Reykjavík 13.02.2023

Advania Signing service - Signet

Webservices version 3

TABLE OF CONTENTS

Version history

Date	Version	Description	Author/Approv.
23.09.19	1.0	First version	Sveinbjörn Óskarsson
02.10.20	1.1	Updated version	Sveinbjörn Óskarsson
03.05.22	1.2	Updated version	Sveinbjörn Óskarsson
13.02.23	1.3	Updated version	Sveinbjörn Óskarsson
Url for the document:	https://info.signet.is/files/documentation/Signet_RESTv3.pdf		

Pictures overview

1 SCOPE AND INTRODUCTION

Signet Advania is a signing solution from Advania which has been in development since 2007 using technologies from companies like Ascertia and iText Software.

The solution is constructed from following units:

- Signing web where individuals can load documents for signing and sign documents which are specific for them.
- Web services where companies can load documents for signing, check the status of the documents, send reminders and finally delete them.
- Team web where defined employee teams can load documents in the name of a company for signing.

This document describes the web services and its usage.

1.1 CHANGES IN V.3.4

Version 3.4 of Signet webservices API now has support for signatures via Evrotrust certificates and test(mock) users (see chapter 10) in test environment. There has also been added an attribute in SignDocumentRequest and TokenRequestV3 which defines the certificate device of user for sending the user to Signet website for signing (see 4.28)

2 URL´S

Signet has both preproduction and production environment, hosted at signet.is. The web services and the Signet web both require digital certificates for authentication. For preproduction, Advania can provide certificates for the web services authentication.

2.1 PREPRODUCTION ENVIRONMENT

The preproduction environment is accessible from following URLs:

- Signing and individual web : https://prufa.signet.is
- Signing and individual web on English interface: https://prufa.signet.is/en/
- Logon to document: https://prufa.signet.is/authed/login/<ID_skjals>
- Logon to document on English interface: https://prufa.signet.is/en/authed/login/<ID_skjals>
- Signing with SSO token: https://prufa.signet.is/token/sign
- Signing with SSO token on English interface: https://prufa.signet.is/en/token/sign
- Signing with SSO token with no preview window (all pages displayed): https://prufa.signet.is/token/signsp
- Signing with SSO token with no preview window (all pages displayed) on English interface: https://prufa.signet.is/en/token/signsp
- Web services: https://prufa.signet.is/SignetService2/v3/

2.2 PRODUCTION ENVIRONMENT

Production environment is accessible from following URLs:

- Signing and individual web: https://www.signet.is
- Signing and individual web on English interface: https://www.signet.is/en/
- Logon to document: https://www.signet.is/authed/login/<ID_skjals>
- Logon to document on English interface: https://www.signet.is/en/authed/login/<ID_skjals>
- Signing with SSO token: https://www.signet.is/token/sign
- Signing with SSO token on English interface: https://www.signet.is/en/token/sign
- Signing with SSO token with no preview window (all pages displayed): https://www.signet.is/token/signsp
- Signing with SSO token with no preview window (all pages displayed) on English interface: https://www.signet.is/en/token/signsp
- Web services: https:// www.signet.is/SignetService/v3/

3 AUTHENTICATION

The non token web services require authentication using electronic certificates at the transport layer. The web services also accept username and password to separate between company accounts.

3.1 CERTIFICATE CHAINS

Following certificate chains are trusted for authentication certificate to web services:

- Íslandsrót – Fullgildur búnaður
- Auðkennisrót – Traust auðkenni – Traustur búnaður
- Advania Profun – Bunadarskilriki Profun (Only for preproduction environment)

3.2 ACCESSES

Web methods also use the string username and the string password to separate between accounts. Advania provides this account. Notice that the password can both contain Icelandic letters and symbols.

3.3 AUTHENTICATION PROBLEMS

A common error in authenticating with digital certificates is:

```
The HTTP request was forbidden with client authentication scheme 'Anonymous'.
```

This error is usually saying that you tried to authenticate with a digital certificate but were unsuccessful in applying the private key. Two common causes are:

- The user (app pool etc) does not have permission to use the private key. This can be sorted by giving the user permission on the certificates private key (in Windows right click the certificate in the certificate manager and select manage private keys).
- The certificate chain is incomplete and the machine is not able to correctly select the certificate. This can be sorted by adding the root and intermediate certificates to the certificate

store. The chain for test authentication certificates can be downloaded from https://certs.advania.is

Another common error is:

```
No authentication certificate present
```

This error suggest you are contacting the WSDL endpoint instead of the correct one. See the URL section (chapter 3).

3.4 TOKEN WEB SERVICE INTERFACE AUTHENTICATION

The token web service interfaces introduced in v.2.0 uses api key for authentication to the service. The service requires the header “apikey” with a key provided by Advania. The service supports OPTION requests for ajax integration. Implementations using this interface require an audit from Signet and is not suited for everyone.

3.5 CALLBACK NOTIFICATION

Supported authentication for notifications (callback) sent by Signet to a registered endpoint are:

- Basic authentication
- Digital certificate
- Static api key (added to header "apikey")
- Static access token (added to header "Authorization-Token")
- Static api key and access token (added to header "apikey" and authorization header "Bearer")

4 WEB SERVICE CLASSES

This chapter describes the classes of the Signet web service. *ResultStatus the return value for each function*, *AddDocumentRequest* used to upload a document, *DocumentSigner* description of the signer, *SignetDocumentInfo* status information for a document and *SignetNotification* used to reflect status changes in the web service. Descriptions of the classes, controllers and actions can also found on the webservice help pages, https://www.signet.is/SignetService/v3/Help

4.1 RESULTSTATUS

All the functions of the web service return ResultStatus if request in unsuccessful, which returns an error messages when appropriate. „ResultStatus“ is constructed as following:

```
public enum ResultType { Success, Informational, Warning, Error }

public ResultType Result;

public string StatusCode;

public string Message;
```

The values for StatusCode í ResultStatus are:

- 00 = Success
- 10xx = Info

- 20xx = Warning
- 30xx = Error

4.1.1 SUBCODE

In version 2.3 a subcode is introduced to the StatusCode value. If there is a subcode the StatusCode ends with a dot (.) and a code which further defines the error. See table in appendix for error codes.

4.2 BASEREQUEST

Most requests have the same base request of the form:

- public string username
  - The account username
- public string password
  - The account password

4.3 ADDDOCUMENTREQUEST

AddDocumentRequest is used to upload a document to be signed, the class is defined as follows:

- public byte[] DocumentBytes
  - Byte array for PDF or XML document
- public string DocumentString
  - Base64 coded PDF or XML document (optional)
- public string DocumentName
  - Name of the document, for example „Agreement_123.pdf“.
  - Note that document names longer than 50 characters will be shortened for compliance with Auðkennis mobile certificate system.
- public string DocumentNotes
  - Further description of the document which is accessible for the user, for example „Account agreement“.
- public string Metadata { get; set; }
  - Metadata string used for information which is not accessible for the user but the web service gets with the document.
    - You can add “;returnUrl=<URL>;” if you would like the user to be redirected to this url after signing if he was not authenticated using token. The query parameter **status** is appended to the url with the possible values
      - **signed** if the document was signed
      - **rejected** if the document was rejected
      - **deleted** if the document has been deleted
    - If the document is an XML document you can add “;xslt=<URL for XSLT>;” if you would like an XSLT style to be applied to the document at viewing/signing.
    - If you want to control the text on the rejection window you can add “;rejectMsg=<msg>;” to show a text below the reason input.
    - If you want to control where to send callback on status changes you can add “;callbackUrl=<URL for callback>;” to the metadata.
- public DateTime? StartSigning { get; set; }
  - Initial date, the date when the signing process can start (not necessary)
- public DateTime? EndSigning { get; set; }

- End date when the signing process ends (not necessary)
- public bool Ordered { get; set; }
  - Shall the signing be done in the order of the signers or not (Signer 1 first, then Signer 2 etc).
- public bool Notify { get; set; }
  - Notify when a document is deleted – not used at this time
- public bool Archive { get; set; }
  - Should the document be archived after signing (copy sent to all signers via Signet Docs)
- public int MininumSigners { get; set; }
  - Minimal number of signers that have to sign the document, if <1 then everyone has to sign the document.
- public DocumentSigner[] Signers { get; set; }
  - Array with signers, described better here below.
- public string Reason { get; set; }
  - Given reason for signing, for example „Approved“ or „Confirmed“.

4.4 DOCUMENTSIGNER

When a document is uploaded, an array of signers must be defined in the form of classes of the type DocumentSigner. In the document status information (SignetDocumentinfo) the class is used to describe the signer status. The class contains following definitions:

- public string SSN { get; set; }
  - Unique identifier of the signer (Kennitala). Mandatory for uploading.
- public string Name { get; set; }
  - Name of the signer. Not used when uploading.
- public string Email { get; set; }
  - Signers email address. If the user is already registered this is ignored.
- public bool Notify { get; set; }
  - Should the signer be notified (email) of document and status changes.
- public string SigningMessage { get; set; }
  - Messages to be shown when signing document.
- public string SigningText { get; set; }
  - Text which appears above the name of the signer on the signing page. Max length of text is 40 characters and will be shortened to first 40 characters if longer and a subscript added to the bottom of the page. Please note that if a signed document is added to Signet this text is ignored.
- public int Order { get; set; }
  - The order of the document signers on the document. If more than one signer have the same order number, any of those signers can use the signature field to sign the document. The first one to sign the document is the signer. Order must be > 0.
- public SigningRole Role { get; set; }
  - Role of the signer. See SigningRole below.
- public DateTime? Signed { get; set; }
  - Date and time when signed. Used in status information.
- public bool HasSigned { get; set; }
  - Has been signed. Used in status information.
- public bool Declined { get; set; }
  - Has been rejected. Used in status information.
- public string Reason { get; set; }
  - Reason for rejection. Used in status information.

