Technical Specification CMC Interface

Guide for integrating applications with the SwissSign Certificate Authority CMC interface

Document Type:	**Interface Document**
Author:	**Ingolf Rauh**
Classification:	**C1 (public)**
Application:	**SwissSign**
Owner:	**Product Management**
Issue Date:	**18.12.2020**
Version:	**1.4**
Status:	**Released**

Version Control

Date	Version	Comments	Author
27.06.2014	0.1		Ingolf Rauh/Loic Etienne
28.04.2015	0.2	Reformat, prepare for cmc revoke release	nichan
01.06.2015	0.3	Generic description, move openssl to separate chapter	nichan
10.07.2015	0.4	Changes from RC6 – subject params etc.	Nichan
20.08.2015	0.5	Changes from RC7&8	nichan
27.08.2015	0.6	Product list	Ingolf
15.08.2015	0.7	Review/new examples	Zvonimir
16.08.2015	0.8	Final Prerelease	Ingolf
17.08.2015	0.9	Header mismatch in 6.2	Ingolf
21.08.2015	0.91	Hint concerning “validity” string and that example values should be replaced by the customer with the specific customer values	Ingolf
3.11.2015	0.92	Parameter description	Ingolf
6.11.2015	0.93	Description key pair / CSR	Ingolf
2.12.2015	0.94	Revocation update	Zvonimir
11.12.2015	0.95	Account update	Christoph
21.12.2015	0.96	Update: Replacement Extended CMC Status Control by Status Control	Ingolf
07.09.2016	1.00	• new: “6 Retrieve a certificate” • 4.1 parameter validity and base64 optional • 4.1 content_type: application/pkcs7-mime for Full PKI Requests • new: 7.1.2 specific CMC Retrieve Errors • 8.3 corrected URL parameter and content type • new: 8.4 Retrieve with openssl • Appendix A: validity for SSL Gold EV Multidomain provided	Ulrich
01.03.2018	1.1	New parameter «ignore_unconsumed» supplemented	Christoph
20.11.2019	1.2	• New optional parameter «suppress_www_domain» supplemented • Appendix A: Products updated	Dany
01.08.2020	1.3	• Changed validity periods • Correction Appendix A • Correction URL for SwissSign MPKI page	Adrian
18.12.2020	1.4	• Update of revocation resson codes (chapter 5.1 and 9.3)	Onur Cebeci

Table of contents

References

Reference	
1 RFC 5272	http://tools.ietf.org/html/rfc5272

1 Purpose

Applications can use the SwissSign CA CMC Interface to perform a variety of certificate related functions including requesting and revoking certificates and performing auto-enrollment. The interface is based on the IETF CMC standard (Certificate Management over CMS https://tools.ietf.org/html/rfc5272). Please have a look into this RFC standard first for any further questions like the difference between Simple PKI Request and Full PKI Request. The current CA Software Version supports the following operations:

- **Request a new certificate (also called certificate enrolment request)**
- **Revoking a certificate**
- **Retrieve an existing certificate**

The guide is intended for developers or system administrators who want to integrate their system or application with the SwissSign CA CMC interface.

Note: The CMS interface supersedes the so-called “remote MPKI” (Managed PKI) interface.

2 Before you start

Before starting with the CMC interface, you require the following:

1) **A valid SwissSign Managed PKI (MPKI) contract that grants you the right to act as a Registration Authority (RA) for certificates you request.**
(cf. https://www.swisssign.com/en/managed-pki/managed-pki-service.html)
2) **An operator certificate and account name provided by SwissSign. These are required to access the interface. The operator certificate will be supplied by SwissSign during setup of a Managed PKI.**
3) **One or more product names and options for which you can request certificates are also provided by SwissSign. They correspond to the products ordered in the Managed PKI contract. See also Appendix A: Products.**
4) **A client or application that supports:**
    - **Generating either PKCS#10 or CRMF request formats**
    - **Issuing HTTP/POST requests**
    - **Parsing PKCS#10 or CRMF responses**
      i.e. Simple and Full PKI Responses of Cryptographic Message Syntax (CMS)

3 Access

Clients must access the CMC interface with the appropriate access (operator) certificate using mutual SSL authentication.

