MSign Integration Guide

The e-Government Center
Government of Moldova

Confidential

Change Log

Version	Date	Description
0.1	27.05.2013	Initial version
0.2	15.10.2013	Added Pdf as ContentType and MultipleSignatures field
0.3	08.11.2013	SignResponse.Results is now also returned for Failed requests
0.4	30.01.2014	Added signature verification operation, test cases and structures
1.0	19.03.2014	Completed Glossary of terms
1.1	22.04.2014	Added SignedAt member to VerificationCertificate structure
1.2	16.07.2014	Fixed some typos and added some clarifications
1.3	07.08.2014	Added SignRequest.SignatureReason field
1.4	13.11.2014	Added Instrument and lang parameters to forms integration. Added sample SOAP messages
1.5	09.03.2015	Added Expired SignStatus. Added optional Language property to VerificationRequest and changed Message property in VerificationResult from VerificationMessage to a simple string. Added optional language parameter to GetSignResponse and added Message property to SignResponse.
1.6	11.04.2016	ExpectedSigner.ID is now not required for PDF signing. If not provided, it will be entered by the user on each signature request, so providing it is highly recommended.
1.7	15.09.2020	Changed test platform address from testmsign.gov.md to msign.staging.egov.md

Table of Contents

1. Executive summary

MSign is a reusable and shared platform-level service the main scope of which is to facilitate the use of digital signature and simplify integrations with various digital signature instruments.

This document describes the technical interfaces exposed by MSign for information systems that will use MSign as digital signature provider and verification utility. Its target audience is the development teams for those information systems.

The document contains all of the relevant information required for a complete understanding of MSign from the integration point of view. It contains integrations development details, security considerations and an API reference.

This document is also accompanied by a .NET sample that exemplify the main interaction scenario, i.e. requesting digital signature for a batch of contents.

2. Glossary of terms

Term	Definition
PAdES	PDF Advanced Electronic Signatures
PAdES-T	PAdES with timestamp field
SOAP	Simple Object Access Protocol
XAdES	XML Advanced Electronic Signatures
XAdES-T	XAdES with timestamp field

3. Introduction

3.1. Scope and target audience

This document describes the technical interfaces exposed by MSign for information systems that will use MSign as digital signatures provider and verification utility. Its target audience is the development teams for those information systems.

The details related to various digital signature instruments integrated with MSign are out of scope of this document.

3.2. Structure of this document

This document contains all of the relevant information required for a complete understanding of MSign from the integration point of view. It is also accompanied by samples that exemplify some integration scenarios using certain technologies.

The recommended reading sequence are the following chapters:

- System context
- Interaction scenarios
- Integration development
- Security considerations

The remaining chapters are for reference purpose.

3.3. Notations

This document contains several notation styles; the following details the styles that have a degree of significance beyond the purpose of communicating information:

Yellow Highlighted Text – Text that is highlighted in yellow irrespective of font attributes (font type, italics, bold, underlined, etc.) means that the text is waiting clarification or verification.

Red Bold Text – Text that is red in color and bold, defines an important piece of information that must be read.

Italic Bold Text – Text that is bold and italic detail actual information or scripts that need to be executed, created, and copied from or to.

4. Organizational context

4.1. Service owner

Service Owner	
Organization	e-Government Center of Moldova
Main Point of Contact	
Name	Iurie Turcanu
Position	Chief Digital Officer
E-mail	iurie.turcanu@egov.md
Technical Point of Contact	
Name	Artur Reaboi
Position	Enterprise Architect
E-mail	artur.reaboi@egov.md

5. System context

5.1. General system capabilities

MSign is a reusable and shared platform-level service the main scope of which is to facilitate the use of digital signature and simplify integrations with various digital signature instruments.

MSign is used as intermediary between various information systems and digital signature instrument providers. Digital signature providers differ significantly from the integration point of view, exposing various APIs that might involve direct user interaction through the browser to access user's cryptographic device or use of cryptographic devices that are not directly connected to user's PC. MSign integrates with these providers, hides the differences and exposes a single unified interface to information systems that require digital signature integration.