- public enum SigningRole
  - The role of the user
    - Signer = 0, - A signer of the document
    - Authorizer = 1, - A referee of the document – not implemented yet
    - Viewer = 2 – Has right to view document

4.5 SIGNETDOCUMENTINFO

When enquiring the status of a document on the legacy controller a class of the type SignetDocumentinfo is returned. Signet Documentinfo is defined as follows:

- public string DocumentId { get; set; }
  - Document ID on GUID format, for examle 6282559D-5765-4C0F-81FC-C0ADD7D34F56
- public DocumentStatus Status { get; set; }
  - Document status
- public DateTime Added { get; set; }
  - When the document was added to Signet
- public DateTime? Modified { get; set; }
  - Time of last document data update, for example when signed.
- public DateTime? Deleted { get; set; }
  - When was the document deleted.
- public string DocumentName { get; set; }
  - Name of the document which was attached when uploaded.
- public string DocumentNotes { get; set; }
  - Further description of the uploaded document.
- public string Digest { get; set; }
  - SHA-1 digest of document data. Note that signing does not change this digest.
- public string Metadata { get; set; }
  - Metadata information that followed the uploaded document.
- public DateTime? StartSigning { get; set; }
  - When the signing period starts. Nullable
- public DateTime? EndSigning { get; set; }
  - When the signing period ends. Nullable
- public bool Ordered { get; set; }
  - Should the signing of the document be ordered.
- public bool Notify { get; set; }
  - Shall the deletion be notified
- public int MininumSigners { get; set; }
  - Minimal number of signing needed for completion of the signing process.
- public DocumentSigner[] Signers { get; set; }
  - Array of signers.
- public string Reason { get; set; }
  - Default reason for signing.
- public enum DocumentStatus
  - Document status
    - New = 0 – No signatures in the document
    - InSigning = 1 – Document in signing process (at least one signature)
    - Cancelled = 2 – Document cancelled.
    - Signed = 3 – Document ready (signed by all signers)
    - InReview = 4 – Document in review. Not supported in version 1.0
    - Reviewed = 5 - Document reviewed. Not supported in version 1.0

- Deleted = 6 – Document deleted
- Rejected = 7 – Document rejected.

4.6 SIGNETDOCUMENTINFOV3

When enquiring the status of a document a class of the type SignetDocumentinfoV3 is returned. SignetDocumentinfoV3 extends SignetDocumentInfo by adding the following attribute:

- public string GroupID
  - The document group ID

4.7 SIGNETNOTIFICATION

When a status of a document changes, information of the change are sent to endpoint which must be able to listen to POST with following class (XML or JSON).

- public string DocID { get; set; }
  - Document ID in GUID form
- public string Notes { get; set; }
  - Further description of a document
- public string DocumentName { get; set; }
  - Name of document
- public string Metadata { get; set; }
  - Metadata information which followed the document
- public DocumentStatus Status { get; set; }
  - Document status
- public enum DocumentStatus
  - Document status
    - New = 0 – No signatures in the document
    - InSigning = 1 – Document in signing (at least one signature)
    - Cancelled = 2 – Document cancelled.
    - Signed = 3 – Document ready (fully signed)
    - InReview = 4 – Document in review. Not supported in version 1.0
    - Reviewed = 5 - Document in review process. Not supported in version 1.0
    - Deleted = 6 – Document deleted.
    - Rejected = 7 – Document rejected.

4.8 DOCUMENTINFO

When getting list of users document in the TokenService the documents are returned in a list of DocumentInfo objects which are as follows (extends SignetDocumentInfo):

- public string DocumentId { get; set; }
  - Document ID on GUID format, for examle 6282559D-5765-4C0F-81FC-C0ADD7D34F56
- public DocumentStatus Status { get; set; }
  - Document status
- public DateTime Added { get; set; }
  - When the document was added to Signet
- public DateTime? Modified { get; set; }
  - Time of last document data update, for example when signed.
- public DateTime? Deleted { get; set; }

    - When was the document deleted.
- public string DocumentName { get; set; }
    - Name of the document which was attached when uploaded.
- public string DocumentNotes { get; set; }
    - Further description of the uploaded document.
- public string Digest { get; set; }
    - SHA-1 digest of document data. Note that signing does not change this digest.
- public string Metadata { get; set; }
    - Metadata information that followed the uploaded document.
- public DateTime? StartSigning { get; set; }
    - When the signing period starts. Nullable
- public DateTime? EndSigning { get; set; }
    - When the signing period ends. Nullable
- public bool Ordered { get; set; }
    - Should the signing of the document be ordered.
- public bool Notify { get; set; }
    - Shall the deletion be notified
- public bool Archive { get; set; }
    - Shall the document be archived after signature
- public int MininumSigners { get; set; }
    - Minimal number of signing needed for completion of the signing process.
- public DocumentSigner[] Signers { get; set; }
    - Array of signers.
- public string Reason { get; set; }
    - Default reason for signing.
- public string Creator { get; set; }
    - Name of creator of document
- public enum DocumentStatus
    - Document status
        - New = 0 – No signatures in the document
        - InSigning = 1 – Document in signing process (at least one signature)
        - Cancelled = 2 – Document cancelled.
        - Signed = 3 – Document ready (signed by all signers)
        - InReview = 4 – Document in review. Not supported in version 1.0
        - Reviewed = 5 - Document reviewed. Not supported in version 1.0
        - Deleted = 6 – Document deleted
        - Rejected = 7 – Document rejected.

4.9 SIMPLENATINFO

When requesting info about SSN the result is given in a SimpleNatInfo object which is as follows:

- public string Ssn { get; set; }
    - SSN of individual or company
- public string Name { get; set; }
    - Name of individual or company
- public string Address { get; set; }
    - Address of individual or company
- public string PostCode { get; set; }
    - Postcode of individual or company
- public string Postal { get; set; }

- Postal of individual or company

4.10 ADDREQUESTV3

The AddRequestV3 extends BaseRequest and also has the following parameters.

- public AddDocumentRequestV3 request { get; set; }
  - The AddDocumentRequest
- public string GroupID { get; set; }
  - The previous group ID if adding to an earlier document group

4.11 ADDDOCUMENTREQUESTV3

AddDocumentRequestV3 is an updated version of the AddDocumentRequest shown previously, the class is defined as follows:

- public bool DontAddPage
  - Don't appent a signature page
- public List<Document> Documents
  - The documents to add
- public string DocumentNotes
  - Further description of the document which is accessible for the user, for example „Account agreement".
- public string Metadata { get; set; }
  - Metadata string used for information which is not accessible for the user but the web service gets with the document.
    - You can add ";returnUrl=<URL>;" if you would like the user to be redirected to this url after signing if he was not authenticated using token. The query parameter **status** is appended to the url with the possible values
      - **signed** if the document was signed
      - **rejected** if the document was rejected
      - **deleted** if the document has been deleted
    - If the document is an XML document you can add ";xslt=<URL for XSLT>;" if you would like an XSLT style to be applied to the document at viewing/signing.
    - If you want to control the text on the rejection window you can add ";rejectMsg=<msg>;" to show a text below the reason input.
    - If you want to control where to send callback on status changes you can add ";callBackUrl=<URL for callback>;" to the metadata.
- public DateTime? StartSigning { get; set; }
  - Initial date, the date when the signing process can start (not necessary)
- public DateTime? EndSigning { get; set; }
  - End date when the signing process ends (not necessary)
- public bool Ordered { get; set; }
  - Shall the signing be done in the order of the signers or not (Signer 1 first, then Signer 2 etc).
- public bool Notify { get; set; }
  - Notify when a document is deleted – not used at this time
- public bool Archive { get; set; }
  - Should the document be archived after signing (copy sent to all signers via Signet Docs)
- public int MininumSigners { get; set; }

    - Minimal number of signers that have to sign the document, if <1 then everyone has to sign the document.
- public List<DocumentSignerV3> { get; set; }
    - Array with signers, described better here below.
- public string Reason { get; set; }
    - Given reason for signing, for example „Approved“ or „Confirmed“.
- public bool AttachXml
    - Attach XML document in list of documents to PDF document
- public enum SealDocuments
    - Should documents be sealed (signed by a digital signature of company)
    - 0 = Do not seal
    - 1 = Seal all documents
    - 2 = Seal XML documents
    - 3 = Seal PDF documents

4.12 DOCUMENT

When adding documents or getting documents from the new controllers a Document class is used which is as following

- public string ID
    - The document ID (when returning)
- public byte[] Data
    - The document data
- public string DocumentName
    - The document filename, i.e. document.pdf
- public DocStatus Status
    - The status of the document (new, signed etc)

4.13 DOCUMENTSIGNERV3

An updated version of DocumentSigner which adds the following attribute

- public FieldLocation Location
    - Location of the signature field for signer (nullable)

4.14 FIELDLOCATION

FieldLocation is a class used to define location of (empty) signature fields, the definition is as follows:

- public in Page
    - The page for the field
- public int XLocation
    - The x coordinate of field
- public int YLocation
    - The y coordinate of field
- public int Height
    - The height of field
- public int Width
    - The width of field

4.15 DOCUMENTREQUESTV3

When requesting a document a DocumentRequest is used which is as follows

- public string Reason
  - The document ID
- public string GroupID
  - The document group ID

4.16 DOCUMENTSREQUEST

When requesting/searching documents the DocumentsRequest class is used which is as follows:

- public string SSN
  - The SSN of signer
- public DateTime? FromTime
  - The date from which document was added
- public DateTime? ToTim
  - The date to which document was added
- public DocumentStatus? Status
  - The status of document

4.17 TOKENREQUESTV3

When requesting a token for sending a user to a document on Signet a TokenRequestV3 is used, the class is as follows:

- public string DocID
  - The ID of the document
  - Mandatory
- public string GroupID
  - The group ID of document(s)
- public string SSN
  - The SSN of signer
  - Mandatory
- public string Onbehalf
  - The SSN of onbehalf signer
  - Not mandatory
- public string Mobile
  - The mobile phone of signer
  - If requesting user to sign with Auðkenni App use SSN of user as mobile
- public string ReturnURL
  - The return URL to send signer after signature
  - Mandatory
- public string UserCert
  - The authentication or signing certificate of signer
  - Helps find the certificate for signing but not necessary
- public string Mandate
  - The mandate ID to use if signing on behalf of someone else
  - Not mandatory
- public CertificateDevice Device

- The device which the user will use for signing. If the device is not sent Auðkenni SIM is assumed.
- The devices are:
  - AudkenniSIM = 0
  - AudkenniApp = 1
  - AudkenniCard = 2
  - Evrotrust = 3

4.18 PINGREQUEST

When pinging the service to check connections and authentication a PingRequest is used which is as follows

- public string input
  - The input message

4.19 BASETOKENREQUEST

When calling the token interface/controller a BaseTokenRequest is used which is as follows

- public string token
  - The authentication token

4.20 TOKENDOCUMENTREQUEST

When requesting a document from the token controller a TokenDocumentRequest is used which is as follows

- public string token
  - The authentication token
- public string DocID
  - The document id
- public string GroupID
  - The group id

4.21 TOKENADDDOCUMENTREQUESTV3

When adding a document using a token controller a TokenAddDocumentRequestV3 is as follows

- public string token
  - The authentication token
- public AddDocumentRequestV3 request
  - The add request
- public string GroupID
  - The group id

4.22 TOKENSIGNDOCUMENTREQUEST

When signing a document a TokenSignDocumentRequest is used which is as follows

- public string token
  - The authentication token

- public string DocID
  - The document id
- public string GroupID
  - The group id
- public string Lang
  - The language for signing message

4.23 TOKENREJECTDOCUMENTREQUEST

When rejecting a document a TokenRejectDocumentRequest is used which is as follows

- public string token
  - The authentication token
- public string DocID
  - The document id
- public string GroupID
  - The group id
- public string Lang
  - The language for signing message
- public string Reason
  - The reason for rejecting the document (require)
- public bool Auth
  - Should we require the user to authenticate the rejection

4.24 ACCOUNTTOKENREQUEST

When getting a token for the AccountToken interface a AccountTokenRequest is used which is as follows

- public string Phone
  - The phone number of signer containing digital certificates
- public string SAML
  - A federation SAML token with account information signed with a certificate containing the SSN of the company
- public string SSN
  - The SSN of signer

4.25 SIGNDOCUMENTREQUEST

To sign a document on the Signet interface a SignDocumentRequest (based on BaseRequest) is used which is as follows.

