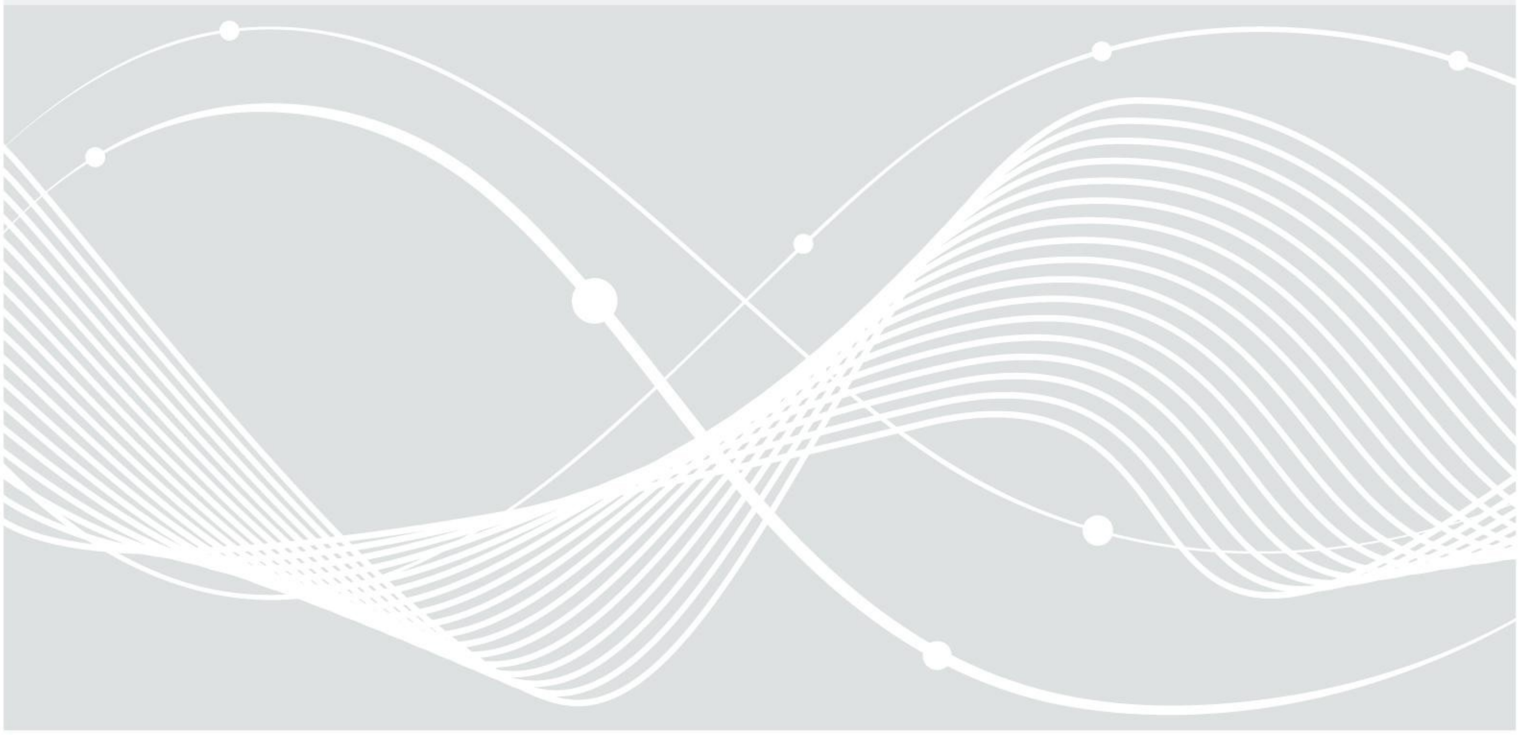
Federal Office
for Information Security

Technical Guideline TR-03129-1

Protocols for the Management of Certificates and CRLs in Public-Key-Infrastructures (PKIs)