After authenticating clients can issue HTTP/POST requests to the CMC interface. There are two environments available:

- **For testing purposes (no public certificates): https://ra.swiss.signdemo.com/ws/cmc**
- **For production: https://ra.swisssign.net/ws/cmc**

4 Request a certificate

4.1 Request Parameters

The format for a certificate request is as follows:

```
POST /ws/cmc?parameters
Host: host
Content-Type:content_type
data
```

All URL parameters are delimited using the ampersand (&) character.

Required parameters:

account: A string with the registration authority account provided by SwissSign during setup.

product: A string with the certificate product to be issued for this request. SwissSign will provide a list of products that are available with your contract. The value of this parameter determines which type of certificate will be created such as SSL, email encryption, Extended Validation. See table in Appendix A: Products.

host: Specifies the target host:
For testing https://ra.swiss.signdemo.com, for production https://ra.swisssign.net.

content_type: must be

- application/pkcs10 for Simple PKI Requests (PKCS#10) .
- application/pkcs7-mime for Full PKI Requests (CRMF)

data: contains either simple PKI request or a full PKI request. See the following sections for details.

Optional parameters:

subject: a URL encoded/escaped UTF-8 string with a list of subject fields. For example *"subject=cn%3Dfirstname%20lastname%2Co%3DMyCompany%2Cl%3DZ%C3%BCrich%2Cc%3Dch"* instructs the CMC to create a certificate with a Subject "*cn=firstname lastname,o=MyCompany,l=Zürich,c=ch"*. This parameter overrides the subject from the CSR. It is not possible to override only one subject attribute (e.g. /O), if this parameter is used, the complete subject needs to be provided.

Please use the following subject parameters and pay attention that they have to be used as lower case characters:
c: country
o: organization
ou: organization detail

dnQualifier: qualifier of the distinguished name
st: state, canton
serialNumber: serial number
cn: common name
l: locality
title: title
surname: surname
givenname: given name
initials: initials
pseudonym: pseudonym
generationQualifier: generation qualifier
dc: domain component
emailAddress: email address
street: street name
uid: User ID
postalCode: postal code
businessCategory: business category
joiL: jurisdiction of incorporation locality
joiST: jurisdiction of incorporation state or province
joiC: jurisdiction of incorporation country

subjectAltName: a URL encoded/escaped string with a list of DNS, IP or Emails. For example "subjectAltName=*dns%3Aserver1.mydomain.com%3Bdns%3Aserver2.mydomain.com%3Beml%3Auser%40mydomain.com"* instructs the CMC to create a certificate with the Subject Alternate Name *"dns:server1.mydomain.com;dns:server2.mydomain.com;eml:user@mydomain.com".*

validity: A string with a validity period for the certificate. Accepted values include "1y", "2y", "3y" for a 1, 2 or 3 year validity period if the product supports multiple validity periods. If the parameter validity is missing the default validity of the product is chosen. SwissSign will provide a list of accepted values. See also Appendix A: Products. Validity specification from the CSR will always be ignored.

base64: If set to the value "1" then output will be made in base64. Otherwise or if parameter is omitted normal DER will be output.

ignore_unconsumed:

Set value to "1" if you want to ignore "unconsumed sdn/san" error and go ahead with certificate creation.

encrKeyHash: encrypted key hash. Used to override the normal unsigned encrKeyHash (1.3.6.1.4.1.311.21.21) attribute in the full PKI requests if needs be. Provided as a helper to avoid resigning requests if needs be.

suppress_www_domain:

Set value to "true" if you want to suppress the automatically added www-domain or set value to "false" to not suppress the automatically added www-domain. If no parameter is set, the default from the product configuration will be taken.

4.1.1 Simple PKI Request

Example of a PKCS#10 request for a product called test using the account "mycompany.ra" to request a certificate for the product "mycompany-ssl-silver" with a validity of 1 year in the test environment. The binary data for has been omitted and represented here as <pkcs10_binary_data>:

```
POST /ws/cmc?account=mycompany.ra&product=mycompany-ssl-silver&validity=1y HTTP/1.1
Host: ra.swiss.signdemo.com
Content-Length: 662
Content-Type:application/pkcs10
Connection: close

<pkcs10_binary_data>
```

4.1.2 Response message

If the request was successful the server returns a HTTP status code 200 with a PKCS#7 containing the certificate chain information as signed data according to RFC 5272 Simple PKI Response. This includes root CA, any intermediate CA's and the requested certificate.