- public string SSN
  - The SSN of signer
- public string Mobile
  - The mobile phone of signer
- public string Lang
  - The language for signing message
- public string DocID
  - The document id
- public string GroupID

    - The group id
- public CertificateDevice Device
    - The device which the user will use for signing. If the device is not sent Auðkenni SIM is assumed. Note that only AudkenniSIM and Evrotrust is supported for /SignDocument
    - The devices are:
        - AudkenniSIM = 0
        - AudkenniApp = 1
        - AudkenniCard = 2
        - Evrotrust = 3

4.26 REJECTDOCUMENTREQUEST

To sign a document on the Signet interface a RejectDocumentRequest (based on BaseRequest) is used which is as follows.

- public string Reason
    - The reason for rejection (mandatory)
- public bool Auth
    - Require user to authenticate the rejection
- public string SSN
    - The SSN of signer
- public string Mobile
    - The mobile phone of signer
- public string Lang
    - The language for signing message
- public string DocID
    - The document id
- public string GroupID
    - The group id

4.27 SIGNDOCUMENTWITHAPPREQUEST

To sign a document on the Signet interface with the Auðkenni app a SignDocumentRequestWithApp (based on BaseRequest) is used which is as follows.

- public string SSN
    - The SSN of signer
- public string Mobile
    - The mobile phone of signer
- public string Lang
    - The language for signing message
- public string DocID
    - The document id
- public string GroupID
    - The group id
- public string SignID
    - The signature id needed for starting the signature (from SignDocumentWithApp response)

4.28 CERTIFICATEDEVICE

CertificateDevice defines the device the user will use for signing. The supported devices are:

- AudkenniSIM = 0
- AudkenniApp = 1
- AudkenniCard = 2
- Evrotrust = 3

5 RESULT CLASSES

5.1 ADDRESPONSE

When adding document(s) to Signet the return value is an AddResponse object which is as follows:

- string GroupID
  - The group ID for documents
- List<string> DocIDs
  - List of individual document IDs

5.2 SIGNDOCUMENTWITHAPPRESPONSE

When signing a document with Auðkenni App a SignDocumentWithAppResponse is returned

- string Code
  - The VCODE (verification code) which will be displayed on users device
- string SignatureID
  - The signature ID needed for starting the signature process after displaying the Code to the user

5.3 DOCUMENTLOG

When fetching document logs an array of DocumentLog object is returned

- string DocumentID
  - The ID of document
- string Time
  - Time of event (log)
  - yyyy-MM-ddTHH:mm:ss.fff
- string Message
  - The log message
- string Level
  - The log level
  - (INFO, ERROR, WARN)

6 LEGACY REST RESULT CLASSES

This chapter describes the objects returned from the Legacy conctrollers.

6.1 BASERESPONSE

The base response is returned from all void functions and is BaseReponse JSON object which is as follows.

- ResultStatus outStatus
  - A ResultStatus (see 5.1) with the results status of request

6.2 PINGRESPONSE

When using the Ping function the response is a PingResponse JSON object which is as follows.

- string PingResult
  - A message containing user account and the input message
- ResultStatus outStatus
  - A ResultStatus (see 5.1) with the results status of request

6.3 ADDDOCUMENTRESPONSE

When adding a document the response is a AddDcoumentResponse JSON object which is as follows.

- string AddDocumentResult
  - A string with the document ID (GUID) if successful
- ResultStatus outStatus
  - A ResultStatus (see 5.1) with the results status of request

6.4 DELETEDOCUMENTRESPONSE

When deleting a document the response is a DeleteDocumentResponse JSON object which is as follows.

- bool DeleteDocumentResult
  - True if delete was successful
- ResultStatus outStatus
  - A ResultStatus (see 5.1) with the results status of request

6.5 GETDOCUMENTRESPONSE

When downloading a document the response is a GetDocumentResponse JSON object which is as follows.

- byte[] GetDocumentResult
  - Byte array with document
- ResultStatus outStatus
  - A ResultStatus (see 5.1) with the results status of request

6.6 GETDOCUMENTSTRINGRESPONSE

When downloading a document the response is a GetDocumentStringResponse JSON object which is as follows.

- string GetDocumentStringResult

    - Base64 encoded document
- ResultStatus outStatus
    - A ResultStatus (see 5.1) with the results status of request

6.7 GETTOKENRESPONSE

When getting an authentication token the response is a GetTokenResponse JSON object which is as follows.

- string GetTokenResult
    - A string with the base64 encoded token
- ResultStatus outStatus
    - A ResultStatus (see 5.1) with the results status of request

6.8 REFRESHTOKENRESPONSE

When refreshing an authentication token the response is a RefreshTokenResponse JSON object which is as follows.

- string RefreshTokenResult
    - A string with the base64 encoded token
- ResultStatus outStatus
    - A ResultStatus (see 5.1) with the results status of request

6.9 GETDOCUMENTINFORESPONSE

When getting users document info the response is a GetDocumentInfoResponse JSON object which is as follows.

- DocumentInfo GetDocumentInfoResult
    - A DocumentInfo object (see 4.5)
- ResultStatus outStatus
    - A ResultStatus (see 5.1) with the results status of request

6.10 GETDOCUMENTLISTRESPONSE

When getting a list of users documents the response is a GetDocumentListResponse JSON object which is as follows.

- List<DocumentInfo> GetDocumentListResult
    - A list of DocumentInfo objects (see 4.5)
- ResultStatus outStatus
    - A ResultStatus (see 5.1) with the results status of request

6.11 RESTGETDOCUMENTRESPONSE

When downloading a document the response is a RestGetDocumentResponse JSON object which is as follows.

- string GetDocumentResult
    - Base64 encoded document
- ResultStatus outStatus

- A ResultStatus (see 5.1) with the results status of request

6.12 GETDOCUMENTIMAGESRESPONSE

When getting document images the response is a GetDocumentImagesResponse JSON object which is as follows:

- List<string> GetDocumentImagesResult
  - List of Base64 encoded images
- ResultStatus outStatus
  - A ResultStatus (see 5.1) with the results status of request

6.13 SIGNDOCUMENTRESULT

When signing a document the response is a SignDocumentResult JSON object which is a follows:

- bool SignDocumentResult
  - True if signing the document was successful
- ResultStatus outStatus
  - A ResultStatus (see 5.1) with the results status of request

6.14 REJECTDOCUMENTRESULT

When rejecting a document the response is a RejectDocumentResult JSON object which is a follows:

- bool RejectDocumentResult
  - True if rejecting the document was successful
- ResultStatus outStatus
  - A ResultStatus (see 5.1) with the results status of request

6.15 SIMPLENATINFORESPONSE

When getting info from national registry the response is a SimpleNatInfoResponse JSON object which is a follows:

- SimpleNatInfo GetNatInfoResult
  - Info about SSN
- ResultStatus outStatus
  - A ResultStatus (see 5.1) with the results status of request

6.16 GETDOCUMENTSRESPONSE

When getting a list of account documents the response is a GetDocumentsResponse JSON object which is as follows.

- List<DocumentInfo> GetDocumentsResult
  - A list of DocumentInfo objects (see 4.5)
- ResultStatus outStatus
  - A ResultStatus (see 5.1) with the results status of request

6.17 GETDOCUMENTCONTENTRESPONSE

When getting the text content of a document the response is a GetDocumentContentResponse JSON object which is as follows.

- string GetdocumentContentResult
  - The document content (text).
- ResultStatus outStatus
  - A ResultStatus (see 5.1) with the results status of request

7 STATUS CHANGES

Advania can register a RESTful endpoint which receives the status changes (POST) of account document. Status changes are sent with SignetNotification class as described above in chapter 5.5. Signet supports endpoints with a) no authentication and b) endpoints with username and password authentication (basic) and c) endpoints with electronic certificates as an authentication method. The status can be POSTed as either XML or JSON.

7.1 DOCUMENT WILL BE DELETED

When a document is to be deleted, the text „TOBeDeleted=1“ is added to the document metadata.

7.2 CODE EXAMPLE

An example web service for status changes can be found at the site where the examples and instructions can be found (https://prufa.signet.is/documentation/).

7.3 IP ADDRESSES

The notification messages will come from the following IP addresses

- 82.221.36.238
  - Pre production environment
- 212.30.225.121
  - Production environment

8 INTERFACE

The v3 web service is a REST API which uses either TLS client certificate or token based authentication.

8.1 SIGNET INTERFACE

For adding to and handling documents in Signet the Signet controller is user. If requests are unsuccessful it will return a Bad Request with ResultStatus as content. Internal errors (500) do not return anything.

8.1.1 ADDDOCUMENT

To add a document to Signet you must PUT a AddRequestV3 to /Signet/AddDocument which is as follows.

```
AddResponse AddDocument(AddRequestV3 request)
```

The functions returns an AddResponse containg group and document ID(s) if successful.

8.1.2 DOWNLOADDOCUMENT

To download a document you must POST a DocumentRequestV3 to /Signet/DownloadDocument

```
List<Document> DownloadDocument(DocumentRequestV3 request)
```

If successful the function returns a list of Documents which conforms to the request.

8.1.3 GETDOCUMENT

To get a document info you must POST a DocumentRequestV3 to /Signet/GetDocument

```
List<SignetDocumentInfoV3> GetDocument(DocumentRequestV3 request)
```

If successful the function returns a list of SignetDocumentInfoV3 which conforms to the request.

8.1.4 GETDOCUMENTS

To get a document info you must POST a DocumentRequestV3 to /Signet/GetDocuments

```
List<SignetDocumentInfoV3> GetDocuments(DocumentsRequest request)
```

If successful the function returns a list of SignetDocumentInfoV3 which conforms to the request.

