SIMA WEB2APP INTEGRATION PROTOCOL

Draft version

V 1.2

Document information

Version	Date	Comment	Author
1.0	27.02.2022	Initial Version of WEB2APP	Farid Ismayilzada

1. Introduction

1.1. Scope

This document describes “web2app” secure data exchange protocol.

1.2 Abbreviations

Abb	Name	Description
RES_A	Resource authority	The client which will integrate to the identity provider
ID_P	Identity Provider	The system which provides trusted identity
TSA	Time Stamp Authority	
CA	Certificate Authority	
H()	Hash function	SHS256 , SHA1 ,etc

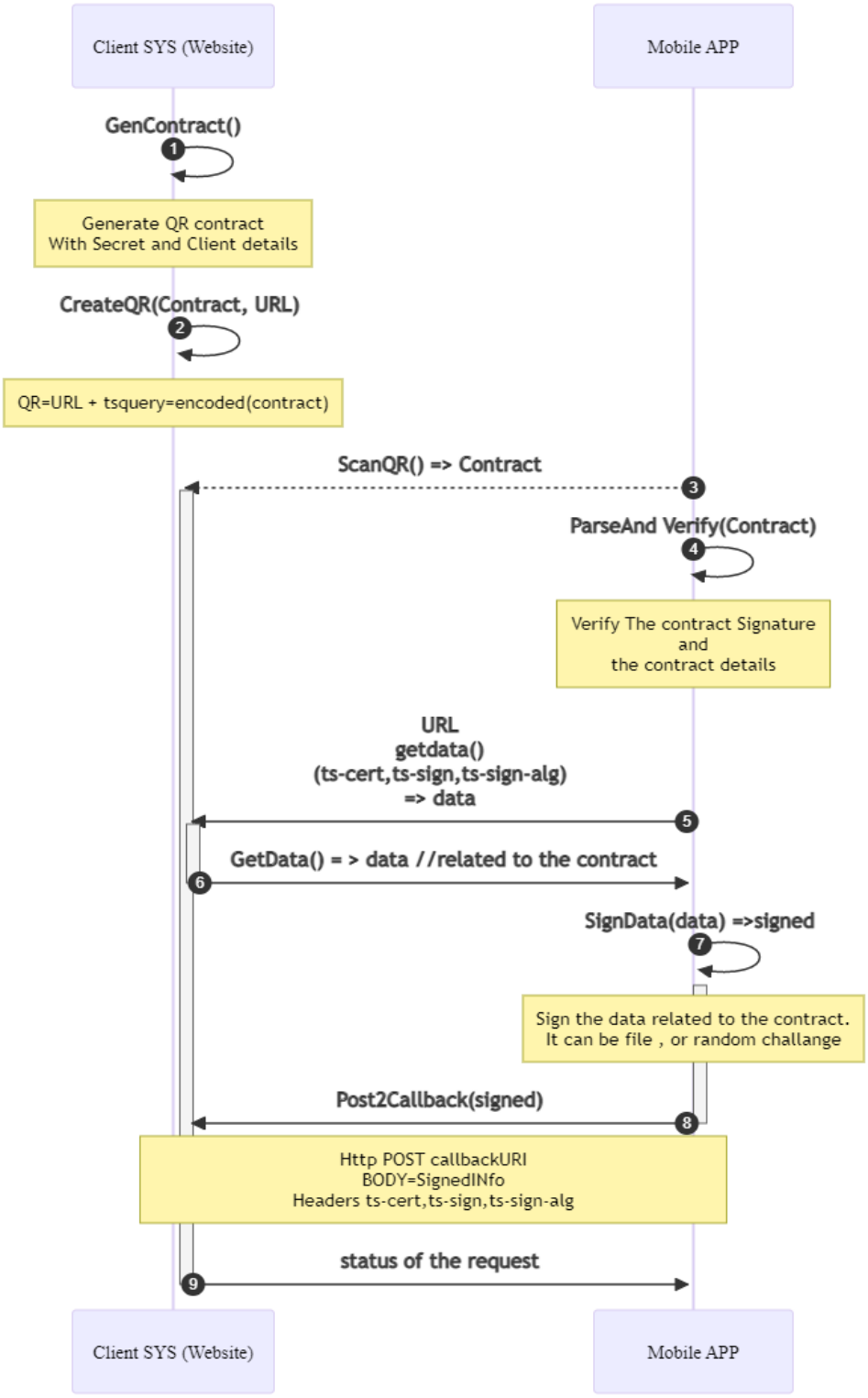
