SwissSign

New CA: Technical Specification of CMC Interface

Guide for integrating applications with the CMC Interface of SwissSign’s new Certificate Authority system.

Document Type: Interface Document
Author: Onur Cebeci
Classification: C1 (public)
Application: SwissSign CA
Owner: Product Management
Issue Date: 05.02.2026
Version: 2.4
Status: released

Version Control

Date	Version	Comments	Author
03.04.2023	2.0	• Deprecation of validity parameter • Unnecessary parameters for new CA removed • msUPN added to subjectAltName • New parameter “sid” added to optional parameters • Appendix A of product names with mapping between legacy and new CA added • Error messages reviewed and updated • Examples with OpenSSL reviewed and updated • Examples of OpenSSL version 1.1.x and 3.0.x added • Retrieve and request updated • Unified all examples to SwissSign Pro S/MIME E-Mail ID Gold • Troubleshooting added	Adrian Mueller Onur Cebeci Zorka Csendes Leonard Dushi
15.03.2024	2.1	• New product multi-domain wildcard • New attributes givenName, surname & pseudonym	Onur Cebeci
06.05.2025	2.2	• New S/MIME Profiles: no UID and no "3y" anymore • New product "SwissSign S/MIME OV" added	Onur Cebeci
26.11.2025	2.3	• Reduced Validity: Shortened Certificate Lifetime: validity in days instead of year for TLS certificates	Onur Cebeci
05.02.2026	2.4	• Reduced Validity: max. validity reduced from 365 down to 198 days for TLS certificates	Onur Cebeci

TABLE OF CONTENTS

References

Reference	Document	Link
RFC 5272	Certificate Management over CMS (CMC)	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). 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 CA software supports the following operations:

- Request a new certificate (also called certificate enrollment request)
- Revoke 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: For the integration with the CMC interface of legacy CA please refer to the corresponding document: https://www.swisssign.com/en/dam/jcr:8ff02777-a6b6-4872-8a6c-535d3cb2d565/CMCInterface_EN.pdf

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 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:
    a. Generating either PKCS#10 or CRMF request formats
    b. Issuing HTTP/POST requests
    c. Parsing PKCS#10 or CRMF responses, i.e. Simple and Full PKI Responses of Cryptographic Message Syntax (CMS)
5) WebUI login and the domain must be validated to be able to request certificates, i.e. mydomain.com
6) Rights to set or modify DNS records such as TXT and in special cases CAA and CNAME

Note: Please check the document CMC API on the new SwissSign CA for known differences and issues, and some more information.

3 ACCESS

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

Clients can issue HTTP/POST requests over the CMC interface after authenticating. There are two environments available:

- Pre-Prod for testing purposes (no public certificates): https://cmc.pre.swisssign.ch/ws/cmc
  - Please check the web page New Pre-Prod environment for SwissSign Partners for actual information about the new Pre-Prod environment, procedure to onboard the environment and differences between Pre-Prod and PROD.
- PROD for production: https://cmc.swisssign.ch/ws/cmc

Target host	Pre-Prod	PROD
CMC URL	https://cmc.**pre**.swisssign.ch/ws/cmc	https://**cmc.swisssign.ch**/ws/cmc
account=	• Assigned from sales office i.e. "MPKI5000003%20-%20Firma%20Muster%20AG"	• New customers: use the account name sent by e-mail i.e. "MPKI5000003%20-%20Firma%20Muster%20AG" • Migration customers: set to the same string as used on legacy CA i.e. "mycompany.ra"
product=	• Pre-Prod product name as URL encoded on Appendix A: Products, i.e. “SwissSign%20Pro%20S%2FMIME%20E-Mail%20ID%20Gold” • Same product name as you use on the legacy CA, i.e. account and product	• PROD product name as URL encoded on Appendix A: Products, i.e. “SwissSign%20Pro%20S%2FMIME%20E-Mail%20ID%20Gold” • Same product name as you use on the legacy CA, i.e. account and product

