計情報が古いためグラフ・リストの表示ができない状態です。画面右上の「更新」ボタンをクリックし、統計情報を更新します。更新が完了すると、統計情報のグラフ・リストが表示されます。

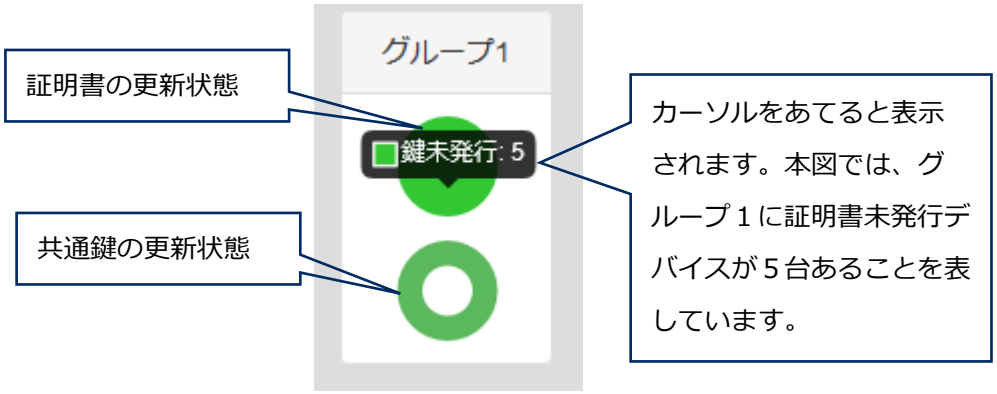


次のメッセージが表示された場合は、グループ化規則が有効になっていません。グループ化規則を

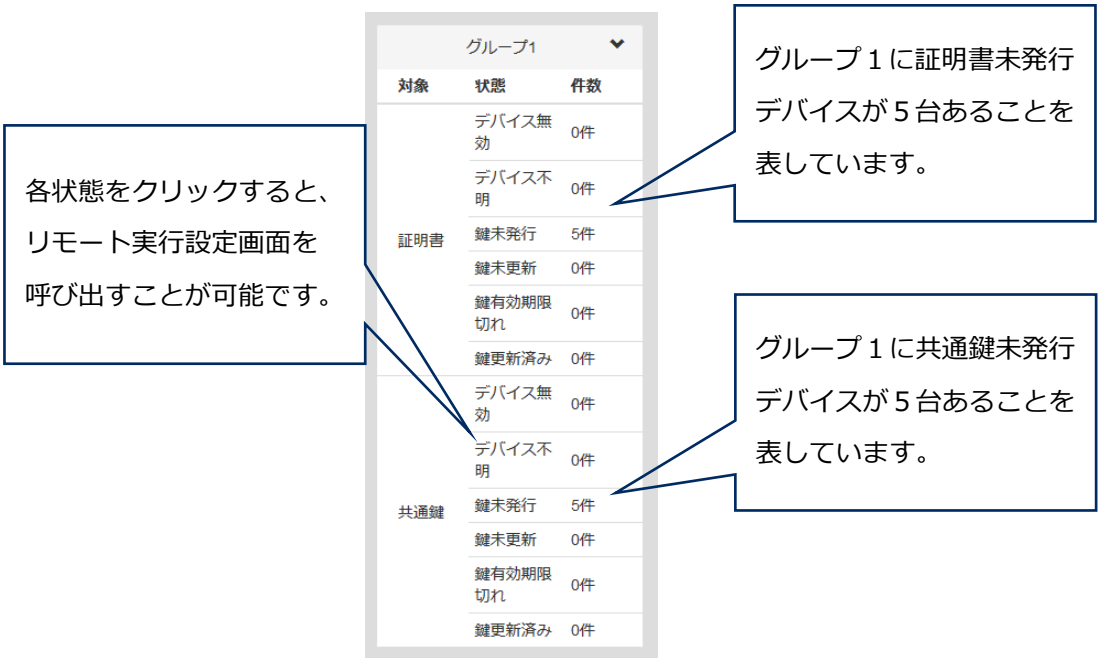
有効化してください。



4. ダッシュボード上のグラフ(画面中央～右部)は、グループ化規則のグループ単位に鍵の更新状況を表示したものです。上部は証明書、下部は共通鍵の更新状態を表しており、各状態を色分けして表示します。グラフの各色にカーソルをあてると、Tip が表示され、各状態のエッジ/デバイス数が確認可能です。



5. ダッシュボード上のリスト(画面左部)は、グループ化規則のグループ単位に鍵の更新状況を表示したものです。グラフをクリック、またはリスト上のグループ名部分をクリックすると、リストが展開されます。リスト内の状態部分をクリックすると、クリックした状態下にあるエッジ/デバイスに対してリモート実行行うことが可能です。



グラフ・リストに表示される状態は、次の通りです。

状態	説明(証明書)	説明(共通鍵)
デバイス無効	デバイスの状態が無効化されているエッジ/デバイスの数。	
デバイス不明	エッジ/デバイス自動検出･登録(11 章)で自動登録したエッジ/デバイスの数。	
鍵未発行	証明書未発行、または保有する証明書がすべて失効済のエッジ/デバイスの数。	共通鍵未発行または紐づいている共通鍵がすべて削除済みのエッジ/デバイスの数。
鍵未更新	鍵更新間隔以内に更新されていない証明書を保有するエッジ/デバイスの数。	鍵更新間隔以内に更新されていない共通鍵を保有するエッジ/デバイスの数。
鍵有効期限きれ	鍵更新間隔以内に更新したが、有効期限きれとなってしまった証明書を保有するエッジ/デバイスの数。	鍵更新間隔以内に更新したが、有効期限きれとなってしまった共通鍵を保有するエッジ/デバイスの数。
鍵更新済み	鍵更新間隔以内に更新し、有効期限内の証明書を保有するエッジ/デバイスの数。	鍵更新間隔以内に更新したが、有効期限内の共通鍵を保有するエッジ/デバイスの数。

6. グラフ・リストから、各グループの鍵の更新状況を確認します。
   必要に応じて、デバイス管理機能やリモート実行(リモート実行による鍵・デバイス管理 9.7 章)を使用して鍵の更新やエッジ/デバイスの状態変更等を行います。

9.7 リモート実行による鍵・デバイス管理

定期更新に失敗した鍵を再更新したいなど、定期更新とは別に手動で鍵管理を行いたい場合があります。

また、グループに新規に登録した複数エッジ/デバイスに鍵を発行したい場合や、グループ内の特定デバイスをまとめて無効化したい場合など、グループ単位での鍵・デバイス管理を行いたい場合があります。

このような場合、「リモート実行」機能を使用することで、グループ単位に任意のタイミングで、手動による鍵・デバイス管理を行うことができます。

9.7.1 リモート実行とは

リモート実行とは、グループに所属するデバイスに対して鍵管理に関する操作を手動で実行要求する機能です。グループに所属するエッジ/デバイスに対してまとめて実行要求することが可能です。また、複数の操作を登録することが可能ですが、操作の実際の実行は CLA(daemon)のポーリング間隔に従い、1 操作ずつの実行となります。

指定可能な操作は、鍵の状態により異なります。各状態により選択可能な操作は、次の通りです。

状態	選択可能な操作	説明
デバイス無効	デバイス有効化	デバイスの状態を有効化する
	操作クリア	リモート実行をキャンセルする(9.7.4 章)
デバイス不明	デバイス有効化	デバイスの状態を有効化する
	デバイス無効化	デバイスの状態を無効化する
	操作クリア	リモート実行をキャンセルする(9.7.4 章)
鍵未発行	鍵発行	鍵(証明書・共通鍵)を発行する
	デバイス無効化	デバイスの状態を無効化する
	操作クリア	リモート実行をキャンセルする(9.7.4 章)
鍵未更新	鍵更新	鍵(証明書・共通鍵)を更新する
	デバイス無効化	デバイスの状態を無効化する
	操作クリア	リモート実行をキャンセルする(9.7.4 章)
鍵有効期限きれ	鍵更新	鍵(証明書・共通鍵)を更新する
	デバイス無効化	デバイスの状態を無効化する
	操作クリア	リモート実行をキャンセルする(9.7.4 章)
鍵更新済み	鍵更新	鍵(証明書・共通鍵)を更新する
	デバイス無効化	デバイスの状態を無効化する
	操作クリア	リモート実行をキャンセルする(9.7.4 章)

9.7.2 鍵/証明書 ダッシュボードからのリモート実行

1. [鍵/証明書管理]-[鍵/証明書 ダッシュボード]を表示します。
2. 操作対象グループのリストを展開し、操作対象の状態をクリックします。

3. ダイアログが表示されます。
   デバイスの一覧から、操作したいエッジ/デバイスを選択します。その後、ダイアログ下部の操作一覧より実行したい操作を選択し、「実行」ボタンをクリックします。

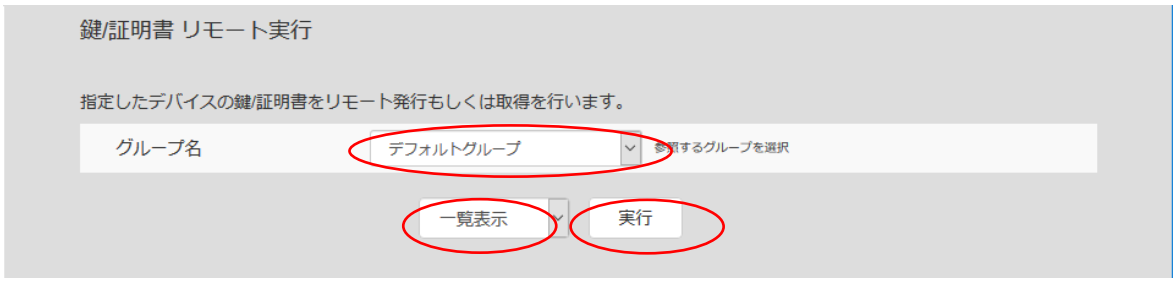
4. 操作の登録に成功すると、メッセージが表示されます。
   エラーが発生した場合は、メッセージに従い原因を取り除き、再度操作を選択して「実行」ボタンをクリックしてください。

9.7.3 鍵/証明書 リモート実行からのリモート実行

1. [鍵/証明書管理]-[鍵/証明書 リモート実行]をクリックします。

2. 「グループ名」のプルダウンリストから、操作対象グループを選択します。

3. 「鍵/証明書 リモート実行」画面が表示されます。
   グループ名のリストから、操作対象のグループを選択します。その後、画面下部のリストから「一覧表示」を選択し、「実行」ボタンをクリックします。



4. グループに所属するデバイスの一覧が表示されます。
   デバイスの一覧から、操作したいエッジ/デバイスを選択します。その後、ダイアログ下部の操作一覧より実行したい操作を選択し、「実行」ボタンをクリックします。

5. 操作の登録に成功すると、メッセージが表示されます。
   エラーが発生した場合は、メッセージに従い原因を取り除き、再度操作を選択して「実行」ボタンをクリックしてください。

9.7.4 リモート実行のキャンセル

実行したい操作を誤った場合など、登録したい操作をキャンセルしたい場合は、次の手順でキャンセルを行います。

1. [鍵/証明書管理]-[鍵/証明書 ダッシュボード]や[鍵/証明書管理]-[鍵/証明書 リモート実行]画面から、操作をキャンセルしたいデバイスを選択します。
2. 操作一覧から「操作クリア」または「クリア」を選択し、「実行」ボタンをクリックします。
3. 操作の登録に成功すると、メッセージが表示されます。

エラーが発生した場合は、メッセージに従い原因を取り除き、再度操作を選択して「実行」ボタンをクリックしてください。

9.8 エッジ/デバイス上の鍵の利用方法

鍵自動更新やリモート実行で発行・更新・取得した鍵は、エッジ/デバイス上の鍵管理ストアに保管しています。

エッジ/デバイス上のアプリケーションから鍵を使用したい場合は、SWCLMKEYCP コマンドを使用して鍵管理ストア内の鍵を任意の場所にコピーし、コピーした鍵をアプリケーションから使用します。

SWCLMKEYCP コマンドの使用方法は、次の通りです。

[SWCLMKEYCP コマンド（Linux)]

格納パス	/opt/nec/pf/swclm/SWCLM/bin/SWCLMKEYCP.sh
使用方法	SWCLMKEYCP.sh –k <key> -d <target directory>
引数	-k <key>　：コピーする鍵のファイル名。次のいずれかを指定 clcacert.pem：CA 証明書(PEM 形式) clcert.pem　：クライアント証明書(PEM 形式) clcert.der　：クライアント証明書(DER 形式) clcert.p12　：クライアント証明書(PKCS#12 形式) clkey.pem　：クライアント証明書秘密鍵 cmkey　：共通鍵 -d <target directory>　：コピー先ディレクトリの絶対パス
仕様	引数に指定した鍵を、鍵管理ストアから指定ディレクトリへコピーします。
留意事項	・本コマンドは、root ユーザで実行します。 ・コピーした鍵の owner、group は「root:root」です。 ・コピーした鍵のアクセス権は、「744」です。 ・コピーした鍵は、利用するアプリケーションの仕様に合わせ、リネームや owner・goroup、アクセス権の変更を実施してください。

[SWCLMKEYCP コマンド（Windows)]

格納パス	c:¥swclm¥SWCLM¥bin¥SWCLMKEYCP.bat
使用方法	SWCLMKEYCP.bat [/F] <key> <target directory>
引数	/F　：<target directory> に同名の鍵がある場合、上書き <key>　：コピーする鍵のファイル名。次のいずれかを指定 clcacert.pem：CA 証明書(PEM 形式) clcert.pem　：クライアント証明書(PEM 形式)

	clcert.der ： クライアント証明書(DER 形式) clcert.p12 ： クライアント証明書(PKCS#12 形式) clkey.pem ： クライアント証明書秘密鍵 cmkey ： 共通鍵 <target directory> ： コピー先ディレクトリの絶対パス
仕様	引数に指定した鍵を、鍵管理ストアから指定ディレクトリへコピーします。
留意事項	・本コマンドは、管理者権限で実行します。 ・コピーした鍵のアクセス権は、次の通りです。 Administrators グループ: フルコントロール SYSTEM ： フルコントロール コマンド実行者 ： フルコントロール ・コピーした鍵は、利用するアプリケーションの仕様に合わせ、リネームやアクセス権の変更を実施してください。