2. Information

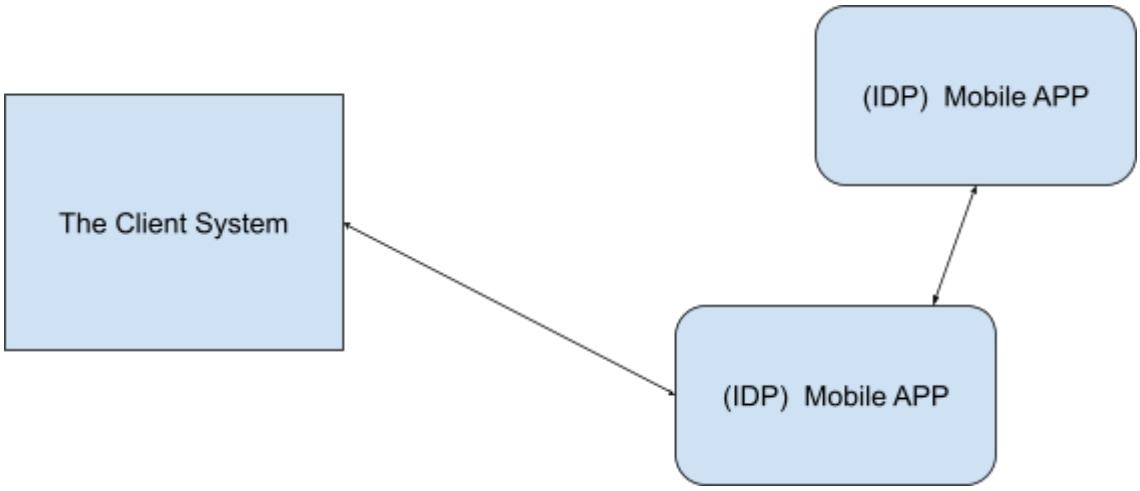
2.1 Initials

The client should have those initials parameters in order to use the protocol :

1. **Client Id -** The id provided by identity provider
2. **MasterKey -** The secret provided or approved with identity provider
3. **TheKeyAlgName -** algorithm name approved with identity provider

3. System Flow





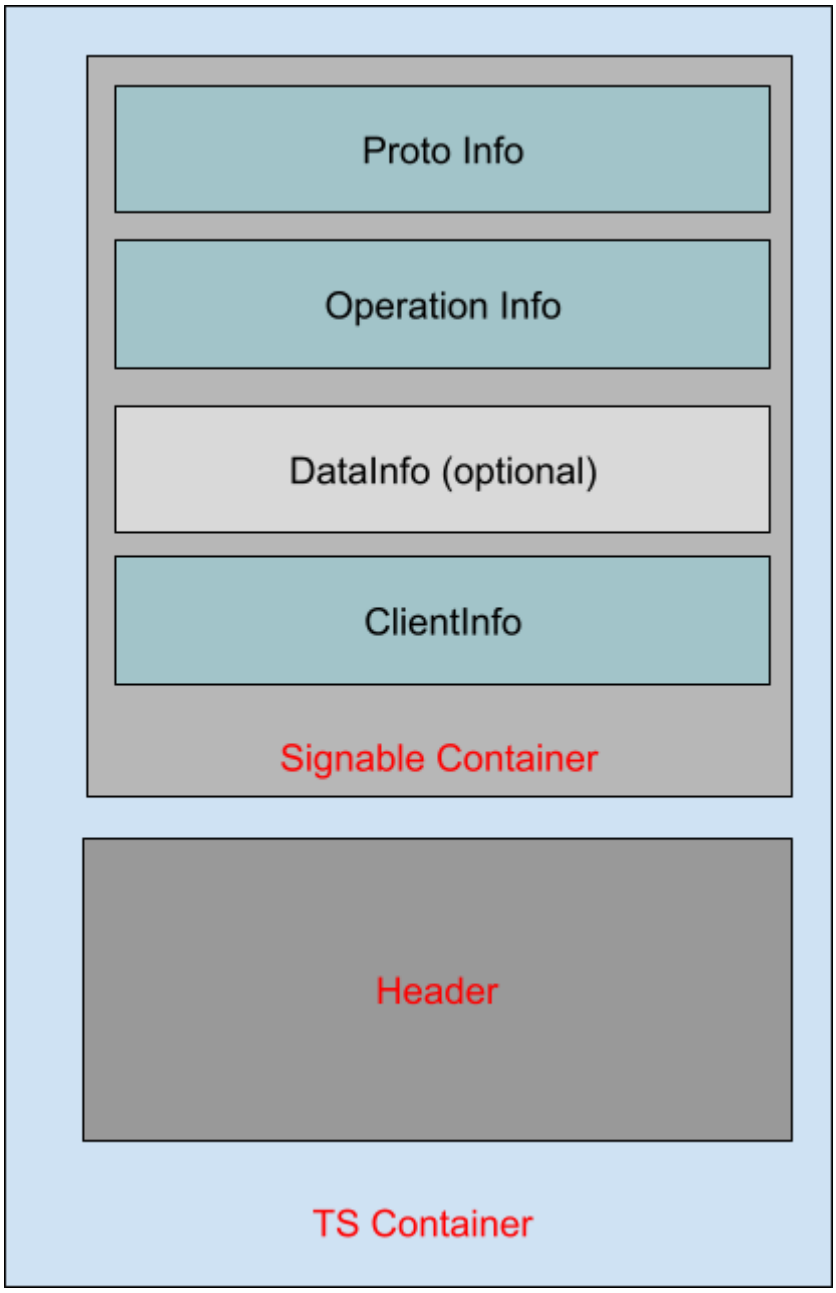
The Client system has not direct integration to the IDP(Identity Provider system). Only data exchange happening between end user app and the client system.

