



SIMA WEB2APP INTEGRATION PROTOCOL

V 1.3

Farid Ismayilzada
Head of Trust Services

Document Versioning

Version	Date	Comment	Author
1.3	31.03.2022	Initial Version of WEB2APP	Farid Ismayilzada
1.4	21.03.2022	<ul style="list-style-type: none">• Update of protocol version : web2app 1.1• Added ClientName to the field ClientInfo• Added DataURI to the field DataInfo Check the references : 4.1 , 4.3 , 5.1	Farid Ismayilzada
1.5	30.07.2022	Added RedirectUri field SignableContainer.ClientInfo.RedirectURI in protocol version 1.3	Farid Ismayilzada

1. Introduction	4
1.1 Scope	4
1.2 Abbreviations	4
2. Information	5
2.1 Initials	5
3. System Flow	6
4. Contract Generation	7
4.1 Contract Details	7
4.2. Contract Examples	10
4.3. Contract representation	12
5. Contract usage	13
5.1. GETDATA(tsQuery)	13
5.2 Callback to Service Provider	17
5.3 Headers for identity provider requests	19

1. Introduction

1.1 Scope

This document describes “web2app” secure data exchange protocol. In this particular document, we have two sides: **provider** and **consumer**. Furthermore, there are two parties as well: **service provider** and **identity provider**(ref 1.2).

1.2 Abbreviations

Abb	Name	Description
Ser_p	Service Provider	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

