WG: Authorization Flows

Can we standardize the several flows related to becoming authorized to acces certain resources?

- Authorization request
- Example AccessToken introspection

Authorization request

Note: Work in progress!

The process of requesting the authorization to perform operations on a set of resources. The custodian must check if all the requirements are met, therefore the response will be a-sync.

Relevant standards

- OpenIDConnect for Credential Issuance

Terms

The following information is needed:

scope The scope of the authorization request. In case of patient data, a patient identifier such as a BSN should be provided.

legal-basis A set of legal basis which can be used to grant the request. Examples: implied consent, explicit consent, legal task, emergency.

custodian Party who has the legal obligation to keep the data safe and only share under certain conditions.

requestor Party who wants to perform the operations on the data.

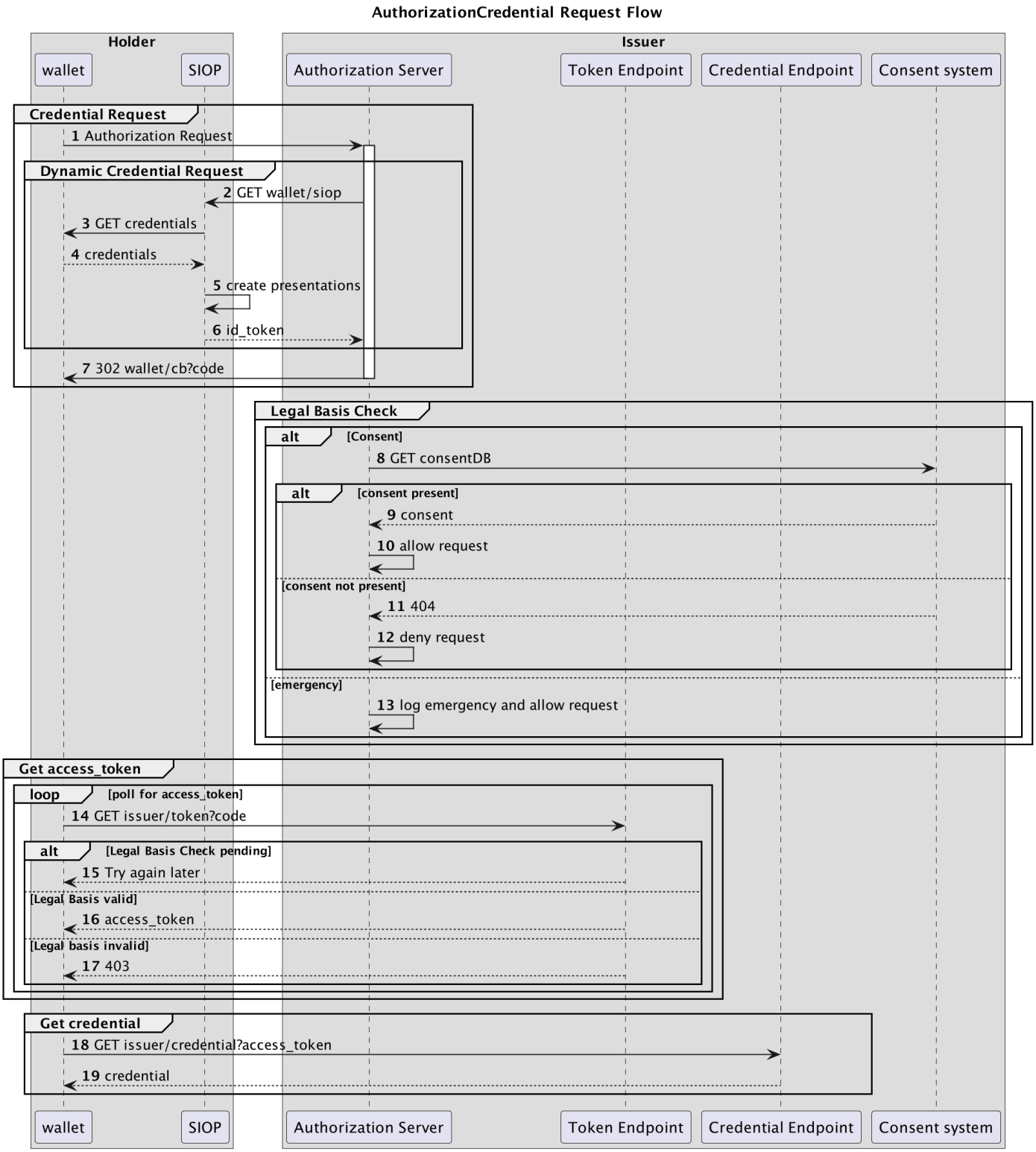
Use-case

A GP wants to read a report about a patient written by a caregiver at another care organization (custodian). The GP requests access to this report(scope). The custodian needs a patient consent in order to grant this request. It will look into its own consent system(s) to find it. If it is there, the GP receives an authorization in the form of a `AuthorizationCredential`.

