

---

# **INTERFACE SPECIFICATION**

**for**

**sign-me API**

**Version: 2.8**

**Revision: 9420**

**Date: 2022-11-15**

**Bundesdruckerei GmbH ©2022**

---

# Contents

<b>1</b>	<b>Introduction</b>	<b>7</b>
1.1	Purpose . . . . .	7
1.2	Audience . . . . .	7
<b>2</b>	<b>Use Cases at the API</b>	<b>8</b>
2.1	Overview . . . . .	8
2.2	Electronic Remote Signing . . . . .	9
2.2.1	Basic Signing of Documents using API / Webapplication . . . . .	9
2.2.2	Advanced Signing of Documents using API / Webapplication . . . . .	10
2.2.3	Qualified Signing of Documents using API / Webapplication . . . . .	11
2.2.4	Basic Signing of Documents using API . . . . .	13
2.2.5	Advanced Signing of Documents using API . . . . .	14
2.2.6	Qualified Signing of Documents using API . . . . .	16
2.2.7	Basic Signing of Digests using API / Webapplication . . . . .	18
2.2.8	Advanced Signing of Digests using API / Webapplication . . . . .	18
2.2.9	Qualified Signing of Digests using API / Webapplication . . . . .	18
2.2.10	Basic Signing of Digests using API . . . . .	18
2.2.11	Advanced Signing of Digests using API . . . . .	20
2.2.12	Qualified Signing of Digests using API . . . . .	21
2.3	Electronic Remote Sealing . . . . .	23
2.3.1	Qualified Sealing of Digests using API . . . . .	23
2.3.2	Sample Code and Testing . . . . .	25
2.4	Registration and Identification . . . . .	26
2.4.1	Registration using sign-me Webapplication . . . . .	26
2.4.2	Registration using API . . . . .	26
2.4.3	Identification based on Communication . . . . .	27
2.4.4	Identification based on Verification Process . . . . .	28
2.5	Authentication . . . . .	29
2.5.1	Partner Login Session Handling . . . . .	29
2.5.2	User Login Session Handling . . . . .	30
2.6	Fast Track Use Cases . . . . .	32
2.6.1	PoS Fast Track Registration, Identification and Signing . . . . .	34
2.6.2	WEB Fast Track Registration, Identification and Signing . . . . .	39
<b>3</b>	<b>Service and Access Points</b>	<b>41</b>
3.1	Service Overview: SignMeApiV2 . . . . .	41
3.2	Access Points for SignMeApiV2 . . . . .	41
3.2.1	Access Point for SignMeApiV2ReferenceTestSystem . . . . .	41
3.2.2	Access Point for SignMeApiV2LiveSystem . . . . .	42
<b>4</b>	<b>Operations of the API</b>	<b>43</b>
4.1	Overview . . . . .	43
4.2	CreateSignatureProcess . . . . .	43
4.3	UpdateSignatureProcess . . . . .	46
4.4	GetSignerStatus . . . . .	47

4.5	CancelSignatureProcess	50
4.6	GetSignatureProcess	51
4.7	GetSignatureTemplate	53
4.8	CreateRegistrationProcess	54
4.9	UpdateRegistrationProcess	55
4.10	GetRegistrationProcess	57
4.11	RegisterIdentity	58
4.12	GetIdentity	59
4.13	ConfirmCommunication	60
4.14	CreateIdentityVerificationProcess	61
4.15	CancelIdentityVerificationProcess	63
4.16	GetIdentityVerificationProcess	64
4.17	ManageIdentity	65
4.18	ManageSubscription	66
4.19	ManageToken	68
4.20	GetCertificate	69
4.21	GetCertificateTemplate	70
4.22	GetIdentificationTemplate	70
4.23	GetToken	71
4.24	GetAuthenticationTokenTypesForIdentity	72
4.25	GetTokensForIdentity	73
4.26	RevokeCertificateOfTokenActivationProcess	75
4.27	AuthorizeProcess	76
4.28	AuthorizeProcessStep2	77
4.29	Login	78
4.30	LoginStep2	79
4.31	RefreshLogin	80
4.32	Logout	81
4.33	GetVersion	81
<b>5</b>	<b>Structured Datatypes of the API</b>	<b>83</b>
5.1	AccessTokenGrant	83
5.2	AdditionalData	84
5.3	Address	85
5.4	AuthenticationTokenSelector	86
5.5	Certificate	87
5.6	CertificateTemplate	89
5.7	ErrorInfo	92
5.8	GenericDate	93
5.9	IdentificationTemplate	94
5.10	Identity	99
5.11	IdentitySelector	103
5.12	IdentityVerificationProcess	104
5.13	ItemProcessingResult	109
5.14	ItemToBeProcessed	110
5.15	ManageIdentityChangeSet	112
5.16	ManageTokenParameters	115
5.17	ManageTokenResult	116
5.18	NameBlobValue	117
5.19	NameStringValue	117
5.20	Organization	118

5.21	Person . . . . .	122
5.22	Place . . . . .	126
5.23	PwChangeSet . . . . .	127
5.24	RedirectURLs . . . . .	127
5.25	RegistrationProcess . . . . .	128
5.26	RoleAssociation . . . . .	130
5.27	SecurityHeader . . . . .	130
5.28	SignatureParameters . . . . .	131
5.29	SignatureProcess . . . . .	135
5.30	SignatureTemplate . . . . .	139
5.31	SignerStatus . . . . .	143
5.32	Subscription . . . . .	151
5.33	Token . . . . .	152
5.34	TokenActivationProcess . . . . .	155
5.35	VersionAndDate . . . . .	159
<b>6</b>	<b>Simple Datatypes of the API</b>	<b>160</b>
6.1	AccessToken . . . . .	160
6.2	ActivationState . . . . .	160
6.3	AdministrativeState . . . . .	160
6.4	AuthenticationQualityLevel . . . . .	161
6.5	AuthenticationTokenType . . . . .	161
6.6	BCP47LanguageTag . . . . .	161
6.7	CertificateType . . . . .	162
6.8	Coins . . . . .	163
6.9	ContentViewURL . . . . .	163
6.10	Currency . . . . .	164
6.11	DTrustAntragsID . . . . .	164
6.12	DateString . . . . .	164
6.13	Duration . . . . .	164
6.14	EmailAddress . . . . .	165
6.15	ErrorId . . . . .	165
6.16	ExternalProcessKey . . . . .	166
6.17	ExternalProcessURL . . . . .	166
6.18	GenericAuthenticationData . . . . .	166
6.19	HugeBlob . . . . .	167
6.20	ICAOCountryName . . . . .	167
6.21	IdentType . . . . .	167
6.22	IdentityVerificationProcessURL . . . . .	167
6.23	IdentityVerificationState . . . . .	167
6.24	IdentityVerificationType . . . . .	168
6.25	IfRevision . . . . .	171
6.26	IpAddressNumeric . . . . .	171
6.27	ItemName . . . . .	171
6.28	LargeBlob . . . . .	171
6.29	ManageTokenAction . . . . .	171
6.30	ManageTokenStatus . . . . .	172
6.31	MediumBlob . . . . .	172
6.32	ObjectId . . . . .	172
6.33	PIN . . . . .	172
6.34	Password . . . . .	173

6.35	PhoneNumber	173
6.36	PlaceFreetextInfo	173
6.37	RegistrationProcessURL	173
6.38	RegistrationState	174
6.39	RoleName	174
6.40	SignatureFormat	174
6.41	SignatureProcessURL	175
6.42	SignatureState	175
6.43	SignatureType	176
6.44	SmallBlob	180
6.45	SmsPin	180
6.46	StreetAddress	181
6.47	SubscriptionState	181
6.48	TinyBlob	181
6.49	TokenActivationProcessURL	181
6.50	TokenCapability	181
6.51	UTF8String1024	182
6.52	UTF8String128	182
6.53	UTF8String180	182
6.54	UTF8String20	182
6.55	UTF8String200	182
6.56	UTF8String256	183
6.57	UTF8String32	183
6.58	UTF8String40	183
6.59	UTF8String400	183
6.60	UTF8String4096	183
6.61	UTF8String512	183
6.62	UTF8String80	183
6.63	Username	183
6.64	ZipCode	184
<b>7</b>	<b>Language and Platform Specific Access Modules</b>	<b>185</b>
7.1	Motivation	185
7.2	Java 8 Access Module	185
7.3	C# Access Module	185
<b>8</b>	<b>Country Codes</b>	<b>186</b>
<b>9</b>	<b>Error Codes</b>	<b>187</b>

# Revision History

Date	IF	Done
2016/12/08	2.0	Initial Version for basic and advanced signing.
2019/04/02	-2.6	[Removed from history.]
2019/05/03	2.6 rev. 8345	Added UpdateSignatureProcess operation (see <a href="#">4.3</a> ) for increased flexibility for signature container creation in connection with ad-hoc certificates (use-case specific short-term certificates) and digest signing. The subject signature certificate to be used in a specific instance of signature process is available directly in the response to createSignatureProcess.
2019/10/11	2.7	RegistrationProcess allows to re-use sign-me UI for user registration (see <a href="#">4.8</a> and RegistrationProcessSample.java). The optional Shadow Account concept allows exclusive user accounts, which have no password and no sign-me self-service UI access (www.sign-me.de). The user account is exclusively useable for the creating partner and his users (see FastTrackShadowAccountSample.java and RegisterPersonShadowAccountSample.java). This feature is restricted to partners who do reliable user authentication in their application (contact <a href="mailto:sales@d-trust.net">sales@d-trust.net</a> ). Initial registration and create of identification process can be made in one step. See documentation of CreateIdentityVerificationProcess with new optional parameter "person" for initial registration data (see <a href="#">4.14</a> ). Added CMS/CaDES signature types which automatically include qualified timestamps, provide OCSP responses/CRLs for certificate chain (see table <a href="#">6.17</a> , Certificate <a href="#">5.5</a> , SignatureProcess <a href="#">5.29</a> ).
2020/05/12	2.7 rev. 9280	Added ContentViewURL attribute as part of ItemToBeProcessed and ItemProcessingResult for digest signing (see <a href="#">4.2</a> ); Added new PADES-B-B/T/LT signature types; LT contains qualified D-TRUST timestamp and LTV information; Batch signing for up to 16 PDF documents; support for subscription model
2021/02/08	2.7 rev. 9330	External user id and pin can be managed (see <a href="#">4.17</a> ); token version (see <a href="#">5.33</a> )
2021/05/05	2.7 rev. 9340	Short-term signature process history possible in signer-status (see <a href="#">4.4</a> )
2021/11/03	2.7 rev. 9370	Additional Data for Identities Registration and Identification (see <a href="#">5.10</a> )
2022/05/19	2.7 rev. 9380	Removed obsolete token activation operations from specification.
2022/10/25	2.8 rev. 9420	Sealing signature types and use case showing use of sealing feature (see <a href="#">6.20</a> ). Clarification of valid Email address characters (see <a href="#">6.14</a> ).

# 1 Introduction

## 1.1 Purpose

Purpose of this document is to specify the interface of the sign-me system or sign-me API. Partners of the sign-me system use the sign-me API to integrate the remote (server) signing functionality into their business processes and workflows. There is only the partner with the partner application, which is involved at the sign-me API. The partner is an organization (like e.g. a bank or telecommunication service provider) which produces PDF/A documents to be signed by its users over the Webapplications of the partner.

The user is sitting at the Webapplication of the partner, and will be redirected to the

The API was available for supporting token activation on ID cards and client signing since 2012, and is now expanded for remote server signing and all needed supporting use cases like registration and identification.

The specification consists of this human readable PDF document and the referenced WSDL/XSD specification (see [\[WSDL\]](#) and [\[XSD\]](#)).

Section [2](#) details and describes all main use cases, which are possible using the sign-me API.

Section [3](#) describes all available access points of the sign-me system.

Section [4](#) "Operations" describes operation of the API starting from most interesting for daily use. Section [5](#) "Structured Datatypes", and [6](#) "Simple Datatypes" describe the available structured datatypes and simple datatypes of the API in alphabetical order. As far as possible all elements of the WSDL/XSD specification are explained in these sections. Please consult the WSDL/XSD specification (see [\[WSDL\]](#) and [\[XSD\]](#)) in case of further details are needed.

## 1.2 Audience

The document is intended to be read by integrators, system architects, and developers of the partner applications.

The use case part (see [2](#)) of the document is also suitable for business process designers, which need to conceive, where to integrate the sign-me functions into the workflows of the partner application.

## 2 Use Cases at the API

### 2.1 Overview

The use cases are specified in significant detail in the following sections. The use cases are written in a self-contained manner as far as possible, so they can be read one at a time, and get the overall picture of the functions involved.

In order to get the complete understanding of operations and parameters used at the sign-me API the reader needs to refer to the Operations section of the document (see section 4), for each of the steps in the use cases. Therefore, important API operations are directly referenced in the use case steps.

The main use cases at the sign-me API can be categorized in the following functional areas:

- Electronic Remote Signing
  - Signing of Documents
    - \* Basic Signing of Documents using API / Webapplication
    - \* Advanced Signing of Documents using API / Webapplication
    - \* Qualified Signing of Documents using API / Webapplication
    - \* Basic Signing of Documents using API
    - \* Advanced Signing of Documents using API
    - \* Qualified Signing of Documents using API
  - Signing of Digests (Hashes)
    - \* Basic Signing of Digests using API / Webapplication
    - \* Advanced Signing of Digests using API / Webapplication
    - \* Qualified Signing of Digests using API / Webapplication
    - \* Basic Signing of Digests using API
    - \* Advanced Signing of Digests using API
    - \* Qualified Signing of Digests using API
- Electronic Remote Sealing
  - Qualified Sealing of Digests using API
- Registration and Identification
  - Registration using Webapplication
  - Registration using API
  - Identification based on Communication
  - Identification based on Verification Process
- Authentication
  - Partner login



- User login
- Fast Track Use Cases
  - PoS Fast Track Registration, Identification and Signing
  - WEB Fast Track Registration, Identification and Signing

## 2.2 Electronic Remote Signing

### 2.2.1 Basic Signing of Documents using API / Webapplication

#### Summary

This use case shows how an embedded PDF signature can be made on a document with the sign-me system. The signature token which is used for signing is certified by a basic D-TRUST CA.

There is only the partner with the partner application, which is involved at the sign-me API. The partner is an organization (like a bank or telecommunication service provider) which produces PDF/A documents to be signed by its users over the Webapplications of the partner.

The user is sitting at the Webapplication of the partner, and will be redirected to the sign-me front-end. Then the user will authorize the signature at the sign-me front-end.

#### Main Scenario

Preconditions:

- Partner application has already connected to sign-me API using the *SignMeApiV2Client* and the basic authentication credentials as communicated by D-TRUST.
- Partner is already logged in with his partner credentials.
- User is already registered and identified in State `COMMUNICATION_VERIFIED` (see section 6.6).

Steps:

1. The User sits in front of a web application of the partner, and in this stage of the process, the user shall sign a document using a basic electronic signature. The user chooses the option to sign the document using sign-me.
2. The partner application asks the user to enter his sign-me username. The user is already a registered and identified user and enters his username.
3. The partner application asks sign-me with the user's username to check the identity verification state and name of the user using the `getIdentity` call (see section 4.12). As supposed the user is in `COMMUNICATION_VERIFIED` (see section 6.6).
4. The partner application creates a signature process, transferring the document to be signed, and the signature type to be used (in this case, a basic signature: `S_-BAS_DOC_PADES.1`) at sign-me API (see section 4.2 for all parameters). sign-me returns a unique *signatureProcessLink*.
5. Partner redirects the user session to sign-me using the *signatureProcessLink*.

6. sign-me presents a series of HTML forms for the user. The user enters his credentials, checks the document to be signed, and finally authorizes the signature in the sign-me web application. sign-me technically signs the document on behalf of the user, applying the default basic signature token of the user.
7. sign-me presents the resulting signed document for download to the user. After acknowledging the download option, the user is redirected to the http-Link *redirectURLOnOk*, which the partner provided in step 4.
8. The partner application checks the status of the signature process. The state is OK\_UNVERIFIED (see section 6.13), which means the document is correctly signed. The partner application retrieves the signed document at sign-me API.

## 2.2.2 Advanced Signing of Documents using API / Webapplication

### Summary

This use case shows how an embedded PDF signature can be made on a document with the sign-me system. The signature token which is used for signing is certified by an advanced D-TRUST CA.

As with basic signing there is only the partner with the partner application, which is involved at the sign-me API. The partner is an organization (like a bank or telecommunication service provider) which produces PDF/A documents to be signed by its users over the Webapplications of the partner.

The user is sitting at the Webapplication of the partner, and will be redirected to the sign-me front-end. Then the user will authorize the signature at the sign-me front-end.

### Main Scenario

Preconditions:

- Partner application has already connected to sign-me API using the *SignMeApiV2Client* and the basic authentication credentials as communicated by D-TRUST.
- Partner is already logged in with his partner credentials.
- User is already registered and identified in State LAWFULLY\_VERIFIED\_HIGH (see section 6.6).

Steps:

1. The User sits in front of a web application of the partner, and in this stage of the process, the user shall sign a document using an advanced electronic signature. The user chooses the option to sign the document using sign-me.
2. The partner application asks the user to enter his sign-me username. The user is already a registered and identified user and enters his username.
3. The partner application asks sign-me with the user's username to check the identity verification state and name of the user using the *getIdentity* call (see section 4.12). As supposed the user is in LAWFULLY\_VERIFIED\_HIGH (see section 6.6).
4. The partner application creates a signature process, transferring the document to be signed, and the signature type to be used (in this case, an advanced signature: S\_ADV\_DOC\_PADES\_1) at sign-me API (see section 4.2 for all parameters). sign-me returns a unique *signatureProcessLink*.

5. Partner redirects the user session to sign-me using the *signatureProcessLink*.
6. sign-me presents a series of HTML forms for the user. The user enters his credentials, checks the document to be signed, and finally authorizes the signature in the sign-me web application. sign-me technically signs the document on behalf of the user, applying the default advanced signature token of the user.
7. sign-me presents the resulting signed document for download to the user. After acknowledging the download option, the user is redirected to the http-Link *redirectURLOnOk*, which the partner provided in step 4.
8. The partner application checks the status of the signature process using the *getSignatureProcess* call (see section 4.6). The state is *OK\_UNVERIFIED* (see section 6.13), which means the document is correctly signed. The partner application retrieves the signed document at sign-me API.
9. As an option, the partner application can verify the signature and retrieve the verification report using the *getSignatureProcess* call (see section 4.6). Then the state is *OK\_VERIFIED* (see section 6.13), which means the document is correctly signed and all cryptographic information and certificates could be successfully verified by a technical verification service. The partner application retrieves the verification report at sign-me API and archives it for further reference.

#### **Variation: User Identity not lawfully verified**

Preconditions:

- Like in Main Scenario, except the User is already registered but not identified in state *LAWFULLY\_VERIFIED\_HIGH*, only *COMMUNICATION\_VERIFIED* (see section 6.6).

Steps modifying Main Scenario Step 3:

1. The partner application asks sign-me with the user's username to check the identity verification state and name of the user using the *getIdentity* call (see section 4.12). The user turns out to be in *COMMUNICATION\_VERIFIED* (see section 6.6).
2. The partner application needs to create an identity verification process to bring the user into state *LAWFULLY\_VERIFIED\_HIGH*. The scenario proceeds like in 2.4.4.

### **2.2.3 Qualified Signing of Documents using API / Webapplication**

#### **Summary**

This use case shows how an embedded PDF signature can be made on a document with the sign-me system. The signature token which is used for signing is certified by a qualified D-TRUST CA.

As with basic or advanced signing there is only the partner with the partner application, which is involved at the sign-me API. The partner is an organization (like a bank or telecommunication service provider) which produces PDF/A documents to be signed by its users over the Webapplications of the partner.

The user is sitting at the Webapplication of the partner, and will be redirected to the sign-me front-end. Then the user will authorize the signature at the sign-me front-end by means of an authorization code received on the user's mobile phone.

## Main Scenario

Preconditions:

- Partner application has already connected to sign-me API using the *SignMeApiV2Client* and the basic authentication credentials as communicated by D-TRUST.
- Partner is already logged in with his partner credentials.
- User is already registered and identified in State `LAWFULLY_VERIFIED_HIGH` (see section 6.6).

Steps:

1. The User sits in front of a web application of the partner, and in this stage of the process, the user shall sign a document using a qualified electronic signature (QES). The user chooses the option to sign the document using sign-me.
2. The partner application asks the user to enter his sign-me username. The user is already a registered and identified user and enters his username.
3. The partner application asks sign-me with the user's username to check the identity verification state and name of the user using the `getIdentity` call (see section 4.12). As supposed the user is in `LAWFULLY_VERIFIED_HIGH` (see section 6.6).
4. The partner application creates a signature process, transferring the document to be signed, and the signature type to be used (in this case, a qualified signature: `S_QES_DOC_PADES_1`) at sign-me API (see section 4.2 for all parameters). sign-me returns a unique *signatureProcessLink*.
5. Partner redirects the user session to sign-me using the *signatureProcessLink*.
6. sign-me presents a series of HTML forms for the user. The user enters his credentials, and sign-me will send an authorization code to the registered mobile phone of the user.
7. The user checks the document to be signed, and finally authorizes the signature in the sign-me web application by entering the authorization code. sign-me technically signs the document on behalf of the user, applying the default qualified signature token of the user.
8. sign-me presents the resulting signed document for download to the user. After acknowledging the download option, the user is redirected to the http-Link *redirectURLOnOk*, which the partner provided in step 4.
9. The partner application checks the status of the signature process using the `getSignatureProcess` call (see section 4.6). The state is `OK_UNVERIFIED` (see section 6.13), which means the document is correctly signed. The partner application retrieves the signed document at sign-me API.
10. As an option, the partner application can verify the signature and retrieve the verification report using the `getSignatureProcess` call (see section 4.6). Then the state of is `OK_VERIFIED` (see section 6.13), which means the document is correctly signed and all cryptographic information and certificates could be successfully verified by a technical verification service. The partner application retrieves the verification report at sign-me API and archives it for further reference.

### Variation: User Identity not lawfully verified

Preconditions:

- Like in Main Scenario, except the User is already registered but not identified in state `LAWFULLY_VERIFIED_HIGH`, only `COMMUNICATION_VERIFIED` (see section 6.6).

Steps modifying Main Scenario Step 3:

1. The partner application asks sign-me with the user's username to check the identity verification state and name of the user using the `getIdentity` call (see section 4.12). The user turns out to be in `COMMUNICATION_VERIFIED` (see section 6.6).
2. The partner application needs to create an identity verification process to bring the user into state `LAWFULLY_VERIFIED_HIGH`. The scenario proceeds like in 2.4.4.

### 2.2.4 Basic Signing of Documents using API

This feature needs contractual extensions and can only be used if activated by D-Trust, please contact [sales@d-trust.net](mailto:sales@d-trust.net).

#### Summary

This use case shows how an embedded PDF signature can be made on a document with the sign-me system. The signature token which is used for signing is certified by a basic D-TRUST CA.

In this use case, there are two roles at the API, which the partner application needs to handle.

- The partner with extident contract providing the partner application, which is involved at the sign-me API like in signing use case with API and sign-me Webapplication 2.2.1.
- The user is sitting at the partner application, and will enter his credentials at the partner application. The partner application will operate at the sign-me API using these credentials in the "USER" role.

#### Main Scenario

Preconditions (same as in 2.2.1):

- Partner application has already connected to sign-me API using the *SignMeApiV2Client* and the basic authentication credentials as communicated by D-TRUST.
- Partner is already logged in with his partner credentials.
- User is already registered and identified in State `COMMUNICATION_VERIFIED` (see section 6.6).

Steps:

1. The User sits in front of a web application of the partner, and in this stage of the process, the user shall sign a document using a basic electronic signature. The user chooses the option to sign the document using sign-me.

2. The partner application asks the user to enter his sign-me username. The user is already a registered and identified user and enters his username.
3. The partner application asks sign-me API with the user's username to check the identity verification state and name of the user using the `getIdentity` call (see section 4.12). As supposed the user is in `COMMUNICATION_VERIFIED` (see section 6.6).
4. The partner application creates a signature process, transferring the document to be signed, and the signature type to be used (in this case, a basic signature: `S.BAS_DOC.PADES.1`) at sign-me API (see section 4.2 for all parameters). sign-me returns a unique *signatureProcessId*.
5. Partner application presents a series of HTML forms for the user. The user enters his credentials and the partner application logs the user into the sign-me system over the API using the operation `Login` (see section 4.29). sign-me returns an access token for the user.
6. The user checks the document to be signed, and finally authorizes the signature in the partner application. The Partner application forwards the authorization to sign-me API using the access token of the user and operations `AuthorizeProcess/AuthorizeProcessStep2` (see section 4.27). sign-me then technically signs the document on behalf of the user, applying the default basic signature token of the user.
7. The partner application checks the status of the signature process. The state is `OK_UNVERIFIED` (see section 6.13), which means the document is correctly signed. The partner application retrieves the signed document at sign-me API and presents it to the user for an optional download.
8. As an option, the partner application can verify the signature and retrieve the verification report using the `getSignatureProcess` call (see section 4.6). Then the state is `OK_VERIFIED` (see section 6.13), which means the document is correctly signed and all cryptographic information and certificates could be successfully verified by a technical verification service. The partner application retrieves the verification report at sign-me API and archives it for further reference.

### 2.2.5 Advanced Signing of Documents using API

This feature needs contractual extensions and can only be used if activated by D-Trust, please contact [sales@d-trust.net](mailto:sales@d-trust.net).

#### Summary

This use case shows how an embedded PDF signature can be made on a document with the sign-me system. The signature token which is used for signing is certified by an advanced D-TRUST CA.

In this use case, there are two roles at the API, which the partner application needs to handle.

- The partner with extident contract providing the partner application, which is involved at the sign-me API like in signing use case with API and sign-me Webapplication 2.2.2.

- The user is sitting at the partner application, and will enter his credentials at the partner application. The partner application will operate at the sign-me API using these credentials in the "USER" role.

## Main Scenario

Preconditions (same as in 2.2.2):

- Partner application has already connected to sign-me API using the *SignMeApiV2Client* and the basic authentication credentials as communicated by D-TRUST.
- Partner is already logged in with his partner credentials.
- User is already registered and identified in State `LAWFULLY_VERIFIED_HIGH` (see section 6.6).

Steps:

1. The User sits in front of a web application of the partner, and in this stage of the process, the user shall sign a document using an advanced electronic signature. The user chooses the option to sign the document using sign-me.
2. The partner application asks the user to enter his sign-me username. The user is already a registered and identified user and enters his username.
3. The partner application asks sign-me with the user's username to check the identity verification state and name of the user using the `getIdentity` call (see section 4.12). As supposed the user is in `LAWFULLY_VERIFIED_HIGH` (see section 6.6).
4. The partner application creates a signature process, transferring the document to be signed, and the signature type to be used (in this case, an advanced signature: `S_ADV_DOC_PADES_1`) at sign-me API (see section 4.2 for all parameters). sign-me returns a unique *signatureProcessId*.
5. Partner application presents a series of HTML forms for the user. The user enters his credentials and the partner application logs the user into the sign-me system over the API using the operation `Login` (see section 4.29). sign-me returns an access token for the user.
6. The user checks the document to be signed, and finally authorizes the signature in the partner application. The Partner application forwards the authorization to sign-me API using the access token of the user and operations `AuthorizeProcess/AuthorizeProcessStep2` (see section 4.27). sign-me then technically signs the document on behalf of the user, applying the default advanced signature token of the user.
7. The partner application checks the status of the signature process using the `GetSignatureProcess` call (see section 4.6). The state is `OK_UNVERIFIED` (see section 6.13), which means the document is correctly signed. The partner application retrieves the signed document at sign-me API and presents it to the user for an optional download.
8. As an option, the partner application can verify the signature and retrieve the verification report using the `getSignatureProcess` call (see section 4.6). Then the state is `OK_VERIFIED` (see section 6.13), which means the document is correctly signed and all cryptographic information and certificates could be successfully verified by a technical verification service. The partner application retrieves the verification report at sign-me API and archives it for further reference.

### Variation: User Identity not lawfully verified

Preconditions:

- Like in Main Scenario, except the User is already registered but not identified in state `LAWFULLY_VERIFIED_HIGH`, only `COMMUNICATION_VERIFIED` (see section 6.6).

Steps modifying Main Scenario Step 3:

1. The partner application asks sign-me with the user's username to check the identity verification state and name of the user using the `getIdentity` call (see section 4.12). The user turns out to be in `COMMUNICATION_VERIFIED` (see section 6.6).
2. The partner application needs to create an identity verification process to bring the user into state `LAWFULLY_VERIFIED_HIGH`. The scenario proceeds like in 2.4.4.

### 2.2.6 Qualified Signing of Documents using API

This feature needs contractual extensions and can only be used if activated by D-Trust, please contact [sales@d-trust.net](mailto:sales@d-trust.net).

#### Summary

This use case shows how an embedded PDF signature can be made on a document with the sign-me system. The signature token which is used for signing is certified by a qualified D-TRUST CA.

In this use case, there are two roles at the API, which the partner application needs to handle.

- The partner with extident contract providing the partner application, which is involved at the sign-me API like in signing use case with API and sign-me Webapplication 2.2.3.
- The user is sitting at the partner application, and will enter his credentials at the partner application. The partner application will operate at the sign-me API using these credentials in the "USER" role.

#### Main Scenario

Preconditions (same as in 2.2.3):

- Partner application has already connected to sign-me API using the *SignMeApiV2Client* and the basic authentication credentials as communicated by D-TRUST.
- Partner is already logged in with his partner credentials.
- User is already registered and identified in State `LAWFULLY_VERIFIED_HIGH` (see section 6.6).

Steps:

1. The User sits in front of a web application of the partner, and in this stage of the process, the user shall sign a document using a qualified electronic signature. The user chooses the option to sign the document using sign-me.



2. The partner application asks the user to enter his sign-me username. The user is already a registered and identified user and enters his username.
3. The partner application asks sign-me with the user's username to check the identity verification state and name of the user using the `getIdentity` call (see section 4.12). As supposed the user is in `LAWFULLY_VERIFIED_HIGH` (see section 6.6).
4. The partner application creates a signature process, transferring the document to be signed, and the signature type to be used (in this case, a qualified signature: `S-QES_DOC_PADES_1`) at sign-me API (see section 4.2 for all parameters). sign-me returns a unique *signatureProcessId*.
5. Partner application presents a series of HTML forms for the user. The user enters his credentials and the partner application logs the user into the sign-me system over the API using the operation `Login` (see section 4.29). sign-me returns an access token for the user.
6. The user checks the document to be signed, and finally authorizes the signature in the partner application. The Partner application forwards the authorization to sign-me API using the access token of the user and operation `AuthorizeProcess` (see section 4.27). Then sign-me sends out an authorization code to the registered mobile phone of the user.
7. The user enters the received authorization code at the partner application, the partner application encrypts the authorization code and transfers it over the API using the operation `AuthorizeProcessStep2` (see section 4.28). sign-me then technically signs the document on behalf of the user, applying the default qualified signature token of the user.
8. The partner application checks the status of the signature process using the `GetSignatureProcess` call (see section 4.6). The state is `OK_UNVERIFIED` (see section 6.13), which means the document is correctly signed. The partner application retrieves the signed document at sign-me API and presents it to the user for an optional download.
9. As an option, the partner application can verify the signature and retrieve the verification report using the `getSignatureProcess` call (see section 4.6). Then the state is `OK_VERIFIED` (see section 6.13), which means the document is correctly signed and all cryptographic information and certificates could be successfully verified by a technical verification service. The partner application retrieves the verification report at sign-me API and archives it for further reference.

### **Variation: User Identity not lawfully verified**

Preconditions:

- Like in Main Scenario, except the User is already registered but not identified in state `LAWFULLY_VERIFIED_HIGH`, only `COMMUNICATION_VERIFIED` (see section 6.6).

Steps modifying Main Scenario Step 3:

1. The partner application asks sign-me with the user's username to check the identity verification state and name of the user using the `getIdentity` call (see section 4.12). The user turns out to be in `COMMUNICATION_VERIFIED` (see section 6.6).

2. The partner application needs to create an identity verification process to bring the user into state `LAWFULLY_VERIFIED_HIGH`. The scenario proceeds like in [2.4.4](#).

### 2.2.7 Basic Signing of Digests using API / Webapplication

#### Summary

This use case shows how a CMS signature can be applied on a digest/hash value with the sign-me system. The signature token which is used for signing is certified by a basic D-TRUST CA.

The use case is exactly the same as the use case for Basic Signing of Documents over API/Webapplication (see [2.2.1](#)), except two things:

- The signature type at sign-me API is for digest signatures (e.g. `S_BAS_DIG_-CADES_1`, see [6.43](#)).
- Verification of the signature against the document is not possible, because the document is not available on the sign-me system in this use case.

### 2.2.8 Advanced Signing of Digests using API / Webapplication

#### Summary

This use case shows how a CMS signature can be applied on a digest/hash value with the sign-me system. The signature token which is used for signing is certified by a advanced D-TRUST CA.

The use case is exactly the same as the use case for Basic Signing of Documents over API/Webapplication (see [2.2.2](#)), except two things:

- The signature type at sign-me API is for digest signatures (e.g. `S_ADV_DIG_-CADES_1`, see [6.43](#)).
- Verification of the signature against the document is not possible, because the document is not available on the sign-me system in this use case.

### 2.2.9 Qualified Signing of Digests using API / Webapplication

#### Summary

This use case shows how a CMS signature can be applied on a digest/hash value with the sign-me system. The signature token which is used for signing is certified by a qualified D-TRUST CA.

The use case is exactly the same as the use case for Basic Signing of Documents over API/Webapplication (see [2.2.3](#)), except two things:

- The signature type at sign-me API is for digest signatures (e.g. `S_QES_DIG_-CADES_1`, see [6.43](#)).
- Verification of the signature against the document is not possible, because the document is not available on the sign-me system in this use case.

### 2.2.10 Basic Signing of Digests using API

This feature needs contractual extensions and can only be used if activated by D-Trust, please contact [sales@d-trust.net](mailto:sales@d-trust.net).

## Summary

This use case shows the application of a digest signature with the sign-me system. The signature token which is used for signing is certified by a basic D-TRUST CA.

In this use case, there are two roles at the API, which the partner application needs to handle (same as in use case "Basic Signing of Documents using API", section 2.2.4):

- The partner with extitend contract providing the partner application, which is involved at the sign-me API like in signing use case with API and sign-me Webap-plication 2.2.4.
- The user is sitting at the partner application, and will enter his credentials at the partner application. The partner application will operate at the sign-me API using these credentials in the "USER" role.

## Main Scenario

Preconditions (same as in 2.2.1):

- Partner application has already connected to sign-me API using the *SignMeApiV2Client* and the basic authentication credentials as communicated by D-TRUST.
- Partner is already logged in with his partner credentials.
- User is already registered and identified in State COMMUNICATION\_VERIFIED (see section 6.6).

Steps:

1. The User sits in front of an application of the partner, and in this stage of the process, the user shall sign a document using a basic electronic signature. The user chooses the option to sign the document using sign-me.
2. The partner application asks the user to enter his sign-me username. The user is already a registered and identified user and enters his username.
3. The partner application asks sign-me API with the user's username to check the identity verification state and name of the user using the *getIdentity* call (see section 4.12). As supposed the user is in COMMUNICATION\_VERIFIED (see section 6.6).
4. The partner application creates a signature process, transferring the digest of the document to be signed (e.g. SHA-256), and the signature type to be used (in this case, a basic signature: S\_BAS\_DIG\_CADES\_1) at sign-me API (see section 4.2 for all parameters). sign-me returns a unique *signatureProcessId*.
5. Partner application presents a series of application forms for the user. The user enters his credentials and the partner application logs the user into the sign-me system over the API using the operation Login (see section 4.29). sign-me returns an access token for the user.
6. The user checks the document to be signed, and finally authorizes the signature in the partner application. The Partner application forwards the authorization to sign-me API using the access token of the user. sign-me then technically signs the digest on behalf of the user, applying the default basic signature token of the user.

7. The partner application checks the status of the signature process. The state is OK\_UNVERIFIED (see section 6.13), which means the digest is correctly signed. The partner application retrieves the signed digest as a PKCS7 signature container at sign-me API, embeds the PKCS7 signature container into the document, and presents it to the user for an optional download.

NOTE: The option, that the partner application can verify the signature and retrieve the verification report as in use case "Basic Signing of Documents using API and sign-me Webapplication" (see section 2.2.1) is not available, because with digest signing there is no document to verify against in the sign-me system.

### 2.2.11 Advanced Signing of Digests using API

This feature needs contractual extensions and can only be used if activated by D-Trust, please contact [sales@d-trust.net](mailto:sales@d-trust.net).

#### Summary

This use case shows how an embedded PDF signature can be made on a document with the sign-me system. The signature token which is used for signing is certified by an advanced D-TRUST CA.

In this use case, there are two roles at the API, which the partner application needs to handle (same as in use case "Advanced Signing of Documents using API", section 2.2.5):

- The partner with extident contract providing the partner application, which is involved at the sign-me API like in signing use case with API and sign-me Webapplication 2.2.5.
- The user is sitting at the partner application, and will enter his credentials at the partner application. The partner application will operate at the sign-me API using these credentials in the "USER" role.