9.9 グループ化規則の作成・更新・削除

本章では、グループ化規則の作成・更新・削除手順について説明します。

9.9.1 グループ化規則の作成

グループ化規則を新規に作成する場合の手順について説明します。

サンプルとして登録されているグループ化規則の更新や、作成したグループ化規則の更新をする場合の手順については、9.9.2 章をご覧ください。

1. WebUI にログインしていない場合は、WebUI へログインします。
   ログイン手順は、5.3.2 章をご覧ください。

2. メニューの[鍵/証明書管理]-[鍵/証明書 グループ化規則編集]をクリックします。

3. 「鍵/証明書 グループ化規則編集」画面が表示されます。

画面下部の「新規作成」をクリックします。

4. 「新規グループ化規則 作成」ダイアログが表示されます。必要事項を入力し、「作成」ボタンをクリックします。

表 9-3 新規グループ化規則 入力項目

項目	説明
グループ化規則名	グループ化規則の名称。
詳細情報	グループ化規則の説明。

5. 作成に成功すると、画面下部「新規作成」の上に作成したグループ化規則が表示されます。

6. 作成したグループ化規則をクリックすると、グループ化条件と鍵の更新に関する設定を入力する画面が表示されます。必要事項を入力し、「保存」ボタンをクリックします。
   入力内容については、9.9.2 章をご覧ください。

7. 保存に成功すると、メッセージが表示されます。
   エラーが発生した場合は、メッセージに従い原因を取り除き、再度「保存」ボタンをクリックしてください。

9.9.2 グループ化規則の更新

グループ化規則を更新する場合の手順について説明します。

更新するグループ化規則は、無効化状態である必要があります。有効化されている場合は、9.10 章の手順に従い無効化を実施後、本手順で更新してください。

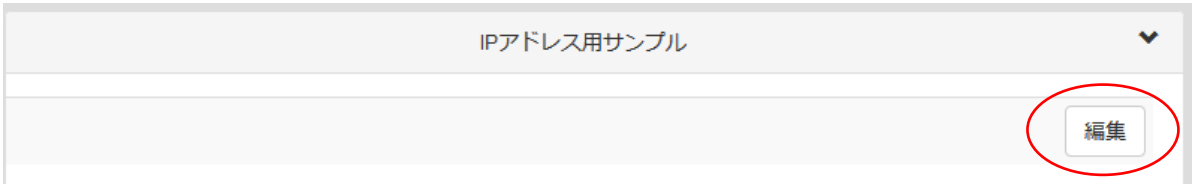
1. WebUI にログインしていない場合は、WebUI へログインします。
   ログイン手順は、5.3.2 章をご覧ください。

2. メニューの[鍵/証明書管理]-[鍵/証明書 グループ化規則編集]をクリックします。

8. 「鍵/証明書 グループ化規則編集」画面が表示されます。更新したいグループ化規則をクリックします。

3. グループ化条件と鍵の更新に関する設定を入力する画面が表示されます。

4. グループ化規則名や詳細情報を更新する場合は、画面右上の「編集」ボタンをクリックします。



「グループ化規則概要情報 編集」ダイアログが表示されます。

必要事項を入力し、「保存」ボタンをクリックします。

表 9-4 グループ化規則概要情報 編集 入力項目

項目	説明
グループ化規則名	グループ化規則の名称。

詳細情報	グループ化規則の説明。

5. グループ化条件を更新します。

必要事項を入力します。①のボタンをクリックすると、グループを削除することができます。また、②のボタンをクリックすると、グループを追加することができます。

グループの行を上下にドラッグ＆ドロップすることで、グループの優先度を入れ替えることが可能です。エッジ/デバイスが複数の条件にマッチする場合、優先度の高いグループに所属します。

表 9-5 グループ化条件 入力項目

項目	説明
条件式検索キー	グループ化条件に使用するデバイスの項目名。 指定可能な項目名は、9.2.2 章を参照。
グループ名	グループの名称。
条件式	グループに所属するエッジ/デバイスを絞り込むための条件。 本項目に指定した値と完全一致するエッジ/デバイスを検索し、グルーピングする。 ただし、条件内に「*」または「?」を使用した場合は、部分一致でエッジ/デバイスを検索し、グルーピングする。 「*」を指定した場合、0 文字以上の任意文字とみなす。また「？」を指定した場合は 1 文字以上の任意文字とみなす。 「*」、「?」を文字として使用したい場合は、「¥」でエスケープして指定すること。 例)

	・完全一致 iotgatewaytest 192.0.2.0 2001:db8:1234:5678:90ab:cdef:: 00:00:5E:00:53:00 ・部分検索 *gateway* gateway?? 10.*.*.*（10.*も可能） 2001:db8:*:*:*:*:*:*（2001:db8:*も可能） 00:00:5E:*:*:*（00:00:5E:*も可能）
証明書 更新間隔	証明書の自動更新間隔を指定。 1 日、7 日、30 日、90 日、180 日、365 日、更新しない のいずれかを選択。
共通鍵 更新間隔	共通鍵の自動更新間隔を指定。 1 日、7 日、30 日、90 日、180 日、365 日、更新しない のいずれかを選択。
ポーリング間隔	CLA(daemon)が CLM に問い合わせを行う間隔を指定。 CLA(daemon) が起動して CLM へのアクセスに成功後、CLA(daemon)は本項目に指定した間隔で CLM に問い合わせを行う。 単位は、「分」。既定値は、1440 分。

6. 各グループの鍵情報を更新します。
   グループの行の右端の「編集」ボタンをクリックします。

7. 「鍵情報 編集」ダイアログが表示されます。
   各項目を更新し、「設定」ボタンをクリックします。

表 9-6 鍵情報 編集 入力項目

項目	説明
共通鍵の形式	エッジ/デバイスに払い出す共通鍵の形式。 「TWINE」または「AES」を選択。
共通鍵長	共通鍵の鍵長。単位は bit。 「80」または「128」を選択。 共通鍵形式が「TWINE」であり、共通鍵長に「128」を指定した場合、当該グループに所属するエッジ/デバイス上には軽量暗号 開発キットがインストールされている必要がある。
更新方法	共通鍵の更新方法。 9.2.1 章で管理方法 A を選択した場合は「共通鍵取得」を選択。 管理方法 B を選択した場合は、「共通鍵更新」を選択。
エイリアス	共通鍵に付与する Alias。 任意の文字列を指定すること。 「%devicename%」と指定した場合、Alias 名としてエッジ/デバイ

	スのデバイス名を使用する。
証明書の形式	エッジ/デバイスに払い出す証明書の形式。 以下のいずれかを選択。 pem: pem 形式の証明書 der：der 形式の証明書 p12：PKCS#12 形式の証明書(chain なし)
発行モード	証明書の発行モード。 以下のいずれかを選択。 クライアント証明書: クライアント証明書を発行する サーバ証明書: サーバ証明書を発行する CA 証明書: CLM から CA 証明書を取得する
PKCS#12 のパスワード	PKCS#12 形式証明書のパスフレーズ。 証明書を PKCS#12 形式に変換する際のパスフレーズに使用する。 「%deviceid%」を指定した場合、エッジ/デバイスのデバイス ID をパスフレーズとして使用する。
コモンネーム 組織名 部門名 市区町村名 都道府県名 国別コード	証明書のサブジェクト。 コモンネーム以外は省略可能。 置換文字列として以下を指定可能。 %devicename%：エッジ/デバイスのデバイス名 %hostname%：エッジ/デバイスのホスト名

8. 更新が完了したら、「保存」ボタンをクリックします。

9.9.3 グループ化規則の削除

グループ化規則を削除する場合の手順について説明します。

削除するグループ化規則は、無効化状態である必要があります。有効化されている場合は、9.10 章の手順に従い無効化を実施後、本手順で削除してください。

1. WebUI にログインしていない場合は、WebUI へログインします。
   ログイン手順は、5.3.2 章をご覧ください。

2. メニューの[鍵/証明書管理]-[鍵/証明書 グループ化規則編集]をクリックします。

3. 「鍵/証明書 グループ化規則編集」画面が表示されます。削除対象のグループ化規則をクリックします。

4. 「削除」ボタンをクリックします。

5. 確認ダイアログが表示されます。「削除」をクリックします。

6. 削除に成功すると、メッセージが表示されます。
   エラーが発生した場合は、メッセージに従い原因を取り除き、再度「削除」ボタンをクリックしてください。

9.10 グループ化規則の有効化・無効化

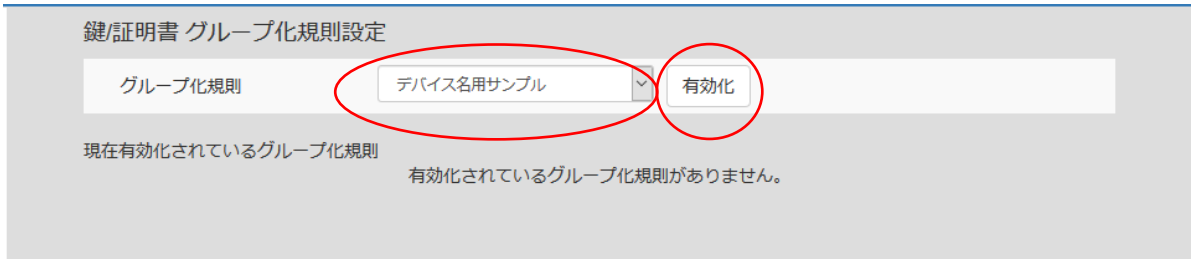
本章では、グループ化規則の有効化・無効化手順について説明します。

9.10.1 グループ化規則の有効化

1. WebUI にログインしていない場合は、WebUI へログインします。
   ログイン手順は、5.3.2 章をご覧ください。

2. メニューの[鍵/証明書管理]-[鍵/証明書 グループ化規則設定]をクリックします。

3. 「鍵/証明書 グループ化規則設定」画面が表示されます。「グループ化規則」のプルダウンリストから有効化したいグループ化規則を選択し、「有効化」ボタンをクリックします。



4. 有効化に成功すると、「現在有効化されているグループ化規則」欄に、有効化したグループ化規則の情報が表示されます。

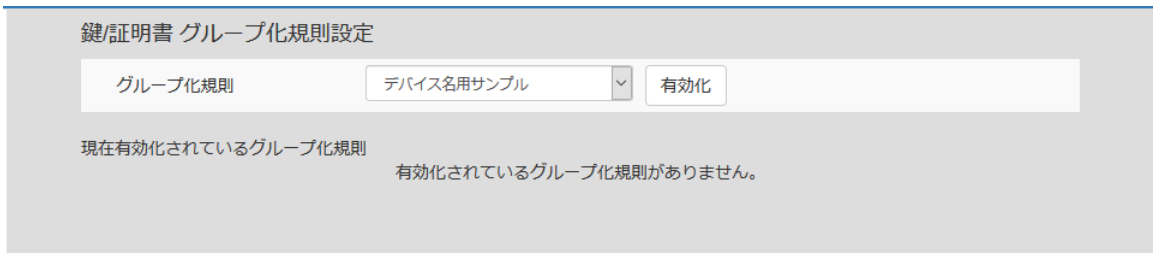
9.10.2 グループ化規則の無効化

1. WebUI にログインしていない場合は、WebUI へログインします。
   ログイン手順は、5.3.2 章をご覧ください。

2. メニューの[鍵/証明書管理]-[鍵/証明書 グループ化規則設定]をクリックします。

3. 「鍵/証明書 グループ化規則設定」画面が表示され、「現在有効化されているグループ化規則」欄に、現在有効化されているグループ化規則の情報が表示されます。
   画面下部の「無効化」ボタンをクリックします。

4. 無効化に成功すると、「現在有効化されているグループ化規則」欄に、「有効化されているグループ化規則がありません」というメッセージが表示されます。

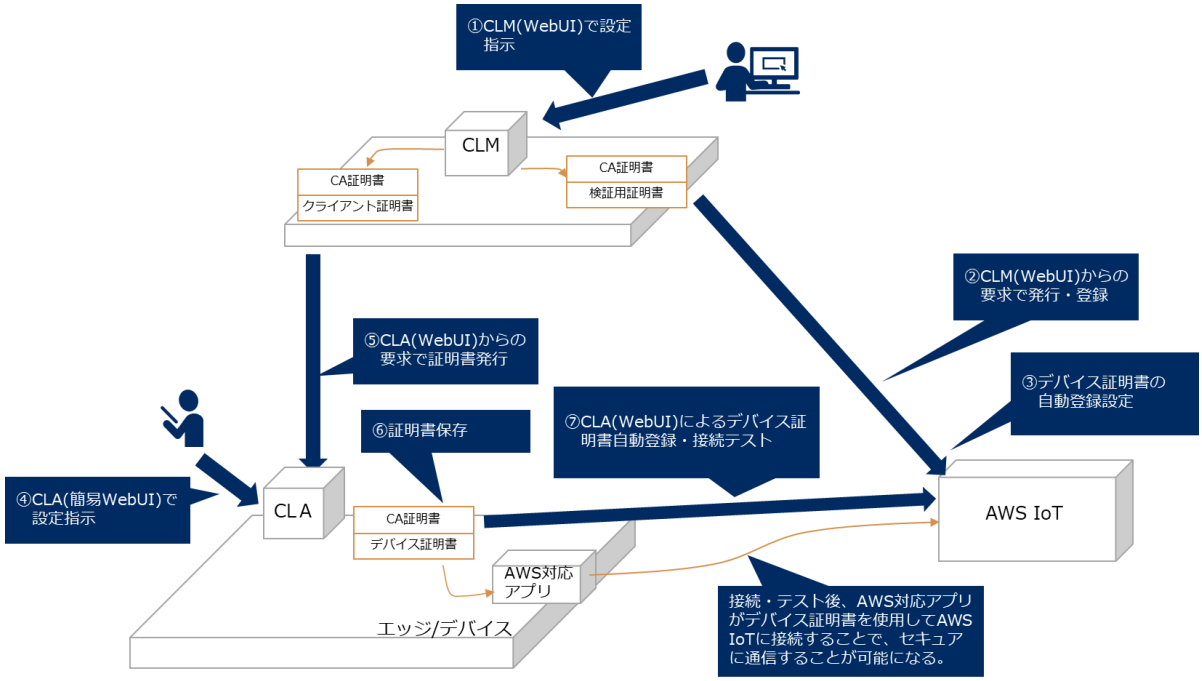


10 AWS IoT 接続設定

エッジ/デバイスがクライアント証明書(デバイス証明書)を使用して AWS IoT に接続するには、AWS IoT への設定や CA 証明書・デバイス証明書登録、エッジ/デバイスへのデバイス証明書インストールなどを行う必要があり、IoT システム構築に煩雑な手作業を伴います。