8.1.5 DELETEDOCUMENT

To delete a document you must POST a DocumentRequestV3 to /Signet/DeleteDocument

```
bool DeleteDocument(DocumentRequestV3 request)
```

If successful the function returns true.

8.1.6 GETTOKEN

To get an authentication token for sending a signer to /token/sign you must POST a TokenRequestV3 to /Signet/GetToken

```
string GetToken(TokenRequestV3 request)
```

If successful the function returns the base64 encoded token.

8.1.7 REMINDDOCUMENT

To remind signers who still need to sign a document you must POST a DocumentRequestV3 to /Signet/RemindDocument

```
void RemindDocument(DocumentRequestV3 request)
```

If successful the function returns HTTP 200.

8.1.8 GETNOTIFICATION

To get a notification about document to registered endpoint you must POST a DocumentRequestV3 to /Signet/GetNotification

```
ResultStatus GetNotification(DocumentRequestV3 request)
```

If successful the function returns ResultStatus.

8.1.9 GETNOTIFICATIONMESSAGE

To get an example notification about document (which would normally be sent to registered endpoint as in GetNotification) you must POST a DocumentRequestV3 to /Signet/GetNotificationMessage

```
SignetNotification GetNotificationMessage(DocumentRequestV3 request)
```

If successful the function returns a SignetNotification of document status.

8.1.10 PING

To ping the service you must POST a PingRequest to /Signet/Ping

```
string Ping(PingRequest request)
```

If successful the function returns a string message.

8.1.11 SIGNDOCUMENT

To have user sign a document you must POST a SignDocumentRequest to /Signet/SignDocument

```
bool SignDocument(SignDocumentRequest request)
```

If successful the function returns true.

8.1.12 SIGNDOCUMENTWITHAPP

To have user sign a document on Auðkenni APP you must POST a SignDocumentRequest to /Signet/SignDocumentWithApp

```
SignDocumentWithAppRespons SignDocumentWithApp(SignDocumentRequest request)
```

If successful the function returns a SignDocumentWithAppResponse with the information needed for next step (StartAppSignature).

8.1.13 STARTAPPSIGNATURE

To have user sign a document on Auðkenni APP you must POST a SignDocumentWithAppRequest to /Signet/StartAppSignature which should contain the SignID from SignDocumentWithApp function above.

```
bool StartAppSignature(SignDocumentWithAppRequest request)
```

If successful the function returns true.

8.1.14 REJECTDOCUMENT

To have user reject a document you must POST a RejectDocumentRequest to /Signet/RejectDocument

```
bool RejectDocument(RejectDocumentRequest request)
```

If successful the function returns true.

8.1.15 GETDOCUMENTLOGS

To fetch all logs for a document you must POST a DocumentRequestV3 to /Signet/GetDocumentLogs

```
[DocumentLog] GetDocumentLogs(DocumentRequestV3 request)
```

If successful the function returns an array of DocumentLog messages.

8.2 LEGACY INTERFACE

For the legacy REST interface the same six functions are provided. The URL for the functions are <SignetURL>/Legacy/<method>. All legacy functions use POST

8.2.1 ADDDOCUMENT

To upload a document to Signet the function AddDocument is used:

```
AddDocumentResponse AddDocument(string username, string password, AddDocumentRequest
request);
```

The variables are:

- `username`
  - string
  - The account username
- `password`
  - string
  - The account password.
- `request`
  - AddDoumentRequest
  - The request is described further in chapter 5.2

The function returns a string with the document ID if the operation was successful (see 6.3).

8.2.2 DELETEDOCUMENT

To delete document that has already been uploaded, the function DeletDocument is used:

```
DeleteDocumentResponse DeleteDocument(string username, string password, string docId);
```

The variables are:

- `username`
  - string
  - The account username
- `password`
  - string
  - The account password
- `docId`
  - string
  - The ID of the document that shall be deleted

The function returns true if it succeeded (see 6.4). Please note that the operation is irreversible.

8.2.3 GETDOCUMENT

To get the document data that has already been added, the function GetDocument is used:

```
GetDocumentResponse GetDocument(string username, string password, string docId);
```

The variables are:

- username
  - string
  - The account username
- password
  - string
  - The account password
- docId
  - string
  - ID of the document that shall be downloaded

The function returns the document data if the operation was successful (see 6.5).

8.2.4 GETDOCUMENTSTRING

To get the document data as base64 encoded string that has already been added, the function GetDocumentString is used:

```
GetDocumentStringResponse GetDocument(string username, string password, string docId);
```

The variables are:

- username
  - string
  - The account username
- password
  - string
  - The account password
- docId
  - string
  - ID of the document that shall be downloaded

The function returns the document data if the operation was successful (see 6.6).

8.2.5 GETDOCUMENTS

To get all documents which account has added, the function GetDocuments is used:

```
GetDocumentsResponse GetDocuments(string username, string password, int daysBack, out
ResultStatus outStatus);
```

The variables are:

- username
  - string
  - The account username
- password
  - string
  - The account password

- daysBack
  - int
  - Number of days to search for
- outStatus
  - ResultStatus
  - Return value is described further in chapter 5.1

If the operation succeeds the functions returns a list of document infos (see 6.14).

8.2.6 GETDOCUMENTINFO

To get information about a document which has already been uploaded to Signet, the function GetDocumentInfo is used:

```
GetDocumentInfoResponse GetDocumentInfo (string username, string password, string docId);
```

The variables are:

- `username`
  - string
  - The account username
- `password`
  - string
  - The account password
- `docId`
  - string
  - ID of the document that shall be downloaded.

If the operation succeeds the function returns information about the document (see 6.8).

8.2.7 REMINDDOCUMENT

To send a reminder to signers that still have not signed the document, the function RemindDocument is used:

```
BaseResponse RemindDocument(string username, string password, string docId);
```

The variables are:

- `username`
  - string
  - The account username
- `password`
  - string
  - The account password
- `docId`
  - string
  - ID of the document that shall be downloaded

The function returns a ResultStatus JSON object with the result (see 6.14).

8.2.8 PING

To test the service and connection information the function Ping can be used:

```
PingResponse Ping(string username, string password, string input);
```

The variables are:

- username
  - string
  - The account username
- password
  - string
  - The account password
- input
  - string
  - Text that is returned with the answer

The function returns a string with a hello message containing the username and the input string if successful (see 6.2).

8.2.9 GETTOKEN

If the user already has been authenticated then an authentication token (base64 coded SAML) for the document and the user can be collected with GetToken. The user can then be sent directly to the document (POST with the parameter token) on <signet URL>/token/sign:

```
GetTokenResponse GetToken(string username, string password, string docId, string kt,
string phone, string returnUrl);
```

The variables are:

- username
  - string
  - The account username
- password
  - string
  - The account password
- dociId
  - string
  - Id of the document that the token is for
- kt
  - string
  - Unique identifier of the signer (kennitala)
- phone
  - string
  - Mobile number from the one who shall sign (+354 xxxxxxx) if the user has certificate on the phone, empty if the user has certificate on a smart card.
- returnUrl
  - string
  - URL that the user will be forwarded to after signing.
- cert
  - string
  - Base64 encoded authentication or signature certificate of the signer (preferred for signing with card). If no certificate is present user will have to authenticate in the beginning of signing process

The function returns string that is base64 coded signed SAML signed using Signet device certificate (”Búnaðarskilríki”) (see 6.7). When the signing is completed the user is returned back to the returnUrl with similar SAML which contains information about the user, document, metadata and the status of the document. An example of the return SAML is in the appendix below.

8.2.10 GETNOTIFICATION

To get the status of the document to a defined endpoint the function GetNotification is used:

```
BaseResponse GetNotification(string username, string password, string docId);
```

The variables are:

- `username`
  - string
  - The account username
- `password`
  - string
  - The account password.
- `docId`
  - string
  - ID of the document that the status is required for

The function returns a ResultStatus JSON object with the result (see 6.1).

8.3 ACCOUNT TOKEN INTERFACE

The account token interface has the following methods which all require apikey header (basic auth format) and a signed token.

8.3.1 GETTOKEN

To get an JWT authentication token to use with other methods on the interface a POST to /AccountToken/GetToken is needed which is as follows:

```
string GetToken(AccountTokenRequest request)
```

If successful the function returns a JWT token to use as authentication on the other methods.

8.3.2 DOCUMENTS

To get a list of users documents you must POST a BaseTokenRequest to /AccountToken/Documents:

```
List<SignetDocumentInfoV3> Documents(BaseTokenRequest request)
```

If successful the function returns a list of SignetDocumentInfoV3.

8.3.3 ADDDOCUMENT

To add a document to Signet you must PUT a TokenAddRequestV3 to /AccountToken/AddDocument which is as follows.

```
AddResponse AddDocument(TokenAddRequestV3 request)
```

The functions returns an AddResponse containg group and document ID(s) if successful. TokenAddRequestV3 is just like AddRequestV3 but with the added token parameter.

8.3.4 GETDOCUMENT

To get a document info you must POST a TokenDocumentRequest to /AccountToken/GetDocument

```
List<SignetDocumentInfoV3> GetDocument(TokenDocumentRequest request)
```

If successful the function returns a list of SignetDocumentInfoV3 which conforms to the request.

8.3.5 DOWNLOADDOCUMENT

To get a document info you must POST a TokenDocumentRequest to /AccountToken/DownloadDocument

```
List<Document> DownloadDocument(TokenDocumentRequest request)
```

If successful the function returns a list of Document which conforms to the request.

8.3.6 GETDOCUMENTCONTENT

To get document content you must POST a TokenDocumentRequest to /AccountToken/GetDocumentContent

```
string GetDocumentContent(TokenDocumentRequest request)
```

If successful the function returns a string with all the text in the document.

8.3.7 GETDOCUMENTIMAGES

To get document pages as images you must POST a TokenDocumentRequest to /AccountToken/GetDocumentContent

```
List<byte[]> GetDocumentImages(TokenDocumentRequest request)
```

If successful the function returns a list of byte arrays with all the pages in the document.

8.3.8 SIGNDOCUMENT

To sign a document you must POST a TokenSignDocumentRequest to /AccountToken/SignDocument

```
bool SignDocument(TokenSignDocumentRequest request)
```

If successful the function returns true.

8.3.9 REJECTDOCUMENT

To reject a document you must POST a TokenRejectDocumentRequest to /AccountToken/RejectDocument

```
bool RejectDocument(TokenSignDocumentRequest request)
```

If successful the function returns true.

8.3.10 DELETEDOCUMENT

To delte a document you must POST a TokenDocumentRequest to /AccountToken/DeleteDocument

```
bool DeleteDocuemtn(TokenDocumentRequest request)
```

If successful the function returns true.