#### Main Scenario

Preconditions (same as in 2.2.5):

- Partner application has already connected to sign-me API using the *SignMeApiV2Client* and the basic authentication credentials as communicated by D-TRUST.
- Partner is already logged in with his partner credentials.
- User is already registered and identified in State LAWFULLY\_VERIFIED\_HIGH (see section 6.6).

Steps:

1. The User sits in front of a web application of the partner, and in this stage of the process, the user shall sign a document using an advanced electronic signature. The user chooses the option to sign the document using sign-me.
2. The partner application asks the user to enter his sign-me username. The user is already a registered and identified user and enters his username.
3. The partner application asks sign-me API with the user's username to check the identity verification state and name of the user using the *getIdentity* call (see section 4.12). As supposed the user is in LAWFULLY\_VERIFIED\_HIGH (see section 6.6).

4. The partner application creates a signature process, transferring the digest of the document to be signed (e.g. SHA-256), and the signature type to be used (in this case, a advanced signature: S\_ADV\_DIG\_CADES\_1) at sign-me API (see section 4.2 for all parameters). sign-me returns a unique *signatureProcessId*.
5. Partner application presents a series of application forms for the user. The user enters his sign-me credentials and the partner application logs the user into the sign-me system over the API using the operation Login (see section 4.29). sign-me returns an access token for the user.
6. The user checks the document to be signed, and finally authorizes the signature in the partner application. The Partner application forwards the authorization to sign-me API using the access token of the user. sign-me then technically signs the digest on behalf of the user, applying the default advanced signature token of the user.
7. The partner application checks the status of the signature process. The state is OK\_UNVERIFIED (see section 6.13), which means the digest is correctly signed. The partner application retrieves the signed digest as a PKCS7 signature container at sign-me API, embeds the PKCS7 signature container into the document, and presents it to the user for an optional download.

NOTE: The option, that the partner application can verify the signature and retrieve the verification report as in use case "Basic Signing of Documents using API and sign-me Webapplication" (see section 2.2.2) is not available, because with digest signing there is no document to verify against in the sign-me system.

#### **Variation: User Identity not lawfully verified**

Preconditions:

- Like in Main Scenario, except the User is already registered but not identified in state LAWFULLY\_VERIFIED\_HIGH, only COMMUNICATION\_VERIFIED (see section 6.6).

Steps modifying Main Scenario Step 3:

1. The partner application asks sign-me with the user's username to check the identity verification state and name of the user using the getIdentity call (see section 4.12). The user turns out to be in COMMUNICATION\_VERIFIED (see section 6.6).
2. The partner application needs to create an identity verification process to bring the user into state LAWFULLY\_VERIFIED\_HIGH. The scenario proceeds like in 2.4.4.

### **2.2.12 Qualified Signing of Digests using API**

This feature needs contractual extensions and can only be used if activated by D-Trust, please contact [sales@d-trust.net](mailto:sales@d-trust.net).

#### **Summary**

This use case shows how an embedded PDF signature can be made on a document with the sign-me system. The signature token which is used for signing is certified by a qualified D-TRUST CA.

In this use case, there are two roles at the API, which the partner application needs to handle (same as in use case "Qualified Signing of Documents using API", section 2.2.6):

- The partner with extident contract providing the partner application, which is involved at the sign-me API like in signing use case with API and sign-me Webapplication 2.2.6.
- The user is sitting at the partner application, and will enter his credentials at the partner application. The partner application will operate at the sign-me API using these credentials in the "USER" role.

## Main Scenario

Preconditions (same as in 2.2.6):

- Partner application has already connected to sign-me API using the *SignMeApiV2Client* and the basic authentication credentials as communicated by D-TRUST.
- Partner is already logged in with his partner credentials.
- User is already registered and identified in State `LAWFULLY_VERIFIED_HIGH` (see section 6.6).

Steps:

1. The User sits in front of a web application of the partner, and in this stage of the process, the user shall sign a document using a qualified electronic signature. The user chooses the option to sign the document using sign-me.
2. The partner application asks the user to enter his sign-me username. The user is already a registered and identified user and enters his username.
3. The partner application asks sign-me API with the user's username to check the identity verification state and name of the user using the `getIdentity` call (see section 4.12). As supposed the user is in `LAWFULLY_VERIFIED_HIGH` (see section 6.6).
4. The partner application creates a signature process, transferring the digest of the document to be signed (e.g. SHA-512), and the signature type to be used (in this case, a advanced signature: `S_QES_DIG_CADES_1`) at sign-me API (see section 4.2 for all parameters). sign-me returns a unique *signatureProcessId*.
5. Partner application presents a series of application forms for the user. The user enters his sign-me credentials and the partner application logs the user into the sign-me system over the API using the operation `Login` (see section 4.29). sign-me returns an access token for the user.
6. The user checks the document to be signed, and finally authorizes the signature in the partner application. The Partner application forwards the authorization to sign-me API using the access token of the user and operation `AuthorizeProcess` (see section 4.27). Then sign-me sends out an authorization code to the registered mobile phone of the user.
7. The user enters the received authorization code at the partner application, the partner application encrypts the authorization code and transfers it over the API using the operation `AuthorizeProcessStep2` (see section 4.28). sign-me then technically signs the document on behalf of the user, applying the default qualified signature token of the user.

8. The partner application checks the status of the signature process. The state is OK\_UNVERIFIED (see section 6.13), which means the digest is correctly signed. The partner application retrieves the signed digest as a PKCS7 signature container at sign-me API, embeds the PKCS7 signature container into the document, and presents it to the user for an optional download.

NOTE: The option, that the partner application can verify the signature and retrieve the verification report as in use case "Basic Signing of Documents using API and sign-me Webapplication" (see section 2.2.3) is not available, because with digest signing there is no document to verify against in the sign-me system.

### **Variation: User Identity not lawfully verified**

Preconditions:

- Like in Main Scenario, except the User is already registered but not identified in state LAWFULLY\_VERIFIED\_HIGH, only COMMUNICATION\_VERIFIED (see section 6.6).

Steps modifying Main Scenario Step 3:

1. The partner application asks sign-me with the user's username to check the identity verification state and name of the user using the getIdentity call (see section 4.12). The user turns out to be in COMMUNICATION\_VERIFIED (see section 6.6).
2. The partner application needs to create an identity verification process to bring the user into state LAWFULLY\_VERIFIED\_HIGH. The scenario proceeds like in 2.4.4.

## **2.3 Electronic Remote Sealing**

### **2.3.1 Qualified Sealing of Digests using API**

#### **Summary**

This use case shows how a digest of a document (e.g. PDF) can be sealed with the sign-me system. All needed interaction with the end-user takes place at the front-end of the partner application. Requests for electronic remote sealing are triggered by the partner application over the sign-me API. Two-factor authorization of the electronic seals are performed using a cryptographic token for challenge signing.

Technically, remote sealing is performed by means of a sealing token, that is hosted by D-TRUST.

The sealing token is owned by the partner organization and is certified by a qualified D-TRUST CA. Application of the token is under sole control of the owning partner institution. At D-TRUST a request for sealing tokens will be initiated by sealing managers of the partner organization. These persons are officials announced for this purpose by the partner organization. They are also responsible to authorize the application of the sealing token in the IT of the partner organization.

In order to request sealing tokens from D-TRUST, the sealing manager will be identified by means of an eIDAS identification process together with the partner organization itself. In addition to the eIDAS identification sealing managers need to proof, that they are allowed to request and authorize sealing token use ("power of attorney") for the organization.



After successful identification, the sealing manager can create a cryptographic token (key pair for asymmetric cryptography) at the eIDAS identification service. The cryptographic token is certified by a qualified certificate. The partner system will use this token as a second factor to authorize the application of qualified seals on the sign-me API.

Following the successful creation of the cryptographic token, the sealing manager will transfer the qualified certificate to D-TRUST and initiate the registration of the certificate as verification means for application of the 2nd factor.

Identification and creation of the cryptographic token are both available at D-TRUST, but offline to the sign-me system and its API.

Please note: Because the basic technology used for sealing is exactly the same as for signing, the API calls remain the same as with the signing use cases. Therefore, e.g. in order to start a sealing process at the API, the API call `CreateSignatureProcess` (see [4.2](#)) will be used.

## Main Scenario

Preconditions:

- Partner application has already connected to sign-me API and partner is already logged in with his partner credentials.
- The identity of the partner organization is verified for remote qualified sealing (available at D-TRUST, but offline to the sign-me system and its API).
- Sealing manager of the partner organization has already received and ordered the activation of a qualified sealing token (at D-TRUST, but offline to the sign-me system and its API). The cryptographic token for the 2nd factor authorization "QUALIFIED SEAL ID P12 (PIN SMS) (EC)" for the partner organization is available in the partner application. The IT department of the partner organization was authorized by the sealing manager to configure the cryptographic token within the partner application. This is a company-internal procedure.
- The end user operating at the partner application and requesting the signature for electronic sealing for sign-me is appropriately authorized by the sealing manager of the partner organization. This is a company-internal procedure.

Steps:

1. The user sits in front of a WEB application of the partner, and in this stage of the process, where the user shall electronically seal a document using an qualified electronic seal (QESEAL). The user chooses the option to seal the document using sign-me.
2. The partner application asks sign-me with the partner's username to check which qualified sealing tokens are available for the partner organization.
3. The partner application presents the available tokens to the user or selects an applicable electronic seal based on the partner application and asks the user to enter the sealing token to use. The sealing token in sign-me is identified by the SEAL-ID Token for 2-factor identification. Based on the SEAL-ID Token, the partner application can generate the username for the sealing manager registered in sign-me. There is a 1:1 relationship between the sealing manager and the SEAL-ID Token for 2-factor identification in sign-me.



4. The partner application creates a signature process, transferring the document to be signed/sealed representation (digest, currently up to 64 in single batch), and the signature type to be used (in this case, a qualified electronic seal: S-QESEAL\_DIG\_CADES\_1) at sign-me API (see CreateSignatureProcess in 4.2). sign-me returns a unique *signatureProcessId* as a result of CreateSignatureProcess which is a handle of the signature process.
5. The partner application will request authorization for the signature process at the sign-me API using an AuthorizeProcess request. sign-me will respond with a challenge (containing a salted/hashed authorization code) which shall be signed with the sealing managers's cryptographic 2 factor token. The lifetime of the authorization will be five minutes maximum.
6. The user checks the document to be electronically sealed, and finally confirms the authorization for usage of the electronic seal in the partner application. The partner application signs the challenge presented by sign-me in the authorization response and sends it to sign-me using the AuthorizationStep2 request.
7. sign-me receives the AuthorizationStep2 and validates the sealing request.
8. sign-me technically signs the document on behalf of the sealing manager of the partner organization owning the electronic seal, applying the default qualified sealing token of the sealing manager.
9. The partner application checks the status of the signature process using the getSignatureProcess call (see GetSignatureProcess in 4.6). The state is OK\_UNVERIFIED (see section 6.42), which means the document is correctly signed. The partner application retrieves the signed hash value (CMS signature container) at sign-me API.
10. The partner application presents the resulting electronically sealed document to the user.

### Remarks and Variations

1. The final check before issuing authorizeStep2 in step 6 can also be done in advance of CreateSignatureProcess in step 4, if the check cannot be an interactive procedure of the partner application in this phase. This is probably the case, if the partner application wants to electronically seal many documents created by many users in one transaction.
2. If in step 5 the X.509 certificate of the cryptographic token for 2 factor authentication is expired the authorization is directly rejected. The remaining validity period of the token must be at least a 5 minutes.
3. If in authorization step 2 of electronic sealing in 7 the X.509 certificate of the cryptographic token for 2 factor authentication is expired or other validation problems exist (like signature invalid) the authorization is directly rejected. Unlike in signing scenarios there is no retry possibility for entering the second factor.

### 2.3.2 Sample Code and Testing

Tests for Remote Sealing of Digests can be based on sample code *DigestSealingQualifiedSample.java* of sign-me API package (see 7.2).

The sample operates the sign-me API, only. There is no browser or user interface interaction.

The sample is preconfigured with a standard test SEAL-ID token for 2nd factor authentication just to show configuration of SEAL-ID tokens. However, sealing will not work using this sample token in combination with your partner account on the reference system.

For the sample to fully operate, on request, one or more SEAL-ID tokens can be created for testing with your partner account on the reference system.

## 2.4 Registration and Identification

### 2.4.1 Registration using sign-me Webapplication

#### Summary

This use case shows how a user can register himself at the sign-me web application. The user is within a use case at a partner website and shall sign a document. The partner offers an electronic possibility with sign-me.

#### Main Scenario

Preconditions:

- User is not registered in sign-me system.

Steps:

1. The user learns about the possibility to use sign-me during a business transaction at some partner web application. The partner asks whether the user wants to use an electronic remote signing server 'sign-me' to sign the document.
2. The user indicates 'yes'. The partner web application asks, whether the user is already a sign-me user. The user indicates 'no'.
3. The partner web application offers the possibility to the user to register for the sign-me service under the link <https://cloud.sign-me.de/signature/start>.
4. The user clicks on the link, enters all required data including an email address and acknowledges the registration.
5. sign-me web application presents a result page and informs, that an email was sent to the user's email address which has to be handled to complete the registration. The user is now in state UNVERIFIED (see section 6.6).
6. After arrival of the email the user proceeds with use case "Identification based on Communication" (see 2.4.3).

### 2.4.2 Registration using API

#### Summary

This use case shows how a user can register himself through a partner web application. The user is within a use case at a partner website and shall sign a document. The partner application offers an electronic possibility with sign-me.

## Main Scenario

Preconditions:

- Partner application has already connected to sign-me API using the *SignMeApiV2Client* and the basic authentication credentials as communicated by D-TRUST.
- Partner is already logged in with his partner credentials.
- User is not registered in sign-me system.

Steps:

1. The user learns about the possibility to use sign-me during a business transaction at some partner web application. The partner asks whether the user wants to use an electronic remote signing server 'sign-me' to sign the document.
2. The user indicates 'yes'. The partner web application asks, whether the user is already a sign-me user. The user indicates 'no'.
3. The partner web application asks, whether the user wants to immediately register him-/herself with the sign-me service. The user indicates 'yes'.
4. The partner application registers the user using sign-me API call `RegisterIdentity` (see section 4.11). Before issuing the register call, the partner application asks the user to define his/her sign-me username and password. Other user data which are used for registration are from the partner application data base or will be requested by the user in this step. Applicable legal information and agreements need to be implemented in this dialog of the partner application.
5. The partner application presents a result page and informs, that an email was sent to the user's email address by the sign-me service which has to be handled to complete the registration. The user is now in state UNVERIFIED (see section 6.6).
6. After arrival of the email the user proceeds with use case "Identification based on Communication" (see 2.4.3).

## 2.4.3 Identification based on Communication

### Summary

This use case shows how a user can complete its registration and do a minimum identification, by verifying his/her email address. The identity verification state changes into `COMMUNICATION_VERIFIED`.

The user is in front of the sign-me web application.

## Main Scenario

Preconditions:

- User is already registered and identified in State UNVERIFIED (see section 6.6).
- User has received an email containing the confirmation link.

Steps:

1. The User sits in front of his/her email client and clicks on the confirmation link in the confirmation email. The sign-me web application opens up in a browser window showing a confirmation dialog and agreement check boxes as well as entry fields for the login credentials.
2. The user reads all documents to agree to and checks all check boxes to signal his/her understanding and agreement. After that the user enters his/her login credentials, and sign-me web application authenticates the user.
3. The sign-me web application uses the 4.13 to convey confirmation of the user. The user is now in state `COMMUNICATION_VERIFIED` (see section 6.6).

#### 2.4.4 Identification based on Verification Process

##### Summary

This use case shows how a partner can bring a user into state `LAWFULLY_VERIFIED_HIGH`. In this state the partner can have the user sign documents.

The partner is involved at the sign-me API by means of the partner application.

The user is sitting at the web application of the partner, and will be redirected to the sign-me front-end. Then the user will perform the identification session at the sign-me front-end.

##### Main Scenario

Preconditions:

- Partner application has already connected to sign-me API using the *SignMeApiV2Client* and the basic authentication credentials as communicated by D-TRUST.
- Partner is already logged in with his partner credentials.
- User is already registered and identified in State `COMMUNICATION_VERIFIED` (see table 6.6).

Steps:

1. The User sits in front of a web application of the partner, and in this stage of the process, the partner knows that the user is in state `COMMUNICATION_VERIFIED` and needs identity verification to bring him/her into `LAWFULLY_VERIFIED_HIGH`.
2. The partner application needs to create an identity verification process to bring the user into state `LAWFULLY_VERIFIED_HIGH`. For that the partner application uses the `CreateIdentityVerificationProcess` call (see section 4.14). sign-me returns an `IdentityVerificationProcess` object containing the `identityVerificationProcessURL`.
3. The Partner redirects the user session to sign-me using the `identityVerificationProcessURL`.
4. sign-me presents a series of HTML forms for the user. The user follows the instructions and will perform the identification session together with the identification service.
5. After the identification session is successfully completed, sign-me will redirect the user to the http-link *redirectURLOnOk*, which the partner provided in step 2.

6. The partner application checks the status of the identity verification process using the `GetIdentityVerificationProcess` call (see section 4.16). The state is `LAWFULLY_VERIFIED_HIGH` (see section 6.6). In this state the user is permitted to apply basic, advanced and qualified signatures.

## 2.5 Authentication

### 2.5.1 Partner Login Session Handling

#### Summary

This use case shows how a partner application can login its partner session.

#### Implementation Notes for Partner Login Sessions

For performance reasons the `AccessTokenGrant` (see section 5.1) for the partner login session shall be kept globally in the application server and should be used for follow-up calls as described here. Usage of the same `AccessTokenGrant` in multiple threads is safe on the server side. However, it requires proper synchronization of `AccessTokenGrant` usage and refresh on the client side.

A logout from a partner session should only be done if there is no further partner operation required in the near future in production operation, or after completing a test-case in test environment. Otherwise, the partner account runs into the session limitation of maximum 10 partner sessions, and the client needs to wait for a timeout (10 min) of the oldest session until the next login is possible.

As a guideline there should only be one partner login session per server node at the same time.

#### Main Scenario

Preconditions:

- Partner application has already connected to sign-me API using the *SignMeApiV2Client* and the basic authentication credentials as communicated by D-TRUST.

Steps:

1. The partner application uses the convenience operation `SignMeApiV2ClientExt.loginUIDPW` (see the language specific access modules in section 7) with parameters IP-Address of the client, `role="PARTNER"` and username/password as communicated by D-TRUST. Alternatively, partner applications can use `Login/LoginStep2` operations (see section 4.29).
2. sign-me responds with an `AccessTokenGrant` structure (see section 5.1), containing the access token, which identifies the partner login session.
3. The access token can now be used to serially issue operation requests for the partner application to the sign-me system and synchronously wait for responses. For this, the partner application builds a security header using the convenience operation `SignMeApiV2ClientExt.getSecurityHeader` from the `AccessTokenGrant`.
4. Optional: For alive checks of the partner login session the `GetVersion` operation (see section 4.33) is used.

5. Preferred: To keep the partner login session alive and refresh the access token at the same time if needed, the convenience operation `SignMeApiV2ClientExt.refreshLoginExt` is used (see the language specific access modules in section 7). The operation is called all 30-60 seconds and does a refresh on the server side if needed. The operation returns a new `AccessTokenGrant` to be used further in the session.
6. The partner application continues with step 2 or, if no further operations on behalf of the partner application, calls convenience operation `SignMeApiV2ClientExt.logoutExt` or `Logout` (see section 4.32).

## 2.5.2 User Login Session Handling

### Summary

This use case shows how an user can login its user session over a partner application.

### Implementation Notes for User Login Sessions

For performance reasons the `AccessTokenGrant` (see section 5.1) for the user login session shall be kept within the user-specific session context in the application server and should be used for follow-up calls for the very same user in the same user session as described here.

A logout from a user login session should be done as part of finishing the user-specific session context in the application server in production operation, or after completing a test-case in test environment. Otherwise, the user account runs into a limitation of maximum sessions, and the client needs to wait for a timeout (10 min) of the oldest session until the next login is possible.

As a guideline there should only be one user login session per user at the same time.

### Main Scenario

Preconditions:

- Partner application has already connected to sign-me API using the *SignMeApiV2Client* and the basic authentication credentials as communicated by D-TRUST.
- Partner is already logged in with his partner credentials.

Steps:

1. The partner application uses the convenience operation `SignMeApiV2ClientExt.loginUIDPW` (see the language specific access modules in section 7) with parameters partner security header (from use case 2.5.1, step 3), IP-Address of the client, role="USER" and username/password as given by the user of the partner application. Alternatively, partner applications can use `Login/LoginStep2` operations (see section 4.29).
2. sign-me responds with an `AccessTokenGrant` structure (see section 5.1), containing the access token, which identifies the user login session.
3. The access token can now be used to serially issue operation requests for the user to the sign-me system and synchronously wait for responses. For this, the partner application builds a user security header using the convenience operation `SignMeApiV2ClientExt.getSecurityHeader` from the `AccessTokenGrant`.
4. Optional: For alive checks of the user login session the `GetVersion` operation (see section 4.33) is used.

5. Only if longer sessions are needed for the user: To keep the user login session alive and refresh the access token at the same time if needed, the convenience operation `SignMeApiV2ClientExt.refreshLoginExt` is used (see the language specific access modules in section 7). The operation is called all 30-60 seconds and does a refresh on the server side if needed. The operation returns a new `AccessTokenGrant` to be used further in the session.
6. The partner application continues with step 2 or, if no further operations on behalf of the user, calls convenience operation `SignMeApiV2ClientExt.logoutExt` or `Logout` (see section 4.32).

## 2.6 Fast Track Use Cases

Figure [2.1](#) shows a summary of activities between user, partner application and sign-me to complete a fast track use case. The figure connects all decisions and actions to be made for all major situations depending on the users identification state. The numbers in parantheses refer to the stepwise use cases description in section [2.6.1](#).



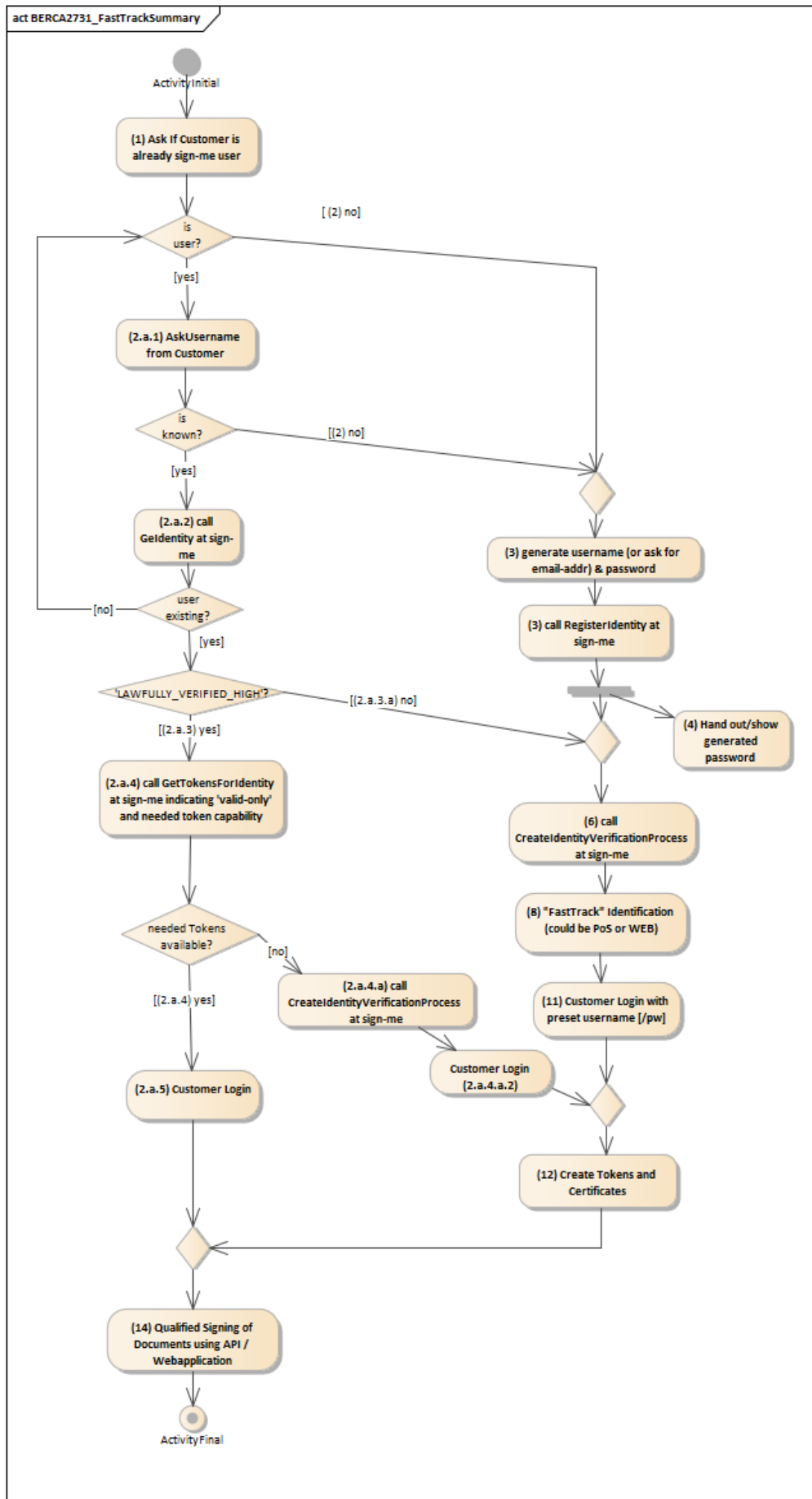


Figure 2.1: Fast Track Summary Flowchart

### 2.6.1 PoS Fast Track Registration, Identification and Signing

This feature needs contractual extensions and can only be used if activated by D-Trust, please contact [sales@d-trust.net](mailto:sales@d-trust.net).

#### Summary

This use case specifies how a Partner Organization can register and identify new (natural) persons and let them sign a document in a Point of Sale (PoS) situation. Therefore this use case combines the following existing use cases:

- [Registration using API](#) (see [2.4.2](#))
- [Identification based on Communication](#) (see [2.4.3](#))
- [Identification based on Verification Process](#) (see [2.4.4](#))
- [Qualified Signing of Documents using API / Webapplication](#) (see [2.2.3](#)) (this could also be any of the other signing use cases).

Combining the use cases into one means that intermediate log-ins can be saved, because we assume one single user session, which takes place locally on partner's premises (Point of Sale).

The following describes the steps needed to complete the use case. Figure [2.2](#) shows the interaction between WebBrowser of *User*, *PoS Partner application* and sign-me API as well as sign-me Webapplication and *PoS Ident application*. Interaction sequences between *User* and *PoS Partner application* need to represent examples, because *PoS Partner application* and *PoS Ident application* are not scope of design here.

#### Level and Scope

User level for sign-me system.

#### Preconditions

- *PoS Partner application* has already connected to sign-me API using the *Sign-MeApiV2Client* and the basic authentication credentials as communicated by D-TRUST.
- *PoS Partner application* is already logged in with its partner credentials.

#### Main Success Scenario

1. *PoS clerk* asks if user is already sign-me user
2. *User* is not sign-me user up to now.
3. *PoS Partner application* takes identification data of *User* known so far and registers the identity of *User* using sign-me API call RegisterIdentity. In addition to identification data of *User*, the *PoS Partner application* provides:
  - sign-me username - The username can either be an Email-Address of *User* or a Username generated compliant with the sign-me username rules (see [6.63](#))
  - sign-me password - is generated via the *PoS application* compliant with the sign-me password rules (see [6.34](#)). Optionally, the password can be set by the *User* in the *PoS Partner application*.

- sign-me revocation string ("Sperrkennwort") - is generated via the *PoS Partner application*. Optionally, the revocation string can be set by the *User* in the *PoS Partner application*.

NOTE: sign-me password and sign-me revocation string can later be changed in sign-me self-service.

4. *PoS clerk* will hand out the created password to the user in a secure way for later use.
5. sign-me will send out an information email about account creation.

NOTE: This email is only a reminder for the *User*. In FastTrack use cases, it is not needed to confirm an email link to proceed with the next steps.

6. *PoS Partner application* creates verification using `CreateIdentityVerificationProcess` indicating "POS\_2\_FT\_IDENT\_4EYE" and the username preset in `RegisterIdentity`. sign-me will initiate a new identification at the PoS Ident Backend at `Identity.tm` using "putOrder". sign-me API returns an `identityVerificationProcessURL` to the *PoS Partner application*.
7. *PoS Partner application* posts to the `identityVerificationProcessURL` and sets "username" and "password" as http parameters (for more information on post see sample application "FastTrackSample.java"). Username and password were preset in step 3. sign-me Webapplication redirects the *User* Browser/Tablet to the *PoS Ident application*, where the actual identification takes place.

NOTE: *PoS Partner application* and *PoS Ident application* could be the same or tightly integrated applications. For the purpose of this use case we assume, that there is a *PoS Ident application* (module) which we can address by means of redirect URLs which are generated during step 6 and which will be used by the *PoS clerk* in the next step.

8. *PoS clerk* uses *PoS Ident application* to identify *User*. The behavior of the *PoS Ident application* and identification process is out of scope here. As a result the identification data consist of the following items:
  - Family Names
  - Given Names
  - optional: (Academic) Title
  - Place of Birth
  - Date of Birth
  - Nationality (as 2-letter ISO or 1/3-letter ICAO Code)
  - Place of Residence (City, ZIP, Street Address, optional State, Country)
  - verified Email-Address - it is verified that the email address belongs to an email account of *User*
  - verified Mobile Phone Number - it is verified that the Mobile Phone Number belongs to a mobile phone of *User*
  - result of a 4-eye check of the identification (completion can wait until step 9)
  - identification transcript - a report of the session including document and person snapshots

All identification data except mobile phone number, email address, 4-eye check result, and transcript shall be written as on the used identification document. After the identification finished *PoS Ident application* will redirect to sign-me Webapplication using the URL given as parameters in 6.

9. sign-me gets the result of identification on a trustworthy connection with the *identity service provider* for "POS\_2\_FT\_IDENT\_4EYE". This includes the 4-eye check result and identification transcript with document and person snapshots etc..
10. sign-me presents a dialog with filled username and password and asks the user to agree upon terms and conditions of D-TRUST for certificate production. The dialog also contains an optional part to directly change the password.
11. *User* agrees and logs into sign-me.
12. sign-me creates basic, advanced and qualified signature tokens for *User*. sign-me redirects back to *PoS Partner application* with a redirect URL defined in step 6 as a parameter for the CreateIdentityVerificationProcess call at sign-me API.
13. *PoS Partner application* shows the document-to-be-signed to the *User*. *User* agrees to sign the document using sign-me.
14. *PoS Partner application* creates signature process and redirects sign-me user to the sign-me Webapplication. The *User* proceeds as described in [Qualified Signing of Documents using API / Webapplication](#) (see 2.2.3) (this could also be any of the other signing use cases). Exception is that the *User* does not have to do the initial login, because *User* is already logged with the first factor from step 11. If the user is ready with signing sign-me redirects to the *PoS application*.
15. *PoS Partner application* stores and presents results of signing process.

## Extensions for Alternate Scenarios

### 2.a Already sign-me user

1. *User* confirms to be sign-me user and tells username.
2. *PoS clerk* enters username into *PoS application* and by means of that checks username with sign-me. This will be performed using GetIdentity at sign-me API (see 4.12).
3. sign-me answers that username is known and *PoS application* further checks that corresponding user is "LAWFULLY\_VERIFIED\_HIGH".
4. *PoS application* checks that user has valid qualified signature token using GetTokensForIdentity at sign-me API (see 4.25).
5. *PoS application* continues with step 14 with preset username (no preset password).

### 3.a Already sign-me user, but not in state "LAWFULLY\_VERIFIED\_HIGH"

1. *PoS application* continues with step 6 with preset username (no preset password).

### 4.a Already sign-me user and in state "LAWFULLY\_VERIFIED\_HIGH", but no qualified signature token

1. *PoS Partner application* creates verification using `CreateIdentityVerificationProcess` indicating "POS\_2\_FT\_IDENT\_4EYE" and the username given by *User*. sign-me will initiate a new identification process and sign-me API returns an `identityVerificationProcessURL` to the *PoS Partner application*. There will be no additional identification at the *PoS Ident application*, we will only repeat the last step of the identification (agreements).
2. *PoS application* redirects to `identityVerificationProcessURL` and sign-me continues with step 10 with preset username (no preset password).

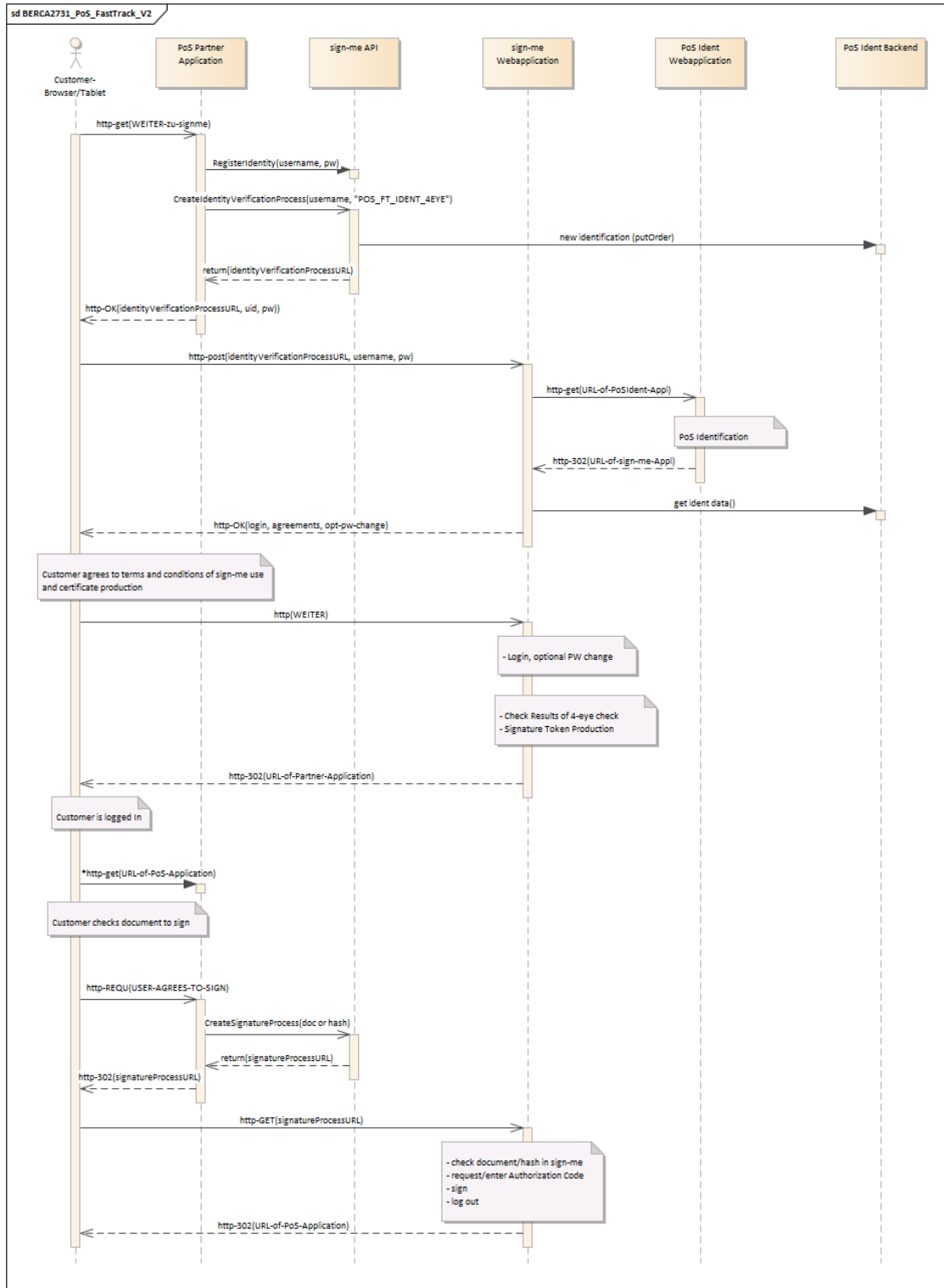


Figure 2.2: Fast Track Interaction - Example PoS

## 2.6.2 WEB Fast Track Registration, Identification and Signing

### Summary

This use case specifies how a Partner Organization can register, identify new (natural) persons and let them sign a document immediately in one session of a Web Application. Therefore this use case combines the following existing use cases in the same way to [PoS \(Point of Sale\) Registration, Identification and Signing](#) (see 2.6.1):

- [Registration using API](#) (see 2.4.2)
- [Identification based on Communication](#) (see 2.4.3)
- [Identification based on Verification Process](#) (see 2.4.4)
- [Qualified Signing of Documents using API / Webapplication](#) (see 2.2.3) (this could also be any of the other signing use cases).

Combining the use cases into one means that intermediate log-ins can be saved, because we assume one single user session, which takes place in the web application of the partner. Typical cases are bank applications where the user is a new user which will perform a video identification and sign the contract for opening his account.