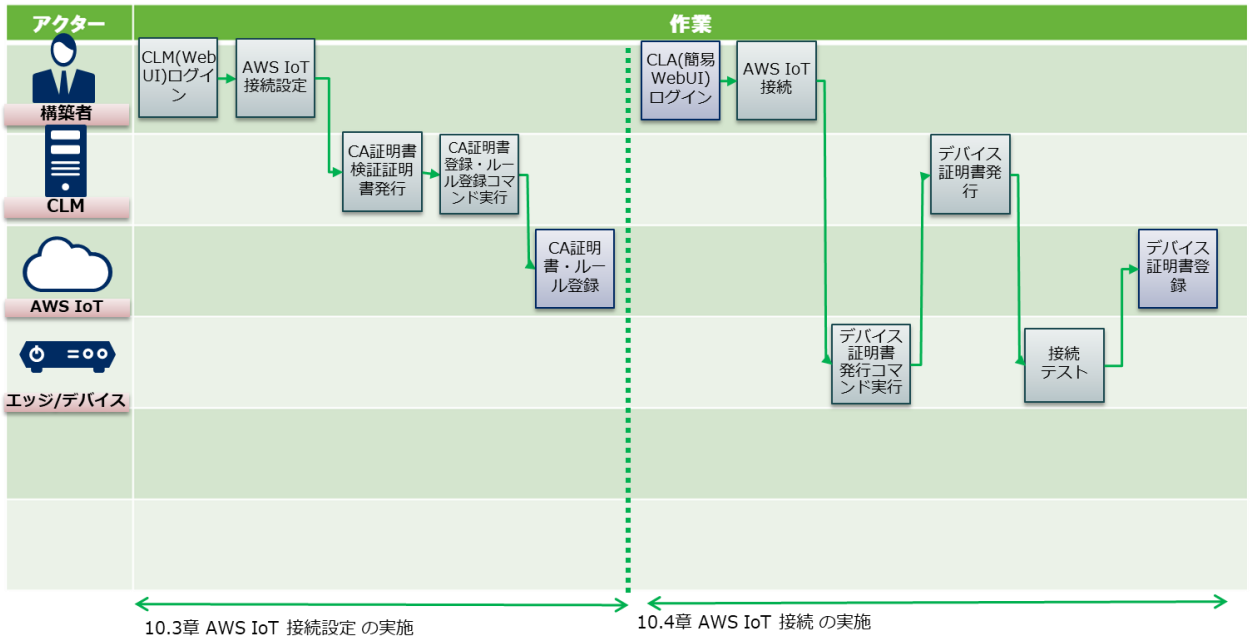
CLM では、IoT システム構築を容易にするため、エッジ/デバイスがデバイス証明書を使用して AWS IoT へ接続するために必要となる設定をワンクリックで自動設定する機能と、クライアント証明書(デバイス証明書)を発行し、発行した証明書で AWS IoT へ接続できるか検証する機能を提供しています。

本機能を使用することで、煩雑な手作業を行うことなく、IoT システムを構築することができます。発行したデバイス証明書は、エッジ/デバイス上のアプリケーションと AWS IoT 間の SSL/TLS の相互認証・暗号化通信で使用可能です。



10.1 設定の流れ

AWS IoT 接続設定の流れ(フロー)は、次の通りです。フローの詳細は、10.3 章、10.4 章で説明します。



10.2　事前準備

AWS IoT 接続設定を行うには、事前に以下の情報入手・設定を行う必要があります。

10.2.1　CLM サーバでの事前準備

本準備は、10.3 章の初回実施前に行います。

- CLM サーバへの AWSCLI インストール

  本機能では、AWSCLI を使用します。

  [Linux]

  1. python3 がインストールされていない場合、python3 をインストールします。

     # yum install python3

  2. pip がインストールされていない場合、pip をインストールします。

     # curl -O https://bootstrap.pypa.io/get-pip.py

     # python3 get-pip.py

  3. AWS CLI をインストールします。

     # pip3 install awscli --upgrade

  4. AWS CLI がインストールされたことを確認します。

     バージョン情報が表示されれば、AWS CLI がインストールされています。

     # aws --version

  [Windows]

  製品媒体内の /package/AWSCLI/AWSCLI64PY3.msi を実行し、C:¥Program Files¥Amazon¥AWSCLI にインストールします。

- AWS アクセスキーID、シークレットアクセスキーの入手

AWS IoT 接続設定を行う際、AWS のアクセスキーID、シークレットアクセスキーを使用します。
AWS アカウントから、アクセスキーID とシークレットアクセスキーを入手します。

- 接続設定を行うリージョンの AWS IoT エンドポイント
  CLA(簡易 WebUI)でエッジ/デバイスの AWS IoT 接続設定を行う際に使用します。
  AWS 管理コンソールから接続設定を行うリージョンの AWS IoT の「設定」を確認することで入手可能です。
- CLM(WebUI)・CLA(簡易 WebUI) 初回アクセス設定
  CLM(WebUI)、CLA(簡易 WebUI)へアクセスする端末・ブラウザに設定を行います。設定は、初回アクセス前に行います。詳細は 5.3.1 章、6.3.1 章をご覧ください。

10.2.2 エッジ/デバイスでの事前準備

本準備は、10.4 章の初回実施前に行います。

- CLM(WebUI)による AWS IoT 接続設定
  エッジ/デバイス側での接続設定を行う前に、CLM(WebUI)を使用して AWS IoT の接続設定を完了しておく必要があります。
- 接続設定を行うリージョンの AWS IoT エンドポイント
  CLA(簡易 WebUI)でエッジ/デバイスの AWS IoT 接続設定を行う際に使用します。
  AWS 管理コンソールから接続設定を行うリージョンの AWS IoT の「設定」を確認することで入手可能です。
- CLM(WebUI)・CLA(簡易 WebUI) 初回アクセス設定
  CLM(WebUI)、CLA(簡易 WebUI)へアクセスする端末・ブラウザに設定を行います。設定は、初回アクセス前に行います。詳細は 5.3.1 章、6.3.1 章をご覧ください。

10.3 AWS IoT 接続設定 の実施

本章では、CLM による、AWS IoT への接続設定を行う手順について説明します。

1. WebUI にログインしていない場合は、WebUI へログインします。
   ログイン手順は、5.3.2 章をご覧ください。

2. クラウド接続設定を行っていない場合は、接続設定を行います。設定については 5.3.3 章をご覧ください。

3. メニューの「AWS 証明書自動登録」をクリックします。

4. 「AWS 証明書自動登録」画面が表示されます。必要事項を入力し、「開始」ボタンをクリックします

表 10-1 AWS 証明書自動登録 入力項目

項目	説明
AWS アクセスキー	10.2.1 章で入手した AWS アクセスキーを指定します。
AWS シークレットアクセスキー	10.2.1 章で入手した AWS シークレットアクセスキーを指定します。
AWS リージョン	AWS IoT 接続設定を行いたいリージョンを選択します。
CA 名	エッジ/デバイスが AWS IoT に接続する際に使用するデバイス証明書を発行する CA の名称を指定します。 「ca1」固定です。

5. 「開始」ボタンをクリックすると、AWS IoT への接続設定が自動で行われます。
設定に成功すると、以下のメッセージが表示されます。

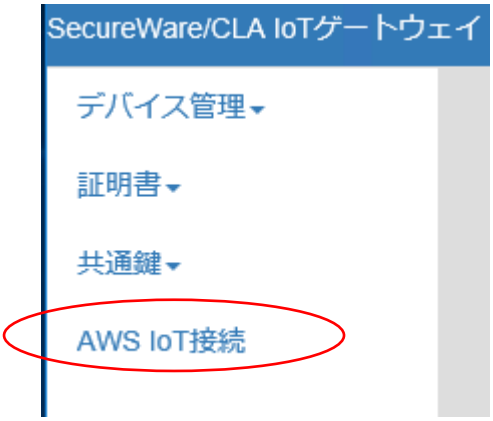
エラー発生時は、CLM(WebUI)および WebUI ログにメッセージが出力されます。メッセージに従い、エラー原因を取り除いてください。その後、「クリア」ボタンをクリックしてください。「クリア」ボタンをクリックすることで、接続設定が巻き戻され、接続設定の再実行が可能になります。

10.4　AWS IoT 接続 の実施

本章では、CLA(簡易 WebUI)による、AWS IoT への接続設定を行う手順について説明します。

1. 簡易 WebUI にログインしていない場合は、簡易 WebUI へログインします。
   ログイン手順は、6.3.2 章をご覧ください。

2. メニューの「AWS IoT 接続」をクリックします。



3. 「AWS IoT 接続」画面が表示されます。必要事項を入力し、「開始」ボタンをクリックします

表 10-2 AWS 証明書自動登録 入力項目

項目	説明
ファイル保存先	CLM で発行したデバイス証明書を保存するディレクトリを絶対パスで指定します。 ディレクトリは、エッジ/デバイス上のディレクトリを指定します。 最大文字列長は、1009 – (「共通鍵ファイル名」の文字列長) byte です。 使用可能文字種は、ASCII 0x21～0x7E(但し、「;」「|」「&」「`」「(」「)」「$」「<」「>」「*」「?」「{」「}」「[」「]」「!」を除く)です。
AWS エンドポイント	10.2.2 章で入手した AWS IoT エンドポイントを指定します。
デバイス名	どの機器(デバイス・エッジ・サーバなど)からデバイス証明書を発行したか CLM が判別するために指定します。 また、CLM で発行するデバイス証明書の subject の CN 値に設定されるとともに、AWS IoT との相互認証にも使用します。 キッティング済みエッジ GW では、既にデバイスは CLM にエッジ ID で登録されていますので、エッジ ID を指定します。 最大文字列長は、64byte です。 使用可能文字種は、英数記号(但し、「;」「|」「&」「`」「(」「)」「$」「<」「>」「*」「?」「{」「}」「[」「]」「!」を除く)です。
CA 名	デバイス証明書を発行する CA の名称を指定します。 「ca1」固定です。

4. 「開始」ボタンをクリックすると、デバイス証明書の発行と、AWS IoT への接続テストが自動で行われます。
   設定に成功すると、以下のメッセージが表示されます。

エラー発生時は、CLA(簡易 WebUI)および WebUI ログにメッセージが出力されます。メッセージに従

い、エラー原因を取り除いてください。その後、再度「開始」ボタンをクリックして接続設定の再実行を行ってください。

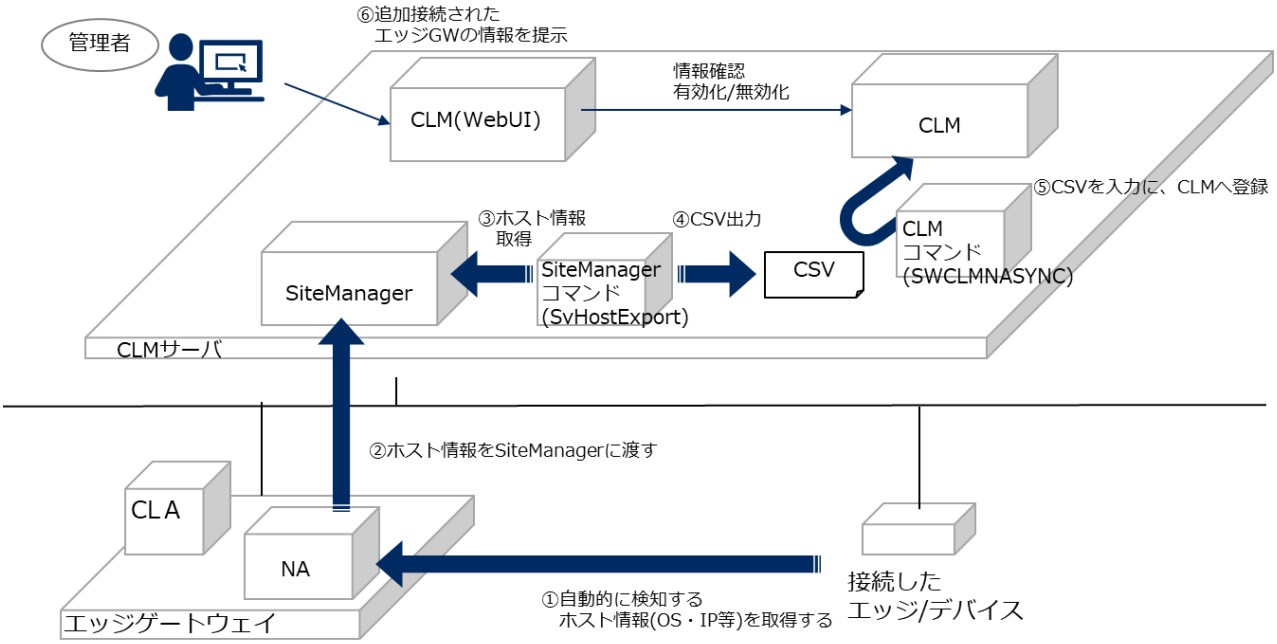
5. 以上で、エッジ/デバイスから、エッジ/デバイス上のデバイス証明書を使用した AWS IoT への接続確認が完了しました。エッジ/デバイス上のアプリケーションから、エッジ/デバイス上のデバイス証明書を使用して AWS IoT へアクセスするように設定することで、エッジ/デバイス上のアプリケーションと AWS IoT 間の SSL/TLS の相互認証・暗号化通信が可能になります。

11 エッジ/デバイス自動検出・登録

CLM は、エッジゲートウェイにインストールした InfoCage 不正接続防止のエージェントである NetworkAgent(以降、NA と称します)と、そのマネージャである SiteManager、コマンド(SvHostExport.exe)と連携し、エッジゲートウェイ上の NA が監視しているネットワークセグメント上に接続されたエッジ/デバイスを NA で自動検出し、CLM へデバイス情報を自動登録します。

