API Reference

The OSIA IDUsage extended authentication

API Version: 1.0.0

The OSIA IDUsage extended authentication
Change log:

- 1.0.0: Initial version

INDEX

Security and Authentication

SECURITY SCHEMES

KEY	TYPE	DESCRIPTION
BearerAuth	http, bearer, JWT	

API

1. AUTHENTICATE

1.1 POST /authenticate

Authenticate a person using one or more authentication factors

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
*transactionId	string	The id of the transaction

REQUEST BODY - application/json

```
{
  context* {
    personId*     string  min:1 chars
                          The identity of the person to authenticate
    dateTime*     string  12 to 30 chars
                          Timestamp in ISO8601
    purpose       string  0 to 256 chars
                          Purpose of the authentication request. Not mandatory but provides a way to ensure people know
                          for what they are doing it
    issuer        string  0 to 250 chars
                          Issuer of the identity. Its optional. Authentication can be rejected in some scenarios if this is not
                          present
    type          string  0 to 150 chars
                          Type of the ID. Its optional and auth can not be rejected if the type is not present.
    environment   string  min:1 chars
    domainUri     string  min:1 chars
  }
  consent* {
    type*     enum    ALLOWED:LINKED, EMBEDDED, NO_CONSENT
                      Consent type. Limited the enum values
    data      string  1 to 256 chars
                      embedded data in jwt format
    schema    string  1 to 256 chars
                      Schema for the consent
    signUri   string  1 to 256 chars
                      signature url. no data of the consent
    linkUri   string  1 to 256 chars
                      unique link to the content of the consent
  }
  authenticationFactors* [{
  Array of object:
    factor*                     string                  1 to 256 chars
                                                        Name of the factor
    data*
      ONE OF
      0 {
        ::props {
          attributeName* string
          operator*      enum    ALLOWED:<, >, =, >=, <=
          value*
```

ONE OF
prop0
string
prop1
integer
prop2
number
prop3
boolean
}
}
prop0
string
OPTION 1 {
biometricType* enum ALLOWED:FACE, FINGER, IRIS, SIGNATURE, UNKNOWN
biometricSubType enum ALLOWED:UNKNOWN, RIGHT_THUMB, RIGHT_INDEX, RIGHT_MIDDLE, RIGHT_RING, RIGHT_LITTLE, LEFT_THUMB, LEFT_INDEX, LEFT_MIDDLE, LEFT_RING, LEFT_LITTLE, PLAIN_RIGHT_FOUR_FINGERS, PLAIN_LEFT_FOUR_FINGERS, PLAIN_THUMBS, UNKNOWN_PALM, RIGHT_FULL_PALM, RIGHT_WRITERS_PALM, LEFT_FULL_PALM, LEFT_WRITERS_PALM, RIGHT_LOWER_PALM, RIGHT_UPPER_PALM, LEFT_LOWER_PALM, LEFT_UPPER_PALM, RIGHT_OTHER, LEFT_OTHER, RIGHT_INTERDIGITAL, RIGHT_THENAR, LEFT_INTERDIGITAL, LEFT_THENAR, LEFT_HYPOTHENAR, RIGHT_INDEX_AND_MIDDLE, RIGHT_MIDDLE_AND_RING, RIGHT_RING_AND_LITTLE, LEFT_INDEX_AND_MIDDLE, LEFT_MIDDLE_AND_RING, LEFT_RING_AND_LITTLE, RIGHT_INDEX_AND_LEFT_INDEX, RIGHT_INDEX_AND_MIDDLE_AND_RING, RIGHT_MIDDLE_AND_RING_AND_LITTLE, LEFT_INDEX_AND_MIDDLE_AND_RING, LEFT_MIDDLE_AND_RING_AND_LITTLE, EYE_UNDEF, EYE_RIGHT, EYE_LEFT, PORTRAIT, LEFT_PROFILE, RIGHT_PROFILE
instance string Used to separate two distincts biometric items of the same type and subtype
image string Base64-encoded image
imageRef string URI to an image
captureDate string
captureDevice
ONE OF
prop0
string
OPTION 1 {
serialNumber* string 0 to 36 chars
make* string 1 to 50 chars
Make of the device
model* string 1 to 50 chars
Model of the device
type* enum ALLOWED:FNGER, IRIS, FACE
Type of the device
deviceSubType enum ALLOWED:SLAP, SINGLE, TOUCHLESS, DOUBLE, FULL_FRONTAL
Additional details of device type
deviceProvider string 1 to 256 chars