For actual signing, MSign exposes web pages that guide the user through digital signature instrument selection, instrument specific data input, actual signing progress and signing process result pages.

For digital signature verification, MSign exposes a verification web service which integrates with various certification authorities to perform the actual verification, including certificate revocation checks.

5.2. Service dependencies

MSign depends on the digital signature providers, so its availability and performance is directly influenced by the availability and performance delivered by the providers.

5.3. Protocols and standards

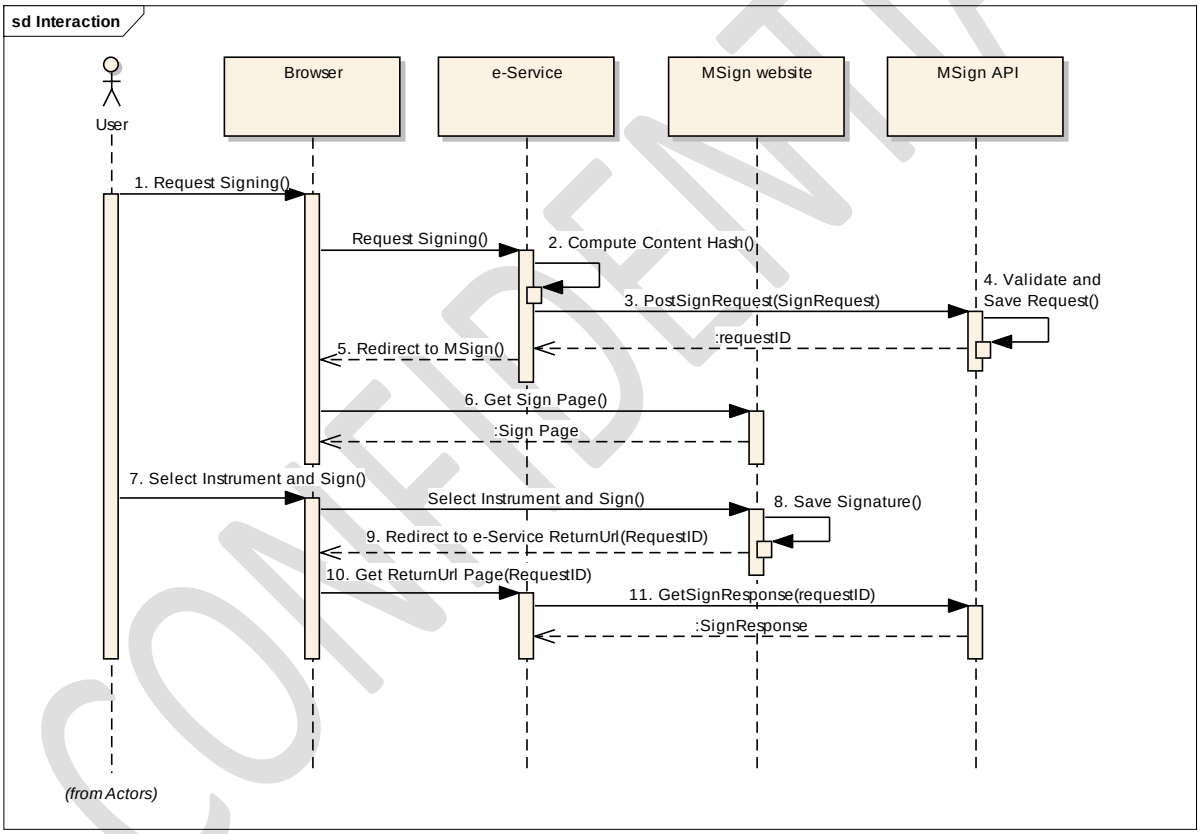
MSign exposes WS-I Basic Profile 1.1 interoperable service over HTTPS which corresponds to basicHttpBinding in WCF. MSign uses SOAP faults for error reporting.