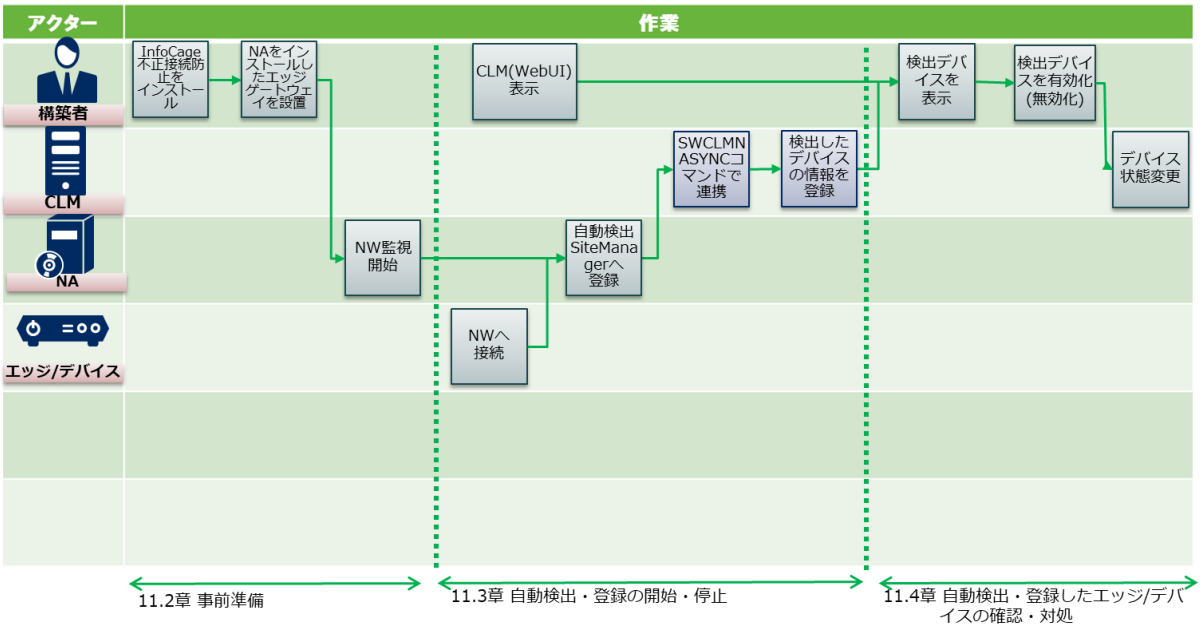
管理者は、NA が検出したエッジ/デバイスの情報を確認し、エッジ/デバイスの状態を有効化して ID 鍵の管理を行うことや、状態を無効化して ID 鍵の管理をできないようにすることが可能です。

また、CLM から検出したエッジ/デバイスへ ssh が可能な場合は、検出したエッジ/デバイスの状態を有効化する際に、エッジ/デバイスへ CLA などのソフトウェアをインストールすることも可能です。



11.1 操作の流れ

エッジ/デバイス自動検出の操作の流れ(作業フロー)は、次の通りです。



11.2　事前準備

自動検出・登録を始めるにあたり、以下の設定を行います。

Linux

- Windows 環境への InfoCage 不正接続防止(SiteManager・コマンド)インストール

  Linux 版 CLM を使用する場合、InfoCage 不正接続防止(SiteManager・コマンド)をインストールする環境を別途用意する必要があります。Windows 環境を用意し、InfoCage 不正接続防止のマニュアルを参照して SiteManager とコマンド(NATools)をインストール・設定を行います。

  また、Linux 版 CLM から後述のコマンド(SvHostExport.exe)でエクスポートした CSV ファイルが参照できるように Windows 環境を設定します。

- エッジゲートウェイへの InfoCage 不正接続防止(NA)インストール

  InfoCage のマニュアルを参照し、エッジゲートウェイへ NA をインストール・設定を行います。

- InfoCage 不正接続防止の動作確認

  NA で監視しているネットワークセグメントにデバイスを接続し、NA でデバイスが検知できることを確認します。また、検出したデバイスに関する情報を、コマンド(SvHostExport.exe)を使用してエクスポートできることを確認します。

  [コマンド]

  コマンド(NATools)インストールサーバ上で実行します。

  > %SVTOOLS インストール PATH%¥SvHostExport.exe -pw %NA_PWD% -f %NA_CSV_FILE% -s localhost -p %SM_PORT% -a %NA_NAME%

  (引数)

  NA_PWD: NA のパスワード

  NA_CSV_FILE: 情報を出力する CSV ファイルのファイル名

SM_PORT: SiteManager が使用しているポート番号

NA_NAME: NA のエージェント名

- SWCLMNASYNC コマンドのタスク化

/opt/nec/swclm/SWCLM/bin に、NA で検出したエッジ/デバイスの情報を CLM へ連携(登録)するためのスクリプト SWCLMNASYNC.sh を格納しています。

SWCLMNASYNC.sh をテキストエディタで開き、ご使用の環境に合わせて設定を変更します。その後、SWCLMNASYNC.sh が一定間隔かつ管理者権限で実行するように cron にタスクとして登録します。

[SWCLMNASYNC.sh の設定]

以下の設定を、環境に合わせて変更します。

・12 行目 NA_NAME ：NA のエージェント名を記載します。

[タスクに登録するコマンドライン]

> /opt/nec/swclm/SWCLM/bin/SWCLMNASYNC.sh -f %CSV_PATH%

(引数)

CSV_PATH: NA で検出したエッジ/デバイスの情報を格納する CSV ファイルの絶対パス。

SvHostExport.exe でエクスポートした、エッジ/デバイスの情報を保存している CSV ファイルを指定します。

Windows

- CLM サーバへの InfoCage 不正接続防止(SiteManager・コマンド)インストール

InfoCage 不正接続防止のマニュアルを参照し、SiteManager とコマンド(NATools)をインストール・設定を行います。

- エッジゲートウェイへの InfoCage 不正接続防止(NA)インストール

InfoCage のマニュアルを参照し、エッジゲートウェイへ NA をインストール・設定を行います。

- InfoCage 不正接続防止の動作確認

NA で監視しているネットワークセグメントにデバイスを接続し、NA でデバイスが検知できることを確認します。また、検出したデバイスに関する情報を、コマンド(SvHostExport.exe)を使用してエクスポートできることを確認します。

[コマンド]

CLM インストールサーバ上で実行します。

> %SVTOOLS インストール PATH%¥SvHostExport.exe -pw %NA_PWD% -f %NA_CSV_FILE% -s localhost -p %SM_PORT% -a %NA_NAME%

(引数)

NA_PWD: NA のパスワード

NA_CSV_FILE: 情報を出力する CSV ファイルのファイル名

SM_PORT: SiteManager が使用しているポート番号

NA_NAME: NA のエージェント名

- SWCLMNASYNC コマンドのタスク化

c:¥swclm¥SWCLM¥bin に、NA で検出したエッジ/デバイスの情報を CLM へ連携(登録)するためのスクリプト SWCLMNASYNC.bat を格納しています。

SWCLMNASYNC.bat をテキストエディタで開き、ご使用の環境に合わせて設定を変更します。その後、SWCLMNASYNC.batが一定間隔かつ管理者権限で実行するようにWindowsのタスクスケジューラにタスクとして登録します。

[SWCLMNASYNC.bat の設定]

以下の設定を、環境に合わせて変更します。

・7 行目 SVTOOLS_HOME：SvTools のインストール PATH を記載します。

・9 行目 SM_PORT ：SiteManager の使用するポート番号を記載します。

・10 行目 NA_PWD ：NA のパスワードを記載します。

・11 行目 NA_NAME ：NA のエージェント名を記載します。

[タスクに登録するコマンドライン]

> c:¥swclm¥SWCLM¥bin¥SWCLMNASYNC.bat -f %CSV_PATH%

(引数)

CSV_PATH: NA で検出したエッジ/デバイスの情報を格納する CSV ファイルの絶対パス。
SvHostExport.exe でエクスポートした、エッジ/デバイスの情報を保存している CSV ファイルを指定します。

11.3 自動検出・登録の開始・停止

- 開始

1. NA をインストールしたエッジゲートウェイを監視対象ネットワークに接続し、ネットワーク監視を開始してエッジ/デバイスの自動検出を開始します。
2. CLM サーバ上で設定したタスクの定期実行を開始し、CLM へのエッジ/デバイス自動登録を開始します。

- 停止

1. CLM サーバ上で設定したタスクの定期実行を停止し、CLM へのエッジ/デバイス自動登録を停止します。
2. NA のネットワーク監視を停止し、エッジ/デバイスの自動検出を停止します。

11.4　自動検出・登録したエッジ/デバイスの確認・対処

NA により自動検出されたエッジ/デバイスは、状態が不明(Idle)なデバイスとして登録されます。
以下の流れで確認を行い、検出したエッジ/デバイスに合わせて対処を行います。

1. CLM(WebUI)の次の画面からエッジ/デバイスの詳細情報を確認します。
   - [デバイス管理] – [デバイス一覧]
   - [鍵/証明書管理] – [鍵/証明書 ダッシュボード] 、[鍵/証明書 リモート実行]

   NA により自動検出されたエッジ/デバイスの情報は、次のように登録されています。

項目	登録値
デバイス名	検出したエッジ/デバイスの MAC アドレス
状態	不明 (Idle)
IP アドレス	検出したエッジ/デバイスの IP アドレス
MAC アドレス	検出したエッジ/デバイスの MAC アドレス
OS 名	検出したエッジ/デバイスの OS 種別 (※)
ホスト名	検出したエッジ/デバイスのホスト名 (※)
コンピュータ名	検出したエッジ/デバイスのコンピュータ名 (※)

   (※) 環境により、登録されない場合があります。

   [デバイス一覧]の操作方法については、別紙「EDMS オプション利用の手引き」をご覧ください。
   [鍵/証明書 ダッシュボード] 、[鍵/証明書 リモート実行]の操作方法については、9 章をご覧ください。

2. 検出したエッジ/デバイスの詳細情報を確認し、状態変更を行います。
   - 既知のエッジ/デバイスであり、CLM で ID 鍵の管理を行いたいエッジ/デバイスである場合は、検出したエッジ/デバイスの状態を有効化します。
     エッジ/デバイスの状態の有効化は、次の画面から行うことができます。
     - [デバイス管理] – [デバイス有効化]
     - [鍵/証明書管理] – [鍵/証明書 ダッシュボード]、[鍵/証明書 リモート実行]

     [デバイス有効化]の詳細は、別紙「EDMS オプション利用の手引き」をご覧ください。
     [鍵/証明書 ダッシュボード] 、[鍵/証明書 リモート実行]の詳細は、9 章をご覧ください。

   - 検出したエッジ/デバイスが、未知のエッジ/デバイスなど不正なエッジ/デバイスや、CLM での管理が不要なエッジ/デバイスである場合は、検出したエッジ/デバイスを削除します。
     エッジ/デバイスの削除は、次の画面から行うことができます。
     - [デバイス管理] – [デバイス削除]

・ [鍵/証明書管理] – [鍵/証明書 ダッシュボード]、[鍵/証明書 リモート実行]

[デバイス削除]の詳細は、別紙「EDMS オプション利用の手引き」をご覧ください。
[鍵/証明書 ダッシュボード] 、[鍵/証明書 リモート実行]の詳細は、9 章をご覧ください。

➢ 検出したエッジ/デバイスの対処を保留したい場合は、状態「Idle」のままとします。

3. 状態を有効化したエッジ/デバイスの情報は、手動で更新することが可能です。
必要に応じ、次の画面から情報を更新します。
・ [デバイス管理] – [デバイス更新]

[デバイス更新]の詳細は、別紙「EDMS オプション利用の手引き」をご覧ください。

12 付録

12.1 ログメッセージ一覧

CLM は error.log に以下のエラーメッセージを出力します。

例外コード	ログメッセージ	説明
10000001	SWCLMServlet.init() ServletContext get error	コンテキストパスが null。
10000301	SWCLMServlet.authenticate() Parameter invalid	引数不正（db、securitykey が null)
10000302	SWCLMServlet.authenticate() Key unauthorized	鍵が不一致
10000303	SWCLMServlet.authenticate() Key status error	キーID のステータスが有効でない
10000304	SWCLMServlet.authenticate() Key expired	キーID が有効期限切れ
10000305	SWCLMServlet.authenticate() KeyID not found	キーID が見つからない
10000306	SWCLMServlet.authenticate() Key authentication error	システムエラーが発生しました。
10000401	SWCLMServlet.getTenantID() Key data get error	キーID が見つからない
10000402	SWCLMServlet.getTenantID() Key data is not one	キーID が DB 中に複数存在する
10010001	IDInfo.doPost() Body data parse error	ボディ部(JSON)の構文エラー
10010002	IDInfo.doPost() URL parse error	クエリストリングの構文エラー
10010003	IDInfo.doPost() Bad mode	mode 不正
10010101	IDInfo.doGet() URL parse error	クエリストリングの構文エラー
10010102	IDInfo.doGet() Get parameter parse error	クエリストリングで指定されたパラメータチェックでエラー
10010201	IDInfo.checkParam() mode parameter not set	mode 指定なし