Part 1: Common Specifications

Version 1.40



Document history

Version	Date	Description
1.00	2009-11-09	Initial public version.
1.21	2012-11-19	added GeneralMessage, GetRootCertificates and SendRootCertificates
1.3	2017	New author guidelines adapted, Document split into logical parts
1.40	2022	New revised version

Federal Office for Information Security
Post Box 20 03 63
D-53133 Bonn

E-Mail: eid@bsi.bund.de
Internet: https://www.bsi.bund.de

Table of Contents

Tables

1 Introduction

This technical guideline specifies generic PKI-related communication protocols for the distribution of certificates and related data between the entities of a PKI.

1.1 Infrastructures for hierarchical PKIs

A hierarchical PKI can be typically distinguished into a Certificate Authority (CA), Sub-CAs and end users. The Certificate Authority, also called Root-CA, is the trust anchor for all issued certificates in the scope of the PKI. Therefore, the CA creates a self-signed certificate upon which all entities of the PKI have to rely. The CA also signs the certificates of the Sub-CAs, thus placing confidence in them to issue end user certificates. The end users can use these certificates to authenticate each other and to establish secure communication channels.

In general a PKI can define several layers of Sub-CAs. Hereby, a Sub-CA signs the certificate of the Sub-CA on a hierarchical lower level.

1.2 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119]. The key word “CONDITIONAL” is to be interpreted as follows:

CONDITIONAL: The usage of an item is dependent on the usage of other items. It is therefore further qualified under which conditions the item is REQUIRED or RECOMMENDED.

When used in tables (profiles), the key words are abbreviated as shown in Table 1.

Key word	**Equivalent**	**Abbrev.**
MUST / SHALL	REQUIRED	m
MUST NOT / SHALL NOT	–	x
SHOULD	RECOMMENDED	r
MAY	OPTIONAL	o
–	CONDITIONAL	c

Table 1: Key words

1.3 Authentication, Confidentiality, and Integrity Protection

The authentication of the sender and receiver of messages as well as the confidentiality and integrity of the contents of the messages are not considered at the level of these messages. It is assumed that authentication, integrity protection and confidentiality are guaranteed by other measures, for example by using transport layer security (TLS) with server and client authentication according to [TR-03116-4].

Each service provider SHALL use the underlying authentication mechanism to verify whether (and to what extent) the caller is authorised to use the service and deny access otherwise. Security relevant information MUST always be protected by appropriate encryption and integrity protection mechanisms.

2 Implementation as Web Services

The implementation of communication protocols shall be realized as web services using SOAP-messages.

The messages should be in conformance with the attached WSDL files.

Each SOAP-message is immediately responded by a return code at least stating the reception of the message and its syntactical (in-)correctness. In general the content of the SOAP-messages can be processed either synchronously or asynchronously.

If a message is processed synchronously, the processing result (if any) is included in the immediate SOAP response. The result is passed in the output parameter part of the response message.

If the message is processed asynchronously, only a return code and optionally a return code message is immediately sent back to the sender. Once the request has been processed, the result is sent back to the sender (of the original message). The result is passed in the input parameter part of the callback message.

If a message is processed asynchronously and the result is sent back using a callback message, both participating entities must work as a client in order to send a message as well as a server in order to receive a message.

A sender of a message may indicate in the message that they cannot handle callback messages. In this case the receiver of the message may process the message synchronously, i.e. the output parameters of the message must contain the final results of processing the message. Else wise, the receiver informs the sender that the message cannot be handled synchronously by sending an appropriate return code.

Polling mechanisms are not available in this protocol, i.e. in the case of an asynchronous message processing the sender cannot ask the receiver frequently if the results of message processing are available for download using messages of this protocol. Exceptions to this rule may appear, which will then be explicitly indicated and explained.

It is assumed that the SOAP-messages are protected by a TLS-channel with mutual authentication, which requires a prior registration of communication partners including the exchange of TLS-certificates.

