Trusted Identity

Trusted Identity is a signed JWS token (in JWT format) that the **Partner/Tenant backend** passes to the Verestro backend to authenticate the request and initiate an end-user session. The token is verified cryptographically (signature + certificate chain in `x5c`) and organizationally (verification of `CN` according to the onboarding configuration).

---

1) Purpose and Use

What it is

- Trusted Identity is a token that **must be generated on the Partner/Tenant backend side**, in an isolated and secure environment. It is passed to Verestro as proof that the Partner/Tenant confirms the identity of the end user identified by `userId` (see the Payload (claims) section).

Why it is used

- To start an end-user session in a component (e.g., SDK) communicating with the Verestro backend.
- To simplify integration: the token is self-contained because it includes `x5c` (certificate chain), so Verestro can verify the signature and trust without querying an external JWKS endpoint at runtime.
- To protect against replay attacks: short TTL based on `iat`.

Token Issuance and Verification – sample usage

@startuml skinparam ParticipantPadding 30 skinparam BoxPadding 30 skinparam noteFontColor #FFFFFF skinparam noteBackgroundColor #1C1E3F skinparam noteBorderColor #1C1E3F skinparam noteBorderThickness 1 skinparam sequence { ArrowColor #1C1E3F ArrowFontColor #1C1E3F ActorBorderColor #1C1E3F ActorBackgroundColor #FFFFFF ActorFontStyle bold ParticipantBorderColor #1C1E3F ParticipantBackgroundColor #1C1E3F ParticipantFontColor #FFFFFF ParticipantFontStyle bold LifeLineBackgroundColor #1C1E3F LifeLineBorderColor #1C1E3F } actor User participant "Partner/Tenant (Backend)" as A participant "Verestro (API)" as B User -> A: Session start request A -> A: Build payload (userId, iat, jti) A -> A: Add x5c (leaf +

intermediates) A -> A: Sign JWS (RS256, private key) A -> B: Send Trusted Identity (JWS) B -> B: Read Root CA + expected CN B -> B: Validate x5c chain to Root CA B -> B: Validate CN/DN === expected B -> B: Verify JWS signature B -> B: Check iat TTL (10 min) B -> B: Start user session for userId B --> User: Session active @enduml

2) Integration / Onboarding

Variant A (recommended): Partner/Tenant provides CSR

The Partner/Tenant:

- generates an RSA key pair (RSA 2048),
- generates a CSR with `CN=V-TenantName-ApplicationName` (instructions for correctly generating a CSR are available in the documentation),
- submits the CSR to Verestro.

Verestro:

- signs the CSR with its internal CA and delivers the signed certificate to the Partner/Tenant.

The Partner/Tenant:

- uses the private key from the generated key pair to sign Trusted Identity tokens,
- includes the leaf + intermediate certificates in `x5c` (if applicable).

Variant B: Partner/Tenant provides Root CA

The Partner/Tenant provides:

- The Root CA to be added as trusted in Verestro's systems.
- The expected `CN`/`DN` of the certificate used for signing.
- The above must be provided by the Partner/Tenant during the onboarding process.

Verestro:

- configures the Root CA in its systems,
- stores the expected `CN`/`DN` for the Partner/Tenant,
- accepts tokens whose `x5c` builds a chain to this Root CA and satisfies the CN/DN requirement.

Onboarding variants

@startuml skinparam ParticipantPadding 30 skinparam BoxPadding 30 skinparam noteFontColor #FFFFFF skinparam noteBackgroundColor #1C1E3F skinparam noteBorderColor #1C1E3F skinparam noteBorderThickness 1 skinparam sequence { ArrowColor #1C1E3F ArrowFontColor #1C1E3F ActorBorderColor #1C1E3F ActorBackgroundColor #FFFFFF ActorFontStyle bold ParticipantBorderColor #1C1E3F ParticipantBackgroundColor #1C1E3F ParticipantFontColor #FFFFFF ParticipantFontStyle bold LifeLineBackgroundColor #1C1E3F LifeLineBorderColor #1C1E3F } participant "Partner/Tenant" as C participant "Verestro" as P alt Variant A (recommended): CSR signed by our CA C -> C: Generate RSA keypair C -> P: Send CSR (CN=uniquePartner/TenantName) P -> P: Sign CSR with our CA P -> C: Return certificate (+ intermediates if any) P -> P: Configure our Root CA + expected CN/DN else Variant B: Partner/Tenant provides Root CA C -> P: Provide Root CA + expected CN/DN P -> P: Configure Root CA + expected CN/DN end @enduml