10010202	IDInfo.checkParam() Create parameter parse error	IDパスワード発行要求の構文エラー
10010203	IDInfo.checkParam() Verify parameter parse error	IDパスワード照合要求の構文エラー
10010204	IDInfo.checkParam() Delete parameter parse error	IDパスワード削除要求の構文エラー
10010205	IDInfo.checkParam() Bad mode	mode 不正
10010501	IDInfo.doPost() Delete key information error	キーID 削除処理でエラー
10020001	CredInfo.doPost() Body data parse error	ボディ部(JSON)の構文エラー
10020002	CredInfo.doPost() URL parse error	クエリストリングの構文エラー
10020003	CredInfo.doPost() Bad mode	mode 不正
10020101	CredInfo.checkParam() mode parameter not set	mode 指定なし
10020102	CredInfo.checkParam() Certificate create parameter parse error	証明書発行要求の構文エラー
10020103	CredInfo.checkParam() Certificate get parameter parse error	証明書取得要求の構文エラー
10020104	CredInfo.checkParam() Certificate update parameter parse error	証明書更新要求の構文エラー
10020105	CredInfo.checkParam() Certificate revoke parameter parse error	証明書失効要求の構文エラー
10020106	CredInfo.checkParam() Certificate notice parameter parse error	証明書状態通知要求の構文エラー
10020107	CredInfo.checkParam() Bad mode	mode 不正
10020201	CredInfo.createCert() Certificate create device ID invalid	デバイス ID 不正
10020301	CredInfo.getCert() Certificate get device ID invalid	デバイス ID 不正
10020401	CredInfo.updateCert() Certificate update device ID invalid	デバイス ID 不正
10020501	CredInfo.revokeCert() Certificate revoke device ID invalid	デバイス ID 不正
10021101	CredInfo.getKeyType() Invalid	システムエラーが発生

	argument	
10021102	CredInfo.getKeyType() keytypename or keyname not found	鍵種別/鍵名称が見つからない
10030001	KeyInfo.doPost() Body data parse error	ボディ部(JSON)の構文エラー
10030002	KeyInfo.doPost() URL parse error	クエリストリングの構文エラー
10030003	KeyInfo.doPost() Bad mode	mode 不正
10030101	KeyInfo.checkParam() mode parameter not set	mode 指定なし
10030102	KeyInfo.checkParam() CommonKey create parameter parse error	共通鍵発行要求の構文エラー
10030103	KeyInfo.checkParam() CommonKey get parameter parse error	共通鍵取得要求の構文エラー
10030104	KeyInfo.checkParam() CommonKey update parameter parse error	共通鍵更新要求の構文エラー
10030105	KeyInfo.checkParam() CommonKey delete parameter parse error	共通鍵削除要求の構文エラー
10030106	KeyInfo.checkParam() CommonKey notice parameter parse error	共通鍵状態通知要求の構文エラー
10030107	KeyInfo.checkParam() Bad mode	mode 不正
10030108	KeyInfo.checkParam() CommonKey verify parameter parse error	共通鍵照合要求の構文エラー
10030109	KeyInfo.checkParam() CommonKey getlist parameter parse error	共通鍵一括取得要求の構文エラー
10030201	KeyInfo.createKey() CommonKey create device ID invalid	デバイス ID 不正
10030301	KeyInfo.getKey() CommonKey get device ID invalid	デバイス ID 不正
10030302	KeyInfo.getKey() CommonKey data not found	共通鍵が見つからない
10030303	KeyInfo.getKey() CommonKey data is not one	共通鍵が DB 中に複数存在する
10030401	KeyInfo.updateKey() CommonKey update device ID invalid	デバイス ID 不正
10030501	KeyInfo.revokeKey() CommonKey	デバイス ID 不正

	delete device ID invalid	
10030502	KeyInfo.revokeKey() CommonKey delete error	共通鍵削除でエラー発生
10031001	KeyInfo.validateKey() CommonKey verify device ID invalid	デバイス ID 不正
10031002	KeyInfo.validateKey() Key unauthorized	共通鍵認証失敗
10031003	KeyInfo.validateKey() Key status error	共通鍵のステータスが有効でない
10031004	KeyInfo.validateKey() Key expired	共通鍵が有効期限切れ
10031005	KeyInfo.validateKey() Key not found	共通鍵が見つからない
10031006	KeyInfo.validateKey() Key authentication error	共通鍵照合でエラー発生
10031101	KeyInfo.getList() parameter is null	引数不正
10031102	KeyInfo.getList() device[要素番号] is null	デバイス情報が null
10031103	KeyInfo.getList() deviceid and devicename are unmatch.: deviceid=[デバイス ID] devicename=[デバイス名]	デバイス ID とデバイス名が不一致
10031104	KeyInfo.getList() device is already deleted: deviceid=[デバイス ID] devicename=[デバイス名]	デバイスは削除状態
11031105	KeyInfo.getList() device and key link not found: deviceid=[デバイス ID] keynumber=[鍵番号]	デバイスと共通鍵の紐付け情報なし
11031106	KeyInfo.getList() commonkey data not found: keynumber=[鍵番号]	共通鍵が見つからない
11031107	KeyInfo.getList() commonkey data is not one: keynumber=[鍵番号]	指定された鍵番号の共通鍵が DB 上に複数存在する
11031108	KeyInfo.getList() commonkey status is not valid: kyunumber={鍵番号]	共通鍵のステータスが Valid 以外
11031109	KeyInfo.getList() commonkey is expired: keynumber=[鍵番号]	共通鍵が有効期限切れ
10031201	KeyInfo.saveRequestJsonString() invalid parameter	引数不正

10031202	KeyInfo.saveRequestJsonString() optioninfo is null	optioninfo が null
10031203	KeyInfo.saveRequestJsonString() directory not exists. path=[履歴ファイル出力ディレクトリ]	履歴ファイル出力ディレクトリが存在しない
10031204	KeyInfo.saveRequestJsonString() SecurityException: {診断メッセージ}	履歴ファイル出力ディレクトリの確認でエラー
10031205	KeyInfo.saveRequestJsonString() ostype is unknown	OS 種別が取得できない
10040001	DataInfo.doGet() URL parse error	クエリストリングの構文エラー
10040002	DataInfo.doGet() Get parameter parse error	テーブル名が null、または、テーブル名が空文字列
10040003	DataInfo.doGet() Table name invalid	テーブル名が不正
10050001	DataBase connect error	DB 接続失敗
10060001	DeviceInfo.doPost() Body data parse error	ボディ部(JSON)の構文エラー
10060002	DeviceInfo.doPost() URL parse error	クエリストリングの構文エラー
10060003	DeviceInfo.doPost() Bad mode	mode 不正
10060101	DeviceInfo.checkParam() mode parameter not set	mode 指定なし
10060102	DeviceInfo.checkParam() Device register parameter parse error	デバイス登録要求の構文エラー
10060103	DeviceInfo.checkParam() Device auth parameter parse error	デバイス認証要求の構文エラー
10060104	DeviceInfo.checkParam() Bad mode	mode 不正
10060105	DeviceInfo.checkParam() Device batchregister parameter parse error	デバイス一括登録要求の構文エラー
10060106	DeviceInfo.checkParam() Device batchinvalidate parameter parse error	デバイス一括無効化要求の構文エラー
10060107	DeviceInfo.checkParam() Device batchvalidate parameter parse error	デバイス一括有効化要求の構文エラー
10060108	DeviceInfo.checkParam() Device batchupdate parameter parse error	デバイス一括更新要求の構文エラー
10060109	DeviceInfo.checkParam() Device	デバイス一括論理削除要求の

	batchdelete parameter parse error	構文エラー
10060110	DeviceInfo.checkParam() Device getlist parameter parse error	デバイス情報取得要求の構文エラー
10060111	DeviceInfo.checkParam() Device getkeylist parameter parse error	デバイス鍵情報取得要求の構文エラー
10060112	DeviceInfo.checkParam() Device search parameter parse error	デバイス検索要求の構文エラー
10060201	DeviceInfo.registerDevice() Create device Name invalid	デバイス名不正
10060301	DeviceInfo.authDevice() DeviceKey unauthorized	デバイスキー不一致
10060302	DeviceInfo.authDevice() DeviceKey status error	デバイスのステータスが有効でない
10060303	DeviceInfo.authDevice() DeviceKey expired	デバイスが有効期限切れ
10060304	DeviceInfo.authDevice() Device not found	デバイスが見つからない
10060305	DeviceInfo.authDevice() DeviceKey authentication error	照合処理での上記以外のエラー
10060601	DeviceInfo.batchCommon() parameter is null	引数不正
10060602	DeviceInfo.batchCommon() mode invalid	mode 不正
10060603	DeviceInfo.batchCommon() device[要素番号] is null	デバイス情報が null
10060604	DeviceInfo.batchCommon() CertRevoke result is invalid	デバイス一括論理削除処理の戻り情報が不正
10060701	DeviceInfo.batchRegister() parameter is null	引数不正
10060702	DeviceInfo.batchRegister() device Name invalid	デバイス名不正
10060801	DeviceInfo.batchInvalidate() parameter is null	引数不正
10060901	DeviceInfo.batchValidate() parameter is null	引数不正
10061001	DeviceInfo.batchUpdate() parameter	引数不正

	is null	
10061101	DeviceInfo.batchDelete() parameter is null	引数不正
10061201	DeviceInfo.getDeviceList() parameter is null	引数不正
10061202	DeviceInfo.getDeviceList() attr not set	情報取得対象の属性名なし
10061203	DeviceInfo.getDeviceList() search result is null	デバイス検索結果が null
10061301	DeviceInfo.getDeviceKeyList() parameter is null	引数不正
10061302	DeviceInfo.getDeviceKeyList() device[要素番号] is null	デバイス情報が null
10061303	DeviceManager.getDeviceKeyList() device is already deleted: デバイスID/デバイス番号	デバイスは削除状態
10061401	DeviceInfo.searchDevice() parameter is null	引数不正
10061402	DeviceInfo.searchDevice() condition not set	検索条件なし
10061403	DeviceInfo.searchDevice() operator invalid	検索条件中の比較演算子が不正
10061404	DeviceInfo.searchDevice() search result is null	デバイス検索結果が null
10061405	DeviceInfo.searchDevice() search result device is null	検索結果中のデバイス情報が null
10061406	DeviceInfo.searchDevice() search clmgroupnumber but no active rule	有効なルールがない状態で、グループ番号による検索を行おうとした
10061501	DeviceInfo.getConditionForGetList() parameter is null	引数不正
10061502	DeviceInfo.getConditionForGetList() search mode invalid	検索モードが不正
10061601	DeviceInfo.getAttrResult() parameter is null	引数不正
10061602	DeviceInfo.getAttrResult() search	検索結果中のデバイス情報が

	result device is null	null
10061701	DeviceInfo.toKeyInfoData() NumberFormat error	変換処理でエラー発生
10061801	DeviceInfo.deleteClmGroupNumber() parameter is null	引数不正
10070001	GroupInfo.doPost() Body data parse error	ボディ部(JSON)の構文エラー
10070002	GroupInfo.doPost() URL parse error	クエリストリングの構文エラー
10070003	GroupInfo.doPost() Bad mode	mode 不正
10070004	GroupInfo.doPost() API busy	API のロック中
10070101	GroupInfo.checkParam() mode parameter not set	mode 指定なし
10070102	GroupInfo.checkParam() Bad mode	mode 不正
10070103	GroupInfo.checkParam() Rule Create parameter parse error	グループ化規則名登録の構文エラー
10070104	GroupInfo.checkParam() Rule Get parameter parse error	グループ化規則名取得の構文エラー
10070105	GroupInfo.checkParam() Rule Update parameter parse error	グループ化規則名変更の構文エラー
10070106	GroupInfo.checkParam() Group Get parameter parse error	グループ化規則取得の構文エラー
10070107	GroupInfo.checkParam() Groups Update parameter parse error	グループ化規則更新の構文エラー
10070108	GroupInfo.checkParam() Rule Delete parameter parse error	グループ化規則削除の構文エラー
10070109	GroupInfo.checkParam() Rule Activate parameter parse error	グループ化規則有効化の構文エラー
10070110	GroupInfo.checkParam() Rule Inactivate parameter parse error	グループ化規則無効化の構文エラー
10070201	GroupInfo.ruleCreate() parameter is null	引数不正
10070202	GroupInfo.ruleCreate() NumberFormat error	設定ファイルから取得した問合せ間隔、証明書更新間隔、共通鍵更新間隔の上限のいずれかが数字以外

10070203	GroupInfo.ruleCreate() queryinterval invalid	設定ファイルから取得した問合せ間隔が上限、もしくは下限を超えている
10070204	GroupInfo.ruleCreate() updateinterval_cert invalid	設定ファイルから取得した証明書更新間隔が上限、もしくは下限を超えている
10070205	GroupInfo.ruleCreate() updateinterval_cmkey invalid	設定ファイルから取得した共通鍵更新間隔が上限、もしくは下限を超えている
10070206	GroupInfo.ruleCreate() NumberFormat error	設定ファイルから取得した問合せ間隔、証明書更新間隔、共通鍵更新間隔のいずれかが数字以外
10070301	GroupInfo.ruleGet() parameter is null	引数不正
10070401	GroupInfo.ruleUpdate() parameter is null	引数不正
10070402	GroupInfo.ruleUpdate() target rule is not existed	対象のグループ化規則が存在しない
10070403	GroupInfo.ruleUpdate() target rule is active	対象のグループ化規則が有効状態
10070501	GroupInfo.groupGet() parameter is null	引数不正
10070601	GroupInfo.groupsUpdate() parameter is null	引数不正
10070602	GroupInfo.groupsUpdate() target rule is not existed	対象のグループ化規則が存在しない
10070603	GroupInfo.groupsUpdate() target rule is active	対象のグループ化規則が有効状態
10070701	GroupInfo.ruleDelete() parameter is null	引数不正
10070702	GroupInfo.ruleDelete() target rule is not existed	対象のグループ化規則が存在しない
10070703	GroupInfo.ruleDelete() target rule is active	対象のグループ化規則が有効状態
10070801	GroupInfo.ruleActivate() parameter is null	引数不正

10070802	GroupInfo.ruleActivate() target rule is not existed	対象のグループ化規則が存在しない
10070803	GroupInfo.ruleActivate() target rule is already active	対象のグループ化規則が有効状態
10070804	GroupInfo.ruleActivate() other rule is already active	他のグループ化規則が有効状態
10070805	GroupInfo.ruleActivate() no searchkey in target rule	条件項目未設定
10070901	GroupInfo.ruleInactivate() parameter is null	引数不正
10070902	GroupInfo.ruleInactivate() target rule is not existed	対象のグループ化規則が存在しない
10070903	GroupInfo.ruleInactivate() target rule is already inactive	対象のグループ化規則が無効状態
10071001	GroupInfo.unlockGroupMethod() Unlock error	ロック解放でエラー発生
10080001	StatisticsInfo.doPost() Body data parse error	ボディ部(JSON)の構文エラー
10080002	StatisticsInfo.doPost() URL parse error	クエリストリングの構文エラー
10080003	StatisticsInfo.doPost() Bad mode	mode 不正
10080301	StatisticsInfo.checkParam() mode parameter not set	mode 指定なし
10080302	StatisticsInfo.checkParam() Statistics get parameter parse error	統計情報取得要求の構文エラー
10080303	StatisticsInfo.checkParam() Statistics getdevice parameter parse error	統計情報デバイス一覧取得要求の構文エラー
10080304	StatisticsInfo.checkParam() Statistics forcecollect parameter parse error	統計情報強制採取要求の構文エラー
10080305	StatisticsInfo.checkParam() Bad mode	mode 不正
10090001	RemoteOperate.doPost() Body data parse error	ボディ部(JSON)の構文エラー
10090002	RemoteOperate.doPost() URL parse error	クエリストリングの構文エラー
10090003	RemoteOperate.doPost() Bad mode	mode 不正