2.1 Determination of Web Service URL

Before the entities of a PKI can exchange web service messages, any caller has to obtain knowledge of the web service URL of the recipient of a message. Especially for asynchronous messages it is assumed, that sender and recipient are able to identify each other and know the respective web service URLs. How these URL's are exchanged depends on the particular PKI and is out-of-scope of this document. Possible exchange mechanisms are:

- a directory service provided by a central instance of the PKI,
- bilateral exchange between participants of the PKI that need to interact with each other.

2.2 Messages Format

In the following chapters abstract definitions of the messages used for the protocols will be given. Each message includes an input part which is sent by the client to the server and an output part which is returned in response. In case of an asynchronous communication, two messages are exchanged each containing an in- and output part.

For each message, the following characteristics are described:

- Intended use
- Input parameters

- Output parameters

  → including possible return codes

Unless specified otherwise, each message can be processed by the receiving entity either asynchronously or synchronously.

2.2.1 Common Parameters

The following parameters are frequently used as input or output parameters:

- `callbackIndicator`

  With this parameter the originator of a message informs the receiver if it can handle callbacks as response to this message. If the originator can handle callbacks, this parameter MUST be set to `callback_possible`. In this case, the receiver can decide if it processes this message synchronously or asynchronously. If the receiver processes this message asynchronously, it will send the response using the appropriate callback message. If the originator cannot handle callbacks, this parameter MUST be set to `callback_not_possible`.

  Application Note: If a specific PKI guideline only allows synchronous messaging, this value should be set to `callback_not_possible` per default.

- `messageID`

  This parameter contains the message identifier. It MUST identify the message uniquely within all messages of the originator. If a response message will be sent to the originator as a result of this message, the response message SHALL contain the same `messageID` as the message being responded. Hence, an incoming response message can be assigned to the correct original message. Construction and allocation of the `messageID` is decided by the originator. This parameter is REQUIRED if the originator of the message indicates, that it can handle a callback as response to this message (i.e. parameter `callbackIndicator` = `callback_possible`). It MUST be missing, if the originator of the message indicates, that it cannot handle callbacks as response to this message (i.e. parameter `callbackIndicator` = `callback_not_possible`).

- `responseURL` **(PARTLY DEPRECATED)**

  Application Note: This parameter is deprecated in the context of SPOC-DV communication and SHOULD NOT be used any more. If a message shall be processed asynchronously, the receiver should identify the sender through the TLS-certificate used by the sender for transmitting the request-message. A specific web service URL is stored in advance for each TLS-certificate of a registered DV. Any response message MUST be send to the web service URL belonging to the sender. The parameter may be removed in a future version of this guideline.

  This parameter contains the URL, at which the originator expects the response message to be sent, if the message will be processed asynchronously. This parameter is OPTIONAL if the originator of the message indicates, that it can handle a callback as response to this message (i.e. parameter `callbackIndicator` = `callback_possible`). It MUST be missing, if the originator of the message indicates, that it cannot handle callbacks as response to this message (i.e. parameter `callbackIndicator` = `callback_not_possible`).

- `returnCode`

  Each message MUST be responded immediately by at least a return code in order to report the result of the message processing. Common return codes are given section 2.2.2. PKI specific returnCodes are defined in the application specific guideline of that PKI.

- `returnCodeMessage`

This parameter may include additional information, which are not covered by the already defined return codes. The possible messages should be defined in the application specific guideline of the PKI. The maximum length of a message MUST NOT exceed 1024 bytes.

- `statusInfo`

  This parameter is only used in the callback messages to an asynchronously processed message and contains a status code to report the result of the process. Therefore, the list of status codes matches the list of return codes for most of the entries.

- `statusInfoMessage`

  This parameter may include additional information, which are not covered by the already defined `statusInfo` codes. The specific requirements for these messages should be defined in the application specific guideline of the PKI. The maximum length of a message MUST NOT exceed 1024 bytes.

- `certReq`

  This parameter contains a certificate request. Its structure and encoding must be defined by the specific guideline of the PKI.