Implementing using the OIDC4VCI credential request

We will base the implementation based on the new OpenID Connect standard. This way wallets and issuers will interact in a standardized way which makes this solution interoperable with other wallets and issuers.

The interactions are shown in the following diagram:



The issuer is the **Custodian** and the Holder is the **Requestor** (GP in our use-case example).

Let's walk through all the steps:

Credential Request

The goal for this group is the **Requestor** asking the **Custodian** for authorization to a specific (set of) resources. The authorization will be given in the form of a credential. This credential can later be used to acquire an `access_token` for the actual resource.

So why not directly issue an `access_token` for the resource instead? Multiple reasons:

1. The authorization request might take some time to process. An actual person might need to check the request for example. The OIDC4CI standard provides a flow for this, a normal OIDC flow doesn't.
2. The `AuthorizationCredential` is issued to the organization, but the `access_token` should be bound to a user. So, for each user session, a new `access_token` coupled to a verifiable user identity should be issued.
3. Access tokens should be short-lived, after they are issued it is no longer possible for the resource server to challenge the holder if it was the original requestor.

1. The requestor performs an AuthorizationRequest.

This is described in 5.1. Using the `authorization_details`:

```
[{
  "type":"openid_credential",
  "format": "ldp_vc",
  "credential_definition": {
    "@context": [
      "https://www.w3.org/2018/credentials/v1",
      "https://nuts.nl/credentials/v1"
    ],
    "types": [
      "VerifiableCredential",
      "NutsAuthorizationCredential"
    ],
    "credentialSubject": {
      "id": "did:nuts:123",
      "purposeOfUse": "careviewer",
      "patient":"bsn:9999992"
    }
  }
}]
```

Example request:

```
GET /authorize?
  response_type=code
  &client_id=s6BhdRkqt3
  &code_challenge=E9Melhoa2OwvFrEMTJguCHaoeK1t8URWbuGJSstw-cM
  &code_challenge_method=S256
```

```
&authorization_details=%5B%7B%22type%22%3A%22openid_credential%22%2C%22format%22%3A%22ldp_vc
%22%2C%22credential_definition%22%3A%7B%22%40context%22%3A%5B%22https%3A%2F%2Fwww.w3.org
%2F2018%2Fcredentials%2Fv1%22%2C%22https%3A%2F%2Fnuts.nl%2Fcredentials%2Fv1%22%5D%2C%22typ
es%22%3A%5B%22VerifiableCredential%22%2C%22NutsAuthorizationCredential%22%5D%2C%22credentialSu
bject%22%3A%7B%22id%22%3A%22did%3Anuts%3A123%22%2C%22purposeOfUse%22%3A%22careviewer%2
2%2C%7D%7D%7D%5D
  &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
Host: https://server.example.com
```

Dynamic Credential Request

The issuer wants to know proofs about the validity of the request. Using the dynamic credential request, as defined in section 5.

Steps 2 - 6

These steps are asking the requestor for additional proofs (VCs) about its identity and capabilities. The issuer sends a challenge and the holder embeds the VCs in a Verifiable presentation.

7. Authorization Response

The Authorization Server has received the request, creates "session" identified by a code and redirects the user-agent to the callback, providing the code in the request parameter.

Example Response:

```
HTTP/1.1 302 Found
  Location: https://client.example.org/cb?
    code=SplxlOBeZQQYbYS6WxSbIA
```

Legal Basis Check

After checking the identity and additional claims of the requestor, the issuer has to check if the request can be allowed. One of the things to check is the legal basis. Depending on the use-case different legal-basis are relevant. This example shows an emergency and an explicit consent provided by the patient. There exists other legal basis which are not shown in this example.

These steps might be almost instantaneously or take hours or days, depending on the use-case and possible manual checks. Meanwhile, the wallet will start polling and retrying based on the interval response from the AS.