The steps needed to complete the use case are the same as in case [PoS \(Point of Sale\) Registration, Identification and Signing](#), except instead of a *PoS Partner application* a *Partner WEB application* like a banking site is used. In the same way instead of a *PoS Ident application* the *Video Ident Webapplication* will be used.

Figure 2.3 shows the interaction between WebBrowser of *User*, *Partner WEB application* and sign-me API as well as sign-me Webapplication and *Video Ident Webapplication*. Interaction sequences between *User* and *Partner WEB application* need to represent examples, because *Partner WEB application* and *Video Ident Webapplication* are not scope of design here.

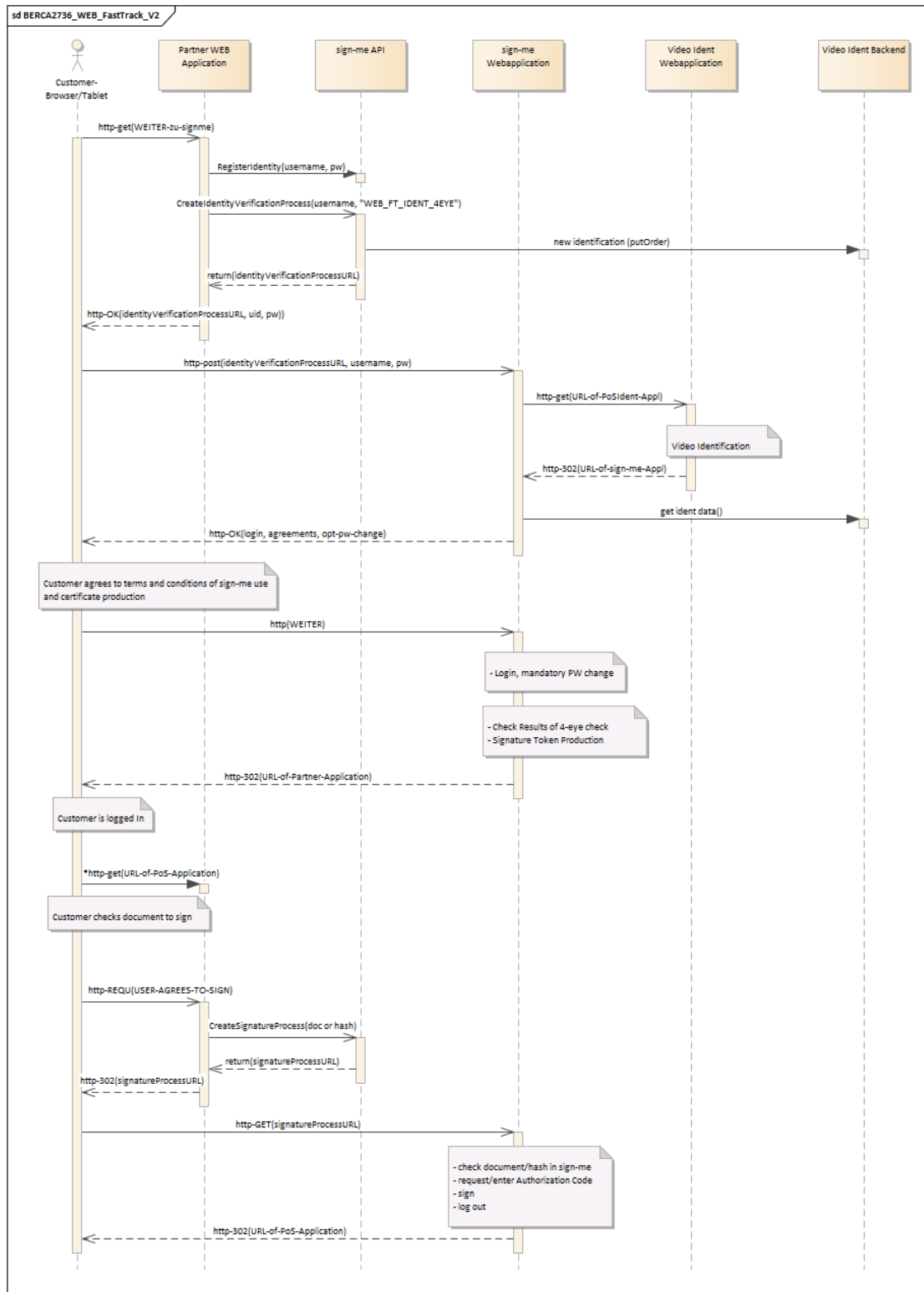


Figure 2.3: Fast Track Interaction - Example WEB



## 3 Service and Access Points

### 3.1 Service Overview: SignMeApiV2

The sign-me API V2 service allows partners of the sign-me system to access operations for login, signing for electronic signatures and seals, as well as PDF signature verification. These services can be offered to users by the partner's web applications.

To support enrollment of users, sign-me API offers supportive services for user registration including user identity verification. The identity verification operations always produce default tokens, which are suitable for use at the achieved identity verification state. There is no need for partners to take care of token management.

For electronic signatures, users are natural persons (see data type in section 5.21 ).

For electronic sealing users are dedicated sealing managers which operate over the API on behalf of their organizations (see data type in section 5.20) to offer sealing for those partner organizations.

Users have no direct access to the sign-me API, however they take part in use-cases for confirmation and authorization purposes. Users do this through login sessions, which are brokered by authenticated partner applications. If a user needs to take part in an use-case, the partner application (or the sign-me web application), authenticates the user by means of the login operation. The partner application can only provide this service by means of an authenticated login session of its own (the application provider session, see Login operation 4.29).

For electronic sealing, the sealing managers operate directly on the partner API on dedicated sealing partner sessions. Sealing managers are authenticated by means of a 2-factor token. In practice, the sealing manager who is legally responsible for handling of the 2-factor token for electronic sealing will hand over operation of the 2-factor token to an IT specialist which will install the 2-factor token in an appropriate partner application. The partner application will automatically order signatures for electronic sealing for all documents which need to be sealed on behalf of the organization.

### 3.2 Access Points for SignMeApiV2

#### 3.2.1 Access Point for SignMeApiV2ReferenceTestSystem

This is the sign-me API access point of the reference (test) system. The reference system has the same functionality and operational configuration as the live system. Thus, it will typically be used for connection and integration tests by the partners of the sign-me system. It does not contain live data of real users, and sign-me partners are advised not to put any real user data into it.

NOTE: Be aware, that the database will be routinely cleaned from test-data which was inserted as part of testing.

For access to SignMeApiV2ReferenceTestSystem use the following URI:

<https://cloud-ref-sp.sign-me.de:443/api/v2>

### **3.2.2 Access Point for SignMeApiV2LiveSystem**

This is the sign-me API access point of the live system.  
For access to SignMeApiV2LiveSystem use the following URI:  
<https://cloud-sp.sign-me.de:443/api/v2>

## 4 Operations of the API

### 4.1 Overview

This section specifies all available operations of the sign-me API. Each operation represents a part of the sign-me service which can be remotely called using the access points specified above (see 3). The operations and its parameters are formally specified by means of a WSDL specification (see [WSDL]).

If possible, we recommend to utilize the sign-me API access modules to save time and effort (see 7).

The set of operations are grouped into the following areas, which are identical to the main use case cases (see section 2):

1. Eletronic Signatures and Sealing: operations from *CreateSignatureProcess* 4.2 to *GetSignatureProcess* 4.6, incl. *AuthorizeProcess* 4.27
2. Registration and Identification: operations from *RegisterIdentity* 4.11 to *ManageIdentity* 4.17
3. Token Activation: *RevokeCertificateOfTokenActivationProcess* 4.26
4. Authentication: operations from *Login* 4.29 to *Logout* 4.32

### 4.2 CreateSignatureProcess

The CreateSignatureProcess operation is used by the partner application to create a signature process. The signature process allows to track the signature of the user for electronic signing or sealing and retrieve the results like signature, verification report and the identity of the signer.

After the signature process is created the partner lets the user authorize the signature. The user can authorize the signature application (through the partner application, of course, see AuthorizeProcess operation), or the partner application can re-direct the user to the sign-me system using a unique URL of the signature process (signatureProcess.signatureProcessURL). In the latter case, the sign-me system would re-direct to the partner application again after completion of the signature (signatureProcess.redirectURLs).

After the signature is applied the partner application can retrieve the results using GetSignature operation.

If the signature shall be created for electronic sealing, the whole process is an API operation comprising CreateSignatureProcess/UpdateSignatureProcess, AuthorizeProcess, AuthorizeProcessStep2, and GetSignatureProcess.

## Input of CreateSignatureProcess

Parameter	Field Type	Occurrence
securityHeader	SecurityHeader	1
signatureType	SignatureType	1
signatureParameters	SignatureParameters	0..1
signerUsername	Username	0..1
signerIdentityId	ObjectId	0..1
tokenId	ObjectId	0..1
itemsToBeProcessed	ItemToBeProcessed	0..256
redirectURLs	RedirectURLs	0..1
suppressFinalDialog	boolean	0..1

Table 4.1: Input Parameters of CreateSignatureProcess

## Input Details of CreateSignatureProcess

Parameter	securityHeader
Type	SecurityHeader [see 5.27]
Occurence	1
Meaning	The security header needs to be related to a valid login session with "PARTNER" role with quality "ONE_FACTOR".

Parameter	signatureType
Type	SignatureType [see 6.43]
Occurence	1
Meaning	The type of signature which shall be requested by the user person of the partner application. The signature type defines quality level (basic, advanced, qualified) and format of the signature, as well if the content to be signed shall be a document or another binary content or is already a digest/hash value of a document.

Parameter	signatureParameters
Type	SignatureParameters [see 5.28]
Occurence	0..1
Meaning	By means of signature parameters the visibility of the signature icon or the usage of a field within (PDF) documents can be controlled. The use of signature parameters must be allowed for the signature type (see 6.43, e.g. S_QES_DOC_PADES_B_LT_1). If the signature parameters are not allowed for a digest signature type (e.g. S_QES_DIG_CADES_1), then an invalid parameter value error is raised.

<b>Parameter</b>	<b>signerUsername</b>
<b>Type</b>	<b>Username</b> [see 6.63]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	<p>The sign-me username of the person for whom the signature process is created and which shall apply the signature.</p> <p>Only this identified sign-me user can authorize the signature later on. Any other user trying to authorize a signature for this signature process will be rejected.</p> <p>This (default) token will be selected based on the username and the required token capability of the signature type.</p> <p>NOTE: username cannot identify an organization in this release.</p>

<b>Parameter</b>	<b>signerIdentityId</b>
<b>Type</b>	<b>ObjectId</b> [see 6.32]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The object id of the identity received by calls to the sign-me system (e.g. getSignatureProcess).

<b>Parameter</b>	<b>tokenId</b>
<b>Type</b>	<b>ObjectId</b> [see 6.32]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Deprecated parameter. Should not be present any more and will be removed in the future.

<b>Parameter</b>	<b>itemsToBeProcessed</b>
<b>Type</b>	<b>ItemToBeProcessed</b> [see 5.14]
<b>Occurrence</b>	<b>0..256</b>
<b>Meaning</b>	<p>This contains a list of items which shall be signed and/or verified. The actual allowed maximum number of items in this list is depending on the signature type. There are also size restrictions on the item content for both the the single item and the list altogether which are depending on the signature type and which are out-of-scope of this specification.</p> <p>NOTE: in this release the limit is 16 for document signing, 64 for qualified digest signing, and 256 for basic and advanced digest signing.</p> <p>If the partner wants to use UpdateSignatureProcess (see 4.3), this list must be an empty list.</p>

<b>Parameter</b>	<b>redirectURLs</b>
<b>Type</b>	<b>RedirectURLs</b> [see 5.24]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	<p>This can contain URLs to redirect to in case of "OK"-, "FAIL"-, and "CANCEL"-results. This redirect is only performed, if the authorization of the user is performed on the sign-me Web Application by using the link from signatureProcess.signatureProcessURL. If the authorization is done under control of the partner application using the API, this kind of integration between the partner's business application and the user authorization screen needs to be made on the partner application.</p>

<b>Parameter</b>	<b>suppressFinalDialog</b>
<b>Type</b>	<b>boolean</b> [see <a href="#">[XSD]</a> ]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	This decides whether after the signing procedure call the user should go to a sign-me page to download the signed document or back to the partner website.

## Output of CreateSignatureProcess

<b>Parameter</b>	<b>signatureProcess</b>
<b>Type</b>	<b>SignatureProcess</b> [see <a href="#">5.29</a> ]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	This signature process contains all state and result operation to apply a signature on the uploaded items. The signature process id can be used for any operation on the process object like GetSignatureProcess to retrieve current state of final results, or CancelSignature.

## Exceptions of CreateSignatureProcess

In case of an error in the operation, a SignMeException is thrown. Detailed explanations of all possible error codes are contained in section [9](#).

## 4.3 UpdateSignatureProcess

The UpdateSignatureProcess operation is used by the partner application to set the content of a signature process, if this was not done in CreateSignatureProcess.

Hence this operation allows to update the content list based on information resulting from CreateSignatureProcess (e.g., creation of a use-case dependent signature certificate or signature process id).

Update of the signature process is only possible under following conditions:

1. Signature process is created by *CreateSignatureProcess* [4.2](#) with empty item list for a digest signing signature type.
2. Signature process is not already requested by signer using *GetSignatureProcess* [4.6](#)
3. Signature process is not already authorized by signer using *AuthorizeProcess* [4.27](#)

In order to fulfill the above conditions in case of electronic signing the partner can use updateSignatureProcess before the partner application redirects the signer (user) to the signatureProcessURL. Only then the user would perform a *GetSignatureProcess* [4.6](#) and *AuthorizeProcess* [4.27](#) by means of the sign-me web application.

In case of a signature for electronic sealing, there is no redirect to signatureProcessURL, but the phase is also between delivery of the final document to be sealed to the partner application and preparing the hash values to be signed. Depending on the signature type and appearance to prepare the hash value, some applications need the signing certificate to be used later for signing beforehand.

## Input of UpdateSignatureProcess

Parameter	Field Type	Occurrence
securityHeader	SecurityHeader	1
signatureProcessId	ObjectId	1
itemsToBeProcessed	ItemToBeProcessed	1..256

Table 4.2: Input Parameters of UpdateSignatureProcess

## Input Details of UpdateSignatureProcess

Parameter	securityHeader
Type	SecurityHeader [see 5.27]
Occurence	1
Meaning	The security header needs to be related to a valid login session with "PARTNER" role with quality "ONE_FACTOR".

Parameter	signatureProcessId
Type	ObjectId [see 6.32]
Occurence	1
Meaning	The id of the signature process whose content shall be updated.

Parameter	itemsToBeProcessed
Type	ItemToBeProcessed [see 5.14]
Occurence	1..256
Meaning	This contains a list of items which shall be signed and/or verified like in CreateSignatureProcess (see 4.2 itemsToBeProcessed). This list fully replaces the (empty) list delivered in CreateSignatureProcess. Hence this operation allows to update this list based on information resulting from CreateSignatureProcess (e.g., creation of a use-case dependent signature certificate or signature process id).

## Output of UpdateSignatureProcess

No output parameters defined for this operation.

## Exceptions of UpdateSignatureProcess

Throws an exception if update is not possible on the signature process, because the signer already started GetSignatureProcess or AuthorizeProcess by means of the sign-me web application.

In case of an error in the operation, a SignMeException is thrown. Detailed explanations of all possible error codes are contained in section 9.

## 4.4 GetSignerStatus

The GetSignerStatus operation is used by the partner application to check whether creating of a specific signature process and subsequent authorization of a signature by the user is possible based on the status of the user in the system and the signature type.

The operation returns a summary expression whether signing is possible or not. In case signing is not possible more detailed status information shows what needs to be done to bring the user into a sufficient status for signing.

The call also returns IDs of any pending identification process or signature processes of the signer with the indicated username. These IDs can be used to get a detailed status of these processes or offer a re-entry into the process for the user by redirecting him to the corresponding process URLs.

**User Guidance Alternatives:** The GetSignerStatus call can be used for enhanced user guidance or for clearing customer care issues. On the other hand, for a more straight forward implementation of user guidance, it is quite possible to ignore this call and just issue a CreateSignatureProcess (see 4.2) for some user and wait if the call is successful or if a SignMeException is thrown. The exception messages have an unique code and are relatively verbose (see section 9) unless there is a security issue, in which case there is a less explicit "unauthorized" exception.

As experience has shown for many security issues the user would be left in the dark, so this call is also utilized by the sign-me UI, which offers guidance if the user logs into the UI at [www.sign-me.de](http://www.sign-me.de). So a viable strategy can be if there is a security issue, the user shall login into sign-me and follow the steps recommended there. If the user lost his or her password then the only way forward remains a user triggered password reset in the login form, of course.

GetSignerStatus is usable for persons which shall electronically sign. GetSignerStatus is not applicable for electronic sealing. For checking, whether a certain sealing manager is registered in the system and the Seal-ID 2-factor Token is still valid, the call GetIdentity (see 4.12) and GetTokensForIdentity (see 4.25) can be used.

## Input of GetSignerStatus

Parameter	Field Type	Occurrence
securityHeader	SecurityHeader	1
signatureType	SignatureType	1
signerUsername	Username	0..1
signerIdentityId	ObjectId	0..1
validAtTimestamp	dateTime	0..1
minimumValidityDurationSec	integer	0..1
maxActiveSignatureProcesses	integer	0..1
maxFinalSignatureProcesses	integer	0..1

Table 4.3: Input Parameters of GetSignerStatus

## Input Details of GetSignerStatus

Parameter	securityHeader
Type	SecurityHeader [see 5.27]
Occurence	1
Meaning	The security header needs to be related to a valid login session with "PARTNER" role with quality "ONE_FACTOR".



<b>Parameter</b>	<b>signatureType</b>
<b>Type</b>	<b>SignatureType</b> [see 6.43]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The type of signature which shall be requested by the user person of the partner application.

<b>Parameter</b>	<b>signerUsername</b>
<b>Type</b>	<b>Username</b> [see 6.63]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The sign-me username of the person for whom the signer status shall be queried. This is the very same person for whom a signature process shall be created and who shall apply the signature.

<b>Parameter</b>	<b>signerIdentityId</b>
<b>Type</b>	<b>ObjectId</b> [see 6.32]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The object id of the identity received by calls to the sign-me system (e.g. getSignatureProcess).

<b>Parameter</b>	<b>validAtTimestamp</b>
<b>Type</b>	<b>dateTime</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	<p>The signer status is valid date/time at this point in time based on the current state. The date/time must be now or in the future. If omitted or in the past, the current point in time will be used.</p> <p>NOTE: If the user changes his status, e.g. locks his account or revokes certificates between now and validAtTimestamp the returned status now is obsolete, of course.</p>

<b>Parameter</b>	<b>minimumValidityDurationSec</b>
<b>Type</b>	<b>integer</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	<p>The signer status is valid at validAtTimestamp and then for at least minimumValidityDurationSec. The minimumValidityDurationSec must be greater/equal 300 seconds. If lower or omitted 300 seconds is used as default.</p> <p>NOTE: If the user changes his status, e.g. locks his account or revokes certificates between now and validAtTimestamp+minimumValidityDurationSec the returned status now is obsolete, of course.</p>

<b>Parameter</b>	<b>maxActiveSignatureProcesses</b>
<b>Type</b>	<b>integer</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The number of active signature processes requested for signing by the partner (state REQUESTED). Value must be between 0..100.

<b>Parameter</b>	<b>maxFinalSignatureProcesses</b>
<b>Type</b>	<b>integer</b> [see <a href="#">XSD</a> ]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The number of final signature processes requested for signing by the partner. Value must be between 0..100. The history will be limited by the value of signatureProcessRetrievableEndTimestamp of the signature process. For erroneous or cancelled processes, the data can be dropped earlier from the list.

## Output of GetSignerStatus

<b>Parameter</b>	<b>signerStatus</b>
<b>Type</b>	<b>SignerStatus</b> [see <a href="#">5.31</a> ]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	Returns the signer status for a username and signature type combination.

## Exceptions of GetSignerStatus

Thrown in case of technical errors or if the partner does not have access rights for the signature type.

In case of an error in the operation, a SignMeException is thrown. Detailed explanations of all possible error codes are contained in section [9](#).

## 4.5 CancelSignatureProcess

The CancelSignatureProcess operation is used by the partner application to cancel a signature process, which is no longer needed, because, e.g. the user does not want to sign, because he rejected an offer or similar. The operation will remove the signature process and items to be processed without electronic signing/sealing if the signature process is not in state signatureState OK\_[UN]VERIFIED. If a signature for electronic signing/sealing was already performed, cancel will have the result to shorten the retention time of the signature or signed document on the system (signatureProcessRetrievableEndTimestamp).

## Input of CancelSignatureProcess

<b>Parameter</b>	<b>securityHeader</b>
<b>Type</b>	<b>SecurityHeader</b> [see <a href="#">5.27</a> ]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The security header needs to be related to a valid login session with "PARTNER" role with quality "ONE_FACTOR".

<b>Parameter</b>	<b>signatureProcessId</b>
<b>Type</b>	<b>ObjectId</b> [see <a href="#">6.32</a> ]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The id of the signature process which shall be cancelled.

## Output of CancelSignatureProcess

No output parameters defined for this operation.

## Exceptions of CancelSignatureProcess

In case of an error in the operation, a SignMeException is thrown. Detailed explanations of all possible error codes are contained in section 9.

## 4.6 GetSignatureProcess

The GetSignatureProcess operation allows a partner or user (logged in through sign-me or partner application) to get the status or results of the signature process.

### Input of GetSignatureProcess

Parameter	Field Type	Occurrence
securityHeader	SecurityHeader	1
signatureProcessId	ObjectId	1
getContentOrSignedContent	boolean	1
getVerificationReport	boolean	1
getSignerIdentity	boolean	1

Table 4.4: Input Parameters of GetSignatureProcess

### Input Details of GetSignatureProcess

Parameter	securityHeader
Type	SecurityHeader [see 5.27]
Occurence	1
Meaning	The security header needs to be related to a valid login session with "PARTNER" or "USER" role with quality "ONE_FACTOR". Only the partner which created the process, or the person which shall authorize the signature can access the signature process and its content.

Parameter	signatureProcessId
Type	ObjectId [see 6.32]
Occurence	1
Meaning	The id of the signature process whose status shall be queried.

<b>Parameter</b>	<b>getContentOrSignedContent</b>
<b>Type</b>	<b>boolean</b> [see <a href="#">XSD</a> ]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	<p>This parameter enables the delivery of the contentOrSignedContent element in the status response. This should be used conservatively and as a rule only be used once, after the document(s)/contents(s) is/(are) signed or verified (state OK_UNVERIFIED or OK_VERIFIED, see section 6.42).</p> <p>Normally, the status of the signature request should be polled with this option set to false, to avoid unnecessary load on the back-end. Sample applications in the language specific access modules show, how the wait time can be controlled in order to save resources in the back-end.</p> <p>Before a signature is applied, this parameter can also be used to get the unsigned content, which was delivered in 'content' parameter of 'ItemsToBeProcessed' in operation 'createSignatureProcess'.</p>

<b>Parameter</b>	<b>getVerificationReport</b>
<b>Type</b>	<b>boolean</b> [see <a href="#">XSD</a> ]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	<p>This parameter enables the delivery of the verificationReport in the status response. This should be used conservatively and as a rule only be used once, after the document is signed or verified. Normally the status of the signature request should to be polled with this option set to false, to avoid unnecessary load on the back-end.</p> <p>RESTRICTION: This option only possible, if the document to verify against is part of the signature process, i.e. it is not possible within signing process using a digest signing signature type.</p>

<b>Parameter</b>	<b>getSignerIdentity</b>
<b>Type</b>	<b>boolean</b> [see <a href="#">XSD</a> ]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	<p>This parameter enables the delivery of the identity id of the signer in the status response. This option will only return the signerIdentityId, after the document is signed or verified (state signature at least OK_UNVERIFIED[_TIMEOUT—ERROR] or OK_VERIFIED). Otherwise there is only a shortened signerIdentityId, which is enough for reference, but not valid for getting all identity attributes by means of getIdentity(signerIdentityId).</p> <p>This option is often not needed anyway, because the identity of the signer is already preset in the signature process, in other words if the signature is successful, it has been done by a known identity.</p> <p>However, for some applications it is useful to compare the complete set of attributes of an identity using the id which can be retrieved with this option. Therefore, the full set of attributes of an identity can be retrieved using getIdentity(signerIdentityId) after a successful signature.</p>

## Output of GetSignatureProcess

Parameter	signatureProcess
Type	SignatureProcess [see 5.29]
Occurrence	1
Meaning	

## Exceptions of GetSignatureProcess

In case of an error in the operation, a SignMeException is thrown. Detailed explanations of all possible error codes are contained in section 9.

## 4.7 GetSignatureTemplate

The GetSignatureProcess operation allows a partner to get the definition of a signature type by means of a signature template.

### Input of GetSignatureTemplate

Parameter	Field Type	Occurrence
securityHeader	SecurityHeader	1
signatureTemplateId	ObjectId	1
includeCoins	boolean	0..1

Table 4.5: Input Parameters of GetSignatureTemplate

### Input Details of GetSignatureTemplate

Parameter	securityHeader
Type	SecurityHeader [see 5.27]
Occurrence	1
Meaning	

Parameter	signatureTemplateId
Type	ObjectId [see 6.32]
Occurrence	1
Meaning	Object Id of the signature template to be returned.

Parameter	includeCoins
Type	boolean [see [XSD]]
Occurrence	0..1
Meaning	If present and true, includes the required coins into the response.

## Output of GetSignatureTemplate

Parameter	signatureTemplate
Type	SignatureTemplate [see 5.30]
Occurrence	1
Meaning	The signature template.

## Exceptions of GetSignatureTemplate

In case of an error in the operation, a SignMeException is thrown. Detailed explanations of all possible error codes are contained in section 9.

## 4.8 CreateRegistrationProcess

The CreateRegistrationProcess operation is used by the partner application to create a registration process. The registration process allows to guide and track the registration of the user and retrieve the results of the registration. This is an alternative to the simple-API-call registration using RegisterIdentity. The advantage of CreateRegistrationProcess is, that the sign-me UI can be re-used for registration by means of registrationProcessURL, and own UI screens do not have to be developed. In case the sign-me UI is used for registration, which is recommended, person.username and person.passwd do not have to be preset. There is also no need to use UpdateRegistrationProcess, because this is applied by the sign-me UI. The recommended use is shown in detail in partnerCreatesRegistrationProcess of the RegistrationProcessSample.java or RegistrationProcessSample.cs.

After the registration process is created the partner can re-direct the user to the sign-me system using a unique URL of the registration process (registrationProcessURL). The sign-me system will re-direct to the partner application again after completion of the registration (redirectURLs).

After the registration is completed the partner application can retrieve the results using GetRegistrationProcess operation.

## Input of CreateRegistrationProcess

Parameter	Field Type	Occurrence
securityHeader	SecurityHeader	1
redirectURLs	RedirectURLs	0..1
person	Person	0..1

Table 4.6: Input Parameters of CreateRegistrationProcess

## Input Details of CreateRegistrationProcess

Parameter	securityHeader
Type	SecurityHeader [see 5.27]
Occurrence	1
Meaning	The security header needs to be related to a valid login session with "PARTNER" role with quality "ONE_FACTOR".

<b>Parameter</b>	<b>redirectURLs</b>
<b>Type</b>	<b>RedirectURLs</b> [see 5.24]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	This can contain URLs to redirect to in case of "OK"-, "FAIL"-, and "CANCEL"-results. This redirect is only performed, if the authorization of the user is performed on the sign-me Web Application by using the link from signatureProcess.signatureProcessURL. If the authorization is done under control of the partner application using the API, this kind of integration between the partner's business application and the user authorization screen needs to be made on the partner application.

<b>Parameter</b>	<b>person</b>
<b>Type</b>	<b>Person</b> [see 5.21]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	<p>The optional preset data of a person to begin with the registration. If provided, as a minimum, the person object must contain a non-empty familyNames and givenNames as well as (Identity.)emailAddress element. See Person datatype for more details. If provided, and if password is preset, then username must also be set.</p> <p>However, in case the sign-me UI is used for registration over registrationProcessURL, which is recommended, person.username and person.passwd do not have to be preset. Preset values of person.username will be overwritten by person.emailAddress and preset person.passwd will be ignored.</p> <p>There is also no need to use UpdateRegistrationProcess, because this is applied by the sign-me UI.</p>

## Output of CreateRegistrationProcess

<b>Parameter</b>	<b>registrationProcess</b>
<b>Type</b>	<b>RegistrationProcess</b> [see 5.25]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	This registration process contains the current state of the registration. The registration process id can be used for any operation on the process object like GetRegistrationProcess to retrieve current state of final results, or UpdateRegistrationProcess to change any presets or complete the registration with final data.

## Exceptions of CreateRegistrationProcess

Throws an exception if parameters are missing or if not authorized.

In case of an error in the operation, a SignMeException is thrown. Detailed explanations of all possible error codes are contained in section 9.

## 4.9 UpdateRegistrationProcess

The UpdateRegistrationProcess operation is used by the partner application to set or update the content of a registration process, if this was not done in CreateRegistrationProcess.

Update of the signature process is only possible if the process is in state 'REQUESTED'.

In case the sign-me UI is used for registration over registrationProcessURL, which is recommended, there no need to use UpdateRegistrationProcess, because this is applied by the sign-me UI. The recommended use is shown in detail in partnerCreatesRegistrationProcess of the RegistrationProcessSample.java or RegistrationProcessSample.cs.

### Input of UpdateRegistrationProcess

Parameter	Field Type	Occurrence
securityHeader	SecurityHeader	1
registrationProcessId	ObjectId	1
person	Person	0..1
complete	boolean	0..1
suppressConfirmationEmail	boolean	0..1

Table 4.7: Input Parameters of UpdateRegistrationProcess

### Input Details of UpdateRegistrationProcess

Parameter	securityHeader
Type	SecurityHeader [see 5.27]
Occurrence	1
Meaning	The security header needs to be related to a valid login session with "PARTNER" role with quality "ONE_FACTOR".

Parameter	registrationProcessId
Type	ObjectId [see 6.32]
Occurrence	1
Meaning	The id of the registration process whose content shall be updated.

Parameter	person
Type	Person [see 5.21]
Occurrence	0..1
Meaning	The optional data of a person to update or complete the registration. If provided, as a minimum, the person object must contain a non-empty familyNames and givenNames as well as (Identity.)emailAddress element. See Person datatype for more details. If provided, and if password is preset, then username must also be set. However, in case the sign-me UI is used for registration over registrationProcessURL, which is recommended, there is no need to use UpdateRegistrationProcess at all, because this is applied by the sign-me UI.

Parameter	complete
Type	boolean [see [XSD]]
Occurrence	0..1
Meaning	If present and true, the registration will be completed.



<b>Parameter</b>	<b>suppressConfirmationEmail</b>
<b>Type</b>	<b>boolean</b> [see <a href="#">XSD</a> ]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If present and true, there will be no confirmation email which is sent to the identity which will be registered. Only relevant if complete=true.

### Output of UpdateRegistrationProcess

No output parameters defined for this operation.

### Exceptions of UpdateRegistrationProcess

Throws an exception if update is not possible, if parameters are missing or if not authorized.

In case of an error in the operation, a SignMeException is thrown. Detailed explanations of all possible error codes are contained in section [9](#).

## 4.10 GetRegistrationProcess

The GetRegistrationProcess operation allows a partner (logged in through sign-me or partner application) to get the status or results of the registration process.

### Input of GetRegistrationProcess

<b>Parameter</b>	<b>securityHeader</b>
<b>Type</b>	<b>SecurityHeader</b> [see <a href="#">5.27</a> ]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The security header needs to be related to a valid login session with "PARTNER" role with quality "ONE_FACTOR". Only the partner which created the process can access the registration process and its content.

<b>Parameter</b>	<b>registrationProcessId</b>
<b>Type</b>	<b>ObjectId</b> [see <a href="#">6.32</a> ]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The id of the registration process whose status or results shall be queried.

### Output of GetRegistrationProcess

<b>Parameter</b>	<b>registrationProcess</b>
<b>Type</b>	<b>RegistrationProcess</b> [see <a href="#">5.25</a> ]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	

### Exceptions of GetRegistrationProcess

In case of an error in the operation, a SignMeException is thrown. Detailed explanations of all possible error codes are contained in section [9](#).

## 4.11 RegisterIdentity

By means of register identity the partner can register new natural persons as sign-me users. After the operation is successful, the registered persons are in state UNVERIFIED (see 6.6). An email is sent to the person, which needs to be confirmed (by pressing the confirmation link). Then the state will change into COMMUNICATION\_VERIFIED.

NOTE: register of organizations are not supported in this server version.

### Input of RegisterIdentity

Parameter	Field Type	Occurrence
securityHeader	SecurityHeader	1
suppressConfirmationEmail	boolean	0..1
person	Person	1
organization	Organization	1

Table 4.8: Input Parameters of RegisterIdentity

### Input Details of RegisterIdentity

Parameter	securityHeader
Type	SecurityHeader [see 5.27]
Occurrence	1
Meaning	The security header needs to be related to a valid login session with "PARTNER" role with quality "ONE_FACTOR".

Parameter	suppressConfirmationEmail
Type	boolean [see XSD]
Occurrence	0..1
Meaning	If present and true, there will be no confirmation email which is sent to the identity which will be registered.

Parameter	person
Type	Person [see 5.21]
Occurrence	1
Meaning	The identity to register is a person.

Parameter	organization
Type	Organization [see 5.20]
Occurrence	1
Meaning	The identity to register is an organization. NOTE: not possible in this release.

## Output of RegisterIdentity

Parameter	identityId
Type	ObjectId [see 6.32]
Occurrence	1
Meaning	The identityId of the newly registered person.

## Exceptions of RegisterIdentity

In case of an error in the operation, a SignMeException is thrown. Detailed explanations of all possible error codes are contained in section 9.

## 4.12 GetIdentity

By means of GetIdentity partners or users can query identification data of registered identities. See 'securityHeader' documentation for access rules. The GetIdentityResponse contains a data structure for persons or organizations based on the type of identity, which was queried. Based on access rules a selection parameter in identitySelector, some attributes of the identity can be filtered (and set to null) in the response.

## Input of GetIdentity

Parameter	Field Type	Occurrence
securityHeader	SecurityHeader	1
identitySelector	IdentitySelector	1
includeTimestamps	boolean	0..1
includeAllowedRoles	boolean	0..1
includeVerificationInfo	boolean	0..1
includeCoins	boolean	0..1
includeLanguageTag	boolean	0..1

Table 4.9: Input Parameters of GetIdentity

## Input Details of GetIdentity

Parameter	securityHeader
Type	SecurityHeader [see 5.27]
Occurrence	1
Meaning	The security header needs to be related to a valid login session with "PARTNER" role with quality "ONE_FACTOR".

Parameter	identitySelector
Type	IdentitySelector [see 5.11]
Occurrence	1
Meaning	The identitySelector selects an identity based on username or identityId. If the selection is made based on username, restrictions may apply on the set of returned values.

<b>Parameter</b>	<b>includeTimestamps</b>
<b>Type</b>	<b>boolean</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If present and true, includes the time-stamps (modified only if modified at least once).

<b>Parameter</b>	<b>includeAllowedRoles</b>
<b>Type</b>	<b>boolean</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If present and true, includes the allowedRoleNames element.

<b>Parameter</b>	<b>includeVerificationInfo</b>
<b>Type</b>	<b>boolean</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If present and true, includes the email address and mobile phone number verification info elements.

<b>Parameter</b>	<b>includeCoins</b>
<b>Type</b>	<b>boolean</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If present and true, includes the coins account into the response.

<b>Parameter</b>	<b>includeLanguageTag</b>
<b>Type</b>	<b>boolean</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If present and true, includes the language tag into the response.

## Output of GetIdentity

<b>Parameter</b>	<b>person</b>
<b>Type</b>	<b>Person</b> [see <a href="#">5.21</a> ]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	This is returned, if the selected identity was a person.

<b>Parameter</b>	<b>organization</b>
<b>Type</b>	<b>Organization</b> [see <a href="#">5.20</a> ]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	This is returned, if the selected identity was an organization.

## Exceptions of GetIdentity

In case of an error in the operation, a `SignMeException` is thrown. Detailed explanations of all possible error codes are contained in section [9](#).

## 4.13 ConfirmCommunication

By confirming the communication of a natural person indicated by `identitySelector`, the identity which is logged-in confirms, that it is reachable via a specific communication

channel. The person to confirm communication need to be logged in through sign-me web application.

NOTE: confirming communication of organizations is not supported in this server version.

### Input of ConfirmCommunication

Parameter	securityHeader
Type	SecurityHeader [see 5.27]
Occurrence	1
Meaning	The security header needs to be related to a valid login session with "USER" role with quality "ONE_FACTOR".

Parameter	code
Type	UTF8String256 [see 6.56]
Occurrence	1
Meaning	The code is an opaque string which was transferred over the communication channel to the confirming identity.

### Output of ConfirmCommunication

No output parameters defined for this operation.

### Exceptions of ConfirmCommunication

In case of an error in the operation, a SignMeException is thrown. Detailed explanations of all possible error codes are contained in section 9.