4. Contract Generation

4.1 Contract Details

The Contract is the main data structure for data exchange between IDP and client system.

Structure of the contract


Proto Info
Operation Info
DataInfo (optional)
ClientInfo
Signable Container
Header
TS Container

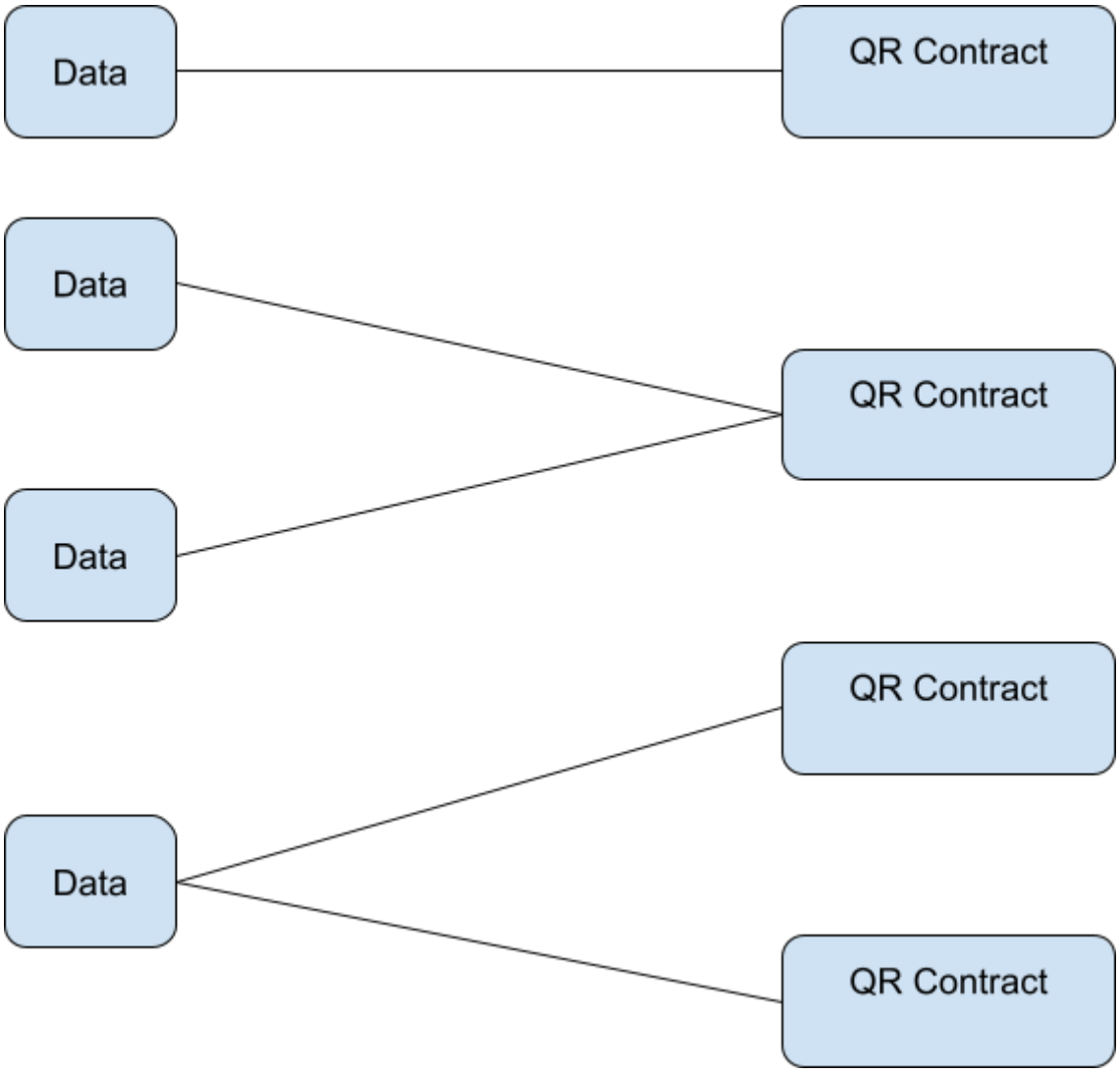
TsContainer (Contract) Version (1.0) - The main data container which contains all information, in other words the contract itself. Ts