Example with binary data omitted and represented here as *<pkcs7_binary_data>*:

```
HTTP/1.1 200 OK
Date: Wed, 21 Jan 2015 18:36:41 GMT
Content-Length: 4753
Cache-Control: max-age=3600
Expires: Wed, 21 Jan 2015 19:36:41 GMT
X-FRAME-OPTIONS: SAMEORIGIN
Connection: close
Content-Type: application/pkcs7-mime
<pkcs7_binary_data>
```

4.1.3 Full PKI Request

A full PKI request must include the necessary fields according the product being requested and conform to the Full PKI Request specified in RFC 5272. https://tools.ietf.org/html/rfc5272#section-3.2. The following request has been shortened to show specific OID's relevant for an email encryption certificate.

```
U.C.SEQUENCE {
 U.P.OBJ_ID 1.2.840.113549.1.7.2 # signedData
 C.C.0 {
  U.C.SEQUENCE {
   U.P.INTEGER 03
   U.C.SET {
    U.C.SEQUENCE {
     U.P.OBJ_ID 1.2.840.113549.1.1.11 # SHA-256
     U.P.NULL ""
    }
   }
   U.C.SEQUENCE {
    U.P.OBJ_ID 1.3.6.1.5.5.7.12.2 #id_kp_clientAuth
    C.C.0 {
        #
        # microsoft specific OID's ommitted
        #
        U.C.SEQUENCE {
         C.C.0 {
          U.P.INTEGER 01
          U.C.SEQUENCE {
           U.C.SEQUENCE {
```

```
U.P.INTEGER 00
U.C.SEQUENCE {
 U.C.SET {
  U.C.SEQUENCE {
   U.P.OBJ_ID 1.2.840.113549.1.9.1
   U.P.IA5STRING "Firstname Lastname"
  }
 }
 U.C.SET {
  U.C.SEQUENCE {
   U.P.OBJ_ID 1.2.840.113549.1.9.1
   U.P.IA5STRING "firstname.lastname@mydomain.com"
  }
 }
}
U.C.SEQUENCE {
 U.C.SEQUENCE {
  U.P.OBJ_ID 1.2.840.113549.1.1.1 # RSA key
  U.P.NULL ""
 }
 U.P.BITSTRING {
  U.C.SEQUENCE {
   U.P.INTEGER
   hex_data
   U.P.INTEGER 01:00:01
  }
```

4.1.4 New CSR – new key pair

Please note that SwissSign has generally the policy not to reuse an existing key for a CSR. In the current implementation an error is returned. In future implementations the existing certificate based on the public key in the CSR will be returned.

5 Revoke a certificate

5.1 Revocation request message

The format for a revocation request is as follows:

```
POST /ws/cmc?parameters
Host: host
Content-Type: application/pkcs7-mime
data
```

All URL parameters are delimited using the ampersand (&) character.

Required parameters:

account: A string with the Registration Authority account provided by SwissSign during setup.

host: Specifies the target host:
For testing https://ra.swiss.signdemo.com, for production https://ra.swisssign.net.

content_type: must be “application/pkcs7-mime”.

data: contains a revocation request control specifying the issuerName, serialNumber and a suggested reason (section 6.11. Revocation Request Control in RFC 5272)

The revocation request within the “data” has to contain a numeric reason code. Allowed revocation reason codes for subscriber certificates are:

- 0=unspecified: No specific reason is given
- 1=keyCompromise: The private key has been stolen or there is the risk that it has been stolen.
- 3=affiliationChanged: Subject information changed, e.g. change of company name or surname.
- 4=superseded: The certificate was re-placed by another one.
- 5=cessationOfOperation: Indicates that the public-key certificate is no longer needed for the purpose for which it was issued.
- 9=privilegeWithdrawn: Authorization revoked, a privilege contained within that public-key certificate has been withdrawn.

The following example shows the asn1 structure which revokes a certificate with serial „520C60926231E90FECCC2F1EE1157E71E87B9DCC“ from issuer CA “CN=SwissSign Server Gold CA 2014 – G22, O=SwissSign AG, C=CH” with reason code “00” (unspecified).