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 for the registration authority account provided by SwissSign during setup.
product	A string for the certificate product to be issued. 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 TLS, S/MIME, Extended Validation etc. See table in Appendix A: Products.
Host	Specifies the target host: • for testing https://cmc.**pre**.swisssign.ch/ws/cmc • for production https://**cmc.swisssign.ch**/ws/cmc
content_type	Must be one of the followings: • 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=surname%3DLastname%2CgivenName%3DFirstname instructs the CMC to create a certificate with a subject “surname=Lastname,givenName=Firstname”. These parameters override the subject from the CSR. Please use the following subject parameters:

Parameter	Meaning	Encoding
cn	common name	• UTF8 String for S/MIME and TLS Wildcard • Printable String for the rest of TLS
email emailAddress	email address	IA5 String
givenName	given name	UTF8 String
surname	surname	UTF8 String
pseudonym	pseudonym	UTF8 String

subjectAltName	A URL encoded/escaped string with a list of DNS, Email or MS UPN. For example: “subjectAltName=dns%3Aserver1.mydomain.com%3Bdns%3Aserver2.mydomain.com%3Beml%3Auser%40mydomain.com%3Bmsupn%3Aprincipal%40mydomain.com instructs the CMC to create a certificate with the Subject Alternative Name “dns:server1.mydomain.com;dns:server2.mydomain.com;eml:user@mydomain.com;msupn:principal@mydomain.com”. • dns: domain name • eml: email address • msupn: microsoft upn (universal principal name)

validity	Value is a string with a validity period for the certificate.
	• For **TLS** certificates, accepted values are in days from "1d" to "198d".
	  ○ No calculation from one unit to another one is being done: e.g. "3M" is not calculated to "90d".
	  ○ "M" for "months" (like 3M for 3 months) is deprecated and shall not be used.
	  ○ "y" for "year" is also deprecated and shall not be used.
	• For **S/MIME** certificates, accepted values are “1y” or “2y” for a 1- or 2-years validity period, respectively.
	Validity input from the CSR will always be ignored. Value in the parameter or default value is taken. The **default** values are
	• "198d" for TLS and
	• "1y" for S/MIME.

base64	If set to the value “1” then output will be made in base64. Otherwise, if parameter is omitted, normal DER will be the 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 needed. Provided as a helper to avoid resigning requests.
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.
sid	Security identifier. A data structure in binary format that contains a variable number of values. The components of a SID are easier to visualize when SIDs are converted from a binary to a string format by using the following standard notation: S-R-X-Y1-Y2-Yn-1-Yn • S Indicates that the string is a SID • R Indicates the revision level • X Indicates the identifier authority value • Y Represents a series of sub-authority values, where n is the number of values Value in request overrides value given in CSR if both are provided.

4.1.1 Simple PKI Request

Example of a PKCS#10 request using the account “MPKI5000003 - Firma Muster AG” to request a certificate for the product “SwissSign Pro S/MIME E-Mail ID Gold” in the test environment. The binary data has been omitted and represented here as <pkcs10_binary_data>:

```
POST /ws/cmc?account=MPKI5000003%20-
%20Firma%20Muster%20AG&product=SwissSign%20Pro%20S%2FMIME%20E-Mail%20ID%20Gold HTTP/1.1
Host: cmc.pre.swisssign.ch
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 an 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, the 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, 18 Jan 2023 18:36:41 GMT
Content-Length: 4753
Cache-Control: max-age=3600
Expires: Wed, 18 Jan 2023 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 to the product being requested and conform to the Full PKI Request specified in section-3.2 of RFC 5272. 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   # CN attribute
               U.P.IA5STRING "Firstname Lastname"
```

```
    }
   }
   U.C.SET {
    U.C.SEQUENCE {
     U.P.OBJ_ID 1.2.840.113549.1.9.1   # emailAddress attribute
     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   # public key modulus
     U.P.INTEGER 65537   # public key exponent
    }
   }
  }
  C.C.0 {
   U.C.SEQUENCE {
    U.P.OBJ_ID 1.2.840.113549.1.9.14   # extensionRequest
    U.C.SET {
     U.C.SEQUENCE {
      U.C.SEQUENCE {
       U.P.OBJ_ID 2.5.29.17   # subjectAltName
       U.P.OCTETSTRING {
        U.C.SEQUENCE {
         U.P.NODE "firstname.lastname@mydomain.com"
```

4.1.4 New CSR – new key pair

Please note that SwissSign has the policy not to reuse an existing key for a CSR. In such a case the following error message is returned: the public key has already been used to issue another certificate. Please generate a new key pair/CSR.