6. Interaction scenarios

6.1. Signing process

The most important integration scenario with MSign is requesting to sign a batch (or a single) digital content and getting the signature(s) back after user interaction.

Remark. Sending a batch of digital content usually requires the user to enter the PIN for each signed content or, in the case of Mobile Signature, even to receive/send multiple SMS messages. Sending multiple contents for signing is practical only in cases when you know your users might have access to bulk messaging instruments, such as when using cryptographic tokens that cache the PIN for multiple use.



Here is a short description of signing process using MSign:

1. After completing a form, uploading a document to be signed (the Content), or selecting a batch of Contents, the User clicks a Sign button in its Browser.
2. The e-Service prepares the Contents to be signed and/or computes a hash of each Content.
3. The e-Service calls MSign API (PostSignRequest operation) with a SignRequest that represents a request to sign a batch of Contents.
4. MSign validates and saves the SignRequest for later signing and returns a generated

RequestID.

5. E-Service instructs the Browser to show the MSign Sign Page for the provided sign request, providing a ReturnUrl. See Web forms integration for more details.
6. The Browser fetches the Sign Page and shows it to User.
7. The User interacts with the Sign Page to select a signing instrument and enter any data related to the selected instrument to perform the actual signing of the batch.
8. MSign saves the resulting signatures for later retrieval.
9. MSign instructs the browser to show the ReturnUrl, providing the RequestID. See Web forms integration for more details.
10. When the Browser request the page indicated by ReturnUrl, the e-Service requests the actual SignResponse from MSign API (GetSignResponse operation). That response contains the signatures for all Contents provided in the SignRequest batch.

6.2. Verification process

MSign also exposes digital signature verification API. The verification process does not expose any user interface for integrated information systems.

To verify a batch of signatures (or a single signature), call VerifySignatures and provide the signature to be checked.

In the case of a XAdES signature (which results after signing a hash), please provide the original hash in the Content parameter and the signature (i.e. XAdES) in the Signature parameter. In the successful verification case, the result will contain a single certificate, i.e. the certificate of the signer.

To verify PDF file signatures (PAdES), just pass the signed document in the Signature parameter. In the successful verification case, the result will contain the certificates for all individual signers.

The result also contains a human readable message translated in multiple languages that the integrated systems shall display to the users.

Due to the fact that the process of verification might take more time than expected, it is advised to invoke this verification operation asynchronously so that the invoking system does not appear as blocked to the users.

7. Integration development

7.1. Obtaining credentials

MSign accepts client certificates generated by Certification Authority of the Center of Special Telecommunications (https://pki.cts.md), by requesting a certificate for authentication.

7.2. Consumer registration and network access

MSign authenticates the clients by client certificate serial number. MSign service consumers must be registered in MSign before being able to call the API.

MSign API is accessible only to a registered set of IP addresses and, for security sensitive information systems, this means configuring routes and/or a VPN between the consumer and MSign.

To register your consumer and get network access, please write a request by e-mail to the Service owner, providing your public IP address or the VPN assigned private IP address and the certificate serial number in your request.

7.3. Obtaining technical contract

The WSDL contract can be obtained from the following URL (base addresses are listed in System environments):

/MSign.svc?singleWsdl

Note that accessing the contract requires valid credentials.

An offline copy of the WSDL can be found in .NET Sample.

7.4. System environments

There are 2 services environments available: a testing and a production environment.

Environment	MSign service URL
Testing	https://msign.staging.egov.md:8443/MSign.svc
Production	https://msign.gov.md:8443/MSign.svc

It is mandatory to develop the integrations and perform tests with the testing environment.

8. Integration testing

8.1. Test cases

Here are some basic test cases that could be added to the test suite that integrates with MSign.