10090101	RemoteOperate.checkParam() mode parameter not set	mode 指定なし
10090102	RemoteOperate.checkParam() Set Device Operation parameter parse error	リモート実行設定要求の構文エラー
10090103	RemoteOperate.checkParam() Query Device Operation parameter parse error	操作問合せ要求の構文エラー
10090104	RemoteOperate.checkParam() Bad mode	mode 不正
10090201	RemoteOperate.setDeviceOperation() parameter is null	引数不正
10090202	RemoteOperate.setDeviceOperation() device[要素番号] is null	デバイス情報が null
10090301	RemoteOperate.queryDeviceOperation() parameter is null	引数不正
10090501	RemoteOperate.getRamdom() NumberFormatException	DB から取得した問合せ間隔が数字以外
11000001	Authenticator.getInstance() Invalid authentication type	認証方式が不正
11010001	BasicAuth.authenticate() UserID or authinfo is null	引数の何れかが null
12000001	CertManager.getInstance() Parameter is null	引数の何れかが null
12000002	CertManager.getInstance() CA type get error	certificateAuthority の設定値を取得できない
12000003	CertManager.getInstance() Invalid ca type	certificateAuthority の設定値が不正
12000101	CertManager.getCACert() certInfo is null	引数 certInfo が null
12000102	CertManager.getCACert() DataBase is null	フィールド db が null
12000401	CertManager.getClientKey() certInfo is null	引数 certInfo が null
12000402	CertManager.getClientKey() Private Key not exist	秘密鍵ファイルが存在しない

12000501	CertManager.getCertificate() certInfo is null	引数 certInfo が null
12000502	CertManager.getCertificate() DataBase is null	フィールド db が null
12000505	CertManager.getCertificate() Key info get error	証明書情報が見つからない
12000508	CertManager.getCertificate() Certificate status invalid	明書のステータスが有効でない
12000509	CertManager.getCertificate() Certificate is expired	証明書が有効期限切れ
12000510	CertManager.getCertificate() Key info get error	DB 中に指定された証明書情報が複数存在
12000601	CertManager.updateCertificate() certInfo is null	引数 certInfo が null
12000602	CertManager.updateCertificate() DataBase is null	引数 db が null
12000603	CertManager.updateCertificate() Certificate not found	紐付け情報なし
12000607	CertManager.updateCertificate() Certificate not valid	更新後の証明書のステータスが有効以外
12000608	CertManager.updateCertificate() Certificate is expired	更新後の証明書が有効期限切れ
12000701	CertManager.deleteCertificate() certInfo is null	引数 certInfo が null
12000702	CertManager.deleteCertificate() DataBase is null	フィールド db が null
12000703	CertManager.deleteCertificate() Certificate is not linkage	紐付け情報なし
12000704	CertManager.deleteCertificate() Certificate revoke error	証明書削除でエラー発生
12000801	CertManager.insertDBClient() certInfo is null	revokeCertOnCA()の戻り値が不正
12000802	CertManager.insertDBClient() DataBase is null	引数 certInfo が null
12000901	CertManager.insertDBCA() certInfo is null	フィールド db が null

12000902	CertManager.insertDBCA() DataBase is null	引数 certInfo が null
12000903	CertManager.insertDBCA() KeyNumber NumberFormat error	フィールド db が null
12000904	CertManager.insertDBCA() CA certificate data error	既に CA 証明書登録済みかチェックしようとしたが、取得した CA 証明書情報の鍵番号が数値以外
12001001	CertManager.updateDB() DataBase is null	既に CA 証明書登録済みかチェックしようとしたが、同じ keyid に対する CA 証明書が複数存在
12001002	CertManager.updateDB() table name get fail	CA 名称に対応するテーブル名なし
12001003	CertManager.updateDB() update result is abnormal	DB 更新でエラー
12001101	CertManager.searchDB() keytypelist get error	keytypelist に CA 名称に対応する情報なし
12001102	CertManager.searchDB() keytypename not PKI	指定された CA 名称は PKI でない
12001201	CertManager.saveCert() certInfo is null	引数 certInfo が null
12001202	CertManager.saveCert() DataBase is null	引数 db が null
12001203	CertManager.saveCert() catype invalid	認証局種別が不正
12001301	CertManager.restoreCert() Parameter is null	引数のいずれかが null
12002101	CertManager.identifyCertificate() keytypelist get error	keytypelist に CA 名称に対応する情報なし
12002102	CertManager.identifyCertificate() keytypename not PKI	指定された CA 名称は PKI でない
12002103	CertManager.identifyCertificate() Key info get error	DB 中に指定された証明書情報が存在しない
12002104	CertManager.identifyCertificate() Key info get error	DB 中に指定された証明書情報が複数存在

12002901	CertManager.createClientCert() parameter is null	引数のいずれかが null
12002902	CertManager.createClientCert() Clientcertpath get error	clientcertpath 設定値の取得に失敗
12003201	CertManager.getCertType() Cert type invalid	証明書種別の組み合わせ不正。CA 証明書、かつ、P12(設定ファイルでの指定値)
12003202	CertManager.getCertType() Cert type invalid	証明書種別の組み合わせ不正。CA 証明書、かつ、P12 (リクエスト中の指定値)
12003203	CertManager.getCertType() Cert type invalid	証明書種別が不正(設定ファイルでの指定値が不正)
12003204	CertManager.getCertType() Cert type invalid	証明書種別が不正(リクエスト中の指定値が不正)
12010001	CertOpenSSL.createClientCert() certInfo is null	引数 certInfo が null
12010002	CertOpenSSL.createClientCert() DataBase is null	フィールド db が null
12010003	CertOpenSSL.createClientCert() Subject is null	サブジェクトなし
12010004	CertOpenSSL.createClientCert() PassPhrase is null	P12 でパスフレーズなし
12010005	CertOpenSSL.createClientCert() OpenSSLConfpath get error	opensslconfpath 設定値の取得に失敗
12010006	CertOpenSSL.createClientCert() CAPassPhrase get error	capassphrase 設定値の取得に失敗
12010007	CertOpenSSL.createClientCert() keytypelist get error	keytypelist に CA 名称に対応する情報なし
12010008	CertOpenSSL.createClientCert() keytypename not PKI	指定された CA 名称は PKI でない
12010009	CertOpenSSL.createClientCert() ostype is unknown	OS 種別が取得できない
12010101	CertOpenSSL.getCACert() certInfo is null	引数 certInfo が null
12010102	CertOpenSSL.getCACert() DataBase is null	フィールド db が null

12010201	CertOpenSSL.getCACert() CA Cert path not exist	cacertfile 設定値の取得に失敗
12010202	CertOpenSSL.getCACert() Invalid encode type	証明書タイプが PEM/DER 以外
12010203	CertOpenSSL.getCACert() ostype is unknown	OS 種別が取得できない
12010301	CertOpenSSL.getClientCert() certInfo is null	引数 certInfo が null
12010302	CertOpenSSL.getClientCert() Certificate file not exist	"証明書ファイルが存在しない (DB からの復元も失敗) 証明書タイプが PEM の場合"
12010303	CertOpenSSL.getClientCert() Certificate file not exist	"証明書ファイルが存在しない (DB からの復元も失敗) 証明書タイプが PEM 以外の場合"
12010304	CertOpenSSL.getClientCert() Passphrase not exist	P12 でパスフレーズなし
12010305	CertOpenSSL.getClientCert() Certificate file not exist	秘密鍵ファイルが存在しない (DB からの復元も失敗)
12010306	CertOpenSSL.getClientCert() ostype is unknown	OS 種別が取得できない
12011401	CertOpenSSL.doScript() opensslpath get error	opensslpath 設定値の取得に失敗
12011402	CertOpenSSL.doScript() ostype is unknown	OS 種別が取得できない
12011901	CertOpenSSL.getCertInfo() Subject Key Identifier not found	実行したコマンドの標準出力が、Subject Key Identifier を含まない
12011902	CertOpenSSL.getCertInfo() Subject Key Identifier format invalid"	次の空白が見つからない
12011903	CertOpenSSL.getCertInfo() separator character not found	実行したコマンドの標準出力が、目的の区切り文字を含まない
12011904	CertOpenSSL.getCertInfo() ostype is unknown	OS 種別が取得できない
12012001	CertOpenSSL.revokeCert()	opensslconfpath 設定値の取

	OpenSSLConfpath get error	得に失敗
12012002	CertOpenSSL.revokeCert() CAPassPhrase get error	capassphrase 設定値の取得に失敗
12012003	CertOpenSSL.revokeCert() ostype is unknown	OS 種別が取得できない
12012401	CertOpenSSL.getKeyType() KeyType NumberFormat error	鍵番号に対応したキータイプの取得で、取得いたキータイプが数値以外
12012901	CertOpenSSL.createClientCert() Clientcertpath get error	clientcertpath 設定値の取得に失敗
12013001	CertOpenSSL.lock() Timeout	ロックタイムアウト
12013002	CertOpenSSL.lock() TryLock error"	ロック獲得失敗
12013101	CertOpenSSL.unlock() Unlock error	ロック解放失敗
13000001	CommonManager.CommonManager() DataBase is null	引数 db が null
13000201	CommonManager.getKeyTypeInfo() parameter is null	引数 keytype が 0 かつ keyname が null
13000301	CommonManager.storeKeyList() DataBase is null	データベース未接続
13000401	CommonManager.searchKeyTypeList( ) DataBase is null	データベース未接続
13000402	CommonManager.searchKeyTypeList( ) KeyType NumberFormat error	DB から取得した keytype が数字以外
13000501	CommonManager.searchKeyStatusLis t() DataBase is null	データベース未接続
13000502	CommonManager.searchKeyStatusLis t() KeyStatus NumberFormat error	DB から取得した keystatus が数字以外
13000601	CommonManager.searchStatisticsSta tusList() DataBase is null	データベース未接続
13040001	LinkManager.LinkManager() DataBase is null	DataBase が null
13040101	LinkManager.storeKeyLinkage() Invalid keyNumber	keyNumber 不正
13040102	LinkManager.storeKeyLinkage() keyID is null	keyID が null
13040103	LinkManager.storeKeyLinkage()	deviceNumber 不正

	Invalid deviceNumber	
13040104	LinkManager.storeKeyLinkage() identification is null	identification が null
13040105	LinkManager.storeKeyLinkage() Invalid issuecount	issuecount 不正
13040106	LinkManager.storeKeyLinkage() Invalid status	status 不正
13040107	LinkManager.storeKeyLinkage() DataBase is null	DataBase が null
13040201	LinkManager.updateKeyLinkage() Invalid keyNumber	keyNumber 不正
13040202	LinkManager.updateKeyLinkage() keyID is null	keyID が null
13040203	LinkManager.updateKeyLinkage() Invalid deviceNumber	deviceNumber 不正
13040204	LinkManager.updateKeyLinkage() identification is null	identification が null
13040205	LinkManager.updateKeyLinkage() DataBase is null	DataBase が null
13040206	LinkManager.updateKeyLinkage() keylinkage info invalid	事前の検索処理で条件合致レコードなし、または、複数レコードが条件合致
13040207	LinkManager.updateKeyLinkage() IssueCount NumberFormat error	issuecount 値が不正(数値でない)
13040208	LinkManager.updateKeyLinkage() udate result is abnormal	更新処理で更新レコード数が0、または、複数レコードが更新された
13040301	LinkManager.deleteKeyLinkage() Invalid keyNumber	keyNumber 不正
13040302	LinkManager.deleteKeyLinkage() keyID is null	keyID が null
13040303	LinkManager.deleteKeyLinkage() Invalid deviceNumber	deviceNumber 不正
13040304	LinkManager.deleteKeyLinkage() identification is null	identification が null
13040305	LinkManager.deleteKeyLinkage()	DataBase が null

	DataBase is null	
13040306	LinkManager.deleteKeyLinkage() No keylinkage data	事前の検索処理で条件合致レコードなし
13040307	LinkManager.deleteKeyLinkage() status NumberFormat error	status 値が不正(数値でない)
13040308	LinkManager.deleteKeyLinkage() udate result is abnormal	更新処理で更新レコード数が0、または、複数レコードが更新された
13040401	LinkManager.identifyLinkage() Invalid keyNumber	keyNumber 不正
13040402	LinkManager.identifyLinkage() keyID is null	keyID が null
13040403	LinkManager.identifyLinkage() Invalid deviceNumber	deviceNumber 不正
13040404	LinkManager.identifyLinkage() identification is null	identification が null
13040405	LinkManager.identifyLinkage() DataBase is null	DataBase が null
13040406	LinkManager.identifyLinkage() keylinkage info invalid	複数レコードが条件合致
13040501	LinkManager.checkKeyLinkageStatus() keylinkage status not set	keylinkage テーブルから status を取得できない
13040502	LinkManager.checkKeyLinkageStatus() keylinkage already update or revoke	status 値が不正(失効または更新済み)
13040503	LinkManager.checkKeyLinkageStatus() keylinkage status error	status 値が不正(数値でない)
13040601	LinkManager.deleteKeyLinkage() Invalid keyNumber	keyNumber 不正
13040602	LinkManager.deleteKeyLinkage() Invalid deviceNumber	deviceNumber 不正
13040603	LinkManager.deleteKeyLinkage() DataBase is null	DataBase が null
13040701	LinkManager.getLinkageList() Invalid deviceid	deviceID が null
13040702	LinkManager.getLinkageList() db is null	DataBase が null