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 for the registration authority account provided by SwissSign during setup.
host	Specifies the target host: • for testing https://cmc.**pre**.swisssign.ch/ws/cmc • for production https://**cmc.swisssign.ch**/ws/cmc
content_type	• application/pkcs7-mime • application/cmc-revoke
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” must contain a numeric reason code. Allowed revocation reason codes for subscriber certificates are:

- 00=unspecified: No specific reason is given.
- 01=keyCompromise: The private key has been compromised or there is a risk that it has been comprimised.
- 03=affiliationChanged: Subject information has changed, e.g. change of company name or surname.
- 04=superseded: The certificate is replaced by another one.
- 05=cessationOfOperation: Indicates that the certificate is no longer needed for the purpose for which it was issued.
- 09=privilegeWithdrawn: Authorization revoked; a privilege contained within that certificate has been withdrawn.

The following example shows the asn1 structure which revokes a certificate with serial ”**520C60926231E91FECCC2F1EE1157E71E87B9DCC**“ from issuer CA ”CN=**SwissSign RSA SMIME NCP ICA 2022 - 2 - DEMO**, O=**SwissSign AG**, C=**CH**” with reason code ”**00**” (unspecified).

Please note that you must replace the example strings like serial “**520C60926231E91FECCC2F1EE1157E71E87B9DCC**” and the issuer “**SwissSign RSA SMIME NCP ICA 2022 - 2 - DEMO**” 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.C.SEQUENCE {
          U.P.INTEGER 01
```

```
            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 RSA SMIME NCP ICA 2022 - 2 - DEMO"
                }
               }
              }
              U.P.INTEGER 520C60926231E91FECCC2F1EE1157E71E87B9DCC   # serialNumber
              U.P.ENUMERATED 00   # revocation reason
             }
            }
           }
          }
          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 a 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 '520C60926231E91FECCC2F1EE1157E71E87B9DCC' successfully revoked from ' C=CH,O=SwissSign
AG,CN=SwissSign RSA SMIME NCP ICA 2022 - 2 - DEMO ' with reason '0' by Client
certificate S/N= 'E443E9ACEC96B32E8C4141EC5D97563' [ account MPKI5000003%20-
%20Firma%20Muster%20AG ]."
           }
          }
         }
        }
        U.C.SEQUENCE {
        }
        U.C.SEQUENCE {
        }
       }
      }
3081b33081ac3081a902010006082b06010505070701318199308196020100300302010000c818b4365727469666963
617465203564346334666564333638323838653864663263363630316632333932636232366162646463623920287
0726f647563742065762920737563636573736756c6c79207265766f6b65642066726f6d20434120636e3d4341473031
2c6f3d2c633d434820776974682072656173 6f6e203020627920757365722053 4d30312e30003000
     }
    }
    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 for the registration authority account provided by SwissSign during setup.
host	Specifies the target host: • for testing https://cmc.**pre**.swisssign.ch/ws/cmc • for production https://**cmc.swisssign.ch**/ws/cmc
content_type	Must be "application/pkcs7-mime"
data	Contains a revocation request control specifying the issuerName and serialNumber (section 6.9. Get Certificate Control in RFC 5272)

The following example shows the asn1 structure which retrieves a certificate with serial "**520C60926231E91FECCC2F1EE1157E71E87B9DCC**" from issuer CA "CN=**SwissSign RSA SMIME NCP ICA 2022 - 2 - DEMO**, O=**SwissSign AG**, C=**CH**".