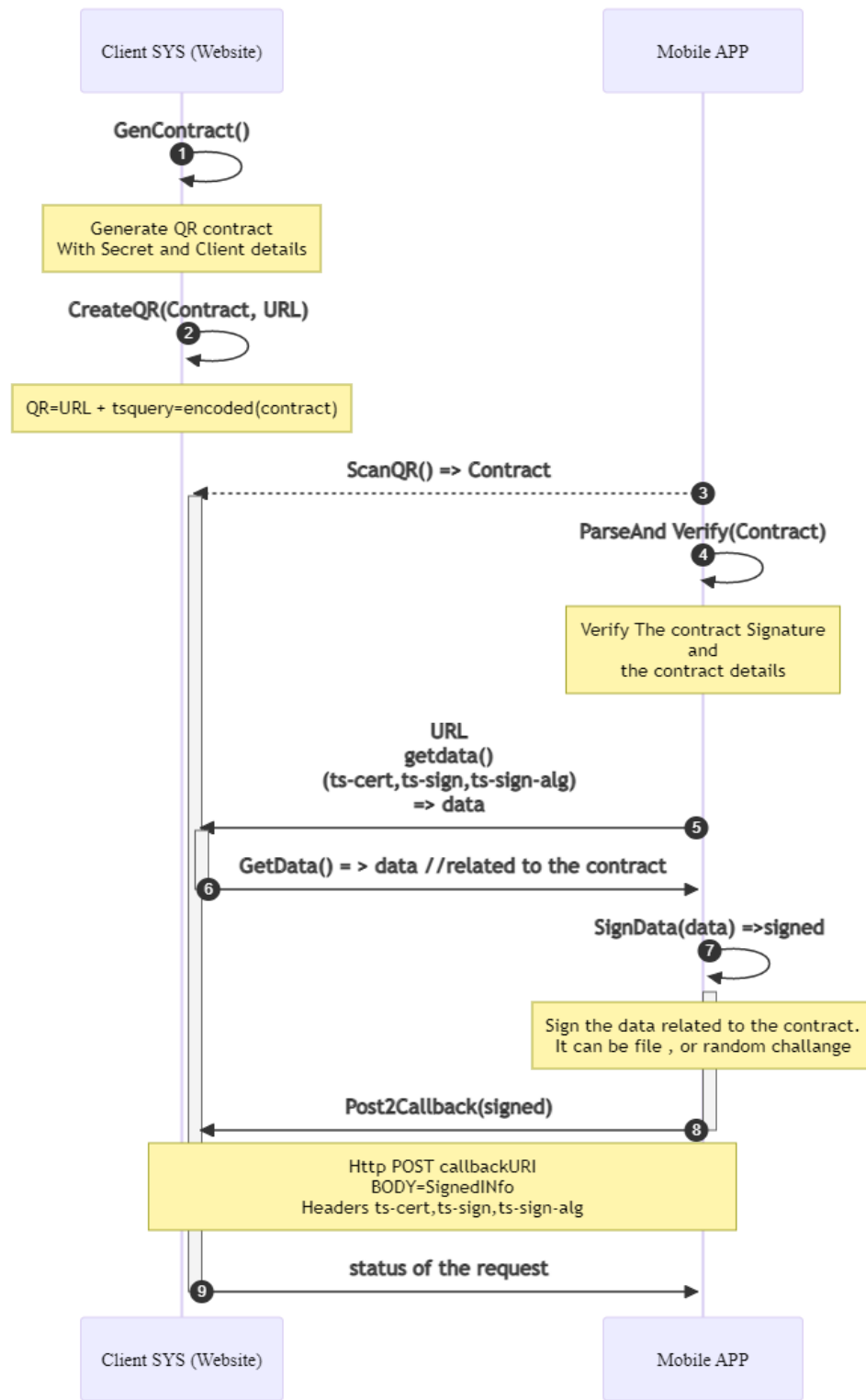
2. Information

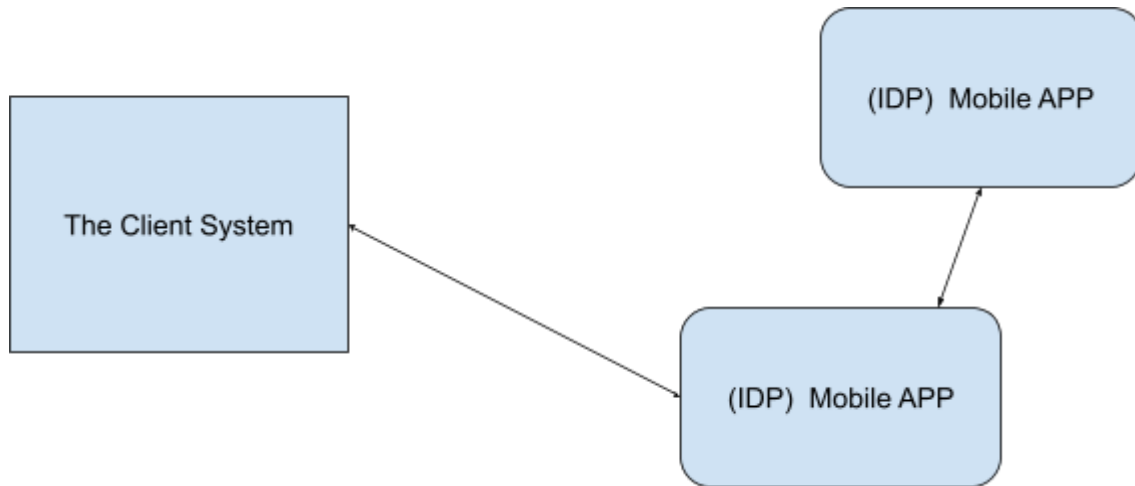
2.1 Initials

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
4. **Trent as Trusted root** - Trusted root certificate

3. System Flow





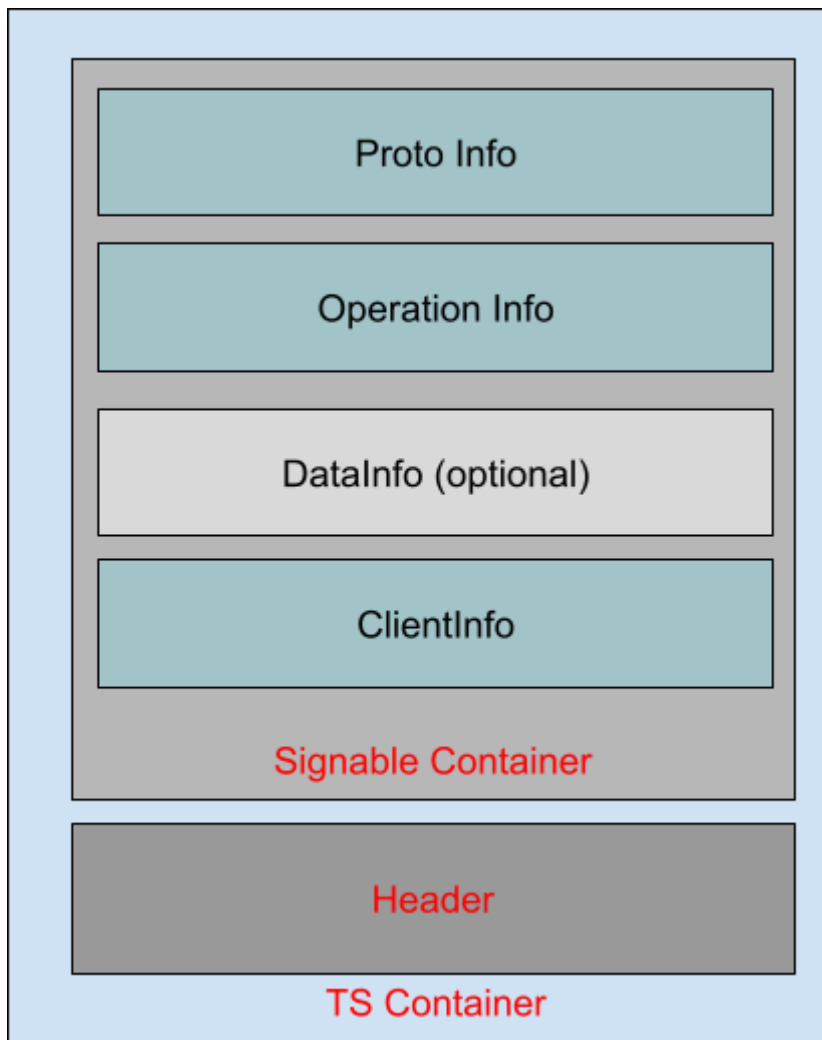
The Client system has not direct integration to the IDP(Identity Provider system). Only data exchange happening between end user app and the client system.