---

3) Structure

The token has the following format:
`{base64url(header)}.{base64url(payload)}.{base64url(signature)}`

3.1 Header

Required fields:

- `alg`: signature algorithm (recommended: `RS256`, RSA 2048).
- `x5c`: X.509 certificate chain in Base64(DER) format, where:
  - the first element is the `leaf` certificate (containing the public key used to verify the signature),
  - subsequent elements are intermediate certificates,
  - the root CA is typically not included (Verestro holds it in its trust store).

Example:

```
{
  "alg": "RS256",
  "typ": "JWT",
  "x5c": [
    "MIIC... (leaf)",
    "MIID... (intermediate)"
  ]
}
```

3.2 Payload (claims)

Required fields:

- `userId`: user identifier:
  - accepted values: **internal ID** (identifier from Verestro) or **external ID** (end-user identifier in the Partner/Tenant system),
  - preferred: **external ID — must be unique within the Partner/Tenant system**,
- `iat`: token issuance time (UNIX epoch in seconds).
- `jti`: unique token identifier (UUID).

Example:

```
{
  "userId": "external-987654",
  "iat": 1739191200,
  "jti": "a3f1c3d6-0d3b-4f2e-9c24-8d6b9b2a9f0d"
}
```

3.3 Signature

- The token must be signed with the Partner/Tenant's private key corresponding to the certificate provided in `x5c`.
- Minimum requirements:
  - RSA 2048
  - `RS256`

---

4) Security Requirements (TTL)

TTL (token validity)

- TTL: **default: 10 minutes** from `iat` (this value can be changed). Once the token expires, a new one must be generated.

---

5) Trust Based on `x5c` and CN/DN Validation

What Verestro verifies

1. Construction and validation of the certificate chain from `x5c` up to the configured Root CA.
2. Validation of the certificate's `CN` against the value established during the onboarding process.
3. JWS signature verification using the public key from the leaf certificate.
4. Validation of `iat` (TTL) and `jti` (replay protection), and presence of `userId`.

CN rule (integration requirement)

- The `CN` (or agreed `DN`) of the certificate is the permanent identifier of the Partner/Tenant.
- The `CN`/`DN` must not change during certificate renewal without prior notification; key/certificate rotations within the same DN are permitted.