## 4.14 CreateIdentityVerificationProcess

The identity (this release: natural person) indicated by identitySelector shall be verified by an identification verifier. The identity information may be incomplete or inaccurate, except the mandatory attributes, and will be completed or corrected by the verifier.

### Input of CreateIdentityVerificationProcess

Parameter	Field Type	Occurrence
securityHeader	SecurityHeader	1
preferredIdentityVerificationType	IdentityVerificationType	0..1
username	Username	0..1
identityId	ObjectId	0..1
person	Person	0..1
suppressConfirmationEmail	boolean	0..1
redirectURLs	RedirectURLs	0..1

Table 4.10: Input Parameters of CreateIdentityVerificationProcess

## Input Details of CreateIdentityVerificationProcess

<b>Parameter</b>	<b>securityHeader</b>
<b>Type</b>	<b>SecurityHeader</b> [see 5.27]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The security header needs to be related to a valid login session with "PARTNER" role with quality "ONE_FACTOR".

<b>Parameter</b>	<b>preferredIdentityVerificationType</b>
<b>Type</b>	<b>IdentityVerificationType</b> [see 6.24]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If present, a specific verification type will be used. Otherwise, the default verification type will be applied.

<b>Parameter</b>	<b>username</b>
<b>Type</b>	<b>Username</b> [see 6.63]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	<p>The username of the identity to be verified. The identity needs to be in state COMMUNICATION_VERIFIED or UNVERIFIED (only in fast-track verification types, which will set the communicationAlreadyVerified attribute to 'true' in the response).</p> <p>The identity will be brought into state LAWFUL_VERIFICATION_REQUESTED, first then the verification starts (see use cases for that). If an identity verification process is already in progress LAWFUL_VERIFICATION_REQUESTED or LAWFUL_VERIFICATION_STARTED for the identity with username, then the identifications will be implicitly cancelled for the identity verification process, which is in progress (this is equivalent to calling CancelIdentityVerificationProcess see 4.15)).</p>

<b>Parameter</b>	<b>identityId</b>
<b>Type</b>	<b>ObjectId</b> [see 6.32]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The object id of the identity received by calls to the sign-me system (e.g. getSignatureProcess).

<b>Parameter</b>	<b>person</b>
<b>Type</b>	<b>Person</b> [see 5.21]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The person to register and identify in the system.

<b>Parameter</b>	<b>suppressConfirmationEmail</b>
<b>Type</b>	<b>boolean</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If present and true, there will be no confirmation email which is sent to the identity which will be registered.

<b>Parameter</b>	<b>redirectURLs</b>
<b>Type</b>	<b>RedirectURLs</b> [see 5.24]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	This can contain URLs to redirect to in case of "OK"-, "FAIL"-, and "CANCEL"-results.

### Output of CreateIdentityVerificationProcess

<b>Parameter</b>	<b>identityVerificationProcess</b>
<b>Type</b>	<b>IdentityVerificationProcess</b> [see 5.12]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The id and initial state of the verification process, as well as the link, where the partner shall redirect the user in order to start the verification process.

### Exceptions of CreateIdentityVerificationProcess

In case of an error in the operation, a `SignMeException` is thrown. Detailed explanations of all possible error codes are contained in section 9.

## 4.15 CancelIdentityVerificationProcess

The `CancelIdentityVerificationProcess` operation is used by the partner application to cancel an identity verification process, which is no longer needed, because, e.g. the user does not want to sign, because he rejected an offer or similar. The operation can only be performed in state `signatureState=LAWFUL_VERIFICATION_REQUESTED` or `LAWFUL_VERIFICATION_STARTED`.

### Input of CancelIdentityVerificationProcess

<b>Parameter</b>	<b>securityHeader</b>
<b>Type</b>	<b>SecurityHeader</b> [see 5.27]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The security header needs to be related to a valid login session with "PARTNER" role with quality "ONE_FACTOR".

<b>Parameter</b>	<b>identityVerificationProcessId</b>
<b>Type</b>	<b>ObjectId</b> [see 6.32]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The id of the identity verification process which shall be cancelled.

### Output of CancelIdentityVerificationProcess

No output parameters defined for this operation.

### Exceptions of CancelIdentityVerificationProcess

In case of an error in the operation, a `SignMeException` is thrown. Detailed explanations of all possible error codes are contained in section 9.

## 4.16 GetIdentityVerificationProcess

This operation is for partners and identities to be verified to query the state of the verification process.

### Input of GetIdentityVerificationProcess

Parameter	Field Type	Occurrence
securityHeader	SecurityHeader	1
identityVerificationProces.. ..sId	ObjectId	1
includeCoins	boolean	0..1
authenticationData	GenericAuthenticationData	0..10

Table 4.11: Input Parameters of GetIdentityVerificationProcess

### Input Details of GetIdentityVerificationProcess

Parameter	securityHeader
Type	SecurityHeader [see 5.27]
Occurrence	1
Meaning	The security header needs to be related to a valid login session with "PARTNER" role with quality "ONE_FACTOR" or "USER" role. Only the partner which created the process and the identity which shall be verified may access the process and it's content.

Parameter	identityVerificationProcessId
Type	ObjectId [see 6.32]
Occurrence	1
Meaning	The key which was obtained by CreateIdentityVerificationProcessResponse. This will be used for accessing the verification process.

Parameter	includeCoins
Type	boolean [see [XSD]]
Occurrence	0..1
Meaning	If present and true, includes recharged coins into the response.

Parameter	authenticationData
Type	GenericAuthenticationData [see 6.18]
Occurrence	0..10
Meaning	If present, this element transfers authentication data for specific identification mechanisms.



## Output of GetIdentityVerificationProcess

Parameter	identityVerificationProcess
Type	IdentityVerificationProcess [see 5.12]
Occurrence	1
Meaning	

## Exceptions of GetIdentityVerificationProcess

In case of an error in the operation, a SignMeException is thrown. Detailed explanations of all possible error codes are contained in section 9.

## 4.17 ManageIdentity

ManageIdentity allows to change selected identity attributes under control of the owner of the identity. Only available, if the identities are logged in over specific partners.

### Input of ManageIdentity

Parameter	Field Type	Occurrence
securityHeader	SecurityHeader	1
identityId	ObjectId	0..1
username	Username	0..1
changeSet	ManageIdentityChangeSet	1

Table 4.12: Input Parameters of ManageIdentity

### Input Details of ManageIdentity

Parameter	securityHeader
Type	SecurityHeader [see 5.27]
Occurrence	1
Meaning	The security header needs to be related to a valid login session with "USER" role with quality "ONE_FACTOR" over specific partners or directly as a specific partner for some sub-operations triggered by specific flags (generateNewPasswd).

Parameter	identityId
Type	ObjectId [see 6.32]
Occurrence	0..1
Meaning	The ID of the Identity (Person, later Organization), where the modifiable identity data shall be managed. Either "identityId" or "username" must be present. Username is only possible in restricted cases, which are explicitly stated und the options of ManageIdentityChangeSet.

<b>Parameter</b>	<b>username</b>
<b>Type</b>	<b>Username</b> [see 6.63]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Alternatively to identityId, the username of the Identity (Person, later Organization), where the modifiable identity data shall be managed. Either "identityId" or username must be present. Username is only possible in restricted cases, which are explicitly stated und the options of ManageIdentityChangeSet.

<b>Parameter</b>	<b>changeSet</b>
<b>Type</b>	<b>ManageIdentityChangeSet</b> [see 5.15]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The change set specifies the parameters which shall be changed in the attribute set of the registered entity.

### Output of ManageIdentity

No output parameters defined for this operation.

### Exceptions of ManageIdentity

In case of an error in the operation, a SignMeException is thrown. Detailed explanations of all possible error codes are contained in section 9.

## 4.18 ManageSubscription

ManageSubscription allows to create or change subscriptions for the identity.

CAUTION: if the parameter startingOffset is set to  $\geq 1$ , please make sure that this value is immediately correct for the intended usage scenario. The subscription will be in state "PENDING\_ACTIVATION" until the starting month indicated by startingOffset is reached. For a subscription once created, it is not possible to shift the start time further into the past, i.e. to let it begin earlier. Therefore, the safe setting for "startingOffset" will be 0, which means immediate activation of the subscription.

The total value of subscription duration (existing plus added by this call) must not exceed maximum of 600 months, otherwise an error is generated.

The subscription feature is only available, if the identities are logged in over specific partners.

### Input of ManageSubscription

Parameter	Field Type	Occurrence
securityHeader	SecurityHeader	1
identityId	ObjectId	0..1
username	Username	0..1
startingOffset	int	1
duration	int	1

Table 4.13: Input Parameters of ManageSubscription

## Input Details of ManageSubscription

<b>Parameter</b>	<b>securityHeader</b>
<b>Type</b>	<b>SecurityHeader</b> [see 5.27]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The security header needs to be related to a valid login session with "USER" role with quality "ONE_FACTOR" over specific partners.

<b>Parameter</b>	<b>identityId</b>
<b>Type</b>	<b>ObjectId</b> [see 6.32]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The ID of the Identity (Person, later Organization), where the modifiable identity data shall be managed. Either "identityId" or username must be present.

<b>Parameter</b>	<b>username</b>
<b>Type</b>	<b>Username</b> [see 6.63]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Alternatively to identityId, the username of the Identity (Person, later Organization), where the modifiable identity data shall be managed. Either "identityId" or username must be present.

<b>Parameter</b>	<b>startingOffset</b>
<b>Type</b>	<b>int</b> [see XSD]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The starting month offset (any of 0..11) for the subscription, if no subscription exists for the identity. The value 0 means, that the subscriptions shall start in the same month immediately. The value 1 means, that the subscription shall start in at the beginning of next month and so forth. If the subscription is already active or pending to be activated for the identity, this parameter will be ignored. Date/time of beginning of Month is based on MET/MEST.

<b>Parameter</b>	<b>duration</b>
<b>Type</b>	<b>int</b> [see XSD]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The duration in number of full calendar months where the subscription shall be active starting from startingOffset. If the subscription is already active or pending to be activated for the identity, the duration in number of full calendar months is simply added starting from the current subscription end date. Date/time of beginning of Month is based on MET/MEST.

## Output of ManageSubscription

Parameter	subscription
Type	Subscription [see 5.32]
Occurrence	1
Meaning	The resulting subscription status.

## Exceptions of ManageSubscription

In case of an error in the operation, a `SignMeException` is thrown. Detailed explanations of all possible error codes are contained in section 9.

## 4.19 ManageToken

`ManageToken` allows to manage tokens for the identity.

The feature is only available, if the identities are logged in over specific partners.

### Input of ManageToken

Parameter	Field Type	Occurrence
securityHeader	SecurityHeader	1
identityId	ObjectId	0..1
manageTokenParameters	ManageTokenParameters	1

Table 4.14: Input Parameters of ManageToken

### Input Details of ManageToken

Parameter	securityHeader
Type	SecurityHeader [see 5.27]
Occurrence	1
Meaning	The security header needs to be related to a valid login session with "USER" role with quality "ONE_FACTOR" over specific partners.

Parameter	identityId
Type	ObjectId [see 6.32]
Occurrence	0..1
Meaning	The ID of the Identity (Person, later Organization), where the modifiable identity data shall be managed.

Parameter	manageTokenParameters
Type	ManageTokenParameters [see 5.16]
Occurrence	1
Meaning	Specific parameters for token management.

## Output of ManageToken

Parameter	<b>manageTokenResult</b>
Type	<b>ManageTokenResult</b> [see 5.17]
Occurrence	<b>1</b>
Meaning	The resulting parameters.

## Exceptions of ManageToken

In case of an error in the operation, a `SignMeException` is thrown. Detailed explanations of all possible error codes are contained in section 9.

## 4.20 GetCertificate

This operation allows to get certificates or complete certificate chains of certificates for the signature tokens in sign-me.

### Input of GetCertificate

Parameter	Field Type	Occurrence
<b>securityHeader</b>	<b>SecurityHeader</b>	<b>1</b>
<b>certificateId</b>	<b>ObjectId</b>	<b>1</b>
<b>includeChain</b>	<b>boolean</b>	<b>1</b>

Table 4.15: Input Parameters of GetCertificate

### Input Details of GetCertificate

Parameter	<b>securityHeader</b>
Type	<b>SecurityHeader</b> [see 5.27]
Occurrence	<b>1</b>
Meaning	The security header needs to be related to a valid login session.

Parameter	<b>certificateId</b>
Type	<b>ObjectId</b> [see 6.32]
Occurrence	<b>1</b>
Meaning	The object id of the certificate queried from the token descriptor.

Parameter	<b>includeChain</b>
Type	<b>boolean</b> [see [XSD]]
Occurrence	<b>1</b>
Meaning	Includes the certificate chain up to root in the response.

## Output of GetCertificate

Parameter	certificate
Type	Certificate [see 5.5]
Occurence	1..8
Meaning	The chain will be included as far as known. The subject certificate will be first in list then its concerning issuer.

## Exceptions of GetCertificate

In case of an error in the operation, a SignMeException is thrown. Detailed explanations of all possible error codes are contained in section 9.

## 4.21 GetCertificateTemplate

Not supported in this server version.

### Input of GetCertificateTemplate

Parameter	securityHeader
Type	SecurityHeader [see 5.27]
Occurence	1
Meaning	

Parameter	certificateTemplateId
Type	ObjectId [see 6.32]
Occurence	1
Meaning	

### Output of GetCertificateTemplate

Parameter	certificateTemplate
Type	CertificateTemplate [see 5.6]
Occurence	1
Meaning	

## Exceptions of GetCertificateTemplate

In case of an error in the operation, a SignMeException is thrown. Detailed explanations of all possible error codes are contained in section 9.

## 4.22 GetIdentificationTemplate

Not supported in this server version.

### Input of GetIdentificationTemplate

Parameter	securityHeader
Type	SecurityHeader [see 5.27]
Occurrence	1
Meaning	

Parameter	identTemplateId
Type	ObjectId [see 6.32]
Occurrence	1
Meaning	

### Output of GetIdentificationTemplate

Parameter	identificationTemplate
Type	IdentificationTemplate [see 5.9]
Occurrence	1
Meaning	

### Exceptions of GetIdentificationTemplate

In case of an error in the operation, a SignMeException is thrown. Detailed explanations of all possible error codes are contained in section 9.

## 4.23 GetToken

By means of this operation partners or persons (logged in through sign-me or partner application) can get the token data. Only meta data and public keys (through the certificate) are returned by this operation.

NOTE: This is not the operation to get the access token for a login session. For this the Login (see section 4.29) operation will be used.

### Input of GetToken

Parameter	Field Type	Occurrence
securityHeader	SecurityHeader	1
tokenId	ObjectId	1
includeSubjectCertificateI..	boolean	0..1
..fAny		
includeActivationProcessIf..	boolean	0..1
..Any		

Table 4.16: Input Parameters of GetToken

### Input Details of GetToken

Parameter	<b>securityHeader</b>
Type	<b>SecurityHeader</b> [see <a href="#">5.27</a> ]
Occurrence	<b>1</b>
Meaning	The security header needs to be related to a valid login session with "PARTNER" or "USER" role with quality "ONE_FACTOR". Only partners, or the owning person can access the token.

Parameter	<b>tokenId</b>
Type	<b>ObjectId</b> [see <a href="#">6.32</a> ]
Occurrence	<b>1</b>
Meaning	Object Id of the token descriptor to be returned.

Parameter	<b>includeSubjectCertificateIfAny</b>
Type	<b>boolean</b> [see <a href="#">XSD</a> ]
Occurrence	<b>0..1</b>
Meaning	Includes the subject certificate if the token has a one.

Parameter	<b>includeActivationProcessIfAny</b>
Type	<b>boolean</b> [see <a href="#">XSD</a> ]
Occurrence	<b>0..1</b>
Meaning	Includes the token activation process descriptor if the token has one.

### Output of GetToken

Parameter	<b>token</b>
Type	<b>Token</b> [see <a href="#">5.33</a> ]
Occurrence	<b>1</b>
Meaning	Returns the token descriptor for token with tokenId.

### Exceptions of GetToken

In case of an error in the operation, a `SignMeException` is thrown. Detailed explanations of all possible error codes are contained in section [9](#).

## 4.24 GetAuthenticationTokenTypesForIdentity

By means of this partners can get the types of tokens available for authentication of identities.

NOTE: This is not the operation to get the access token for a login session. For this the Login (see section [4.29](#)) operation will be used.



### Input of GetAuthenticationTokenTypesForIdentity

Parameter	securityHeader
Type	SecurityHeader [see 5.27]
Occurrence	1
Meaning	The security header needs to be related to a valid login session with "PARTNER" or "USER" role with quality "ONE_FACTOR". Only partners, or the owning person can access the token.

Parameter	identitySelector
Type	IdentitySelector [see 5.11]
Occurrence	1
Meaning	Identity for which the authentication token descriptors are to be returned.

### Output of GetAuthenticationTokenTypesForIdentity

Parameter	authenticationTokenTypes
Type	AuthenticationTokenType [see 6.5]
Occurrence	0..16
Meaning	Returns the token descriptors for all authentication tokens of identity.

### Exceptions of GetAuthenticationTokenTypesForIdentity

In case of an error in the operation, a SignMeException is thrown. Detailed explanations of all possible error codes are contained in section 9.

## 4.25 GetTokensForIdentity

By means of this operation partners can get handles and descriptions of tokens of their users. By means of this information partners can offer choices to their users, for which tokens to use for different operations.

NOTE: This is not the operation to get the access token for a login session. For this the Login (see section 4.29) operation will be used.

## Input of GetTokensForIdentity

Parameter	Field Type	Occurrence
securityHeader	SecurityHeader	1
identitySelector	IdentitySelector	1
havingTokenCapabilities	TokenCapability	0..8
includeOnlyIfDefaultToken	boolean	0..1
includeOnlyIfHasValidCerti.. ..ficateOnly	boolean	0..1
includeSubjectCertificateI.. ..fAny	boolean	0..1
includeActivationProcessIf.. ..Any	boolean	0..1

Table 4.17: Input Parameters of GetTokensForIdentity

## Input Details of GetTokensForIdentity

Parameter	securityHeader
Type	SecurityHeader [see 5.27]
Occurence	1
Meaning	The security header needs to be related to a valid login session with "PARTNER" or "USER" role with quality "ONE_FACTOR". Only partners, or the owning person can access the token.

Parameter	identitySelector
Type	IdentitySelector [see 5.11]
Occurence	1
Meaning	Identity for which the token descriptors are to be returned.

Parameter	havingTokenCapabilities
Type	TokenCapability [see 6.50]
Occurence	0..8
Meaning	Includes all tokens having any of the capabilities in the list. Otherwise all tokens of the identity are returned.

Parameter	includeOnlyIfDefaultToken
Type	boolean [see [XSD]]
Occurence	0..1
Meaning	Includes the token only, if it is a default token. If not present, includes default/non-default tokens.

Parameter	includeOnlyIfHasValidCertificateOnly
Type	boolean [see [XSD]]
Occurence	0..1
Meaning	If true, includes the token only, if it has a valid certificate. If false or not present, includes tokens with valid or invalid certificates.

<b>Parameter</b>	<b>includeSubjectCertificateIfAny</b>
<b>Type</b>	<b>boolean</b> [see <a href="#">XSD</a> ]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Includes the subject certificate if the token has one.

<b>Parameter</b>	<b>includeActivationProcessIfAny</b>
<b>Type</b>	<b>boolean</b> [see <a href="#">XSD</a> ]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Includes the token activation process descriptor if the token has one.

### Output of GetTokensForIdentity

<b>Parameter</b>	<b>token</b>
<b>Type</b>	<b>Token</b> [see <a href="#">5.33</a> ]
<b>Occurrence</b>	<b>0..16</b>
<b>Meaning</b>	Returns the token descriptors for all tokens which fulfill select criteria.

### Exceptions of GetTokensForIdentity

In case of an error in the operation, a `SignMeException` is thrown. Detailed explanations of all possible error codes are contained in section [9](#).

## 4.26 RevokeCertificateOfTokenActivationProcess

The certificate which was created during the token activation process on this system can be revoked by the partner application which created the activation process.

### Input of RevokeCertificateOfTokenActivationProcess

<b>Parameter</b>	<b>securityHeader</b>
<b>Type</b>	<b>SecurityHeader</b> [see <a href="#">5.27</a> ]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The security header needs to be related to a valid login session with "PARTNER" or "USER" role with quality "ONE_FACTOR". Only the partner which created the process or the owning user can revoke the certificate created as part of activation process.

<b>Parameter</b>	<b>activationProcessId</b>
<b>Type</b>	<b>ObjectId</b> [see <a href="#">6.32</a> ]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The key of the activation process of which the certificate shall be revoked.

### Output of RevokeCertificateOfTokenActivationProcess

No output parameters defined for this operation.

### Exceptions of RevokeCertificateOfTokenActivationProcess

In case of an error in the operation, a `SignMeException` is thrown. Detailed explanations of all possible error codes are contained in section [9](#).

## 4.27 AuthorizeProcess

By means of this operation persons (logged in through sign-me or partner application) will authorize a specific process step in signature or token activation processes. If indicated by the return data of this operation, the call of AuthorizeProcessStep2 is needed. This is the case if the login session which is used for AuthorizeProcess does not have a sufficient authentication quality level.

E.g., if a person in USER role is only authenticated with INTIAL authentication quality level, then AuthorizeProcessStep2 is needed for authorization of a basic or advanced signature. This is because for basic or advanced signatures the authentication quality level ONE\_FACTOR is mandatory.

NOTE: AuthorizeProcess by organizations are not supported in this server version.

### Input of AuthorizeProcess

Parameter	Field Type	Occurrence
securityHeader	SecurityHeader	1
roleName	RoleName	1
authenticationTokenSelector	AuthenticationTokenSelector	1
processId	ObjectId	1

Table 4.18: Input Parameters of AuthorizeProcess

### Input Details of AuthorizeProcess

Parameter	securityHeader
Type	SecurityHeader [see 5.27]
Occurence	1
Meaning	The security header needs to be related to a valid login session with "USER" role with quality "INITIAL", or better "ONE_FACTOR". Only the person with username which is indicated in the process creation operation can authorize the process.

Parameter	roleName
Type	RoleName [see 6.39]
Occurence	1
Meaning	This identifies the role under which the client wants to authorize.

Parameter	authenticationTokenSelector
Type	AuthenticationTokenSelector [see 5.4]
Occurence	1
Meaning	This selects an authentication token of the person or organization who wants to authorize.

Parameter	<b>processId</b>
Type	<b>ObjectId</b> [see 6.32]
Occurrence	<b>1</b>
Meaning	This is the signature process or activation process ID, which shall be authorized.

### Output of AuthorizeProcess

Parameter	<b>accessTokenGrant</b>
Type	<b>AccessTokenGrant</b> [see 5.1]
Occurrence	<b>1</b>
Meaning	

### Exceptions of AuthorizeProcess

In case of an error in the operation, a `SignMeException` is thrown. Detailed explanations of all possible error codes are contained in section 9.

## 4.28 AuthorizeProcessStep2

`AuthorizeProcessStep2` is needed if the authentication quality level is not appropriate for the authorization, and a challenge or similar needs to be processed by the client to get to the new level. See also `AuthorizeProcess` operation (see section 4.27).

### Input of AuthorizeProcessStep2

Parameter	<b>securityHeader</b>
Type	<b>SecurityHeader</b> [see 5.27]
Occurrence	<b>1</b>
Meaning	The security header needs to be related to a valid login session, which was modified by the preceding authorize operation.

Parameter	<b>authenticationData</b>
Type	<b>GenericAuthenticationData</b> [see 6.18]
Occurrence	<b>0..10</b>
Meaning	If present, this element signals, that before granting a useful access token, token specific processing of these authentication data on the server side is required to provide a useful access token. This might be a password hash or one-time code or similar.

### Output of AuthorizeProcessStep2

Parameter	<b>accessTokenGrant</b>
Type	<b>AccessTokenGrant</b> [see 5.1]
Occurrence	<b>1</b>
Meaning	The <code>accessTokenGrant</code> to be used in the login session from this point.

## Exceptions of AuthorizeProcessStep2

In case of an error in the operation, a `SignMeException` is thrown. Detailed explanations of all possible error codes are contained in section 9.

## 4.29 Login

Login is used to get the access token grant by means of an authentication token and credentials. The login by partners is made without application provider security header. The Login operation results in an access token, which is not useful for business operations, because there are not enough access rights. Depending on the authentication token type used, there is a `LoginStep2` to complete the authentication on a useful authentication quality level. The exact processing of Login/LoginStep2 is dependent on the token type. See the parameter description and use cases for that. In general, for all supported authentication token types, there are convenience functions in the language specific access modules, and there will be for future token types.

The following feature needs contractual extensions and can only be used if activated by D-Trust, please contact [sales@d-trust.net](mailto:sales@d-trust.net).

The login by users (persons in "USER" role) is performed using the security header of the special partner (`PARTNER_EXIDENT`), which provides the used application. Thus, a user can not directly login into the system. He/She can only login through partners as their (web) application providers.

### Input of Login

Parameter	Field Type	Occurrence
<code>appProviderSecurityHeader</code>	<code>SecurityHeader</code>	0..1
<code>clientIpAddress</code>	<code>IpAddressNumeric</code>	1
<code>roleName</code>	<code>RoleName</code>	1
<code>authenticationTokenSelector</code>	<code>AuthenticationTokenSelector</code>	1
<code>requiredIfRevision</code>	<code>IfRevision</code>	0..1

Table 4.19: Input Parameters of Login

### Input Details of Login

Parameter	<code>appProviderSecurityHeader</code>
Type	<code>SecurityHeader</code> [see 5.27]
Occurence	0..1
Meaning	The security header of the provider of the application (logged-in in role "PARTNER_EXIDENT"), through which a user in "USER" role wants to login. This item is not present for a login as "PARTNER" itself.

Parameter	<code>clientIpAddress</code>
Type	<code>IpAddressNumeric</code> [see 6.26]
Occurence	1
Meaning	The IP-address of the client.

<b>Parameter</b>	<b>roleName</b>
<b>Type</b>	<b>RoleName</b> [see 6.39]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	This optional string identifies the role under which the client wants to login. If omitted, the default role is PARTNER.

<b>Parameter</b>	<b>authenticationTokenSelector</b>
<b>Type</b>	<b>AuthenticationTokenSelector</b> [see 5.4]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	This selects an authentication token of the person or organization who wants to login. In case of authenticationTokenType="UIDPW" or "UIDEMAIL", username is not empty and authenticationData is empty.

<b>Parameter</b>	<b>requiredIfRevision</b>
<b>Type</b>	<b>IfRevision</b> [see 6.25]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	This optional string identifies the required interface revision for a partner session. The server will comply to the revision required. The revision must be higher than 6955, otherwise it will default to 6955. This mechanism will only work within minor version interface updates.

## Output of Login

<b>Parameter</b>	<b>accessTokenGrant</b>
<b>Type</b>	<b>AccessTokenGrant</b> [see 5.1]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	In case of authenticationTokenType="UIDPW", accessTokenGrant.authenticationData contains a salt (special byte array value) as the only element. The salt is to be used for login step2.

## Exceptions of Login

In case of an error in the operation, a SignMeException is thrown. Detailed explanations of all possible error codes are contained in section 9.

## 4.30 LoginStep2

LoginStep2 is needed to complete the authentication. The exact processing of LoginStep2 is dependent on the token type. See the parameter description and use cases for that. In general, for all supported authentication token types, there are convenience functions in the language specific access modules.

## Input of LoginStep2

Parameter	<b>securityHeader</b>
Type	<b>SecurityHeader</b> [see 5.27]
Occurrence	<b>1</b>
Meaning	The security header needs to be related to a valid login session, which was initiated by the preceding Login operation.

Parameter	<b>authenticationData</b>
Type	<b>GenericAuthenticationData</b> [see 6.18]
Occurrence	<b>0..10</b>
Meaning	If present, this element signals, that before granting a useful access token, token specific processing of these authentication data on the server side is required to provide a useful access token. This might be a password hash or one-time code or similar. In case of authenticationTokenType="UIDPW", authenticationData contains the salted and hashed password as the only element. See GenericAuthData. In case of "UIDEMAIL" step 2 is not needed.

## Output of LoginStep2

Parameter	<b>accessTokenGrant</b>
Type	<b>AccessTokenGrant</b> [see 5.1]
Occurrence	<b>1</b>
Meaning	The accessTokenGrant to be used in the login session from this point.

## Exceptions of LoginStep2

In case of an error in the operation, a SignMeException is thrown. Detailed explanations of all possible error codes are contained in section 9.

## 4.31 RefreshLogin

RefreshLogin allows to refresh an authentication token. It should only be used if needed, to reduce load on the back-end. See the use-case description. It is recommended to use the convenience function in the language specific access modules, which can decide locally, if a refresh is needed on the server side.

## Input of RefreshLogin

Parameter	<b>securityHeader</b>
Type	<b>SecurityHeader</b> [see 5.27]
Occurrence	<b>1</b>
Meaning	The security header needs to be related to a valid login session.



## Output of RefreshLogin

Parameter	accessTokenGrant
Type	AccessTokenGrant [see 5.1]
Occurrence	1
Meaning	Returns a new access token grant with new lifetime.

## Exceptions of RefreshLogin

In case of an error in the operation, a SignMeException is thrown. Detailed explanations of all possible error codes are contained in section 9.

## 4.32 Logout

Logout terminates a Login session connected with the access token. Especially for Login sessions in "USER" role it is good practice to keep the session as short as possible, and logout after use, instead of letting the session run into timeout.

### Input of Logout

Parameter	securityHeader
Type	SecurityHeader [see 5.27]
Occurrence	1
Meaning	The security header should be related to a valid login session.

### Output of Logout

No output parameters defined for this operation.

### Exceptions of Logout

In case of an error in the operation, a SignMeException is thrown. Detailed explanations of all possible error codes are contained in section 9.

## 4.33 GetVersion

GetVersion returns the interface version (major.minor) of the service.

### Input of GetVersion

Parameter	securityHeader
Type	SecurityHeader [see 5.27]
Occurrence	1
Meaning	The security header needs to be related to a valid login session with "PARTNER" role with quality "ONE_FACTOR" or "USER" role.

## Output of GetVersion

Parameter	versionAndDate
Type	VersionAndDate [see <a href="#">5.35</a> ]
Occurence	1
Meaning	Returns version and date of the interface.

## Exceptions of GetVersion

In case of an error in the operation, a SignMeException is thrown. Detailed explanations of all possible error codes are contained in section [9](#).

## 5 Structured Datatypes of the API

### 5.1 AccessTokenGrant

This type specifies the information returned by Login[Step2]/Authorize[Step2]/Refresh operations to provide the result of the authentication or specific authorization.

#### Summary of AccessTokenGrant

Field Name	Field Type	Occurrence
accessToken	AccessToken	1
reachedAuthenticationQualityLevel	AuthenticationQualityLevel	1
needsStep2	boolean	1
tokenGrantedAtUTCMillsec	long	1
validityDurationMillsec	long	1
authenticationData	GenericAuthenticationData	0..10

Table 5.1: Fields of type AccessTokenGrant

#### Details of AccessTokenGrant

Field	accessToken
Type	AccessToken [see 6.1]
Occurrence	1
Meaning	<p>The accessToken which grants access to further service requests for a defined time or until logout. The access level is restricted by token type, user role and resource ownership. It may also be restricted by other factors like network location etc. However, the access token is completely ignorant of these factors and is indistinguishable from a random byte sequence.</p> <p>The access token identifies a login session on the sign-me system which is associated to a single identity (see 5.21 or 5.20).</p>

Field	reachedAuthenticationQualityLevel
Type	AuthenticationQualityLevel [see 6.4]
Occurrence	1
Meaning	<p>The quality level of the authentication reached for the access token granted.</p>

<b>Field</b>	<b>needsStep2</b>
<b>Type</b>	<b>boolean</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	Needs a authentication step 2 to reach the final authentication quality level possible with the authentication token.

<b>Field</b>	<b>tokenGrantedAtUTCMillsec</b>
<b>Type</b>	<b>long</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The time-stamp in milliseconds since UTC 01/01/1970 when the token was granted by the login service.

<b>Field</b>	<b>validityDurationMillsec</b>
<b>Type</b>	<b>long</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The duration of the validity of the access token in milliseconds.

<b>Field</b>	<b>authenticationData</b>
<b>Type</b>	<b>GenericAuthenticationData</b> [see <a href="#">6.18</a> ]
<b>Occurrence</b>	<b>0..10</b>
<b>Meaning</b>	If present, this element signals, that before granting a useful access token (other than reachedAuthenticationQualityLevel=INITIAL), token specific processing of these authentication data on the client side is required to provide a useful access token. This might be a challenge needed to be answered by the client or similar.

## 5.2 AdditionalData

Additional data for specific use cases.

### Summary of AdditionalData

Field Name	Field Type	Occurrence
nameStringValue	NameStringValue	<b>0..16</b>
nameBlobValues	NameBlobValue	<b>0..16</b>

Table 5.2: Fields of type AdditionalData

### Details of AdditionalData

<b>Field</b>	<b>nameStringValue</b>
<b>Type</b>	<b>NameStringValue</b> [see <a href="#">5.19</a> ]
<b>Occurrence</b>	<b>0..16</b>
<b>Meaning</b>	An optional list of name/string value pairs, which are used for specific use cases.

<b>Field</b>	<b>nameBlobValues</b>
<b>Type</b>	<b>NameBlobValue</b> [see 5.18]
<b>Occurrence</b>	<b>0..16</b>
<b>Meaning</b>	An optional list of name/binary value pairs, which are used for specific use cases.

## 5.3 Address

This specifies typical address information to be found on ID cards.

### Summary of Address

Field Name	Field Type	Occurrence
street	StreetAddress	0..1
city	UTF8String180	1
state	UTF8String180	0..1
country	ICAOCountryName	1
zipCode	ZipCode	0..1

Table 5.3: Fields of type Address

### Details of Address

<b>Field</b>	<b>street</b>
<b>Type</b>	<b>StreetAddress</b> [see 6.46]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Street name and number.

<b>Field</b>	<b>city</b>
<b>Type</b>	<b>UTF8String180</b> [see 6.53]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	Name of city/town/village etc.

<b>Field</b>	<b>state</b>
<b>Type</b>	<b>UTF8String180</b> [see 6.53]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Name of state if known and applicable.

<b>Field</b>	<b>country</b>
<b>Type</b>	<b>ICAOCountryName</b> [see 6.20]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	ICAO country code of address country (e.g., for "Deutschland" -> D). See section 8 for a table of all possible values.

<b>Field</b>	<b>zipCode</b>
<b>Type</b>	<b>ZipCode</b> [see 6.64]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Zip/Postal code of the address.

## 5.4 AuthenticationTokenSelector

AuthenticationTokenSelector type documentation

### Summary of AuthenticationTokenSelector

Field Name	Field Type	Occurrence
authenticationTokenType	AuthenticationTokenType	1
username	Username	0..1
identityId	ObjectId	0..1
authenticationData	GenericAuthenticationData	0..10

Table 5.4: Fields of type AuthenticationTokenSelector

### Details of AuthenticationTokenSelector

<b>Field</b>	<b>authenticationTokenType</b>
<b>Type</b>	<b>AuthenticationTokenType</b> [see 6.5]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	This is the token type which shall be used for authentication of the person or organization.

<b>Field</b>	<b>username</b>
<b>Type</b>	<b>Username</b> [see 6.63]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	This is a user ID of the person or organization which wants to login. This is to be used in case authenticationTokenType="UIDPW" or "UIDE-MAIL" or "SIDTK".

<b>Field</b>	<b>identityId</b>
<b>Type</b>	<b>ObjectId</b> [see 6.32]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	This is an identity ID of the person or organization which wants to login. This is to be used for restricted clients in case authenticationTokenType="IID".

<b>Field</b>	<b>authenticationData</b>
<b>Type</b>	<b>GenericAuthenticationData</b> [see 6.18]
<b>Occurrence</b>	<b>0..10</b>
<b>Meaning</b>	Depending on the authentication token type specific processing of these authentication data on the server side might be required to provide a useful access token (e.g. password hash or one-time code or similar).

## 5.5 Certificate

The subject certificate of a Person which certifies the public key of a signature token, or a Sub-CA or Root-CA certification used during activation of signature tokens. Some important attributes of the certificate are explicitly available in this structure. Moreover, the certificate is also available in binary form.

### Summary of Certificate

Field Name	Field Type	Occurrence
certificateId	ObjectId	1
subjectDN	UTF8String200	1
issuerDN	UTF8String200	1
serialNumber	UTF8String80	1
notValidBefore	dateTime	1
notValidAfter	dateTime	1
issuerCertificateId	ObjectId	0..1
certificate	MediumBlob	1
ocspResponse	MediumBlob	0..1
ocspResponseProducedAt	dateTime	0..1
x509CRL	LargeBlob	0..1

Table 5.5: Fields of type Certificate

### Details of Certificate

Field	certificateId
Type	ObjectId [see 6.32]
Occurence	1
Meaning	The sign-me object id of a certificate used to get this descriptor over the sign-me API.

Field	subjectDN
Type	UTF8String200 [see 6.55]
Occurence	1
Meaning	The subject DN of the certificate.

Field	issuerDN
Type	UTF8String200 [see 6.55]
Occurence	1
Meaning	The issuer DN of the certificate.