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



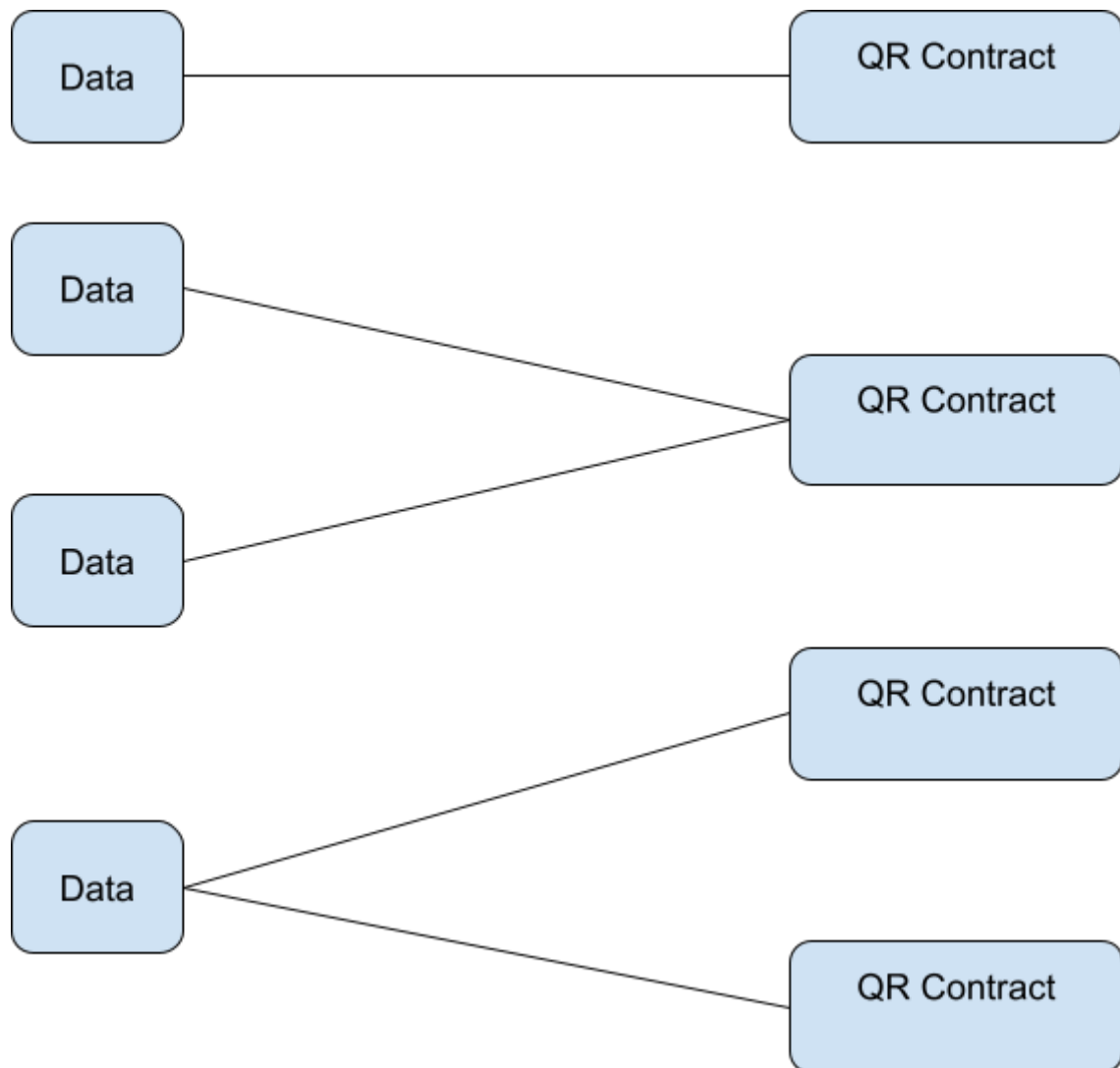
TsContainer (Contract) Version (1.1) - The main data container which contains all information, in other words the contract itself. Ts