---

7. Examples and Test Methods – those should be used for development and testing purposes only.

7.1 Trusted Identity example:

Token generated with the following payload will look like this:

```
{
  "iat": "1770981506093",
  "jti": "1b290b62-8e1e-4168-bb24-5c29d22ee699",
  "userId": "some-external-user-id-1234"
}
```


```
"eyJhbGciOiJSUzI1NiIsIng1YyI6WyJNSUlGeFRDQ0E2MmdBd0lCQWdJVWNMWmtmN0EyZFdvNHdKT1NlVEdpbnMvdlVQT
XdEUVlKS29aSWh2Y05BUUVMQlFBd2NqRUxNQWtHQTFVRUJoTUNVRXd4RkRBU0JnTlZCQWdNQzAxaGVtOTNhV1ZqYTJsbE1
ROHdEUVlEVlFRSERBWlhZWEp6WVhjeEZEQVNCZ05WQkFvTUMxWmxjbVZ6ZEhKdklGTkJNUXd3Q2dZRFZRUUxEQU5FUlZZe
EdEQVdCZ05WQkFNTUQxWmxjbVZ6ZEhKdlgwTkJYMFJsZGpBZUZ3MHlNVEF5TURnd056TTVOVEZhRncweU5qQXlNRGN3Tnp
NNU5URmFNSEl4Q3pBSkJnTlZCQVlUQWxCTU1SUXdFZ1lEVlFRSURBdE5ZWHB2ZDJsbFkydHBaVEVQTUEwR0ExVUVCd3dHV
jJGeWMyRjNNUlF3RWdZRFZRUUtEQXRXWlhKbGMzUnlieUJUUVRFTU1Bb0dBMVVFQ3d3RFJFVldNUmd3RmdZRFZRUUREQTl
XWlhKbGMzUnliMTlEUVY5RVpYWXdnZ0lpTUEwR0NTcUdTSWIzRFFFQkFRVUFBNElDRHdBd2dnSUtBb0lDQVFEbkZIZzB4Q
```

```
0prckYwNFVSc0p0em5aVndxOWxrVUd2Z3J3K2RVVzh5aS9MV0VoMC9QU2hLVmUxNEFqWktXSXJaU3FRNDR6d0Mwbk04c0N
yRjgvWnJxVFYvc0Q3dTQ5WnMzeko2TkVOQk83VTI0blZpeUxiaDNQQkZReEcyYWd6Q0ZTUXpEMTdpR0ZxckxYa29vQWVOM
VllczRUc0FSdzlzK21aMlhydzlEaGdVa3pFZUZwcXo1c3J2N3huMFpLdjh4czMrZjd1SUlGdWdSWFhYVWltUmdCWXQ3dzk
0SmgxblRpQis0SU1RSXBFR2Vmd29HTmlKL0IvWThsVXJVLzd1TnhKTHc2UU54dG52ZkFNaVJReGV4ektPc21RMWVxd1MrW
WlQTVc3MG8vOVJHMmRRaWZyYWRpOU9JRjlqZmUxbzhxSVRQUG1nRG9aVWI1dWYrSGFybXdrSEs1SjBUcWpkZWZaNjlwYTV
wWUppQzNsOUNOYlh1YzI3VVhuZmQxK2dwckdZMjdhMjgyemMwQnZad1FjL085blZIclVBeVh4OGR3MjIwK3RkUDZzSkFKM
UV6YUpaVWltcDBka0hENkhLbmp3WjJ1L2RtMi83RzFYSnpiY0IzSStDMWlmbXdEYXhoWlVUbGpweGRBbklpbFU5WUpmOU1
4UVUwRGJ6bGVxMS9Sc01DWU9rSUpoZjVwZXdPMXFmSVlLK0V2a05iRzI5VzRqdGtNOFNISTdVRUFNN3hrWk1qVkNaamNMZ
nBzUk5LMWNjTnpFTEF3V2lHWG8yaVhEVmdBUStVUVU2cHBQdFQyb0ZVWWU3aWpFSit0RnEzQWd3UzIvMldnL1RvOVlhbit
WM2luN2IxUURmTlJVc1BxeDgrTFlLcHhCRGVhMjBlR3FBZ1VTYzNEVFFJREFRQUJvMU13VVRBZEJnTlZIUTRFRmdRVWlte
VhrSDE4bGhzRXc1ai9UaldCZ0d2NDNhSXdId1lEVlIwakJCZ3dGb0FVaW15WGtIMThsaHNFdzVqL1RqV0JnR3Y0M2FJd0R
3WURWUjBUQVFIL0JBVXdBd0VCL3pBTkJna3Foa2lHOXcwQkFRc0ZBQU9DQWdFQVhlM0ZGMm1zRC9qdllyYW03SnBpQlUvb
FB0dEx0bmdpL3pCTGM4Y0dKbGlhcTM2S1h1ZWdpRWp3cjJtT1ExeXNaUXRQSnUrZjg4TXBPWU9QdDFBZm1QN1RRZ0ZZTjR
yOFJVU2RxOEMwVTlUOFZvNUJxaXh4WFl6Z0xBSi8wTkZpZmx3YVJLN2FKOUdza1MrNkFoRUppRTdyNkl1cTZScWVyS3VvY
kZvaEtQdHZLZkM5ZjBvcEwxVDBlV296L3BVNTBOTDM5QXpETkk5Tkh5MWdPQnNGNElYSHRUeHN1eTJCNjVyTkpjSjdKbmN
kc0xhdWVhL3doa1hoUWRkUTVrN25MbUJ4eFl2M0RWZmNkZ2dYUXF0S3ZKQnh4SE9Gak1yUzdDYXVud2h6WDlJQW05Ny82M
U8zNXVkd1dKVHZ0eFJnNEFWTGNOWXhjRHZTNlN3WVhud0dQak5BS0QvNXdpcStTMlZib1Z4UkJqSm12cEVhRU5WQ3djQnl
vZ3ZidnlXcEd6WkhOUGdDRmVUVFRIYU51ZWJxL05XYmJ4QWM2bWhxM3d5bC9mYnBwakRhTno1U2hlbVFCdVVhbVZ1WlBZN
VJDS2FUc2twSElTT0UyT1RrMkEzVkpuajA1cm0wQUY1QWlRbjJjWThJM3dYODlseFZRc3YrRjQyR2VMalVRMnVqMDR5dEl
iSjNnYUlBVFE3ODMvRjVuRkFVY0o0VWNGcXRFRXZEYVVkMEkrbFRtZkQzTEc0Y3E2STRheUd2Z04zS2NURW1PUU43ZlZPN
WlvRjJFSk5oK1dRTHNuQ1RaTkgxc2dOSk53ajRRSlduSzVaYThPcTNzZk9jRDVraVE0ZmoxbW9HNmhRWkxQM3J6OVZZR2l
CWTF2aWlGVG5RcWhJYjlOdzlrQ1NDaWEwPSJdfQ.eyJpYXQiOiIxNzcwOTgxNTA2MDkzIiwianRpIjoiMWIyOTBiNjItOG
UxZS00MTY4LWJiMjQtNWMyOWQyMmVlNjk5IiwidXNlcklkIjoic29tZS1leHRlcm5hbC11c2VyLWlkLTEyMzQifQ.m75hz
T1Wn4RW4wxVLdNm5wukVMtT-5J6h8PldaHfQ19WGqlx2x4wa5Ut2xKORaoMZOOkGMP1VKbOa-
_MgFWeSR6nwyiUXoa7zDh7MNnUD7C1yCk-eVQDdSoFIQIg7UBjVj0uJZYkUOmScz0MjCpeZ-hhFVQInnHK-8YNdR1A5L-
S5-d57pubz0C2GarIJu7z1d-
qytfQFGdqvMAMuu9cPwc4oIfB4VajYqsz_NWWqSrsBNGnZIhuVKsWs1KXki3B2Z1v55tbET61bvZzuc7lEK27HzYaELC_z
a4zK0OP0GlqEl0gYzbj60E2sEE23jlUbiGj1Craq4KRrGpBAMDsgX0_kOicfQDjeYJ2wIouU2svDxLtI_BBiz7yT-
opRTu0A-uDworfyWyt5IaZi9ZGOc_4--gwTWvrv1Bp9BPLZ8t6huSeJ4pjQTWMAZ_T2s2m69HWIIkIp0xDgkWBWlByEh-
3ptzY_X3tZGtt7wjRH8AJrSeOXKPCDn9nZKu_sfaNFaDJGb28FajHR7u5xXUgif4DK9yFisaZf_SaXCkcO1KYO0oZD5Vy0
gX-ZNJSVGX0yKm7wxHXBG1JPmHrvL-ieztM0PLBSR38I5CUJCHNLBlL2Cbw8YBuo-
ubU0JxpgZ0i1g25Rxv1wVgYuw8x5ckxNpTjgsm08m-m-WcDxEAk_s"
```


