OID4VCI: Como uma Wallet obtém Credenciais Verificáveis

Integração com Inji Certify + Gov.br SSO

O fluxo completo — da autenticação do cidadão à emissão da VC.

Visão geral: 3 fases

```
  Wallet              Gov.br SSO           Inji Certify        Data Provider
    |                     |                     |                     |
    |  1. /authorize      |                     |                     |
    |  (PKCE + login)     |                     |                     |
    |-------------------->|                     |                     |
    |  code               |                     |                     |
    |<--------------------|                     |                     |
    |  2. /token          |                     |                     |
    |  (code + verifier)  |                     |                     |
    |-------------------->|                     |                     |
    |  access_token (JWT) |                     |                     |
    |<--------------------|                     |                     |
    |                                           |                     |
    |  3. POST /issuance/credential             |                     |
    |  (Bearer token + proof JWT)               |                     |
    |------------------------------------------>|  getData(cpf)       |
    |                                           |-------------------->|
    |                                           |  { dataNascimento } |
    |                                           |<--------------------|
    |  Verifiable Credential (JSON-LD, signed)  |                     |
    |<------------------------------------------|                     |
```

Discovery: `.well-known/openid-credential-issuer`

```
curl -s "https://injicertify.credenciaisverificaveis-hml.dataprev.gov.br/v1/certify/issuance\
/.well-known/openid-credential-issuer?issuer_id=MGI" | jq .
```

```
{
  "credential_issuer": "https://injicertify.credenciaisverificaveis-hml.dataprev.gov.br",
  "authorization_servers": ["https://sso.staging.acesso.gov.br"],
  "credential_endpoint": ".../issuance/credential",
  "credential_configurations_supported": {
    "ECACredential": {
      "format": "ldp_vc",
      "proof_types_supported": {
        "jwt": { "proof_signing_alg_values_supported": ["RS256","PS256","ES256"] }
      }
    }
  }
}
```

Fase 1: Autenticação (OAuth2 + PKCE)

PKCE é apenas um string aleatório e seu hash SHA-256:

```
code_verifier  = secrets.token_urlsafe(32)[:43]       # string aleatória (segredo)
code_challenge = base64url(sha256(code_verifier))      # hash enviado ao SSO
```

1) Authorize — redireciona o cidadão ao Gov.br:

```
GET https://sso.staging.acesso.gov.br/authorize?
    response_type=code&client_id=<ID>&scope=openid email profile
    &redirect_uri=http://localhost:3004/redirect
    &code_challenge=<CODE_CHALLENGE>&code_challenge_method=S256&nonce=<R>&state=<R>
```

2) Token Exchange — cidadão faz login, Gov.br redireciona com `?code=XXXX`:

```
POST https://sso.staging.acesso.gov.br/token?
    grant_type=authorization_code&code=<CODE>
    &redirect_uri=http://localhost:3004/redirect&code_verifier=<CODE_VERIFIER>
Authorization: Basic base64(client_id:client_secret)
```

Resultado: `access_token` (JWT assinado pelo Gov.br, com **CPF** no claim `sub`)

Fase 2: Wallet Proof JWT

A wallet assina um JWT com sua chave privada. Chave pública vai no header para que o Certify extraia e vincule à VC.

```
// HEADER                              // PAYLOAD
{ "typ": "openid4vci-proof+jwt",       { "iss": "<access_token.aud>",
  "alg": "RS256",                        "aud": "https://injicertify.credenciaisverificaveis-hml.dataprev.gov.br",
  "jwk": { "kty": "RSA",                "iat": 1740000000,
            "n": "<modulus b64url>",     "exp": 1740000300 }
            "e": "AQAB" } }
```

Validações do Certify no proof:

Validação	Regra
`typ`	`"openid4vci-proof+jwt"`
`alg`	∈ `{RS256, PS256, ES256}`
`jwk` / `kid`	Exatamente um presente
Assinatura	Verificada com a chave pública do header
`aud`	= `https://injicertify.credenciaisverificaveis-hml.dataprev.gov.br`
`iat` / `exp`	Token não expirado

Resultado: chave pública → `did:jwk:<base64url(jwk)>` → `credentialSubject.id` **(holder binding)**

Fase 3: Credential Request + Resposta

```
POST /v1/certify/issuance/credential
Authorization: Bearer <access_token>
Content-Type: application/json
```

```
{ "format": "ldp_vc", "issuerId": "MGI", "doctype": "ECACredential",
  "credential_definition": {
    "@context": ["https://www.w3.org/2018/credentials/v1"],
    "type": ["VerifiableCredential", "ECACredential"] },
  "proof": { "proof_type": "jwt", "jwt": "eyJ0eXAiOi..." } }
```