Object Name		Description	Value
SignableContainer	ProtoInfo	Name	Protocol name.
		Version	The version of the protocol .
	OperationInfo	Type	Type of the operation. The reason the contract is generated.
		OperationId	Transaction ID for the contract. Any string data.
		NbfUTC	Not before or activation date as UNIX UTC timestamp.
		ExpUTC	Not after or expiration date as UNIX UTC timestamp.
		Assignee	People that expected to sign / to authenticate.
	DataInfo (Optional)	DataURI	Data behind the contract
		AlgName (Optional)	The name of Hash Algorithm
		FingerPrint (Optional)	Checksum of the data behind. base64 format
	ClientInfo	ClientId	Client ID that is provided by IDP
		ClientName	The name of the resource Application
		IconURI	Public Icon URL. (svg,Png,Jpeg ..etc)
		Callback	Public callback URL of the system
		RedirectURI Optional	Redirect Url for app to redirect
		HostName	List of allowed hosts if exists []
Header	AlgName		Signature Algorithm name for the contract. Mainly it uses MAC
	Signature		<p>Base64 encoded signature.</p> <p>H() - hash function (by default SHA256) K - Secret key(master Key) S - Signature CH - checksum with H()</p> <p><i>Signature creation process:</i></p> <p>CH=H(SignableContainer)</p> <p>S=HMAC(CH , K) => EncodeBase64(S)</p>

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": "1.3"
    },
    "OperationInfo": {
      "Type": "Auth",
      "OperationId": "123456789",
      "NbfUTC": 1649721600,
      "ExpUTC": 1650326400,
      "Assignee": []
    },
    "ClientInfo": {
      "ClientId": 1,
      "ClientName": "ScanMe APP",
      "IconURI": "Icon Public URL",
      "Callback": "callbackURL",
      "RedirectURI": "test.az/tran/123456789"
    }
  },
  "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": "1.3"
    },
    "OperationInfo": {
      "Type": "Sign",
      "OperationId": "123456789",
      "NbfUTC": 1649721600,
      "ExpUTC": 1650326400,
      "Assignee": ["XXXXXXX", "YYYYYYYY", "ZZZZZZ"]
    },
    "ClientInfo": {
      "ClientId": 1,
      "ClientName": "ScanMe APP",
      "IconURI": "Icon Public URL",
      "Callback": "callbackURL",
      "RedirectURI": "test.az/tran/123456789"
    }
  },
  "Header": {
    "AlgName": "HMACSHA256",
    "Signature": "nQxNMasxoL1WuLJ2x1kRhFamwFTbDxyjMGnF1Tycyr0="
  }
}
```

Example contract with **DataInfo**:

```
{
  "SignableContainer": {
    "ProtoInfo": {
      "Name": "web2app",
      "Version": "1.3"
    },
    "OperationInfo": {
      "Type": "Sign",
      "OperationId": "123456789",
      "NbUTC": 1649721600,
      "ExpUTC": 1650326400,
      "Assignee": ["XXXXXXX", "YYYYYYY", "ZZZZZZ"]
    },
    "DataInfo": {
      "DataURI": "https://scanme.sima.az/home/getdata"
    },
    "ClientInfo": {
      "ClientId": 1,
      "ClientName": "ScanMe APP",
      "IconURI": "Icon Public URL",
      "Callback": "callbackURL",
      "RedirectURI": "test.az/tran/123456789"
    }
  },
  "Header": {
    "AlgName": "HMACSHA256",
    "Signature": "nQxNMasxoL1WuLJ2x1kRhFamwFTbDxyjMGnF1Tycyr0="
  }
}
```

4.3. Contract representation

Scenario 1 :

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

- First, generate 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 :

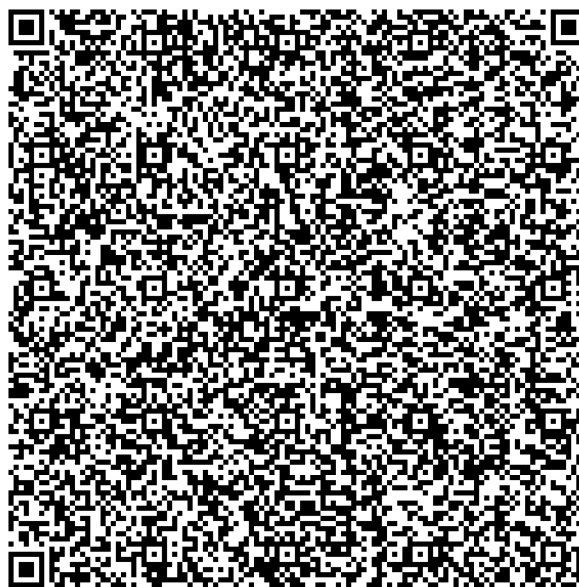


Image 1 .

Scenario 2 : with DataInfo

- First, generate the contract (**TsContainer**) with DataInfo field
- **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. OR represent it as button behind the link

1. Option convert to QR as Image.1

2. Convert to Button with link bellow :

sima://web-to-app?data=<https://example.com?tsquery=ZBwkVzQT0ifX0>

...

Or

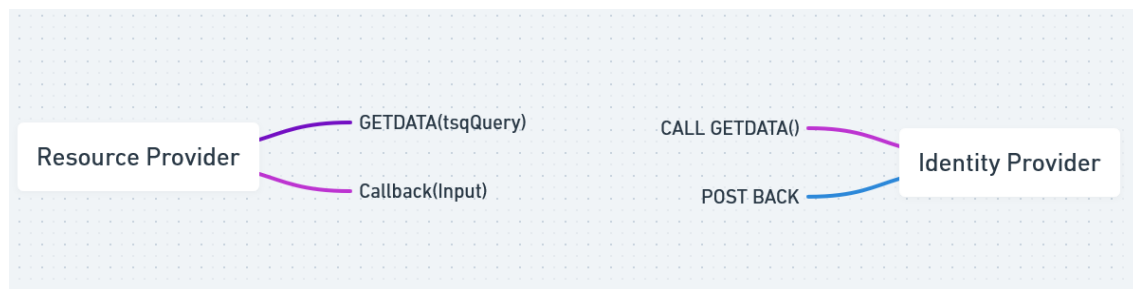
sima://web-to-app?tsquery={base64encoded-tsquery}

NOTE: This option needs if the identity app and the resource app are in the same device .
For example, mobile browser and mobile app in the same phone .

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
5lciI6eyJQcm90b0luZm8iOOnsiTmFtZSI6IndlYjJhcHAiLCJWZXJzaW9uIjoimS4wI
n0sIk9wZXJhdGlvbkluZm8iOOnsiVHlwZSI6IkF1dGgiLCJpcGVyYXRpb25JZCI6IjEy
MjMxMjMxMjMiLCJOYmZVVEMiOjE2NDkyMDMyMDAsIkV4cFVUQyI6MTY1MDU4NTYwMCw
iQXNzaWduZWUiO1tdfSwiQ2xpZW50SW5mbyI6eyJDbGllbnRjZCI6MSwiSWNvb1VSSS
I6Imh0dHBzOi8vZmlsZXMuZm8iOOnsiZG4uY29tL3YxL2ZpbGVzLzQ0ODYwMjYxL2Nvb
nRlbnQuc3ZnP3NpZ25hdHVyZT1LMUFTUHdlUGlyU1dGQXVYS3IwU0xJb0dLNzAiLCJD
YWxsYmFjayI6Imh0dHBzOi8vc2Nhbm1lLnNpbWwEuYXovSG9tZS9jYWxsYmFjayJ9fSw
iSGVhZGVyIjp7IkFsZ05hbWUiOiJITUFDU0hBMjU2IiwuU2lnbmF0dXJlIjoiemtqTE
1MeUt4dWlSMkNNbHVrdVpuR21rS2lKbjJvY2w0ZDI4aGZBWKvZQT0ifX0=
  