Field	serialNumber
Type	UTF8String80 [see 6.62]
Occurence	1
Meaning	The serial number of the certificate.

<b>Field</b>	<b>notValidBefore</b>
<b>Type</b>	<b>dateTime</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The notValidBefore of the certificate.

<b>Field</b>	<b>notValidAfter</b>
<b>Type</b>	<b>dateTime</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The notValidAfter of the certificate.

<b>Field</b>	<b>issuerCertificateId</b>
<b>Type</b>	<b>ObjectId</b> [see <a href="#">6.32</a> ]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The sign-me object id of the issuer certificate if known in the system.

<b>Field</b>	<b>certificate</b>
<b>Type</b>	<b>MediumBlob</b> [see <a href="#">6.31</a> ]]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The certificate in X.509 ASN.1 representation (binary encoded).

<b>Field</b>	<b>ocspResponse</b>
<b>Type</b>	<b>MediumBlob</b> [see <a href="#">6.31</a> ]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	<p>The OCSP response in X.509 ASN.1 representation (binary encoded). Only available, if any LT-signature types were successfully used signing with this certificate and its chain and results were retrieved. The OCSP response is only available for subject certificates or sign-me users. For other certificates CRLs are provided at the issuer certificate.</p> <p>NOTE: The retrieval could not be possible due to unavailability of the respective OCSP. In this case a retry of GetSignatureProcess to get all results of a successful signature is needed.</p>

<b>Field</b>	<b>ocspResponseProducedAt</b>
<b>Type</b>	<b>dateTime</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	<p>The producedAt time-stamp of the OCSP response. Only available, if any LT-signature types were successfully used signing with this certificate and its chain and results were retrieved.</p> <p>NOTE: see note for ocspResponse element.</p>



<b>Field</b>	<b>x509CRL</b>
<b>Type</b>	<b>LargeBlob</b> [see 6.28]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	<p>If this is an issuer certificate, the CRL for the certificates of this issuer in X.509 ASN.1 representation (binary encoded). Only available, if any LT-signature types were successfully used signing with this certificate and its chain and results were retrieved.</p> <p>NOTE: The retrieval could not be possible due to unavailability of the respective CRL end-point. In this case a retry of GetSignatureProcess to get all results of a successful signature is needed.</p>

## 5.6 CertificateTemplate

Specifies the properties of certificates of a type indicated by 'certificateType'.

### Summary of CertificateTemplate

Field Name	Field Type	Occurrence
certificateTemplateId	ObjectId	1
certificateType	CertificateType	1
activeForReservation	boolean	0..1
authorizedOrganizations	UTF8String4096	0..1
reservationDuration	Duration	0..1
usableDuration	Duration	0..1
retrievableDuration	Duration	0..1
archivingDuration	Duration	0..1
contentType	UTF8String256	0..1
includeRequestorEmailAddress	boolean	0..1
subjectOrganization	UTF8String80	0..1
subjectOrganizationalUnit	UTF8String80	0..1
extensionRestriction	UTF8String1024	0..1
extensionAdditionalInforma.. ..tion	UTF8String1024	0..1
extensionMonetaryLimitAmount	double	0..1
extensionMonetaryLimitExpo.. ..nent	double	0..1
extensionMonetaryLimitCurr.. ..ency	Currency	0..1
cmSDKCertificationProcedure	UTF8String80	0..1
cmSDKPublishingProcedure	UTF8String80	0..1
cmSDKPublishingProcedureWi.. ..thLDAP	UTF8String80	0..1
keyAlgo	UTF8String20	0..1
rsaKeySize	integer	0..1
ecdsaNamedCurve	UTF8String20	0..1
tokenCapability	TokenCapability	0..1

Table 5.6: Fields of type CertificateTemplate

## Details of CertificateTemplate

<b>Field</b>	<b>certificateTemplateId</b>
<b>Type</b>	<b>ObjectId</b> [see 6.32]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	

<b>Field</b>	<b>certificateType</b>
<b>Type</b>	<b>CertificateType</b> [see 6.7]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	

<b>Field</b>	<b>activeForReservation</b>
<b>Type</b>	<b>boolean</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Template can only be used for reservations.

<b>Field</b>	<b>authorizedOrganizations</b>
<b>Type</b>	<b>UTF8String4096</b> [see 6.60]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Authorized organizations

<b>Field</b>	<b>reservationDuration</b>
<b>Type</b>	<b>Duration</b> [see 6.13]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Validity period of the reservation.

<b>Field</b>	<b>usableDuration</b>
<b>Type</b>	<b>Duration</b> [see 6.13]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Validity period of the certificate.

<b>Field</b>	<b>retrievableDuration</b>
<b>Type</b>	<b>Duration</b> [see 6.13]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The duration within the certificate process following this template is retrievable from the system.

<b>Field</b>	<b>archivingDuration</b>
<b>Type</b>	<b>Duration</b> [see 6.13]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The duration within the certificate process following this template will be archived.

<b>Field</b>	<b>contentType</b>
<b>Type</b>	<b>UTF8String256</b> [see 6.56]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	ContentType of the certificate. It determines the mapping in a container class in the archive system (Saperion). The container class rules over the deletion and retention period.
<b>Field</b>	<b>includeRequestorEmailAddress</b>
<b>Type</b>	<b>boolean</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Certificate must include email if true.
<b>Field</b>	<b>subjectOrganization</b>
<b>Type</b>	<b>UTF8String80</b> [see 6.62]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Organization of the certificate owner
<b>Field</b>	<b>subjectOrganizationalUnit</b>
<b>Type</b>	<b>UTF8String80</b> [see 6.62]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Organization unit of the certificate owner
<b>Field</b>	<b>extensionRestriction</b>
<b>Type</b>	<b>UTF8String1024</b> [see 6.51]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Certificate usage restrictions
<b>Field</b>	<b>extensionAdditionalInformation</b>
<b>Type</b>	<b>UTF8String1024</b> [see 6.51]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Certificate additional information
<b>Field</b>	<b>extensionMonetaryLimitAmount</b>
<b>Type</b>	<b>double</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Amount of the monetary limit according to PKIX. value := extensionMonetaryLimitAmount * (10**extensionMonetaryLimitExponent)
<b>Field</b>	<b>extensionMonetaryLimitExponent</b>
<b>Type</b>	<b>double</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Exponent of the monetary limit according to PKIX.

<b>Field</b>	<b>extensionMonetaryLimitCurrency</b>
<b>Type</b>	<b>Currency</b> [see 6.10]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Currency of the monetary limit according to PKIX.

<b>Field</b>	<b>cmSDKCertificationProcedure</b>
<b>Type</b>	<b>UTF8String80</b> [see 6.62]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Names of the token procedures used for certification.

<b>Field</b>	<b>cmSDKPublishingProcedure</b>
<b>Type</b>	<b>UTF8String80</b> [see 6.62]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Names of the publishing procedures.

<b>Field</b>	<b>cmSDKPublishingProcedureWithLDAP</b>
<b>Type</b>	<b>UTF8String80</b> [see 6.62]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Names of the publishing procedures for LDAP.

<b>Field</b>	<b>keyAlgo</b>
<b>Type</b>	<b>UTF8String20</b> [see 6.54]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Key algorithm

<b>Field</b>	<b>rsaKeySize</b>
<b>Type</b>	<b>integer</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	RSA key size

<b>Field</b>	<b>ecdsaNamedCurve</b>
<b>Type</b>	<b>UTF8String20</b> [see 6.54]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	ECDSA named curve

<b>Field</b>	<b>tokenCapability</b>
<b>Type</b>	<b>TokenCapability</b> [see 6.50]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Token capability required for this certificate type.

## 5.7 ErrorInfo

This is the information which is made available as an exception on the client side in case of any errors in the sign-me service for a specific operation.

## Summary of ErrorInfo

Field Name	Field Type	Occurrence
date	dateTime	1
errorId	ErrorId	1
errorMessage	UTF8String512	1
detailedErrorId	UTF8String32	1

Table 5.7: Fields of type ErrorInfo

## Details of ErrorInfo

Field	date
Type	dateTime [see <a href="#">XSD</a> ]
Occurrence	1
Meaning	Date and time of the occurrence of the error.

Field	errorId
Type	ErrorId [see <a href="#">6.15</a> ]
Occurrence	1
Meaning	A general code for the type of error happened.

Field	errorMessage
Type	UTF8String512 [see <a href="#">6.61</a> ]
Occurrence	1
Meaning	Contains a written description of the error, and optionally a handling description. IMPORTANT REMARK: The element errorMessage shall be transparently forwarded into logs or user interfaces. Do not depend on values of this element, because it may change without further notice.

Field	detailedErrorId
Type	UTF8String32 [see <a href="#">6.57</a> ]
Occurrence	1
Meaning	This is a unique id of the error and is described in the error list out of this specification (see <a href="#">9</a> ). Any nationalized error strings should be based on this code.

## 5.8 GenericDate

GenericDate type documentation

## Summary of GenericDate

Field Name	Field Type	Occurrence
dateString	DateString	1
dateValue	date	0..1

Table 5.8: Fields of type GenericDate

## Details of GenericDate

Field	dateString
Type	DateString [see <a href="#">6.12</a> ]
Occurrence	1
Meaning	

Field	dateValue
Type	date [see <a href="#">XSD</a> ]
Occurrence	0..1
Meaning	

## 5.9 IdentificationTemplate

Specifies the properties of identifications of a type indicated by 'identType'.

## Summary of IdentificationTemplate

Field Name	Field Type	Occurrence
identificationTemplateId	ObjectId	1
identType	IdentType	1
activeForReservation	boolean	0..1
identProviders	UTF8String128	0..1
communicationWasConfirmed	boolean	0..1
allowLoginPreset	boolean	0..1
pwChangeMandatory	boolean	0..1
organizationRechargeCoinAm.. ..ount	UTF8String4096	0..1
organizationChargeCoinAmount	UTF8String4096	0..1
authorizedOrganizations	UTF8String4096	0..1
usableDuration	Duration	0..1
retrievableDuration	Duration	0..1
archivingDuration	Duration	0..1
videoRetrievalRequired	boolean	0..1
sanctionListCheckRequired	boolean	0..1
fixnetSmsAllowed	boolean	0..1
onDemandTokenActivationReq.. ..uiredForQualified	boolean	0..1
sapMaterialNumber	UTF8String20	0..1
legalIdentificationValidit.. ..yPeriod	Duration	0..1
useOfDocument Validity Allowed	boolean	0..1
directIdentityDataTakover	boolean	0..1
directIdentityDataTakoverM.. ..ethod	UTF8String32	0..1
pwSenderName	UTF8String40	0..1
pwMessageText	UTF8String200	0..1
itmRefWithPersonId	boolean	0..1
itmGetTargetLink	boolean	0..1
itmGetVideoHashes	boolean	0..1
itmProductId	integer	0..1
itmCustomerId	integer	0..1
itmAdditionalValue	integer	0..1
itmIsPointOfSale	boolean	0..1

Table 5.9: Fields of type IdentificationTemplate

## Details of IdentificationTemplate

Field	identificationTemplateId
Type	ObjectId [see 6.32]
Occurence	1
Meaning	

<b>Field</b>	<b>identType</b>
<b>Type</b>	<b>IdentType</b> [see 6.21]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	

<b>Field</b>	<b>activeForReservation</b>
<b>Type</b>	<b>boolean</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Template can only be used for reservations.

<b>Field</b>	<b>identProviders</b>
<b>Type</b>	<b>UTF8String128</b> [see 6.52]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	List of ProviderIDs of equivalent identity verification service Provider that can be chosen by the the IdentityVerifierService

<b>Field</b>	<b>communicationWasConfirmed</b>
<b>Type</b>	<b>boolean</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	User email address needs to be verified and confirmed by the partner if true.

<b>Field</b>	<b>allowLoginPreset</b>
<b>Type</b>	<b>boolean</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Login data can be preset on start if true.

<b>Field</b>	<b>pwChangeMandatory</b>
<b>Type</b>	<b>boolean</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Password change will be mandatory after login if true.

<b>Field</b>	<b>organizationRechargeCoinAmount</b>
<b>Type</b>	<b>UTF8String4096</b> [see 6.60]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	List of organizations and the coin amount to be topped up to the user after successful identification.

<b>Field</b>	<b>organizationChargeCoinAmount</b>
<b>Type</b>	<b>UTF8String4096</b> [see 6.60]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	List of organizations and the coin amount to be charged following a successful identification.



<b>Field</b>	<b>authorizedOrganizations</b>
<b>Type</b>	<b>UTF8String4096</b> [see 6.60]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	authorized organizations

<b>Field</b>	<b>usableDuration</b>
<b>Type</b>	<b>Duration</b> [see 6.13]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Period during which the identification can take place.

<b>Field</b>	<b>retrievableDuration</b>
<b>Type</b>	<b>Duration</b> [see 6.13]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The duration within the identification processes following this template is retrievable from the system.

<b>Field</b>	<b>archivingDuration</b>
<b>Type</b>	<b>Duration</b> [see 6.13]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The duration within the identification process following this template will be archived.

<b>Field</b>	<b>videoRetrievalRequired</b>
<b>Type</b>	<b>boolean</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Video protocol of the successfully completed identification must be retrieved if true.

<b>Field</b>	<b>sanctionListCheckRequired</b>
<b>Type</b>	<b>boolean</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Sanction-list check required before identification can successfully completed if true

<b>Field</b>	<b>fixnetSmsAllowed</b>
<b>Type</b>	<b>boolean</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Usage of SMS on landline (as opposed to mobile) allowed if true

<b>Field</b>	<b>onDemandTokenActivationRequiredForQualified</b>
<b>Type</b>	<b>boolean</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	On demand token activation required for qualified if true.

<b>Field</b>	<b>sapMaterialNumber</b>
<b>Type</b>	<b>UTF8String20</b> [see 6.54]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	SAP material number

<b>Field</b>	<b>legalIdentificationValidityPeriod</b>
<b>Type</b>	<b>Duration</b> [see 6.13]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The duration the current successfully completed identification will remain valid.

<b>Field</b>	<b>useOfDocumentValidityAllowed</b>
<b>Type</b>	<b>boolean</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Validity period of the identification can be taken from the identity document used

<b>Field</b>	<b>directIdentityDataTakover</b>
<b>Type</b>	<b>boolean</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Identification data can be taken directly from the organization if true.

<b>Field</b>	<b>directIdentityDataTakoverMethod</b>
<b>Type</b>	<b>UTF8String32</b> [see 6.57]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Method used for identity takeover if directIdentityDataTakover true

<b>Field</b>	<b>pwSenderName</b>
<b>Type</b>	<b>UTF8String40</b> [see 6.58]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	password sender name

<b>Field</b>	<b>pwMessageText</b>
<b>Type</b>	<b>UTF8String200</b> [see 6.55]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Text of the password message

<b>Field</b>	<b>itmRefWithPersonId</b>
<b>Type</b>	<b>boolean</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	identity.tm / id.now specific value Verification process link includes PersonID if true verificationProcess.getID().getId() + ":" + personToBeVerified.getID().getId()

<b>Field</b>	<b>itmGetTargetLink</b>
<b>Type</b>	<b>boolean</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	identity.tm / id.now specific value targetLink shall be returned by this ident provider if true

<b>Field</b>	<b>itmGetVideoHashes</b>
<b>Type</b>	<b>boolean</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	identity.tm / id.now specific value videoHashes shall be returned by this ident provider if true

<b>Field</b>	<b>itmProductId</b>
<b>Type</b>	<b>integer</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	identity.tm / id.now specific value SHOP_IDENT = 8; VIDEO_IDENT = 12; E_ID = 13; GIRO_IDENT = 14; ESIGN = 15;

<b>Field</b>	<b>itmCustomerId</b>
<b>Type</b>	<b>integer</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	identity.tm / id.now specific value selects provider environment to send the requests to: development, staging, production

<b>Field</b>	<b>itmAdditionalValue</b>
<b>Type</b>	<b>integer</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	identity.tm / id.now specific value NO_ACQUISITION_BY_PHONE = 32; RECORD_VIDEO = 131072; POS_ADD_VALUE = 4194304;

<b>Field</b>	<b>itmIsPointOfSale</b>
<b>Type</b>	<b>boolean</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	identity.tm / id.now specific value identification type is POS (SHOP_IDENT) if true

## 5.10 Identity

Identity Attributes of natural persons or organizations, which are mainly derived from legal documents. This data is enriched by other ID information and supporting data. Depending on the use case and access rights only parts of the elements will be supplied.

## Summary of Identity

Field Name	Field Type	Occurrence
identityId	ObjectId	0..1
allowedRoleNames	RoleAssociation	0..10
username	Username	0..1
identityVerificationState	IdentityVerificationState	0..1
identityAdministrativeState	AdministrativeState	0..1
createdTimestamp	dateTime	0..1
modifiedTimestamp	dateTime	0..1
passwd	Password	0..1
emailAddress	EmailAddress	0..1
emailAddressVerified	boolean	0..1
allowExtendedEmailAddressUse	boolean	0..1
publishRequest	boolean	0..1
revocationStringHash	TinyBlob	0..1
indoctrination	boolean	0..1
coins	Coins	0..1
subscription	Subscription	0..1
preferredLanguage	BCP47LanguageTag	0..1
additionalData	AdditionalData	0..1

Table 5.10: Fields of type Identity

## Details of Identity

<b>Field</b>	<b>identityId</b>
<b>Type</b>	<b>ObjectId</b> [see 6.32]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The object ID of the identity. Only mirrored in responses on GetIdentity if the request was made using identityId. Otherwise there is only a shortened identityId, which is enough for reference, but not valid for getting all identity attributes by means of getIdentity(identityId).

<b>Field</b>	<b>allowedRoleNames</b>
<b>Type</b>	<b>RoleAssociation</b> [see 5.26]
<b>Occurrence</b>	<b>0..10</b>
<b>Meaning</b>	The role names which the identity can use to login including any applicable restrictions. Only present in responses on GetIdentity if requested and only if security rules allow.

<b>Field</b>	<b>username</b>
<b>Type</b>	<b>Username</b> [see 6.63]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The username of the identity which will be used for further logins after identifications. Optional field for use in RegisterIdentity/Create-/UpdateRegistrationProcess.

<b>Field</b>	<b>identityVerificationState</b>
<b>Type</b>	<b>IdentityVerificationState</b> [see 6.23]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The verification state of the identity. Based on this state signing or token activation operations are possible for different types of certificates (see section 6.7) and signatures (see section 6.43). See type description for IdentityVerificationState in section 6.23. Only present in responses on GetIdentity.

<b>Field</b>	<b>identityAdministrativeState</b>
<b>Type</b>	<b>AdministrativeState</b> [see 6.3]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If the identity is locked the identity itself has no access any more with any of its authentication tokens. Only present in responses on GetIdentity.

<b>Field</b>	<b>createdTimestamp</b>
<b>Type</b>	<b>dateTime</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The date/time when the identity was created in sign-me. Only present in responses on GetIdentity if requested.

<b>Field</b>	<b>modifiedTimestamp</b>
<b>Type</b>	<b>dateTime</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The date/time when the identity was last modified in sign-me. This might include internal modifications not visible over the interface. Only present in responses on GetIdentity if requested.

<b>Field</b>	<b>passwd</b>
<b>Type</b>	<b>Password</b> [see 6.34]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If present, the password for the user ID which will be used through the login service. The password will of course never be available in responses, but only used for setting purposes in an initial state of the identification process. Optional field for use in RegisterIdentity/Create-/UpdateRegistrationProcess.

<b>Field</b>	<b>emailAddress</b>
<b>Type</b>	<b>EmailAddress</b> [see 6.14]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Valid contact email address of the identity. Mandatory field for use in RegisterIdentity/Create- /UpdateRegistrationProcess.

<b>Field</b>	<b>emailAddressVerified</b>
<b>Type</b>	<b>boolean</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	This indicates whether the email address was verified by the system. Only present in responses on GetIdentity if requested.

<b>Field</b>	<b>allowExtendedEmailAddressUse</b>
<b>Type</b>	<b>boolean</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Deprecated.

<b>Field</b>	<b>publishRequest</b>
<b>Type</b>	<b>boolean</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	This indicates the request of the identity that certificates shall be published on an LDAP directory service. Optional field for use in RegisterIdentity/Create- /UpdateRegistrationProcess.

<b>Field</b>	<b>revocationStringHash</b>
<b>Type</b>	<b>TinyBlob</b> [see 6.48]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Deprecated field. Must not be present any more.

<b>Field</b>	<b>indoctrination</b>
<b>Type</b>	<b>boolean</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	This is the summary expression whether all informations were read and all agreements were accepted. Mandatory field for use in RegisterIdentity/UpdateRegistrationProcess(complete=true). Optional in CreateRegistrationProcess and UpdateRegistrationProcess(complete=false).

<b>Field</b>	<b>coins</b>
<b>Type</b>	<b>Coins</b> [see 6.8]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The number of coins available for consumption by means of signatures. The signatures, which consume coins are of special pre-paid signature types. The coin consumption can be checked at the signature template, which defines the signature type. Only present in responses on GetIdentity if requested.

<b>Field</b>	<b>subscription</b>
<b>Type</b>	<b>Subscription</b> [see 5.32]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The subscription, if present. If not present, there is no active subscription at all.

<b>Field</b>	<b>preferredLanguage</b>
<b>Type</b>	<b>BCP47LanguageTag</b> [see 6.6]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The language preferred in communications by the identity. Specified according to IETF BCP 47 standard mainly as language code (ISO639) and region (2 character country code acc. to ISO3166-1). Samples: "de-DE" or "de" "en-US" or "en" or "fr-FR" or "fr". If the specific language or language tag is not supported, the default language will be assigned. Optional field for use in RegisterIdentity/Create-/UpdateRegistrationProcess.

<b>Field</b>	<b>additionalData</b>
<b>Type</b>	<b>AdditionalData</b> [see 5.2]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Additional data for specific use cases. Optional field for use in RegisterIdentity/Create-/UpdateRegistrationProcess/CreateIdentityVerificationProcess. Restricted for specific use cases.

## 5.11 IdentitySelector

Allows to select an identity based on the unique username or the identityId

### Alternative Fields of IdentitySelector

Field Name (any of)	Field Type	Occurrence
identityId	ObjectId	1
username	Username	1

Table 5.11: Fields of type IdentitySelector

## Details of IdentitySelector

<b>Field</b>	<b>identityId</b>
<b>Type</b>	<b>ObjectId</b> [see <a href="#">6.32</a> ]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The object id of the identity received by calls to the sign-me system (e.g. getSignatureProcess).

<b>Field</b>	<b>username</b>
<b>Type</b>	<b>Username</b> [see <a href="#">6.63</a> ]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The username of registered users the sign-me system.

## 5.12 IdentityVerificationProcess

The identity verification process is used to handle the progress of identification verification requests by the partner, identification session of the user, and retrieval of the results.



## Summary of IdentityVerificationProcess

Field Name	Field Type	Occurrence
identityVerificationProces.. ..sId	ObjectId	1
identityVerificationProces.. ..sURL	IdentityVerificationProcessURL	1
organizationId	ObjectId	1
identityId	ObjectId	1
redirectURLs	RedirectURLs	0..1
externalProcessKey	ExternalProcessKey	0..1
externalProcessLink	ExternalProcessURL	0..1
createdTimestamp	dateTime	0..1
startedTimestamp	dateTime	0..1
identificationCompletedTim.. ..estamp	dateTime	0..1
chargingOrBillingCompleted.. ..Timestamp	dateTime	0..1
defaultTokensCreatedTimest.. ..amp	dateTime	0..1
identityVerificationProces.. ..sUsableEndTimestamp	dateTime	1
identityVerificationProces.. ..sRetrievableEndTimestamp	dateTime	1
identityVerificationType	IdentityVerificationType	0..1
communicationWasConfirmed	boolean	0..1
allowLoginPreset	boolean	0..1
pwChangeMandatory	boolean	0..1
state	IdentityVerificationState	1
statusText	UTF8String80	1
failureReason	UTF8String80	0..1
rechargedCoinAmount	Coins	0..1
sanctionListProtId	long	0..1
providerId	UTF8String32	0..1
providerName	UTF8String80	0..1
providerWebsiteURL	anyURI	0..1

Table 5.12: Fields of type IdentityVerificationProcess

## Details of IdentityVerificationProcess

Field	identityVerificationProcessId
Type	ObjectId [see 6.32]
Occurence	1
Meaning	The object id of the identity verification process received by calls to the sign-me system.

<b>Field</b>	<b>identityVerificationProcessURL</b>
<b>Type</b>	<b>IdentityVerificationProcessURL</b> [see 6.22]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The unique URL where the identity verification WEB application is reachable for the specific verification process. Partners redirect users (Persons in role USER) to this URL, where the users can complete the identity verification.

<b>Field</b>	<b>organizationId</b>
<b>Type</b>	<b>ObjectId</b> [see 6.32]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The id of the organization which created this identification verification process.

<b>Field</b>	<b>identityId</b>
<b>Type</b>	<b>ObjectId</b> [see 6.32]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The id of the identity for which the identification process was created. The full identityId is only returned after the identification is successful (state must be LAWFULLY_VERIFIED_HIGH). Otherwise there is only a shortened identityId, which is enough for reference, but not valid for getting all identity attributes by means of getIdentity(identityId). Therefore, the full set of attributes of an identity can be retrieved using getIdentity(identityId) after a successful identification with this identity verification process.

<b>Field</b>	<b>redirectURLs</b>
<b>Type</b>	<b>RedirectURLs</b> [see 5.24]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	This can contain URLs to redirect to in case of "OK"-, "FAIL"-, and "CANCEL"-results.

<b>Field</b>	<b>externalProcessKey</b>
<b>Type</b>	<b>ExternalProcessKey</b> [see 6.16]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If present, identifies the external process.

<b>Field</b>	<b>externalProcessLink</b>
<b>Type</b>	<b>ExternalProcessURL</b> [see 6.17]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If present, allows to access the external process via this link.

<b>Field</b>	<b>createdTimestamp</b>
<b>Type</b>	<b>dateTime</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The date/time when the partner created the identity verification process (in state <code>LAWFUL_IDENTIFICATION_REQUESTED</code> or already <code>LAWFULLY_VERIFIED_*</code> ).

<b>Field</b>	<b>startedTimestamp</b>
<b>Type</b>	<b>dateTime</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The date/time when the person started the identification (in state <code>LAWFUL_IDENTIFICATION_STARTED</code> ).

<b>Field</b>	<b>identificationCompletedTimestamp</b>
<b>Type</b>	<b>dateTime</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The date/time when the person completed the identification session (in state <code>LAWFULLY_VERIFIED_HIGH</code> ).

<b>Field</b>	<b>chargingOrBillingCompletedTimestamp</b>
<b>Type</b>	<b>dateTime</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The date/time when the person completed the identification session and all follow-up tasks could be completed within the system (except default token generation). (in state <code>LAWFULLY_VERIFIED_HIGH</code> ).

<b>Field</b>	<b>defaultTokensCreatedTimestamp</b>
<b>Type</b>	<b>dateTime</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The date/time of default token generation. (in state <code>LAWFULLY_VERIFIED_HIGH</code> ).

<b>Field</b>	<b>identityVerificationProcessUsableEndTimestamp</b>
<b>Type</b>	<b>dateTime</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The date/time until the verification process is valid for use and the verification can be made.

<b>Field</b>	<b>identityVerificationProcessRetrievableEndTimestamp</b>
<b>Type</b>	<b>dateTime</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The date/time until positive or partly positive verification process results can be retrieved without being changed any more.

<b>Field</b>	<b>identityVerificationType</b>
<b>Type</b>	<b>IdentityVerificationType</b> [see 6.24]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If present, the type of identity verification which is used in this verification process. Otherwise a default is used.

<b>Field</b>	<b>communicationWasConfirmed</b>
<b>Type</b>	<b>boolean</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If present and true, communication is already confirmed before creating this process.

<b>Field</b>	<b>allowLoginPreset</b>
<b>Type</b>	<b>boolean</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If present and true, it is allowed to preset the login dialog.

<b>Field</b>	<b>pwChangeMandatory</b>
<b>Type</b>	<b>boolean</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If present and true, a password change is mandatory.

<b>Field</b>	<b>state</b>
<b>Type</b>	<b>IdentityVerificationState</b> [see 6.23]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The current state of the verification process.

<b>Field</b>	<b>statusText</b>
<b>Type</b>	<b>UTF8String80</b> [see 6.62]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	Status of the verification in readable English.

<b>Field</b>	<b>failureReason</b>
<b>Type</b>	<b>UTF8String80</b> [see 6.62]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Failure reason of the verification in readable English.

<b>Field</b>	<b>rechargedCoinAmount</b>
<b>Type</b>	<b>Coins</b> [see 6.8]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The number of coins which were recharged as part of this identification. Only present in responses on GetIdentityVerificationProcess if requested.

<b>Field</b>	<b>sanctionListProtId</b>
<b>Type</b>	<b>long</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Only present in responses to GetIdentityVerificationProcess and restricted to specific clients.

<b>Field</b>	<b>providerId</b>
<b>Type</b>	<b>UTF8String32</b> [see <a href="#">6.57</a> ]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If present, identifies to the identity verification provider.

<b>Field</b>	<b>providerName</b>
<b>Type</b>	<b>UTF8String80</b> [see <a href="#">6.62</a> ]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If present, shows the display name of the identity verification provider.

<b>Field</b>	<b>providerWebsiteURL</b>
<b>Type</b>	<b>anyURI</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If present, refers to the URL of the website of the identity verification provider.

## 5.13 ItemProcessingResult

The result of processing the item on the sign-me system.

### Summary of ItemProcessingResult

Field Name	Field Type	Occurrence
<b>name</b>	<b>ItemName</b>	<b>1</b>
<b>signatureState</b>	<b>SignatureState</b>	<b>1</b>
<b>contentViewURL</b>	<b>ContentViewURL</b>	<b>0..1</b>
<b>contentOrSignedContent</b>	<b>HugeBlob</b>	<b>0..1</b>
<b>verificationReport</b>	<b>HugeBlob</b>	<b>0..1</b>

Table 5.13: Fields of type ItemProcessingResult

### Details of ItemProcessingResult

<b>Field</b>	<b>name</b>
<b>Type</b>	<b>ItemName</b> [see <a href="#">6.27</a> ]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The name of the item which shall be signed and/or verified, e.g. user friendly name or filename of document to be signed.

<b>Field</b>	<b>signatureState</b>
<b>Type</b>	<b>SignatureState</b> [see 6.42]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The current state of the item which is processed.

<b>Field</b>	<b>contentViewURL</b>
<b>Type</b>	<b>ContentViewURL</b> [see 6.9]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	<p>The URL for visualization of content to be signed in case of digest/hash signing.</p> <p>This URL MUST represent a document which exactly views the content which shall be signed through digest/hash signing.</p> <p>This parameter must not be present in case of document signing. The URL must have protocols http or https. The URL should locate a document in the local area network, where the signature client is operated.</p>

<b>Field</b>	<b>contentOrSignedContent</b>
<b>Type</b>	<b>HugeBlob</b> [see 6.19]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	<p>The element is present if requested by the getContentOrSignedContent directive.</p> <p>In case the signature was successful (signatureState=QES_signature_ok_*), this element contains the content which was signed and/or verified will be contained in this parameter, e.g. a PDF document or binary file.</p> <p>In case the signature was not successful or not done at all (signatureState!=QES_signature_ok_*), this element contains the unsigned content, which was delivered using ItemToBeProcessed.</p>

<b>Field</b>	<b>verificationReport</b>
<b>Type</b>	<b>HugeBlob</b> [see 6.19]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	This contains a document describing the verification procedure and result of the verification for archiving purposes, if requested by the getVerificationReport directive.

## 5.14 ItemToBeProcessed

An item to be processed by the sign-me system for signing and / or verification.

### Summary of ItemToBeProcessed

Field Name	Field Type	Occurrence
name	ItemName	1
content	HugeBlob	1
contentViewURL	ContentViewURL	0..1
detachedSignature	LargeBlob	0..1

Table 5.14: Fields of type ItemToBeProcessed

## Details of ItemToBeProcessed

<b>Field</b>	<b>name</b>
<b>Type</b>	<b>ItemName</b> [see 6.27]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The name of the item which shall be signed and/or verified, e.g. user friendly name or filename of document to be signed.

<b>Field</b>	<b>content</b>
<b>Type</b>	<b>HugeBlob</b> [see 6.19]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	<p>The content which shall be signed and/or verified will be contained in this parameter.</p> <p>The valid content of this parameter is determined by the signatureType parameter in the operation CreateSignatureProcess. Depending on the requested signature type, this could be a digest/hash value or a document/binary file value.</p> <p>In case of digest/hash values the content format and length is defined by the used digest/hash algorithm. The digest/hash algorithm used may be SHA-256, SHA-384, or SHA-512. The digest value can be plain bytes or formatted as a DER-encoded DigestInfo acc. to RFC3447.</p> <p>In case of documents/binary files, the size of this content has some practical limits below the syntactical definition of 'LargeBlob'. These limitations are depending on the signature type or on the implementation and therefore out of scope of this document. If this limit is violated, any processing of this items shall be rejected. Values within these practical limits are configured into the signature type based on the customers requirements.</p>

<b>Field</b>	<b>contentViewURL</b>
<b>Type</b>	<b>ContentViewURL</b> [see 6.9]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	<p>The URL for visualization of content to be signed in case of digest/hash signing.</p> <p>This URL MUST represent a document which exactly views the content which shall be signed through digest/hash signing.</p> <p>This parameter must not be present in case of document signing. The URL must have protocols http or https. The URL should locate a document in the local area network, where the signature client is operated.</p>

<b>Field</b>	<b>detachedSignature</b>
<b>Type</b>	<b>LargeBlob</b> [see 6.28]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	For verification-only upload of documents (or serial signing, if the content had a detached signature). Triggered by the verificationRequest the document will be verified against this detached signature.

## 5.15 ManageIdentityChangeSet

Following optional elements are present, if the change is required, otherwise they are missing from the sequence.

### Summary of ManageIdentityChangeSet

Field Name	Field Type	Occurrence
emailAddressToChange	EmailAddress	0..1
allowExtendedEmailAddressUseToChange	boolean	0..1
generateNewConfirmationCode	boolean	0..1
mobilePhoneNumberToChange	PhoneNumber	0..1
smsPinToChange	SmsPin	0..1
indoctrinationToChange	boolean	0..1
publishRequestToChange	boolean	0..1
revocationStringHashToChange	TinyBlob	0..1
usernameToChange	Username	0..1
passwdToChange	Password	0..1
passwdToChangeV2	PwChangeSet	0..1
generateNewPasswd	boolean	0..1
createDefaultTokenForCapability	TokenCapability	0..1
identityAdministrativeStateToChange	AdministrativeState	0..1
organizationAttributesToChange	Organization	0..1
preferredLanguageToChange	BCP47LanguageTag	0..1
removeIdentity	boolean	0..1
personAttributesToChange	Person	0..1
maxSecurityTokens	integer	0..1

Table 5.15: Fields of type ManageIdentityChangeSet

### Details of ManageIdentityChangeSet

Field	emailAddressToChange
Type	EmailAddress [see 6.14]
Occurrence	0..1
Meaning	If present, this new email address will be set.

Field	allowExtendedEmailAddressUseToChange
Type	boolean [see [XSD]]
Occurrence	0..1
Meaning	Deprecated.



<b>Field</b>	<b>generateNewConfirmationCode</b>
<b>Type</b>	<b>boolean</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If present, a new confirmation code will be generated and sent via email.

<b>Field</b>	<b>mobilePhoneNumberToChange</b>
<b>Type</b>	<b>PhoneNumber</b> [see <a href="#">6.35</a> ]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If present, the new value for the mobile phone number will be set. Will reset the identification state of the identity.

<b>Field</b>	<b>smsPinToChange</b>
<b>Type</b>	<b>SmsPin</b> [see <a href="#">6.45</a> ]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If present, the new value of the SMS PIN will be set. Will reset the identification state of the identity.

<b>Field</b>	<b>indoctrinationToChange</b>
<b>Type</b>	<b>boolean</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If present, the new value for the indoctrination (agreement) will be set.

<b>Field</b>	<b>publishRequestToChange</b>
<b>Type</b>	<b>boolean</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If present, the new value for the publicRequest attribute will be set.

<b>Field</b>	<b>revocationStringHashToChange</b>
<b>Type</b>	<b>TinyBlob</b> [see <a href="#">6.48</a> ]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Deprecated field. Must not be present any more.

<b>Field</b>	<b>usernameToChange</b>
<b>Type</b>	<b>Username</b> [see <a href="#">6.63</a> ]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If present, the new value for the username will be set.

<b>Field</b>	<b>passwdToChange</b>
<b>Type</b>	<b>Password</b> [see <a href="#">6.34</a> ]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Deprecated.

