SIMA WEB2APP INTEGRATION PROTOCOL

Draft version

V 1.0

Feb 27, 2022

Farid Ismayilzada

The documentation is designed to implement integration between mobile SIMA app and web applications.

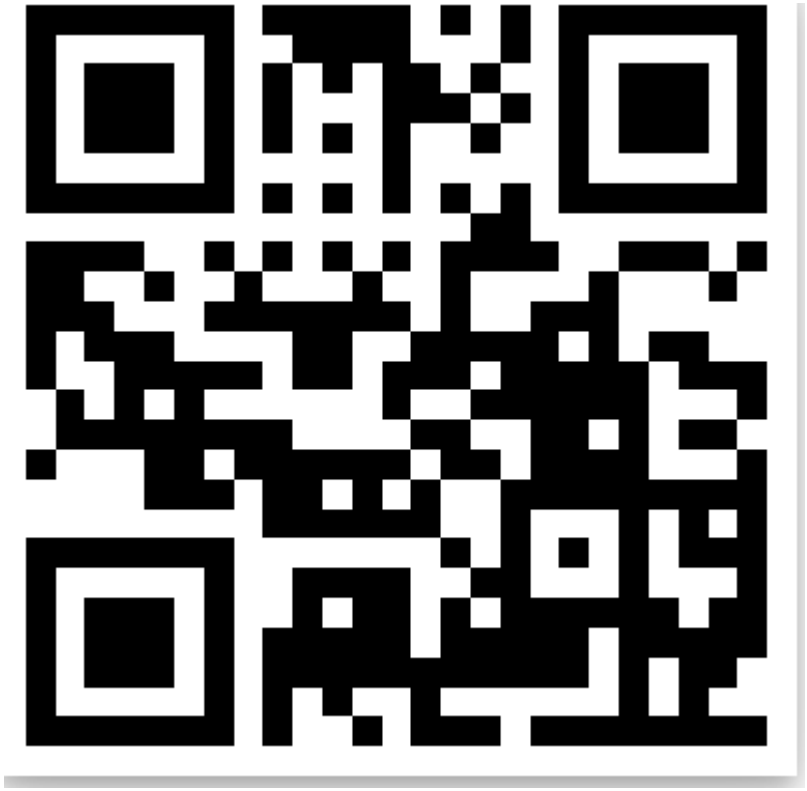
For example : Qr

Stages:

1. Generate the Qr code

   How to generate the QR code ? Qr contains the URI of the resource provider. For example

https://sima.az/?tsquery=exchangecontract(base64)



What is exchange contract ?

The data instructions of the application that will scan that data, decode and verify .

QR code -> URI

URI = “https:// *******.com*******tsquery=base64”

//https://sima.az/?tsquery=base64

tsquery= base64(jsonContainer)

Just build the dataformat .
Here is 2 main containers

- SignableContainer - Which contains all the signed information about the operation
- Header - Unsigned information and signature algorithm info with the digest itself

1. Web portal generates the QR with the URI.
2. Mobile APP Scan the QR code and decode.
3. Mobile APP verify the QR code validity and authenticity. Define the issuer , validate issuer signature etc ...
4. Request to the URI with Certificate based authentication .

Http GET https://example.com/api/sima/?tsquery=base64

Headers

- ts-cert: certificate(base64)
- ts-sign: signature(base64)
- ts-sign-alg: Signature algorithm name (example : ECDSA_SHA256)

Ts-cert - the user certificate file in base64 . This is the identity

Ts-sign- the signature of the request that created by the user private key within the app.

Ts-sign-alg : the algorithm of the signature.

Response in case of success :

```
{
  "data": “base64”
}
```

The data for the signing in App .

5 . The app will sign the data .

And post it into the callback URI that has been provided in encoded exchange data.

Body:

```
{
  "Type": null, //auth / sign
  "OperationId": null, // transaction id that the  portal provided
  "DataSignature": null, // Signature
  "SignedDataHash": null,
  "AlgName": null
}
```

Headers :

- ts-cert: certificate(base64)
- ts-sign: signature(base64)
- ts-sign-alg: Signature algorithm name (example : ECDSA_SHA256)

6. The web app will compare the data that has been generated by him and verify it with reposted data. If correct, then the system will take the subject field from **ts-cert** and the user can be onboarded.

Name	Description	Value
SignableContainer	Container with Signable attributes	Below
.ProtoInfo	The protocol information object	
Name	ProtoName	SimaWeb2Sign
Version	Version of the protocol	1.0
.OperationInfo	The operation information	
. Type	Type of the operation	Auth / Sign
.OperationId	Transaction id	
.NbfUTC	Not before	UnixTime stamp UTC
.ExpUTC	Not after	UnixTime stamp UTC
.Limit	Operation Limit. Should be controlled by the client	1,2,3,4 .... * * - unlimited
.Assignee	People that expected to sign / to authenticate	PIN codes ☐ Optional
.DataInfo	The information about the data hosted in URI	
.AlgName	Digest algorithm name	Base SHA256
FingerPrint	The hash of the data hosted in URI	Hash(Data)
.ClientInfo	Client information	
. ClientId	The ID of the client	Will be provided

.IconURI	Public icon of the client	
. Callback	Callback URI from Client	
Header	Unsigned Container	Contains the information about the signature of the SignableContainer .
.AlgName	Algorithm Of the Signature	HMACSHA256 By default
.Signature	HMAC (secretkey , SignableContainer)	Signature of the SignableContainer

Exchange Input format :TSContainer


```
{
  "SignableContainer": {
      "ProtoInfo": {
      "Name": null,
      "Version": null
      },
      "OperationInfo": {
      "Type": null,
      "OperationId": null,
      "NbfUTC": 0,
      "ExpUTC": 0,
      "Limit": null,
      "Assignee": null
      },
      "DataInfo": {
      "AlgName": null,
      "FingerPrint": null
      },
      "ClientInfo": {
      "ClientId": null,
      "IconURI": null,
      "Callback": null
      }
  },
  "Header": {
      "AlgName": null,
      "Signature": null
  }
}
```