```



So from this Contract identity provider gets:

Domain = <https://scanme.sima.az>

Host = scanme.sima.az

Encodedcontract = tsquery =

```
eyJTaWduYWJsZUNvbhKhaw51ciI6eyJQcm90b0luZm8iOnsiMmFtZSI6IndlYjJhcHAiLC
JWZXJzaW9uIjoMS4wIn0sIk9wZXJhdGlvbkluZm8iOnsiVHlwZSI6IkF1dGgiLCJpcGVy
YXRpb25JZCI6IjEyMjMxMjMxMjMxMjMxMjMxMjMxMjMxMjMxMjMxMjMxMjMxMjMxMjMx
Y1MDU0NTYwMCI6IjEwMjMxMjMxMjMxMjMxMjMxMjMxMjMxMjMxMjMxMjMxMjMxMjMxMjMx
SWNvb1VSSSI6Imh0dHBzOi8vZm1sZXNubG9nb3NjZG4uY29tL3YxL2ZpbGVzLzQ0ODYwMj
YxL2Nvb3RlbnQuc3ZnP3NpZ25hdHVyZT1LMUFTUhd1UGlyU1dGQXVYS3IwU0xjb0dLNzAi
LCJCYWxsYmFjayI6Imh0dHBzOi8vc2Nhbm1lLnNpbWUeYXovSG9tZS9jYXxsYmFjayJ9fS
wiSGVhZGVyIjp7IkFsZ05hbWUiOiJITUFDU0hBMjU2IiwuU2lnbmF0dXJlIjoiemtqTE1M
eUt4dW1SMkNNbHVrdVpuR21rS2lkBjJvY2w0ZDI4aGZBwKvZQT0ifX0=
```

Additionally, in **web2app** protocol version **1.1** also supports a new approach for getting data . If **DataInfo** field exists, it has to contain **DataURI** field from where we can get the data behind the contract.

```
Tscontainer = decode(Encodedcontract)
```

```
{
  "SignableContainer": {
    "ProtoInfo": {
      "Name": "web2app",
      "Version": "1.3"
    },
    "OperationInfo": {
      "Type": "Auth",
      "OperationId": "1223123123",
      "NbUTC": 1649203200,
      "ExpUTC": 1650585600,
      "Assignee": []
    },
    "ClientInfo": {
      "ClientId": 1,
      "ClientName": "Sima Application" ,
      "IconURI":
        "https://files.logoscdn.com/v1/files/44860261/content.svg?signature=K1AmPwePirSWFAuXKr0SLIoGK70",
      "Callback": "https://scanme.sima.az/Home/callback",
      "RedirectURI": "test.az/tran/123456789"
    }
  },
  "Header": {
```

```

    "AlgName": "HMACSHA256",
    "Signature": "zkjLMLyKxuiR2CMlukuZnGmkKiJn2oc14d28hfAZEsa="
  }
}