8.3.11 GETSIGNTOKEN

To get sign a document you must POST a GetSignTokenRequest to /AccountToken/GetSignToken

```
string GetSignToken(GetSignTokenRequest request)
```

If successful the function returns an authentication token to use for card signing on signet.is.

8.4 LEGACY ACCOUNT TOKEN INTERFACE

The legacy account token interface has the following methods. Note that all methods require apikey header for account (basic auth format) and should be POST'ed to <SignetURL>/LegacyAccountToken/<method>.

8.4.1 GETTOKEN

To get an JWT authentication token to use with other methods on the interface a call to GetToken is needed which is as follows:

```
GetTokenResponse GetToken(string saml, string ssn, string phone, out ResultStatus
outStatus);
```

The variables are:

- `saml`
  - string
  - SAML which includes the attribute Account (in AttributeStatement/Attribute node) or NameID (in Subject) with the Signet account and signed with a trusted certificate which includes SSN of company.
- `ssn`
  - string
  - SSN of user we are interfacing with.
- phone
  - string
  - The users mobile number with digital certificates ([+354][1-9]{1}[0-9]{6}, needed for signing)
- `outStatus`
  - ResultStatus
  - Further description of the result

The function returns string that is base64 coded signed JWT and is used on other methods as a way of authenticating user.

8.4.2 GETSIGNTOKEN

To get a signature token (i.e. in the event the user only has certificates on smart card) for sending user to <signet url>/token/sign or signsp the method GetSignToken is used:

```
bool GetSignToken(string token, string docId, string returnUrl out ResultStatus
outStatus);
```

The variables are:

- `token`

- string
- The token issued from GetToken

- docId
  - string
  - ID of the document that shall be signed
- returnUrl
  - string
  - URL that the user will be forwarded to after signing.
- outStatus
  - ResultStatus
  - Return value is described further in chapter 5.1

The function returns string that is base64 coded signed SAML signed using Signet device certificate (”Búnaðarskilríki”) (see 6.7). When the signing is completed the user is returned back to the returnUrl with similar SAML which contains information about the user, document, metadata and the status of the document. An example of the return SAML is in the appendix below.

8.4.3 GETDOCUMENTLIST

To get a list of users documents from account the method GetDocumentList is used which is as follows:

```
GetDocumentListResponse GetDocumentList(string token, out ResultStatus outStatus);
```

The variables are:

- token
  - string
  - The token issued from GetToken
- outStatus
  - ResultStatus
  - Further description of the result

The function returns a list of users documents in a list of DocumentInfo objects as described in 4.5.

8.4.4 GETDOCUMENTINFO

To get info about a single document the method GetDocumentInfo is used. The method is as follows

```
GetDocumentInfoResponse GetDocumentInfo(string token, string docId, out ResultStatus
outStatus);
```

The variables are:

- token
  - string
  - The token issued from GetToken
- docId
  - string
  - ID of the document that shall be fetched
- outStatus
  - ResultStatus
  - Further description of the result

The function returns info about the document in a DocumentInfo object as described in 4.5.

8.4.5 ADDDOCUMENT

To add a document to Signet the method AddDocument is used which is as follows:

```
string AddDocumentResponse(string token, AddDocumentRequest request, out ResultStatus
outStatus);
```

The variables are:

- token
  - string
  - The token issued from GetToken
- request
  - AddDoumentRequest
  - The request is described further in chapter 5.2
- outStatus
  - ResultStatus
  - Further description of the result

The function returns a string with the document ID if the operation was successful.

8.4.6 GETDOCUMENT

To fetch a user document the method GetDocument is used which is as follows:

```
GetDocumentResponse GetDocument(string token, string docId, out ResultStatus
outStatus);
```

The variables are:

- token
  - string
  - The token issued from GetToken
- docId
  - string
  - ID of the document that shall be downloaded
- outStatus
  - ResultStatus
  - Return value is described further in chapter 5.1

The function returns the document data if the operation was successful. Please note that the user will get an authentication request to his phone to allow getting the document.

8.4.7 GETDOCUMENTIMAGES

To get a list/array of images of a document the method GetDocumentImages is used which is as follows:

```
GetDocumentImagesResponse GetDocumentImages(string token, string docId, out
ResultStatus outStatus);
```

The variables are:

- token
  - string
  - The token issued from GetToken
- docId

    - string
    - ID of the document images that shall be downloaded
- outStatus
    - ResultStatus
    - Return value is described further in chapter 5.1

The function returns a list of images (PNG) of document pages if the operation was successful.

8.4.8 SIGNDOCUMENT

To sign a document that still needs a user signature the method SignDocument is used which is a follows:

```
SignDocumentResponse SignDocument(string token, string docId, string lang, out ResultStatus outStatus);
```

The variables are:

- token
    - string
    - The token issued from GetToken
- docId
    - string
    - ID of the document that shall be signed
- lang
    - string
    - The language to use for signing message (IS or EN at time of write - not required). Default IS if null sent
- outStatus
    - ResultStatus
    - Return value is described further in chapter 5.1

The function returns true if the operation was successful. Please note that the user will get a signature request on his mobile.

8.4.9 REJECTDOCUMENT

To reject a document that still needs a user signature the method RejectDocument is used which is a follows:

```
RejectDocumentResponse RejectDocument(string token, string docId, string reason, string lang, out ResultStatus outStatus);
```

The variables are:

- token
    - string
    - The token issued from GetToken
- docId
    - string
    - ID of the document that shall be signed
- reason
    - string
    - The reason for rejection (required)
- lang

- string
- The language to use for rejection message (IS or EN at time of write - not required). Default IS if null sent.

- `auth`
  - bool
  - To send an authentication request to signer to authorize rejection (true) or not (false).
- outStatus
  - ResultStatus
  - Return value is described further in chapter 5.1

The function returns true if the operation was successful. Please note that the user will get an authentication request on his mobile.

8.4.10 DELETEDOCUMENT

To delete a document the method DeleteDocument is used. The method is as follows:

```
DeleteDocumentResponse DeleteDocument(string token, string docId, out ResultStatus outStatus);
```

The variables are:

- `token`
  - string
  - The token issued from GetToken
- `docId`
  - string
  - ID of the document that shall be signed
- outStatus
  - ResultStatus
  - Return value is described further in chapter 5.1

The function returns true if the delete was successful.

8.4.11 GETDOCUMENTCONTENT

To get the text content from a document wrapped in elements the method GetDocumentContent is used. The method is as follows:

```
GetDocumentContentResponse GetDocumentContent(string token, string docId, out ResultStatus outStatus);
```

The variables are:

- `token`
  - string
  - The token issued from GetToken
- `docId`
  - string
  - ID of the document that shall be read
- outStatus
  - ResultStatus
  - Return value is described further in chapter 5.1

The function returns a string with document text if successful.

9 EXAMPLES AND INSTRUCTIONS

The code examples and instructions can be found at: https://info.signet.is

10 TEST USERS

The following users can be used for testing the process of authentication and signing.

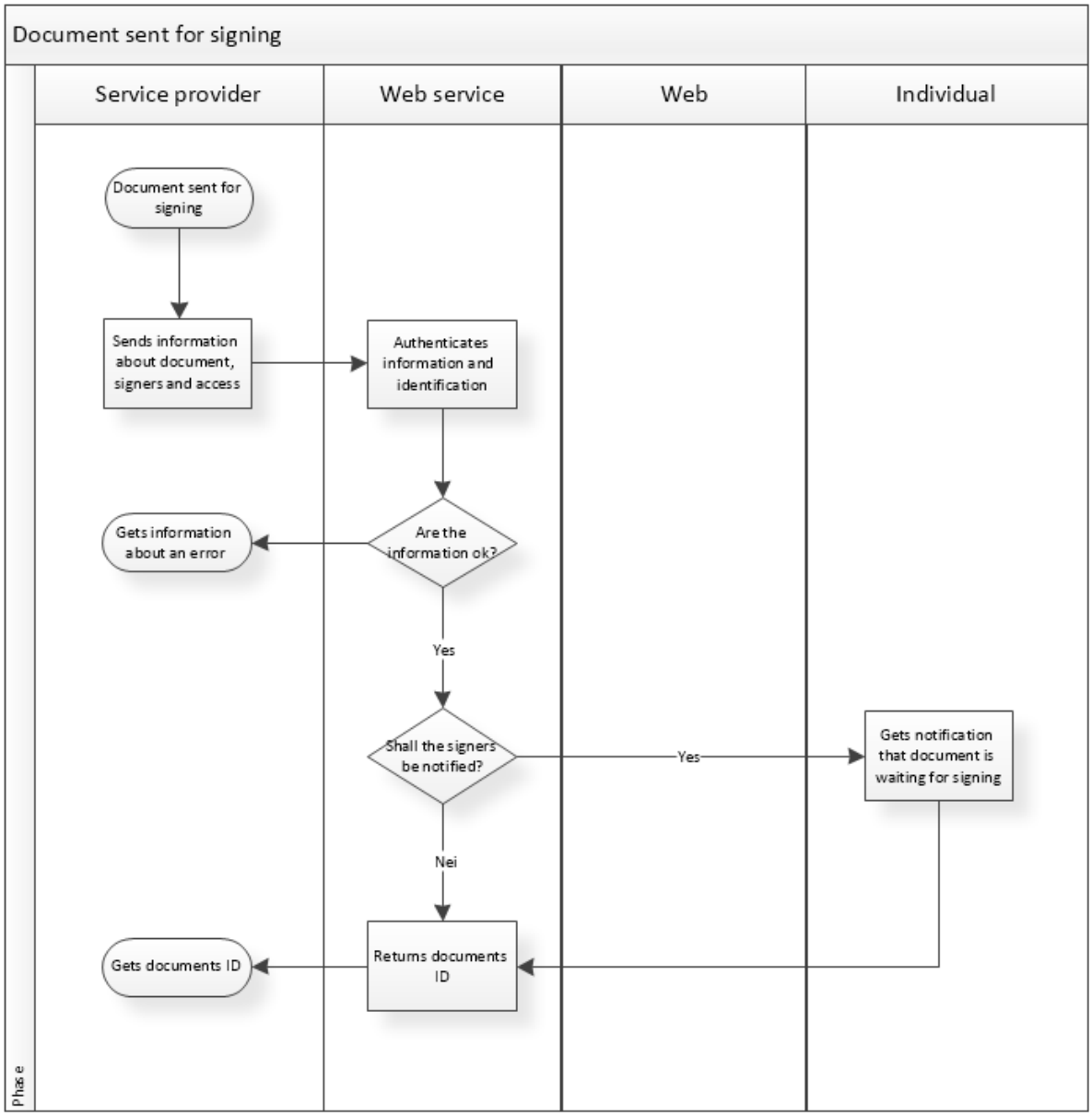
Name	SSN	Mobile	Description
Test Notandi	1234567890	+3541111111	Will authenticate and sign successfully
Test Notandi Cancel	0987654321	+3542222222	Will always cancel operation
Test Notandi Timeout	0101011234	+3543333333	Will always timeout operation
Test Notandi Problem	0202021234	+3544444444	Doesn‘t have a certificate
Test Notandi 2	1234567899	+3541111112	Will authenticate and sign successfully

11 PROCESS FOR SENDING AND SIGNING DOCUMENTS

In this chapter are process flows which describe the main operations using web services.