Object Name			Description	Value
SignableContainer	ProtoInfo	Name	Protocol name.	**web2app**
		Version	The version of the protocol .	Default : 1.0
	OperationInfo	Type	Type of the operation. The reason the contract is generated.	**String enum** type : **[Auth / Sign]** **Not NUll**
		OperationId	Transaction ID for the contract. Any string data.	**Value : String** **Not NUll**
		NbfUTC	Not before or activation date as UNIX UTC timestamp.	**INT Value** Example : 1649331917
		ExpUTC	Not after or expiration date as UNIX UTC timestamp.	**INT Value** Example : 1649331930
		Assignee	People that expected to sign / to authenticate.	**Slice of string :** **PINs as string[]** **Not null**
	DataInfo (Optional)	AlgName	The name of Hash Algorithm	**SHA256**
		FingerPrint	Checksum of the data behind. base64 format	base64encodeded(#checksum)
	ClientInfo	ClientId	Client ID that is provided by IDP	**Int Value :** **Not null**
		IconURI	**Public Icon URL. (svg,Png,Jpeg ..etc)**	**Example: https://sima.az/img/logo.24d4a9b7.svg** **Not null**
		Callback	**Public callback URL of the system**	**Example: https://scanme.sima.az/Home/callback** **Not null**
Header	AlgName		**Signature Algorithm name for the contract. Mainly it uses MAC**	**Value string enum name : HMACSHA256** **Not null**
	Signature		**Base64 encoded signature.** **H()** - hash function (by default SHA256) **K** - Secret key(master Key) **S** - Signature **CH** - checksum with H() *Signature creation process:* **CH=H(SignableContainer)** **S=HMAC(CH , K) => EncodeBase64(S)**	

4.2. Contract Examples

Resource Service keeps the data according to each contract. Resource Service—the service who is going to integrate with Identity provider. It can be any web related systems.

There are no restrictions. The Same data can be related to multiple contracts, as well as the same contract can be related to multiple data.



There are two types of contracts depending on “**OperationInfo.Type**” :

- **Auth** - Authentication Contract
- **Sign** - Signature Contract

When the type is **Auth**, the data related to the contract can be random challenge data (random nonce) or some small data preferred by resource service. If the type is **Sign**, then the data related to the contract is the document, for example the PDF file.

~How to create the contract ?

In order to create the contract, you need to get client ID and the key related to that ID. We call it master key. For example.

Simple example of the Contract : Example1

```
{
  "SignableContainer": {
        "ProtoInfo": {
        "Name": "web2app",
        "Version": "1.0"
        },
        "OperationInfo": {
        "Type": "Auth",
        "OperationId": "123456789",
        "NbfUTC": 1649721600,
        "ExpUTC": 1650326400,
        "Assignee": []
        },
        "ClientInfo": {
        "ClientId": 1,
        "IconURI": "Icon Pulic URL",
        "Callback": "callbackURL"
        }
  },
  "Header": {
        "AlgName": "HMACSHA256",
        "Signature": "nQxNMasxoL1WuLJ2x1kRhFAmwFTbDxyjMGnF1Tycyr0="
  }
}
```

You can test with the contract generation tool. https://scanme.sima.az

Example1 represents the simple form of contract object where anyone with the identity mobile tool can scan and sign the data behind.

Note: If you want to assign this contract to the special individuals, you have to put their ID codes (in some cases it is PIN—Personal Identity Number) into `"Assignee": []` object list.

It can be represented as the following:

```
{
  "SignableContainer": {
        "ProtoInfo": {
        "Name": "web2app",
        "Version": "1.0"
        },
        "OperationInfo": {
        "Type": "Sign",
        "OperationId": "123456789",
        "NbfUTC": 1649721600,
        "ExpUTC": 1650326400,
        "Assignee": [“XXXXXXX” , “YYYYYYYY” , “ZZZZZZZ”]
        },
        "ClientInfo": {
        "ClientId": 1,
        "IconURI": "Icon Pulic URL",
        "Callback": "callbackURL"
        }
  },
  "Header": {
        "AlgName": "HMACSHA256",
        "Signature": "nQxNMasxoL1WuLJ2x1kRhFAmwFTbDxyjMGnF1Tycyr0="
  }
}
```

4.3. Contract representation

After building the contract, we need to represent it in a representation format.