8. GET consent from local consent system

Steps 8 to 12 describe the flow where a consent is the legal basis.

The custodian has the responsibility to administer patient consent for sharing data. The custodian has to check its administration if there is a consent.

This request must be standardized so all consent systems can support the API and make integrations between Authorization Servers and consent systems universal.

9. Consent found

The consent system response positively on the query.

10. The Authorization Server allows the request

After receiving positive confirmation from the consent system, the AS registers this request as valid.

Steps 11 - 12 No consent

No consent was found, deny the request

13. Emergency

This is an example where an emergency can be a legal basis. The request is logged and allowed instantaneously.

Get access_token

After step 7, when the wallet receives the authorization code, the wallets starts polling the token endpoint for the `access_token`.

14. GET AS/token?code

The wallet tries to get the access_token from the token endpoint. According to OIDC4VCI section 6.1

```
POST /token HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW

  grant_type=authorization_code
  &code=SplxlOBeZQQYbYS6WxSbIA
  &code_verifier=dBjftJeZ4CVP-mB92K27uhbUJU1p1r_wW1gFWFOEjXk
  &redirect_uri=https%3A%2F%2FWallet.example.org%2Fcb
```

Note: the value of the Authorization header must be specified. Probably based on RFC7523

15. Legal basis check pending

The legal basis checks are not finished yet, the token endpoint instructs the client to try again later by responding with the `authorization_pending` field.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

 {
    "authorization_pending": true,
    "interval": 300
 }
```

16. Legal basis valid

The legal base is valid, the wallet now receives the access_token:

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

 {
  "access_token": "eyJhbGciOiJSUzI1NiIsInR5cCI6Ikp..sHQ",
  "token_type": "bearer",
  "expires_in": 86400,
  "c_nonce": "tZignsnFbp",
  "c_nonce_expires_in": 86400
 }
```

17. Legal basis invalid

OIDC4VCI section 6.3 There was not valid legal basis, the token endpoint responds with an error response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
  "error": "legal basis invalid"
}
```

Note: this should be specified in more detail.

18. Get credential

Using the `access_token` and the `n_once`, the wallet can request the credential:

```
POST /credential HTTP/1.1
Host: server.example.com
Content-Type: application/json
Authorization: BEARER czZCaGRSa3F0MzpnWDFmQmF0M2JW

{
  "format": "ldp_vc",
  "credential_definition": {
    "@context": [
      "https://www.w3.org/2018/credentials/v1",
      "https://nuts.nl/credentials/v1"
    ],
    "types": [
      "VerifiableCredential",
      "NutsAuthorizationCredential"
    ],
    "credentialSubject": {
      "id": "did:nuts:123",
      "purposeOfUse": "careviewer",
      "patient":"bsn:9999992"
    }
  },
  "proof": {
    "proof_type": "jwt",
    "jwt": "eyJraWQiOiJkaWQ6ZXhhbXBsZ...KPxgihac0aW9EkL1nOzM"
  }
}
```

Note: the value of the Authorization header must be specified. Probably based on RFC7523

19. Credential response

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
```

```
{
   "format": "ldp_vc",
   "credential": {
      "@context": [
         "https://www.w3.org/2018/credentials/v1",
         "https://nuts.nl/credentials/v1"
      ],
      "id": "did:web:issuer#123",
      "type": [
         "VerifiableCredential",
         "NutsAuthorizationCredential"
      ],
      "issuer": "did:web:custodian",
      "issuanceDate": "2010-01-01T00:00:00Z",
      "credentialSubject": {
         "id": "did:web:requestor",
         "patient":"bsn:999992",
         "purposeOfUse": "careviewer",
      },
      "proof": {
         "type": "Ed25519Signature2020",
         "created": "2022-02-25T14:58:43Z",
         "verificationMethod": "https://example.edu/issuers/565049#key-1",
         "proofPurpose": "assertionMethod",
         "proofValue": "zeEdUoM7m9cY8ZyTpey83yBKeBcmcvbyrEQzJ19rD2UXArU2U1jPGoEt
                        rRvGYppdiK37GU4NBeoPakxpWhAvsVSt"
      }
   },
   "c_nonce": "fGFF7UkhLa",
   "c_nonce_expires_in": 86400
}
```

Example AccessToken introspection

Example of an accessToken introspection response.

