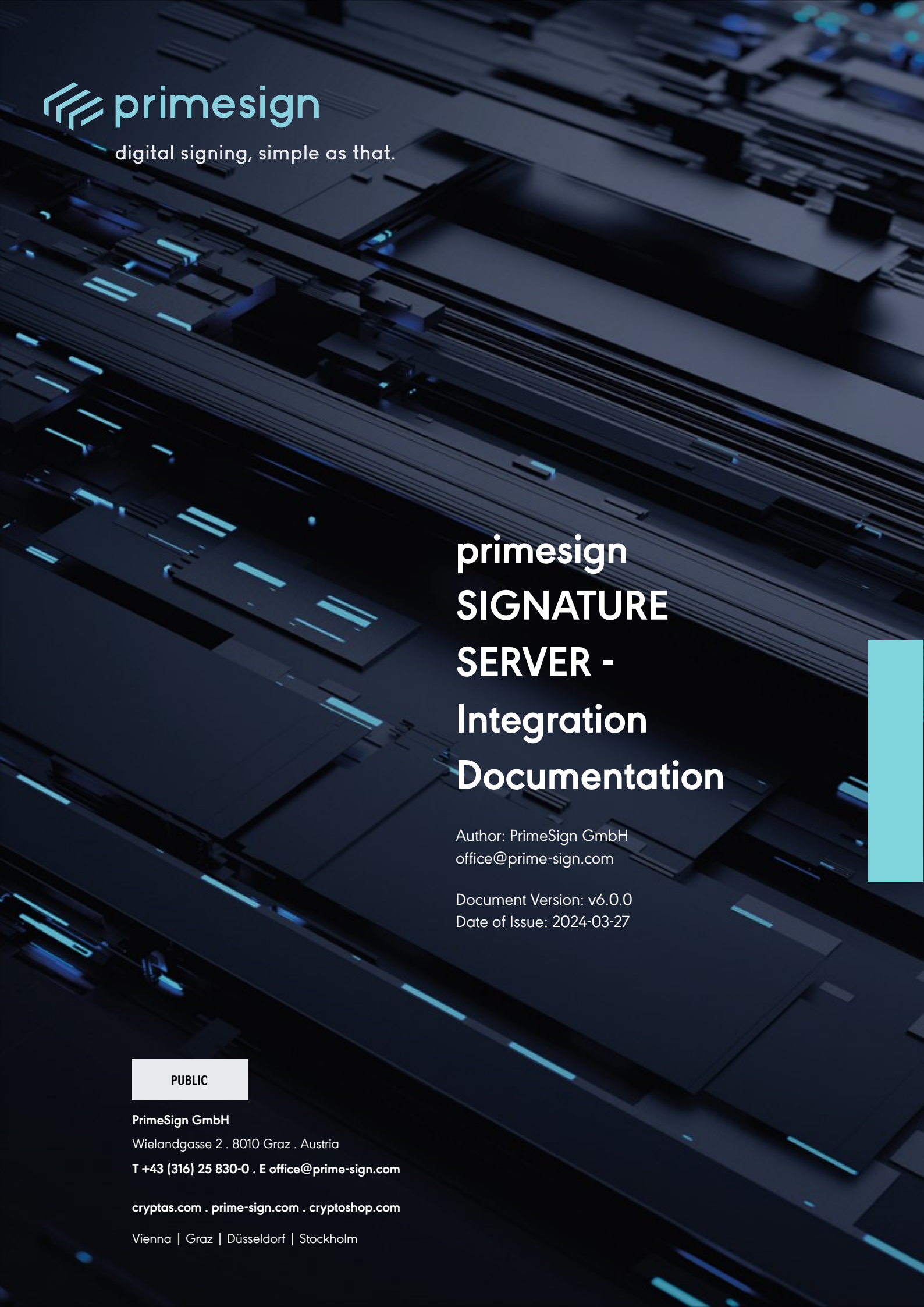


primesign

digital signing, simple as that.

primesign SIGNATURE SERVER - Integration Documentation

Author: PrimeSign GmbH
office@prime-sign.com

Document Version: v6.0.0
Date of Issue: 2024-03-27

PUBLIC

PrimeSign GmbH
Wielandgasse 2 . 8010 Graz . Austria
T +43 (316) 25 830-0 . E office@prime-sign.com

cryptas.com . prime-sign.com . cryptoshop.com

Vienna | Graz | Düsseldorf | Stockholm

TABLE OF CONTENTS

1. Document History

Date	Author	Type of Change	Version
05.08.2021	Gernot Schaberl, Thomas Knall	Creating initial document.	5.2.4
22.12.2021	Sandra Kreuzhuber	Update on placeholder detection.	5.5.0
31.01.2022	Gernot Schaberl	Update screenshot.	5.5.0
14.02.2022	Sandra Kreuzhuber	Update signature profile preselection.	5.6.0
12.05.2022	Sandra Kreuzhuber	Require cookies for full ui	5.6.3
09.06.2022	Sandra Kreuzhuber	Add primesign WRAPTOR as signature creation device. Add notice about support for bulk signing.	5.6.3
11.07.2022	Sandra Kreuzhuber	Add workflow property `pdfACompliance` to specify the desired PDF/A conformance level for document conversion.	5.6.4
16.09.2022	Sandra Kreuzhuber	Add ID Austria.	5.7.0
10.10.2022	Sandra Kreuzhuber	Add primesign MOBILE 2.0. primesign MOBILE 1.1 is deprecated.	5.7.0
27.10.2022	Sandra Kreuzhuber	Add infos on embedding the primesign SIGNATURE SERVER UI, see Section 4.3 (Do not use iFrames anymore!).	5.7.0
11.01.2023	Sandra Kreuzhuber	Add maximum number of documents for bulk signing. Specify allowed characters for workflow property `transactionId`. See Section 4.7.1.	5.7.1

Date	Author	Type of Change	Version
19.04.2023	Thomas Knall	Remove error codes 511, 700, 701 and 702 since related feature has been removed. Limit signature type for keystore/hsm related signatures to `PAdES`. Add new error code 512 for timestamping issues. Add note about certification signatures to Section 5.4. Mark `breite` and `hoehe` (width and height) parameters of SignaturService deprecated (Section 5.4). Remove deprecated signature parameters `schluesselId` and `text` from Section 5.4.	5.7.2
19.04.2023	Sandra Kreuzhuber, Thomas Knall	Update Section 4.7.1 to add new version of primesign WRAPTOR. Update example for `signaturZeitpunktFormat` in Section 5.4. Add description of new error code 424 to Section 5.5.	5.7.2
12.09.2023	Thomas Knall	Add note for error code 16 and description of new error codes 21, 22, 23 and 24 to Table 3.	5.7.3
18.09.2023	Sandra Kreuzhuber	Add documentation for non-interactive one-time signing. See values for `interfaceType` in Section 4.7.1.	5.7.3
14.02.2024	Sandra Kreuzhuber	Correct description for property `signatureName` in Section 5.4.	6.0.0
07.03.2024	Simon Roth	Description for property `benutzer` in Section 5.4.	6.0.0
07.03.2024	Sebastian Zöscher	Add signature positioning description for new experimental properties `end` and `footer` in Section 4.7.1 and Section 5.4.	6.0.0

2. Introduction

The primesign SIGNATURE SERVER provides SOAP web services allowing a simple and lightweight way integrating signature services into another applications (e.g. DMS applications). This document explains which services can be used for which integrating szenarios (see Section 3).

In addition an advanced technical documentation for supported SOAP requests is provided.

The following SOAP web services are available:

- **PrimeSignWorkflowService** - user personally signing documents
  e.g. using their personal (qualified) elektronic signature (see Section 4)
- **SignaturService** - document signing without user interaction
  e.g. using a software key store or HSM (see Section 5)

By default, all SOAP web services are disabled. For more information about enabling and configuring those services see [1].

To check which SOAP web services are currently enabled on the primesign SIGNATURE SERVER consult https://<EXT_HOST_NAME>/primesign/services

Primesign SIGNATURE SERVER supports both SOAP version 1.1 (for legacy purposes only) and 1.2 (recommended).

	SOAP 1.1 namespace	SOAP 1.2 namespace
WSDL	http://schemas.xmlsoap.org/wsdl/soap/	http://schemas.xmlsoap.org/wsdl/soap12/
Messages	http://schemas.xmlsoap.org/soap/envelope	http://www.w3.org/2003/05/soap-envelope

Table 1. SOAP versions and namespaces

Audience

This document is intended for software developers who want to integrate their own or a public primesign SIGNATURE SERVER instance into their application.

Prerequisites

In order to integrate a primesign SIGNATURE SERVER web service make sure the following requirements are fulfilled:

- The integrating application's source IP matches the **allowed IP range** of the primesign SIGNATURE SERVER (see [1]),
- the respective web **service is enabled** (see [1]) and
- the communication between the integrating application and the primesign SIGNATURE SERVER is allowed (e.g. not blocked by **firewall rules**).

3. Use Cases

This chapter presents solutions for typical requirements of an application dealing with documents that need to be signed.

3.1. With user interaction (PrimeSignWorkflowService)

This section deals with signatures requiring some kind of user interaction like when conducting Qualified Signatures.

It is recommended to start with the following use case. All subsequent cases with user interaction (advanced use cases or additional options) are based on this basic approach. Furthermore, the following examples use the operation StartWorkflow of the PrimeSignWorkflowService.

3.1.1. A single document should be signed by a specific person

Primesign SIGNATURE SERVER may be integrated in existing workflows when document(s) need to be signed by specific person(s).

Imagine an application performing some workflow dealing with documents that - at a specific point - need to be signed by a specific person, like when releasing offers or concluding contracts. This task of signing documents in a legally valid manner can be delegated to primesign SIGNATURE SERVER (PSS). Once signed the document(s) are returned to the initiating application which in turn resumes its original workflow, e.g. archiving the signed documents, sending documents to external partners or even making further persons sign the document.

The following diagram shows the minimum sequence flow required for PSS integration. Note that this reflects a simplified view only with components the integrating application needs to communicate with.

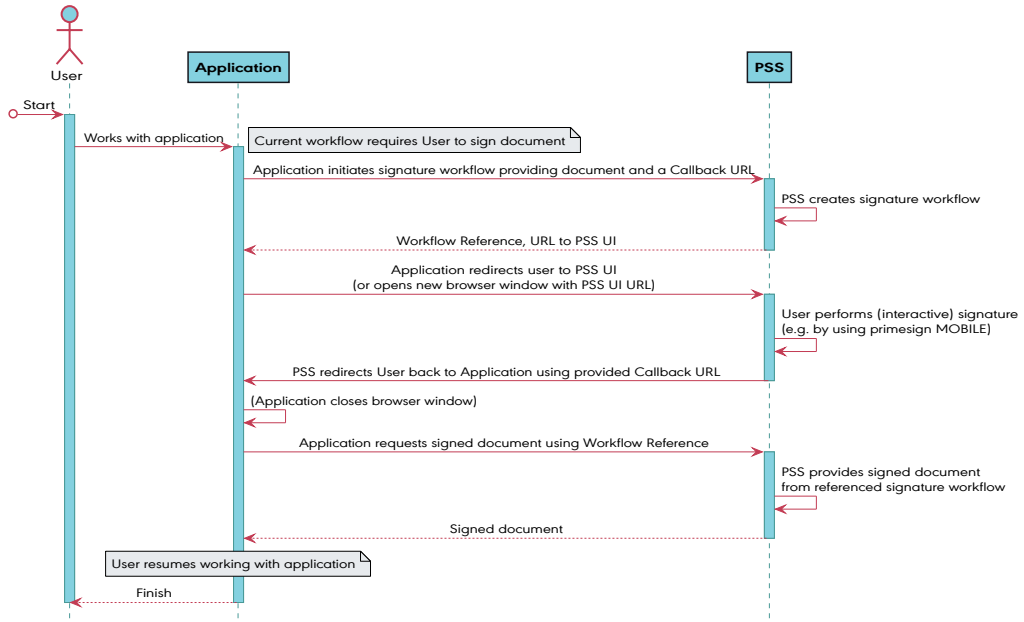


Figure 1. High level sequence flow for PSS integrated by (third party) application

Description of the flow

A workflow may reach the point where a document needs to be signed in legal manner. Therefore the underlying application suspends its current workflow (implied with `processId=1234` in the example below) and delegates the signature to PSS.

1. The application uses the SOAP webservice to create a PSS signature workflow. The application has to provide the document to be signed as well as two callback URLs which will be used by PSS to redirect the user back to the application when the signature has been conducted or has been cancelled.
2. The PSS creates a PrimeSign workflow and returns a reference to that workflow (in form of a UUID) and a URL.
3. The application redirects the user to PSS using the provided URL. This makes PrimeSign the user's active browser application. (Alternatively the URL can be opened in a new browser tab or popup window, see Section 4.3 for more information.)
4. The user conducts the signature within the PSS UI. Depending on the configuration multiple choices of signature devices (e.g. primesign MOBILE or a smartcard based signature) might be available.
5. The user selects a certain signature device and performs the signature.
6. After completing the signature the user is redirected back to the application using the URL

provided on signature workflow creation.

7. The application uses the SOAP webservice to retrieve the signed document using the workflow reference received from PSS when creating the signature workflow.