```

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=eyJTaWduYWJsZUNvbnRhaW51ciI6eyJQcm90b0luZm8iOnsiTmFtZSI6IndlYjJhcHAiLCJWZXJzaW9uIjojMS4wIn0sIk9wZXJhdGlvbkluZm8iOnsiVHlwZSI6IkF1dGgiLCJPcGVyYXRpb25JZCI6IjEyMzQ1Njc4OSIsIk5iZlVUQyI6MTY0OTcyMTYwMCwiRXhwVVRDIjoxNjUwMzI2NDALCjBc3NpZ25lZSI6W119LCJDbG1bnRjbmZvIjp7IkNsaWVudElkIjoxLCJjY29uVGVJJIjoiaHR0cHM6Ly93d3cuaw5zdGFncmFtLmNvbS9zdGF0aWVhZ2VzL3dlYi9tb2JpbGVfbmF2X3R5cGVfbG9nby0yeC5wbmcvMWI0N2Y5ZDBlNTk1LnBuZyIsIkNhbgxiYWNRiJoiaHR0cHM6Ly9zY2FubWUuc2ltYS5hei9Ib211L2NhbgxiYWNRIn19LCJIZWFKZXIiOnsiQWxnTmFtZSI6IkhnQUJNTSEyNTYiLCJTaWduYXR1cmUiOiJuUXhOTWFzeG9MMVd1TEoyeDErUmhGOW13R1RiRHh5ak1HbkYxVH1jeXIwPSJ9fQ==>

ts-cert:MIIEDjCCA2+gAwIBAgIOH9GmwMxv9DcAAAAANMx0wCgYIKoZIzj0EAwMwFDELMAKGA1UEBhMCQVoxOzA5BgNVBAoTMk5hdGlvbmFsIENlcnRpZmljYXRlIFNlcnZpY2VzIENlbnRlcjBvZiBBemVyYmFpamFuMTAwLgYDVQQDEydBemVyYmFpamFuIE5hdGlvbmFsIElzc3VpbmcgTW9iaWx1IENBMDEwHhcNMjIwMDY0MTIyWhcNMjUwMzI2NDALCjBc3NpZ25lZSI6W119LCJDbG1bnRjbmZvIjp7IkNsaWVudElkIjoxLCJjY29uVGVJJIjoiaHR0cHM6Ly93d3cuaw5zdGFncmFtLmNvbS9zdGF0aWVhZ2VzL3dlYi9tb2JpbGVfbmF2X3R5cGVfbG9nby0yeC5wbmcvMWI0N2Y5ZDBlNTk1LnBuZyIsIkNhbgxiYWNRiJoiaHR0cHM6Ly9zY2FubWUuc2ltYS5hei9Ib211L2NhbgxiYWNRIn19LCJIZWFKZXIiOnsiQWxnTmFtZSI6IkhnQUJNTSEyNTYiLCJTaWduYXR1cmUiOiJuUXhOTWFzeG9MMVd1TEoyeDErUmhGOW13R1RiRHh5ak1HbkYxVH1jeXIwPSJ9fQ==

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": "challenge",
  "data": "YjY1MmEyYTAtYmQ4NS00Njg3LThjYWMTYWMTOTk0ODYzMWMx"
}
```

Example 2 : With DataInfo

Contract behind the QR is :

<https://scanme.sima.az/Home/GetFile/?tsquery=eyJTaWduYWJsZUNvbnRhaW5lcjI6eyJQcm90b0luZm8iOnsiTmFtZSI6IndlYyJhcHAiLCJWZXJzaW9uIjoIMS4wIn0sIk9wZXJhdGlvbklucyM8iOansiVHlwZSI6Imk6LmdGgiLCJPcGVyXHRpb25JCiI6IjEyMjMxMjMxMjMiLCJOYmZWVEMiojeE2NDkyMDMyMDAsIkV4cFUUcyI6MTY1MDU4NTYwMCwiOXNzaWduZWUiOi1tdfSwiO2xpZW50SW5mbyi6eyJDbbGllbnRJZCI6MSwiSwnbblVSSSI6Imh0dHBzoi8vZmlsZXMuMG9nb3NjaZG4uY29tLTZhbmZzcGU...>

```
Encodedcontract = tsquery =  
eyJTaWduYWJsZUNvbnRhaW5lciE6eyJQcm90b0luZm8iOnsiTmFtZSI6IndlYjJhcHAiLCJWZX  
Jzaw9uIjoimS4wIn0sIk9wZXJhdGlvbklwZm8iOnsiVHlwZSI6IkF1dGgiLCJPcGVyYXRpb25J  
ZCI6IjEyMjMxMjMxMjMiLCJOYmZVVEMiOjE2NDkyMDMyMDAsIkV4cFVUQyI6MTY1MDU4NTYwMC  
wiQXNzaWduZWUiOltdfSw...
```