7.2 Test methods

- Generating Trusted Identity:

```
curl --location 'https://datacore.verestro.dev/test/generate-jws-x5c-token' \
--header 'Content-Type: application/json' \
--header 'Accept: application/json' \
```

```
--header 'Issuer-Code: YOUR_TENANT_ID' \
--header 'Application-Id: YOUR_TENANT_APP_ID' \
--header 'Authorization: Basic dGVzdDE6dGVzdDEyMw==' \
--header 'Collection: internal' \
--data '{
    "iat": "1770981506093",
    "jti": "1b290b62-8e1e-4168-bb24-5c29d22ee699",
    "userId": "some-external-user-id-1234"
}'
```

- Validating Trusted Identity:

```
curl --location 'https://datacore.verestro.dev/test/read-jws-x5c-token' \
--header 'Content-Type: application/json' \
--header 'Accept: application/json' \
--header 'Issuer-Code: YOUR_TENANT_ID' \
--header 'Application-Id: YOUR_TENANT_APP_ID' \
--header 'Authorization: Basic dGVzdDE6dGVzdDEyMw==' \
--header 'Collection: internal' \
--data '{
    "token": "eyJhbGciOiJSUzI1NiIsIng1YyI6WyJNSUlGeF.................0qYz-H_0LEK0M"
}'
```

Output:

```
{
  "verified": true,
  "token": {
    "iat": "1770981506093",
    "jti": "1b290b62-8e1e-4168-bb24-5c29d22ee699",
    "userId": "some-external-user-id-1234"
  }
}
```

Revision #10
Created 31 March 2026 10:43:30 by Artur Mazur
Updated 31 March 2026 11:14:34 by Artur Mazur