The application resumes the previously suspended workflow...

Request for simple workflow signing a document

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:v1="http://primesign.at/workflow/v1">
   <soap:Header />
   <soap:Body>
      <v1:StartWorkflowRequest>
         <v1:DocumentOrRef>
            <v1:Document>
               <v1:Filename>test.pdf</v1:Filename>
               <v1:MediaType>application/pdf</v1:MediaType>
               <v1:Content>...</v1:Content>
            </v1:Document>
         </v1:DocumentOrRef>
         <v1:Events>
            <v1:OnCancel>
               <v1:Redirect
uri="https://my.application.org/cancelProcess?processId=1234" />
            </v1:OnCancel>
            <v1:OnComplete>
               <v1:Redirect
uri="https://my.application.org/finishProcess?processId=1234" />
            </v1:OnComplete>
         </v1:Events>
      </v1:StartWorkflowRequest>
   </soap:Body>
</soap:Envelope>
```

Response

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
   <soap:Body>
      <StartWorkflowResponse xmlns="http://primesign.at/workflow/v1">
         <WorkflowInstanceId>317931f4-942d-4828-b3e1-
359486af4360</WorkflowInstanceId>
         <DocumentRef>
            <DocumentId>cf9a4c37-9db4-4998-930f-f0c7e1fc3644</DocumentId>
         </DocumentRef>

<UserInterfaceURI>https://pss.somewhere.org/primesign/resume/wf/317931f4-
942d-4828-b3e1-359486af4360</UserInterfaceURI>
      </StartWorkflowResponse>
   </soap:Body>
</soap:Envelope>
```

3.1.2. Multiple documents should be signed by a specific person (Bulk Signature)

Sometimes a workflow requires more than one document to be signed by one person. The natural approach of invoking the process for a single signature (as shown in Section 3.1.1) multiple times will do the trick but the signing person would have to enter both credentials and a (different) TAN for each document.

primesign SIGNATURE SERVER provides a specific signature mode ("Bulk Signature") allowing to approve the signature process involving multiple documents in one step, meaning that credentials and TAN need only be entered once.

This signature mode can be enabled setting a certain `WorkflowProperty` when creating a signature workflow using the SOAP webservice.

Section 4.7 provides an overview of available WorkflowProperties.

Not all signature creation devices support bulk signing. Furthermore, some signature creation devices are limited to signing a maximum number of 30 documents in one step. See Section 4.7 for more details.

For bulk signatures the value of the property `signatureMode` has to be set to `bulk`. Note that using bulk signature this way automatically activates a minimal UI representation (Section 3.1.5.1).

Example for signing multiple documents

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:v1="http://primesign.at/workflow/v1">
   <soap:Header />
   <soap:Body>
      <v1:StartWorkflowRequest>
         <v1:DocumentOrRef>
            <v1:Document>
               <v1:Filename>doc1.pdf</v1:Filename>
               <v1:MediaType>application/pdf</v1:MediaType>
               <v1:Content>...</v1:Content>
            </v1:Document>
         </v1:DocumentOrRef>
         <v1:DocumentOrRef>
            <v1:Document>
               <v1:Filename>doc2.pdf</v1:Filename>
               <v1:MediaType>application/pdf</v1:MediaType>
               <v1:Content>...</v1:Content>
            </v1:Document>
         </v1:DocumentOrRef>
         <v1:WorkflowProperty name="signatureMode" value="bulk" />
      </v1:StartWorkflowRequest>
   </soap:Body>
</soap:Envelope>
```

Response

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
   <soap:Body>
      <StartWorkflowResponse xmlns="http://primesign.at/workflow/v1">
         <WorkflowInstanceId>31b04a6a-7283-4421-93aa-
276e8d351498</WorkflowInstanceId>
         <DocumentRef>
            <DocumentId>0b628b65-7b71-412a-996c-31e7dcf448bd</DocumentId>
         </DocumentRef>
         <DocumentRef>
            <DocumentId>c7186bfc-419a-4494-be25-9cb9d89c73ca</DocumentId>
         </DocumentRef>

<UserInterfaceURI>https://pss.somewhere.org/primesign/resume/wf/31b04a6a-
7283-4421-93aa-276e8d351498</UserInterfaceURI>
      </StartWorkflowResponse>
   </soap:Body>
</soap:Envelope>
```

3.1.3. Document(s) should be signed by multiple persons

In many cases, a document or a list of documents should be signed by multiple persons. So the first person signs the document(s) and the second person adds an additional signature to the already signed document(s). For that case, primesign SIGNATURE SERVER also allows to pick up the already signed document within consecutive StartWorkflow Requests.

If placeholders are used (see further explanations in Section 3.1.4.2) and document(s) with multiple placeholders should be signed by multiple persons, it is mandatory to use this process flow.

If an already signed document should be picked up for the consecutive StartWorkflow Request, you have to make sure that this happens within the lifetime of the workflow. Otherwise the previously signed document may already be deleted. See Section 3.1.6 for more information on specifying a lifetime.

Description of the flow

An example workflow where a document with placeholders should be signed by two persons:

1. The application uses the SOAP webservice to create a PSS signature workflow. The application has to provide the document to be signed (including the two QR code placeholders) as well as two callback URLs which will be used by PSS to redirect the user back to the application when the signature has been conducted or has been cancelled. If placeholder detection is not activated on a server-level, placeholder detection must be enabled for this specific signature workflow (see Section 3.1.4.2).
2. The PSS creates a PrimeSign workflow and returns a reference to that workflow (in form of a UUID) and a URL.
3. The application redirects the user to PSS using the provided URL. The signature block is positioned at the first placeholder position. The user may change the position of the signature block and signs the document. After completing the signature the user is redirected back to the application using the URL provided on signature workflow creation.
4. The application uses the SOAP webservice to retrieve **a reference** to the signed document using the workflow reference received from PSS when creating the signature workflow.
5. The application creates a new PSS signature workflow, but **instead of uploading a document the application simply references the already signed document**.
6. PSS creates a PrimeSign workflow and returns a reference to that workflow (in form of a UUID) and a URL. Note, that this is a new workflow with a new UUID and a new URL.

7. The application redirects the user to PSS using the provided URL. The signature block is positioned at the first placeholder position not already used for signature. The user signs the document. After completing the signature the user is redirected back to the application using the URL provided on signature workflow creation.
8. The application uses the SOAP webservice to retrieve the signed PDF document using the workflow reference received from PSS when creating the second signature workflow.

The following sample requests show the process flow:

Request for creating the first workflow

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:v1="http://primesign.at/workflow/v1">
   <soap:Header />
   <soap:Body>
      <v1:StartWorkflowRequest>
         <v1:DocumentOrRef>
            <v1:Document>
               <v1:Filename>test.pdf</v1:Filename>
               <v1:MediaType>application/pdf</v1:MediaType>
               <v1:Content>...</v1:Content>
            </v1:Document>
         </v1:DocumentOrRef>
         <v1:Events>
            <v1:OnCancel>
               <v1:Redirect
uri="https://my.application.org/cancelProcess?processId=1234" />
            </v1:OnCancel>
            <v1:OnComplete>
               <v1:Redirect
uri="https://my.application.org/finishProcess?processId=1234" />
            </v1:OnComplete>
         </v1:Events>
      </v1:StartWorkflowRequest>
   </soap:Body>
</soap:Envelope>
```

Response

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
   <soap:Body>
      <StartWorkflowResponse xmlns="http://primesign.at/workflow/v1">
         <WorkflowInstanceId> ❶
            317931f4-942d-4828-b3e1-359486af4360
         </WorkflowInstanceId>
         <DocumentRef>
            <DocumentId>cf9a4c37-9db4-4998-930f-f0c7e1fc3644</DocumentId> ❷
         </DocumentRef>
         <UserInterfaceURI> ❸
            https://pss.somewhere.org/primesign/resume/wf/317931f4-942d-4828-
b3e1-359486af4360
         </UserInterfaceURI>
      </StartWorkflowResponse>
   </soap:Body>
</soap:Envelope>
```

❶ The `WorkflowInstanceId` of the newly generated workflow.

❷ The `DocumentId` of the unsiged document that has been uploaded with StartWorkflowRequest.

❸ The `UserInterfaceURI` that should be presented to the first user for signing the document.

Request for retrieving the workflow result after the first signature

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:v1="http://primesign.at/workflow/v1">
   <soap:Header/>
   <soap:Body>
      <v1:GetWorkflowResultRequest resultType="DocumentRef"> ❶
         <v1:WorkflowInstanceId> ❷
            317931f4-942d-4828-b3e1-359486af4360
         </v1:WorkflowInstanceId>
      </v1:GetWorkflowResultRequest>
   </soap:Body>
</soap:Envelope>
```

❶ Specify resultType `DocumentRef`.

❷ The `WorkflowInstanceId` of the previously generated workflow.

Response

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
   <soap:Body>
      <GetWorkflowResultResponse xmlns="http://primesign.at/workflow/v1">
         <DocumentOrRef>
            <DocumentRef>
               <DocumentId> ❶
                  9fe32cfa-6e17-11ec-90d6-0242ac120003
               </DocumentId>
            </DocumentRef>
         </DocumentOrRef>
      </GetWorkflowResultResponse>
   </soap:Body>
</soap:Envelope>
```

❶ The `DocumentId` of the document signed by the first user.

Request for starting the second workflow by referencing the already signed document

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:v1="http://primesign.at/workflow/v1">
   <soap:Header />
   <soap:Body>
      <v1:StartWorkflowRequest>
         <v1:DocumentOrRef>
            <v1:DocumentRef>
               <v1:DocumentId> ❶
                  9fe32cfa-6e17-11ec-90d6-0242ac120003
               </v1:DocumentId>
            </v1:DocumentRef>
         </v1:DocumentOrRef>
         <v1:Events>
            <v1:OnCancel>
               <v1:Redirect
uri="https://my.application.org/cancelProcess?processId=1234" />
            </v1:OnCancel>
            <v1:OnComplete>
               <v1:Redirect
uri="https://my.application.org/finishProcess?processId=1234" />
            </v1:OnComplete>
         </v1:Events>
      </v1:StartWorkflowRequest>
   </soap:Body>
</soap:Envelope>
```

❶ The `DocumentId` of the document signed by the first user, so that the second user signs the already signed document.

Response

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
   <soap:Body>
      <StartWorkflowResponse xmlns="http://primesign.at/workflow/v1">
         <WorkflowInstanceId> ❶
            feb891ac-6e17-11ec-90d6-0242ac120003
         </WorkflowInstanceId>
         <DocumentRef>
            <DocumentId> ❷
               9fe32cfa-6e17-11ec-90d6-0242ac120003
            </DocumentId>
         </DocumentRef>
         <UserInterfaceURI> ❸
            https://pss.somewhere.org/primesign/resume/wf/feb891ac-6e17-11ec-
90d6-0242ac120003
         </UserInterfaceURI>
      </StartWorkflowResponse>
   </soap:Body>
</soap:Envelope>
```

❶ The `WorkflowInstanceId` of the second workflow. This is a new workflow with a new `WorkflowInstanceId`.

❷ The `DocumentId` of the previously uploaded document (document that has been signed by the first user).

❸ The `UserInterfaceURI` that should be presented to the second user for signing the document.

Request for retrieving the workflow result after the second signature

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:v1="http://primesign.at/workflow/v1">
   <soap:Header/>
   <soap:Body>
      <v1:GetWorkflowResultRequest resultType="Document"> ❶
         <v1:WorkflowInstanceId> ❷
            feb891ac-6e17-11ec-90d6-0242ac120003
         </v1:WorkflowInstanceId>
      </v1:GetWorkflowResultRequest>
   </soap:Body>
</soap:Envelope>
```

❶ ResultType `Document` (default value, can also be omitted) to return the binary content of the signed document.

❷ The `WorkflowInstanceId` of the second workflow.

Response

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
   <soap:Body>
      <GetWorkflowResultResponse xmlns="http://primesign.at/workflow/v1">
         <DocumentOrRef>
            <Document>
               <Filename>test.pdf</Filename>
               <MediaType>application/pdf</MediaType>
               <Content>...</Content> ❶
            </Document>
         </DocumentOrRef>
      </GetWorkflowResultResponse>
   </soap:Body>
</soap:Envelope>
```

❶ The binary content of the PDF document signed by both users.

3.1.4. Additional options

This section covers options that can be set in addition to the use cases shown above. Note that the described options can also be combined (e.g. both selecting UI language and enabling scanning of QR code) within a single workflow.

3.1.4.1. Preselect UI language

In case the initiating application wants to pass its current UI language to PSS for sake of usability, this can be done using the WorkflowProperty `locale`. Supported values are `de` (German), `en` (English), `el` (Greek).

When the property is omitted PSS automatically selects the UI language based on either the user's browser language or based on the user's manual selection (the user may select the language within the UI; the selection will be stored using a browser cookie).