Please note that you must replace the example strings like serial "**520C60926231E91FECCC2F1EE1157E71E87B9DCC**" and the issuer "**SwissSign RSA SMIME NCP ICA 2022 - 2 - DEMO**" 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 RSA SMIME NCP ICA 2022 - 2 - DEMO"
                  }
                }
              }
              U.P.INTEGER 520C60926231E91FECCC2F1EE1157E71E87B9DCC   # serial
            }
          }
        }
      }
      U.C.SEQUENCE { }
      U.C.SEQUENCE { }
      U.C.SEQUENCE { }
    }
  }
308187307f307d02010106082b06010505070711316e306c3055310b300906035504061302434831153013060355040
a0c0c53776973735369676e204147312f302d06035504030c2653776973735369676e20506572736f6e616c2053696c
766572204341203230382d20473202100086c35d2303e87b53910cc74c83bef50a0100300030003000
    }
  }
  U.C.SET { }
 }
}
```

6.2 RETRIEVAL RESPONSE MESSAGE

If the request was successful, the server returns an 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, 18 Jan 2023 18:51:16 GMT
Cache-Control: private, no-cache
Pragma: no-cache
Expires: Wed, 18 Jan 2023 19:51:16 GMT
Allowed: POST
Content-Transfer-Encoding: BINARY
Content-Length: 3094
Strict-Transport-Security: max-age=31536000; includeSubDomains
Connection: close
<pkcs7_binary_data>
```

7 ERRORS

If a request fails, an HTTP response with HTTP status code not equal to 200 is returned containing a 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.

```
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 <CMCFailInfo>
     U.C.SEQUENCE {
      U.P.INTEGER 00
     }
     U.P.UTF8STRING <statusString>
    }
   }
  }
 }
 U.C.SEQUENCE { }
 U.C.SEQUENCE { }
}
```

Tables in the following sections list values for <**CMCFailInfo**> and <**statusString**> including the corresponding HTTP status codes.

7.1 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 or application/pkcs7-mime	415
2	No account available.	400
2	Too many accounts specified. Please specify which one you would like to use in the ‘account’ URL parameter.	400
2	There is no available product for you as of now.	400
2	Please specify which product you would 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 [TLS: 1d..198d] or [S/MIME: 1y or 2y]	400
2	Error in request (pkcs10RequestData OR DirectoryNameItem OR cmc OR keyid not unique OR keyType). • The public key has already been used to issue another certificate. Please generate a new keypair/CSR. • DN Attribute value should contain one of '<emailaddress>' • Invalid key size. Expected RSA key size 2048/3072/4096/8192. Request RSA key size '1,024'. • Value is not UTF8String	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
2	Method not allowed.	400
2	URI provided SDN '<dn>' parse error	400
2	URI provided SAN '<san>' parse error	400
2	Invalid value for option suppress_www_domain. It should be suppress_www_domain=true or suppress_www_domain=false	400
2	Unable to request certificate: <error>	400
2	Unable to issue certificate for $serial: <error>	400

7.2 CMC Revoke Errors

CMCFailInfo	statusString	HTTP Status Code
2	No certificate found matching: <serial_no> and <issuer>	400
2	Certificate already revoked	500
2	Certificate does not belong to account MPKI5000003%20-%20Firma%20Muster%20AG	400
2	No Account available: <account> and <serialnumber>	400
2	Invalid revocation reason code {revocation_code}.	500
2	Operator access denied	400

7.3 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 show step-by-step examples with OpenSSL (version 1.1.x and 3.0.x) that can be executed in a Linux environment to request, revoke and retrieve certificates via CMC interface as a client.

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 the 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 1.1.x
openssl pkcs12 \
  -in operator.p12 \
  -out operator.pem \
  -passin pass:abcd \
  -passout pass:efgh
```

```
# OpenSSL 3.0.x
openssl pkcs12 \
  -in operator.p12 \
  -out operator.pem \
  -passin pass:abcd \
  -passout pass:efgh \
  -legacy
```

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.

- CN="Firstname Lastname"
- emailAddress=firstname.lastname@mydomain.com
- Private key password="ijkl"
- Prepare the binary PKCS#10 file for the request.

```
# create certificate signing request (CSR) in binary der format
openssl req \
  -new \
  -newkey rsa:2048 \
  -utf8 \
  -subj "/CN=Firstname Lastname/emailAddress=firstname.lastname@mydomain.com" \
  -addext "subjectAltName=email:firstname.lastname@mydomain.com" \
  -passout "pass:ijkl" \
  -keyout key.pem \
  -outform DER \
  -out request.der
```

8.2.2 Submit and extract the response

The request can be submitted to the CMC interface using an operator certificate. Replace the account name "MPKI5000003 – Firma Muster AG" with your account name and the product name "SwissSign Pro S/MIME E-Mail ID Gold" with the appropriate product name according to chapter 4 which was enabled for your account in URL encoded form.

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. The https call specifies the operator certificate with the "–

cert" parameter (created in step 8.1) and the password with the "–pass" parameter. Replace the password value "efgh" with the appropriate password. Use the encrypted operator cert with the certificate chain and with the password, otherwise the request will fail.

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.