Test Case ID	TC_FUNCT_01		
Description	Verify the successful integration of the e-Service and MSign		
Applicable for	List of browsers as specified in requirements		
Requirements	REQ_FUNCT_XX		
Initial Conditions	Open the e-Service under test in a browser window. Go through the e-Service web pages, enter the data or upload the document to be signed and navigate to the page with Sign button, in order to perform the signing.		
Step	**Task**	**Expected Result**	**Actual Result**
1	Click the Sign button.	The MSign sign page is displayed to the user.	**Pass / Fail**
2	Select the first signing instrument and enter the data related to the selected instrument. Submit data.	The browser re-displays the e-Services page, requested signatures being successfully provided.	**Pass / Fail**
3	Repeat previous step (Step2) for each available signing instrument.	The browser re-displays the e-Services page, requested signatures being successfully provided.	**Pass / Fail**

Test Case ID	TC_FUNCT_02		
Description	Verify failed signing by cancelling the signature process		
Applicable for	List of browsers as specified in requirements		
Requirements	REQ_FUNCT_XX		
Initial Conditions	Open the e-Service under test in a browser window. Go through the e-Service web pages, enter the data or upload the document to be signed and navigate to the page with Sign button, in order to perform the signing.		
Step	**Task**	**Expected Result**	**Actual Result**
1	Click the Sign button.	The MSign sign page is displayed to the user.	**Pass / Fail**
2	Select a signing instrument that asks for a PIN or confirmation while signing (such as Mobile Signature) and cancel the signing	MSign displays that signature failed with a meaningful reason.	**Pass / Fail**
3	Review the MSign message and click OK to continue	The browser re-displays the e-Services page that clearly shows that the signature failed. Note that calling GetSignResponse for this request will return a response with SignStatus = Failure	**Pass / Fail**

Test Case ID	TC_FUNCT_03		
Description	Check sign request re-sending		
Applicable for	List of browsers as specified in requirements		
Requirements	REQ_FUNCT_XX		
Initial Conditions	Open the e-Service under test in a browser window. Go through the e-Service web pages, enter the data or upload the document to be signed and navigate to the page with Sign button, in order to perform the signing.		
Step	**Task**	**Expected Result**	**Actual Result**
1	Click the Sign button.	The MSign sign page is displayed to the user.	**Pass / Fail**
2	Go back to the e-Service page and click Sign button again.	The MSign sign page is displayed to the user.	**Pass / Fail**

Test Case ID	TC_FUNCT_04		
Description	Check correct error handling		
Applicable for	List of browsers as specified in requirements		
Requirements	REQ_FUNCT_XX		
Initial Conditions	Open the e-Service under test in a browser window. Go through the e-Service web pages, enter the data or upload the document to be signed and navigate to the page with Sign button, in order to perform the signing.		
Step	**Task**	**Expected Result**	**Actual Result**
1	Use some invalid data (such as invalid ExpectedSigner.ID, invalid MSISDN) and click the Sign button.	The e-Service displays some error message and does not redirect to MSign.	**Pass / Fail**

Test Case ID	TC_FUNCT_05		
Description	Check valid signature verification functionality		
Applicable for	List of browsers as specified in requirements		
Requirements	REQ_FUNCT_XX		
Initial Conditions	Sign some data or document in the integrated e-Service (i.e. follow TC_FUNCT_01) and navigate to the page with Verify button, in order to perform the verification.		
Step	**Task**	**Expected Result**	**Actual Result**
1	Click on Verify button.	The e-Service displays a clearly visible successful signature verification result.	**Pass / Fail**

Test Case ID	TC_FUNCT_06		
Description	Check corrupted signature verification functionality		
Applicable for	List of browsers as specified in requirements		
Requirements	REQ_FUNCT_XX		
Initial Conditions	Sign some data or document in the integrated e-Service (i.e. follow TC_FUNCT_01) and navigate to the page with Verify button, in order to perform the verification. Modify the signed data, document or the actual signature (if possible) directly in the database or in the resulting binary files.		
Step	**Task**	**Expected Result**	**Actual Result**
1	Click on Verify button.	The e-Service displays a clearly visible failed signature verification result.	**Pass / Fail**

8.2. Integrations review and audit

There are no special requirements related to integration review.