Example for preselection of UI language

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:v1="http://primesign.at/workflow/v1">
   <soap:Header />
   <soap:Body>
      <v1:StartWorkflowRequest>
         <v1:DocumentOrRef>
            <v1:Document>
               <v1:Filename>doc1.pdf</v1:Filename>
               <v1:MediaType>application/pdf</v1:MediaType>
               <v1:Content>...</v1:Content>
            </v1:Document>
         </v1:DocumentOrRef>
         <v1:WorkflowProperty name="locale" value="en" />
      </v1:StartWorkflowRequest>
   </soap:Body>
</soap:Envelope>
```

3.1.4.2. Use QR code image in document as position placeholder

PSS supports detection of desired signature position by scanning the respective document for certain embedded QR codes. This feature is usually disabled by default, but can be enabled when creating a workflow.

In order to enable this feature (for the created workflow only) set the WorkflowProperty `signaturePlaceholders` to `detectAndRemove`.

Example for enabling QR code image as signature position placeholder

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:v1="http://primesign.at/workflow/v1">
   <soap:Header />
   <soap:Body>
      <v1:StartWorkflowRequest>
         <v1:DocumentOrRef>
            <v1:Document>
               <v1:Filename>doc1.pdf</v1:Filename>
               <v1:MediaType>application/pdf</v1:MediaType>
               <v1:Content>...</v1:Content>
            </v1:Document>
         </v1:DocumentOrRef>
         <v1:WorkflowProperty name="signaturePlaceholders"
value="detectAndRemove" />
      </v1:StartWorkflowRequest>
   </soap:Body>
</soap:Envelope>
```

The detection can also be enabled within the PSS configuration. If enabled, all documents uploaded via user interface, SOAP or directory scanner will be scanned for QR codes. See [1].

Enabling this option means that PSS scans the document for certain QR codes, remembers the respective QR code positions within the document and removes the image containing the QR code. When the signature is conducted, the visual signature will be placed at the first detected position.

If document(s) with multiple placeholders should be signed by multiple persons make sure to use the process flow as described in Section 3.1.3. Placeholders are removed from a document after detection and the placeholder position is stored with the document meta data. If the document is downloaded after the first signature and then uploaded again, placeholder information is lost.

- If placeholder detection is disabled within the PSS configuration, but enabled within `StartWorkflowRequest` document(s) are only scanned for placeholder if provided as `Document` and not as `DocumentRef`.
- With the recommended default configuration signed documents will not be scanned for QR codes. Furthermore, QR code images will not be removed when the document is already signed since signed documents must not be modified.
- If the signature profile used for signing is larger than the placeholder image document content may be hidden. See [4] for guidance how to create documents with placeholder.

3.1.4.3. Preselect a certain signature profile

A signature profile, by and large, reflects the visual appearance of a signature. Usual setups allow a user to select from multiple signature profiles when conducting signatures. In case an integrating application wants to preselect a certain profile for the user, this can be done by setting the WorkflowProperty `signatureProfileId` to the respective identifier of the profile. When using the Rich UI for signing, the user is still able to switch to another signature profile. If the Minimal UI is used, the user cannot change the preselected signature profile.

Beware that the referenced signature profile must exist and the access control list (acl) of the signature profile must ensure that the signer can use this profile.
This can be realised in different ways:
a) By using the Admin UI to extend the acl settings of the signature profile to allow access for group `WEBSERVICE` (*recommended*) or
b) using a publicly accessible signature profile that can be used by all users.

Example for preselecting a certain signature profile

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:v1="http://primesign.at/workflow/v1">
   <soap:Header />
   <soap:Body>
      <v1:StartWorkflowRequest>
         <v1:DocumentOrRef>
            <v1:Document>
               <v1:Filename>doc1.pdf</v1:Filename>
               <v1:MediaType>application/pdf</v1:MediaType>
               <v1:Content>...</v1:Content>
            </v1:Document>
         </v1:DocumentOrRef>
         <v1:WorkflowProperty name="signatureProfileId" value="NEUTRAL_DE" />
      </v1:StartWorkflowRequest>
   </soap:Body>
</soap:Envelope>
```

3.1.4.4. Select document archive

PSS supports automatic archival of signed documents, meaning that PSS can be configured to store signed documents within the file system (or to mounted shares). An instance may support several archives (e.g. multiple different mounted shares). Each archive has its own identifier.

The selection of the respective archive can be done when creating the underlying workflow, using the WorkflowProperty `archiveInstanceId`. Set the value to the respective identifier of a configured archive.

Example for selecting a certain document archive

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:v1="http://primesign.at/workflow/v1">
   <soap:Header />
   <soap:Body>
      <v1:StartWorkflowRequest>
         <v1:DocumentOrRef>
            <v1:Document>
               <v1:Filename>timesheet-2021-08-john_doe.pdf</v1:Filename>
               <v1:MediaType>application/pdf</v1:MediaType>
               <v1:Content>...</v1:Content>
            </v1:Document>
         </v1:DocumentOrRef>
         <v1:WorkflowProperty name="archiveInstanceId"
value="timesheetArchive" />
      </v1:StartWorkflowRequest>
   </soap:Body>
</soap:Envelope>
```

3.1.5. Advanced Usage

After successfully having integrated the basic use case the following advanced cases can be adressed.

3.1.5.1. PSS should only show the minimum possible UI

The primesign SIGNATURE SERVER offers a minimal UI, that allows for a light-weight user experience, for example when displayed within a browser dialog (popup window).

PSS distinguishes between a "rich" UI (which is default for desktop devices) and a so called "minimal" UI (which is default for mobile devices).

In order to use the fully fletched rich UI the WorkflowProperty `interfaceType` need to be set to `full`, in case of the minimal approach, the value should be set to `minimum`. WorkflowProperties can be set when creating a signature workflow using the SOAP webservice.

Using bulk signature (Section 3.1.2) together with `interfaceType=full` is not supported.

Section 4.7 provides an overview of available WorkflowProperties.

3.1.5.2. A certain (selection of) signature creation device(s) should be enforced

In case an integrating application requires that a person uses a certain signature creation device, e.g. using qualified primesign MOBILE signature instead of a software keystore, the choice of available devices can be restricted to a certain set of devices, provided that at least one device remains available for signature.

In order to restrict the list of signature devices to primesign MOBILE 2.0 and ID Austria / Austrian Mobile Phone Signature set the WorkflowProperty `signatureCreationEntities=pcss,mobileSignature`.

In case only one signature creation device remains, the selection dialogue of the UI will be automatically skipped.

Section 4.7 provides an overview of available WorkflowProperties, in particular the values available for restriction of signature creation devices.

3.1.5.3. Preselect signature position

Recommended for use case "*A single document should be signed by a specific person*" (Section 3.1.1)

When using the Rich UI for conducting signatures (refer to use case "Section 3.1.5.1" for further information) the stamper reflecting the intended signature position is automatically positioned within the last page of the document. The user may move the stamper to another position or even to another page if desired.

When creating a workflow using this service the initial proposed signature position may be set. The stamper will be placed at the specified position. Note that the user still may move the stamper to another positon.

In order to preset the signature position for a new workflow use the WorkflowProperty `signaturePosition`. Section "Section 4.7" describes the syntax of the value.

Example for preselection of the stamper position (autopositioning on page 3)

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:v1="http://primesign.at/workflow/v1">
   <soap:Header />
   <soap:Body>
      <v1:StartWorkflowRequest>
         <v1:DocumentOrRef>
            <v1:Document>
               <v1:Filename>test.pdf</v1:Filename>
               <v1:MediaType>application/pdf</v1:MediaType>
               <v1:Content>...</v1:Content>
            </v1:Document>
         </v1:DocumentOrRef>
         <v1:WorkflowProperty name="signaturePosition" value="p:3" />
         <v1:Events>
            <v1:OnCancel>
               <v1:Redirect
uri="https://my.application.org/cancelProcess?processId=1234" />
            </v1:OnCancel>
            <v1:OnComplete>
               <v1:Redirect
uri="https://my.application.org/finishProcess?processId=1234" />
            </v1:OnComplete>
         </v1:Events>
      </v1:StartWorkflowRequest>
   </soap:Body>
</soap:Envelope>
```

Priority of signature positions:

If multiple signature positions are given, the following priorization applies:

- Position chosen by the user (e.g. by manually positioning the stamper in the Rich UI)
- WorkflowProperty `signaturePosition`
- The following have the same priority and would be merged (Note: primesign does not recommend using both in the same request.):
  - DocumentProperty `signaturePosition`
  - Placeholder (QR code images) placed within the document (see Section 3.1.4.2)
- Auto-positioning

Unsupported: It is not supported to mix `signaturePosition` as DocumentProperty and WorkflowProperty in the same request.

3.1.5.4. Enforce signature position

Recommended for use cases "*Multiple documents should be signed by a specific person (Bulk Signature)*" (Section 3.1.2) and "*PSS should only show the minimum possible UI*" (Section 3.1.5.1)

This feature allows to enforce a certain signature position, meaning that the user is not able to change the final signature position.

The main difference between this feature and the feature described in "Section 3.1.5.3" is that the position is defined on document level and not on workflow level. This allows to set individual signature positions in combination with bulk signature (Section 3.1.2) (and implicitely with minimal UI (Section 3.1.5.1)).

Example for enforcing the stamper position

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:v1="http://primesign.at/workflow/v1">
   <soap:Header />
   <soap:Body>
      <v1:StartWorkflowRequest>
         <v1:DocumentOrRef>
            <v1:Document>
               <v1:Filename>doc1.pdf</v1:Filename>
               <v1:MediaType>application/pdf</v1:MediaType>
               <v1:Content>...</v1:Content>
               <v1:DocumentProperty name="signaturePosition" value="p:2" />
            </v1:Document>
         </v1:DocumentOrRef>
         <v1:DocumentOrRef>
            <v1:Document>
               <v1:Filename>doc2.pdf</v1:Filename>
               <v1:MediaType>application/pdf</v1:MediaType>
               <v1:Content>...</v1:Content>
               <v1:DocumentProperty name="signaturePosition"
value="p:1;x:100;y:200" />
            </v1:Document>
         </v1:DocumentOrRef>
         <v1:WorkflowProperty name="interfaceType" value="minimum"/>
         <v1:Events>
            <v1:OnCancel>
               <v1:Redirect
uri="https://my.application.org/cancelProcess?processId=1234" />
            </v1:OnCancel>
            <v1:OnComplete>
               <v1:Redirect
uri="https://my.application.org/finishProcess?processId=1234" />
            </v1:OnComplete>
         </v1:Events>
      </v1:StartWorkflowRequest>
   </soap:Body>
</soap:Envelope>
```

Priority of signature positions:

If multiple signature positions are given, the following priorization applies:

- WorkflowProperty `signaturePosition`
- The following have the same priority and would be merged (Note: primesign does not recommend using both in the same request.):

- DocumentProperty `signaturePosition`
- Placeholder (QR code images) placed within the document (see Section 3.1.4.2)

- Auto-positioning

3.1.5.5. Modify UI workflow buttons

Workflow related buttons (for completing (`CompleteButton`) or cancelling (`CancelButton`) a workflow) can be customized or hidden.

- Using the `display` parameter corresponding button can be displayed (value: `true` [default]) or hidden (value: `false`).
- The `Label` element holds the text of the associated button (e.g. "Cancel TEXT").
- With the `tooltip` parameter a text can be defined which will be shown if the user moves the pointer over the button.

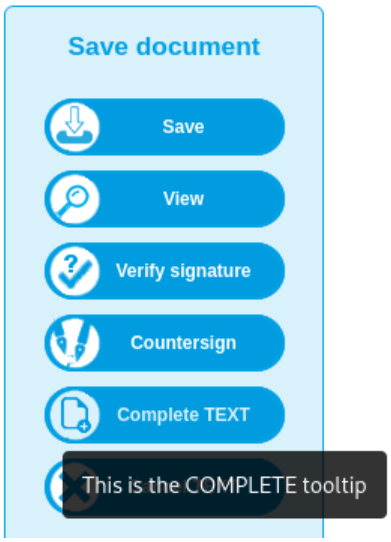


Figure 2. Customizable workflow complete button (UI).

Example for modifying workflow related buttons in UI

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:v1="http://primesign.at/workflow/v1">
   <soap:Header/>
   <soap:Body>
      <v1:StartWorkflowRequest>
         <v1:DocumentOrRef>
            <v1:Document>
               <v1:Filename>doc1.pdf</v1:Filename>
               <v1:MediaType>application/pdf</v1:MediaType>
               <v1:Content>...</v1:Content>
            </v1:Document>
         </v1:DocumentOrRef>
         <v1:UICustomizations>
            <v1:CancelButton display="true">
               <v1:Label tooltip="This is the CANCEL tooltip">Cancel
TEXT</v1:Label>
            </v1:CancelButton>
            <v1:CompleteButton display="true">
               <v1:Label tooltip="This is the COMPLETE tooltip">Complete
TEXT</v1:Label>
            </v1:CompleteButton>
         </v1:UICustomizations>
      </v1:StartWorkflowRequest>
   </soap:Body>
</soap:Envelope>
```

Both UI customizations are only applicable for **interfaceType** `full`.

3.1.6. Create a signature workflow with expiration

In case a signature workflow should be available by a certain date or for a certain amount of time (e.g. a week), a workflow lifetime can be defined. This is useful if the signing process can't be finished timely. Each workflow features a lifetime (default: 1 day) which starts after creation of a workflow. The lifetime can be defined as duration (`ExpiresIn`) or by an explicit date (`ExpiresOn`).

Example for setting a workflow lifetime manually

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:v1="http://primesign.at/workflow/v1">
   <soap:Header/>
   <soap:Body>
      <v1:StartWorkflowRequest>
         <v1:DocumentOrRef>
            <v1:Document>
               <v1:Filename>doc1.pdf</v1:Filename>
               <v1:MediaType>application/pdf</v1:MediaType>
               <v1:Content>...</v1:Content>
            </v1:Document>
         </v1:DocumentOrRef>
         <v1:ExpiresIn>P7D</v1:ExpiresIn>
      </v1:StartWorkflowRequest>
   </soap:Body>
</soap:Envelope>
```

In this example the workflow `ExpiresIn` seven days. The duration has to satisfy the datatype `xsd:duration` format and specifies a time interval as `PnYnMnDTnHnMnS` whereupon

- `P` indicating the period.
- `nY`, `nM`, `nD` has to be replaced by a certain value, e.g. `1Y` for "one-year" etc. (`Y` means years, `M` means months, `D` means days)
- `T` indicating the start of the time section.
- `nH`, `nM`, `nS` has to be replaced by a certain value, e.g. `1H` for "one-hour" etc. (`H` means hours, `M` means minutes, `S` means seconds)

If a specific date should be set for expiring the workflow the `ExpiresOn` setting has to satisfy `xsd:dateTime` specifications (e.g. `YYYY-MM-DDThh:mm:ss`), e.g. `2021-08-01T09:41:00`.