13040801	LinkManager.getLastCommonkeyByAlias() keyinfo is null	引数 keyinfo が null
13040802	LinkManager.getLastCommonKeyByAlias() alias is null	引数 alias が null
13040803	LinkManager.getLastCommonKeyByAlias() db is null	DataBase が null
13040901	LinkManager.getLinkageKeySummary() devicenumber invalid	引数 devicenumber が不正
13040902	LinkManager.getLinkageKeySummary() type is null	引数 type が null
13040902	LinkManager.getLinkageKeySummary() db is null	DataBase が null
13041001	LinkManager.isLinkDeviceAndKey() devicenumber invalid	引数 devicenumber が不正
13041002	LinkManager.isLinkDeviceAndKey() keynumber invalid	引数 keynumber が不正
13041003	LinkManager.isLinkDeviceAndKey() type is null	引数 type が null
13041004	LinkManager.isLinkDeviceAndKey() db is null	DataBase が n ull
13050001	TableManager.TableManager() DataBase is null	引数 db が null
13050101	TableManager.getTableList() parameter is null	引数 tableName が null
13050102	TableManager.getTableList() NumberFormat error	status 値が不正(数値でない)
14000001	CryptAES NoSuchAlgorithmException: {例外メッセージ}	指定したアルゴリズムをサポートするプロバイダが存在しない
14000101	CryptAES.encrypt() key is null	暗号鍵が null
14000102	CryptAES.encrypt() NoSuchAlgorithmException: {例外メッセージ}	暗号化処理で NoSuchAlgorithmException 発生
14000103	CryptAES.encrypt() NoSuchPaddingException: {例外メッセージ}	暗号化処理で NoSuchPaddingException 発生

14000104	CryptAES.encrypt() InvalidKeyException: {例外メッセージ}	暗号化処理で InvalidKeyException 発生
14000105	CryptAES.encrypt() IllegalBlockSizeException: {例外メッセージ}	暗号化処理で IllegalBlockSizeException 発生
14000106	CryptAES.encrypt() BadPaddingException: {例外メッセージ}	暗号化処理で BadPaddingException 発生
14000107	CryptAES.encrypt() SecurityException: {例外メッセージ}	暗号化処理で SecurityException 発生
14000108	CryptAES.encrypt() IllegalStateException: {例外メッセージ}	暗号化処理で IllegalStateException 発生
14000109	CryptAES.encrypt() UnsupportedEncodingException: {例外メッセージ}	暗号化処理で UnsupportedEncodingException 発生
14000201	CryptAES.decrypt() key is null	暗号鍵が null
14000202	CryptAES.decrypt() NoSuchAlgorithmException: {例外メッセージ}	復号化処理で NoSuchAlgorithmException 発生
14000203	CryptAES.decrypt() NoSuchPaddingException: {例外メッセージ}	復号化処理で NoSuchPaddingException 発生
14000204	CryptAES.decrypt() InvalidKeyException: {例外メッセージ}	復号化処理で InvalidKeyException 発生
14000205	CryptAES.decrypt() IllegalBlockSizeException: {例外メッセージ}	復号化処理で IllegalBlockSizeException 発生
14000206	CryptAES.decrypt() BadPaddingException: {例外メッセージ}	復号化処理で BadPaddingException 発生
14000207	CryptAES.decrypt() ArrayIndexOutOfBoundsException: {例外メッセージ}	復号化処理で ArrayIndexOutOfBoundsException 発生

14000208	CryptAES.decrypt() NegativeArraySizeException: {例外メッセージ}	復号化処理で NegativeArraySizeException 発生
14000209	CryptAES.decrypt() RuntimeException: {例外メッセージ}	復号化処理で RuntimeException 発生
14000210	CryptAES.decrypt() UnsupportedEncodingException: {例外メッセージ}	復号化処理で UnsupportedEncodingException 発生
14000211	CryptAES.decrypt() InvalidAlgorithmParameterException: {例外メッセージ}	復号化処理で InvalidAlgorithmParameterException 発生
14010001	SecureWare.generateRandom() keysize invalid	引数 keySize が 0
14010002	SecureWare.generateRandom() return fail. errcode={SwGenerateRandom()の戻り値}	SwGenerateRandom()の戻り値が SEC_FAILURE(-1)
14010003	SecureWare.generateRandom() random size not match	生成した乱数のサイズが、引数 keySize と一致しない
14020001	TWINEOTR.TWINEOTR() parameter is null	引数が null
14020002	TWINEOTR.TWINEOTR() twoCommandPath not exists	twoCommandPath 設定値が未定義
14020101	TWINEOTR.createKeyFile() parameter is null	引数が null
14020201	TWINEOTR.createRawKey() parameter is null	引数が null
14020301	TWINEOTR.decrypt() parameter is null	引数が null
15000001	ResponseTableData.getInstance() Table name is null	引数 tableName が null
16000001	DataBase.getInstance() Invalid parameter	引数 conf、または、contextPath が null
16000002	DataBase.getInstance() Database type get error	設定ファイルにデータベースタイプの指定なし
16000003	DataBase.getInstance() Database	設定ファイルに指定された

	type not supported	データベースタイプが未サポート
16000101	DataBase.init() Invalid parameter	引数 conf、または、contextPath が null
16000102	DataBase.init() Database type get error	設定ファイルにデータベースタイプの指定なし
16000103	DataBase.init() Database type not supported	設定ファイルに指定されたデータベースタイプが未サポート

12.2　バックアップ・リストア

ディスク障害等のトラブル対応として、CLMおよび認証局を定期的にバックアップすることを推奨します。

12.2.1　バックアップ・リストアにおける注意事項

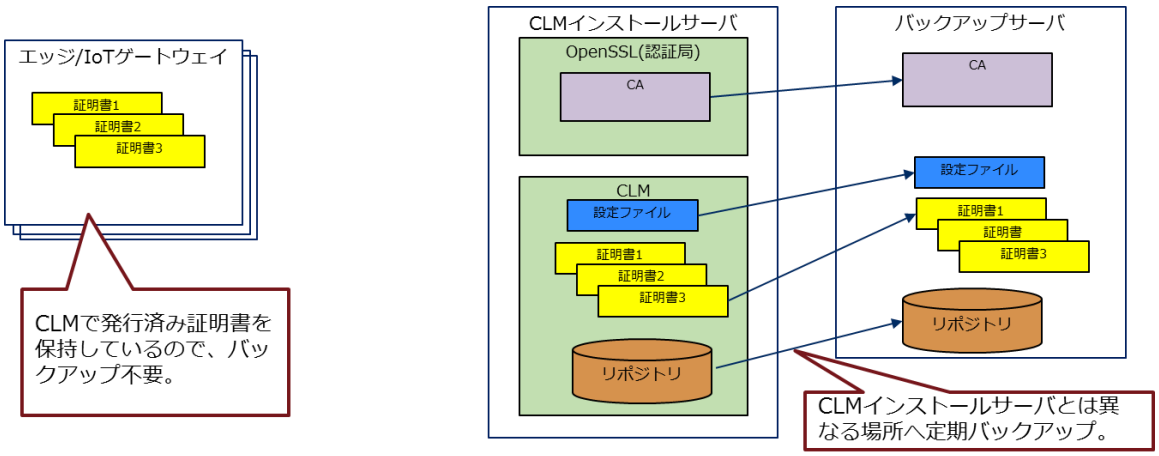
- CLM、OpenSSL(認証局)、リポジトリデータで静止点を同一にするため、バックアップ時は CLM をオフラインとし、バックアップ実行中は運用を停止する必要があります。
- 障害発生時の復旧地点は、バックアップ採取日時となります。

12.2.2　バックアップ・リストア概要

バックアップ

- CLM
  - CLM インストールサーバ、認証局の復旧に必要となる以下のデータをバックアップします。
    - CLM の設定ファイル、リポジトリデータ、発行した証明書ファイル
    - 認証局
  - バックアップは以下の方法で行い、 ID 鍵管理基盤サーバとは異なるセキュリティの確保された場所に保管します。
    - CLM の設定ファイル、発行した証明書ファイル、認証局は、ファイルまたはディレクトリを直接バックアップします。
    - リポジトリデータは、PostgreSQL が提供するコマンドを用いてバックアップします。
- エッジゲートウェイ等、CLM で管理している証明書・共通鍵を保有しているエッジ・デバイス

  証明書・共通鍵は、CLM で管理しています。よって、バックアップは不要です。



リストア

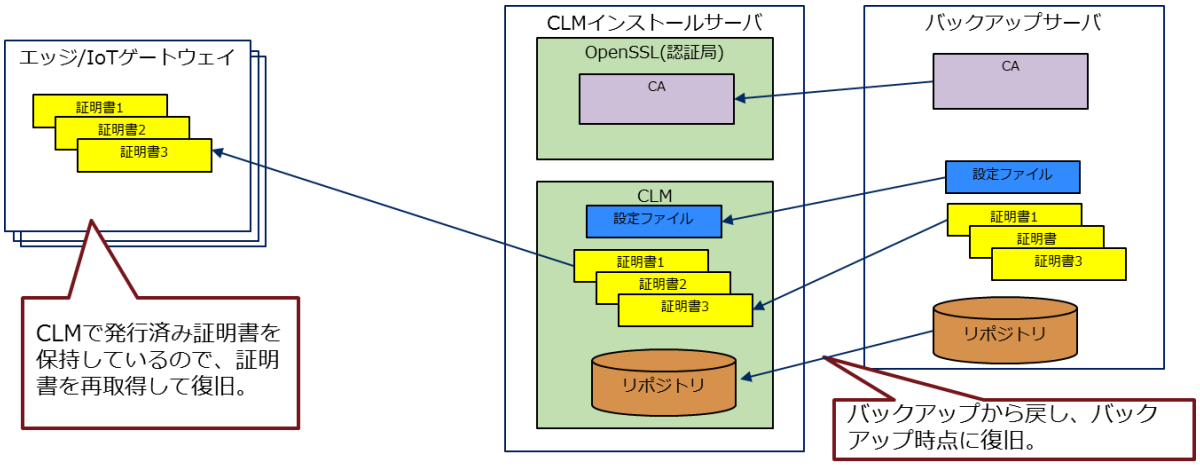
- CLM

  障害発生時は、CLM、リポジトリデータ、認証局をバックアップから戻し、 ID 鍵管理基盤サーバ、認

証局をバックアップ時点の状態に復旧します。

- エッジゲートウェイ等、CLM で管理している証明書・共通鍵を保有しているエッジ・デバイス

  エッジ/IoT ゲートウェイへ配布した証明書は、CLM で管理しています。CLM から証明書を再取得し、復旧します。



12.2.3　バックアップ手順

以下の手順で行います。本作業は、管理者権限を保有したユーザで行います。

1. CLM インストールサーバにログインします。
2. Tomcat が起動している場合は停止します。

   # /usr/local/tomcat/bin/shutdown.sh
3. 認証局をバックアップします。

バックアップ対象	**説明**
/usr/local/myopenssl/ca/openssl.cnf	OpenSSL の設定ファイル。CA 構築および CLM の運用で使用。
/usr/local/myopenssl/ca/ca1	CA を構築したディレクトリ

4. CLM をバックアップします。

バックアップ対象	**説明**
/etc/swclm	CLM 設定ファイル格納ディレクトリ
/opt/nec/pf/swclm/SWCLM/cert	CLM が発行した証明書ファイル格納ディレクトリ

5. リポジトリをバックアップします。

   データベース管理ユーザ「postgres」で、DB のバックアップを採取します。

   # su – postgres

   $ pg_dump swclm -f backup.sql
6. Tomcat を起動します。

   # /usr/local/tomcat/bin/startup.sh

12.2.4　リストア手順 (CLM)

以下の手順で行い、バックアップ地点に復旧します。本作業は、管理者権限を保有したユーザで行います。

1. サーバを構築します。
2. 認証局と CLM を再インストールします。
   ① 別紙「SecureWare/Credential Lifecycle Manager セットアップカード」に従い、CLM を再インストールします。
   ② Tomcat が起動している場合は、停止します。
   # /usr/local/tomcat/bin/shutdown.sh
3. CLM を復旧します。
   ① リポジトリのデータを復旧します。
   データベース管理ユーザ「postgres」で、DB をリストアします。
   # su – postgres
   $ psql -f backup.sql
   ② 上述 2 で構築した CLM に対して、CLM のバックアップデータを上書きします。
4. 認証局を復旧します。
   上述 2 で構築した CA に対して、認証局のバックアップデータを上書きします。
5. Tomcat を起動します。
   # /usr/local/tomcat/bin/startup.sh

12.2.5　リストア手順 (エッジ・デバイス)

エッジ・デバイスで障害が発生し、証明書・共通鍵の復旧が必要となった場合は、CLM から証明書・共通鍵を再取得します。

再取得は、以下のいずれかの方法で行います。

- WebAPI
  証明書取得 API または共通鍵発行 API で取得可能です。
  詳細は、別紙「WebAPI リファレンス」の「4 .証明書管理 API」「5.共通鍵管理 API」をご覧ください。
- WebAPI 実行コマンド
  証明書取得コマンドまたは共通鍵取得コマンドで取得可能です。
  詳細は、別紙「コマンドリファレンス」をご覧ください。
- 簡易 WebUI
  証明書取得または共通鍵取得で取得可能です。詳細は、6.3.7 章、7.3.3 章をご覧ください。

12.3　CLM が連携する認証局

本章では、CLM に同梱・連携する認証局(以降、ca1 と呼称します)の CA 証明書プロファイルと、ca1 の認証局設定を変更する場合の手順について説明します。

12.3.1　CA 証明書プロファイル

ca1 の CA 証明書プロファイルは、次の通りです。

領域名	クリティカルフラグ	設定値	説明
Version (バージョン番号)	–	2	バージョン 3 (固定) Integer 型
Serial Number (シリアル番号)	–		自動割当
Signature (署名アルゴリズム)	–		署名アルゴリズム
Algorithm identifier (アルゴリズム識別子)		1.2.840.113549.1.1.11	sha256WithRSAEncryption
Issuer (発行者)	–	cn = ***ca1*** o = ***nec*** l = ***Minato-Ku*** st = ***Tokyo*** c = JP	Subject の内容を自動的に設定
Validity (証明書正当有効期間)	–		CA 証明書の有効期間
Not Before(発行日)		(例)161201000000Z	CA 証明書有効期間の開始日。発行時の現在時間(UTC（協定世界時）) YYMMDDHHMMSSZ
Not After(終了日)		491231235959Z (2049/12/31 23:59:59)	CA 証明書有効期間の終了日。UTC（協定世界時）で指定。 YYMMDDHHMMSSZ
Subject (主体者)	–	cn = ***ca1*** o = ***nec*** l = ***Minato-Ku*** st = ***Tokyo*** c = JP	識別名（DN）を示す。 Country 属性のみ Printable String 型 それ以外は UTF8 String 型