Please note that you have to replace the example strings like „CH”, “SwissSign AG” etc. with the real parameters of your request!

```
U.C.SEQUENCE {
 U.P.OBJ_ID 1.2.840.113549.1.7.2   # Signed data
 C.C.0 {
  U.C.SEQUENCE {
   U.P.INTEGER 03
```

```
  U.C.SET {
  }
  U.C.SEQUENCE {
   # id-cct-PKIData
   U.P.OBJ_ID 1.3.6.1.5.5.7.12.2 # id-cct-PKIData
   C.C.0 {
    U.P.OCTETSTRING {
     U.C.SEQUENCE {
      U.C.SEQUENCE {
       U.C.SEQUENCE {
        U.C.SEQUENCE {
         U.P.INTEGER 01
         # id-cmc-revokeRequest
         U.P.OBJ_ID 1.3.6.1.5.5.7.7.17   # id-cmc-revokeRequest
         U.C.SET {
          U.C.SEQUENCE {
           U.C.SEQUENCE {
            U.C.SET {
             U.C.SEQUENCE {
              U.P.OBJ_ID 2.5.4.6
              # Country name
              U.P.PRINTABLESTRING “CH”
             }
            }
            U.C.SET {
             U.C.SEQUENCE {
              U.P.OBJ_ID 2.5.4.10
              # Organization name
              U.P.UTF8STRING “SwissSign AG”
             }
            }
            U.C.SET {
             U.C.SEQUENCE {
              U.P.OBJ_ID 2.5.4.3
              # Common name
              U.P.UTF8STRING “SwissSign Gold CA 2014 – G22”
             }
            }
           }
           # serialNumber
           U.P.INTEGER 520C60926231E90FECCC2F1EE1157E71E87B9DCC
           # revocation reason
           U.P.ENUMERATED 00
          }
         }
        }
       }
       U.C.SEQUENCE {
       }
       U.C.SEQUENCE {
       }
       U.C.SEQUENCE {
       }
      }
     }
    }
   }
   U.C.SET {
   }
  }
 }
}
```

5.2 Revocation response message

A successful response message is a Full PKI Response and contains an CMC Status Info Control structure indicated by OBJECT IDENTIFIER 1.3.6.1.5.5.7.7.1 and provides a message indicating the values provided in the request including: certificate serial, product, issuing CA, reason and account.

```
U.C.SEQUENCE {
 U.P.OBJ_ID 1.2.840.113549.1.7.2   # Signed data
 C.C.0 {
  U.C.SEQUENCE {
   U.P.INTEGER 01
   U.C.SET {
   }
   U.C.SEQUENCE {
    U.P.OBJ_ID 1.3.6.1.5.5.7.12.3   # id-cct-PKIResponse
    C.C.0 {
     U.P.OCTETSTRING {
      U.C.SEQUENCE {
       U.C.SEQUENCE {
        U.C.SEQUENCE {
         U.C.SEQUENCE {
          U.P.INTEGER 00
          U.P.OBJ_ID 1.3.6.1.5.5.7.7.1   # id-cmc-cMCStatusInfo
          U.C.SET {
           U.C.SEQUENCE {
            U.P.INTEGER 00
            U.C.SEQUENCE {
             U.P.INTEGER 00
            }
            U.P.UTF8STRING "
Certificate '520c60926231e90feccc2f1ee1157e71e87b9dcc' successfully revoked from CA '
cn=SwissSign Gold CA 2014 – G22,o=SwissSign
AG,c=CH' with reason '0' (unspecified) by /CN=MyUser/OU=CompanyOU/O=MyCompany/C=CH (se
rial: E443E9ACEC96B32E8C4141EC5D97563 ) [ account mycompany.ra ]."
           }
          }
         }
        }
        U.C.SEQUENCE {
        }
        U.C.SEQUENCE {
        }
       }
      } #
3081b33081ac3081a902010006082b06010505070701318199308196020100300302010000c818b436572
74696669636174652035643463346665643336383238386538646632633636303166323339326362323
366162646463623920287072 6f64756374206576292073756363657373667 56c6c79207265766f6b656
42066726f6d20434120636e3d43414730312c6f3d2c633d434820776974682072656173 6f6e203020627
9207573657220534d30312e30003000
     }
    }
    U.C.SET {
    }
   }
  }
 }
```

6 Retrieve a certificate

6.1 Retrieval request message

The format for a retrieval request is as follows:

```
POST /ws/cmc?parameters
Host: host
Content-Type: application/pkcs7-mime
data
```

All URL parameters are delimited using the ampersand (&) character.