See W3C - Duration and W3C - DateTime for details.

3.2. Without user interaction (SignaturService)

This section features synchronous use cases where no user interaction is required, like signatures based on software keystores or HSM based signatures.

3.2.1. Sign document using (Qualified) seal

Signing a document without user interaction requires to use the synchronous **"SignaturService"** (see Section 5 for details). Note that this service has its own endpoint address.

In order to use a keystore/hsm signature the respective signature profile must be equipped with either a software keystore or must be associated with a HSM partition/key handle.

The request must contain the (single) document to be signed as well as the identifier of the signature profile (element `profilId`) to be used. Although the profile id is optional, referencing a specific profile with the associated sealing certificate is highly recommended.

Note that the elements are named in German language.

If auto positioning (usually on the last page of the document) should be used, omit the parameter `position`. If placeholder detection is enabled within the primesign SIGNATURE SERVER configuration, *all* uploaded documents will be scanned for signature placeholder and the visual representation of the signature will be placed on the first placeholder. Beware, the parameter `position` has higher priority: If a `position` is set explicitely, placeholder will be ignored.

Example for seal signature

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:v1="http://www.dt-i.at/primesign/services/signaturService/v1"
   xmlns:v11="http://www.dt-i.at/primesign/types/v1">
   <soapenv:Header />
   <soapenv:Body>
      <v1:dokumentSignieren schemaVersion="1.1">
         <v1:signaturParameter>
            <v11:profilId>NEUTRAL_DE</v11:profilId>
            <v11:position>
               <v11:seite>1</v11:seite>
               <v11:x>100</v11:x>
               <v11:y>200</v11:y>
            </v11:position>
         </v1:signaturParameter>
         <v1:dokument>
            <v11:dateiname>test.pdf</v11:dateiname>
            <v11:medienTyp>application/pdf</v11:medienTyp>
            <v11:daten>...</v11:daten>
         </v1:dokument>
      </v1:dokumentSignieren>
   </soapenv:Body>
</soapenv:Envelope>
```

Response

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
      <ns2:dokumentSignierenResponse schemaVersion="1.1"
         xmlns:ns3="http://www.dt-i.at/common/types/v1"
         xmlns:ns2="http://www.dt-i.at/primesign/services/signaturService/v1"
         xmlns="http://www.dt-i.at/primesign/types/v1">
         <ns2:dokument>
            <dateiname>test.pdf</dateiname>
            <medienTyp>application/pdf</medienTyp>
            <daten>...</daten>
         </ns2:dokument>
      </ns2:dokumentSignierenResponse>
   </soap:Body>
</soap:Envelope>
```

4. PrimeSignWorkflowService

This SOAP based webservice allows third party applications to integrate PSS into their document oriented workflows for the purpose of applying (Qualified) Signatures.

4.1. Endpoints

- **PrimeSignWorkflowServiceMTOM (recommended)**
  - Endpoint: https://<EXT_HOST_NAME>/primesign/services/workflowMTOM
  - WSDL: https://<EXT_HOST_NAME>/primesign/services/workflowMTOM?wsdl
- **PrimeSignWorkflowService**
  - Endpoint: https://<EXT_HOST_NAME>/primesign/services/workflow
  - WSDL: https://<EXT_HOST_NAME>/primesign/services/workflow?wsdl

4.2. Process sequence

The following figure shows the sequence of a typical integration of PSS by a third party application (aka "integrating application").

The process involves a **user** doing some work using the (integrating) **application** with a web **browser**. At some point the user needs to sign a document provided by the application. The application delegates the signing to **PSS**, which processes the document, interacts with the user, prepares the signature and delegates to a signature creation device (**SCD**) for creating the actual signature. Once the signature is created the SCD provides the signature value to PSS, PSS finishes the document and provides the signed document to the integrating application.

Note that the SCD (e.g. a Qualified Signature device like primesign MOBILE, primesign WRAPTOR or ID Austria / Austrian Mobile Phone Signature) may involve additional steps like interacting with an Identity Provider, enforcing multi-factor authentication etc. For the sake of simplicity we assume that username/password credentials are sufficient.

An integrating application needs to implement steps 4, 6, 7, 31, and 33.

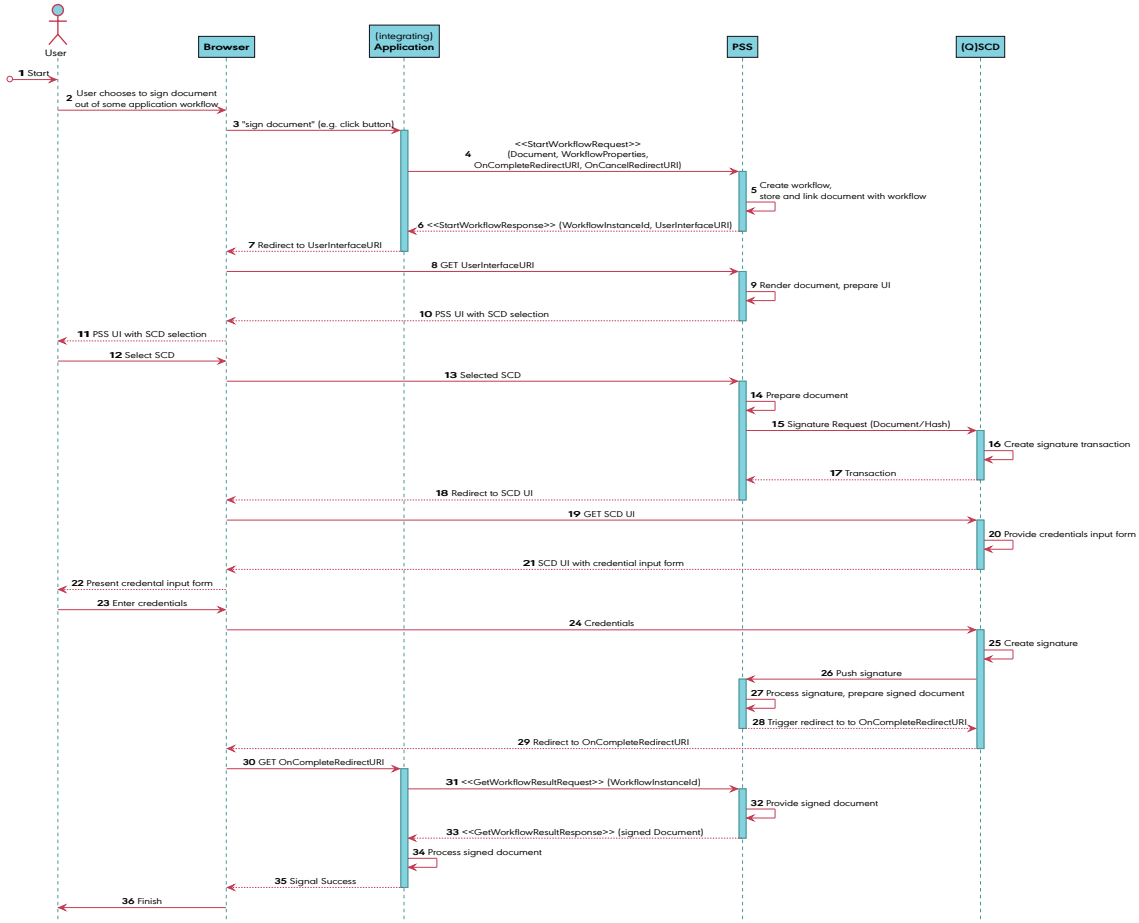


Figure 3. Sequence flow for PSS integrated by (third party) application

The following steps describe the steps shown above (assuming that the application has already integrated PSS via SOAP webservice).

1. The user handles a workflow using a certain application.
2. At some point a document signature is required.
3. The user selects the signature function of the current application (something like clicking a button).
4. The application send a `StartWorkflowRequest` to the respective SOAP webservice endpoint of the PSS. Besides the document to be signed, the application may optionally provide `WorkflowProperties` (key/value pairs for customization of a workflow; refer to Section 4.7) as well as a Callback-URL the user is redirected to when the signature is completed (`OnCompleteRedirectURI`) or cancelled (`OnCancelRedirectURI`) respectively.

5. PSS creates a workflow (considering WorkflowProperties if any), stores the document and links the document to the newly created workflow.
6. PSS responds with a `StartWorkflowResponse` providing a reference (`workflowInstanceId`) to the newly created workflow to the application and a URI to the PSS UI (`UserInterfaceURI`). The `workflowInstanceId` can later be used to retrieve the completed workflow (the signed document).
7. The application redirects the user to the just received `UserInterfaceURI` .
8. The user's browser follows the redirect to PSS.
9. PSS prepares the UI, renders the document (in case of rich UI mode) and provides a selection of available signature creation devices (SCDs).
10. PSS provides the SCD selection.
11. The user's browser shows the SCD selection.
12. The user selects the respective SCD.
13. The browser sends the user's selection to PSS.
14. PSS prepares the document so be signed (the visual representation of the signature is being embedded in the document).
15. PSS initiates the signature with the respective SCD sending either the document or the digest to be signed.
16. The SCD creates a signature transaction/session. Note that this and the following steps regarding the SCD may deviate depending on the respective SCD...
17. The SCD returns a reference to the signature session.
18. PSS redirects the user's browser to the SCD UI.
19. The user's browser follows the redirection.
20. The SCD prepares a kind of input form for the user's credentials.
21. The SCD sends the input form to the user's browser.
22. The user's browser shows the SCDs input form.
23. The user enters the respective credentials.
24. The credentials are sent via form POST to the SCD.
25. The SCD creates the signature of the document or the hash provided beforehand.
26. While the user's browser is waiting for response after POSTing the credentials, the SCD send the signature value to PSS.
27. PSS integrates the signature (value) into the pdf document. The signature is actually completed, the document is signed. Now PSS needs to notify the implementing application that the workflow result can be retrieved...
28. PSS makes the SCD redirecting the user to the `OnCompleteRedirectURI` from the `StartWorkflowRequest` .

29. The user is redirected to `OnCompleteRedirectURI` (the integrating application).
30. The user's browser redirects the user to the integrating application.
31. The integrating application sends a `GetWorkflowResultRequest` to PSS with the respective `workflowInstanceId` in order to retrieve the signed document.
32. PSS provides the signed document.
33. PSS returns a `GetWorkflowResultResponse` with the signed document.
34. The integrating application receives and processes the signed document.
35. The application shows a success message to the user.
36. The user just finished the signature workflow.

4.3. User Interface Integration

The integrating application must display the `UserInterfaceURI` to allow the user to sign the document. This can be implemented in two ways:

- **full page redirect**
- **new browser tab or popup window**

4.3.1. Full Page Redirect

Your application redirects the user to the provided `UserInterfaceURI`. As soon as the user has signed the document or cancelled signing the user is redirected back to your application (to the specified `OnComplete` and `OnCancel` respectively).

4.3.2. Browser Tab or Popup Window

Open a new browser tab or popup window via window.open and set the provided `UserInterfaceURI`. The window.open method supports various parameters for customization, e.g. open as popup window, defining the width and heigth of the new window as well as the position on screen.

As soon as the user has signed the document or cancelled signing the user is redirected back to your application (to the specified `OnComplete` and `OnCancel` respectively). Note,that this redirect is performed within the previously opened browser tab or popup window. It is up to the integrating application to continue processing and closing the tab or popup.

Do not use iFrames.

In the last years browser support for iFrames has continously been restricted. Using iframes to embed the primesign SIGNATURE SERVER user interface can lead to various issues:

- Feature limitations: signature with primesign MOBILE is not compatible with iframes, as the OAuth security standard recommends to avoid the use of iframes (see RFC 6819 section 4.4.1.9). Furthermore, there are many problems when Cookies are used in combination with iFrames. This leads to features not working when embedded in iFrames.
- Browser support constantly changes: Iframe support from browser vendors is constantly changing, new security- and privacy-related restrictions are implemented. This is due to the browser vendors efforts to make user tracking more difficult. Integrations still working today, may stop working with a new browser version.
- Security: There are causes for concern when using an iframe from the perspective of security. For instance, iframes allow for clickjacking, as the user cannot see the primesign SIGNATURE SERVER URL to check that the destination is correct.