11.1 DOCUMENT SENT FOR SIGNING

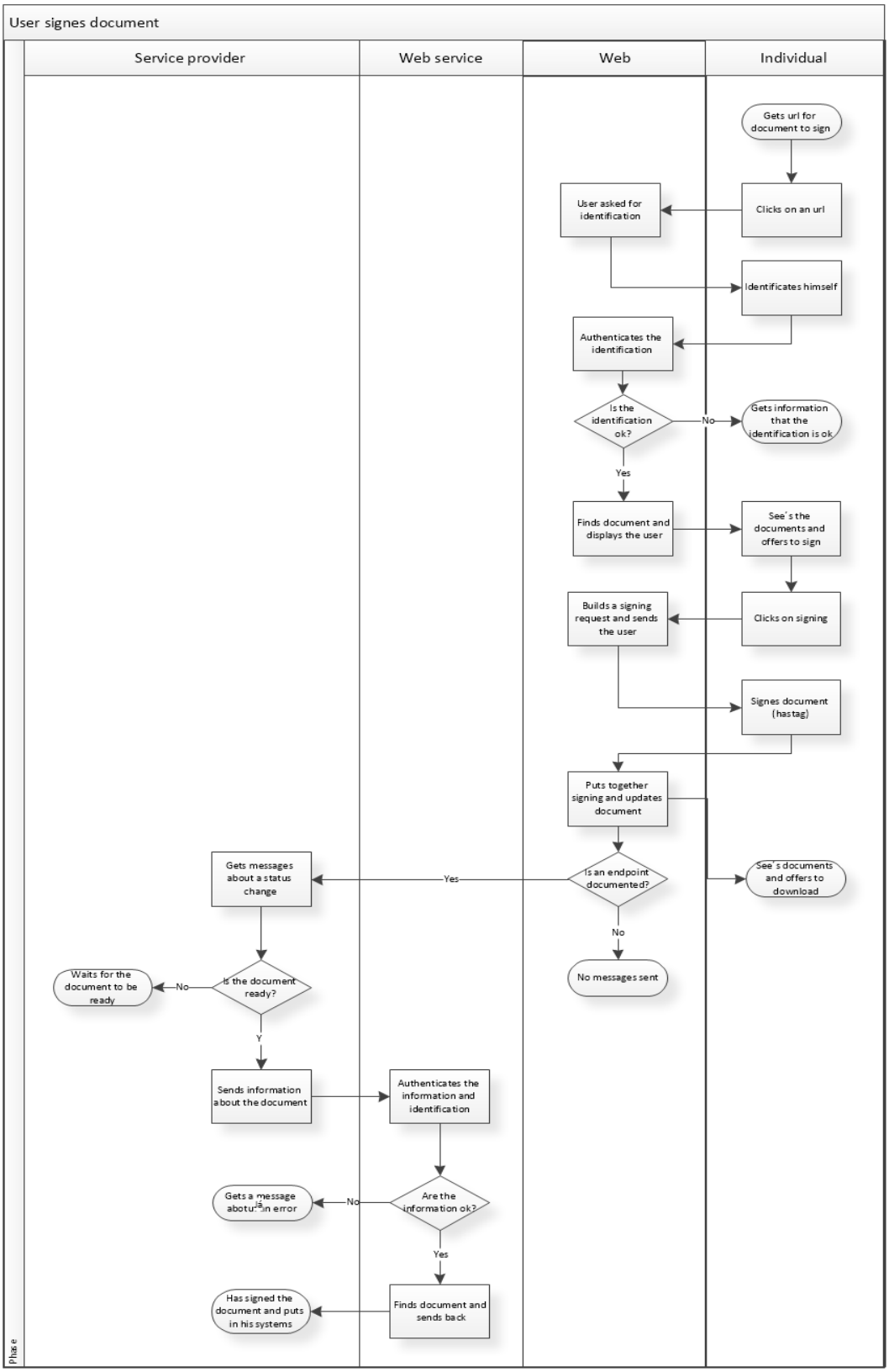
The following picture describes the process of sending a document through the web service to be signed.



Picture 1. Document sent for signing

11.2 DOCUMENT SIGNED

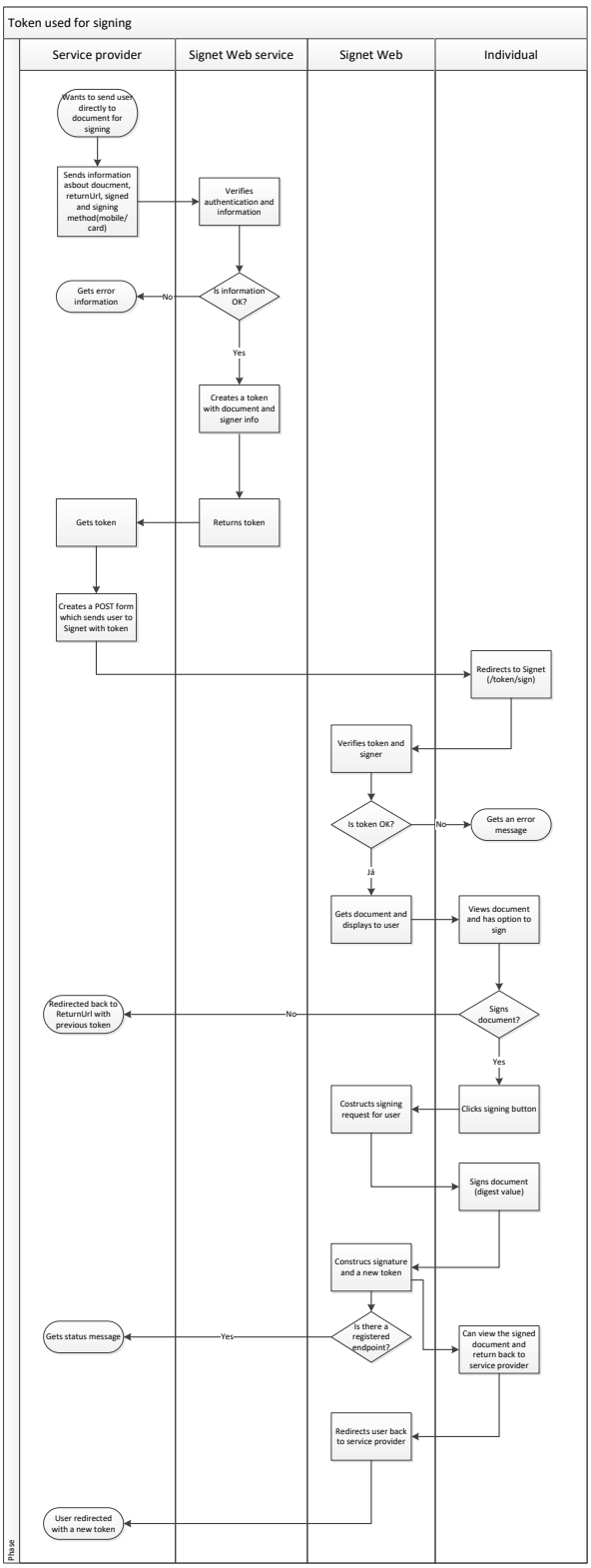
Following picture shows the process when a document sent to the web service is signed in Signet.



Picture 2. Document signed in Signet

11.3 TOKEN FOR THE DOCUMENT

Following picture describes the process when SAML token is requested for a document that has previously been sent through web service. The user is then forwarded to Signet with a token which he can use to sign.



Picture 3. Token for document used

11.3.1 REJECTING DOCUMENTS

The button to reject document is hidden by default on <Signet URL>/token/sign web interface. The button can be displayed by overriding the CSS with the following:

```
#rejectionbutton {
      display: block!important;
}
```

12 EPILOGUE

For further information about the solution and technical support contact Security solutions at Advania – signet@advania.is.

13 APPENDIX

13.1 SAML REPLY FOR SSO ON SIGNET

Here is an example of SAML (base64 decoded) received as a response from the to GetToken function.

<?xml version="1.0" encoding="UTF-8"?>
<Response xmlns="**urn:oasis:names:tc:SAML:2.0:protocol**" Destination="**https://prufa.signet.is/**" IssueInstant="**2015-06-04T10:25:40.3992909Z**" Version="**2.0**" ID="**_a2c5d7f9-edbe-4271-ac93-125c4f6dcb3a**" xmlns:xsi="**http://www.w3.org/2001/XMLSchema-instance**" xmlns:xsd="**http://www.w3.org/2001/XMLSchema**"><Issuer xmlns="**urn:oasis:names:tc:SAML:2.0:assertion**">Signet Advania</Issuer><Signature xmlns="**http://www.w3.org/2000/09/xmldsig#**"><SignedInfo><CanonicalizationMethod Algorithm="**http://www.w3.org/TR/2001/REC-xml-c14n-20010315**"/><SignatureMethod Algorithm="**http://www.w3.org/2000/09/xmldsig#rsa-sha1**"/><Reference URI="**#_a2c5d7f9-edbe-4271-ac93-125c4f6dcb3a**"><Transforms><Transform Algorithm="**http://www.w3.org/2000/09/xmldsig#enveloped-signature**"/><Transform Algorithm="**http://www.w3.org/2001/10/xml-exc-c14n#**"/></Transforms><DigestMethod Algorithm="**http://www.w3.org/2001/04/xmlenc#sha256**"/><DigestValue>U4n+brRDDxWEuGKhXf5UbJbz7RtVAIhiOQ6K0kfjr/A=</DigestValue></Reference></SignedInfo><SignatureValue>lF8lJFxT0YX0evBZpB8FBJ2z0T1HtwfA9P50Q0iMcIzGslDlIa3Xeq1E2Y4NrQqfImusSilcRG1gbcvM90g/BLwjX1E/9kNdddUduOD1vACw6D45PFYnzD1We4K872191LNzwS+PtHOoxDjLv3+eyw+x194AuKDTpuihIClS28a3LZpPdo7LcEpX1bQQEcga4sx7oTvlfMn8VSMF1SyXlrViqfpgl7YWh5+YeNSpIoBR/sOu/79/XPXNWh/WrIMxuPkWpAaUoZ0S70DXqpA0jAfgjqQggzm6S6NGXNVzz3U0ijwtvSssmvpblui1jDDXezF3sXhv9+kNFluVdUjqhg==</SignatureValue><KeyInfo><X509Data><X509Certificate>MIIF/jCCBOagAwIBAgICDkIwDQYJKoZIhvcNAQEFBQAwgZIxCzAJBgNVBAYTAklTMRMwEQYDVQQFEwo1MjEwMDAyNzkwMRYwFAYDVQQKEw1BdWRrZW5uaSBlaGYuMSMwIQYDVQQLExpVdGdlZmFuZGkgYnVuYWRhcnNraWxyaWtqYTEWMBQGA1UECxMNTWlsbGlza2lscmlraTEZMBcGA1UEAxMQVHJhdXN0dXIgYnVuYWR1cjAeFw0xNDA0MDkxNTAwNTZaFw0xNjA0MDkxNTAwNTZaMIG0MQswCQYDVQQGEwJJUzEUMBIGA1UEChMLQWR2YW5pYSBoZi4xGDAWBgNVBAsTD0J1bmFkYXJza2lscmlraTElMCMGA1UECwwcVW5kaXJyaXR1biBlw7BhIGF1w7BrZW5uaW5nIDEgMB4GCSqGSIb3DQEJARYRc2lnbmV0QGFkdmFuaWEuaXMxEzARBgNVBAUTCjU5MDI2OTcxOTkxFzAVBgNVBAMTDlNpZ25ldCBBZHZhbmlhMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAnJmNybg/TyA590K9UsKw7NKyozGodkgzcPzQkpZUFX6MfpnllxnEvp3TJcQdbk9X2NSPFtn5cYg/r5pqqZ79xr6rZp2HvzVbOGnCLdJSq4jirwPuSpYe9/3Kg+izML0wBPK/91PT42hLvP4Frz2trLgGMCI1TXuCXbpIlWHNd2sMjC5LhPftDzwNrjIYNPHmXuIk4h7ky7u8JRFVPDRBvE3aBRBdC+PQ8IH6VLSvmcSp2Z7U3uzyz4+jgrogRsXlmy0eUYWKKAaCtDuJVeAEv/BCEa1xFutSBQQ6IMngmMzJw8i9cYvmQdDEfcdz4R2