9. Security considerations

9.1. Authentication

All calls to MSign operations are authenticated by MSign. The authentication is performed by using the client certificate used for HTTPS transport.

For information regarding obtaining a client certificate and registration, see Obtaining credentials and Consumer registration and network access.

Important. The description of the process of installing, registering or explicitly trusting the obtained client certificate in the operating system or framework used by the integrating e-Service has to be done accordingly, is specific to that environment and it's out of the scope of this document.

9.2. Authorization

After successful authentication, all registered service consumers are authorized to invoke all of the exposed operations.

In the case of requesting a SignResponse that was originally placed by another service consumer, MSign will return a fault with AuthorizationFailed fault code.

9.3. Encryption

All communication with MSign SOAP service is encrypted by using standard TLS protocol (HTTPS). The client certificate used to initiate the encrypted transport is also used for Authentication.

10. API Reference

10.1. Error handling rules

For errors resulted for SOAP interface invocations, MSign returns SOAP faults with fault codes and fault reasons describing the fault in plain English. If there is no SOAP fault returned by MSign, the service consumer should expect that the returned operation result, according to MSign service contract, is valid and can be used directly without additional error checking.

Note that a SignResponse contains SignStatus, which can have Pending, Failure or Expired values, meaning there is are no signing Results returned.

Fault Code	**Description**
AuthenticationFailed	Service consumer authentication process failed. See Authentication
AuthorizationFailed	Service consumer authorization process failed. See Authorization
InvalidParameter	Some input parameter is invalid. Please review the returned Fault Reason text and called operation description.
RequestNotFound	The provided requestID when calling GetSignResponse was not found by MSign. It might be incorrect or expired (i.e. removed from online DB).

The consumers using programming languages that support try… catch blocks, catching framework specific SOAP Fault exceptions is the correct way to handle service invocation errors.

10.2. Service operations

Signature	PostSignRequest(request: SignRequest): string	
Description	Post a signature request for later signing.	
Returns	A string representing the request ID that can be later used with GetSignResponse.	
Input parameters		
Name	Type	Description
request	SignRequest	A structure representing the signature request.
Faults		
Code		Reason
AuthenticationFailed		Invalid authentication certificate provided

	Unknown service consumer: {certificate serial number}
InvalidParameter	Some input parameter is invalid. Please review the returned Fault Reason text and called operation description.

Signature	GetSignResponse(requestID: string, language: string): SignResponse	
Description	Get the status and result of the related signature request.	
Returns	A structure that contains the status and signature results.	
Input parameters		
Name	Type	Description
requestID	string	The ID of SignRequest posted earlier using PostSignRequest operation.
language	string	The language to be used for response localization. Allowed values: "ro", "ru", "en". For backward compatibility, this parameter is optional and the default value is "ro".
Faults		
Code		Reason
AuthenticationFailed		Invalid authentication certificate provided
		Unknown or unregistered system: {certificate serial number}
AuthorizationFailed		This signature request was not initiated by this system
InvalidParameter		Some input parameter is invalid. Please review the returned Fault Reason text and called operation description.
RequestNotFound		Cannot find such request

Signature	VerifySignatures(request: VerificationRequest): VerificationResponse	
Description	Request signature verification. Due to the fact that the process of verification might take more time than expected, it is advised to invoke this operation asynchronously so that the invoking application does not appear as blocked.	
Returns	A structure that contains the result and signature verification.	
Input parameters		
Name	Type	Description
request	VerificationRequest	A structure representing the verification request.
Faults		
Code		Reason
AuthenticationFailed		Invalid authentication certificate provided
		Unknown service consumer: {certificate serial number}
InvalidParameter		Some input parameter is invalid. Please review the returned Fault Reason text and called operation description.