4.4. Providing documents

PrimeSignWorkflowService supports two ways of embedding documents (binary data) in SOAP requests.

Either a document is **base64 encoded** and embedded (see Base64 example) within the XML-structure of the request or it is transmitted in **binary format using additional mime parts** (see MTOM example).

Base64-encoding binary data increased the total size of the payload by about a third and hence be less efficient. The preferred and further elaborated way is using the optimized transmission method for binary data (MTOM/XOP).

4.5. SOAP requests in detail

This section shows and describes basic `PrimeSignWorkflowService` SOAP requests.

4.5.1. MTOM example

This example shows how the document can efficiently be transferred as binary mime part (avoiding any base64 encoding). This reflects the recommended way.

Request

```
POST https://<EXT_HOST_NAME>/primesign/services/workflowMTOM HTTP/1.1 ❶
Accept-Encoding: gzip,deflate
Content-Type: multipart/related; type="application/xop+xml";
start="<rootpart@soapui.org>"; start-info="application/soap+xml";
action="StartWorkflow"; boundary="----=_Part_20_2141670619.1613116331667" ❷
[...]

------=_Part_20_2141670619.1613116331667 ❸
Content-Type: application/xop+xml; charset=UTF-8;
type="application/soap+xml"; action="StartWorkflow"
Content-Transfer-Encoding: 8bit
Content-ID: <rootpart@soapui.org>

<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:v1="http://primesign.at/workflow/v1"> ❹
   <soap:Header/>
   <soap:Body> ❺
      <v1:StartWorkflowRequest>
         <v1:DocumentOrRef>
            <v1:Document>
               <v1:Filename>test.pdf</v1:Filename>
               <v1:MediaType>application/pdf</v1:MediaType>
               <v1:Content><inc:Include href="cid:test.pdf"
xmlns:inc="http://www.w3.org/2004/08/xop/include"/></v1:Content> ❻
            </v1:Document>
         </v1:DocumentOrRef>
      </v1:StartWorkflowRequest>
   </soap:Body> ❺
</soap:Envelope> ❹

------=_Part_20_2141670619.1613116331667 ❸
Content-Type: application/pdf ❻
Content-Transfer-Encoding: binary
Content-ID: <test.pdf>
Content-Disposition: attachment; name="test.pdf"

%PDF-1.4 ❼
[...]
```

❶ initiates the POST HTTP/1.1 message to the respective endpoint.

❷ sets the content as `multipart/related` message which also may include binary data (`xop`) calling an action/operation `StartWorkflow` splitted into parts separated by a `boundary`.

❸ marks the beginning of a part of the multipart message.

❹ specifies the SOAP envelope setting the SOAP 1.2 namespace (http://www.w3.org/2003/05/soap-envelope).

❺ shows the SOAP body containing the request for PSS.

❻ header of the binary data part (containing the document to be signed).

❼ PDF binary content

Response

```
HTTP/1.1 200
[...]
Content-Type: multipart/related; type="application/xop+xml";
boundary="uuid:f704d9b2-ea8c-4100-8388-d502a5a2782a";
start="<root.message@cxf.apache.org>"; start-
info="application/soap+xml";charset=UTF-8
MIME-Version: 1.0
Content-Length: 106103
[...]

--uuid:f704d9b2-ea8c-4100-8388-d502a5a2782a
Content-Type: application/xop+xml; charset=UTF-8; type="application/soap+xml"
Content-Transfer-Encoding: binary
Content-ID: <root.message@cxf.apache.org>

<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
   <soap:Body>
      <StartWorkflowResponse xmlns="http://primesign.at/workflow/v1">
         <WorkflowInstanceId>efa2aee4-2ec6-4591-a74b-
908d3fbe5cd3</WorkflowInstanceId>
         <DocumentRef>
            <DocumentId>ccbe8fba-07c5-414b-9514-9f7ee01f1e9a</DocumentId>
         </DocumentRef>

<UserInterfaceURI>https://<EXT_HOST_NAME>/primesign/resume/wf/efa2aee4-2ec6-
4591-a74b-908d3fbe5cd3</UserInterfaceURI>
      </StartWorkflowResponse>
   </soap:Body>
</soap:Envelope>
--uuid:f704d9b2-ea8c-4100-8388-d502a5a2782a--
```

4.5.2. Base64 example

This example shows how a document can be directly embedded into the XML request.

Request

```
POST http://<EXT_HOST_NAME>/primesign/services/workflow HTTP/1.1
Accept-Encoding: gzip,deflate
Content-Type: application/soap+xml;charset=UTF-8;action="StartWorkflow"
Content-Length: 1506
Host: <EXT_HOST_NAME>
Connection: Keep-Alive
User-Agent: Apache-HttpClient/4.5.5 (Java/12.0.1)

<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:v1="http://primesign.at/workflow/v1">
   <soap:Header/>
   <soap:Body>
      <v1:StartWorkflowRequest>
         <v1:DocumentOrRef>
            <v1:Document>
               <v1:Filename>test.pdf</v1:Filename>
               <v1:MediaType>application/pdf</v1:MediaType>
               <v1:Content> ❶

JVBERi0xLjQKJeLjz9MKMSAwIG9iago8PC9MZW5ndGggMjgvRmlsdGVyL0ZsYXRl
                  [...]
                  MWMzMjA+XT4+CnN0YXJ0eHJlZgo0ODUKJSVFT0YK
               </v1:Content>
            </v1:Document>
         </v1:DocumentOrRef>
      </v1:StartWorkflowRequest>
   </soap:Body>
</soap:Envelope>
```

❶ In contrast to SOAP requests with MTOM usage (see MTOM example) test.pdf is directly embedded as base64 encoded value of the `Content` element.

Embedded base64 encoded binary data increases the size of the request **by about a third** and hence is regarded less efficient. The preferred way is to use the MTOM approach.

Response

```
HTTP/1.1 200
[...]
Content-Type: application/soap+xml;charset=UTF-8
Content-Length: 474
Date: Tue, 06 Jul 2021 13:49:54 GMT
Keep-Alive: timeout=20
Connection: keep-alive

<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
   <soap:Body>
      <StartWorkflowResponse xmlns="http://primesign.at/workflow/v1">
         <WorkflowInstanceId>9d87e9e7-f8c6-4b4e-ad01-
03e74ff8054e</WorkflowInstanceId>
         <DocumentRef>
            <DocumentId>80e776f1-f705-4d13-a7d2-65d19bb8626b</DocumentId>
         </DocumentRef>

<UserInterfaceURI>http://<EXT_HOST_NAME>/primesign/resume/wf/9d87e9e7-f8c6-
4b4e-ad01-03e74ff8054e</UserInterfaceURI>
      </StartWorkflowResponse>
   </soap:Body>
</soap:Envelope>
```

4.6. Operations

PSS supports following SOAP operations:

- **Primary operations**
  - `StartWorkflow`
  - `GetWorkflowResult`
- **Further operations**
  - `GetWorkflowStatus`
  - `RemoveWorkflow`
  - `ExecuteWorkflow` (see [2])
  - `UploadDocument` (**deprecated**, use `StartWorkflow` instead)
  - `DownloadDocument` (**deprecated**, use `GetWorkflowResult` instead)

In the following section the usage of those operations will be explained giving detailed information and examples. Keep in mind that the following requests are fragments for better understanding (e.g. binaries are not fully printed).

4.6.1. StartWorkflow

The operation `StartWorkflow` creates and starts a PrimeSign workflow and can contain following elements:

- **Workflow name** (*deprecated*)
- **Document(s) or Document reference(s)**
- **Workflow lifetime** (see Workflow with expiration)
- **Workflow properties** (see Predefined workflow properties)
- **UI customizations** (see Modify UI workflow buttons)
- **Events** (see Integration with user interaction)
- **Secret** (*deprecated*)

The operation requires at least one document. This document can be

- either included within the request (via MTOM or base64 encoded data)
- or referenced in case the document has been uploaded beforehand with the operation UploadDocument.

It's also possible to provide more than one document in a single request, see Bulk Signature for details.

Workflow specific aspects may be influenced by providing certain **workflow properties** (key/value pairs, reflecting certain settings like the signature profile to be used). See Workflow Properties for detailed information.

Example Request

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:v1="http://primesign.at/workflow/v1">
   <soap:Header/>
   <soap:Body>
      <v1:StartWorkflowRequest>
         <v1:DocumentOrRef> ❶
            <v1:Document> ❷
               <v1:Filename>test.pdf</v1:Filename>
               <v1:MediaType>application/pdf</v1:MediaType>
               <v1:Content>cid:1139767465325</v1:Content>
            </v1:Document>
         </v1:DocumentOrRef>
      </v1:StartWorkflowRequest>
   </soap:Body>
</soap:Envelope>
```

❶ The StartWorkflow request may contain multiple `DocumentOrRef` elements each holding a single document or document reference.

❷ `Document` represents the actual document to be signed which holds following **required** elements:

- **Filename** (e.g. `test.pdf`)
- **MediaType** according to RFC 2046 (e.g. `application/pdf`)
- **Content** `cid:1139767465325` references the `Content-ID` of an additional mime part with the binary data. Refer to MTOM Example.

Optionally the following properties can be set:

- **DocumentType** (*deprecated*)
- **DocumentProperty** (see Document Properties for details)

If a document has been separately uploaded with the operation UploadDocument the resulting document reference (here `b15a3c92-44a5-45ca-8ec2-b10dd747848d`) can be used to reference this document:

```
...
<v1:DocumentOrRef>
   <v1:DocumentRef>
      <v1:DocumentId>b15a3c92-44a5-45ca-8ec2-b10dd747848d</v1:DocumentId>
   </v1:DocumentRef>
</v1:DocumentOrRef>
...
```

Example Response

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
   <soap:Body>
      <StartWorkflowResponse xmlns="http://primesign.at/workflow/v1">
         <WorkflowInstanceId>efa2aee4-2ec6-4591-a74b-
908d3fbe5cd3</WorkflowInstanceId> ❶
         <DocumentRef>
            <DocumentId>ccbe8fba-07c5-414b-9514-9f7ee01f1e9a</DocumentId> ❷
         </DocumentRef>
         <UserInterfaceURI>https://
<EXT_HOST_NAME>/primesign/resume/wf/efa2aee4-2ec6-4591-a74b-
908d3fbe5cd3</UserInterfaceURI> ❸
      </StartWorkflowResponse>
   </soap:Body>
</soap:Envelope>
```

❶ shows the resulting `WorkflowInstanceId` which can later be used to reference this workflow instance.

❷ shows the `DocumentId`, reflecting the document that has been associated with the workflow.

❸ a `UserInterfaceURI` which is the url the user should be redirected to in order to conduct the signature.

4.6.2. GetWorkflowResult

This operation is used in order to retrieve the results of a **finished** workflow.

A workflow is considered finished (not necessarily successful) if

- a user completes the workflow by clicking `CompleteButton` after conducting the signature or if
- a user cancels the workflow by clicking the `CancelButton` button (see Section 3.1.5.5) or if
- an error occurred (see Error Codes)

Required elements:

- **WorkflowInstanceId**: Reference to the respective workflow. This reference has been obtained with the operation `StartWorkflow`.
- **Secret** (*deprecated*)

Optional elements:

- **TimeOut** (*deprecated*): Depending on the respective PSS configuration incoming

GetWorkflowResultRequests referencing NOT YET finished workflows may either immediately receive a WorkflowFault or may be blocked until the workflow finally reaches a finished state (or a timeout occurs). Note that the default timeout is one hour. The `TimeOut` is realized as datatype `xsd:duration` and specifies a time interval as `PnYnMnDTnHnMnS` whereupon

- `P` indicating the period.
- `nY`, `nM`, `nD` has to be replaced by a certain value, e.g. `1Y` for "one-year" etc. (`Y` means years, `M` means months, `D` means days)
- `T` indicating the start of the time section.
- `nH`, `nM`, `nS` has to be replaced by a certain value, e.g. `1H` for "one-hour" etc. (`H` means hours, `M` means minutes, `S` means seconds)

Optional attributes:

- **resultType**: Specifies how the signed document(s) are returned to the caller. Possible values:
  - `Document` (**default**) returns the binary content of the signed document(s).
  - `DocumentRef` returns a list of document identifiers that reference the signed document(s).