- First, encode the contract.

  encodedContract = encode2base64(**TsContainer**)

- Represent the encoded contract

  The contract keeper parameter- **tsquery** is defined to keep the encoded contract in the representation URL.

  Example :
  https://scanme.sima.az/Home/GetFile/?tsquery={**encodedContract** }

- Convert it to the QR code.

Example :

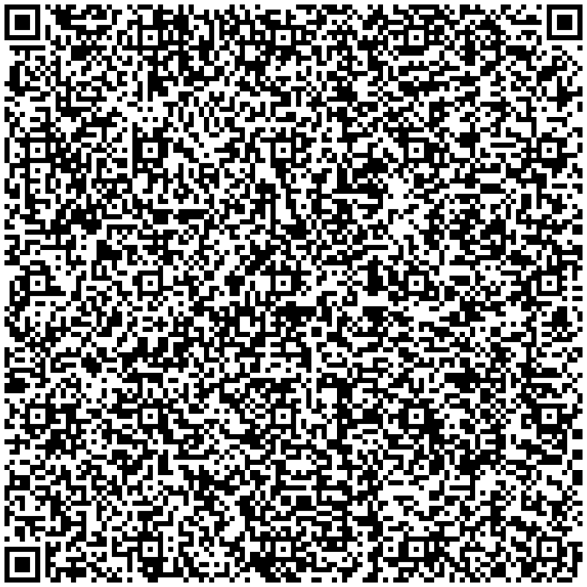
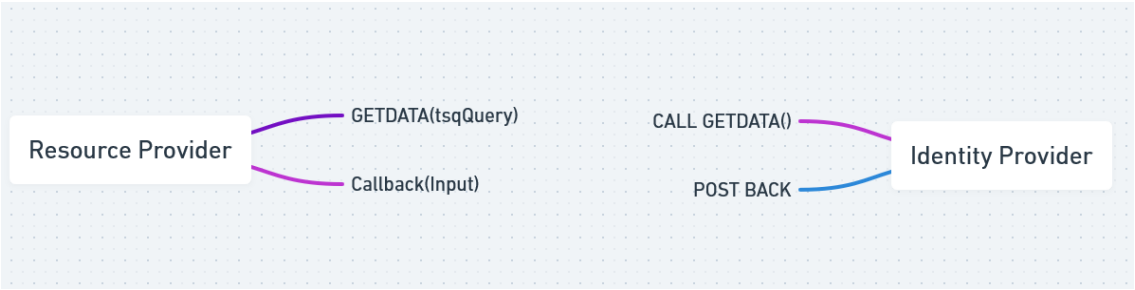


Image 1 .

5. Contract usage

5.1. GETDATA(tsQuery)

This function should be implemented by the **Resource Service** (ref 4.2). It allows us to get the data behind the generated contract. It works through the **HTTP** protocol. The **Identity provider** (ref 1.2) will always call this function with authorization parameters.



There are several ways of providing the data behind the contract. The service provider can provide the data with or without **authentication**, but in any scenario, the identity provider will always provide the identity of the end user. In other words, if it is without authentication, the service provider will bypass the authentication process (signature verification , etc).

After parsing the qr contract, the identity provider calls the GETDATA() function.

For example the QR contract behind the image 1 is :

```
https://scanme.sima.az/Home/GetFile/?tsquery=eyJTaWduYWJsZUNvbnRhaW
5lciI6eyJQcm90b0luZm8iOnsiTmFtZSI6IndlYjJhcHAiLCJWZXJzaW9uIjoiMS4wI
n0sIk9wZXJhdGlvbkluZm8iOnsiVHlwZSI6IkF1dGgiLCJPcGVyYXRpb25JZCI6IjEy
MjMxMjMxMjMiLCJOYmZVVEMiOjE2NDkyMDMyMDAsIkV4cFVUQyI6MTY1MDU4NTYwMCw
iQXNzaWduZWUiOltdfSwiQ2xpZW50SW5mbyI6eyJDbGllbnRJZCI6MSwiSWNvblVSSS
I6Imh0dHBzOi8vZmlsZXMubG9nb3NjZG4uY29tL3YxL2ZpbGVzLzQ0ODYwMjYxL2Nvb
nRlbnQuc3ZnP3NpZ25hdHVyZT1LMUFtUHdlUGlyU1dGQXVYS3IwU0xJb0dLNzAiLCJD
YWxsYmFjayI6Imh0dHBzOi8vc2Nhbm1lLnNpbWEuYXovSG9tZS9jYWxsYmFjayJ9fSw
iSGVhZGVyIjp7IkFsZ05hbWUiOiJITUFDU0hBMjU2IiwiU2lnbmF0dXJlIjoiemtqTE
1MeUt4dWlSMkNNbHVrdVpuR21rS2lKbjJvY2w0ZDI4aGZBWkVzQT0ifX0=
```