vu9HFM3W2PMB2xfVNBzsMqQIDAQABo4ICODCCAjQwDAYDVR0TAQH/BAIwADCCARwGA1UdIASCA
RMwggEPMIIBCwYJYIJgAQIBAQQBMIH9MIHEBggrBgEFBQcCAjCBtxqBtFRoaXMgY2VydGlmaWNhdGU
gaXMgaW50ZW5kZWQgZm9yIGRpZ2l0YWwgc2lnbmF0dXJlcyBhbmQgYXV0aGVudGljYXRpb24uIFRoa
XMgY2VydGlmaWNhdGUgZnVsZmlscyB0aGUgcmVxdWlyZW1lbnRzIG9mIG5vcm1hbGl6ZWQgY2VydG
lmaWNhdGUgcG9saWN5IChOQ1ApIGRlZmluZWQgaW4gRVRTSSBUUyAxMDIgMDQyLjA0BggrBgEFBQ
cCARYoaHR0cDovL2NwLmF1ZGtlbm5pLmlzL3RyYXVzdHVyYnVuYWR1ci9jcDBzBggrBgEFBQcBAQRnM
GUwIwYIKwYBBQUHMAKGF2h0dHA6Ly9vY3NwLmF1ZGtlbm5pLmlzMD4GB2CCYAIBYwaGM2h0dHA6
Ly9jZHAuYXVka2VubmkuaXMvc2tpbHJpa2kvdHJhdXN0dXJidW5hZHVyLnA3YjALBgNVHQ8EBAMCBeA
wHwYDVR0jBBgwFoAUb+zbESwBA2sbYK62+GjZQAHNjjgwQgYDVR0fBDswOTA3oDWgM4YxaHR0cDo
vL2NybC5hdWRrZW5uaS5pcy90cmF1c3R1cmJ1bmFkdXIvbGF0ZXN0LmNybDAdBgNVHQ4EFgQUfgNN
OMH8NKPhYw/La9fDiKNdPYEwDQYJKoZIhvcNAQEFBQADggEBAJP/GlrxYA6zB+WBrL/io/kMqgYjPtMLJ
b1HgQD4zxVc2k237PSREDbTWFN6jU6LLlGco4hfxdXSj5NXgvoqVWGAhH4cT1TRKp2ioLK3gFrwLLUrdo
sTxePsvtK+sSfc9TYPGITim7i2KQRn0HSenLSEaqEH4BNluH3l06WlrAvbG+BC8z9OQ7L3t8topIXHA0ee/R
Ns7164q8TLEzMGGqJLvxuUTEATGSWTpvQ9zSwQA7+ZZAtKt9Jtqt+l7+GdCQvHdp6Q3cAas15IXWUR2J
FZNGfh/vEhQa79e5XKxfQeZDnT9yShVRGA3Oo7Kn8qlUR3vDskjWd3M/ERt4k/eiU=</X509Certificate>
</X509Data></KeyInfo></Signature><Status><StatusCode
Value="**urn:oasis:names:tc:SAML:2.0:status:Success**"/></Status><Assertion
xmlns="**urn:oasis:names:tc:SAML:2.0:assertion**" IssueInstant="**2015-06-04T10:25:40.3992909Z**"
Version="**2.0**" ID="**_36775615-df51-4487-beea-7478e2c21a50**"><Issuer>Signet
Advania</Issuer><Subject><NameID
NameQualifier="**signet.is**">Undirritunarþjónusta</NameID><SubjectConfirmation
Method="**urn:oasis:names:tc:SAML:2.0:cm:bearer**"><SubjectConfirmationData
Recipient="**https://prufa.signet.is/**" NotOnOrAfter="**2015-06-04T10:35:40.3992909Z**"
Address="**127.0.0.1**"/></SubjectConfirmation></Subject><Conditions NotOnOrAfter="**2015-06-
04T10:35:40.3992909Z**" NotBefore="**2015-06-
04T10:24:40.3992909Z**"><AudienceRestriction><Audience>prufa.signet.is</Audience></AudienceRe
striction></Conditions><AuthnStatement AuthnInstant="**2015-06-
04T10:25:40.3992909Z**"><SubjectLocality
Address="**172.16.193.82**"/><AuthnContext><AuthnContextClassRef>urn:oasis:names:tc:SAML:2.0:ac
:classes:X509</AuthnContextClassRef></AuthnContext></AuthnStatement><AttributeStatement><A
ttribute NameFormat="**urn:oasis:names:tc:SAML:2.0:attrname-format:basic**"
Name="**UserSSN**"><AttributeValue
xsi:type="**xsd:string**">1909825569</AttributeValue></Attribute><Attribute
NameFormat="**urn:oasis:names:tc:SAML:2.0:attrname-format:basic**"
Name="**Name**"><AttributeValue xsi:type="**xsd:string**">Sveinbjörn
Óskarsson</AttributeValue></Attribute><Attribute
NameFormat="**urn:oasis:names:tc:SAML:2.0:attrname-format:basic**"
Name="**RequesterSSN**"><AttributeValue
xsi:type="**xsd:string**">5902697199</AttributeValue></Attribute><Attribute
NameFormat="**urn:oasis:names:tc:SAML:2.0:attrname-format:basic**"
Name="**DocumentID**"><AttributeValue xsi:type="**xsd:string**">d195d2de-da61-4560-8425-
9e6149bf9607</AttributeValue></Attribute><Attribute
NameFormat="**urn:oasis:names:tc:SAML:2.0:attrname-format:basic**"
Name="**Mobile**"><AttributeValue xsi:type="**xsd:string**">+354-
8471543</AttributeValue></Attribute><Attribute
NameFormat="**urn:oasis:names:tc:SAML:2.0:attrname-format:basic**"
Name="**Authentication**"><AttributeValue
xsi:type="**xsd:string**">SimCertificate</AttributeValue></Attribute><Attribute
NameFormat="**urn:oasis:names:tc:SAML:2.0:attrname-format:basic**"
Name="**ReturnURL**"><AttributeValue
xsi:type="**xsd:string**">http://localhost:59393/home/signed</AttributeValue></Attribute></Attribute
Statement></Assertion></Response>