Required parameters:

account: **A string with the Registration Authority account provided by SwissSign during setup.**

host: **Specifies the target host:**
For testing https://ra.swiss.signdemo.com, for production https://ra.swisssign.net.

content_type: **must be “application/pkcs7-mime”.**

data: **contains a retrieval request control specifying the issuerName, serialNumber (section 6.9. Get Certificate Control in RFC 5272)**

The following example shows the asn1 structure which retrieves a certificate with serial „18B26C7FB392F6A7AF68AA2237EE88EE3C“ from issuer CA “CN=SwissSign Server Gold CA 2014 – G22, O=SwissSign AG, C=CH”.

Please note that you have to replace the example strings like “18B26C7FB392F6A7AF68AA2237EE88EE3C”, „CH”, “SwissSign AG” etc. with the real parameters of your request!

```
U.C.SEQUENCE {
 U.P.OBJ_ID 1.2.840.113549.1.7.2   # Signed data
 C.C.0 {
  U.C.SEQUENCE {
   U.P.INTEGER 03
   U.C.SET {
   }
   U.C.SEQUENCE {
    U.P.OBJ_ID 1.3.6.1.5.5.7.12.2   # id-cct-PKIData
    C.C.0 {
     U.P.OCTETSTRING {
      U.C.SEQUENCE {
       U.C.SEQUENCE {
        U.C.SEQUENCE {
         U.P.INTEGER 01
         U.P.OBJ_ID 1.3.6.1.5.5.7.7.15   # id-cmc-getCert
         U.C.SET {
          U.C.SEQUENCE {
           U.C.SEQUENCE {
```

```
            U.C.SET {
             U.C.SEQUENCE {
              U.P.OBJ_ID 2.5.4.6   # Country name
              U.P.PRINTABLESTRING "CH"
             }
            }
            U.C.SET {
             U.C.SEQUENCE {
              U.P.OBJ_ID 2.5.4.10   # Organization name
              U.P.UTF8STRING "SwissSign AG"
             }
            }
            U.C.SET {
             U.C.SEQUENCE {
              U.P.OBJ_ID 2.5.4.3   # Common name
              U.P.UTF8STRING "SwissSign Gold CA - G2"
             }
            }
           }
           U.P.INTEGER 18B26C7FB392F6A7AF68AA2237EE88EE3C   # serial
          }
         }
        }
        U.C.SEQUENCE {
        }
        U.C.SEQUENCE {
        }
        U.C.SEQUENCE {
        }
       }
     } #
308187307f307d02010106082b06010505070711316e306c3055310b300906035504061302434831153013060355040a0c0c53776973735369676e204147312f302d06035504030c2653776973735369676e20506572736f6e616c2053696c766572204341203230303820d20473202100086c35d2303e87b53910cc74c83bef50a0100300030003000
    }
   }
   U.C.SET {
   }
  }
 }
}
```

6.2 Retrieval response message

If the request was successful, the server returns a HTTP status code 200 with a PKCS#7 containing the certificate information as signed data according to RFC 5272 Simple PKI Response (CMS Signed Data with no SignerInfo). The response includes root CA, any intermediate CA's and the requested certificate.

Example with binary data omitted and represented here as *<pkcs7_binary_data>*:

```
HTTP/1.1 200 OK
Date: Wed, 24 Aug 2016 14:51:16 GMT
Cache-Control: private, no-cache
Pragma: no-cache
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Allowed: POST
Content-Transfer-Encoding: BINARY
Content-Length: 3094
Strict-Transport-Security: max-age=31536000 ; includeSubDomains
Connection: close
C<pkcs7_binary_data>
```

7 Errors

If a request fails a HTTP response with HTTP status code not equal to 200 is returned containing an CMC Status Info Control structure indicated by OBJECT IDENTIFIER 1.3.6.1.5.5.7.7.1. The response is encoded in DER format.

```
SEQUENCE(3 elem)
 SEQUENCE(1 elem)
  SEQUENCE(3 elem)
   INTEGER0
   OBJECT IDENTIFIER1.3.6.1.5.5.7.7.1
   SET(1 elem)
    SEQUENCE(3 elem)
    INTEGER<CMCFailInfo>
     SEQUENCE(1 elem)
      INTEGER0
      UTF8String<statusString>
 SEQUENCE(0 elem)
 SEQUENCE(0 elem)
```

The following tables lists possible values for <CMCFailInfo> and <statusString> including the corresponding HTTP status codes and indicates whether the status is valid for requests, revokes or both:

7.1.1 CMC General Request Errors

CMCFailInfo	statusString	HTTP Status Code
12	Please try again Later.	500
2	Content-type not allowed: $type. Content-type expected: application/pkcs10, application/pkcs7-mime or application/cmc-revoke	415
7	No account available.	400
7	Too many accounts available. Please specify which one you would like to use in the ‘account’ URL parameter.	400
7	The account is not allowed to perform the operation.	400
2	There is no available product for you as of now.	400
2	Please specify which product would you like to use in the ‘product’ URL parameter.	400
2	You are not allowed to use the product requested.	400
2	Validity must be one of $$prod_rec{validity}	400
2	No validity (must one of $$prod_rec{validity})	400
2	POST/Multipart not allowed.	400
2	Error in request (pkcs10, cmc OR keyid not unique OR keysize < 2048?).	400
2	Product error: $msg	400

2	Invalid request: $msg	400
2	Unconsumed SDN (i.e.: SDN attributes not needed and not utilized; please remove them and resubmit your request):	400
2	Unconsumed SAN (i.e.: SAN attributes not needed and not utilized; please remove them and resubmit your request):	400
12	Error during commit. Transaction rolled back:	500
2	Method not allowed:	400
12	Request collisions: serial_number; Certificate collisions: serial_number.	500

7.1.2 CMC Revoke Errors

CMCFailInfo	statusString	HTTP Status Code
2	No such certificate serial <serial_no> OR certificate already revoked OR certificate does not belong to account mycompany.ra OR account not allowed to revoke certs OR trying to revoke a CA certificate.	500

7.1.3 CMC Retrieve Errors

CMCFailInfo	statusString	HTTP Status Code
4	No certificate found matching: serial=<serial_no> and issuer=<certificate_issuer>	400

8 Examples with openssl

The following chapters shows a step-by-step example using openssl as a client for the CMC interface to request and revoke a certificate.

8.1 Convert operator certificate

As an initial step the operator certificate required to access the interface must be converted from ".p12" to ".pem" format. You will use pem file later when submitting requests.

Replace the following values accordingly:

- filename with operator certificate in p12 format: "operator.p12"
- filename of operator certificate in pem format: "operator.pem"
- password for p12: abcd
- password for pem: efgh

```
openssl pkcs12 \
  -in operator.p12 \
  -out operator.pem \
  -passin pass:abcd \
  -passout pass:efgh \
  -clcerts
```

8.2 Request with openssl

8.2.1 Prepare the request

The first command creates a new key and signs a PKCS#10 request containing data for the certificate. Change the values accordingly.

Note: depending on the product and the policies applied based on your MPKI contract additional values may be necessary for example organization often has to be specified using

- O=<organization_name>.
- CN="Firstname Lastname"
- emailAddress=firstname.lastname@mydomain.com
- Operator certificate password="ijkl"

```
openssl req \
  -new \
  -newkey rsa:2048 \
  -utf8 \
  -subj '/CN=Firstname Lastname/emailAddress=firstname.lastname@mydomain.com' \
  -passout 'pass:ijkl' \
  -keyout key.pem \
  -outform DER \
  -out request.der
```

Prepare the binary PKCS#10 file for the request.

8.2.2 Submit and extract the response

The request can be submitted to the CMC interface using an operator certificate (see 6.1). Replace the account name “my_account” with your account name and the product name “abc-perso-gold-1y” with the appropriate product name according to chapter 4 which was enabled for your account. abc-perso-gold-1y is the typical definition of products for existing customers (before November 2015). New customers will get only the product “<company>-perso-gold” or “<company>-perso-silver”. They can specify the validity with e.g. “&validity=1y”. If no validity string is specified, the first allowed validity for this product will be taken. If a validity is chosen, which was not allowed for this product, an error will be produced. “y” will mean years, “M” will mean months and “d” will be days.

The connection address is given in the –connect parameter. In case of the productive system it must be replaced with the URL of the production system (ra.swisssign.net). The https call specifies the operator certificate with the –cert parameter (created in step 6.1) and the password with the –pass parameter. Replace the password value “efgh” with the appropriate password.

Please use the account name announced by the SwissSign fulfillment during setup of your Managed PKI and the appropriate product names which were configured for you.

```
echo -ne \
   "POST /ws/cmc?account=my_account&product=abc-perso-gold-1y HTTP/1.1" \
   "\r\nHost: ra.swiss.signdemo.com" \
   "\r\nContent-Length: `cat request.der | wc -c`" \
   "\r\nContent-Type:application/pkcs10\r\nConnection: close" \
   "\r\n\r\n" \
| cat - request.der \
| openssl s_client \
  -connect ra.swiss.signdemo.com:443 \
  -servername ra.swiss.signdemo.com \
  -cert operator.pem \
  -pass pass:efgh\
  -quiet \
| sed '1,/^\r$/d' \
> cmc_response.der
```

If the request was successful, the cmc_response.der file contains the certificate chain including the issued certificates. The following commands extract two files.