So from this Contract identity provider gets:
Domain = https://scanme.sima.az
Host = `scanme.sima.az`

```
Encodedcontract = tsquery =
eyJTaWduYWJsZUNvbnRhaW5lciI6eyJQcm90b0luZm8iOnsiTmFtZSI6IndlYjJhcHAiLC
JWZXJzaW9uIjoiMS4wIn0sIk9wZXJhdGlvbkluZm8iOnsiVHlwZSI6IkF1dGgiLCJPcGVy
YXRpb25JZCI6IjEyMjMxMjMxMjMiLCJOYmZVVEMiOjE2NDkyMDMyMDAsIkV4cFVUQyI6MT
Y1MDU4NTYwMCwiQXNzaWduZWUiOltdfSwiQ2xpZW50SW5mbyI6eyJDbGllbnRJZCI6MSwi
SWNvblVSSSI6Imh0dHBzOi8vZmlsZXMubG9nb3NjZG4uY29tL3YxL2ZpbGVzLzQ0ODYwMj
YxL2NvbnRlbnQuc3ZnP3NpZ25hdHVyZT1LMUFtUHdlUGlyU1dGQXVYS3IwU0xJb0dLNzAi
LCJDYWxsYmFjayI6Imh0dHBzOi8vc2Nhbm1lLnNpbWEuYXovSG9tZS9jYWxsYmFjayJ9fS
wiSGVhZGVyIjp7IkFsZ05hbWUiOiJITUFDU0hBMjU2IiwiU2lnbmF0dXJlIjoiemtqTE1M
eUt4dWlSMkNNbHVrdVpuR21rS2lKbjJvY2w0ZDI4aGZBWkVzQT0ifX0=
```

Tscontainer = decode(`Encodedcontract`)


```
{
  "SignableContainer": {
        "ProtoInfo": {
        "Name": "web2app",
        "Version": "1.0"
        },
        "OperationInfo": {
        "Type": "Auth",
        "OperationId": "1223123123",
        "NbfUTC": 1649203200,
        "ExpUTC": 1650585600,
        "Assignee": []
        },
        "ClientInfo": {
        "ClientId": 1,
        "IconURI":
"https://files.logoscdn.com/v1/files/44860261/content.svg?signature=K1AmPwePirSWFAu
XKr0SLIoGK70",
        "Callback": "https://scanme.sima.az/Home/callback"
        }
  },
  "Header": {
        "AlgName": "HMACSHA256",
        "Signature": "zkjLMLyKxuiR2CMlukuZnGmkKiJn2ocl4d28hfAZEsA="
  }
}
```