13.2 SAML RETURN AFTER SIGNING WITH SSO

Here is an example of SAML (bas64 dedoded) which is returned with the user after signing with SSO.


```
<?xml version="1.0" encoding="UTF-8"?>
<Response xmlns="urn:oasis:names:tc:SAML:2.0:protocol"
Destination="http://localhost:59393/home/signed" IssueInstant="2015-06-04T10:12:31.9081247Z"
Version="2.0" ID="_bcacc9ca-c795-40ba-b227-acfc3aaa9498"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"><Issuer
xmlns="urn:oasis:names:tc:SAML:2.0:assertion">Signet Advania</Issuer><Signature
xmlns="http://www.w3.org/2000/09/xmldsig#"><SignedInfo><CanonicalizationMethod
Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/><SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/><Reference URI="#_bcacc9ca-c795-
40ba-b227-acfc3aaa9498"><Transforms><Transform
Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/><Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/></Transforms><DigestMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/><DigestValue>P6MeDPUVO9rrniKdYvTf
+ZeLOSlS4hcD/CYcKHcUd2A=</DigestValue></Reference></SignedInfo><SignatureValue>SZdaTW3Y
Tk9m05pkEGy7TlasafAp2BOwiDaLW9d+9byH+oRi5E4Fs/No6rlr/WzkjnzfEVBUYTD90DiKFWz6s0CEwE
uDCMs0X3Rlm/vV56d3GBs5gPrRZpxW54oTLYLFpqiQ+S1ty4XdNV4lI+myBvhiJA+x4sJCTtQtceG4a43+zJ
pn91mDuoqkaMUPtHWyB05WtmkeseUDpDJzcv9MPJWC2wwYOwKhtrqo9JyOxZqRDF25yE22YCIEee
NzQ4UdjaE8ByBJ8StrfKHImt1vqLVmLdmnXa5rLzytzQcboq3/pY4ObCdCONhpGxJhS1YXF/5/Buei1QqM
0ChrBVpZtA==</SignatureValue><KeyInfo><X509Data><X509Certificate>MIIF/jCCBOagAwIBAgICDkI
wDQYJKoZIhvcNAQEFBQAwgZIxCzAJBgNVBAYTAklTMRMwEQYDVQQFEwo1MjEwMDAyNzkwMRYwF
AYDVQQKEw1BdWRrZW5uaSBlaGYuMSMwIQYDVQQLExpVdGdlZmFuZGkgYnVuYWRhcnNraWxyaWt
qYTEWMBQGA1UECxMNTWlsbGlza2lscmlraTEZMBcGA1UEAxMQVHJhdXN0dXlgYnVuYWR1cjAeFw0x
NDA0MDkxNTAwNTZaFw0xNjA0MDkxNTAwNTZaMIG0MQswCQYDVQQGEwJJUzEUMBIGA1UEChML
QWR2YW5pYSBoZi4xGDAWBgNVBAsTD0J1bmFkYXJza2lscmlraTElMCMGA1UECwwcVW5kaXJyaXR1bi
Blw7BhIGF1w7BrZW5uaW5nIDEgMB4GCSqGSIb3DQEJARYRc2lnbmV0QGFkdmFuaWEuaXMxEzARBg
NVBAUTCjU5MDI2OTcxOTkxFzAVBgNVBAMTDlNpZ25ldCBBZHZhbmlhMIIBIjANBgkqhkiG9w0BAQEFA
AOCAQ8AMIIBCgKCAQEAnJmNybg/TyA590K9UsKw7NKyozGodkgzcPzQkpZUFX6MfpnllxnEvp3TJcQdb
k9X2NSPFtn5cYg/r5pqqZ79xr6rZp2HvzVbOGnCLdJSq4jirwPuSpYe9/3Kg+izML0wBPK/91PT42hLvP4Frz
2trLgGMCI1TXuCXbpIlWHNd2sMjC5LhPftDzwNrjIYNPHmXuIk4h7ky7u8JRFVPDRBvE3aBRBdC+PQ8IH6
VLSvmcSp2Z7U3uzyz4+jgrogRsXlmy0eUYWKKAaCtDuJVeAEv/BCEa1xFutSBQQ6IMngmMzJw8i9cYvm
QdDEfcdz4R2vu9HFM3W2PMB2xfVNBzsMqQIDAQABo4ICODCCAjQwDAYDVR0TAQH/BAIwADCCARw
GA1UdIASCARMwggEPMIIBCwYJYIJgAQIBAQQBMIH9MIHEBggrBgEFBQcCAjCBtxqBtFRoaXMgY2VydGl
maWNhdGUgaXMgaW50ZW5kZWQgZm9yIGRpZ2l0YWwgc2lnbmF0dXJlcyBhbmQgYXV0aGVudGljYXR
pb24uIFRoaXMgY2VydGlmaWNhdGUgZnVsZmlscyB0aGUgcmVxdWlyZW1lbnRzIG9mIG5vcm1hbGl6Z
WQgY2VydGlmaWNhdGUgcG9saWN5IChOQ1ApIGRlZmluZWQgaW4gRVRTSSBUUyAxMDIgMDQyLjA0
BggrBgEFBQcCARYoaHR0cDovL2NwLmF1ZGtlbm5pLmlzL3RyYXVzdHVyYnVuYWR1ci9jcDBzBggrBgEFB
QcBAQRnMGUwIwYIKwYBBQUHMAKGF2h0dHA6Ly9vY3NwLmF1ZGtlbm5pLmlzMD4GB2CCYAIBYwaG
M2h0dHA6Ly9jZHAuYXVka2VubmkuaXMvc2tpbHJpa2kvdHJhdXN0dXJidW5hZHVyLnA3YjALBgNVHQ8
EBAMCBeAwHwYDVR0jBBgwFoAUb+zbESwBA2sbYK62+GjZQAHNjjgwQgYDVR0fBDswOTA3oDWgM4
YxaHR0cDovL2NybC5hdWRrZW5uaS5pcy90cmF1c3R1cmJ1bmFkdXIvbGF0ZXN0LmNybDAdBgNVHQ4
EFgQUfgNNOMH8NKPhYw/La9fDiKNdPYEwDQYJKoZIhvcNAQEFBQADggEBAJP/GlrxYA6zB+WBrL/io/k
MqgYjPtMLJb1HgQD4zxVc2k237PSREDbTWFN6jU6LLlGco4hfxdXSj5NXgvoqVWGAhH4cT1TRKp2ioLK3
gFrwLLUrdosTxePsvtK+sSfc9TYPGITim7i2KQRn0HSenLSEaqEH4BNluH3l06WlrAvbG+BC8z9OQ7L3t8to
pIXHA0ee/RNs7164q8TLEzMGGqJLvxuUTEATGSWTpvQ9zSwQA7+ZZAtKt9Jtqt+l7+GdCQvHdp6Q3cAa
s15IXWUR2JFZNGfh/vEhQa79e5XKxfQeZDnT9yShVRGA3Oo7Kn8qlUR3vDskjWd3M/ERt4k/eiU=</X50
9Certificate></X509Data></KeyInfo></Signature><Status><StatusCode
Value="urn:oasis:names:tc:SAML:2.0:status:Success"/></Status><Assertion
```

xmlns="**urn:oasis:names:tc:SAML:2.0:assertion**" IssueInstant="**2015-06-04T10:12:31.9081247Z**" Version="**2.0**" ID="**_45b3c765-5c61-4700-ab3f-a1b42dddb76d**"><Issuer>Signet Advania</Issuer><Subject><NameID NameQualifier="**signet.is**">Undirritunarþjónusta</NameID><SubjectConfirmation Method="**urn:oasis:names:tc:SAML:2.0:cm:bearer**"><SubjectConfirmationData Recipient="**http://localhost:59393/home/signed**" NotOnOrAfter="**2015-06-04T10:22:31.9081247Z**" Address="**127.0.0.1**"/></SubjectConfirmation></Subject><Conditions NotOnOrAfter="**2015-06-04T10:22:31.9081247Z**" NotBefore="**2015-06-04T10:11:31.9081247Z**"><AudienceRestriction><Audience>localhost:59393</Audience></AudienceRestriction></Conditions><AuthnStatement AuthnInstant="**2015-06-04T10:12:31.9081247Z**"><SubjectLocality Address="**172.16.193.82**"/><AuthnContext><AuthnContextClassRef>urn:oasis:names:tc:SAML:2.0:ac:classes:X509</AuthnContextClassRef></AuthnContext></AuthnStatement><AttributeStatement><Attribute NameFormat="**urn:oasis:names:tc:SAML:2.0:attrname-format:basic**" Name="**UserSSN**"><AttributeValue xsi:type="**xsd:string**">1909825569</AttributeValue></Attribute><Attribute NameFormat="**urn:oasis:names:tc:SAML:2.0:attrname-format:basic**" Name="**Name**"><AttributeValue xsi:type="**xsd:string**">Sveinbjörn Óskarsson</AttributeValue></Attribute><Attribute NameFormat="**urn:oasis:names:tc:SAML:2.0:attrname-format:basic**" Name="**RequesterSSN**"><AttributeValue xsi:type="**xsd:string**">5902697199</AttributeValue></Attribute><Attribute NameFormat="**urn:oasis:names:tc:SAML:2.0:attrname-format:basic**" Name="**DocumentID**"><AttributeValue xsi:type="**xsd:string**">57cb1594-b6e0-442c-a678-fd262296d473</AttributeValue></Attribute><Attribute NameFormat="**urn:oasis:names:tc:SAML:2.0:attrname-format:basic**" Name="**Authentication**"><AttributeValue xsi:type="**xsd:string**">SimCertificate</AttributeValue></Attribute><Attribute NameFormat="**urn:oasis:names:tc:SAML:2.0:attrname-format:basic**" Name="**MetaData**"><AttributeValue xsi:type="**xsd:string**"/></Attribute><Attribute NameFormat="**urn:oasis:names:tc:SAML:2.0:attrname-format:basic**" Name="**Status**"><AttributeValue xsi:type="**xsd:string**">Signed</AttributeValue></Attribute></AttributeStatement></Assertion></Response>

13.3 ERROR CODES (SUBCODES)

The following table defines the subcodes for various warning and error reasons.

Result	StatusCode	Subcode	Reason
Informational	10.1	1	Unable to validate signature
Warning	20.1	1	Document not ready for signing yet
Warning	20.2	2	Document not available for signing any more
Warning	20.3	3	Document already signed/rejected
Warning	20.4	4	User not signer

Warning	20.5	5	No mobile in token
Warning	20.6	6	Could not lock document. Document locked by xxx
Warning	20.7	7	Signing location not found
Warning	20.8	8	No signet user found
Warning	20.9	9	No document found
Warning	20.10	10	No account found
Warning	20.11	11	Invalid token
Warning	20.12	12	Invalid API key
Warning	20.13	13	Invalid company
Warning	20.14	14	Invalid login
Warning	20.15	15	Token does not belong to account
Warning	20.16	16	No images for document found
Warning	20.17	17	No data for document
Warning	20.18	18	Invalid signers
Warning	20.19	19	Invalid certificate
Warning	20.20	20	Invalid SAML
Warning	20.21	21	Invalid reason
Warning	20.22	22	Phone does not match signer
Warning	20.23	23	Transaction mismatch
Warning	20.24	24	User can’t see document
Warning	20.25	25	Certificate SSN does not match request SSN
Warning	20.26	26	Signing request (ID) expired.
Warning	20.90	90	Invalid data
Warning	20.99	99	General
Warning	20.208	208	Transaction timed out
Warning	20.401	401	User cancel
Warning	20.402	402	PIN blocked
Warning	20.403	403	Card blocked
Warning	20.404	404	No key found

Warning	20.422	422	No certificate
Warning	20.425	425	Error certificate
Error	30.99	99	General

13.4 ERRORS/WARNINGS FROM AUÐKENNI

The error codes above 99 come from Auðkenni during signing and authentication. These errors are explained in the following table.

Code	Description
101	Wrong parameter. Error among the request arguments.
102	Missing parameter. An argument is missing from the request.
103	Wrong data length. A field in the request contains too long data.
104	Unauthorized access. The AP is unknown or the password is wrong.
105	Unknown client. The end user targeted by the AP is unknown to Valimo Signature Server.
107	Inappropriate data. Valimo Signature Server cannot handle the given data.
108	Incompatible interface. The minor version or the major version parameter is inappropriate or the request is not supported.
109	Unsupported profile. The AP has specified a mobile signature profile that is not supported.
208	Expired transaction. Transaction expiry date has been reached, or a timeout has elapsed.
209	OTA error. Valimo Signature Server has not succeeded to contact the end user's mobile equipment.
401	User Cancel. The end-user has cancelled the signing or already in an other transaction
402	PIN blocked. The PIN for the key to be used has been blocked
403	Card Blocked. The SIM (or signing PUK) has been blocked
404	No Key Found. Signing was requested for a key that does not exist
422	No certificate. No certificate has been found for this MSISDN.
425	Error certificate. Error in certificate validation.
900	Internal error. An internal error has occurred in Valimo Signature Server.