- `certificateSeq`

  This parameter contains one or more certificates. Unless stated otherwise, it is REQUIRED if certificates have to be sent with the message. It MUST NOT be part of the message if no certificates will be sent with the message.

- `certReference`

  This parameter contains an application-specific reference to the certificate(s) needed by the sender. The parameter should be supplied by the sender, if the (Sub)-CA needs additional information to retrieve the appropriate certificate chain. This could be e.g. the distinguished name of the issuer and serial number or the public key of the certificate to be verified. The specific values are defined in the sector-specific documents parts of this guideline.

- `cmsContainer`

  This parameter contains the CMS container of `ContentType SignedData` according to [RFC 5652]. The CMS container is used to transmit Request Signer Certificate (RSC). A dedicated certificate, so-called Request Signer Certificate (RSC), is used to sign the CMS container. The specific values are defined in the sector-specific documents parts of this guideline.

2.2.2 Common Return Codes

The following codes are frequently used as `returnCode` or `statusInfo` for several messages:

- `ok_syntax`

  The reception of the SOAP-message is acknowledged. The syntax of the SOAP-message has been verified successfully. The further processing of the message will be done asynchronously. The result of the processing will be sent to the web service URL of the sender of the request-message.

- `ok_received_correctly`

  The message has been received and processed synchronously. No output is generated.

- `ok_cert_available`:
  The message has been processed successfully. The parameter `certificateSeq` or cmsContainer contains one or more certificates.

- `failure_syntax`

  The received SOAP-message is syntactically not correct.

- `failure_incorrect_request`

  The received application-specific content of the request-message was incorrectly formatted and could not be processed by the application.
- `failure_synchronous_processing_not_possible`

  The sender has indicated that he does not accept callback messages, hence the message must be processed synchronously by the receiver. But the receiver cannot process this message synchronously. In this case further procedures must be determined by organisational measures.
- `failure_messageID_unknown`

  The contained `messageID` cannot be matched with any message formerly sent.
- `failure_other_error`

  An error occurred which is not covered by an already defined `returnCode` or `returnCodeMessage` (or `statusInfo` or `statusInfoCode`).
- `failure_internal_error`

  This code should only be returned, if an unexpected error occurred within the server/ software of the sender of this messsage, e.g. in case the application hangs or crashes.
- `failure_cert_not_available`

  The corresponding message has been processed correctly but the requested certificates are not available.
- `failure_eContentType`

  This code should be returned, if the `eContentType` of `cmsContainer` does not exist or if it does not match the `eContent`.
- `failure_expired`

  The certificate needed to verify the outer signature is expired.
- `failure_inner_signature`

  The verification of the signature of the transported certificates in the CMS container failed.
- `failure_outer_signature`

  The verification of the signature of the CMS container failed..

2.3 Default Messages

The following messages provide for a generic communication framework within and between Public Key Infrastructures.

2.3.1 GetCertificates

This message is sent by an entity, who is trying to validate a certificate, to the issuing (Sub)-CA where the certificate has been issued in order to retrieve all relevant certificates required for the verification. The message requests the (Sub-)CA to return a list of certificates to be used for validation. Unlike `RequestCertificate` (see section 2.3.2), `GetCertificates` is not used as request for issuing a certificate.

If the parent (Sub-)CA processes this message asynchronously, it MUST respond using the `SendCertificates` message.

Input parameters:

- `callbackIndicator` **(REQUIRED)**
- `messageID` **(CONDITIONAL)**
- `responseURL` **(DEPRECATED), (CONDITIONAL)**
- `certReference` **(REQUIRED)**

  This parameter contains an application-specific reference to the certificate(s) needed by the sender. The parameter MUST be used by the sender to request the appropriate certificate chain. This could be e.g. the distinguished name of the issuer and serial number of the certificate or the public key of the certificate to be verified.