10.3. Structures

Important. The order in which the members are described below is for description purposes only. The order of the elements in the actual XML structures, as defined in WSDL, is alphabetical. To get a correct implementation, it is recommended to use an automatic conversion tool from WSDL to your programming language or environment.

Member	Type	Required/Optional	Description
SignRequest			
ContentDescription	string (512)	Optional, default: same as ShortContentDescription	The description of the content to be signed. Displayed by MSing web pages.
ShortContentDescription	string (90)	Required	The short description of the content to be signed. Displayed by mobile phone if Mobile Signature is used.
SignatureReason	string (255)	Optional	The reason for signature, e.g. Resolution, Approved, Reviewed, etc. Currently applicable to PDF only.
ContentType	ContentType enumeration	Required	The type of the content to be signed.
Contents	Array of SignContent	Required, at least one element	The actual batch of contents to be signed.
ExpectedSigner	ExpectedSigner	Optional	If provided, MSign will verify the actual signer to match the provided information.
SignResponse			
Status	SignStatus enumeration	Required	Signature request status
Message	string (100)	Optional, returned for requests that have Failure or Expired status	Signature request failure message, localized according to language parameter.
Results	Array of SignResult	Available when Status is not Pending	Signature results for the requested signature request.
VerificationRequest			
SignedContentType	ContentType enumeration	Required	The type of the content that was previously signed.
Language	string (2)	Optional, default: ro	The language to be used for response localization. Allowed values: “ro”, “ru”,

			"en"
Contents	Array of VerificationContent	Required, at least one element	The actual batch of signatures to be verified.
VerificationResponse			
Results	Array of VerificationResult	Required	Verification results for the verification request.
SignContent			
CorrelationID	string (36)	Optional	The correlation ID for this content. Must be unique within a signature request.
MultipleSignatures	Bool	Optional, default: false	Specifies if the content could have multiple signatures (i.e. it can be co-signed). Currently, this setting applies only to PDF.
Name	string (256)	Optional	Name of the PDF file, for Hash this property is redundant.
Content	Array of byte	Required	The actual content to be signed. Currently this can be 20-bytes SHA1 hash or a PDF file.
ExpectedSigner			
ID	String	Required	Personal identifier number of the expected signer. Note that if not provided, user will be asked to enter it when signing PDF using mobile signature.
DelegatorType	DelegatorType enumeration	Optional, default: None	The type of the delegator.
DelegatorID	String	Required when DelegatorType is not None	The identifier of the person or organization that the expected signer can represent (is delegated by).
DelegatedRoleID	Int	Optional, default: 0	The role of the expected signer in relationship with the delegator.
SignResult			
CorrelationID	string (36)	Returned as in SignContent	The correlation ID for the signed content, as originally given in SignContent.
Certificate	Array of byte	Optional, present if signature succeeded	Certificate of the signer in X509 v3 format.
Signature	Array of byte	Optional, present if	For hash content type this

		signature succeeded	is the actual digital signature in XAdES-T format, for PDF content type - the signed PDF document.
VerificationContent			
CorrelationID	string (36)	Optional	The correlation ID for this content. Must be unique within a verification request.
Content	Array of byte	Required only for Hash content.	The hash that was originally signed. Note that this parameter is required only for checking hash signatures. Its value is required for complete signature verification.
Signature	Array of byte	Required	The actual signature to be verified. This must be a XAdES or signed PDF.
VerificationResult			
CorrelationID	string (36)	Returned as in VerificationContent	The correlation ID for the verification content, as originally given in VerificationContent.
SignaturesValid	Bool	Required	Returned as true if all signatures applied to the content are valid.
Message	string (100)	Required	Verification result message, localized according to VerificationRequest.Language.
Certificates	Array of VerificationCertificate	Optional, present if any certificates where identified in the signature	The list of certificates (one for signed hash in XAdES case) of the signers. Returned for display purposes.
VerificationCertificate			
SignatureValid	Bool	Required	Returned as true if the signature corresponding to this certificate is valid.
Subject	string (250)	Required	Subject details from certificate. Returned as convenience for display purposes.
Certificate	Array of byte	Required	Certificate of the signer in X509 v3 format.
SignedAt	Datetime	Optional	Date and time of the signature. Returned only if

			a valid timestamp was applied.