CallbackURL = https://scanme.sima.az/Home/callback

After taking all variables, the identity provider will call **GETDATA** as the following :

```
GET
https://scanme.sima.az/Home/GetFile/?tsquery=eyJTaWduYWJsZUNvbnRhaW5lciI6eyJQcm90b0luZm8iOnsiTmFtZSI6IndlYjJhcHAiLCJWZXJzaW9uIjoiMS4wIn0sIk9wZXJhdGlvbkluZm8iOnsiVHlwZSI6IkF1dGgiLCJPcGVyYXRpb25JZCI6IjEyMzQ1Njc4OSIsIk5iZlVUQyI6MTY0OTcyMTYwMCwiRXhwVVRDIjoxNjUwMzI2NDAwLCJBc3NpZ25lZSI6W119LCJDbGllbnRJbmZvIjp7IkNsaWVudElkIjoxLCJJY29uVVJJIjoiaHR0cHM6Ly93d3cuaW5zdGFncmFtLmNvbS9zdGF0aWMvaW1hZ2VzL3dlYi9tb2JpbGVfbmF2X3R5cGVfbG9nby0yeC5wbmcvMWI0N2Y5ZDBlNTk1LnBuZyIsIkNhbGxiYWNrIjoiaHR0cHM6Ly9zY2FubWUuc2ltYS5hei9iIb21lL2NhbGxiYWNrIn19LCJIZWFkZXIiOnsiQWxnTmFtZSI6IkhNQUNTSEEyNTYiLCJTaWduYXR1cmUiOiJuUXhOTWFzeG9MMVd1TEoyeDFrUmhGQW13RlRiRHh5ak1HbkYxVHljeXIwPSJ9fQ==
ts-cert:MIIEDjCCA2+gAwIBAgIOH9GmwMxv9DcAAAANMx0wCgYIKoZIzj0EAwMwfDELMAkGA1UEBhMCQVoxOzA5BgNVBAoTMk5hdGlvbmFsIENlcnRpZmljYXRlIFNlcnZpY2VzIENlbnRlciBvZiBBemVyYmFpamFuMTAwLgYDVQQDEydBemVyYmFpamFuIE5hdGlvbmFsIElzc3VpbmcgTW9iaWxlIENBMDEwHhcNMjIwMzE0MDY0MTIyWhcNMjUwMzEzMDY0MTIyWjB5MQswCQYDVQQGEwJBWjEuMCwGA1UEAwwlRsaPUsSwRCDEsFNNQVlJTFpBRMaPIMSwU1JBRsSwTCBPxJ5MVTEWMBQGA1UEBAwNxLBTTUFZSUxaQUTGjzEQMA4GA1UEKgwHRsaPUsSwRDEQMA4GA1UEBRMHNTZNS0ZSWTBZMBMGByqGSM49AgEGCCqGSM49AwEHA0IABJcnIQlvQGxkpDLla7NAnoKUj1hlihQMudOmsReFC1vtxXpnTilOJi3S5l8zkDClW+zfSAa2ttWwsa4ouy68+RijggHWMIIB0jAOBgNVHQ8BAf8EBAMCBsAwHQYDVR0OBBYEFLnwcNxU/4Lohi2gCDuf3pTZqaRqMB8GA1UdIwQYMBaAFI2LbEK25/guXeefWs5oS9s7fjSAMD0GA1UdHwQ2MDQwMqAwoC6GLGh0dHA6Ly9tb2JpbGUuZS1pbXphLmF6L2NkcGFpYS9BWk5JTUNBMDEuY3JsMHIGCCsGAQUFBwEBBGYwZDA4BggrBgEFBQcwAoYsaHR0cDovL21vYmlsZS5lLWltemEuYXovY2RwYWlhL0FaTklNQ0EwMS5jcnQwKAYIKwYBBQUHMAGGHGh0dHA6Ly9tb2JpbGUuZS1pbXphLmF6L29jc3AwPQYJKwYBBAGCNxUHBDAwLgYmKwYBBAGCNxUIgq3GFIGzhxG5kw2G5/lUge2pDIFhhrbOb4GAi2UCAWQCAQkwHwYDVR0lBBgwFgYIKwYBBQUHAwQGCisGAQQBgjcKAwwwKQYJKwYBBAGCNxUKBBwwGjAKBggrBgEFBQcDBDAMBgorBgEEAYI3CgMMMEIGA1UdIAQ7MDkwNwYKKwYBBAGCgEsHATApMCcGCCsGAQUFBwIBFhtodHRwczovL21vYmlsZS5hei9yZXBvc2l0cnkwCgYIKoZIzj0EAwMDgYwAMIGIAkIBZH/fNQEcV8YV89pdIq9OVdyLqP7Yxw9myb3rRiAhvP2adRXoLGvcKoPnSvl+fdYw3YFaFNLvA1BkXBJAh61Yn74CQgD+rH+iKmJ8KWMIzRiXyVf1EtEd3LoB/D3IUETaAgTnx8h1Ud7aZBCsrn5ZIxHzv545hgp4kaoIqsM0bV6Ni8O7Ig==
ts-sign-alg:ECDSA_SHA256
ts-sign:MEUCIF9scLvsvojRhBzxwRHAIXcALrkoIytHBfbcF7x5uVwLAiEA1xhAyNKAgbUHOKL2c7QAx7sLo25i90vmnS3ywm8i4Vo=
```

This method is called by the identity provider to the service provider. So `scanme.sima.az` is the service provider.

The service provider should return the http result as following :

```
HTTP/1.1 200 OK
{
```

```
  "filename": "challange",
  "data": "YjY1MmEyYTAtYmQ4NS00Njg3LThjYWMtYWMxOTk0ODYzMWMx"
}
```

5.2 Callback to Service Provider

This is the example for the authentication contract. In this case the data is random nonce. If the contract type is **“Sign”** , the data property will be the file that needs to be signed.