Example Request

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:v1="http://primesign.at/workflow/v1">
   <soap:Header/>
   <soap:Body>
      <v1:GetWorkflowResultRequest>
         <v1:WorkflowInstanceId>efa2aee4-2ec6-4591-a74b-
908d3fbe5cd3</v1:WorkflowInstanceId>
      </v1:GetWorkflowResultRequest>
   </soap:Body>
</soap:Envelope>
```

Depending on the type of workflow the result may contain

- a single (newly) signed document (in case of Single Signature) or
- a list of (newly) signed documents (in case of Bulk Signature).

Example Response

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
   <soap:Body>
      <GetWorkflowResultResponse xmlns="http://primesign.at/workflow/v1">
         <DocumentOrRef>
            <Document> ❶
               <Filename>test.pdf</Filename>
               <MediaType>application/pdf</MediaType>
               <Content>
                  <xop:Include href="cid:e6375f23-18de-409e-b245-
4e9f630c4aab-60@primesign.at"
xmlns:xop="http://www.w3.org/2004/08/xop/include"/>
               </Content>
            </Document>
         </DocumentOrRef>
         <DocumentOrRef>
            <DocumentRef> ❷
               <DocumentId>ad9aa9a4-6df8-4851-a491-e66bdfd5ef38</DocumentId>
            </DocumentRef>
         </DocumentOrRef>
      </GetWorkflowResultResponse>
   </soap:Body>
</soap:Envelope>
```

❶ The signed document is included in the response. This example shows the response of the MTOM approach, where the the `<Content>` holds a reference to an additional mime part (with `Content-ID` < `e6375f23-18de-409e-b245-4e9f630c4aab-60@primesign.at` >).

❷ The original (unsigned) document is referenced with a `DocumentId`. (See DownloadDocument for retrieving the corresponding document)

Example Request with ResultType DocumentRef

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:v1="http://primesign.at/workflow/v1">
   <soap:Header/>
   <soap:Body>
      <v1:GetWorkflowResultRequest resultType="DocumentRef">
         <v1:WorkflowInstanceId>efa2aee4-2ec6-4591-a74b-
908d3fbe5cd3</v1:WorkflowInstanceId>
      </v1:GetWorkflowResultRequest>
   </soap:Body>
</soap:Envelope>
```

Example Response with 1 DocumentRef

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
   <soap:Body>
      <GetWorkflowResultResponse xmlns="http://primesign.at/workflow/v1">
         <DocumentOrRef>
            <DocumentRef> ❶
               <DocumentId>341eb7bd-4094-4f57-9f4c-527f68154f28</DocumentId>
            </DocumentRef>
         </DocumentOrRef>
      </GetWorkflowResultResponse>
   </soap:Body>
</soap:Envelope>
```

❶ The `DocumentId` of the signed document.

Example Response with multiple DocumentRefs

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
   <soap:Body>
      <GetWorkflowResultResponse xmlns="http://primesign.at/workflow/v1">
         <DocumentOrRef>
            <DocumentRef> ❶
               <DocumentId>341eb7bd-4094-4f57-9f4c-527f68154f28</DocumentId>
            </DocumentRef>
         </DocumentOrRef>
         <DocumentOrRef>
            <DocumentRef> ❷
               <DocumentId>28f8a73e-6e2e-11ec-90d6-0242ac120003</DocumentId>
            </DocumentRef>
         </DocumentOrRef>
         <DocumentOrRef>
            <DocumentRef> ❸
               <DocumentId>2e055b0a-6e2e-11ec-90d6-0242ac120003</DocumentId>
            </DocumentRef>
         </DocumentOrRef>
      </GetWorkflowResultResponse>
   </soap:Body>
</soap:Envelope>
```

❶ The `DocumentId` of the first signed document. Beware that the order of the `DocumentId` values is identical to the order of the documents uploaded via `StartWorkflow`.

❷ The `DocumentId` of the second signed document.

❸ The `DocumentId` of the third signed document.

If the workflow has **not (yet) been successfully completed** when the operation `GetWorkflowResult` is called, e.g.

- if the user has not yet signed,
- an error occurred or
- the user cancelled the process

a `WorkflowFault` like shown below is returned.

Example WorkflowFault Response

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
   <soap:Body>
      <soap:Fault>
         <soap:Code>
            <soap:Value>soap:Receiver</soap:Value>
         </soap:Code>
         <soap:Reason>
            <soap:Text xml:lang="en">Aborted by user.</soap:Text>
         </soap:Reason>
         <soap:Detail>
            <WorkflowFault xmlns="http://primesign.at/workflow/v1">
               <code>1</code>
               <message>Aborted by user. (tx=b9fe270)</message>
            </WorkflowFault>
         </soap:Detail>
      </soap:Fault>
   </soap:Body>
</soap:Envelope>
```

Besides the **error code** a **message with a transaction id** (`tx`) is provided which can be logged by the integrating application.

4.6.3. GetWorkflowStatus

The operation `GetWorkflowStatus` requests information about the state of a specific workflow (referenced via `WorkflowInstanceId`). The response contains the state of the referenced workflow or a **WorkflowFault** in case the referenced workflow does not exist.

The workflow will be deleted if the Workflow lifetime expires (see Workflow Lifetime). In this case a `WorkflowFault` will be responded since the workflow does not exist any more.

For requesting the status of a certain workflow the corresponding `WorkflowInstanceId` must be provided.

Example Request

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:v1="http://primesign.at/workflow/v1">
   <soap:Header/>
   <soap:Body>
      <v1:GetWorkflowStatusRequest>
         <v1:WorkflowInstanceId>efa2aee4-2ec6-4591-a74b-
908d3fbe5cd3</v1:WorkflowInstanceId>
      </v1:GetWorkflowStatusRequest>
   </soap:Body>
</soap:Envelope>
```

Example Response

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
   <soap:Body>
      <GetWorkflowStatusResponse xmlns="http://primesign.at/workflow/v1">
         <WorkflowStatus>COMPLETED</WorkflowStatus>
      </GetWorkflowStatusResponse>
   </soap:Body>
</soap:Envelope>
```

A `WorkflowStatus` can show one of the following status values:

- `STARTED`
  The workflow has been started, but is not yet completed. The workflow's result is not available yet.
- `CANCELLED`
  The workflow has been cancelled.
- `COMPLETED`
  The workflow has been completed. The workflow's result can be retrieved using the operation GetWorkflowResult.

4.6.4. RemoveWorkflow

The `RemoveWorkflow` operation stops and removes a certain workflow. Documents bound to this workflow instance will be removed provided that they are not being used by other workflows. (Documents explicitely shared between multiple workflows will not be removed until the last referencing workflow expires or is being removed).

For removing a certain workflow the corresponding `WorkflowInstanceId` must be provided.

Example Request

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:v1="http://primesign.at/workflow/v1">
   <soap:Header/>
   <soap:Body>
      <v1:RemoveWorkflowRequest>
         <v1:WorkflowInstanceId>efa2aee4-2ec6-4591-a74b-
908d3fbe5cd3</v1:WorkflowInstanceId>
      </v1:RemoveWorkflowRequest>
   </soap:Body>
</soap:Envelope>
```

Example Response

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
   <soap:Body/>
</soap:Envelope>
```

4.6.5. UploadDocument

Deprecated, use `StartWorkflow` instead.

The operation `UploadDocument` transmits a single document for later use with workflow(s). It returns a reference (`DocumentRef`) to the uploaded document. The reference may be used when creating workflows using the StartWorkflow operation.

The document will be automatically removed after one hour if not being used within a workflow.

The following example shows a request uploading a document using the MTOM approach. The document is provided by an additional mime part (not shown here) with the `Content-ID <1277721795378>`.

Example Request

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:v1="http://primesign.at/workflow/v1">
   <soap:Header/>
   <soap:Body>
      <v1:UploadDocumentRequest>
         <v1:Document> ❶
            <v1:Filename>test2.pdf</v1:Filename>
            <v1:MediaType>application/pdf</v1:MediaType>
            <v1:Content>cid:1277721795378</v1:Content>
         </v1:Document>
      </v1:UploadDocumentRequest>
   </soap:Body>
</soap:Envelope>
```

- **Filename** (e.g. `test2.pdf`)
- **MediaType** according to RFC 2046 (e.g. `application/pdf`)
- **Content** `cid:1277721795378` references the `Content-ID` of an additional mime part with the binary data. Refer to MTOM Example.

Optionally the following properties can be set:

- **DocumentType** (*deprecated*)
- **DocumentProperty** (see Document Properties for details)

Example Response

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
   <soap:Body>
      <UploadDocumentResponse xmlns="http://primesign.at/workflow/v1">
         <DocumentRef>
            <DocumentId>b15a3c92-44a5-45ca-8ec2-b10dd747848d</DocumentId>
         </DocumentRef>
      </UploadDocumentResponse>
   </soap:Body>
</soap:Envelope>
```

4.6.6. DownloadDocument

Deprecated, use `GetWorkflowResult` instead.

The operation `DownloadDocument` delivers a document corresponding to a specific `DocumentId`.

If a signed document should be received from a workflow operation (e.g. StartWorkflow), use GetWorkflowResult operation instead!

Example Request

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:v1="http://primesign.at/workflow/v1">
   <soap:Header/>
   <soap:Body>
      <v1:DownloadDocumentRequest>
         <v1:DocumentRef>
            <v1:DocumentId>b15a3c92-44a5-45ca-8ec2-
b10dd747848d</v1:DocumentId>
         </v1:DocumentRef>
      </v1:DownloadDocumentRequest>
   </soap:Body>
</soap:Envelope>
```

The element `DownloadDocumentRequest` holds a `DocumentRef` section which requires a `DocumentId` to request the download of a specific document.

Example Response

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
   <soap:Body>
      <DownloadDocumentResponse xmlns="http://primesign.at/workflow/v1">
         <Document>
            <Filename>test2.pdf</Filename>
            <MediaType>application/pdf</MediaType>
            <Content>
               <xop:Include href="cid:e6375f23-18de-409e-b245-4e9f630c4aab-
60@primesign.at" xmlns:xop="http://www.w3.org/2004/08/xop/include"/>
            </Content>
         </Document>
      </DownloadDocumentResponse>
   </soap:Body>
</soap:Envelope>
```

As result the requested document and its meta information (`DocumentType`, `Filename`, `MediaType`) is delivered.

This example shows the response of the MTOM approach, where the the `<Content>` holds a

reference to an additional mime part (with `Content-ID` `<e6375f23-18de-409e-b245-4e9f630c4aab-60@primesign.at>`).

4.7. Properties

The PrimeSign WorkflowService offers two types of properties:

- WorkflowProperties
- DocumentProperties

While WorkflowProperties affect whole workflows (including all their related documents) providing means for workflow customization, DocumentProperties only concern a specific (single) document (which may be part of a workflow).

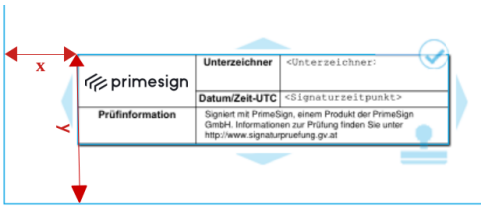
4.7.1. Predefined workflow properties

Setting WorkflowProperties allows to influence the graphical representation, the functional behaviour and the semantic integration of a workflow.

The following table shows predefined workflow properties supported by PSS:

Name	(Allowed) Value(s)	Description
`interfaceType`	`full`, `minimum`, `non-interactive` (Default: auto detection)	This property sets the type of the graphical user interface. Choosing `full` enables the "rich UI" (optimized for desktop devices) and `minimum` the "minimum UI" (optimized for mobile devices). `non-interactive` allows to sign documents without displaying a PSS user interface and can only be used by authorized Registration Authorities of the primesign TRUST CENTER. If no interfaceType is specified, PSS inspects the browser's user agent string to choose a suitable interfaceType. PSS automatically selects the optimized UI type based on the underlying device type (`full` for desktop and `minimum` for mobile devices and tablets). Please note, that interface type `full` requires cookies. The primesign SIGNATURE SERVER application must be allowed to set cookies within the user's browser. Furthermore, interface type `full` cannot be selected in combination with bulk signatures (selecting `signatureMode`: `bulk` implicitly selects `interfaceType`: `minimum`). `non-interactive` allows to perform one-time signatures without user interaction. `non-interactive` can only be used for one-time signing by authorized Registration Authorities of the primesign TRUST CENTER (requires signer authorization with level SCAL2). In case `non-interactive` is chosen, the WorkflowProperties `artefact` and `transactionId` are mandatory. Non-interactive one-time signatures are only supported by signature creation device `pcss`.

Name	(Allowed) Value(s)	Description
`signatureProfileId`	**<PROFILE_ID>** (Default: default profile, depending on the instance configuration)	The identifier of the signature profile to be used. When using the Rich UI for signing, the user is still able to switch to another signature profile. If the Minimal UI is used, the user cannot change the preselected signature profile. Beware to correctly set the acl entries for the signature profile via the primesign SIGNATURE SERVER Admin UI, see also Section 3.1.4.3.

Name	(Allowed) Value(s)	Description
`signaturePosition`	**[p:integer][;x:float;y:float][end][;f:float]** or `invisible` (Default: auto positioning)	This property defines the position of the visible signature or sets the signature invisible. The visible position requires at least a page number (starting with 1) or `end` flag. In case X/Y-position is omitted the signature is auto-positioned within the specified page.  *Figure 4. Choosing the signature position.* A page with A4 (height of 297mm, width of 210mm) dimension and a default pixel density of 72dpi corresponds to a dimension (in points) of 842pt x 595pt. EXPERIMENTAL `end:` - Indicates that, in case of autopositioning, only a signatureposition at the end of the page (beneath all other rendered PDF-Elements on the same page) is allowed. It can be used in combination with a specific page (e.g.: `p:7;end`) or standalone. If it is used without a specific page, the last page will be used for autopositioning. If there is no space left on the last page, a new page will be created. It is not allowed to use the flag in combination with a fixed signature position (`x`, `y`). **Exception**: It is possible to provide a footer property `f:` that defines a footer-space at the end of the page, which will be ignored in combination with `end` (e.g.: `p:1;f:100;end`). In this case, a automatic signatureposition above all rendered PDF-Elements in the defined footer-space is possible. The usage of the `f:` property is only allowed in combination with `end`.