This functionality is currently *work in progress* and is likely to change. Please use the official documentation for the latest up-to-date and supported information.

This accessToken has been generated at 31 January 2024 as part of a POC demonstrating the presentation by a client of both the example URA and Vektis credentials which can be used by the relying party to interact with the Mitz consent service.


```
{
  "active": true,
  "client_id": "did:web:nuts-node-gbz.nuts.example.nl:iam:d73ca5d4-4dc8-4137-8992-578e825e3f36",
  "exp": 1706689514,
  "iat": 1706688614,
  "input_descriptor_constraint_id_map": {
    "uracredential_uraNumber": "32475534",
    "vektisOrgCredential_orgType": "0110"
  },
  "iss": "did:web:nuts-node-zkh.nuts.example.nl:iam:2333ea28-a719-4896-9a12-f855b225755b",
  "presentation_definition": {
    "format": {
      "jwt_vc": {
        "alg": [
          "PS256",
          "PS384",
          "PS512",
          "ES256",
          "ES384",
          "ES512",
          "ES256K"
        ]
      },
      "jwt_vp": {
        "alg": [
          "PS256",
          "PS384",
          "PS512",
          "ES256",
```

```
            "ES384",
            "ES512",
            "ES256K"
        ]
    },
    "ldp_vc": {
        "proof_type": [
            "JsonWebSignature2020"
        ]
    },
    "ldp_vp": {
        "proof_type": [
            "JsonWebSignature2020"
        ]
    }
},
"id": "geboortekaart_policy",
"input_descriptors": [
    {
        "constraints": {
            "fields": [
                {
                    "filter": {
                        "const": "CibgUraCredential",
                        "type": "string"
                    },
                    "path": [
                        "$.type"
                    ]
                },
                {
                    "filter": {
                        "const": "did:web:cibg.mitz-x-nuts.headease.nl",
                        "type": "string"
                    },
                    "path": [
                        "$.issuer"
                    ]
                },
                {
                    "id": "uracredential_uraNumber",
```

```
            "filter": {
               "type": "string"
            },
            "path": [
               "$.credentialSubject.uraNumber"
            ]
         }
      ]
   },
   "id": "cibg_ura_credential"
},
{
   "constraints": {
      "fields": [
         {
            "filter": {
               "const": "VektisOrgCredential",
               "type": "string"
            },
            "path": [
               "$.type"