10.4. Enumerations

Member	Description
ContentType	
Hash	The content to be signed is a SHA1 hash.
Pdf	The content to be signed is a PDF file.
DelegatorType	
None	There is no delegator.
Person	The delegator is a person.
Organization	The delegator is an organization.
SignStatus	
Pending	The signing is pending.
Success	The signing is finished and the signature is valid.
Failure	The signing failed. The signature request is now invalid.
Expired	The signature request is expired. The signature request is now invalid.

11. Web forms integration

Method	POST (recommended) or GET
URL	https://msign.gov.md/{requestID}
Description	Direct user to perform the actual signing. Notice that the requestID is embedded in the method URL

Form or URL parameters			
Name	Type	Required/Optional	Description
ReturnUrl	string	Required	The URL that will receive the result of transaction signing
Instrument	string	Optional and not recommended	The signing instrument to be used, i.e. skipping signing instrument selection page. Allowed values: "mobile", "moldsign", "nationalid", "securesign", "tax". Note that for "mobile" instrument to work without instrument selection, you have to provide MSISDN and ExpectedSigner.ID of the expected signer.
MSISDN	string containing digits	Optional	The mobile phone number of the expected signer, if known
RelayState	string	Optional	Optional string that will be returned back unmodified after signing
lang	string	Optional	Language to be used by MSign user interface. Allowed values: "ro", "ru", "en"

Method	POST
URL	The provided ReturnUrl in the above request
Description	Redirects user to the information system that requested the signature, while informing the system about finished SignRequest processing. This Url is open only after the result of the signing is known (i.e. SignStatus is either Failure or Success).

Form parameters			
Name	Type	Required/Optional	Description
RequestID	string	Required	The ID of the finished SignRequest
RelayState	string	Optional	The unmodified value of RelayState, as sent in request

12. Samples

12.1. .NET Sample

This document is accompanied by a .NET integration sample. If the archive is encrypted (for e-mail filtering pass-through purposes), the password is "msign" (without quotes).

12.1.1. Software requirements

The sample is build using Visual Studio 2013 based on ASP.NET 4.5, MVC 5 and WCF using C# and NuGet package manager. There are no third party libraries or licenses required to build the sample.

12.1.2. Sample overview

After opening the MSign.Sample.sln, please go to Web.config and modify the appropriate attributes for clientCertificate (under MSignBehavior behaviour) to load the correct client certificate.

To ensure the correct client certificate is used, please try to access the MSign WSDL, i.e. the client endpoint address with singleWsdl parameter (https://msign.staging.egov.md:8443/MSign.svc?singleWsdl) in a browser that uses Windows Certificate Store (Internet Explorer, Chrome or Safari). The browser should ask you to select the certificate and display the WSDL.

You'll find all of the important code in HomeController. Just run it and follow the code.

Please observe the following:

- How Sign.cshtml builds a HTML form and redirects to MSign, providing the ReturnUrl point out to Home's controller Signed action;
- How HomeController.Signed method retrieves the SignResponse;
- How HomeController.Verify method constructs a VerificationRequest and handles the VerificationResponse returned by MSign;
- How the sample handles faults, by catching System.ServiceModel.FaultException.

12.2. SOAP messages

We will present here samples of exchanged SOAP messages. This might be useful for those that integrate with MSign but do not fully support WSDL-based service proxy generation.