<b>Field</b>	<b>passwdToChangeV2</b>
<b>Type</b>	<b>PwChangeSet</b> [see 5.23]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If present, the old value is compared and the new value for the password will be set. Old and new passwords must not be the same.
<b>Field</b>	<b>generateNewPasswd</b>
<b>Type</b>	<b>boolean</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	This option is only available for specific partners. If the identity is in state 'LAWFULLY_VERIFIED_HIGH', then a Identity-Verification needs to be started again using operation CreateIdentityVerificationProcess after the password reset.
<b>Field</b>	<b>createDefaultTokenForCapability</b>
<b>Type</b>	<b>TokenCapability</b> [see 6.50]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If present, a new default token is created of indicated capability.
<b>Field</b>	<b>identityAdministrativeStateToChange</b>
<b>Type</b>	<b>AdministrativeState</b> [see 6.3]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If present, the new value for the administrative state will be set. This option is only available for specific partners.
<b>Field</b>	<b>organizationAttributesToChange</b>
<b>Type</b>	<b>Organization</b> [see 5.20]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If present, the new values of attributes will be set. This option is only available for specific partners.
<b>Field</b>	<b>preferredLanguageToChange</b>
<b>Type</b>	<b>BCP47LanguageTag</b> [see 6.6]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If present, the language preferred in communications by the identity will be changed.
<b>Field</b>	<b>removeIdentity</b>
<b>Type</b>	<b>boolean</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If present and true, the identity will be removed from the system. Restricted to specific clients.

<b>Field</b>	<b>personAttributesToChange</b>
<b>Type</b>	<b>Person</b> [see 5.21]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	<p>If present, the new values of attributes will be set. This option is available for specific partners. This option is also available for the partner ("creating partner"), which created a person account, until the person account is lawfully verified.</p> <p>By means of this option the following personal data can be changed: AcademicTitle, GivenNames, FamilyNames, PlaceOfResidence, Nationality, PlaceOfBirth, DateOfBirth. All other attributes of Person need to kept on the same values as got from a getIdentity call.</p> <p>This may be done for error correction or supplying missing data. In any case, if a person's identity was already verified by confirmCommunication or/and createIdentityVerificationProcess, the identityVerificationState will be reset to "UNVERIFIED" and any valid certificates will be revoked. To get into a identityVerificationState where signing is possible again confirmCommunication (for basic signing) as well as createIdentityVerificationProcess (for advanced and qualified signing) needs to be repeated.</p> <p>The "creating partner" can also change attributes until the person is lawfully verified: emailAddress, indoctrination (only to true), mobilePhoneNumber, username, passwd (init-passwd), preferredLanguage, publishRequest.</p>

<b>Field</b>	<b>maxSecurityTokens</b>
<b>Type</b>	<b>integer</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If present, the maximum amount of security tokens for this user will be changed. This operation is restricted to specific clients.

## 5.16 ManageTokenParameters

Specific parameters for token management.

### Summary of ManageTokenParameters

Field Name	Field Type	Occurrence
action	ManageTokenAction	1
type	UTF8String32	0..1
platform	UTF8String32	0..1
authenticationData	GenericAuthenticationData	0..10

Table 5.16: Fields of type ManageTokenParameters

## Details of ManageTokenParameters

Field	<b>action</b>
Type	<b>ManageTokenAction</b> [see 6.29]
Occurrence	<b>1</b>
Meaning	The token management action.

Field	<b>type</b>
Type	<b>UTF8String32</b> [see 6.57]
Occurrence	<b>0..1</b>
Meaning	The type of token or credential.

Field	<b>platform</b>
Type	<b>UTF8String32</b> [see 6.57]
Occurrence	<b>0..1</b>
Meaning	The platform for the token.

Field	<b>authenticationData</b>
Type	<b>GenericAuthenticationData</b> [see 6.18]
Occurrence	<b>0..10</b>
Meaning	Depending on the authentication token type specific processing of these authentication data on the server side might be required to provide a useful access token (e.g. password hash or one-time code or similar).

## 5.17 ManageTokenResult

Old and new password for changing it. Old and new passwords must not be the same.

### Summary of ManageTokenResult

Field Name	Field Type	Occurrence
token	<b>Token</b>	<b>0..1</b>
status	<b>ManageTokenStatus</b>	<b>0..1</b>
authenticationData	<b>GenericAuthenticationData</b>	<b>0..10</b>

Table 5.17: Fields of type ManageTokenResult

### Details of ManageTokenResult

Field	<b>token</b>
Type	<b>Token</b> [see 5.33]
Occurrence	<b>0..1</b>
Meaning	The token under management.

<b>Field</b>	<b>status</b>
<b>Type</b>	<b>ManageTokenStatus</b> [see 6.30]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The status of the token under management.

<b>Field</b>	<b>authenticationData</b>
<b>Type</b>	<b>GenericAuthenticationData</b> [see 6.18]
<b>Occurrence</b>	<b>0..10</b>
<b>Meaning</b>	Depending on the authentication token type specific processing of these authentication data on the server side might be required to provide a useful access token (e.g. password hash or one-time code or similar).

## 5.18 NameBlobValue

Additional data for specific use cases.

### Summary of NameBlobValue

Field Name	Field Type	Occurrence
name	UTF8String80	1
value	HugeBlob	1

Table 5.18: Fields of type NameBlobValue

### Details of NameBlobValue

<b>Field</b>	<b>name</b>
<b>Type</b>	<b>UTF8String80</b> [see 6.62]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	Name of the binary value with use case specific meaning.

<b>Field</b>	<b>value</b>
<b>Type</b>	<b>HugeBlob</b> [see 6.19]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	Binary value with use case specific meaning.

## 5.19 NameStringValue

Additional data for specific use cases.

### Summary of NameStringValue

Field Name	Field Type	Occurrence
name	UTF8String80	1
value	UTF8String512	1

Table 5.19: Fields of type NameStringValue

## Details of NameStringValue

Field	name
Type	UTF8String80 [see 6.62]
Occurrence	1
Meaning	Name of the string value with use case specific meaning.

Field	value
Type	UTF8String512 [see 6.61]
Occurrence	1
Meaning	String value with use case specific meaning.

## 5.20 Organization

Data derived from self-registration of legal persons and accompanying identity approval documents, enriched and verified by other ID information and supporting data. Depending on the use case only parts of the elements will be supplied.

### Summary of Organization

Field Name	Field Type	Occurrence
Extension of	Identity [see 5.10]	1
legalName	UTF8String80	0..1
displayName	UTF8String80	0..1
shortName	UTF8String20	0..1
debtorNumber	UTF8String80	0..1
contractId	UTF8String80	0..1
address	Address	0..1
phoneNumber	PhoneNumber	0..1
webSite	anyURI	0..1
webSiteForCommunication	anyURI	0..1
webSiteForFAQ	anyURI	0..1
webSiteForSecurityInfo	anyURI	0..1
responsibleNaturalPersonId	ObjectId	0..10
organizationIcon	LargeBlob	0..1
organizationIconMimeType	UTF8String32	0..1
additionalInfo	UTF8String400	0..1
usageWarnEmailAddress	EmailAddress	0..1
usageWarnThresholdQES	int	0..1
usageWarnThresholdIdents	int	0..1
usageWarnPeriodBeginTimeStamps	dateTime	0..1
usageWarnPeriodLengthMonths	int	0..1
usageWarnSentTimestampQES	dateTime	0..1
usageWarnSentTimestampIdents	dateTime	0..1
usageWarnActive	boolean	0..1

Table 5.20: Fields of type Organization

## Details of Organization

<b>Field</b>	<b>legalName</b>
<b>Type</b>	<b>UTF8String80</b> [see 6.62]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The full legal name of the legal person, e.g. company name as registered with the registration body.

<b>Field</b>	<b>displayName</b>
<b>Type</b>	<b>UTF8String80</b> [see 6.62]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	This name of the legal person which identifies the company on displays, WEB application front-ends etc. E.g. the portal or shop name as used in company communications.

<b>Field</b>	<b>shortName</b>
<b>Type</b>	<b>UTF8String20</b> [see 6.54]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Short name of the organization. Only present in responses and maybe restricted based on the role.

<b>Field</b>	<b>debitorNumber</b>
<b>Type</b>	<b>UTF8String80</b> [see 6.62]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Number for accounting purposes. Maybe restricted based on the role.

<b>Field</b>	<b>contractId</b>
<b>Type</b>	<b>UTF8String80</b> [see 6.62]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Number for accounting purposes. Maybe restricted based on the role.

<b>Field</b>	<b>address</b>
<b>Type</b>	<b>Address</b> [see 5.3]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The address of the legal person, e.g. company headquarter or subsidiary address as registered with the registration body.

<b>Field</b>	<b>phoneNumber</b>
<b>Type</b>	<b>PhoneNumber</b> [see 6.35]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The phone number of the legal person, e.g. the company head-quarters public main line.

<b>Field</b>	<b>webSite</b>
<b>Type</b>	<b>anyURI</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The web site of the legal person, e.g. the web site of the company headquarters or of the subsidiary.

<b>Field</b>	<b>webSiteForCommunication</b>
<b>Type</b>	<b>anyURI</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The web site of the legal person, which shall be used for communication to customers, if the sign-me front-end would show a reference to the legal person during the signing process. This could be the web site of the companies shop or banking application. Otherwise if empty, webSite will be used.

<b>Field</b>	<b>webSiteForFAQ</b>
<b>Type</b>	<b>anyURI</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The web site of the legal person, which shall be used to answer frequently asked questions of customers about the signature or activation process. This is shown by the sign-me front-end as a reference to the legal person in the context of the signing process. If not available, a sign-me FAQ site will be used.

<b>Field</b>	<b>webSiteForSecurityInfo</b>
<b>Type</b>	<b>anyURI</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The web site of the legal person, which shall be used to inform customers about security topics related to the signing or activation process. This is shown by the sign-me front-end as a reference to the legal person in the context of the signing or activation process. This could be the web site of the companies shop or banking application. If not available, a sign-me defined security info site will be used.

<b>Field</b>	<b>responsibleNaturalPersonId</b>
<b>Type</b>	<b>ObjectId</b> [see <a href="#">6.32</a> ]]
<b>Occurrence</b>	<b>0..10</b>
<b>Meaning</b>	The object ID of natural persons which are responsible for acting on behalf of the legal entity on that system, e.g. applying seals in the legal person's name or changing data as far as allowed. For the number of responsible persons there are of course practical limits, which are out-of-scope of this specification.

<b>Field</b>	<b>organizationIcon</b>
<b>Type</b>	<b>LargeBlob</b> [see <a href="#">6.28</a> ]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The icon of the organization to customize dialogs of the web application.



<b>Field</b>	<b>organizationIconMimeType</b>
<b>Type</b>	<b>UTF8String32</b> [see 6.57]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The mime type of the icon of the organization to customize dialogs of the web application. Allowed values are: 'image/png', 'image/gif', 'image/jpeg'.
<b>Field</b>	<b>additionalInfo</b>
<b>Type</b>	<b>UTF8String400</b> [see 6.59]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Additional info. Maybe restricted based on the role.
<b>Field</b>	<b>usageWarnEmailAddress</b>
<b>Type</b>	<b>EmailAddress</b> [see 6.14]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Valid contact email address of the organization to send usage warn emails to.
<b>Field</b>	<b>usageWarnThresholdQES</b>
<b>Type</b>	<b>int</b> [see XSD]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The threshold for usage of QES signatures (not signature processes) by the organization. If this threshold is passed since usageWarnPeriodBeginTimestamp a warning email is issued once to usageWarnEmailAddress.
<b>Field</b>	<b>usageWarnThresholdIdents</b>
<b>Type</b>	<b>int</b> [see XSD]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The threshold for usage of identifications (all types of identifications) by the organization. If this threshold is passed since usageWarnPeriodBeginTimestamp a warning email is issued once to usageWarnEmailAddress.
<b>Field</b>	<b>usageWarnPeriodBeginTimestamp</b>
<b>Type</b>	<b>dateTime</b> [see XSD]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The date/time when the counting period to evaluating the usage warn thresholds begins. The date/time value will be rounded down to the month begin midnight (MET/MEST).

<b>Field</b>	<b>usageWarnPeriodLengthMonths</b>
<b>Type</b>	<b>int</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The length of the current usage warn period in months. The period starts from usageWarnPeriodBeginTimestamp and is restarted, if the usageWarnPeriodBeginTimestamp plus usageWarnPeriodLengthMonths is exceeded.

<b>Field</b>	<b>usageWarnSentTimestampQES</b>
<b>Type</b>	<b>dateTime</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The date/time when the warning for QES usage was sent.

<b>Field</b>	<b>usageWarnSentTimestampIdents</b>
<b>Type</b>	<b>dateTime</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The date/time when the warning for identification usage was sent.

<b>Field</b>	<b>usageWarnActive</b>
<b>Type</b>	<b>boolean</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If present and true, this indicates that the usage warn function is active. The warn function is automatically deactivated, if both thresholds are passed.

## 5.21 Person

Data contained in an ID card or passport for natural persons, enriched by other ID information and supporting data. Depending on the use case only parts of the elements will be supplied.

## Summary of Person

Field Name	Field Type	Occurrence
Extension of	<b>Identity</b> [see 5.10]	<b>1</b>
familyNames	<b>UTF8String180</b>	<b>0..1</b>
givenNames	<b>UTF8String180</b>	<b>0..1</b>
academicTitle	<b>UTF8String180</b>	<b>0..1</b>
artisticName	<b>UTF8String180</b>	<b>0..1</b>
dateOfBirth	<b>GenericDate</b>	<b>0..1</b>
placeOfBirth	<b>Place</b>	<b>0..1</b>
placeOfResidence	<b>Address</b>	<b>0..1</b>
nationality	<b>ICAOCountryName</b>	<b>0..1</b>
mobilePhoneNumber	<b>PhoneNumber</b>	<b>0..1</b>
mobilePhoneNumberVerified	<b>boolean</b>	<b>0..1</b>
smsPin	<b>SmsPin</b>	<b>0..1</b>
externalUserId	<b>Username</b>	<b>0..1</b>
maxSecurityTokens	<b>integer</b>	<b>0..1</b>
creatingOrganizationId	<b>ObjectId</b>	<b>0..1</b>
numberOfCertificates	<b>integer</b>	<b>0..1</b>
numberOfSignatures	<b>integer</b>	<b>0..1</b>

Table 5.21: Fields of type Person

## Details of Person

<b>Field</b>	<b>familyNames</b>
<b>Type</b>	<b>UTF8String180</b> [see 6.53]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Family names (surnames) of the person. Mandatory field for use in RegisterIdentity/Create- /UpdateRegistrationProcess.

<b>Field</b>	<b>givenNames</b>
<b>Type</b>	<b>UTF8String180</b> [see 6.53]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Given names (first names) of the person. Mandatory field for use in RegisterIdentity/Create- /UpdateRegistrationProcess.

<b>Field</b>	<b>academicTitle</b>
<b>Type</b>	<b>UTF8String180</b> [see 6.53]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Academic titles of the person (if visible on the ID document). Optional field for use in RegisterIdentity/Create- /UpdateRegistrationProcess.

<b>Field</b>	<b>artisticName</b>
<b>Type</b>	<b>UTF8String180</b> [see 6.53]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Artistic names of the person (if visible on the ID document). Optional field for use in RegisterIdentity/Create- /UpdateRegistrationProcess.

<b>Field</b>	<b>dateOfBirth</b>
<b>Type</b>	<b>GenericDate</b> [see 5.8]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Date of birth of the person. Optional field for use in RegisterIdentity/Create- /UpdateRegistrationProcess. Only present in responses on GetIdentity if the request was made using identityId.

<b>Field</b>	<b>placeOfBirth</b>
<b>Type</b>	<b>Place</b> [see 5.22]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Typically only the city of birth, which will be forwarded in parameter 'placeOfBirth.freetextPlace'. If city and country of birth is explicitly known, then 'placeOfBirth.structuredPlace.city' and 'placeOfBirth.structuredPlace.country' must be used to encode the information. If none of the above is known about the place of birth, indicative information needs to be passed in element 'placeOfBirth.noPlaceInfo'. Optional field for use in RegisterIdentity/Create- /UpdateRegistrationProcess. Only present in responses on GetIdentity if the request was made using identityId.

<b>Field</b>	<b>placeOfResidence</b>
<b>Type</b>	<b>Address</b> [see 5.3]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	In place of residence city, country, street and zipCode needs to be defined. The state field is optional. Optional field for use in RegisterIdentity/Create- /UpdateRegistrationProcess. Only present in responses on GetIdentity if the request was made using identityId.

<b>Field</b>	<b>nationality</b>
<b>Type</b>	<b>ICAOCountryName</b> [see 6.20]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	<p>The nationality of the person as specified on the ID document as ICAO country name.</p> <p>Optional field for use in RegisterIdentity/Create-/UpdateRegistrationProcess.</p> <p>Only present in responses on GetIdentity if the request was made using identityId.</p>

<b>Field</b>	<b>mobilePhoneNumber</b>
<b>Type</b>	<b>PhoneNumber</b> [see 6.35]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	<p>The mobile phone number of the natural person, which will be used for contacting the person or for multi-factor authentication.</p> <p>Optional field for use in RegisterIdentity/Create-/UpdateRegistrationProcess.</p> <p>Will be truncated and prefixed with '***' in responses on GetIdentity.</p>

<b>Field</b>	<b>mobilePhoneNumberVerified</b>
<b>Type</b>	<b>boolean</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	<p>This indicates whether the mobile phone number address was verified by the system.</p> <p>Only present in responses to GetIdentity.</p>

<b>Field</b>	<b>smsPin</b>
<b>Type</b>	<b>SmsPin</b> [see 6.45]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	<p>The SMS can be secured by this in specific applications. If used, this will be announced on a project basis.</p> <p>Optional field for use in RegisterIdentity/Create-/UpdateRegistrationProcess.</p> <p>Will be null in responses on GetIdentity.</p>

<b>Field</b>	<b>externalUserId</b>
<b>Type</b>	<b>Username</b> [see 6.63]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	<p>The external user id of the identity used on specific external components.</p> <p>Only present in responses to GetIdentity.</p>

<b>Field</b>	<b>maxSecurityTokens</b>
<b>Type</b>	<b>integer</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	<p>The maximum amount of signature tokens for this person. Only present in responses to GetIdentity and restricted to specific clients.</p>

<b>Field</b>	<b>creatingOrganizationId</b>
<b>Type</b>	<b>ObjectId</b> [see 6.32]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The ID of the organization, which registered the person. Only present in responses to GetIdentity and restricted to specific clients.

<b>Field</b>	<b>numberOfCertificates</b>
<b>Type</b>	<b>integer</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The number of certificates. Only present in responses to GetIdentity and restricted to specific clients.

<b>Field</b>	<b>numberOfSignatures</b>
<b>Type</b>	<b>integer</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The number of signatures. Only present in responses to GetIdentity and restricted to specific clients.

## 5.22 Place

Will be used for PlaceOfBirth type location information.

If only the city is defined use choice 'freetextPlace'.

If city and country is explicitly known, then 'structuredPlace.city' and 'structuredPlace.country' will be used to encode the information. If more is known, use other attributes.

If none of the above is known about the place, indicative information needs to be passed in element 'placeOfBirth.noPlaceInfo'.

### Alternative Fields of Place

Field Name (any of)	Field Type	Occurrence
structuredPlace	Address	1
freetextPlace	PlaceFreetextInfo	1
noPlaceInfo	PlaceFreetextInfo	1

Table 5.22: Fields of type Place

### Details of Place

<b>Field</b>	<b>structuredPlace</b>
<b>Type</b>	<b>Address</b> [see 5.3]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	Structured information about place is known.

<b>Field</b>	<b>freetextPlace</b>
<b>Type</b>	<b>PlaceFreetextInfo</b> [see 6.36]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	Free form place info.

<b>Field</b>	<b>noPlaceInfo</b>
<b>Type</b>	<b>PlaceFreetextInfo</b> [see 6.36]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	Indicative information why place info cannot be specified.

## 5.23 PwChangeSet

This structure contains old and new password for changing the password. Old and new passwords must not be the same if a manageIdentity operation with these parameters is issued.

### Summary of PwChangeSet

Field Name	Field Type	Occurrence
oldPw	Password	1
newPw	Password	1

Table 5.23: Fields of type PwChangeSet

### Details of PwChangeSet

<b>Field</b>	<b>oldPw</b>
<b>Type</b>	<b>Password</b> [see 6.34]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The old value of password.

<b>Field</b>	<b>newPw</b>
<b>Type</b>	<b>Password</b> [see 6.34]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	If old value of password is o.k., and new is not same, the new value will be set.

## 5.24 RedirectURLs

These are redirect-URLs where the user of the sign-me application front-end is redirected to in case the request processing is performed o.k., or in case there was an active cancel by the user or a hard failure detected by the system. The redirect-URLs have a maximum length of 256 characters. They must consist of well-formed URLs. The only protocol to be used is http or https.

### Summary of RedirectURLs

Field Name	Field Type	Occurrence
redirectURLOnOk	anyURI	1
redirectURLOnCancel	anyURI	1
redirectURLOnFail	anyURI	1

Table 5.24: Fields of type RedirectURLs

## Details of RedirectURLs

<b>Field</b>	<b>redirectURLOnOk</b>
<b>Type</b>	<b>anyURI</b> [see [XSD]]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The redirect URL defined by the partner shop/portal in case the request was successfully executed.

<b>Field</b>	<b>redirectURLOnCancel</b>
<b>Type</b>	<b>anyURI</b> [see [XSD]]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The redirect URL defined by the partner shop/portal in case the request was cancelled by the user (not through the API).

<b>Field</b>	<b>redirectURLOnFail</b>
<b>Type</b>	<b>anyURI</b> [see [XSD]]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The redirect URL defined by the partner shop/portal in case the request processing failed.

## 5.25 RegistrationProcess

The registration process is used to handle the progress of registration request by the partner and retrieval of the results.

### Summary of RegistrationProcess

Field Name	Field Type	Occurrence
registrationProcessId	ObjectId	1
registrationProcessURL	RegistrationProcessURL	0..1
registrationState	RegistrationState	1
organizationId	ObjectId	1
personIdentityId	ObjectId	0..1
redirectURLs	RedirectURLs	0..1
createdTimestamp	dateTime	1
completedTimestamp	dateTime	1
registrationProcessUsableEndTimestamp	dateTime	1
registrationProcessRetrievalUsableEndTimestamp	dateTime	1

Table 5.25: Fields of type RegistrationProcess



## Details of RegistrationProcess

<b>Field</b>	<b>registrationProcessId</b>
<b>Type</b>	<b>ObjectId</b> [see 6.32]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The object id of the registration process received by calls to the sign-me system.

<b>Field</b>	<b>registrationProcessURL</b>
<b>Type</b>	<b>RegistrationProcessURL</b> [see 6.37]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The unique URL where the registration WEB application is reachable for the specific registration process..

<b>Field</b>	<b>registrationState</b>
<b>Type</b>	<b>RegistrationState</b> [see 6.38]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The current state is of the item which is processed.

<b>Field</b>	<b>organizationId</b>
<b>Type</b>	<b>ObjectId</b> [see 6.32]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	This id of the organization which created this registration process.

<b>Field</b>	<b>personIdentityId</b>
<b>Type</b>	<b>ObjectId</b> [see 6.32]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If present, the id of the person being registered in this registration process. Guaranteed to be present only in case of state='SUCCESS'.

<b>Field</b>	<b>redirectURLs</b>
<b>Type</b>	<b>RedirectURLs</b> [see 5.24]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	This can contain URLs to redirect to in case of "OK"-, "FAIL"-, and "CANCEL"-results.

<b>Field</b>	<b>createdTimestamp</b>
<b>Type</b>	<b>dateTime</b> [see [XSD]]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The date/time when the partner created the registration process.

<b>Field</b>	<b>completedTimestamp</b>
<b>Type</b>	<b>dateTime</b> [see [XSD]]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The date/time when the process was completed (update/complete=true and all data passed plausibility checks).

<b>Field</b>	<b>registrationProcessUsableEndTimestamp</b>
<b>Type</b>	<b>dateTime</b> [see <a href="#">XSD</a> ]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The date/time until the registration process is valid for use and registrations can be made (registrationState=REQUESTED).

<b>Field</b>	<b>registrationProcessRetrievableEndTimestamp</b>
<b>Type</b>	<b>dateTime</b> [see <a href="#">XSD</a> ]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The date/time until registration process results can retrieved without being changed any more (registrationState=SUCCESSFUL or TIMEOUT).

## 5.26 RoleAssociation

The combination of role and optional applicable restrictions (not used so far).

### Summary of RoleAssociation

Field Name	Field Type	Occurrence
allowedRoleName	RoleName	1
restriction	string	0..1

Table 5.26: Fields of type RoleAssociation

### Details of RoleAssociation

<b>Field</b>	<b>allowedRoleName</b>
<b>Type</b>	<b>RoleName</b> [see <a href="#">6.39</a> ]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The role names which the identity can use to login.

<b>Field</b>	<b>restriction</b>
<b>Type</b>	<b>string</b> [see <a href="#">XSD</a> ]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Not used.

## 5.27 SecurityHeader

The security header is placed as a mandatory requirement at the beginning of all parameter lists of request, except the login-request. It identifies the login session by means of the access token and the client IP address. Both parameters must not change during a login session until the next authorize/authorizeStep2 or refresh operation.

## Summary of SecurityHeader

Field Name	Field Type	Occurrence
accessToken	AccessToken	1
clientIpAddress	IpAddressNumeric	1

Table 5.27: Fields of type SecurityHeader

## Details of SecurityHeader

Field	accessToken
Type	AccessToken [see 6.1]
Occurrence	1
Meaning	The access token initially from Login/LoginStep2 response. May be renewed by Authorize/AuthorizeStep2 or Refresh operations during the lifetime of a login session.

Field	clientIpAddress
Type	IpAddressNumeric [see 6.26]
Occurrence	1
Meaning	The IP-address of the client.

## 5.28 SignatureParameters

By means of signature parameters the visibility of the signature field or the usage of a field within (PDF) documents can be controlled (The use of signature parameters must be allowed for the signature type, see 6.43, e.g. S\_QES\_DOC\_PADES\_B\_B1).

### CAUTION:

Using these parameters, the API client of the partner is mainly in control and therefore responsible for the visualization of the signature field. Please, make sure based on test signatures on the target documents in the reference test system, that the result fits your purpose.

## Summary of SignatureParameters

Field Name	Field Type	Occurrence
existingFieldName	UTF8String80	0..1
visible	boolean	0..1
xPosition	short	0..1
yPosition	short	0..1
xSize	short	0..1
ySize	short	0..1
pageRangeType	UTF8String32	0..1
pageNumber	short	0..1
iconSourceType	UTF8String32	0..1
customSignatureIcon	LargeBlob	0..1
customSignatureIconMimeType	UTF8String32	0..1
rotateArc	short	0..1
fontSize	short	0..1
textVerticalAlignment	UTF8String32	0..1
textHorizontalAlignment	UTF8String32	0..1
signatureGraphicHorizontal.. ..Alignment	UTF8String32	0..1

Table 5.28: Fields of type SignatureParameters

## Details of SignatureParameters

<b>Field</b>	<b>existingFieldName</b>
<b>Type</b>	UTF8String80 [see 6.62]
<b>Occurence</b>	0..1
<b>Meaning</b>	<p>If present, an existing field in the document with this name will be used for placing the visible signature.</p> <p>If this option is used, elements for creation of fields must not be present (x/yPosition, x/ySize, pageRangeType/number, rotateArc) and vice versa.</p> <p>One or more named signature fields to be addressed by this element can be inserted into the PDF files using the usual PDF designers, one of many examples being <a href="https://www.intarsys.de/dl/pdfal">https://www.intarsys.de/dl/pdfal</a>.</p>
<b>Field</b>	<b>visible</b>
<b>Type</b>	boolean [see [XSD]]
<b>Occurence</b>	0..1
<b>Meaning</b>	<p>If not present, the visibility depends on default value for the visibility defined for the signature type (see 6.43).</p> <p>If present and false, the signature will be invisible.</p> <p>If present and true, the visibility depends on default values defined for the signature type (see 6.43) and the other parameters in this data structure.</p>

<b>Field</b>	<b>xPosition</b>
<b>Type</b>	<b>short</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If present and $\geq 0$ , a field will be created at this x position in the document for placing the visible signature. The unit is PDF Points equivalent to 1/72 inch. The lower/left corner of the document is xPosition=0,yPosition=0. The maximum value is 6740 PDF Points. If this option is used, yPosition must also be present.

<b>Field</b>	<b>yPosition</b>
<b>Type</b>	<b>short</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If present and $\geq 0$ , a field will be created at this y position in the document for placing the visible signature. The unit is PDF Points equivalent to 1/72 inch. The lower/left corner of the document is xPosition=0,yPosition=0. The maximum value is 6740 PDF Points. If this option is used, xPosition must also be present.

<b>Field</b>	<b>xSize</b>
<b>Type</b>	<b>short</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If present and $\geq 0$ , a field will be created with this x size in the document for placing the visible signature. The unit is PDF Points equivalent to 1/72 inch. The maximum value is 6740 PDF Points. If this option is used, ySize must also be present.

<b>Field</b>	<b>ySize</b>
<b>Type</b>	<b>short</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If present and $\geq 0$ , a field will be created with this y size in the document for placing the visible signature. The unit is PDF Points equivalent to 1/72 inch. The maximum value is 6740 PDF Points. If this option is used, xSize must also be present.

<b>Field</b>	<b>pageRangeType</b>
<b>Type</b>	<b>UTF8String32</b> [see <a href="#">6.57</a> ]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If present, the option decides which pages will hold the signature field. Allowed values are: onFirst, onLast, onSingle. If option onSingle is used, element pageNumber must also be present. DEPRECATED: onAll option is deprecated and will be mapped to onFirst in the future.

<b>Field</b>	<b>pageNumber</b>
<b>Type</b>	<b>short</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	This element must be present if option onSingle is chosen for pageRangeType. This defines the page number where the signature field is placed.

<b>Field</b>	<b>iconSourceType</b>
<b>Type</b>	<b>UTF8String32</b> [see <a href="#">6.57</a> ]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	<p>If present, the strategy can be chosen for the icon which shall be visible in the signature field. Allowed values are: useCustomIcon, usePartnerIcon, dontUseIcon.</p> <p>If this is set to useCustomIcon, then the icon defined in customSignatureIcon will be used for the signature field.</p> <p>If this is set to usePartnerIcon, then the icon defined in the partner organization will be used for the signature field.</p> <p>If this is set to dontUseIcon, then there will not be an icon in the signature field.</p> <p>Otherwise, the default icon defined for the signature type will be used.</p> <p>If option useCustomIcon is used, elements customSignatureIcon and customSignatureIconMimeType must also be present.</p>

<b>Field</b>	<b>customSignatureIcon</b>
<b>Type</b>	<b>LargeBlob</b> [see <a href="#">6.28</a> ]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	<p>The icon to be used for a specific instance of signature process.</p> <p>CAUTION:</p> <p>Please, make sure, that you or your organization is the holder of all required rights to use the icon for the purpose of signature visibility customization !</p>

<b>Field</b>	<b>customSignatureIconMimeType</b>
<b>Type</b>	<b>UTF8String32</b> [see <a href="#">6.57</a> ]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	<p>The mime type of the icon contained in customSignatureIcon.</p> <p>Allowed values are: 'image/png', 'image/gif', 'image/jpeg'.</p>

<b>Field</b>	<b>rotateArc</b>
<b>Type</b>	<b>short</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	<p>If present, a field will be created in the document for placing the visible signature and rotated by the arc given in this field.</p> <p>Allowed values are: 0,90,180,270.</p> <p>DEPRECATED: this option is deprecated and will be mapped to 0 in the future.</p>

<b>Field</b>	<b>fontSize</b>
<b>Type</b>	<b>short</b> [see <a href="#">XSD</a> ]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If present, the specified font size will be used for a visible signature field and its annotation. Default for visible signature fields is '0' which means the Font will be scaled to fill the size of the available space of the signature field.

<b>Field</b>	<b>textVerticalAlignment</b>
<b>Type</b>	<b>UTF8String32</b> [see <a href="#">6.57</a> ]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If present, the vertical alignment of text can be controlled for the signature field. Allowed values are: TOP, MIDDLE, BOTTOM. Default is TOP.

<b>Field</b>	<b>textHorizontalAlignment</b>
<b>Type</b>	<b>UTF8String32</b> [see <a href="#">6.57</a> ]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If present, the horizontal alignment of text can be controlled for the signature field. Allowed values are: LEFT, RIGHT, CENTER, JUSTIFIED. Default is RIGHT.

<b>Field</b>	<b>signatureGraphicHorizontalAlignment</b>
<b>Type</b>	<b>UTF8String32</b> [see <a href="#">6.57</a> ]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If present, the horizontal alignment of the icon can be controlled for the signature field. Allowed values are: LEFT, RIGHT. Default is LEFT.

## 5.29 SignatureProcess

The signature process is used to handle the progress of signature request by the partner, authorization by the user, verification and retrieval of the results.

## Summary of SignatureProcess

Field Name	Field Type	Occurrence
signatureProcessId	ObjectId	1
signatureProcessURL	SignatureProcessURL	0..1
organizationId	ObjectId	1
signerIdentityId	ObjectId	1
certificate	MediumBlob	0..1
certificateV2	Certificate	0..1
timestampCertificate	Certificate	0..1
ocspCertificate	Certificate	0..1
redirectURLs	RedirectURLs	0..1
suppressFinalDialog	boolean	0..1
createdTimestamp	dateTime	1
signerRequestedContentBefo..	dateTime	1
..reSigningTimestamp		
signaturePerformedTimestamp	dateTime	1
signatureValidatedTimestamp	dateTime	1
signatureProcessUsableEndT..	dateTime	1
..imestamp		
signatureProcessRetrievabl..	dateTime	1
..eEndTimestamp		
itemsAndTheirProcessingRes..	ItemProcessingResult	0..N
..ults		
signatureTemplateId	ObjectId	1
signatureParameters	SignatureParameters	0..1
lastError	ErrorInfo	0..1

Table 5.29: Fields of type SignatureProcess

## Details of SignatureProcess

Field	signatureProcessId
Type	ObjectId [see 6.32]
Occurence	1
Meaning	The object id of the signature process received by calls to the sign-me system.

Field	signatureProcessURL
Type	SignatureProcessURL [see 6.41]
Occurence	0..1
Meaning	The unique URL where the signature WEB application is reachable for the specific signature process. Can be empty/not present in case of signature types without further user interaction.

Field	organizationId
Type	ObjectId [see 6.32]
Occurence	1
Meaning	This id of the organization which created this signature process.



<b>Field</b>	<b>signerIdentityId</b>
<b>Type</b>	<b>ObjectId</b> [see 6.32]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The id of the identity which will sign/signed the content during this signature process. See GetSignatureProcess for the availability and content of this attribute in the response.

<b>Field</b>	<b>certificate</b>
<b>Type</b>	<b>MediumBlob</b> [see 6.31]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The (binary) certificate of the key, which can verify the signature(s) performed in this signature process. Only present and non-empty, if the document is signed, the certificate could be derived and if requested by the getContentOrSignedContent directive. DEPRECATED: use certificateV2 instead.

<b>Field</b>	<b>certificateV2</b>
<b>Type</b>	<b>Certificate</b> [see 5.5]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The certificate of the key, which will be used in this signature process. This structure contains binary representation including the most important meta-data and references to the issuer certificate for querying on this API. This is the same representation like in GetTokensForIdentity (see 4.25). Present, if interface revision greater 8315 is required.

<b>Field</b>	<b>timestampCertificate</b>
<b>Type</b>	<b>Certificate</b> [see 5.5]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The certificate of the time-stamp, which was used for the signature. Only available for signature types, which include qualified time-stamps, and only after the signature is successful (OK_UNVERIFIED or OK_VERIFIED).

<b>Field</b>	<b>ocspCertificate</b>
<b>Type</b>	<b>Certificate</b> [see 5.5]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The certificate of the OCSP responder, which was queried for the signature. Only available for signature types, which provide LTV information, and only after the signature is successful (OK_UNVERIFIED or OK_VERIFIED).

<b>Field</b>	<b>redirectURLs</b>
<b>Type</b>	<b>RedirectURLs</b> [see 5.24]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	This can contain URLs to redirect to in case of "OK"-, "FAIL"-, and "CANCEL"-results.

<b>Field</b>	<b>suppressFinalDialog</b>
<b>Type</b>	<b>boolean</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	This decides whether after the signing procedure call the user should go to a sign-me page to download the signed document or back to the partner website.
<b>Field</b>	<b>createdTimestamp</b>
<b>Type</b>	<b>dateTime</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The date/time when the partner created the signature process.
<b>Field</b>	<b>signerRequestedContentBeforeSigningTimestamp</b>
<b>Type</b>	<b>dateTime</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The date/time when the signer requested the unsigned content (in state signatureState=REQUESTED).
<b>Field</b>	<b>signaturePerformedTimestamp</b>
<b>Type</b>	<b>dateTime</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The date/time when the signer signed the content successfully (transition from signatureState=REQUESTED to OK_UNVERIFIED).
<b>Field</b>	<b>signatureValidatedTimestamp</b>
<b>Type</b>	<b>dateTime</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The date/time when the partner validated the content (transition from signatureState=OK_UNVERIFIED to OK_VERIFIED; only possible for document signing).
<b>Field</b>	<b>signatureProcessUsableEndTimestamp</b>
<b>Type</b>	<b>dateTime</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The date/time until the signature process is valid for use and signatures can be made (signatureState=REQUESTED or OK_UNVERIFIED). This time is depending on the signature type.
<b>Field</b>	<b>signatureProcessRetrievableEndTimestamp</b>
<b>Type</b>	<b>dateTime</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The date/time until positive or partly positive signature process results can retrieved without being changed any more (signatureState=OK_UNVERIFIED* and OK_VERIFIED). This time is depending on the signature type. Cancelled or time-out signature processes will be removed earlier than this time-stamp.