            ]
         },
         {
            "filter": {
               "type": "string"
            },
            "path": [
               "$.issuer"
            ]
         },
         {
            "id": "vektisOrgCredential_orgType",
            "filter": {
               "type": "string"
            },
            "path": [
               "$.credentialSubject.orgType"
            ]
         }
      ]
```

```
            },
            "id": "vektis_org_credential"
         }
      ],
      "name": "Geboortekaart Vereiste Input Descriptors",
      "purpose": "Het vinden van de juiste credentials voor het doen van een mitzXnuts access_token aanvraag"
   },
   "presentation_submission": {
      "definition_id": "geboortekaart_policy",
      "descriptor_map": [
         {
            "format": "jwt_vc",
            "id": "cibg_ura_credential",
            "path": "$.verifiableCredential[0]"
         },
         {
            "format": "jwt_vc",
            "id": "vektis_org_credential",
            "path": "$.verifiableCredential[1]"
         }
      ],
      "id": "dd5a0697-45e6-4d02-9c6a-b518377be4be"
   },
   "scope": "geboortekaart",
   "sub": "did:web:nuts-node-zkh.nuts.example.nl:iam:2333ea28-a719-4896-9a12-f855b225755b",
   "vps": [

"eyJhbGciOiJFUzI1NiIsImtpZCI6ImRpZDp3ZWI6bnV0cy1ub2RlLWdiei5taXR6LXgtbnV0cy5oZWFkZWFzZS5ubDppY
W06ZDczY2E1ZDQtNGRjOC00MTM3LTg5OTItNTc4ZTgyNWUzZjM2IzAiLCJ0eXAiOiJKV1QifQ.eyJhdWQiOlsiZGlkOndl
YjpudXRzLW5vZGUtemtoLm1pdHoteC1udXRzLmhlYWRlYXNlLm5sOmlhbToyMzMzZWEyOC1hNzE5LTQ4OTYtOWE
xMi1mODU1YjIyNTc1NWIiXSwiZXhwIjoxNzA2Njg4NjE5LCJqdGkiOiJkaWQ6d2ViOm51dHMtbm9kZS1nYnoubWl0ei1
4LW51dHMuaGVhZGVhc2Uubmw6aWFtOmQ3M2NhNWQ0LTRkYzgtNDEzNy04OTkyLTU3OGU4MjVlM2YzNiNlODVh
NjVhZi1lNGU4LTQ5ZTQtOGY1Ni1iZjc2YzU2MTliZDIiLCJuYmYiOjE3MDY2ODg2MTQsIm5vbmNlIjoiMERvQlhRQV9fV1
pxSENsX1lpeVplZyIsInN1YiI6ImRpZDp3ZWI6bnV0cy1ub2RlLWdiei5taXR6LXgtbnV0cy5oZWFkZWFzZS5ubDppYW
06ZDczY2E1ZDQtNGRjOC00MTM3LTg5OTItNTc4ZTgyNWUzZjM2IiwidnAiOnsiQGNvbnRleHQiOlsiaHR0cHM6Ly93d
3cudzMub3JnLzIwMTgvY3JlZGVudGlhbHMvdjEiXSwidHlwZSI6IlZlcmlmaWFibGVQcmVzZW50YXRpb24iLCJ2ZXJpZml
hYmxlQ3JlZGVudGlhbCI6WyJleUpoYkdjaU9pSkZVekkxTmtzaUxDSjBlWEFpT2lKS1YxUWlMQ0pyYVdRaU9pSmthV1E
2ZDJWaU9tTnBZbWN1YldsMGVpMTRMVzUxZEhNdWFHVmhaR1ZoYzJVdWJtd2piazVpWTJsc1N6bHhlVGxNZW1WTk
1WTlJNMWs1T1hSbFYwbDVkMmRKWm0xM0xXbG5ZM2wyTmpWUVl5SjkuZXlKMll5STZleUpBWTI5dWRHVjRkQ0k2
V3lKb2RIUndjem92TDNkM2R5NTNNeTV2Y21jdk1qQXhPQzlqY21Wa1pXNTBhV0ZzY3k5Mk1TSXNJbWgwZEhCek9p
OHZZMmxpWnk1dWJDOHlNREkwTDJOeVpXUmxiblJwWVd4ekwyTnBZbWQxY21GamNtVmtaVzUwYVdGc0lsMHNJb
```

```