Output parameters:

- `certificateSeq` **(CONDITIONAL)**
- `returnCode` **(REQUIRED)**

  The following return codes are possible:

  - `failure_cert_not_available`
  - `failure_internal_error`
  - `failure_other_error`
  - `failure_synchronous_processing_not_possible`
  - `failure_syntax`
  - `ok_cert_available`
  - `ok_syntax`
- `returnCodeMessage` **(OPTIONAL)**

2.3.2 RequestCertificate

This message is used by an end-user or by a Sub-CA for requesting the generation of a new certificate for one of its keys from a Sub-CA or the Root-CA, respectively. If the parent (Sub-)CA processes this message asynchronously, it MUST respond using the `SendCertificates` message.

Input parameters:

- `callbackIndicator` **(REQUIRED)**
- `messageID` **(CONDITIONAL)**
- `responseURL` **(DEPRECATED), (CONDITIONAL)**
- `certReq` **(REQUIRED)**

  This parameter contains the certificate request. Its construction and encoding is PKI application specific and should be defined in the PKI specific guideline.

Output parameters:

- `certificateSeq` **(CONDITIONAL)**
- `returnCode` **(REQUIRED)**

  The following return codes are possible:

  - `failure_incorrect_request`
  - `failure_internal_error`
  - `failure_other_error`
  - `failure_synchronous_processing_not_possible`
  - `failure_syntax`
  - `ok_cert_available`
  - `ok_syntax`
- `returnCodeMessage` **(OPTIONAL)**

2.3.3 SendCertificates

If a Root-CA or a Sub-CA processes one of the messages `RequestCertificate` or `GetCertificates` asynchronously, it uses the response message `SendCertificates` to communicate the result of its processing.

This message can also be used to notify registered entities about the availability of new certificates. In this case the `messageID` MUST be omitted.

This message is responded by a return code only.

Input parameters:

- `messageID` **(CONDITIONAL)**
- `statusInfo` **(REQUIRED)**

  The following status codes are possible:
  - `failure_incorrect_request`
  - `failure_internal_error`
  - `failure_other_error`
  - `failure_syntax`
  - `ok_cert_available`
- `statusInfoMessage` **(OPTIONAL)**
- `certificateSeq` **(CONDITIONAL)**

Output parameters:

- `returnCode` **(REQUIRED)**

  The following return codes are possible:
  - `failure_internal_error`
  - `failure_messageID_unknown`
  - `failure_other_error`
  - `failure_syntax`
  - `ok_received_correctly`
- `returnCodeMessage` **(OPTIONAL)**

2.4 Optional Messages

2.4.1 SendRSCCert

This message is sent by one PKI entity to another in order to submit the new Request Signer Certificate (RSC) intended to replace the previously submitted RSC. Request Signer Certificates may be used to sign CMS container which transport securely other objects like certificates or certificate signing requests necessary for PKI communication.

The certificate shall be shipped within a CMS container of `ContentType SignedData`. To sign the CMS container, a valid RSC shall be used.

The use of this webservice message is defined in the sector-specific documents parts of this guideline.

Input parameters:

- `cmsContainer` **(REQUIRED)**

Output parameters:

- `returnCode` **(REQUIRED)**

  The following return codes are possible:

  - `failure_eContentType`
  - `failure_expired`
  - `failure_inner_signature`
  - `failure_internal_error`
  - `failure_other_error`
  - `failure_outer_signature`
  - `failure_syntax`
  - `ok_received_correctly`
- `returnCodeMessage` **(OPTIONAL)**

A WSDL and XML Scheme specifications

The corresponding XML scheme definitions and WSDL descriptions are part of this specification and provided as separate files.

Reference Documentation

RFC 2119	Bradner, Scott: Key words for use in RFCs to indicate requirement levels, 1997
TR-03116-4	BSI: TR-03116 Kryptographische Vorgaben für Projekte der Bundesregierung Teil 4: Kommunikationsverfahren in Anwendungen, 2022
RFC 5652	Housley, Russell: Cryptographic Message Syntax (CMS), 2009