Resposta (200 OK):

```
{ "format": "ldp_vc",
  "credential": {
    "@context": ["https://www.w3.org/2018/credentials/v1"],
    "type": ["VerifiableCredential", "ECACredential"],
    "issuer": "did:web:example.com",
    "issuanceDate": "2026-02-24T12:00:00Z",
    "credentialSubject": {
      "id": "did:jwk:eyJrdHkiOiJSU0EiLC...",
      "isOver12": true, "isOver14": true, "isOver16": true, "isOver18": true },
    "proof": { "type": "Ed25519Signature2020",
      "verificationMethod": "did:web:example.com#key-0",
      "proofValue": "z3FXQjecWufY46..." } } }
```

Validações do servidor

Access Token (assinado pelo Gov.br)

- Assinatura verificada contra JWK set do SSO (`/jwk.json`)
- `iss` = URI do authorization server
- `aud` ∈ lista de audiences permitidos
- `exp` > agora, `iat` < agora
- `sub` presente (contém o CPF)

Credential type

- `doctype` deve existir no metadata do `issuerId`

Erros mais frequentes

HTTP	Código	Causa
400	`invalid_proof`	`aud` do proof ≠ `mosip.certify.identifier`
400	`proof_header_invalid_typ`	`typ` ≠ `"openid4vci-proof+jwt"`
400	`unsupported_credential_type`	`doctype` não configurado para o issuer

Exemplo: Fases 2 + 3 (Python executável)

```
# Requer um access_token válido (obtido na Fase 1 via Gov.br SSO)
ACCESS_TOKEN="eyJ..." python3 scripts/short.py
```

```python
#!/usr/bin/env python3
"""Fases 2+3 do OID4VCI — requer ACCESS_TOKEN no ambiente."""
import os, base64, json, time, urllib.request
from cryptography.hazmat.primitives.asymmetric import rsa, padding
from cryptography.hazmat.primitives import hashes

CERTIFY_URL = os.environ.get("CERTIFY_URL",
    "https://injicertify.credenciaisverificaveis-hml.dataprev.gov.br/v1/certify")
CERTIFY_ID = os.environ.get("CERTIFY_IDENTIFIER",
    "https://injicertify.credenciaisverificaveis-hml.dataprev.gov.br")
access_token = os.environ["ACCESS_TOKEN"]
# client_id = claim "aud" do access_token
at_payload = json.loads(base64.urlsafe_b64decode(
    access_token.split(".")[1] + "=="))
client_id = at_payload["aud"]

b64url = lambda d: base64.urlsafe_b64encode(
    d if isinstance(d, bytes) else d.encode()).rstrip(b"=").decode()
int2b = lambda n: n.to_bytes((n.bit_length() + 7) // 8, "big")

# — Fase 2: Proof JWT —
key = rsa.generate_private_key(public_exponent=65537, key_size=2048)
pub = key.public_key().public_numbers()
jwk = {"kty": "RSA", "n": b64url(int2b(pub.n)), "e": b64url(int2b(pub.e))}
header = {"typ": "openid4vci-proof+jwt", "alg": "RS256", "jwk": jwk}
now = int(time.time())
payload = {"iss": client_id, "aud": CERTIFY_ID, "iat": now, "exp": now + 300}
si = f"{b64url(json.dumps(header))}.{b64url(json.dumps(payload))}"
proof_jwt = f"{si}.{b64url(key.sign(si.encode(), padding.PKCS1v15(), hashes.SHA256()))}"

# — Fase 3: Credential Request —
body = json.dumps({"format": "ldp_vc", "issuerId": "MGI", "doctype": "ECACredential",
    "credential_definition": {"@context": ["https://www.w3.org/2018/credentials/v1"],
        "type": ["VerifiableCredential", "ECACredential"]},
    "proof": {"proof_type": "jwt", "jwt": proof_jwt}})
req = urllib.request.Request(f"{CERTIFY_URL}/issuance/credential", method="POST",
    data=body.encode(), headers={"Content-Type": "application/json",
        "Authorization": f"Bearer {access_token}"})
with urllib.request.urlopen(req) as resp:
```

Referências

- **OpenID4VCI Spec:** https://openid.net/specs/openid-4-verifiable-credential-issuance-1_0.html
- **Inji Certify:** https://github.com/mosip/inji-certify
- **Digital Credential Plugins:** https://github.com/mosip/digital-credential-plugins
- **Gov.br SSO (staging):** https://sso.staging.acesso.gov.br
- **W3C VC Data Model:** https://www.w3.org/TR/vc-data-model/

Endpoint	Descrição
`GET /.well-known/openid-credential-issuer?issuer_id=MGI`	Metadata do issuer
`GET /.well-known/did.json`	DID Document do Certify
`POST /issuance/credential`	Emissão de credencial
`POST /issuance/vd12/credential`	Emissão (OID4VCI Draft 12)
`POST /issuance/vd11/credential`	Emissão (OID4VCI Draft 11)