```
# submit certificate signing request (request.der) over cmc/https with client authentication
echo -ne \
   "POST /ws/cmc?account=MPKI5000003%20-
%20Firma%20Muster%20AG&product=SwissSign%20Pro%20S%2FMIME%20E-Mail%20ID%20Gold HTTP/1.1" \
   "\r\nHost: cmc.pre.swisssign.ch" \
   "\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 cmc.pre.swisssign.ch:443 \
  -servername cmc.pre.swisssign.ch \
  -cert operator.pem \
  -pass pass:efgh \
  -quiet \
| sed '1,/^\r$/d' \
> cmc_response.request.der
```

If the request was successful, the cmc_response.request.der file contains the certificate chain including the issued certificate. The following commands extract two files (cert_chain.pem and cert.pem) out of the response.

1. cert_chain.pem: contains the entity certificate and the complete cert chain with the CA certificates

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

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

```
# extract end entity certificate
openssl x509 -in cert_chain.pem -out 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
```

8.2.4 Merge private key with public signed cert to p12

The following command merges the private key and the signed public key into a p12 certificate.

```
# merge key.pem (encrypt private key) with cert_chain.pem into cert.p12 and set new password
openssl pkcs12 \
  -export \
  -in cert_chain.pem \
  -inkey key.pem \
  -passin pass:ijkl \
  -passout pass:mnop \
  -out cert.p12
```

8.3 REVOKE WITH OPENSSL

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

First, get the serial number of the certificate that needs to be revoked. The following command gives the serial number of the certificate in the correct format back:

```
# get serial number out of leaf cert
openssl x509 -noout -serial -in cert.pem | sed 's/serial=//' | tr '[:upper:]' '[:lower:]'
```

Second, get the issuer info out of the certificate. The following command gives the issuer info of the certificate back:

```
# get issuer info out of leaf cert
openssl x509 -noout -issuer -in cert_chain.pem -nameopt sep_multiline
```

Third, 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 starting with “0x” (first step)
- **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 (second step C= ...)
- **organization**: organization from the subject of the issuing CA (second step O = ...)
- **commonName**: CN from the subject of the issuing CA (second step CN = ...)

```
# create openssl config file to revoke the leaf cert
cat <<EOF > revoke.cnf
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:0x520c60926231e91feccc2f1ee1157e71e87b9dcc
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 RSA SMIME NCP ICA 2022 - 2 - DEMO
EOF
```

Fourth, generate the DER file of the request by executing the following statement:

```
# generate PKCS7 revocation request in binary der format
openssl asn1parse -genconf revoke.cnf -out revoke.der
```

Fifth, submit revocation request to the CA:

```
# submit revocation request (revoke.der) over cmc/https with client authentication
echo -ne \
   "POST /ws/cmc?account=MPKI5000003%20-%20Firma%20Muster%20AG HTTP/1.1" \
   "\r\nHost: cmc.pre.swisssign.ch" \
   "\r\nContent-Length: `cat revoke.der | wc -c`" \
   "\r\nContent-Type: application/pkcs7-mime" \
   "\r\nConnection: close" \
   "\r\n\r\n" \
| cat - revoke.der \
| openssl s_client \
  -connect cmc.pre.swisssign.ch:443 \
  -servername cmc.pre.swisssign.ch \
  -cert operator.pem \
  -pass pass:efgh \
  -quiet \
| sed '1,/^\r$/d' \
> cmc_response.revoke.der
```

Sixth, decode the cmc response:

```
# get response out of cmc_response.revoke.der
strings cmc_response.revoke.der
```

8.4 RETRIEVE WITH OPENSSL

The following example shows how the CMC interface should be used to **retrieve** a certificate using OpenSSL.

First, get the serial number of the certificate that needs to be retrieved. The following command gives the serial number of the certificate in the correct format back:

```
# get serial number of cert
openssl x509 -noout -serial -in cert.pem | sed 's/serial=//' | tr '[:upper:]' '[:lower:]'
```

Second, get the issuer info out of the certificate. The following command gives the issuer info of the certificate back:

```
# get issuer info of cert
openssl x509 -noout -issuer -in cert_chain.pem -nameopt sep_multiline
```

Third, 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 starting with “0x” (first step)
- **country**: country from the subject of the issuing CA (second step C= ...)
- **organization**: organization from the subject of the issuing CA (second step O = ...)
- **commonName**: CN from the subject of the issuing CA (second step CN = ...)

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

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

```
# create openssl config file to retrieve the leaf cert
cat <<EOF > retrieve.cnf
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:0x520c60926231e91feccc2f1ee1157e71e87b9dcc
[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 RSA SMIME NCP ICA 2022 - 2 - DEMO
EOF
```

Fourth, generate the DER file of the request by executing the following statement:

```
# generate PKCS7 retrieve request in binary der format
openssl asn1parse -genconf  retrieve.cnf -out retrieve.der
```

Fifth, submit retrieval request to the CA:

```
# submit retrieve request (retrieve.der) over cmc/https with client authentication
echo -ne \
    "POST /ws/cmc?account=MPKI5000003%20-%20Firma%20Muster%20AG HTTP/1.1" \
    "\r\nHost: cmc.pre.swisssign.ch" \
    "\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 cmc.pre.swisssign.ch:443 \
   -servername cmc.pre.swisssign.ch \
   -cert operator.pem \
   -pass pass:efgh \
   -quiet \
| sed '1,/^\r$/d' \
> cmc_response.retrieve.der
```

If the request was successful, the cmc_response.request.der file contains the certificate chain including the issued certificate. The following commands extract two files (cert_chain.pem and cert.pem) out of the response.

1. cert_chain.retrieve.pem: contains the entity certificate and the complete cert chain with the CA certificates

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

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