Subject Public Key Info (主体者公開鍵情報)	–		CA 証明書の公開鍵アルゴリズム
Algorithm Identifier (アルゴリズム識別子)		1.2.840.113549.1.1.1	公開鍵アルゴリズム識別子 RSA　Encryption
Parameter （パラメータ）		NULL	
Public Key(公開鍵)		※ 公開鍵の値	証明書の公開鍵の値　鍵長は 2048bit Bit String 型。
Extensions (証明書拡張領域)			
Subject Key Identifier (主体者鍵識別子)	FALSE		主体者鍵識別子
Key Identifier		ハッシュ値	CA の公開鍵のハッシュ値
Authority Key Identifier (認証局鍵識別子)	FALSE		認証局鍵識別子
Key Identifier		ハッシュ値	発行者(CA)の公開鍵のハッシュ値。CA 証明書(自己署名証明書)の場合は自分自身。
Authority Cert Issuer		ou = xxxx o = yyyy c = JP	認証局の識別名（DN） Country 属性のみ PrintableString 型 それ以外は UTF8 String 型
Authority Cert Serial Number		(例)1	CA 証明書のシリアル番号を示す。
Key Usage (鍵用途)	TRUE	“000001100” （2 進数表記）	鍵の使用目的を指定
Digital Signature		0	「Key Cert Sign」「CRL Sign」の目的を除くデジタル署名の検証ができる。
Non Repudiation		0	否認防止用の署名検証ができる。
Key Encipherment		0	共通鍵等の鍵を暗号化できる。
Data Encipherment		0	データを直接暗号化できる。
Key Agreement		0	鍵の共有・交換ができる。

	Key Cert Sign		1	証明書の CA 署名の検証ができる。
	CRL Sign		1	CRL 署名の検証ができる。
	Encipher Only		0	交換した鍵でデータを暗号化できる。(Key Agreement がセットされている場合のみ指定可)
	Decipher Only		0	交換した鍵でデータを復号化できる。(Key Agreement がセットされている場合のみ指定可)
Basic Constraints (基本制限)		TRUE		
	CA		TRUE	CA 証明書であるか否かを示す。 (変更不要)
	Path Length Constraint		NONE(制限なし)	CA の下位 CA となることができる階層数を示す
Issuer's Signature (発行者署名)		–		証明書に付与した署名の値を示す。
	Algorithm Identifier (アルゴリズム識別子)		1.2.840.113549.1.1.11	sha256WithRSAEncryption
	Encrypted		※署名値	証明書に付与した署名の値を示す。

12.3.2　認証局の初期化

CLM が同梱・連携する認証局(CA)の初期化手順について説明します。

CA を初めて使用する場合は、本手順を実施してください。本手順を実施することで、CA が構築され、利用できるようになります。また、CA の CA 証明書について、次の設定を変更する場合も、本手順を実施します。

- CA 証明書の有効期限

初期化時の注意事項

- 本手順は、管理者権限を保有したユーザで行います。
- CA 証明書作成時に入力する秘密鍵(パスフレーズ)については、別紙「SecureWare/Credential Lifecycle Manager セットアップカード」をご覧ください。
- 本手順を実施する際は、CLM を停止してください。
- 運用中に本手順を実施して CA 証明書の有効期限を変更する場合、CA 証明書は再作成となります。再作成された CA 証明書は、有効期限の他、シリアル番号や公開鍵等が変更となりますのでご注意ください。
- 運用中に本手順を実施する場合は、本手順実施前に CA のバックアップを実施してください。

初期化手順 (Linux)

1. CLM インストールサーバにログインします。
2. 管理者権限でコマンドプロンプトを起動し、$CLM インストールディレクトリ¥bin へ移動します。

   # cd /opt/nec/pf/swclm/SWCLM/bin

3. コマンドを実行し、CA の初期化を行います。

   ・ 初期化を行う場合

   以下のコマンドを実行します。

   # ./SWCLMCAUTIL.sh -init

   ・ 有効期限を変更する場合

   以下のコマンドを実行します。

   [有効期限として、終了日日時を指定する場合]

   # ./SWCLMCAUTIL.sh -init -enddate <終了日時>

   終了日時のフォーマットは、「YYMMDDhhmmssZ」です。また日時は、UTC(協定世界時)で指定します。

   終了日時を 2038/12/31 00:00:00 とする場合は、次のように指定します。

./SWCLMCAUTIL.sh -init -enddate 381231000000Z

[有効期限として、CA 証明書発行日時からの日数を指定する場合]

./SWCLMCAUTIL.sh -init -days <日数>

日数は、CA 証明書発行日時に本指定日数を加算した際に 2049/12/31 23:59:59 を超えない日数を指定する必要があります。

日数を 3 年(1095 日)とする場合は、次のように指定します。

./SWCLMCAUTIL.sh -init -days 1095

4. コマンドを実行すると、CA 証明書のパスフレーズの入力を求められます。
   別紙「SecureWare/Credential Lifecycle Manager セットアップカード」記載のパスフレーズを入力してください。

```
# ./SWCLMCAUTIL.sh -init
Enter PEM pass phrase: 【セットアップカード記載のパスフレーズを入力し、Enter】
Verifying - Enter PEM pass phrase: 【上記★で入力したパスフレーズを入力し、Enter】
Enter pass phrase for /usr/local/myopenssl/ca/ca1/private/cakey.pem: 【上記★で入力したパスフレーズを入力し、Enter】

...

CA certificate is in /usr/local/myopenssl/ca/ca1/cacert.pem
```

5. コマンド実行後、以下のファイルが出力されていることを確認します。

   /usr/local/myopenssl/ca/ca1/cacert.pem …CA 証明書

   /usr/local/myopenssl/ca/ca1/careq.pem …証明書要求(CSR)

   /usr/local/myopenssl/ca/ca1/private/cakey.pem …秘密鍵

   初期化に失敗した場合。、標準出力やイベントログにメッセージが表示されます。メッセージに従い原因を取り除き、再度初期化を実施してください。

初期化手順 (Windows)

1. CLM インストールサーバにログインします。
2. 管理者権限でコマンドプロンプトを起動し、%CLM インストールディレクトリ%¥bin へ移動します。

   > cd c:¥swclm¥SWCLM¥bin

   .

3. コマンドを実行し、CA の初期化を行います。

・ 初期化を行う場合

以下のコマンドを実行します。

> .¥SWCLMCAUTIL.ba -init

・ 有効期限を変更する場合

以下のコマンドを実行します。

[有効期限として、終了日日時を指定する場合]

> .¥SWCLMCAUTIL.bat -init -enddate <終了日時>

終了日時のフォーマットは、「YYMMDDhhmmssZ」です。また日時は、UTC(協定世界時)で指定します。

終了日時を 2038/12/31 00:00:00 とする場合は、次のように指定します。

> .¥SWCLMCAUTIL.bat -init -enddate 381231000000Z

[有効期限として、CA 証明書発行日時からの日数を指定する場合]

> .¥SWCLMCAUTIL.bat -init -days <日数>

日数は、CA 証明書発行日時に本指定日数を加算した際に 2049/12/31 23:59:59 を超えない日数を指定する必要があります。

日数を 3 年(1095 日)とする場合は、次のように指定します。

> .¥SWCLMCAUTIL.bat -init -days 1095

4. コマンドを実行すると、CA 証明書のパスフレーズの入力を求められます。

別紙「SecureWare/Credential Lifecycle Manager セットアップカード」記載のパスフレーズを入力してください。

```
> .¥SWCLMCAUTIL.bat -init
Enter PEM pass phrase: 【セットアップカード記載のパスフレーズを入力し、Enter】
Verifying - Enter PEM pass phrase: 【上記★で入力したパスフレーズを入力し、Enter】
Enter pass phrase for /usr/local/myopenssl/ca/ca1/private/cakey.pem: 【上記★で入力したパスフレーズを入力し、Enter】

...

CA certificate is in /usr/local/myopenssl/ca/ca1/cacert.pem
```

5. コマンド実行後、以下のファイルが出力されていることを確認します。

c:¥myopenssl¥ca¥ca1¥cacert.pem …CA 証明書

c:¥myopenssl¥ca¥ca1¥careq.pem …証明書要求(CSR)

c:¥myopenssl¥ca¥ca1¥private¥cakey.pem …秘密鍵

初期化に失敗した場合。、標準出力やイベントログにメッセージが表示されます。メッセージに従い原因を取り除き、再度初期化を実施してください。

12.4　エッジゲートウェイのキッティングに関する留意事項

NEC 製 エッジゲートウェイでキッティングを行う際の留意事項について説明します。

エッジゲートウェイおよびエッジゲートウェイ付属の WebUI についての詳細は、エッジゲートウェイのマニュアルをご覧ください。

12.4.1　WebUI での入力項目

キッティングで接続する CLM のバージョンおよび、CLA のバージョンにより、エッジゲートウェイの WebUI の[Security Information]-[Security ID]および[Password]に入力する内容が異なります。

- CLM V1.1.0 以降に接続する場合

  SecurityID およびパスワードを、WebUI の[Security Information]-[Security ID]および[Password]に入力します。

  SecurityID、パスワードは、キッティング実施前に CLM へ登録しておく必要があります。登録方法については、別紙「SecureWare/Credential Lifecycle Manager セットアップカード」をご覧ください。

- CLM V1.0 に接続する場合

  CLM V1.0 では、[Security Information]-[Security ID]および[Password]は使用しません。

  次バージョン以降で使用するフィールド(将来のための予約領域)です。

  但し、[Security Information]-[Security ID]および[Password]は入力必須の項目であり未入力とすることはできないため、キッティングの際には任意の文字列を入力してください。

- CLM (限定版)に接続しなければならない場合 (CLM(限定版)から CLM V1.0 以降への移行期間中など)

  CLM 環境設定ファイルに設定した ID/パスワードを WebUI の[Security Information]-[Security ID]および[Password]に入力します。

12.5　証跡データのエクスポートとクリーンアップ

CLM では、リポジトリに認証情報と共に、どのエッジデバイスにどの ID 鍵を発行したかという記録(証跡データ)を保有しています。

長期間の運用に伴って、リポジトリの証跡データが膨大となり、ディスクの圧迫や CLM の性能劣化を引き起こす可能性があります。既にエッジデバイス上に該当する鍵情報が存在せず、証跡としての記録だけがリポジトリに残っている場合は、定期的に証跡データをファイルに出力(エクスポート)し、リポジトリの証跡データを物理削除(クリーンアップ)してください。

以下は、証跡データをファイルに出力し、証跡データを物理削除するスクリプトの参考実装例です。具体的には、発行して 12 か月経過した共通鍵の証跡データをファイルに出力し、証跡データを物理削除する場合のスクリプトです。

以下の例を参考にスクリプトとして実装し、定期的にタスクなどでスクリプトを実行することで証跡データを定期エクスポート・物理削除することができます。

表 12-1 証跡データエクスポート・クリーンアップスクリプト(例)

```
#!/bin/bash

DBUSER=''
DBPORT=5432
DELETEFLAG=0 #1 = data delete
TMP_DIR=/tmp

USAGE1="$0 -u <db access user> -f <file path> [ -p <db port> ] [ -d ]"
USAGE2="      -d : delete data 12 month ago."

RETCODE_SUCCESS=0
RETCODE_INVALID_ARGS=1

# check argument
if [ $# -lt 4 ]; then
    echo $USAGE1
    echo $USAGE2
    exit $RETCODE_INVALID_ARGS
fi

while getopts 'u:f:p:d' OPTION
do
    case $OPTION in
    p)
        # db port
        DBPORT="$OPTARG"
        ;;
    u)
        # db access user
        DBUSER="$OPTARG"
        ;;
    f)
        # csv file path
```

```
            CSV_FILE="$OPTARG"
            ;;
        d)
            # data delete
            DELETEFLAG=1
            ;;
        ?)
            echo $USAGE1
            echo $USAGE2
            exit $RETCODE_SUCCESS
        esac
done

if [ -z "$DBUSER" ] ; then
        echo $USAGE1
        echo $USAGE2
        exit $RETCODE_INVALID_ARGS
fi
if [ -z "$CSV_FILE" ] ; then
        echo $USAGE1
        echo $USAGE2
        exit $RETCODE_INVALID_ARGS
fi

# create csv
psql -U $DBUSER -d swclm -p $DBPORT -c "select keylinkage.seqnumber,keylinkage.keynumber,commonkey.keyalias,k2.keystatusname as ¥"commonkey status¥",to_timestamp(commonkey.registration/1000) as ¥"commonkey registration¥",to_timestamp(commonkey.expiration/1000) as ¥"commonkey expiration¥",keylinkage.keyid,keylinkage.devicenumber,device.devicename,keylinkage.identification,keylinkage.issuecount,to_timestamp(keylinkage.registration/1000) as ¥"trail registration¥",to_timestamp(keylinkage.lastupdate/1000) as ¥"trail lastupdate¥",k1.keystatusname as ¥"trail status¥" from keylinkage inner join device on keylinkage.devicenumber=device.devicenumber inner join commonkey on keylinkage.keynumber=commonkey.keynumber and (to_timestamp(commonkey.registration/1000) < now() + '-12 month') inner join keystatuslist k1 on keylinkage.status = k1.keystatus inner join keystatuslist k2 on commonkey.status = k2.keystatus" -A -F, > $CSV_FILE
wk_retcode=$?

if [ $DELETEFLAG -eq 1 ] ; then
    #delete data
    psql -U $DBUSER -d swclm -p $DBPORT -c "delete from keylinkage where keynumber in (select keynumber from commonkey where to_timestamp(commonkey.registration/1000) < now() + '-12 month')"
fi

exit $wk_retcode
```