1) cert_chain.pem: contains the CA certificates

```
# extract certificate chain from cmc_response.der
openssl pkcs7 \
  -inform der \
  -in cmc_response.der \
  -print_certs \
> cert_chain.pem
```

2) cert.pem: contains the end entity certificate that was issued based on the values in the request

```
# extract end user certificate (match ascii part of the subject)
sed -n '/subject=.*firstname.lastname/,/^-* *END CERTIFICATE/p' cert_chain.pem \
> cert.pem
```

8.2.3 Check and convert

The following command verifies the end entity certificate and converts it to DER format:

```
# check cert.pem and convert it to cert.der
openssl x509 -in cert.pem -outform DER -out cert.der
```

8.3 Revoke with openssl

The following example shows how the CMC interface should be used to revoke a certificate using openssl.

First, generate a revocation request by creating an openssl asn1parse “revoke.cnf” config file. The following parameters must be adapted and are marked bold:

serial: serial number of the certificate to be revoked in hex

reason: reason for the revocation. See chapter 5.1 Revocation request message for allowed reason codes and their meaning.

country: country from the subject of the issuing CA

organization: organization from the subject of the issuing CA

commonName: CN from the subject of the issuing CA

```
asn1=SEQUENCE:SignedData
[SignedData]
msgtype=OID:pkcs7-signedData
envel=EXPLICIT:0C,SEQUENCE:PKIdata
[PKIdata]
ptype=INTEGER:03
vset=SET
dat=SEQUENCE:Internal
couic=SET
[Internal]
cct=OID:id-cct-PKIData
envel=EXPLICIT:0C,OCTWRAP,SEQUENCE:PayLoad
[PayLoad]
lay1=SEQUENCE:PayLoad2
drop1=SEQUENCE
drop2=SEQUENCE
drop3=SEQUENCE
[PayLoad2]
lay2=SEQUENCE:PayLoad3
[PayLoad3]
lay3=INTEGER:01
lay4=OID:id-cmc-revokeRequest
cabridge=SETWRAP,SEQUENCE:CAout
[CAout]
caseq=SEQUENCE:CA
serial=INTEGER:0x86C35D2303E87B53910CC74C83BEF5
reason=ENUMERATED:00
[CA]
lay5=SETWRAP,SEQUENCE:C
lay6=SETWRAP,SEQUENCE:O
lay7=SETWRAP,SEQUENCE:CN
[C]
laya1=OID:countryName
country=PRINTABLESTRING:CH
[O]
laya3=OID:organizationName
organization=UTF8String:SwissSign AG
[CN]
laya5=OID:commonName
commonname=UTF8String:SwissSign Personal Silver CA 2014 - G22
```

Secondly, generate the DER file of the request by executing a statement like this:

```
openssl asn1parse -genconf revoke.cnf -out revoke.der
```

Thirdly, submit revocation request to the CA:

```
# submit revocation revoke.der over cmc/https (with client authentication)
echo -ne \
   "POST /ws/cmc?account=my_account& HTTP/1.1" \
   "\r\nHost: ra.swiss.signdemo.com" \
   "\r\nContent-Length: `cat revoke.der | wc -c`" \
   "\r\nContent-Type: application/pkcs7-mime \r\nConnection: close" \
   "\r\n\r\n" \
| cat - reqvoke.der \
| openssl s_client \
  -connect ra.swiss.signdemo.com:443 \
  -servername ra.swiss.signdemo.com \
  -cert operator.pem \
  -pass pass:efgh\
  -quiet \
| sed '1,/^\r$/d' \
> cmc_response.der
```

8.4 Retrieve with openssl

The following example shows how the CMC interface should be used to retrieve a certificate using openssl:

First, generate a retrieval request by creating an openssl asn1parse “retrieve.cnf” config file. The following parameters must be adapted and are marked bold:

serial: serial number of the certificate to be retrieved in hex

country: country from the subject of the issuing CA

organization: organization from the subject of the issuing CA

commonName: CN from the subject of the issuing CA

The parameter in the file differs from the corresponding revoke request configuration only in two points.