```
# extract end entity certificate
openssl x509 -in cert_chain.retrieve.pem -out cert.retrieve.pem
```

Depending on your request you must modify the search condition suitable for finding the certificate or extract the pem file manually with an editor from the cert_chain.retrieve.pem.

9 TROUBLESHOOTING

In special cases, certificates cannot be requested even if the CMC interface has been configured with the correct values that you received from us. For such cases, we have listed problems that may cause the application to not be able to connect to us, or if you do connect, you may be rejected by the firewalls (client, server, network and application side) or by our application. In such cases, please check this list first and ensure that none of these cases apply before contacting the support of your CMC gateway provider or SwissSign.

Please note that we do not provide hardware, network, operating system, appliance, or OpenSSL support.

9.1 OPERATOR CERTIFICATE

- Needs to have the whole cert chain (leaf, issuer and root certs) and the encrypted private key.
- Make sure that your application sends the whole chain and not only the leaf cert.
- If your application doesn't recognize the private operator cert, then you maybe must install the private root cert to the applications CA root store.
- Use first the password from the p12 file and test if it works.
- Don’t use the password of the p12 file in any other setting except for the operator cert.

9.2 REQUEST VALUES

- Host entry must be everywhere correct, check that you don't mix the Pre-Prod and PROD values.
- Header setting must be correct, the minimal settings are in our examples.
- Post parameters must be the same as described in this document.
- Check that you have the correct serial number of the certificate that you want to retrieve or revoke.
- Check that you don’t forget the “0x” before the serial number in the retrieve or revocation request.
- Check that you have the correct issuer of the certificate that you want to retrieve or revoke.
- All requests must work with UTF-8 encoding and the request must be URL encoded.
- Check if you have the products available which you are trying to request.
- Check if the account name is correct.
- Check that you request with the correct values and data types.
- Don’t use wrong or own OID's to request public certificates, our application will reject such requests.
- Don't reuse private keys to create new CSR’s, our application will reject such requests.
- Check the error message of the responses to find out why a request fails.
- Check with an asn1 decoder that the binary der file contains the correct data structure and the correct values.

9.3 DOMAIN(S) USED IN THE REQUEST

- Must be validated otherwise the request will fail.
- Check if the domain has expired when a request fails.
- Check that you have no spaces in the TXT record.
- Check that you don't use non-breaking hypes, en dashes, em dashes or similar special dashes in the TXT record.
- If you set CAA entries, then they must be correct and be sure how you set the critical flag.
- Domain(s) must be registered, and the NS servers must be set correctly.
- DNS server should be configured correctly.
- If you set DNSSec, then it must be set correctly (Check with https://dnsviz.net or https://dnssec-analyzer.verisignlabs.com).
- If you use a CNAME record, then check that it points correctly.
- Check with DNS tools to see, if the DNS settings are all correct.
- Check with DNS tools if the DNS servers are responding in an appropriate time (some milliseconds).

9.4 FIREWALL AND NETWORK ISSUES

- Check that you can reach our webUI from the network, if you try to connect to our cmc interface.
- Check that port 443 is not blocked to the URL of the CMC.
- Check that you get the cert from our server and that the chain is not broken, and that the cert is valid.
- Check that no firewalls break the cert chain of our server and if it does so then check with your specialist to verify, if a cmc connection can still be established.
- Capture a TCP dump to examine, if you get our server cert with the cert chain and if your application sends the operator cert chain before you send the post request.
- Check with a TCP dump to verify, if the server and client cert could be verified.

9.5 SYSTEM ISSUES

- System locals need to be set to UTF-8; any other setting may cause problems.
- System date, time and time zone must be set correctly.
- Please check that you have no quotation errors in the examples.

10 APPENDIX A: PRODUCTS

The following standard products are defined on both Pre-Prod and PROD systems.

Product	Legacy CA product names * being any account name string	New URL encoded product names	Validity
SwissSign EV SSL Gold Single-Domain	*-ev-gold	SwissSign%20EV%20SSL%20Gold%20Single-Domain	[1d..198d]
SwissSign EV SSL Gold Multi-Domain	*-ev-gold-multi	SwissSign%20EV%20SSL%20Gold%20Multi-Domain	[1d..198d]
SwissSign OV SSL Gold Wildcard	*-gold-wild	SwissSign%20OV%20SSL%20Gold%20Wildcard	[1d..198d]
SwissSign OV SSL Gold Single-Domain	*-ssl-gold	SwissSign%20OV%20SSL%20Gold%20Single-Domain	[1d..198d]
SwissSign OV SSL Gold Multi-Domain	n/a	SwissSign%20OV%20SSL%20Gold%20Multi-Domain	[1d..198d]
SwissSign OV SSL Gold Multi-Domain Wildcard	*-ssl-gold-multi	SwissSign%20OV%20SSL%20Gold%20Multi-Domain%20Wildcard	[1d..198d]
SwissSign DV SSL Silver Single-Domain	*-ssl-silver	SwissSign%20DV%20SSL%20Silver%20Single-Domain	[1d..198d]
SwissSign DV SSL Silver Multi-Domain	n/a	SwissSign%20DV%20SSL%20Silver%20Multi-Domain	[1d..198d]
SwissSign DV SSL Silver Wildcard	*-silver-wild	SwissSign%20DV%20SSL%20Silver%20Wildcard	[1d..198d]
SwissSign Pro S/MIME E-Mail ID Gold	*-perso-org-gold	SwissSign%20Pro%20S%2FMIME%20E-Mail%20ID%20Gold	1y or 2y
SwissSign Pro S/MIME E-Mail ID Gold with Auth.	*-perso-org-gold-auth	SwissSign%20Pro%20S%2FMIME%20E-Mail%20ID%20Gold%20with%20Auth	1y or 2y
SwissSign Pro S/MIME E-Mail ID Gold RSASSA-PSS	*-perso-org-gold-rsassapss	SwissSign%20Pro%20S%2FMIME%20E-Mail%20ID%20Gold%20RSASSA-PSS	1y or 2y
SwissSign Personal S/MIME E-Mail ID Silver	*-perso-silver-emailonly	SwissSign%20Personal%20S%2FMIME%20E-Mail%20ID%20Silver	1y or 2y
SwissSign S/MIME OV	n/a	SwissSign%20S%2FMIME%20OV	1y or 2y