Vp2Y20xaGRDSTZJbXAzZEY5Mll5SXNJbWxrSWpvaU16TmxORE00WVRNdE5EVmhZaTAwWVRjMExXSXdOVEV0TVR
sallUSTROVE0yWTJRd0lpd2lkSGx3WlNJNld5SldaWEpwWm1saFlteGxRM0psWkdWdWRHbGhiQ0lzSWtOcFltZFZjbU
ZEY21Wa1pXNTBhV0ZzSWwwc0ltTnlaV1JsYm5ScFlXeFRkV0pxWldOMElqcDdJblZ5WVU1MWJXSmxjaUk2SWpNeU5
EYzFOVE0wSWl3aWFXUWlPaUprYVdRNmQyVmlPbTUxZEhNdGJtOWtaUzFuWW5vdWJXbDBlaTE0TFc1MWRITXVhR
1ZoWkdWaGMyVXVibXc2YVdGdE9tUTNNMk5oTldRMExUUmtZemd0TkRFek55MDRPVGt5TFRVM09HVTRNalZsTTJZ
ek5pSjlMQ0pwYzNOMVpYSWlPaUprYVdRNmQyVmlPbU5wWW1jdWJXbDBlaTE0TFc1MWRITXVhR1ZoWkdWaGMyV
XVibXdpTENKcGMzTjFZVzVqWlVSaGRHVWlPaUl5TURJMExUQXhMVE13OkRFME9qVXhPalV5TGprME5Wb2lmU3dpY
VdGMElqb3hOekEyTmpJMk16RXlMQ0pwYzNNaU9pSmthV1E2ZDJWaU9tTnBZbWN1YldsMGVpMTRMVzUxZEhNdWF
HVmhaR1ZoYzJVdWJtd2lMQ0psZUhBaU9qRTNNRGt5TVRnek1USXNJbk4xWWlJNkltUnBaRHAzWldJNmJuVjBjeTF1YjJ
SbExXZGllaTV0YVhSNkxYZ3RiblYwY3k1b1pXRmtaV0Z6WlM1dWJECHBZVzA2WkRjelkyRTFaRFF0TkdSak9DMDBNV
E0zTFRnNU9USXROVGM0WlRneU5XVXpaak0ySW4wLm9yXzRLQnIwM3ZFd3ZVWkhuc21Sdy1FRFduQzJfcGxyaWot
d0stN0FDVEstNzE5RmRGQ2lQdmYzZEhib2ZlR19NOFZ6TFpuaXgxc3VjcTZkYk5YTi13liwiZXlKaGJHY2lPaUpGVXpJM
U5rc2lMQ0owZVhBaU9pSktWMVFpTENKcmFXUWlPaUprYVdRNmQyVmlPblpsYTNScGN5NXRhWFI2TFhndGJuVjBjeT
VvWldGa1pXRnpaUzV1YkNOaWJqZHdUMVZ1WTBSbWRESjZjamxyUkdjM2VIbEdSalo0VUVaSmVWVkNSMlI1YTJwN
WVIZE5SVXBGSW4wLmV5SjJZeUk2ZXlKQVkyOXVkR1Y0ZENJNld5Sm9kSFJ3Y3pvdkwzZDNkeTUzTXk1dmNtY3ZNa
kF4T0M5amNtVmtaVzUwYVdGc2N5OTJNU0lzSW1oMGRIQnpPaTh2WTJsaVp5NXViQzh5TURJMEwyTnlaV1JsYm5ScF
lXeHpMM1psYTNScGMyOXlaMk55WldSbGJuUnBZV3dpWFN3aVptOXliV0YwSWpvaWFuZDBYM1pqSWl3aWFXUWlP
aUkwTlRWbVl6YzFOUzFrT0RZd0xUUmpOelF0T1RFNU15MWxaV0V6WWpZMlltUmtaalFpTENKMGVYQmxJanBiSWxa
bGNtbG1hV0ZpYkdWRGNtVmtaVzUwYVdGc0lpd2lWbVZyZEdselQzSm5RM0psWkdWdWRHbGhiQ0pkTENKamNtV
mtaVzUwYVdGc1UzVmlhbVZqZENJNmV5SjFjbUZPZFcxaVpYSWlPaUl6TWpRM05UVXpOQ0lzSW1sa0lqb2laR2xrT25
kbFlqcHVkWFJ6TFc1dlpHVXRaMko2TG0xcGRIb3RlQzF1ZFhSekxtaGxZV1JsWVhObWxoYlRwa056TmpZV
FZrTkMwMFpHTTRMVFF4TXpjdE9EazVNaTAxTnpobE9ESTFaVE5tTXpZaWZTd2lhWE56ZFdWeUlqb2laR2xrT25kbFl
qcDJaV3QwYVhNdWJXbDBlaTE0TFc1MWRITXVhR1ZoWkdWaGMyVXVibXdpTENKcGMzTjFZVzVqWlVSaGRHVWlPaU
l5TURJMExUQXhMVE13VkRFME9qVXlPakF4TGpnMk5sb2lmU3dpYVdGMElqb3hOekEyTmpJMk16SXhMQ0pwYzNNa
U9pSmthV1E2ZDJWaU9uWmxhM1JwY3k1dGFYUjZMWGd0Ym5WMGN5NW9aV0ZrWldGelpTNXViQ0lzSW1WNGND
STZNVGN3T1RJeE9ETXlNU3dpYzNWaUlqb2laR2xrT25kbFlqcHVkWFJ6TFc1dlpHVXRaMko2TG0xcGRIb3RlQzF1ZFhS
ekxtaGxZV1JsWVhObWxoYlRwa056TmpZVFZrTkMwMFpHTTRMVFF4TXpjdE9EazVNaTAxTnpobE9ESTFa
VE5tTXpZaWZRLmZOTFZSV1BWNUMzZkFrbUI3VUVCMGltdUVlcHllUENfTUp1dVNxVDQtSmptajFsWVBRRXNaNGcx
MFExMzlYbGtCNFZSR1Y4NzJNNUZpZ251Z2FsNWpBIl19fQ.qfw3wz7rtWD_28lVz6Lq8hWt_ZuXEn4N5IKCNDmW0Tp
kbYpJT0Ni7tfZMNHqnLPVsoIBR1QpqFfYYgaKWuzpOQ"
  ]
}
```