Name	(Allowed) Value(s)	Description
		Examples: • EXPERIMENTAL `end` • `p:7` • EXPERIMENTAL `end;f:100` • `p:1;x:100;y:200` • EXPERIMENTAL `p:7;end` • EXPERIMENTAL `p:7;end;f:100` • `invisible` The signature position declared using this "WorkflowProperty" applies to all documents within this workflow. The signature position can also be specified per document using the respective DocumentProperty `signaturePosition`. It is not supported to mix `signaturePosition` as DocumentProperty and WorkflowProperty in the same request.

Name	(Allowed) Value(s)	Description
`signatureCreationEntities`	`pcss` primesign MOBILE 2.0 `pcssWraptor` primesign WRAPTOR 2.0 Remark: This option is only available, if activated explicitely by the administrator. See Appliance Documentation section 5.8 or contact the administrator. `egofyMW` primesign MOBILE 1.1 (*deprecated*) `wraptor` primesign WRAPTOR 1.1 (*deprecated*) `onboarding` PSS offers an entry point to an external onboarding service for users who do not yet own a (qualified) certificate. `mobileSignature` ID Austria / Austrian Mobile Phone Signature ("Handy-Signatur"). Beware, the public API of the ID Austria / Austrian Mobile Phone Signature does not support bulk signing! `atrustSS` ID Austria / Austrian Mobile Phone Signature (Signature-Box). `localCCS` Locally installed signature creation environment like trustDesk Basic, MOCCA, etc.	This property allows to restrict the list of available/selectable signature creation entities for this workflow. e.g. `pcss`, `mobileSignature` • At least one entity must remain available (meaning that the intersection of available entities from instance configuration and this list must not be empty). • In case only one single entity remains selectable for the user the selection dialogue will be skipped.

Name	(Allowed) Value(s)	Description
`signatureMode`	`single`, `bulk` (Default: `single`)	Enables bulk signature (`bulk`) for this workflow or uses default (`single`) signature. primesign MOBILE and primesign WRAPTOR allow to sign a maximum of 30 documents in one bulk signature request. Please note that bulk signatures cannot be used in combination with `interfaceType`: `full`. Furthermore, the public API of the ID Austria / Austrian Mobile Phone Signature does not support bulk signing.

Name	(Allowed) Value(s)	Description
`signaturePlaceholders`	`ignore` Ignores placeholders within the document. `detect` Enables scanning a document's image(s) for signature placeholder(s), deriving desired signature positions(s). `detectAndRemove` Enables scanning like with "detect" but additionally removes detected placeholder image(s) from the document.	This property configures if and how PSS scans the document for certain QR codes, remembers the respective QR code positions within the document and removes the QR code images.  *Figure 5. Sample placeholder for signature replacement. Contact PrimeSign GmbH for further examples [3].* Refer to use case "Use QR code image in document as position placeholder" for further information. **Remarks:** • If placeholder detection is disabled within the PSS configuration, but enabled within `StartWorkflowRequest` document(s) are only scanned for placeholder if provided as `Document` and not as `DocumentRef`. • With the recommended default configuration signed documents will not be scanned for QR codes. Furthermore, QR code images will not be removed when the document is already signed since signed documents must not be modified. • If the signature profile used for signing is larger than the placeholder image document content may be hidden. See [4] for guidance how to create documents with placeholder.

Name	(Allowed) Value(s)	Description
`archiveInstanceId`	**<INSTANCE_ID>** (Default: depends on instance configuration)	Sets a certain archive for archiving signed documents together with verification reports.
`locale`	**<BCP-VALUE>** Default: depends on instance configuration)	This property can be used to set the UI locale for a certain workflow. Refer to use case "Preselect UI language" for further information. Allowed values are IETF BCP 47 language tag strings. **Examples:** • `en` • `de-AT` • `el-GR`
`transactionId`	**<TRANSACTION_ID>**	This property can be used to pass transaction identifiers from the integrating application context to PSS for logging and billing purposes. Mandatory in case of signatures with interfaceType `non-interactive`. Allowed characters: `a-z A-Z 0-9 _ @ - + . :` Maximal length: 200 characters.

Name	(Allowed) Value(s)	Description
`pdfACompliance`	`none`, `PDF/A-1a`, `PDF/A-1b` (Default: `none`)	This property can be used to specify the desired PDF/A conformance level for document conversion. Beware, this is only relevant in case a non-PDF document (e.g. Word, Excel etc.) is uploaded. PSS automatically converts these documents to PDF. The values `PDF/A-1a` or `PDF/A-1b` are recommended only in combination with PDF/A enabled signature profiles. This ensures that PDF documents remain PDF/A compliant after signature creation. Furthermore, the desired PDF/A conformance level can also be configured globally. Any values specified via `StartWorkflowRequest` overwrite the global configuration. Please contact your administrator in case of questions.
`accountId`	**<ACCOUNT_ID>**	This property can be used to pass an account identifier from the integrating application context to PSS for billing purposes. Please contact PrimeSign GmbH for obtaining an account id.
`artefact`	**<ARTEFACT_VALUE>**	References a preceding signer authorization (identity assertion). Required for one-time signing. Can only be used by authorized Registration Authorities of the primesign TRUST CENTER.

4.7.2. Predefined document properties

A DocumentProperty contains specific document settings and metadata.

Name	(Allowed) Value(s)	Description
`signaturePosition`	**p:integer[;x:float;y:float]** or `invisible` (Default: value from workflow property `signaturePosition` (if any) or auto positioning)	This property allows to define the `signaturePosition` on a document level. The corresponding workflow property defines the signature position on a workflow level, thus for *all* documents within the workflow. **Unsupported: It is not supported to mix `signaturePosition` as DocumentProperty and WorkflowProperty in the same request.** Refer to use case "Enforce signature position" for further information.

Table 2. Predefined document properties

4.8. Error codes

In case of error the primesign WorkflowService returns a `WorkflowFault` response like shown below.

This `WorkflowFault` contains an error `code` (`xsd:int`) and an error `message` (`xsd:string`). While the error `message` is intended to be used for logging purposes the error `code` can be used by the integrating application to adjust its process and in order to provide a suitable user message.

Example for a WorkflowFault returned after requesting the result of a non-existent workflow

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
   <soap:Body>
      <soap:Fault> ❶
         <soap:Code>
            <soap:Value>soap:Receiver</soap:Value>
         </soap:Code>
         <soap:Reason>
            <soap:Text xml:lang="en">Unable to return workflow
result.</soap:Text>
         </soap:Reason>
         <soap:Detail>
            <WorkflowFault xmlns="http://primesign.at/workflow/v1">
               <code>7</code> ❷
               <message>Unable to find workflow instance 'e2ee039b-85d5-4e92-
8196-0b9707287bac' with transaction context. The workflow needs to be created
using StartWorkflowRequest.</message> ❸
            </WorkflowFault>
         </soap:Detail>
      </soap:Fault>
   </soap:Body>
</soap:Envelope>
```

❶ In case of an error, a `soap:Fault` is returned.

❷ Containing an error `code` which can be used to provide a suitable message for the user. See Table 3 for specific error codes and their meanings.

❸ A `message` for logging purposes.

The following list reflects all possible error codes and their meaning of the primesign WorkflowService.

Error Code	Description
1	Indicates that the workflow instance has been cancelled by the user.
2	Indicates a non specific error that ocurred when processing the document.
3	Indicates an I/O error while preprocessing and storing the document.
4	Indicates that the document preprocessing could not be done successfully.

Error Code	Description
5	Indicates that the workflow could not be created.
6	Indicates that the respective document cannot be retrieved.
7	Indicates that the respective workflow cannot be retrieved.
8	Indicates that the respective workflow that should be returned as signed is not signed.
9	Indicates that the respective workflow could not be removed.
10	Indicates that the respective workflow has not yet been started.
11	Indicates that a timeout occurred waiting for the respective workflow result.
12	Indicates that the feature has not been implemented yet.
13	Indicates that the request references a workflow name that is not supported.
14	Indicates that the request is invalid regarding the selected workflow.
15	Indicates that the provided file format is not supported.
16	Indicates that the request contains invalid data. Refer to the specific error message provided by the SOAP response.
17	Indicates that the invoker does not have appropriate rights to perform the desired action.
18	Indicates that a secret is missing for the respective workflow.

Error Code	Description
19	Indicates that a given secret is incorrect for the respective workflow.
20	Error code for a situation where the caller tried to retrieve a workflow's result while the result is not yet available.
21	The primesign IDENTITY PROVIDER reported an error. Please contact PrimeSign GmbH [3].
22	Indicates that an unspecific error occurred while signing the workflow.
23	There was either no `accountId` provided with the request, no account id associated with the user or the account has no credits left.
24	The provided artefact has expired, has already been consumed or the SCAL2 lifetime has exceeded.

Table 3. Specific error codes

5. SignaturService

This webservice provides software keystore or hsm based signatures (without involving any user interaction).

Please note this service uses German identifiers and names, e.g. "*Signatur*" instead of "*Signature*" or "*Dokument*" instead of "*Document*".

5.1. General requirements

- In order to use a keystore/hsm signature the respective signature profile must be equipped with either a software keystore or must be associated with a HSM partition/key handle (see [1] for details).
- This service requires requests to use the optimized transmission method (**MTOM/XOP**) for transmission of binary content.

5.2. Endpoint

- **SignaturService**
  - Endpoint: https://<EXT_HOST_NAME>/primesign/services/signaturService/v1
  - WSDL: https://<EXT_HOST_NAME>/primesign/services/signaturService/v1?wsdl

5.3. Operation "dokumentSignieren"

The operation `dokumentSignieren` signs a **single** document without requiring any user interaction (server-side signature).

The request may contain following elements:

- `signaturParameter`: *optional* (for a detailed list of signature parameters refer to SignaturParameter)
- `dokument`: ***required*** (document)
  - `dateiname`: *optional* (document file name)
  - `medientyp`: *optional* (media type)
  - `daten`: ***required*** (file content)
- `benutzer`: *optional* (user) (for further details refer to Benutzer)

- `vollerName`: *optional* (full username)

The request must contain a **single** document to be signed. It is recommended to provide the identifier of the signature profile (`signaturParameter` → `profilId`) like shown below:

Example Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:v1="http://www.dt-i.at/primesign/services/signaturService/v1"
xmlns:v11="http://www.dt-i.at/primesign/types/v1">
   <soapenv:Header/>
   <soapenv:Body>
      <v1:dokumentSignieren schemaVersion="1.1"> ❶
         <v1:signaturParameter>
            <v11:profilId>MY_SIGNATURE_PROFILE_ID</v11:profilId> ❷
         </v1:signaturParameter>
         <v1:dokument>
            <v11:medienTyp>application/pdf</v11:medienTyp>
            <v11:daten>cid:173241990224</v11:daten> ❸
         </v1:dokument>
      </v1:dokumentSignieren>
   </soapenv:Body>
</soapenv:Envelope>
```

❶ Setting the `schemaVersion` to **1.1** is **required**.

❷ It's recommended to set a signature profile (`profilId`).

❸ The element `daten` contains a reference to an additional mime part with `Content-ID <173241990224>` containing the binary document (mime part with binary data not shown here).