```
Tscontainer = decode(Encodedcontract)
```

```
{
  "SignableContainer": {
    "ProtoInfo": {
      "Name": "web2app",
      "Version": "1.1"
    },
    "OperationInfo": {
      "Type": "Auth",
      "OperationId": "1223123123",
      "NbfUTC": 1649203200,
      "ExpUTC": 1650585600,
      "Assignee": []
    },
    "DataInfo": {
      "DataURI": "https://scanme.sima.az/home/getdata"
    },
    "ClientInfo": {
```

```

    "ClientId": 1,
    "ClientName": "Sima Application" ,
    "IconURI":
    "https://files.logoscdn.com/v1/files/44860261/content.svg?signature=K1AmPwePirSWFAu
    XKr0SLIoGK70",
    "Callback": "https://scanme.sima.az/Home/callback",
    "RedirectURI": "test.az/tran/123456789"
  }
},
"Header": {
  "AlgName": "HMACSHA256",
  "Signature": "zkjLMLyKxuiR2CMLukuZnGmkKiJn2oc14d28hfAZEsa="
}
}

```

DataSource= <https://scanme.sima.az/home/getdata>
 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/getdata
ts-cert:MIIEJjCCA2+gAwIBAgIOH9GmwMxv9DcAAAANMx0wCgYIKoZIZj0EAWMwfDEL
MAKGA1UEBhMCQVoxOzA5BgNVBAoTMk5hdGlvbmFsIENlcnRpZmljYXRlIFNlcnZpY2Vz
IENlbnRlciBvZiBBemVyYmFpamFuMTAwLgYDVQQDEydBemVyYmFpamFuIE5hdGlvbmFs
IElzc3VpbmcgTW9iaWx1IENBMDEwHhcNMjIwMzE0MDY0MTIyWhcNMjIwMzE0MDY0MTIy
WjB5MQswCQYDVQQGEwJBWjEuMCwGA1UEAwlRsaPUSwRCDEsFNNQV1JTFpBRMaPIMSsw
U1JBRsSwTCBPxJ5MVTEWMBQGA1UEBAwNxBTTFUZFZSUxaQUTGjzEQMA4GA1UEKgwHRsaP
UsSwRDEQMA4GA1UEBRMHNTZNS0ZSWTBZMBMGBYqGSM49AgEGCCqGSM49AwEHA0IABJcn
IQ1vQGxkpDL1a7NAnoKUj1hlihQMudOmsReFC1vtXxpnTilOJi3S518zkDCLw+zfSAa2
ttWwsa4ouy68+RijggHwMIIB0jAOBgNVHQ8BAf8EBAMCBsAwHQYDVR00OBBYEFLnwcNxU
/4Lohi2gCDuf3pTZqRqMB8GA1UdIwQYMBaAFI2LbEK25/guXeeFws5oS9s7fjSAMD0G
A1UdHwQ2MDQwMqAwoC6GLGh0dHA6Ly9tb2JpbGUuZS1pbXphLmF6L2NkcGFpYS9Bwk5J
TUNBMDEuY3JSMHIGCCsGAQUFBwEBBGYwZDA4BggrBgEFBQcwAoYsaHR0cDovL21vYm1s
ZS51LlWltemEuYXovY2RwYWIhL0FaTklNQ0EwMS5jcjcnQwKAYIKwYBBQUHMAGGHGh0dHA6
Ly9tb2JpbGUuZS1pbXphLmF6L29jc3AwPQYJKwYBBAGCNxUHBDALgYmKwYBBAGCNxUI
gq3GFIGzhxG5kw2G5/1Uge2pDIFhhrbOb4GAi2UCAWQCAQkwHwYDVR01BBgwFgYIKwYB
BQUHAWQGCisGAQQBgjcKAwwwKQYJKwYBBAGCNxUKBBwwGjAKBggrBgEFBQcDBDAMBgor
BgEEAYI3CgMMMEIGA1UdIAQ7MDkwNwYKKwYBBAGCgEsHATApMCcGCCsGAQUFBwIBFhto
dHRwc2ovL21vYm1sZS5hei9yZXBvc210cnkwCgYIKoZIZj0EAWMDgYwAMIGIAkIBZH/f
NQEcV8YV89pdIq9OVdyLqP7Yxw9myb3rRiAhvP2adRXoLGvcKoPnSv1+fdYw3YFaFNLv
A1BkXBJAh61Yn74CQgD+rH+iKmJ8KWMizRiXyVf1EtEd3LoB/D3IUETaAgTnx8h1Ud7a
ZBCsrn5ZIXHzv545hgp4kaoIqsM0bV6Ni807Ig==
ts-sign-alg:ECDSA_SHA256
ts-sign:MEUCIF9scLvsojRhBzxwRHAIXcALrkoIyTHBfbcF7x5uVwLAiEA1xhAyNKA
gbUHOKL2c7QAx7sLo25i90vmnS3ywm8i4Vo=