<b>Field</b>	<b>itemsAndTheirProcessingResults</b>
<b>Type</b>	<b>ItemProcessingResult</b> [see 5.13]
<b>Occurrence</b>	<b>0..N</b>
<b>Meaning</b>	The result of processing the items on the sign-me system which were passed during createSignatureProcess.

<b>Field</b>	<b>signatureTemplateId</b>
<b>Type</b>	<b>ObjectId</b> [see 6.32]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The object id of the signature template used to create the signature. The signature template corresponds to the signature type which was ordered to be applied in this process.

<b>Field</b>	<b>signatureParameters</b>
<b>Type</b>	<b>SignatureParameters</b> [see 5.28]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	By means of signature parameters the visibility of the signature field or the usage of a field within (PDF) documents can be controlled. This parameter is only available, if used during create. Otherwise defaults according to the signature type will be used. The use of signature parameters must be allowed for the signature type (see 6.43).

<b>Field</b>	<b>lastError</b>
<b>Type</b>	<b>ErrorInfo</b> [see 5.7]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The last error seen and persisted during the signature process, if any.

## 5.30 SignatureTemplate

Specifies the properties of signatures of a type indicated by 'signatureType'.

## Summary of SignatureTemplate

Field Name	Field Type	Occurrence
signatureTemplateId	ObjectId	1
administrativeState	AdministrativeState	1
signatureType	SignatureType	1
requiredTokenCapability	TokenCapability	1
oneFactorAuthenticationAllowed	boolean	1
signatureFormat	SignatureFormat	1
forDigestSigning	boolean	1
usableDuration	Duration	1
retrievableDuration	Duration	1
archivingDuration	Duration	1
coinsRequired	Coins	0..1
activeForReservation	boolean	0..1
signParamsAllowed	boolean	0..1
onDemandTokenActivationPossible	boolean	0..1
organizationChargeCoinAmount	UTF8String4096	0..1
onDemandCertificateTypeShortTerm	UTF8String32	0..1
onDemandCertificateTypeLongTerm	UTF8String32	0..1
authorizedOrganizations	UTF8String4096	0..1
signatureParameters	SignatureParameters	0..1
partnerIcon	LargeBlob	0..1
partnerIconMimeType	UTF8String32	0..1
tanSenderName	UTF8String40	0..1
tanMessageText	UTF8String200	0..1
appPurposeText	UTF8String200	0..1
sapMaterialNumber	UTF8String20	0..1

Table 5.30: Fields of type SignatureTemplate

## Details of SignatureTemplate

Field	signatureTemplateId
Type	ObjectId [see 6.32]
Occurrence	1
Meaning	The object id of the signature template received by calls to the sign-me system.

Field	administrativeState
Type	AdministrativeState [see 6.3]
Occurrence	1
Meaning	If the signature template is unlocked, signature processes can be created using the signature type defined by this template. Otherwise creation of signature processes using this type is not possible.

<b>Field</b>	<b>signatureType</b>
<b>Type</b>	<b>SignatureType</b> [see 6.43]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The type of signature which is defined by this template.
<b>Field</b>	<b>requiredTokenCapability</b>
<b>Type</b>	<b>TokenCapability</b> [see 6.50]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The token capability which is required to sign according to these signature types.
<b>Field</b>	<b>oneFactorAuthenticationAllowed</b>
<b>Type</b>	<b>boolean</b> [see [XSD]]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	Signing is allowed for identities authenticated by a single factor, otherwise two factors are needed.
<b>Field</b>	<b>signatureFormat</b>
<b>Type</b>	<b>SignatureFormat</b> [see 6.40]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The format of the signature.
<b>Field</b>	<b>forDigestSigning</b>
<b>Type</b>	<b>boolean</b> [see [XSD]]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The signature type is used for digest signing, i.e. the content to be signed must be a digest/hash value. Verification of signatures is not possible for these signature types.
<b>Field</b>	<b>usableDuration</b>
<b>Type</b>	<b>Duration</b> [see 6.13]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The duration within the signature processes following this template are usable for signing.
<b>Field</b>	<b>retrievableDuration</b>
<b>Type</b>	<b>Duration</b> [see 6.13]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The duration within the signature processes following this template is retrievable from the system.
<b>Field</b>	<b>archivingDuration</b>
<b>Type</b>	<b>Duration</b> [see 6.13]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The duration within the signature processes following this template will be archived.

<b>Field</b>	<b>coinsRequired</b>
<b>Type</b>	<b>Coins</b> [see 6.8]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The number of coins which will be charged for signatures defined by this type, hence need to be present in the coins counter of the identity before the signature. If the field is not present, or the zero, no coins will be charged for this signature type.

<b>Field</b>	<b>activeForReservation</b>
<b>Type</b>	<b>boolean</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Template can only be used for reservations.

<b>Field</b>	<b>signParamsAllowed</b>
<b>Type</b>	<b>boolean</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Signature Parameters allowed in template.

<b>Field</b>	<b>onDemandTokenActivationPossible</b>
<b>Type</b>	<b>boolean</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Token can be activated on demand.

<b>Field</b>	<b>organizationChargeCoinAmount</b>
<b>Type</b>	<b>UTF8String4096</b> [see 6.60]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	organization ChargeCoinAmount properties

<b>Field</b>	<b>onDemandCertificateTypeShortTerm</b>
<b>Type</b>	<b>UTF8String32</b> [see 6.57]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The certificate type used for short-term certificates.

<b>Field</b>	<b>onDemandCertificateTypeLongTerm</b>
<b>Type</b>	<b>UTF8String32</b> [see 6.57]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The certificate type used for long-term certificates.

<b>Field</b>	<b>authorizedOrganizations</b>
<b>Type</b>	<b>UTF8String4096</b> [see 6.60]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	authorized organizations

<b>Field</b>	<b>signatureParameters</b>
<b>Type</b>	<b>SignatureParameters</b> [see 5.28]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	By means of signature parameters the visibility of the signature field or the usage of a field within (PDF) documents can be controlled. This parameter is only available, if used during create. Otherwise defaults according to the signature type will be used. The use of signature parameters must be allowed for the signature type (see 6.43).

<b>Field</b>	<b>partnerIcon</b>
<b>Type</b>	<b>LargeBlob</b> [see 6.28]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The icon of the organization to customize dialogs of the web application.

<b>Field</b>	<b>partnerIconMimeType</b>
<b>Type</b>	<b>UTF8String32</b> [see 6.57]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The mime type of the icon of the organization to customize dialogs of the web application. Allowed values are: 'image/png', 'image/gif', 'image/jpeg'.

<b>Field</b>	<b>tanSenderName</b>
<b>Type</b>	<b>UTF8String40</b> [see 6.58]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	TAN sender name

<b>Field</b>	<b>tanMessageText</b>
<b>Type</b>	<b>UTF8String200</b> [see 6.55]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Text of the TAN message

<b>Field</b>	<b>appPurposeText</b>
<b>Type</b>	<b>UTF8String200</b> [see 6.55]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	Text shown on the 2FA for signing

<b>Field</b>	<b>sapMaterialNumber</b>
<b>Type</b>	<b>UTF8String20</b> [see 6.54]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	SAP material number

### 5.31 SignerStatus

Contains the signer status for a username and signature type combination.  
The signer status is grouped into three categories:

- Global Status
- Detailed Status
- Informational Only

**Global Status:** The signer status structure contains a summary expression whether signing is possible or not (see `signingPossible/Global Status`).

**Detailed Status:** In case signing is not possible more detailed status information shows what needs to be done to bring the user into a sufficient status for signing (see `signerExisting/Detailed Status` etc.).

The structure also contains IDs of any pending identity verification process or signature processes of the signer with the indicated username.

These IDs can be used to get a detailed status of these processes or offer a re-entry into the process for the user by redirecting him to the corresponding process URLs.

**Informational Only:** Further information like timestamps or IDs, or states are useful for logging or communication about the issue.

**Application:** Please refer to sample code `FastTrackSample.java` to see a practical application of the signer status call. This sample shows in which sequence the **Detailed Status** information can be checked in case signing is not possible, and which follow up action can be offered to the user to fix his/her signer status.

A complete priority sequence of checking (indicated as priority below) if `signingPossible=false` is the following:



Prio	Detailed Status	Follow-Up Action
1	<b>signerExisting</b>	If <b>false</b> a signer needs to be registered and identified. The signer needs to agree to terms and conditions as well as certificate production. Afterwards signing can start in same session, if required.
2	<b>signerEmailAddress.. ..NeedsConfirmation</b>	If <b>true</b> a signer needs to confirm its e-mail address, because he or she has registered outside of the session at the <a href="http://www.sign-me.de">www.sign-me.de</a> web site, but has not completed, yet.
3	<b>signerPassword.. ..NeedsReset</b>	If <b>true</b> a signer needs to reset its password using the login form at <a href="http://www.sign-me.de">www.sign-me.de</a> , because he/she has locked it's account by entering a wrong password 5 times.
4	<b>signerPassword.. ..NeedsChange- OfInitPassword</b>	If <b>true</b> a signer needs to change its initial password which was sent by email after login at <a href="http://www.sign-me.de">www.sign-me.de</a> . Afterwards the signer can continue any outstanding identification process, if any (prio 5), or continue with priority 6.
5	<b>activeIdentityVerfi.. ..cationProcessURL</b>	If <b>not null</b> a signer needs to complete an outstanding identity verification process. This can be triggered by redirect in the same session as shown in the <code>FastTrackSample.java</code> or after login at <a href="http://www.sign-me.de">www.sign-me.de</a> . Afterwards signing can start in same session, if required.
6	<b>signerNeeds.. ..IdentificationAnd- TokenCreation</b>	If <b>true</b> a signer needs to be identified by creation of an identity verification process. This can be created by the partner and signer can be redirected to it in the same session as shown in the <code>FastTrackSample.java</code> . Afterwards signing can start in same session, if required.

Table 5.31: Prioritized Check/Action Sequence of Signer Status

NOTE: When testing on the reference system, use <https://cloud-ref.sign-me.de/signature/start> instead of [www.sign-me.de](http://www.sign-me.de). E.g. in item 3 above, if the signer status is queried from the reference system and it says "signer password needs reset", this password reset needs to be done on the reference system, too.

## Summary of SignerStatus

Field Name	Field Type	Occurrence
signingPossible	boolean	1
signerExisting	boolean	1
signerIdentityId	ObjectId	0..1
signerEmailAddressNeedsConfirmation	boolean	1
signerNeedsToAgreeToTermsAndConditions	boolean	1
signerPasswordNeedsReset	boolean	1
signerPasswordNeedsChangeOnceInitPassword	boolean	1
activeIdentityVerificationProcessId	ObjectId	0..1
activeIdentityVerificationProcessURL	IdentityVerificationProcessURL	0..1
signerNeedsIdentificationAndTokenCreation	boolean	1
identityVerificationStateSufficient	boolean	1
currentIdentityVerificationState	IdentityVerificationState	0..1
minimumIdentityVerificationState	IdentityVerificationState	1
validTokenExisting	boolean	1
tokenId	ObjectId	0..1
onDemandTokenCreationPossible	boolean	1
onDemandTokenActivationPossibleForType	boolean	1
onDemandTokenActivationRequiredForQualifiedForIdentity	boolean	1
identificationStillValidForTokenCreation	boolean	1
identificationValidityEndDate	dateTime	0..1
subscriptionActive	boolean	0..1
coinBalanceSufficient	boolean	1
activeSignatureProcessIds	ObjectId	0..N
finalSignatureProcessIds	ObjectId	0..N
validAtTimestamp	dateTime	1
minimumValidityDurationSec	integer	1

Table 5.32: Fields of type SignerStatus

## Details of SignerStatus

Field	<b>signingPossible</b>
Type	boolean [see <a href="#">XSD</a> ]
Occurrence	1
Meaning	<b>Global Status:</b> If true, signing is possible: The partner can create the signature process and the user with "signerUsername" can sign the content based on its current state in the system. Otherwise, signing is not possible, and following details need to be investigated.

Field	<b>signerExisting</b>
Type	boolean [see <a href="#">XSD</a> ]
Occurrence	1
Meaning	<b>Detailed Status:</b> If false, this is the <b>1st priority</b> reason that signing is not possible. In this case, first thing must be to register the user in the system. Afterwards identification is required.

Field	<b>signerIdentityId</b>
Type	ObjectId [see <a href="#">6.32</a> ]
Occurrence	0..1
Meaning	<b>Informational Only:</b> the shortened identityId of the signer, if the signer is existing. Only for checking with other calls and/or logging on the client software side.

Field	<b>signerEmailAddressNeedsConfirmation</b>
Type	boolean [see <a href="#">XSD</a> ]
Occurrence	1
Meaning	<b>Detailed Status:</b> If true, this is the <b>2nd priority</b> reason that signing is not possible. In this case, first the user must confirm the account creation by clicking on the URL sent to his e-mail account. Afterwards, an identification is possible and required for making signing possible.

Field	<b>signerNeedsToAgreeToTermsAndConditions</b>
Type	boolean [see <a href="#">XSD</a> ]
Occurrence	1
Meaning	<b>Informational only:</b> If true, the signer cannot sign, but this item will be solved the next time the signer logs in or before identification. Then the signer will be asked for agreement to terms and conditions.

Field	<b>signerPasswordNeedsReset</b>
Type	boolean [see <a href="#">XSD</a> ]
Occurrence	1
Meaning	<b>Detailed Status:</b> If true, this is the <b>3rd priority</b> reason that signing is not possible. In this case, first the user must reset its password by means of the option in the login form of sign-me. Then the user must confirm the reset by clicking on the URL sent to his e-mail account. Afterwards, changing the initial password is required.

Field	<b>signerPasswordNeedsChangeOfInitPassword</b>
Type	boolean [see <a href="#">XSD</a> ]
Occurrence	1
Meaning	<b>Detailed Status:</b> If true, this is the <b>4th priority</b> reason that signing is not possible. In this case, the user must complete the password reset process and change the initial password in the password reset form or, if this was lost, in the self-service area in password settings of sign-me. Afterwards a new identification is required for security reasons.

Field	<b>activeIdentityVerificationProcessId</b>
Type	ObjectId [see <a href="#">6.32</a> ]
Occurrence	0..1
Meaning	<b>Detailed Status:</b> if not empty this ID of the active identity verification process and needs to be followed upon by the user as <b>priority 5</b> . The active identity verification process is currently a process with identity verification state LAWFUL_VERIFICATION_REQUESTED or LAWFUL_VERIFICATION_STARTED only which are created by the partner organization. Processes created by other partners are not visible.

Field	<b>activeIdentityVerificationProcessURL</b>
Type	IdentityVerificationProcessURL [see <a href="#">6.22</a> ]
Occurrence	0..1
Meaning	<b>Detailed Status:</b> the URL for the process of ID activeIdentityVerificationProcessId above.

Field	<b>signerNeedsIdentificationAndTokenCreation</b>
Type	boolean [see <a href="#">XSD</a> ]
Occurrence	1
Meaning	<b>Detailed Status:</b> If false, this is <b>priority 6</b> reason that signing is not possible. In this case, an identification of the user or only the final step of token creation is required. In any case this is triggered by means of API request createIdentityVerificationProcess or in self-service UI.

Field	<b>identityVerificationStateSufficient</b>
Type	boolean [see <a href="#">XSD</a> ]
Occurrence	1
Meaning	<b>Informational Only:</b> If false, an identification of the user is required (by means of API using createIdentityVerificationProcess or self-service UI).

Field	<b>currentIdentityVerificationState</b>
Type	IdentityVerificationState [see <a href="#">6.23</a> ]
Occurrence	0..1
Meaning	<b>Informational Only:</b> the current identity verification state of the signer, if the signer is existing. Only for checking with other calls and/or logging on the client software side.

<b>Field</b>	<b>minimumIdentityVerificationState</b>
<b>Type</b>	<b>IdentityVerificationState</b> [see 6.23]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	<b>Informational Only:</b> the minimum identity verification state of the signer, if the signer was existing and wanted to sign with signatureType. Only for checking with other calls and/or logging on the client software side.

<b>Field</b>	<b>validTokenExisting</b>
<b>Type</b>	<b>boolean</b> [see [XSD]]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	<b>Informational Only:</b> If false, a signature token needs to be created before signing. In case the onDemand method is not feasible, re-creating of valid signature tokens is needed also through triggering an identification (by means of API using createIdentityVerificationProcess or self-service UI).

<b>Field</b>	<b>tokenId</b>
<b>Type</b>	<b>ObjectId</b> [see 6.32]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	<b>Informational Only:</b> the shortened tokenId of the valid token of the signer, if the token is existing. Only for checking with other calls and/or logging on the client software side.

<b>Field</b>	<b>onDemandTokenCreationPossible</b>
<b>Type</b>	<b>boolean</b> [see [XSD]]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	<b>Informational Only:</b> If this is false re-creating of valid signature tokens is not needed also through triggering an identification but will be done on-demand before signing.

<b>Field</b>	<b>onDemandTokenActivationPossibleForType</b>
<b>Type</b>	<b>boolean</b> [see [XSD]]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	<b>Informational Only:</b> If true on-demand mode for token activation is possible at all for signature type.

<b>Field</b>	<b>onDemandTokenActivationRequiredForQualifiedForIdentity</b>
<b>Type</b>	<b>boolean</b> [see [XSD]]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	<b>Informational Only:</b> If true on-demand mode for signer is required for level qualified.

<b>Field</b>	<b>identificationStillValidForTokenCreation</b>
<b>Type</b>	<b>boolean</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	<b>Informational Only:</b> If true the identification is still valid for token creation. Otherwise, a new identification is needed (by means of API using createIdentityVerificationProcess or self-service UI).

<b>Field</b>	<b>identificationValidityEndDate</b>
<b>Type</b>	<b>dateTime</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	<b>Informational Only:</b> the current date when the validity of the current identification of the signer ends. If this date is expired, the signer can still sign with valid tokens and certificates, but new tokens cannot be created. In this case a new identification needs to be performed. Only present, if the signer exists. Only for checking with other calls and/or logging on the client software side.

<b>Field</b>	<b>subscriptionActive</b>
<b>Type</b>	<b>boolean</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	<b>Informational Only:</b> If this is present and true a subscription is active.

<b>Field</b>	<b>coinBalanceSufficient</b>
<b>Type</b>	<b>boolean</b> [see <a href="#">XSD</a> ]]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	<b>Detailed Status:</b> If this is false a top-up of the coin balance is required to sign. This is only relevant for prepaid billing of signatures (top-up is possible at <a href="http://www.sign-me.de">www.sign-me.de</a> ).

<b>Field</b>	<b>activeSignatureProcessIds</b>
<b>Type</b>	<b>ObjectId</b> [see <a href="#">6.32</a> ]]
<b>Occurrence</b>	<b>0..N</b>
<b>Meaning</b>	<b>Detailed Status:</b> the IDs of active signature processes, which need to be followed upon by the user. These are currently processes with signature state REQUESTED only which are created by the partner organization doing the status query. Processes created by other partners are not visible. The list is syntactically unbounded but will be limited by the system to practical limits (currently 100) and by the parameter maxActiveSignatureProcesses which ever is lower.

<b>Field</b>	<b>finalSignatureProcessIds</b>
<b>Type</b>	<b>ObjectId</b> [see 6.32]
<b>Occurrence</b>	<b>0..N</b>
<b>Meaning</b>	<b>Detailed Status:</b> the IDs of final signature for lookup in the history (as long as they are stored). These are processes with signature state unequal REQUESTED and which are created by the partner organization doing the status query. Processes created by other partners are not visible. The list is syntactically unbounded but will be limited by the system to practical limits (currently 100) and by the parameter maxFinalSignatureProcesses which ever is lower.

<b>Field</b>	<b>validAtTimestamp</b>
<b>Type</b>	<b>dateTime</b> [see [XSD]]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	<b>Informational Only:</b> The signer status is valid date/time at this point in time.

<b>Field</b>	<b>minimumValidityDurationSec</b>
<b>Type</b>	<b>integer</b> [see [XSD]]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	<b>Informational Only:</b> The signer status is valid at validAtTimestamp and then for at least minimumValidityDurationSec.

## 5.32 Subscription

The current information about the subscription of an identity.

### Summary of Subscription

Field Name	Field Type	Occurrence
subscriptionProcessId	ObjectId	1
subscriptionState	SubscriptionState	1
startTimestamp	dateTime	1
endTimestamp	dateTime	1
duration	int	1
enforceAppUse	boolean	0..1

Table 5.33: Fields of type Subscription

### Details of Subscription

<b>Field</b>	<b>subscriptionProcessId</b>
<b>Type</b>	<b>ObjectId</b> [see 6.32]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The process id of the subscription activation or last change for this identity just for reference purposes.

<b>Field</b>	<b>subscriptionState</b>
<b>Type</b>	<b>SubscriptionState</b> [see <a href="#">6.47</a> ]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The state of a subscription during the current point in time.

<b>Field</b>	<b>startTimestamp</b>
<b>Type</b>	<b>dateTime</b> [see <a href="#">[XSD]</a> ]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The date/time when the subscription starts to be active.

<b>Field</b>	<b>endTimestamp</b>
<b>Type</b>	<b>dateTime</b> [see <a href="#">[XSD]</a> ]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The date/time when the subscription terminates to be active.

<b>Field</b>	<b>duration</b>
<b>Type</b>	<b>int</b> [see <a href="#">[XSD]</a> ]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The cumulative duration in number of full calendar months where the subscription shall be active. Equivalent with time between startTimestamp and endTimestamp. Date/time of beginning of Month is based on MET/MEST.

<b>Field</b>	<b>enforceAppUse</b>
<b>Type</b>	<b>boolean</b> [see <a href="#">[XSD]</a> ]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If true, the subscription requires use of the sign-me 2FA app for authorizing signatures.

## 5.33 Token

Token type documentation



## Summary of Token

Field Name	Field Type	Occurrence
tokenId	ObjectId	1
ownerIdentityId	ObjectId	0..1
activationProcessId	ObjectId	0..1
activationProcess	TokenActivationProcess	0..1
tokenAdministrativeState	AdministrativeState	1
tokenName	UTF8String80	0..1
tokenTypeDescription	UTF8String256	1
tokenDetails	UTF8String4096	0..1
tokenVersion	int	0..1
tokenCapabilitySet	TokenCapability	1..8
defaultToken	boolean	1
authTokenType	AuthenticationTokenType	0..1
certificateId	ObjectId	0..1
certificate	Certificate	0..1

Table 5.34: Fields of type Token

## Details of Token

<b>Field</b>	<b>tokenId</b>
<b>Type</b>	ObjectId [see 6.32]
<b>Occurrence</b>	1
<b>Meaning</b>	The sign-me object id of a token used to get this descriptor over the sign-me API.

<b>Field</b>	<b>ownerIdentityId</b>
<b>Type</b>	ObjectId [see 6.32]
<b>Occurrence</b>	0..1
<b>Meaning</b>	<p>The object ID of the owner identity of the token. May not be present, if retention time of process object is over, or if access rights are missing. This is only a shortened identityId, which is enough for reference, but not valid for getting all identity attributes by means of getIdentity(identityId).</p> <p>The full set of attributes of an identity can be retrieved using getIdentity(identityId) after a successful registration, identification, or signature.</p>

<b>Field</b>	<b>activationProcessId</b>
<b>Type</b>	ObjectId [see 6.32]
<b>Occurrence</b>	0..1
<b>Meaning</b>	The object ID of the activation process which was used to activate the token. May not be present, if retention time of process object is over, or if access rights are missing.

<b>Field</b>	<b>activationProcess</b>
<b>Type</b>	<b>TokenActivationProcess</b> [see 5.34]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The complete activation process (descriptor) which was used to activate the token, if required. May not be present, if not required within the query, retention time of process object is over, or if access rights are missing.

<b>Field</b>	<b>tokenAdministrativeState</b>
<b>Type</b>	<b>AdministrativeState</b> [see 6.3]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	If the token is locked, the token cannot be used for any of its purposes any more.

<b>Field</b>	<b>tokenName</b>
<b>Type</b>	<b>UTF8String80</b> [see 6.62]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The name of the token.

<b>Field</b>	<b>tokenTypeDescription</b>
<b>Type</b>	<b>UTF8String256</b> [see 6.56]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	Describes the token type in readable English.

<b>Field</b>	<b>tokenDetails</b>
<b>Type</b>	<b>UTF8String4096</b> [see 6.60]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The token details.

<b>Field</b>	<b>tokenVersion</b>
<b>Type</b>	<b>int</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The version of the token.

<b>Field</b>	<b>tokenCapabilitySet</b>
<b>Type</b>	<b>TokenCapability</b> [see 6.50]
<b>Occurrence</b>	<b>1..8</b>
<b>Meaning</b>	The set of capabilities of the token, e.g. "SIGNATURE_ADVANCED".

<b>Field</b>	<b>defaultToken</b>
<b>Type</b>	<b>boolean</b> [see <a href="#">XSD</a> ]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	This shows, whether the token is the default token for the capabilities of tokenCapabilitySet. The default token is used during token application (e.g. signature), if no other token is explicitly indicated. E.g., if a basic signature shall be applied using an signature type for basic signature, the default token with "SIGNATURE_BASIC" in tokenCapabilitySet will be used.

<b>Field</b>	<b>authTokenType</b>
<b>Type</b>	<b>AuthenticationTokenType</b> [see <a href="#">6.5</a> ]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	If this token has authentication capability, this identifies the specific type of the authentication token, which can be used for login.

<b>Field</b>	<b>certificateId</b>
<b>Type</b>	<b>ObjectId</b> [see <a href="#">6.32</a> ]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The certificate which is used to certify the public key of the token, if there is any. If the token does not have a public key, there is no certificateId.

<b>Field</b>	<b>certificate</b>
<b>Type</b>	<b>Certificate</b> [see <a href="#">5.5</a> ]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The complete certificate which is used to certify the public key of the token, if there is any and it was required as part of the query. If the token does not have a public key, there is no certificate.

## 5.34 TokenActivationProcess

The token activation process is used to track the lifecycle of tokens for specific clients. Partners can rely on the fact, that default signature tokens are always available, which correspond to the identity verification state of the user identities (Persons).

## Summary of TokenActivationProcess

Field Name	Field Type	Occurrence
activationProcessId	ObjectId	1
dTrustAntragsID	DTrustAntragsID	1
activationProcessURL	TokenActivationProcessURL	0..1
organizationId	ObjectId	1
requesterIdentityId	ObjectId	1
redirectURLs	RedirectURLs	0..1
createdTimestamp	dateTime	0..1
activatedTimestamp	dateTime	0..1
certificatePublishedTimest..	dateTime	0..1
..amp		
certificateRevokedTimestamp	dateTime	0..1
revokeReason	UTF8String256	0..1
reservationEndTimestamp	dateTime	1
activationProcessRetrievab..	dateTime	1
..leEndTimestamp		
activationState	ActivationState	1
certificateId	ObjectId	0..1
tokenId	ObjectId	0..1
certificateTemplateId	ObjectId	1

Table 5.35: Fields of type TokenActivationProcess

## Details of TokenActivationProcess

Field	activationProcessId
Type	ObjectId [see 6.32]
Occurence	1
Meaning	The object id of the token activation process received by calls to the sign-me system.

Field	dTrustAntragsID
Type	DTrustAntragsID [see 6.11]
Occurence	1
Meaning	The readable short ID of this token activation process.

Field	activationProcessURL
Type	TokenActivationProcessURL [see 6.49]
Occurence	0..1
Meaning	The unique URL where the token activation WEB application is reachable for the specific token activation process. Can be empty/not present in case of certificate types without further user interaction.

<b>Field</b>	<b>organizationId</b>
<b>Type</b>	<b>ObjectId</b> [see 6.32]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The id of the organization which created this token activation process.

<b>Field</b>	<b>requesterIdentityId</b>
<b>Type</b>	<b>ObjectId</b> [see 6.32]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	<p>The sign-me object id of the identity for which the token activation process was created. Only this identity can activate a token via the authorize and reportUserConfirmation operations.</p> <p>This is only a shortened identityId, which is enough for reference, but not valid for getting all identity attributes by means of getIdentity(identityId).</p> <p>The full set of attributes of an identity can be retrieved using getIdentity(identityId) after a successful registration, identification, or signature.</p>

<b>Field</b>	<b>redirectURLs</b>
<b>Type</b>	<b>RedirectURLs</b> [see 5.24]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	This can contain URLs to redirect to in case of "OK"-, "FAIL"-, and "CANCEL"-results.

<b>Field</b>	<b>createdTimestamp</b>
<b>Type</b>	<b>dateTime</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The date/time when the activation process was created.

<b>Field</b>	<b>activatedTimestamp</b>
<b>Type</b>	<b>dateTime</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The date/time when new tokens and certificates were created.

<b>Field</b>	<b>certificatePublishedTimestamp</b>
<b>Type</b>	<b>dateTime</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The date/time when the certificates were published on OCSP and optionally LDAP.

<b>Field</b>	<b>certificateRevokedTimestamp</b>
<b>Type</b>	<b>dateTime</b> [see [XSD]]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The date/time when the certificates were revoked.

<b>Field</b>	<b>revokeReason</b>
<b>Type</b>	<b>UTF8String256</b> [see 6.56]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The reason for revoking certificates activated under this activation process.

<b>Field</b>	<b>reservationEndTimestamp</b>
<b>Type</b>	<b>dateTime</b> [see [XSD]]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The date/time until the reservation is valid and the activation of a token/creation of a certificate can be performed (activationState=RESERVED or OK_UNCONFIRMED). This time is depending on the certificate type.

<b>Field</b>	<b>activationProcessRetrievableEndTimestamp</b>
<b>Type</b>	<b>dateTime</b> [see [XSD]]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The date/time until positive or partly positive activation process results can retrieved without being changed any more (activationState=OK_UNCONFIRMED* and OK_CONFIRMED), except being revoked over this API. This time is depending on the certificate type. Cancelled or time-out activation processes will be removed earlier than this timestamp.

<b>Field</b>	<b>activationState</b>
<b>Type</b>	<b>ActivationState</b> [see 6.2]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The current state of the token activation process.

<b>Field</b>	<b>certificateId</b>
<b>Type</b>	<b>ObjectId</b> [see 6.32]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The sign-me object id of the certificate which was created as part of the activation process.

<b>Field</b>	<b>tokenId</b>
<b>Type</b>	<b>ObjectId</b> [see 6.32]
<b>Occurrence</b>	<b>0..1</b>
<b>Meaning</b>	The sign-me object id of the token which was created/activated as part of the activation process.

<b>Field</b>	<b>certificateTemplateId</b>
<b>Type</b>	<b>ObjectId</b> [see 6.32]
<b>Occurrence</b>	<b>1</b>
<b>Meaning</b>	The sign-me object id of the certificate template used to create the certificate. The certificate template corresponds to the certificate type which was ordered to be activated in this process.

## 5.35 VersionAndDate

Contains version and date of the interface.

### Summary of VersionAndDate

Field Name	Field Type	Occurrence
version	string	1
revision	IfRevision	0..1
date	string	1

Table 5.36: Fields of type VersionAndDate

### Details of VersionAndDate

Field	version
Type	string [see <a href="#">XSD</a> ]
Occurrence	1
Meaning	Version consists of a major.minor version specinfo.version field in the WSDL file. Sample version string is: "2.8".

Field	revision
Type	IfRevision [see <a href="#">6.25</a> ]
Occurrence	0..1
Meaning	The revision specinfo.date field in the WSDL file.

Field	date
Type	string [see <a href="#">XSD</a> ]
Occurrence	1
Meaning	The date string specinfo.date field in the WSDL file.

## 6 Simple Datatypes of the API

### 6.1 AccessToken

Base type: **string** [see [XSD](#)]; length: **0..128**

The access token which can be got through Login/LoginStep2 response. May be renewed by Authorize/AuthorizeStep2 or Refresh operations as part of the [5.1](#).

### 6.2 ActivationState

Enumeration base type: **string** [see [XSD](#)]

The specification of states of a token activation process.

ActivationState	Meaning
<b>RESERVED</b>	The token activation was reserved by a partner application.
<b>OK_UNCONFIRMED</b>	The token activation was authorized by the user, but is not confirmed to be published on OCSP/LDAP.
<b>OK_CONFIRMED</b>	The token activation was authorized by the user and confirmed to be published on OCSP/LDAP.
<b>NOK_TIMEOUT</b>	The token activation was not authorized by the user before time-out.
<b>NOK_ERROR</b>	The token activation was not authorized by the user, because a hard error occurred.
<b>OK_UNCONFIRMED_- TIMEOUT</b>	The token activation was authorized by the user, but not confirmed to be published on OCSP/LDAP before time-out.
<b>OK_UNCONFIRMED_ER- ROR</b>	The token activation was authorized by the user, but not confirmed to be published on OCSP/LDAP, because a hard error occurred.
<b>CANCELLED</b>	The token activation was not authorized by the user, because cancelled by the partner portal/shop.
<b>CERTIFICATE_REVOKED</b>	The certificate, which was successfully published on OCSP/LDAP before, was revoked by the partner portal/shop or user.
<b>CERTIFICATE_EXPIRED</b>	The certificate, which was successfully published on OCSP/LDAP is expired.

Table 6.1: Values of enumeration type ActivationState

### 6.3 AdministrativeState

Enumeration base type: **string** [see [XSD](#)]



The administrative state controls whether the object can be used or is prohibited for use. This state is readonly and can only be changed as an administrative action.

AdministrativeState	Meaning
UNLOCKED	The resource is open for usage.
SHUTTING_DOWN	The resource completes existing transactions, but is not open for new transactions.
LOCKED	The resource is not open for usage.

Table 6.2: Values of enumeration type AdministrativeState

## 6.4 AuthenticationQualityLevel

Enumeration base type: **string** [see [XSD](#)]

The authentication quality level reached in a login session. The quality level reachable depends on the token capabilities.

AuthenticationQualityLevel	Meaning
INITIAL	Will be reached after initial, successful Login or Authorize.
ONE_FACTOR	Will be reached after successful LoginStep2 or AuthorizeStep2, if the token has capability AUTHENTICATION (see TokenCapability description in section 6.50).
TWO_FACTOR	Will be reached after successful LoginStep2 or AuthorizeStep2, if the token has capability AUTHENTICATION_WITH_CLIENT_CHALLENGE (see TokenCapability description in section 6.50) or two different authentication tokens were used for Login and/or Authorize.
MULTI_FACTOR	Will be reached after more than two different authentication tokens were used for Login and/or Authorize.

Table 6.3: Values of enumeration type AuthenticationQualityLevel

## 6.5 AuthenticationTokenType

Base type: **string** [see [XSD](#)]; length: **3..20**

Currently the only possible values are: "UIDPW" or "UIDEMAIL" for Login and 1-factor authentication as part of authorizeProcess, and "AC" for 2-factor authentication as part of authorizeProcess for qualified/advanced electronic signing, and "SIDTK" for 2-factor authentication as part of authorizeProcess for qualified electronic sealing.

## 6.6 BCP47LanguageTag

Base type: **string** [see [XSD](#)]; length: **2..5**

The language tag specified according to IETF BCP 47 standard as language code (ISO639) and optionally region (2 character country code acc. to ISO3166-1 alpha-2). Other elements of IETF BCP 47 are not supported.

Samples: "de", "en", "fr", "de-DE" , "en-US" , "fr-FR". Other components may be used, but there is no guarantee, that the language tag will be supported. Otherwise a default language tag will be assigned.

## 6.7 CertificateType

Base type: **string** [see [XSD](#)]; length: **0..32**

This specifies the type of certificate which shall be created. In case of certificates for server signature or sealing tokens (the only possibility in sign-me with API V2) the certificate type implicitly defines some properties of the private and public key, which is hosted as a server side token in a HSM.

This string is either defining one of several default certificate types which will be available on the reference and live system.

Or, it is a project specific signature agreed upon in a project. The project specific values are out of scope of this specification.

Default certificates are created as part of the default tokens after the identity verification for the required quality level is completed (see [4.13](#) or [4.14](#)).

The default certificate types are the following:

Certificate Type	Purpose
C_BAS_2Y_E_1	A certificate type which certifies a signature token for basic signing. The signature token is a HSM held ECDSA key pair with elliptic curve P-256. The signature certificate has a validity duration of two years. The CA used to certify the public key of the signature token is a basic CA of D-TRUST. The identity verification state of the subject of the certificate of this type is LAWFULLY_VERIFIED_HIGH (see 6.6).
C_ADV_2Y_R_1	A certificate type which certifies a signature token for advanced signing. The signature token is a HSM held RSA-PSS key pair. The signature certificate has a validity duration of two years. The CA used to certify the public key of the signature token is an advanced CA of D-TRUST. The identity verification state of the subject of the certificate of this type is LAWFULLY_VERIFIED_HIGH (see 6.6).
C_QES_2Y_R_1	A certificate type which certifies a signature token for qualified signing. The signature token is a HSM held RSA-PSS key pair. The signature certificate has a validity duration of two years. The CA used to certify the public key of the signature token is a qualified CA of D-TRUST. The identity verification state of the subject of the certificate of this type is LAWFULLY_VERIFIED_HIGH (see 6.6) and signer status onDemandTokenActivationRequiredForQualifiedForIdentity=false (see 5.31).
C_QES_24H_R_1	A certificate type which certifies a signature token for qualified signing. The signature token is a HSM held RSA-PSS key pair. The signature certificate has a validity duration of 24 hours. The CA used to certify the public key of the signature token is a qualified CA of D-TRUST. The identity verification state of the subject of the certificate of this type is LAWFULLY_VERIFIED_HIGH (see 6.6) and signer status onDemandTokenActivationRequiredForQualifiedForIdentity=true (see 5.31).