Example Response

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
      <ns2:dokumentSignierenResponse schemaVersion="1.1"
xmlns:ns3="http://www.dt-i.at/common/types/v1" xmlns:ns2="http://www.dt-
i.at/primesign/services/signaturService/v1" xmlns="http://www.dt-
i.at/primesign/types/v1">
         <ns2:dokument>
            <medienTyp>application/pdf</medienTyp>
            <daten> ❶
               <xop:Include href="cid:3b7dee7d-9f7f-4c9f-a0c4-699e36079863-
4@www.dt-i.at" xmlns:xop="http://www.w3.org/2004/08/xop/include"/>
            </daten>
         </ns2:dokument>
      </ns2:dokumentSignierenResponse>
   </soap:Body>
</soap:Envelope>
```

❶ The signed document is included within the `daten` element as reference to an additional mime part with `Content-ID` < `3b7dee7d-9f7f-4c9f-a0c4-699e36079863-4@www.dt-i.at` > (mime part and binary document not shown here).

5.4. Request parameters

5.4.1. SignaturParameter

A signature parameter (`signaturParameter`) can be set in order to modify the type of signature, the graphical representation and the position of the visual signature.

Name	(Allowed) Value(s)	Description
`profilId`	**<PROFILE_ID>** (Default: default profile; depending on the instance configuration)	The identifier of the signature profile which should be used. It is recommended to set this optional element in order to avoid dependency to any default profile.
`typ`	`PAdES` (Default: `PAdES`)	Sets the type of the signature. • `PAdES` A PAdES compliant server-side signature will be created (keystore/hsm).

Name	(Allowed) Value(s)	Description
`zertifizierungsStufe`	`NOT_CERTIFIED`, `CERTIFIED_NO_CHANGES_ALLOWED`, `CERTIFIED_FORM_FILLING`, `CERTIFIED_FORM_FILLING_AND_ANNOTATIONS` (Default: `NOT_CERTIFIED`)	Sets the certification level of the signature. Certification signatures are used to specify the allowed changes (see list below) in a document. This is usually done by the author of a document. Certification signatures may only be applied to unsigned documents. • `NOT_CERTIFIED` - No certification. Modifications of the document are possible. Only parts of the document might be signed. • `CERTIFIED_NO_CHANGES_ALLOWED` - The document is final, no modifications are allowed. • `CERTIFIED_FORM_FILLING` - Only form fields can be modified. • `CERTIFIED_FORM_FILLING_AND_ANNOTATIONS` - Form fields can be modified and annotations can be added.
`sichtbareSignatur`	`true`, `false` (Default: `true`)	Defines if the signature should be visible (default) or invisible.

Name	(Allowed) Value(s)	Description
`position` • `seite` (page) • `x` • `y` • `breite` (width, deprecated) • `hoehe` (height, deprecated) • `ende` • `fusszeile`	• `seite` : int | `neu` • `x` : int (pt) • `y` : int (pt) • `breite` : int (pt) • `hoehe` : int (pt) • EXPERIMENTAL `ende` : boolean • EXPERIMENTAL `fusszeile` : int (pt)	If auto positioning (usually on the last page of the document) should be used, omit the parameter `position`. If placeholder detection is enabled within the primesign SIGNATURE SERVER configuration, *all* uploaded documents will be scanned for signature placeholders and the visual representation of the signature will be placed on the first placeholder. Beware, the parameter `position` has higher priority: If a `position` is set explicitely, placeholders will be ignored. Summarizing, the priority of the signature positions is defined as follows: • parameter `position` • document placeholder(s) • auto positioning **Defining the Signature Position:** The visual representation of a signature can be placed on a page (`seite`) starting with 1. If no page is chosen the signature will automatically be placed on the last page. If not enough visual space is available a new page at the end of the document will be added holding the signature. It is also possible to force this behaviour by choosing the keyword `neu` instead of a page number. The origin of the common coordinate system used within PDF-files is located on the bottom left corner of a page and is measured in points. Depending on the pixel density (dpi) and size of a page a position can be figured.

Name	(Allowed) Value(s)	Description
		A page with A4 (height of 297mm, width of 210mm) dimension and a default pixel density of 72dpi corresponds to the dimension (in points) of 842pt x 595pt. `x` - distance between the left edge of the page and the left edge of the visual signature `y` - distance between the lower edge of the page and the upper edge of the visual signature `breite` - width of the visual signature, deprecated since already provided by the selected signature profile `hoehe` - height of the visual signature, deprecated since already provided by the selected signature profile EXPERIMENTAL `ende` - Indicates that, in case of autopositioning, only a signatureposition at the end of the page (beneath all other rendered PDF-Elements on the same page) is allowed. `ende = true` can be used in combination with a specific page (e.g.: `p:1;end`) or standalone. If it is used without a specific page, the last page will be used for autopositioning. If there is no space left on the last page, a new page will be created. It is not allowed to use the flag in combination with a fixed signature position (`x`, `y`). EXPERIMENTAL `fusszeile`: This property defines a footer-space at the end of the page, which will be ignored in combination with `ende = true` (e.g.: `p:1;f:100;end`). In this case, a automatic signaturposition above all rendered PDF-Elements in the defined footer-space is possible. The usage of the `fusszeile` property is only allowed in combination with `ende = true`.

Name	(Allowed) Value(s)	Description
`signaturZeitpunktFormat`	**'SOME_TEXT' <DATE_TIME_PATTERN>**	Defines a time format which should be used within the visual signature. See SimpleDateFormat for detailed information. **Example:** `yyyy.MM.dd HH:mm:ssXXX` results in: `2021.01.17 16:00:01+01:00`
`hinweisText`	**<NOTE>**	Adds an additional text field to the visual signature representation, usually used for information about signature validation posibilities as `Notes`.
`anlass`	**<REASON>**	Sets a reason for the signature. The reason is displayed in the signature properties of a PDF compliant reader.
`signaturName`	**<SIGNATURE_NAME>**	Sets a name for the signature field. The name of the signature field is displayed in the signature properties of a PDF compliant reader. The signature name must be unique within a document and must not contain any dots (`.`).

Table 4. Signature parameters

5.4.2. Benutzer Parameter

To dynamically add a username to the visual signature, you can optionally include a `benutzer` (user) element in the request. This element requires a `vollerName` parameter containing the

user's full name (i.e., the signer's name). For this to function correctly, an administrator must also pre-configure the referenced profile (referenced by the `profilId` parameter within the `signaturParameter` element). Refer to the Admin UI documentation, section add new signature profile, for additional details.

Example benutzer element

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:v1="http://www.dt-i.at/primesign/services/signaturService/v1"
xmlns:v11="http://www.dt-i.at/primesign/types/v1">
   <soapenv:Header/>
   <soapenv:Body>
      <v1:dokumentSignieren schemaVersion="1.1">
         <v1:benutzer>
            <!-- Name of the user triggering the signature -->
            <v11:vollerName>Max Mustermann</v11:vollerName>
         </v1:benutzer>
         <v1:signaturParameter>
            <v11:profilId>MY_SIGNATURE_PROFILE_ID</v11:profilId>
         </v1:signaturParameter>
         <v1:dokument>
            <v11:medienTyp>application/pdf</v11:medienTyp>
            <v11:daten>cid:173241990224</v11:daten>
         </v1:dokument>
      </v1:dokumentSignieren>
   </soapenv:Body>
</soapenv:Envelope>
```

5.5. Error codes

The primesign SignaturService returns a `soap:Fault` in case an error occurs.

The following example shows the result of an incorrect usage, namely referencing a non existing signature profile with id `myProfileID`.

Example for a soap fault using the dokumentSignieren operation

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
      <soap:Fault>
         <faultcode>soap:Server</faultcode>
         <faultstring>Unable to find signature profile
'myProfileID'.</faultstring> ❶
         <detail>
            <ns2:dokumentSignierenBusinessFault xmlns:ns3="http://www.dt-
i.at/common/types/v1" xmlns:ns2="http://www.dt-
i.at/primesign/services/signaturService/v1" xmlns="http://www.dt-
i.at/primesign/types/v1"> ❷
               <ns3:timestamp>2021-07-30T08:57:29.406+02:00</ns3:timestamp>
❸
               <ns3:transaktionsId>a54c28d</ns3:transaktionsId> ❹
               <ns3:code>421</ns3:code> ❺
               <ns3:text>Es konnte kein Signaturprofil für die Signatur
bestimmt werden.</ns3:text> ❻
            </ns2:dokumentSignierenBusinessFault>
         </detail>
      </soap:Fault>
   </soap:Body>
</soap:Envelope>
```

❶ `faultstring` contains an error message.

❷ `dokumentSignierenBusinessFault` reflects the corresponding fault type. A business fault is a fault caused by the invoking party (could be seen as "bad request"). A system fault (`dokumentSignierenSystemFault`) indicates a technical error.

❸ `timestamp` reflects the time at which the error in question occurred.

❹ `transaktionsId` identifies the transaction in which the error occurred.

❺ `code` represents the number of the respective error.

❻ `text` reflects an optional text that describes the error in more detail.

The values of `faultstring` and `text` are intended to be used for logging purposes the error `code` can be used in order to provide a suitable user message.

The following table provides an overview of the possible error groups:

Code	Error group
1xx	General errors
2xx	Workflow related errors
4xx	Client related errors
5xx	Signature related errors
6xx	License related errors
7xx	Validation related errors

Table 5. Error code groups

Specific error codes, messages (german) and their meanings can be found in the table below:

Code	Message	Description
100	Unbekannter Fehler beim Aufruf von PrimeSign.	This error is representative of an unclassified error, i.e. an "unknown" error.
101	Das Dokument mit der ID <documentId> konnte nicht verarbeitet werden. Das Dokument konnte nicht verarbeitet werden.	This error occurs when the document could not be processed for unspecified reason.

Code	Message	Description
102	Das Webservice ist derzeit deaktiviert und kann nicht verwendet werden.	This error is returned if the webservice has been deactivated, i.e. that no integration support is currently offered.
200	Fehler beim Starten des Workflows.	This error occurs when the underlying workflow could not be started.
201	Fehler beim Upload des Dokuments.	This error occurs when the document could not be uploaded.
202	Fehler beim Download des Dokuments.	This error occurs when the (signed) document could not be downloaded.
203	Das Dokument mit der ID <documentId> wurde nicht akzeptiert. Das Dokument wurde nicht akzeptiert.	The document was not accepted by PSS. This message results from special workflow use cases. Not relevant in the standard version of PSS.
204	Das Dokument mit der ID <documentId> wurde zurückgewiesen. Das Dokument wurde zurückgewiesen.	The document was explicitly rejected. This message results from special workflow use cases. Not relevant in the standard version of PSS.
400	Kein signierbares Dokument gefunden.	This error occurs if the document to be signed was not specified in the request.

Code	Message	Description
401	Das Dokument mit der ID <documentId> ist geschützt und kann nicht signiert werden. Das Dokument ist geschützt und kann nicht signiert werden.	This error occurs when attempting to transmit a protected or encrypted document.
402	Das Dokumentformat des Dokuments mit der ID <documentId> wird nicht unterstützt. Das Dokumentformat wird nicht unterstützt.	This error occurs when the format of the transmitted document is not supported.
407	Das Dokument enthält XFA-Formulare, die derzeit nicht unterstützt werden.	The document was rejected because it contains XFA forms. XFA forms are not supported.
408	Das Dokument enthält dynamische Formular-Elemente. Aus Sicherheitsgründen können Dokumente mit dynamischen Inhalten nicht signiert werden.	The document was rejected because it contains dynamic elements that may affect document view after the signature is applied.
417	Das Dokument enthält XFA Formular-Elemente. Aus Sicherheitsgründen können Dokumente mit XFA-Formular-Inhalten nicht signiert werden.	The document contains XFA form elements. For security reasons, documents with XFA forms cannot be signed.
420	Das angegebene Signaturprofil ist dem verwendeten Benutzerkonto nicht zugeordnet.	The user account named in the signature request must have the authorisation to use the signature profile in question. (considered deprecated)

Code	Message	Description
421	Es konnte kein Signaturprofil für die Signatur bestimmt werden.	No signature profile was referenced in the signature request. The automatic search for a standard profile to be used was not successful. Please specify the signature profile to be used in the request.
422	Der angegebene Signaturname ist ungültig. Der Signaturname darf keinen Punkt enthalten.	The specified signature name is invalid. The signature name must not contain a dot.
423	Der angegebene Signaturname kann nicht verwendet werden. Ein Feld mit diesem Namen existiert bereits.	The specified signature name cannot be used. A field with this name already exists.
424	Zumindest einer der Signaturparameter (z.B. `signaturZeitpunktFormat`) enthält einen ungültigen Wert.	At least one of the signature parameters (e.g. `signaturZeitpunktFormat`) contains an invalid value.
500	Das Dokument konnte nicht signiert werden.	This is an unspecified error.
501	IO-Fehler beim Signieren des Dokuments.	Occurs when the Pdf document could not be read or written (e.g. because it is defective).
503	Allgemeiner Fehler beim Aufruf der Signaturfunktion.	This is an unspecified error.
507	Die Signatur konnte aufgrund eines Konfigurationsproblems nicht erfolgen. Es muss mehr Speicher für die Signaturdaten reserviert werden.	The configuration for PDF-AS based signatures is incomplete.

Code	Message	Description
508	Der Signaturtyp erfordert eine vollständige Angabe der Signaturposition.	The selected signature type does not support automatic positioning.
509	Das Signaturservice kann nicht erreicht werden.	This error occurs if, in the case of a server signature based on an externally connected signature service (e.g. MOA via web service), the service in question cannot be reached.
510	Der Signaturschlüssel konnte nicht geladen werden.	This error indicates that the key associated with the signature profile could not be loaded (e.g. wrong password, etc).
512	Es war nicht möglich, einen Zeitstempel mit der Signatur aufzubringen.	This error indicates that a signature timestamp could not be created.
600	Die Lizenz ist abgelaufen.	The license was limited in time. The validity period has expired. The license must be renewed.
601	Die Lizenz ist noch nicht gültig.	The license is limited in time. The validity period has not yet begun.
602	Die lizensierte Anzahl an unterschiedlichen Dokumenttypen wurde überschritten.	The license contains a limitation of the number of different document types. This number was exceeded.
603	Die lizensierte Anzahl an Signaturen wurde überschritten.	The license contains a limitation of the number of possible signatures. This number has been exceeded.

Code	Message	Description
604	Die lizensierte Anzahl an Benutzern wurde überschritten.	The license contains a limitation of the number of possible users. This number has been exceeded.
605	Die lizensierte Anzahl an Tokens wurde überschritten.	The license contains a limitation of the number of different tokens (cards, mobile signatures, server certificates). This number has been exceeded.

Table 6. Specific error codes, messages and their meanings.

References

- [1] primesign SIGNATURE SERVER - Appliance Documentation
- [2] primeCONVERT - Integration Documentation
- [3] CRYPTAS Support-Center
  https://support.cryptas.com/
  basicsupport@cryptas.com
- [4] primesign SIGNATURE SERVER - Benutzerhandbuch