```

The service provider should return the http result as following :

```
HTTP/1.1 200 OK
{
  "filename": "challenge",
  "data": "YjY1MmEyYtAtYmQ4NS00Njg3LThjYWMTYWMxOTk0ODYzMWMx"
}
```

All other processes are the same with proto web2app 1.0

The same scenario works with sign format

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
x0zASBgNVBAoTMk5hdGlvbmFsIEN1cnRpZmljYXRlIFN1cnZpY2VzIEN1bnRlciBvZiBBemVyYmFpamFuMT
AwLgYDVQDEydBemVyYmFpamFuIE5hdGlvbmFsIElzc3VpbmcgTW9iaWx1IENBMDEwHhcnMjIwMzE0MDY0M
TIyWWhcnMjUwMzEzMDY0MTIyWjB5MQswCQYDVQGEwJBWjEuMCwGA1UEAwW1RsaPUSwRCDEsFNNQV1JTFpB
RMApIMSwU1JBRsSwTCBPxJ5MVTEWMBQGA1UEBAWNBXLTBTUUFZSUxaQUTGjzEQMA4GA1UEKgwHRsaPUSwRDE
QMA4GA1UEBRMHNTZNS0ZSWTBZMBMGBYqGSM49AgEGCCqGSM49AwEHA0IABJcnIQlvQGxkpDL1a7NAnoKUj1
hlihQMudOmsReFC1vtXpnTilOJi3S518zkDCLw+zfSAa2ttWwsa4ouy68+RijggHwMIIB0jA0BgNVHQ8BA
f8EBAMCBsAwHQYDVDR0BBYEFlnwcNwU/4Lohi2gCDuf3pTZqaRqMB8GA1UdIwQYMBaAFI2LbEK25/guXeeF
Ws5oS9s7fjSAMD0GA1UdHwQ2MDQwMqAwoC6GLGh0dHA6Ly9tb2JpbGUuZS1pbXphLmF6L2NkcGFpYS9Bwk5
JTUNBMDEuY3JsmHIGCCsGAQUFBwEBBGYwZDA4BggrBgEFBQcwAoYsaHR0cDovL21vYm1sZS51LWltemEuYX
ovY2RwYWlhL0FaTk1NQ0EwMS5jcnQwKAYIKwYBBQUHMAIGH0dHA6Ly9tb2JpbGUuZS1pbXphLmF6L29jc
3AwPQYJKwYBBAGCNxUHBDawLgYmKwYBBAGCNxUIgq3GFIGzhxG5kw2G5/1Uge2pDIFhhrbOb4GAi2UCAWQC
AQkwHwYDVDR01BBGwFgYIKwYBBQUHAWQGcisGAQQBgjckAwwwKQYJKwYBBAGCNxUKBBwwGjAKBggrBgEFBQc
DBDAMBgorBgEEAYI3CgMMIEIGA1UdIAQ7MDkwnWYKKwYBBAGCgEsHATApMCCGCCsGAQUFBwIBFhtodHRwc
ovL21vYm1sZS5hei9yZXBvc2l0cnkwCgYIKoZIZj0EAWMDgYwAMIGIAKIBZH/fNQEcV8YV89pdIq9OVdyLq
P7Yxw9myb3rRiAhvP2adRXoLGvcKoPnSv1+fdYw3YFaFNLvA1BkXBJAh61Yn74CQgD+rH+ikMJ8KwMIzRiX
yVf1EtEd3LoB/D3IUETaAgTnx8h1Ud7aZBCsrn5ZIXHzv545hgp4kaoIqsM0bV6Ni807Ig==
ts-sign-alg:ECDSA_SHA256
ts-sign:MEUCIGZCLjtrJntX4PLE2sToyXz2fo6oLPJj0MoBaJwSrQ4BAiEAogZRicKMYzmnF2aHqL/L0IW
OrVxAwmh/CJoT/ES1JuM=

{
  "Type": "Auth",
  "OperationId": "123456789",
  "DataSignature":
    "MEUCIQCN+6Di6lGyJlMTji0whBZQHxtyOb6e6IE4c6RnsguggIgNt0pufpX1E12vTrtGPIqPoiB96qA50
    1SA7JwTI7HMCm=",
  "SignedDataHash": "uwQ7frHv201Bn3/iaSu+yXgdM+66i5FmJJoqJiubN9k=",
  "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

NOTE: If you are using protocol version 1.3 you can use the field **RedirectURI**. The app will redirect to that URI after sing complete action

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	<p>Signature of the request</p> <p>If the http Method = POST</p> <p>Signature = sign(request.Body,ts-sign-alg , privKey)</p> <p>Method = GET sign(qrcontract - domain,ts-sign-alg , privKey)</p>