- **The OID specifying the request is now *id-cmc-getCert* instead of *id-cmc-revokeRequest*.**
- **The line “reason=ENUMERATED:00” is dropped.**

```
asn1=SEQUENCE:SignedData
[SignedData]
msgtype=OID:pkcs7-signedData
envel=EXPLICIT:0C,SEQUENCE:PKIdata
[PKIdata]
ptype=INTEGER:03
vset=SET
dat=SEQUENCE:Internal
couic=SET
[Internal]
cct=OID:id-cct-PKIData
```

```
envel=EXPLICIT:0C,OCTWRAP,SEQUENCE:PayLoad
[PayLoad]
lay1=SEQUENCE:PayLoad2
drop1=SEQUENCE
drop2=SEQUENCE
drop3=SEQUENCE
[PayLoad2]
lay2=SEQUENCE:PayLoad3
[PayLoad3]
lay3=INTEGER:01
lay4=OID:id-cmc-getCert
cabridge=SETWRAP,SEQUENCE:CAout
[CAout]
caseq=SEQUENCE:CA
serial=INTEGER:0x3a4378e5917d747276749ab51a466781d3264c5b
[CA]
lay5=SETWRAP,SEQUENCE:C
lay6=SETWRAP,SEQUENCE:O
lay7=SETWRAP,SEQUENCE:CN
[C]
laya1=OID:countryName
country=PRINTABLESTRING:CH
[O]
laya3=OID:organizationName
organization=UTF8String:SwissSign AG [CN]
laya5=OID:commonName
commonname=UTF8String:CASM01commonname=UTF8String:SwissSign Personal Silver CA 2014 -
G22
```

Secondly, generate the DER file of the request by executing a statement like this:

```
openssl asn1parse -genconf  retrieve.cnf -out retrieve.der
```

Thirdly, submit retrieval request to the CA:

```
# submit revocation revoke.der over cmc/https (with client authentication)
echo -ne \
   "POST /ws/cmc?account=my_account  HTTP/1.1" \
   "\r\nHost: ra.swiss.signdemo.com" \
   "\r\nContent-Length: `cat retrieve.der | wc -c`" \
   "\r\nContent-Type: application/pkcs7-mime \r\nConnection: close" \
   "\r\n\r\n" \
| cat - retrieve.der \
| openssl s_client \
  -connect ra.swiss.signdemo.com:443 \
  -servername ra.swiss.signdemo.com \
  -cert operator.pem \
  -pass pass:efgh\
  -quiet \
| sed '1,/^\r$/d' \
> cmc_response.der
```

If the request was successful, the cmc_response.der file contains the certificate chain including the issued certificate. The following commands extract two files.

1) **cert_chain.pem: contains the CA certificates**

```
# extract certificate chain from cmc_response.der
openssl pkcs7 \
  -inform der \
  -in cmc_response.der \
  -print_certs \
```

```
> cert_chain.pem
```

2) **cert.pem: contains the end entity certificate that was issued based on the values in the request**

```
# extract end user certificate (match ascii part of the subject)
sed -n '/subject=.*firstname.lastname/,/^-* *END CERTIFICATE/p' cert_chain.pem \
> cert.pem
```

Depending on your request you have to modify the search condition suitable for finding the certificate or extract the pem file manually with an editor form the cert_chain.pem

Appendix A: Products

The following standard products are defined for the managed PKI usage according to the order placed on the www.swisssign.com page.

Product ordered	"product" specification according Request Parameters	Possible parameters "validity" according Request Parameters
SSL Gold EV	<org>-ev-gold	1y
SSL Gold EV Multidomain	<org>-ev-gold-multi	1y
SSL Gold	<org>-ssl-gold	1y
SSL Gold Wildcard	<org>-gold-wild	1y
SSL Gold Multidomain	<org>-ssl-gold-multi	1y
SSL Silver	<org>-ssl-silver	1y
SSL Silver Wildcard	<org>-silver-wild	1y
E-Mail ID Silver	<org>-email-silver	1y,2y,3y
E-Mail ID Gold	<org>-email-org-gold	1y,2y,3y
E-Mail ID Gold Authentication	<org>-email-org-gold-auth	1y,2y,3y
E-Mail ID Gold RSASSA-PSS	<org>-email-org-gold-rsassapss	1y,2y,3y