12.2.1. PostSignRequest

Request:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    <Action s:mustUnderstand="1"
xmlns="http://schemas.microsoft.com/ws/2005/05/addressing/none">https://msign.gov.md/IMSign
/PostSignRequest</Action>
  </s:Header>
  <s:Body>
    <PostSignRequest xmlns="https://msign.gov.md">
      <request xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
        <ContentDescription>Sample long description</ContentDescription>
        <ContentType>Hash</ContentType>
        <Contents>
          <SignContent>
            <Content>ZhKVycv51rL2QoQUUEqN7tMCBkE=</Content>
            <CorrelationID>3408cc344e474a529f3425176a75d08e</CorrelationID>
          </SignContent>
        </Contents>
        <ShortContentDescription>MSign Sample.</ShortContentDescription>
      </request>
    </PostSignRequest>
  </s:Body>
</s:Envelope>
```

Response:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body>
    <PostSignRequestResponse xmlns="https://msign.gov.md">
      <PostSignRequestResult>eec7709d372b41109e2ea3e200e99727</PostSignRequestResult>
    </PostSignRequestResponse>
  </s:Body>
</s:Envelope>
```

12.2.2. GetSignResponse

Request:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    <Action s:mustUnderstand="1"
xmlns="http://schemas.microsoft.com/ws/2005/05/addressing/none">https://msign.gov.md/IMSign
/GetSignResponse</Action>
  </s:Header>
  <s:Body>
    <GetSignResponse xmlns="https://msign.gov.md">
      <requestID>eec7709d372b41109e2ea3e200e99727</requestID>
      <language>en</language>
    </GetSignResponse>
  </s:Body>
</s:Envelope>
```

Response:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body>
    <GetSignResponseResponse xmlns="https://msign.gov.md">
      <GetSignResponseResult xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
        <Results>
          <SignResult>
```

```
            <Certificate>MIIG… </Certificate>
            <CorrelationID>3408cc344e474a529f3425176a75d08e</CorrelationID>
            <Signature>PD94… </Signature>
          </SignResult>
        </Results>
        <Status>Success</Status>
      </GetSignResponseResult>
    </GetSignResponseResponse>
  </s:Body>
</s:Envelope>
```

12.2.3. VerifySignatures

Request:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    <Action s:mustUnderstand="1"
xmlns="http://schemas.microsoft.com/ws/2005/05/addressing/none">https://msign.gov.md/IMSign
/VerifySignatures</Action>
  </s:Header>
  <s:Body>
    <VerifySignatures xmlns="https://msign.gov.md">
      <request xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
        <Contents>
          <VerificationContent>
            <Content>ZhKVycv51rL2QoQUUEqN7tMCBkE=</Content>
            <CorrelationID>bd73ed7eabc44ab290b18181a9e7fd2b</CorrelationID>
            <Signature>PD94… </Signature>
          </VerificationContent>
        </Contents>
        <Language>en</Language>
        <SignedContentType>Hash</SignedContentType>
      </request>
    </VerifySignatures>
  </s:Body>
</s:Envelope>
```

Response:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body>
    <VerifySignaturesResponse xmlns="https://msign.gov.md">
      <VerifySignaturesResult xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
        <Results>
          <VerificationResult>
            <Certificates>
              <VerificationCertificate>
                <Certificate>MIIG… </Certificate>
                <SignatureValid>true</SignatureValid>
                <Subject>O=Centrul de Guvernare Electronică (e-government) 1010600034203,
OU=IT, C=MD, PostalCode=MD-2033, T=Functia, STREET=Piața Marii Adunări Naționale 1,
Phone=022250234, S=Republica Moldova, L=Chișinău, SERIALNUMBER=IDNP, CN=Nume
Prenume</Subject>
              </VerificationCertificate>
            </Certificates>
            <CorrelationID>bd73ed7eabc44ab290b18181a9e7fd2b</CorrelationID>
            <Message>The signature is valid</Message>
            <SignaturesValid>true</SignaturesValid>
          </VerificationResult>
        </Results>
      </VerifySignaturesResult>
```

```
    </VerifySignaturesResponse>
  </s:Body>
</s:Envelope>
```