After taking the data from the service provider, the identity provider will sign and post it back to the service provider. Example :

```
POST https://scanme.sima.az/Home/callback
Content-Type: application/json
ts-cert:MIIEDjCCA2+gAwIBAgIOH9GmwMxv9DcAAAANMx0wCgYIKoZIzj0EAwMwfDELMAkGA1UEBhMCQVo
xOzA5BgNVBAoTMk5hdGlvbmFsIENlcnRpZmljYXRlIFNlcnZpY2VzIENlbnRlciBvZiBBemVyYmFpamFuMT
AwLgYDVQQDEydBemVyYmFpamFuIE5hdGlvbmFsIElzc3VpbmcgTW9iaWxlIENBMDEwHhcNMjIwMzE0MDY0M
TIyWhcNMjUwMzEzMDY0MTIyWjB5MQswCQYDVQQGEwJBWjEuMCwGA1UEAwwlRsaPUsSwRCDEsFNNQVlJTFpB
RMaPIMSwU1JBRsSwTCBPxJ5MVTEWMBQGA1UEBAwNxLBTTUFZSUxaQUTGjzEQMA4GA1UEKgwHRsaPUsSwRDE
QMA4GA1UEBRMHNTZNS0ZSWTBZMBMGByqGSM49AgEGCCqGSM49AwEHA0IABJcnIQlvQGxkpDLla7NAnoKUj1
hlihQMudOmsReFC1vtxXpnTilOJi3S5l8zkDClW+zfSAa2ttWwsa4ouy68+RijggHWMIIB0jAOBgNVHQ8BA
f8EBAMCBsAwHQYDVR0OBBYEFLnwcNxU/4Lohi2gCDuf3pTZqaRqMB8GA1UdIwQYMBaAFI2LbEK25/guXeef
Ws5oS9s7fjSAMD0GA1UdHwQ2MDQwMqAwoC6GLGh0dHA6Ly9tb2JpbGUuZS1pbXphLmF6L2NkcGFpYS9BWk5
JTUNBMDEuY3JsMHIGCCsGAQUFBwEBBGYwZDA4BggrBgEFBQcwAoYsaHR0cDovL21vYmlsZS5lLWltemEuYX
ovY2RwYWlhL0FaTklNQ0EwMS5jcnQwKAYIKwYBBQUHMAGGHGh0dHA6Ly9tb2JpbGUuZS1pbXphLmF6L29jc
3AwPQYJKwYBBAGCNxUHBDAwLgYmKwYBBAGCNxUIgq3GFIGzhxG5kw2G5/lUge2pDIFhhrbOb4GAi2UCAWQC
AQkwHwYDVR0lBBgwFgYIKwYBBQUHAwQGCisGAQQBgjcKAwwwKQYJKwYBBAGCNxUKBBwwGjAKBggrBgEFBQc
DBDAMBgorBgEEAYI3CgMMMEIGA1UdIAQ7MDkwNwYKKwYBBAGCgEsHATApMCcGCCsGAQUFBwIBFhtodHRwcz
ovL21vYmlsZS5hei9yZXBvc2l0cnkwCgYIKoZIzj0EAwMDgYwAMIGIAkIBZH/fNQEcV8YV89pdIq9OVdyLq
P7Yxw9myb3rRiAhvP2adRXoLGvcKoPnSvl+fdYw3YFaFNLvA1BkXBJAh61Yn74CQgD+rH+iKmJ8KWMIzRiX
yVf1EtEd3LoB/D3IUETaAgTnx8h1Ud7aZBCsrn5ZIxHzv545hgp4kaoIqsM0bV6Ni8O7Ig==
ts-sign-alg:ECDSA_SHA256
ts-sign:MEUCIGZCLjtrJntX4PLE2sToyXz2fo6oLPJj0MoBaJwSrQ4BAiEAogZRicKMYzmnF2aHqL/L0IW
OrVxAwmh/CJoT/ESlJuM=

{
 "Type": "Auth",
 "OperationId": "123456789",
 "DataSignature":
"MEUCIQCN+6Di6lGyJlhMTji0whBZQHXtyOb6e6IE4c6RnsguggIgNt0pufpX1El2vTrtGPIqPoiB96qA50
1SA7JwTI7HMCM=",
 "SignedDataHash": "uwQ7frHv20lBn3/iuaSu+yXgdM+66i5FmJoqJiubN9k=",
 "AlgName": "SHA256"
}
```

Result

```
HTTP/1.1 200 OK
Date: Tue, 19 Apr 2022 20:29:51 GMT
Content-Type: application/json; charset=utf-8
Transfer-Encoding: chunked
Connection: close

{

  "status": "success"
}
```

Callback Model

Name	Type	Value
Type	Enum string	Auth / Sign - Required
OperationId	String	required
DataSignature	String - base64	sign(`data) ->base64 required`
SignedDataHash	String -base64	Hash(data) - optional
AlgName	String enum	Hashing alg name - optional

5.3 Headers for identity provider requests

The required header values provider by the identity provider for each request :

Name	type	value
`ts-sign-alg`	String enum	The signature algorithm name
`ts-cert`	String - base64	The x509 certificate of the client
`ts-sign`	String - base64	Signature of the request If the http **Method = POST** Signature = sign(request.Body,`ts-sign-alg` , privKey) **Method = GET** sign(qrcontract - domain,`ts-sign-alg` , privKey)