Provider name as per the certifcation

deviceProviderId string 1 to 50 chars

Unique provider id assigned by the certifier

dateTime* string min:1 chars

ISO Timestamp of the device

}

width integer the width of the image

height integer the height of the image

bitdepth integer

resolution integer the image resolution (in DPI)

compression enum ALLOWED:NONE, WSQ, JPEG, JPEG2000, PNG

metadata string An optional string used to convey information vendor-specific

comment string A comment about the biometric data

}

OPTION 2 {

}

hash string 1 to 512 chars

sha256 in hex format in upper case (previous hash + sha256 hash of the current biometric ISO data before encryption)

sessionKey string min:1 chars

Session key used for encrypting bioValue, encrypted with the system public key (dynamically selected based on the URI) and base64 encoded

thumbprint string min:1 chars

SHA256 representation of the certificate (HEX encoded) that was used for encryption of session key. All texts to be treated as uppercase without any spaces or hyphens

algorithm string

}]

}

HEADER PARAMETERS

NAME	TYPE	DESCRIPTION
signature	string	JWS signature of the entire http body of the body header..signature

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

{

version* string Version of this response

responseDateTime string ISO 8601

purpose* string Same value as purpose in the request

factorsVerified* [string]

consentVerified* boolean Consent verification was performed or not.

authenticationResult* {

Result of the authentication

verified* boolean result of the authentication. A simple boolean

tokenId* string 12 to 500 chars

Random Token either specific for this authentication or specific for an individual & Relying party combination.

kycCredential {

Credential

issuer* string 1 to 256 chars

Who has issued this credential.

id* string 1 to 256 chars
ID of the credential

issuedTo* string 1 to 256 chars
to whom the id is issued to

issuanceDate* string min:1 chars

protectedAttributes* [string]

credentialSubject* {
Subject information for the credential

id* string 10 to 50 chars
id of subject

idOf* [{
Array of object:

attributeName string 1 to 256 chars
Name of the atribute

value string 0 to 1024 chars
Value for the given atribute. Note: It could be encrypted if the atribute name is part of the protected array.

language string 2 to 3 chars
language in ISO 639-1 format. USe 639-2 only when the code is not available in 639-1

}]

}

type* [string] 1 to 50 chars

consent* {

type* enum ALLOWED:LINKED, EMBEDDED, NO_CONSENT
Consent type. Limited the enum values

data string 1 to 256 chars
embedded data in jwt format

schema string 1 to 256 chars
Schema for the consent

signUri string 1 to 256 chars
signature url. no data of the consent

linkUri string 1 to 256 chars
unique link to the content of the consent

}

proof* {
Credential proof

type* enum 1 to 10 chars
ALLOWED:RsaSignature2018, ED25519
Type of the proof

created* string What was it created

proofPurpose* string 0 to 256 chars
What is the purpose of this proof

verificationMethod* string 1 to 256 chars
URL to the keys

jws* string 0 to 10000 chars
JWS of the format header..signature

}

}

kycUri string Link pointing to the KYC data credential

}

errors [{
Array of object:

code string unique error code

message string description of the error message

language string ISO 639-2

action string action that we want the user to make in order to correct the error.

}]

}

STATUS CODE - 400: Bad Request, Validation Errors, ...

RESPONSE MODEL - application/json

```
{
  code*    integer
  message* string
}
```

STATUS CODE - 401: Unauthorized

STATUS CODE - 403: Operation not allowed

STATUS CODE - 500: Internal server error

RESPONSE MODEL - application/json

```
{
  code*    integer
  message* string
}
```