Table 6.4: Values of CertificateType for Default Certificates

## 6.8 Coins

Base type: **integer** [see [XSD]]; length: **0..N**

The number of coins available for consumption by means of signatures. The signatures, which consume coins are of special pre-paid signature types. Coins are re-charged based on identification operations.

## 6.9 ContentViewURL

Base type: **anyURI** [see [XSD]]; length: **0..256**

The URL for visualization of content to be signed in case of digest/hash signing. This URL MUST represent a document which exactly views the content which shall be signed through digest/hash signing.

This parameter must not be present in case of document signing. The URL must have protocols http or https. The URL should locate a document in the local area network, where the signature client is operated.

## 6.10 Currency

Base type: **string** [see [XSD](#)]; length: **0..3**  
Currency type alphanumeric value according to ISO4217

## 6.11 DTrustAntragsID

Base type: **string** [see [XSD](#)]; length: **0..40**  
The unique printable identifier for formal, written communication about the activation process or certificate reservation. This identifier should be presented to the certificate requester in a readable manner on a GUI of the portal or shop.

## 6.12 DateString

Base type: **string** [see [XSD](#)]; length: **0..20**  
The date string in alpha-numeric form. The content can be fully specified by following pattern:

`"yyyyMMdd"`

Or, in case day or month are unspecified, they can be replaced by spaces in the following ways for unspecified days:

`"yyyyMM__"`

or for unspecified days and months:

`"yyyy----"`

(' ' means space).

## 6.13 Duration

Base type: **string** [see [XSD](#)]; length: **3..30**  
Specification of time duration in following units YEARS, MONTHS, DAYS (at least minimum 24h and midnight), DAYS.EXACT, HOURS, MINUTES, SECONDS. Format of the duration string is:

```
duration :=  
number " " unit
```

## 6.14 EmailAddress

Base type: **string** [see [XSD](#)]; length: **0..80**

This is an email address according to RFC822 with additional restrictions on special symbols and with the possibility of RFC3490 domain names.

Only the following 7-Bit ASCII characters are allowed in the user name part of the email address (before the '@'):

```
Letters: A-Z a-z
Digits: 0-9
Punctuation:
.+_-
```

Only the following Unicode characters are allowed in the domain name part of the email address (after the '@'):

```
Unicode letters
Unicode
digits
Punctuation: .+_-
```

In the domain part of the email address domain names with additional non-ASCII characters will be converted to International Domain Names before use according to RFC3490 'toASCII' conversion. Restrictions may apply based on the implementation.

## 6.15 ErrorId

Enumeration base type: **string** [see [XSD](#)]

-

ErrorId	Meaning
rejected	The completion of the request was rejected, because some of the required more complex preconditions were not fulfilled by the request parameters or state of the system. E.g., an activation of a new QES is not allowed for an activation key, if a successful activation was already performed under this key.
invalidData	The completion of the request was not started, because simple preconditions were not fulfilled by the request parameters or state of the system. E.g., an activation of a new QES is not allowed for an activation key, if the activation key is not known in the system. Another example is, the date of birth was not specified in the certification request.
processingError	The processing of the request was aborted by an unexpected error.
other	The processing of the request was aborted by any other error.

Table 6.5: Values of enumeration type ErrorId

## 6.16 ExternalProcessKey

Base type: **string** [see [XSD](#)]; length: **0..128**

The optional key for an external service for identification. Normally, except for specific partners, only relevant for support purposes.

## 6.17 ExternalProcessURL

Base type: **anyURI** [see [XSD](#)]; length: **0..1024**

The optional link for an external service for identification. Normally, except for specific partners, only relevant for support purposes.

## 6.18 GenericAuthenticationData

Base type: **tns:MediumBlob** [see [XSD](#)]; length: **0..N**

### UIDPW Authentication-Token

Parameters of this datatype contain the SHA-512 hash of the password for the user ID which will be used for standard Username/Password logins. The hash operation is as follows ('+' means byte concatenation): pwHash:=SHA-512(salt1+password).

Normally, clients do not need to be concerned about this operation, because there are login helper functions in the language and platform specific access modules, which do a complete token specific login including salting and hashing.

## Other Tokens (Future Releases)

The authentication tokens or IDs of authentication tokens, typically hash values of some sort, which were registered with the registration service. The number and format of any authenticationIds is token type specific.

## 6.19 HugeBlob

Base type: **base64Binary** [see [XSD](#)]; length: **0..16777216**

Huge blob is a byte array of maximum size of 16MB before Base64 encoding.

## 6.20 ICAOCountryName

Base type: **string** [see [XSD](#)]; length: **1..3**

The country code according to ICAO as a 3 character code, and in some exceptional cases single character code (e.g. 'D' for Germany).

The list is from ICAO Doc 9303 (6th Ed. 2006) Part 1 Vol 1 Appendix 7 Part A. This in turn references ISO 3166-1 alpha-3 (3 character) code specification.

For the complete list of allowed values at the sign-me API see section [8](#).

## 6.21 IdentType

Base type: **string** [see [XSD](#)]; length: **0..32**

General name grouping the properties and settings for a given identification procedure

## 6.22 IdentityVerificationProcessURL

Base type: **anyURI** [see [XSD](#)]; length: **0..256**

The unique and persistent URL for a identification verification process at the sign-me WEB application front-end. The WEB application will give access to the specific verification process in the sign-me system. The link is only valid until the date/time given in 'reservationEndTimestamp'.

## 6.23 IdentityVerificationState

Enumeration base type: **string** [see [XSD](#)]

The verification state of identities. This state describes the quality level which is achieved after verification of the identity. Based on this state signing or token activation operations are possible for different types of certificates (see section [6.7](#)) and signatures (see section [6.43](#)).

<b>IdentityVerificationState</b>	<b>Meaning</b>
<b>UNVERIFIED</b>	An syntactically correct email address is available. Signing is not possible in this state.
<b>COMMUNICATION_VERIFIED</b>	In addition to UNVERIFIED the email addr. is verified after the registration process.
<b>LAWFUL_VERIFICATION_REQUESTED</b>	In addition to COMMUNICATION_VERIFIED a verification of the identity is requested by a partner.
<b>LAWFUL_VERIFICATION_STARTED</b>	The verification of the identity was started with the user. If the verification is successfully, the state changes to LAWFULLY_VERIFIED_HIGH. In case of finally unsuccessful verification we change to COMMUNICATION_VERIFIED. Signing is still not possible in this state.
<b>LAWFULLY_VERIFIED</b>	Deprecated.
<b>LAWFULLY_VERIFIED_HIGH</b>	The verification of the identity is successfully completed. This state is sufficient to create server signature tokens of "SIGNATURE_BASIC", "SIGNATURE_ADVANCED", and "SIGNATURE_QUALIFIED" including basic, advanced, and qualified certificates for the verified identity.
<b>CANCELLED</b>	The verification of the identity was cancelled. If the identity to be verified was in any states of LAWFUL_VERIFICATION_REQUESTED or LAWFUL_VERIFICATION_STARTED, then the identity state will be reset to COMMUNICATION_VERIFIED. Signing is not possible in this state.
<b>ERROR</b>	The verification of the identity was not completed because of error. The same rules for reset of the identity state as in CANCELLED apply. Signing is not possible in this state.
<b>TIMEOUT</b>	The verification of the identity was not completed before timeout. The same rules for reset of the identity state as in CANCELLED apply. Signing is not possible in this state.
<b>AWAITING_SANCTIONLST_SEARCH</b>	Effectively Same state LAWFUL_VERIFICATION_STARTED.
<b>AWAITING_SANCTIONLST_PROT_REVIEW</b>	Same state like LAWFUL_VERIFICATION_STARTED.
<b>SANCTIONLST_CHECKED_NEGATIVE</b>	Same state like ERROR.
<b>SANCTIONLST_ERROR</b>	Same state like ERROR.

Table 6.6: Values of enumeration type IdentityVerificationState

## 6.24 IdentityVerificationType

Base type: **string** [see [XSD](#)]; length: **0..32**



The type of identity verification which shall be used. There are some standard types and project specific types will be agreed with D-TRUST.

**Standard Types on Production System** The standard identity verification types are the following:

Type	Description	URL valid	Fast-track
<b>VIDEO_4EYE</b>	There is a web session under the unique IdentityVerificationProcessURL (see 6.22) where a video identification is taking place. The video identification allows the creation of ad-hoc short-term certificates for QES before signing. This is the default identification type.	30 Days	no
<b>WEB_FT_-IDENT_4EYE</b>	Like <b>VIDEO_4EYE</b> in fasttrack mode. That means that the confirmation of the e-mail address is not required during the web session, because the e-mail address is guaranteed to be verified by the partner.	1 Day	yes
<b>POS_FT_-IDENT_4EYE</b>	Point of Sale identification in fasttrack mode. That means that the confirmation of the e-mail address is not required during the web session, because the e-mail address is guaranteed to be verified by the partner. There are some preconditions like on-premise equipment which need to be specified on a project specific basis. This type of identification allows the creation of long-term certificates for QES immediately after identification.	1 Day	yes
<b>E_ID</b>	There is a web session under the unique IdentityVerificationProcessURL (see 6.22) where a eID using German ID card is taking place. If there is problem with equipment or otherwise, a fallback to a video identification is possible. The eID identification allows the creation of long-term certificates for QES immediately after identification, if the video fallback was not used.	30 Days	no
<b>FT_E_ID</b>	Like <b>E_ID</b> in fasttrack mode. That means that the confirmation of the e-mail address is not required during the web session, because the e-mail address is guaranteed to be verified by the partner.	1 Day	yes

Table 6.7: Standard values of IdentityVerificationType for production system (part 1)

Type	Description	URL valid	Fast-track
<b>FT_AW_ID</b>	There is a web session under the unique IdentityVerificationProcessURL (see 6.22) where an eID using German ID card is taking place. Very comfortable implementation with desktop (Windows, Apple MacOS) and mobile support (Android and iOS). The eID identification allows the creation of long-term certificates for QES immediately after identification. Uses fasttrack mode. That means that the confirmation of the e-mail address is not required during the web session, because the e-mail address is guaranteed to be verified by the partner.	1 Day	yes
<b>FT_TKOV_ID</b>	There is a web session under the unique IdentityVerificationProcessURL (see 6.22) where certificate production will be agreed to by the identified person. The identification data will be transferred as initial data. Precondition: identification data must be sourced from an eIDAS compliant identification suitable for long-term certificates.	1 Hour	yes
<b>FT_TKOV_ID_FROM_VIDEO</b>	Like <b>FT_TKOV_ID</b> , but identification data must be sourced from an eIDAS compliant identification suitable for short-term certificates (e.g. video-ident).	1 Hour	yes

Table 6.8: Standard values of IdentityVerificationType for production system (part 2)

**Standard Types on Reference System** On the reference system the same identity verification types are configured as on the production system. The difference is for types in table part 1 (see table 6.7), without further action, all identification will be simulated on the reference system. There is no actual identification session, moreover all simulation results will be like the result of a successful video identification session. For identity verification types in table part 2 (see table 6.8) there is no simulation on the reference system.

Within simulation, specific identity verification types can be enforced if the academic title of the person to be identified will be set to one of the following values within registration or change of personal data:

- Video
- eID
- Shop (=POS)

By means of that workaround the result of an actual identification can be simulated. Please note that the actual chosen identification can be different from the one initially preset at the API. For several of the identity verification types from the table above, the user can try a different identification, if problems occur. In case of Video the ad-hoc procedure for creating qualified certificates will be selected by sign-me, with all

other identity verification types the usual long-term certificates will be created after identification also in case of the qualified certificate.

It is possible to have real identification sessions also on the reference system. For this to happen, please append a \_T to the identity verification types from the table above (for TRUE). Of course, on the reference system this true identification session only leads to test certificates from a test CA. Use of this possibility shall be restricted to testing the technical integration of the identification mechanisms into web applications of the partner. For system or API tests, the simulated variants shall be preferred.

## 6.25 IfRevision

Base type: **string** [see [XSD](#)]; length: **0..32**

Interface revision consists of an opaque string identifying the revision of the interface for compatibility checks. The string is from specinfo.revision field in the WSDL file. Sample revision string is: "9420".

## 6.26 IpAddressNumeric

Base type: **string** [see [XSD](#)]; length: **0..80**

An IP V4 or V6 address string in numeric form.

## 6.27 ItemName

Base type: **string** [see [XSD](#)]; length: **0..80**

The name of the item which shall be signed and/or verified, e.g. user friendly name or filename of document to be signed.

Allowed Alphanumeric ASCII characters are:

0-9  
a-z  
A-Z

Allowed ASCII special characters within Itemname are <space>and:

\_ . , - ( ) ~ §

Allowed Unicode characters are letters and digits from BMP (U+0000 bis U+FFFF).

## 6.28 LargeBlob

Base type: **base64Binary** [see [XSD](#)]; length: **0..262144**

Large blob is a byte array of maximum size of 256KB before Base64 encoding.

## 6.29 ManageTokenAction

Enumeration base type: **string** [see [XSD](#)]

Possible actions for token management.

ManageTokenAction	Meaning
<b>UPGRADE</b>	Request change of token.
<b>AUTHENTICATE</b>	Multi-factor authentication for token management.
<b>SET_CREDENTIAL</b>	Credential setting for token management.
<b>STATUS</b>	Status query for token management.

Table 6.9: Values of enumeration type ManageTokenAction

## 6.30 ManageTokenStatus

Enumeration base type: **string** [see [XSD](#)]

Possible values for status of token management.

ManageTokenStatus	Meaning
<b>INITIAL</b>	Initial state before any action.
<b>UPGRADE_STARTED</b>	Requested upgrade change of token started (e.g. from TAN to APP handling for AC token type).
<b>UPGRADE_STARTED_-FROM_REGISTERED</b>	Requested upgrade change of token started (e.g. from APP to other APP handling for AC token type).
<b>AUTHENTICATED</b>	Multi-factor authentication for token management succeeded(e.g. hashed/salted TAN delivery).
<b>SET_CREDENTIAL_DONE</b>	Credential setting for token management successful (e.g. hashed/salted PIN setting).
<b>REGISTERED</b>	Sign-up call back received and registered.
<b>FAILED</b>	If any failure occurred in last token management.

Table 6.10: Values of enumeration type ManageTokenStatus

## 6.31 MediumBlob

Base type: **base64Binary** [see [XSD](#)]; length: **0..16384**

Medium blob is a byte array of maximum size of 16KB before Base64 encoding.

## 6.32 ObjectID

Base type: **string** [see [XSD](#)]; length: **0..64**

The unique ID for objects (process objects, tokens, certificates etc.) which were communicated over this interface. The persistence of the ID can be restricted to the login session identified by the accessToken. In other words, if a specific signature token was created or queried in a specific login session, the ID can be different in a follow-up session.

## 6.33 PIN

Base type: **string** [see [XSD](#)]; length: **4..512**

An alphanumeric sequence of UTF-8 characters.

## 6.34 Password

Base type: **string** [see [XSD](#)]; length: **8..512**

The password must consist of at least 1 upper char, 1 lower char, 1 digit, and at least one of the following special characters:

`!+-_*$?:#&@;~%`

In addition a minimum complexity is required for the password. It is recommended to generate the password using a random generator mechanism or use well-known tools like 'KeePass' or similar.

NOTE: The password string needs to be encoded as an UTF-8 string in all configuration files and language specific transformations to avoid any corruption of special characters.

## 6.35 PhoneNumber

Base type: **string** [see [XSD](#)]; length: **6..40**

Valid (mobile) phone number in the MSISDN format "+CountryCode NationalDest-CodeAndSubscriberNumber". Sample:

`+49 9991234567890`

The following format is also accepted, but will be converted to the above format in queries:

`+49(999)1234567890`

Tolerated separators in the subscriber number: part are:

`- _/`

They are tolerated during input, but will be removed during normalization.

## 6.36 PlaceFreertextInfo

Base type: **string** [see [XSD](#)]; length: **0..360**

Information if the exact place is not known, or only known as unstructured free text.

## 6.37 RegistrationProcessURL

Base type: **anyURI** [see [XSD](#)]; length: **0..256**

The unique and persistent link which identifies a registration process at the sign-me WEB application front-end.

The registration process URL is used to start a user specific sequence of web forms at the sign-me web application to guide the user for authorizing the electronic registration.

## 6.38 RegistrationState

Enumeration base type: **string** [see [XSD](#)]

The specification of states of a registration process.

RegistrationState	Meaning
REQUESTED	The registration was requested by the partner shop/portal, and is not completed by the user, yet. The registration can be completed until registrationProcessUsableEndTimestamp.
SUCCESS	The registration was successfully completed. The registration result can be retrieved until registrationProcessRetrievableEndTimestamp.
TIMEOUT	The registration timed out without successful completion. The registration result can be retrieved until registrationProcessRetrievableEndTimestamp.

Table 6.11: Values of enumeration type RegistrationState

## 6.39 RoleName

Base type: **string** [see [XSD](#)]; length: **0..80**

The role name to identify the role used to login. The standard role names to be used by standard partner applications, is "PARTNER". In cases, where the partner application is also application provider for users, the role USER is also relevant. For electronic sealing SEALING\_MANAGER is a valid role name for persons which are sealing managers for a partner organization.

## 6.40 SignatureFormat

Base type: **string** [see [XSD](#)]; length: **0..40**

This specifies the format of the signature (container) which shall be created. This string is either one of several standard types which will be available on the test and live system. Or, it is a project specific signature agreed upon in a project. The project specific values are out of scope of this specification.

The standard signature formats are the following:

SignatureFormat	Purpose
PDF_PADES_BASIC	PDF signature format according to PAdES Profiles Part2: Basic Profile (ETSI TS 102 778-2)
PDF_PADES_ENHANCED	PDF signature format according to PAdES Profiles Part3: Enhanced - PAdES-BES and PAdES-EPES Profiles (ETSI TS 102 778-3)
CMS_BASIC	A basic signature format using a PKCS7 container but without signed attributes
CADES	A standard CADES-B-B signature according to ETSI standard
CADES_T	A standard CADES-B-T signature according to ETSI standard
CADES4PADES	Same CMS profile like CADES, but without PKCS9 signing-time attribute. Embedded into a PDF document this is the CMS basis for the PADES-B-B signature.
CADES4PADES_T	Same CMS profile like CADES4PADES including qualified timestamp token. Embedded into a PDF document this is the CMS basis for the PADES-B-T signature.
CADES4PADES_LT	Same CMS profile like CADES4PADES_T. In addition, after a successful signature of this format, OCSP responses for the used certificate chain are provided in Certificate data type (see 5.5). Embedded into a PDF document this is the CMS basis for the PADES-B-LT signature.
CMS_PKCS7	Same as CADES. Deprecated.
CMS4PDF	Same as CADES4PADES. Deprecated.

Table 6.12: Standard values of SignatureFormat

## 6.41 SignatureProcessURL

Base type: **anyURI** [see [XSD]]; length: **0..256**

The unique and persistent link which identifies a signature process at the sign-me WEB application front-end.

The signature process URL is used to start a user specific sequence of web forms at the sign-me web application to guide the user for authorizing the electronic signature.

To preset the initial login form with a sign-me username, the partner can transmit the username using a query parameter in the URL using the following format:

```
<signatureProcessURL>&USERNAME=<username>
```

- <signatureProcessURL> is the URL as returned by sign-me.
- <username> is the username as given by the sign-me user to the partner application in URL-Encoding.

## 6.42 SignatureState

Enumeration base type: **string** [see [XSD]]

The specification of states of a signature process, resp. its itemsToBeProcessed.

SignatureState	Meaning
<b>REQUESTED</b>	The signature was requested by the partner shop/portal, and is not authorized by the user, yet.
<b>OK_UNVERIFIED</b>	The signature was authorized by the user, but has not been verified by the partner shop/portal so far. This is the successful final state for digest signatures.
<b>OK_VERIFIED</b>	The signature was authorized by the end- user, and has been be verified and is valid by the partner shop/portal. This is the successful final state for document signatures.
<b>NOK_TIMEOUT</b>	The signature was not successfully authorized by the user, and the signature process timed-out (date after signatureRequestEndTimestamp). This is an unsuccessful final state.
<b>NOK_ERROR</b>	The signature was not authorized by the user, and the signature process was terminated based on a hard error situation (e.g. potential security violation). This is an unsuccessful final state.
<b>OK_UNVERIFIED_TIMEOUT</b>	The signature was successfully authorized by the user, but the signature could not be verified before the signature process timed out (date after signatureRequestEndTimestamp). This is an unsuccessful final state.
<b>OK_UNVERIFIED_ERROR</b>	The signature was successfully authorized by the user, but the signature could not be verified because a verification request was terminated based on a hard error situation (e.g. potential security violation detected in the signature). This is an unsuccessful final state.
<b>CANCELLED</b>	The signature process was terminated by the partner portal (before any signature was authorized on the document or digest).

Table 6.13: Values of enumeration type SignatureState

## 6.43 SignatureType

Base type: **string** [see [XSD](#)]; length: **0..32**

This specifies the type of signature which shall be created for electronic signing or sealing. This string is either one of several standard types which will be available on the test and live system.

Or, it is a project specific signature agreed upon in a project. The project specific values are out of scope of this specification.

The standard signature types are the following:



Signature Type	Purpose
<b>S_QES_DOC_PADES_B_B_1</b>	A PDF document signature following PADES B-B format. A token with capability for qualified signing will be used. The sign-me signature field is visible in the PDF document on the lower/right side of the first page by default, showing common name of the used signature certificate. The signature visualization can be adapted using SignatureParameters during CreateSignatureProcess.
<b>S_QES_DOC_PADES_B_T_1</b>	Like <b>S_QES_DOC_PADES_B_B_1</b> but in addition the signature includes a D-TRUST qualified time-stamp.
<b>S_QES_DOC_PADES_B_LT_1</b>	Like <b>S_QES_DOC_PADES_B_B_1</b> but in addition the signature includes a D-TRUST qualified time-stamp and long-term-validation (LTV) information.
<b>S_QES_DOC_PADES_3</b>	Like <b>S_QES_DOC_PADES_B_LT_1</b> , but there is no visible sign-me signature icon in the PDF document.
<b>S_QES_DOC_PADES_5</b>	Like <b>S_QES_DOC_PADES_B_LT_1</b> , but by default the text visualized in the signature icon does not scale (fontSize=9). However, the signature visualization can be adapted using SignatureParameters during CreateSignatureProcess.

Table 6.14: Standard values of SignatureType for PDF document signing with qualified CA

Signature Type	Purpose
<b>S_ADV_DOC_PADES_B_B_1</b>	Technically, the same signature format like S_QES_DOC_PADES_B_B_1, but a token with capability for advanced signing will be used.
<b>S_ADV_DOC_PADES_3</b>	Like <b>S_ADV_DOC_PADES_B_B_1</b> , but there is no visible sign-me signature icon in the PDF document.
<b>S_ADV_DOC_PADES_5</b>	Like <b>S_ADV_DOC_PADES_B_B_1</b> , but by default the text visualized in the signature icon does not scale (fontSize=9). However, the signature visualization can be adapted using SignatureParameters during CreateSignatureProcess.

Table 6.15: Standard values of SignatureType for PDF document signing with advanced CA

Signature Type	Purpose
<b>S_BAS_DOC_PADES_B_B_1</b>	Technically, the same signature format like S_QES_DOC_PADES_B_B_1, but a token with capability for basic signing will be used.
<b>S_BAS_DOC_PADES_3</b>	Like <b>S_BAS_DOC_PADES_B_B_1</b> , but there is no visible sign-me signature icon in the PDF document.
<b>S_BAS_DOC_PADES_5</b>	Like <b>S_BAS_DOC_PADES_B_B_1</b> , but by default the text visualized in the signature icon does not scale (font-Size=9). However, the signature visualization can be adapted using SignatureParameters during CreateSignatureProcess.

Table 6.16: Standard values of SignatureType for PDF document signing with basic CA

Signature Type	Purpose
<b>S_QES_DIG_CMS_BASIC_1</b>	Standard Signature Type for DigestSigning with format CMS_BASIC. A token with capability for qualified signing will be used. The CMS signature contains no signed attributes.
<b>S_QES_DIG_CMS_BASIC_T_1</b>	Like S_QES_DIG_CMS_BASIC_1, but a qualified signature timestamp is included as a CMS unsigned attribute (identified by CaDES id-aa-signatureTimeStampToken OID).
<b>S_QES_DIG_CADES_1</b>	Standard Signature Type for DigestSigning with format standard CADES-B-B. A token with capability for qualified signing will be used.
<b>S_QES_DIG_CADES_T_1</b>	Like S_QES_DIG_CADES_1, but a qualified signature timestamp is included as a CMS unsigned attribute (identified by CaDES id-aa-signatureTimeStampToken OID). This is compliant with the CADES-B-T standard.
<b>S_QES_DIG_-CADES4PADES_1</b>	Standard Signature Type for DigestSigning with format standard CADES4PADES. A token with capability for qualified signing will be used. Same CMS profile like S_QES_DIG_CADES_1, but without PKCS9 signing-time attribute. Embedded into a PDF document this is the CMS basis for the PADES-B-B signature.
<b>S_QES_DIG_-CADES4PADES_T_1</b>	Like S_QES_DIG_CADES4PADES_1, but a qualified signature timestamp is included as a CMS unsigned attribute (identified by CaDES id-aa-signatureTimeStampToken OID). Embedded into a PDF document this is the CMS basis for the PADES-B-T signature.
<b>S_QES_DIG_-CADES4PADES_LT_1</b>	Like S_QES_DIG_CADES4PADES_T_1. In addition, after a successful signature of this format, OSCP responses and/or CRLs for the used certificate chain are provided in Certificate data type (see 5.5). Embedded into a PDF document this is the CMS basis for the PADES-B-LT signature.

Table 6.17: Standard values of SignatureType for digest signing with qualified CA

Signature Type	Purpose
<b>S_ADV_DIG_CMS_BASIC_1</b>	Technically, the same signature format like S.QES_DIG_CMS_BASIC_1, but a token with capability for advanced signing will be used.
<b>S_ADV_DIG_CADES_1</b>	Technically, the same signature format like S.QES_DIG_CADES_1, but a token with capability for advanced signing will be used.
<b>S_ADV_DIG_CADES4PADES_1</b>	Technically, the same signature format like S.QES_DIG_CADES4PADES_1, but a token with capability for advanced signing will be used.

Table 6.18: Standard values of SignatureType for digest signing with advanced CA

Signature Type	Purpose
<b>S_BAS_DIG_CMS_BASIC_1</b>	Technically, the same signature format like S.QES_DIG_CMS_BASIC_1, but a token with capability for basic signing will be used.
<b>S_BAS_DIG_CADES_1</b>	Technically, the same signature format like S.QES_DIG_CADES_1, but a token with capability for basic signing will be used.
<b>S_BAS_DIG_CADES4PADES_1</b>	Technically, the same signature format like S.QES_DIG_CADES4PADES_1, but a token with capability for basic signing will be used.

Table 6.19: Standard values of SignatureType for digest signing with basic CA

Signature Type	Purpose
<b>S_QESEAL_DIG_CMS_- BASIC_1</b>	Standard Signature Type for qualified electronic digest sealing with format CMS_BASIC. A token with capability for qualified electronic sealing will be used. The CMS signature contains no signed attributes.
<b>S_QESEAL_DIG_CMS_- BASIC_T_1</b>	Like S_QESEAL_DIG_CMS_BASIC_T_1 for qualified electronic sealing with format CMS_BASIC, but a qualified signature timestamp is included as a CMS unsigned attribute (identified by CaDES id-aa-signatureTimeStampToken OID).
<b>S_QESEAL_DIG_- CADES_1</b>	Standard Signature Type for qualified electronic digest sealing with format standard CADES-B-B. A token with capability for qualified electronic sealing will be used.
<b>S_QESEAL_DIG_- CADES_T_1</b>	Like S_QESEAL_DIG_CADES_1, but a qualified signature timestamp is included as a CMS unsigned attribute (identified by CaDES id-aa-signatureTimeStampToken OID). This is compliant with the CADES-B-T standard.
<b>S_QESEAL_DIG_- CADES4PADES_1</b>	Standard Signature Type for qualified electronic digest sealing with format standard CADES4PADES. A token with capability for qualified electronic sealing will be used. Same CMS profile like S_QESEAL_DIG_CADES_1, but without PKCS9 signing-time attribute. Embedded into a PDF document this is the CMS basis for the PADES-B-B signature.
<b>S_QESEAL_DIG_- CADES4PADES_T_1</b>	Like S_QESEAL_DIG_CADES4PADES_1, but a qualified signature timestamp is included as a CMS unsigned attribute (identified by CaDES id-aa-signatureTimeStampToken OID). Embedded into a PDF document this is the CMS basis for the PADES-B-T signature.
<b>S_QESEAL_DIG_- CADES4PADES_LT_1</b>	Like S_QESEAL_DIG_CADES4PADES_T_1. In addition, after a successful signature of this format, OSCP responses and/or CRLs for the used certificate chain are provided in Certificate data type (see 5.5). Embedded into a PDF document this is the CMS basis for the PADES-B-LT signature.

Table 6.20: Standard values of SignatureType for qualified electronic digest sealing with qualified CA

## 6.44 SmallBlob

Base type: **base64Binary** [see [XSD](#)]; length: **0..4096**

Small blob is a byte array of maximum size of 4KB before Base64 encoding.

## 6.45 SmsPin

Base type: **string** [see [XSD](#)]; length: **4..4**

A PIN to secure SMS for specific applications. If used will be announced in project. A (currently) numerical string of exact length 4. Leading zeros are significant.

## 6.46 StreetAddress

Base type: **string** [see [XSD](#)]; length: **0..180**

The street name and other street address info such as house number etc.

## 6.47 SubscriptionState

Enumeration base type: **string** [see [XSD](#)]

The state of a subscription during the current point in time.

SubscriptionState	Meaning
INACTIVE	The subscription was never requested by the partner shop/portal or the current point in time is after endTimestamp.
PENDING_ACTIVATION	The subscription was requested by the partner shop/portal but the current point in time is before startTimestamp.
ACTIVE	The subscription was requested by the partner shop/portal but the current point in time is within startTimestamp and endTimestamp (inclusive).

Table 6.21: Values of enumeration type SubscriptionState

## 6.48 TinyBlob

Base type: **base64Binary** [see [XSD](#)]; length: **0..128**

Tiny blob is a byte array of maximum size of 128 before Base64 encoding.

## 6.49 TokenActivationProcessURL

Base type: **anyURI** [see [XSD](#)]; length: **0..256**

The unique and persistent URL which identifies an activation process at the sign-me WEB application front-end. The unique and persistent URL where the QES activation WEB application is reachable for the specific activation process, e.g. key and certificate creation, and confirmation etc. The WEB application will give access to the specific activation process in the D-Trust token activation system. The link is only valid until the date/time given in 'reservationEndTimestamp'.

## 6.50 TokenCapability

Enumeration base type: **string** [see [XSD](#)]

The token capabilities specified here are potentially supported by tokens in the sign-me system. If a token supports the capability, it contains the capability in the capabilities list.

TokenCapability	Meaning
<b>IDENTIFICATION</b>	The token type supports identification.
<b>AUTHENTICATION</b>	The token type supports authentication.
<b>AUTHENTICATION_- WITH_CLIENT_CHAL- LENCE</b>	The token type supports authentication with an additional client challenge, in other words a second factor.
<b>SIGNATURE_BASIC</b>	The token type supports electronic signatures using basic CAs of D-TRUST.
<b>SIGNATURE_ADVANCED</b>	The token type supports electronic signatures using advanced CAs of D-TRUST.
<b>SIGNATURE_QUALIFIED</b>	The token type supports electronic signatures using qualified CAs of D-TRUST.
<b>SEALING_BASIC</b>	The token type supports electronic seals using basic CAs of D-TRUST.
<b>SEALING_ADVANCED</b>	The token type supports electronic seals using advanced CAs of D-TRUST.
<b>SEALING_QUALIFIED</b>	The token type supports electronic seals using qualified CAs of D-TRUST.

Table 6.22: Values of enumeration type TokenCapability

## 6.51 UTF8String1024

Base type: **string** [see [XSD](#)]; length: **0..1024**

UTF-8 string with maximum 1024 characters (maybe more bytes because of DBCS).

## 6.52 UTF8String128

Base type: **string** [see [XSD](#)]; length: **0..128**

UTF-8 string with maximum 128 characters (maybe more bytes because of DBCS).

## 6.53 UTF8String180

Base type: **string** [see [XSD](#)]; length: **0..180**

UTF-8 string with maximum 180 characters (maybe more bytes because of DBCS).

## 6.54 UTF8String20

Base type: **string** [see [XSD](#)]; length: **0..20**

UTF-8 string with maximum 20 characters (maybe more bytes because of DBCS).

## 6.55 UTF8String200

Base type: **string** [see [XSD](#)]; length: **0..200**

UTF-8 string with maximum 200 characters (maybe more bytes because of DBCS).

## 6.56 UTF8String256

Base type: **string** [see [XSD](#)]; length: **0..256**

UTF-8 string with maximum 256 characters (maybe more bytes because of DBCS).

## 6.57 UTF8String32

Base type: **string** [see [XSD](#)]; length: **0..32**

UTF-8 string with maximum 32 characters (maybe more bytes because of DBCS).

## 6.58 UTF8String40

Base type: **string** [see [XSD](#)]; length: **0..40**

UTF-8 string with maximum 40 characters (maybe more bytes because of DBCS).

## 6.59 UTF8String400

Base type: **string** [see [XSD](#)]; length: **0..400**

UTF-8 string with maximum 400 characters (maybe more bytes because of DBCS).

## 6.60 UTF8String4096

Base type: **string** [see [XSD](#)]; length: **0..4096**

UTF-8 string with maximum 4096 characters (maybe more bytes because of DBCS).

## 6.61 UTF8String512

Base type: **string** [see [XSD](#)]; length: **0..512**

UTF-8 string with maximum 512 characters (maybe more bytes because of DBCS).

## 6.62 UTF8String80

Base type: **string** [see [XSD](#)]; length: **0..80**

UTF-8 string with maximum 80 characters (maybe more bytes because of DBCS).

## 6.63 Username

Base type: **string** [see [XSD](#)]; length: **6..80**

There are 2 possibilities for usernames:

(1) A free-form Username can contain the following characters: Letters, Digits (0-9) and the following special chars:

`-+&.`

(2) The Username can be an Email Address. Then Username needs to conform to datatype EmailAddress (see [6.14](#)).

## 6.64 ZipCode

Base type: **string** [see [XSD](#)]; length: **0..10**  
Zip/Postal code in applicable format.



## 7 Language and Platform Specific Access Modules

### 7.1 Motivation

It is generally possible to use the operations of the sign-me API by generating own access software with a WSDL toolkit (e.g., Apache CXF or Microsoft .NET tools). However, if possible, we recommend to utilize the sign-me API access modules to save time and effort.

The modules provide ready-to-use language and platform specific solutions for using the sign-me API. In addition to provide language bindings for the sign-me API these modules also offer convenience functions for login sequence and other helper functions. There are two different languages and platforms which are addressed by a separate access module:

- Oracle Java 8 and Apache CXF 3.x
- C# and Microsoft .NET runtime 4.x

For every access module there is a versionized zip-File containing:

- A README file for a quick start in the programming environment.
- A pre-compiled library implementing the access module.
- Source code of the access module including helper functions.
- The WSDL specification of the sign-me API.
- The platform specific build environment for building the library from the WSDL specification.
- Samples in the target language which show the programming of the main use cases from section 2.
- A configuration file with useful presets for quickly executing the samples on Reference Test System.

### 7.2 Java 8 Access Module

The Java 8 Access Module is delivered as a zip-File containing this document (under doc/) and the Java implementation. Sample zip file name is:

`signme-api-java-2.8-9420.zip`

### 7.3 C# Access Module

The C# Access Module is delivered as a zip-File containing this document (under doc/) and the C# implementation. Sample zip file name is:

`signme-api-cs-2.8-9420.zip`

## 8 Country Codes

All supported ICAO country codes which can be used over sign-me API along with their equivalent ISO3166 2-letter codes are listed in [countryMappingFile.xlsx](#).

## 9 Error Codes

All error codes which can be generated over sign-me API are listed in [signme\\_server\\_errorcodes.xlsx](#).

## Bibliography

[WSDL] WSDL Specification of the sign-me API. Contained in file *SignMeApiV2.wsdl* of one of the sign-me API access modules.

[XSD] Specification of the XML Schema Types, which comprise standard types like boolean, int, date etc. Contained in file <http://www.w3.org/2001/XMLSchema> and <http://www.w3.org/TR/xmlschema11-2/>.