MyData Architecture Framework

Data Transfer Specification

v. 2.0

Notice

Implementation or use of certain elements of this document may require licenses under third party intellectual property rights, including without limitation, patent rights. The Participants of and any other contributors to the Specification are not and shall not be held responsible in any manner for identifying or failing to identify any or all such third party intellectual property rights. This Specification is provided "AS IS", and no Participant makes any warranty of any kind, expressed or implied, including any implied warranties of merchantability, non-infringement of third party intellectual property rights, and fitness for a particular purpose.

MyData Architecture defines the operations and APIs between the Operational Roles (Operator, Source, Sink etc.). Any descriptions or figures of the roles' internal structure or operations are for illustrative purposes only.

1. Introduction

This document specifies Data Transfer in MyData Architecture Framework.

This document is part of the MyData Architecture Framework release 2.0. The reader is assumed to be familiar with the 'MyData Architecture Framework' document and with the parallel technical specification documents available at https://github.com/mydata-sdk/mydata-docs/tree/master/architecture_specs .

1.1 Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

1.2 Terminology

Key terminology used in this specification is defined in the Glossary of MyData Architecture Framework release 2.0 available at https://github.com/mydata-sdk/docs/.

1.3 Formats

In this framework, all data records and their respective digital signatures exchanged between actors are expressed using Javascript Object Notation (JSON). Digital signatures are expressed as JSON Web Signature (JWS) structures and cryptographic keys as JSON Web Key (JWK) structures.

In this document, JSON definitions of the data records are presented without JWS structures.
All Timestamps are in UTC in the NumericDate format as defined in [RFC7519].

2. Data Transfer Model

A ***Data Transfer*** is an authorised transfer of data from a specific Source to a specific Sink. The authorisation is given by Account Owner in the MyData Consenting transaction, and multiple requests from Sink to Source may happen as long as the authorisation is active.

2.1 Data Request

Sink requesting data based on consent MUST ensure that Consent Record is Active, the intended use of data is listed in Consent Record's purposes and that the request is made to protected resource listed in Consent Record's resource set description.

Sink request to a protected resource MUST be signed using mechanism defined in [draft-ietf-oauth-signed-http-request] to prove possession of an access token and its associated key.

When the Source receives Data Request it MUST verify the request, token, Consent Record status and sufficiency. The Source MUST ensure that the request is for a dataset listed in Consent Record's resource set description, was made to corresponding dataset distribution URL listed, and that the potential constraints set in the Consent Record are not violated.
Detailed steps for request construction and verification are described in 3.2.1 and 3.2.2 respectively.

2.2 Authorisation Token

MyData Authorisation Token is an Operator-issued cryptographically signed proof-of-possession token given to Sink and expressed in the JSON Web Token (JWT) format as described in [RFC7519]. The token is used in data request to refer to a specific consent.

The Operator MUST provide token to the Sink when requested if the associated consent record is valid and the previously issued token has expired or is about to expire. The Operator MAY provide previously issued valid token if the difference between current time and token expiration time is not within the threshold. The contents of the token and token verification are defined in chapter 4.

3. Data Transfer Transactions

This section describes the Data Transfer flows. There are two main transactions: Sink requesting Authorisation Token from Operator and Sink requesting data from Source.

3.1 Sink Requests Authorisation Token

If Sink does not have an Authorisation Token or the token has expired, Sink MUST request a new token before making a data request. Sink MAY also request a new token even if the previous token has not yet expired. Operator MUST provide new token to the client if the associated Consent Record is valid and the previously issued token has expired or is about to expire. It MAY provide previously issued valid token if the difference between current time and token expiration time is not within the threshold.

Transaction is depicted in Figure 3.1.

Prerequisites: Sink has a valid Consent Record.

Process:

- Sink requests a new Authorisation Token from Operator by using the corresponding Consent Record ID.
- When Operator receives a request for a new token, it fetches the indicated Consent Record
- Operator MUST check that the consent is valid and active. If it is not, Operator MUST end the process and respond with an appropriate error code and message.
- Operator decides if it generates a new Authorisation Token or provides previously generated Authorisation Token. Decision MUST be based on predefined threshold between current time and token expiration time.
- Operator returns token to Sink.

Outcome: Sink receives an Authorisation Token.

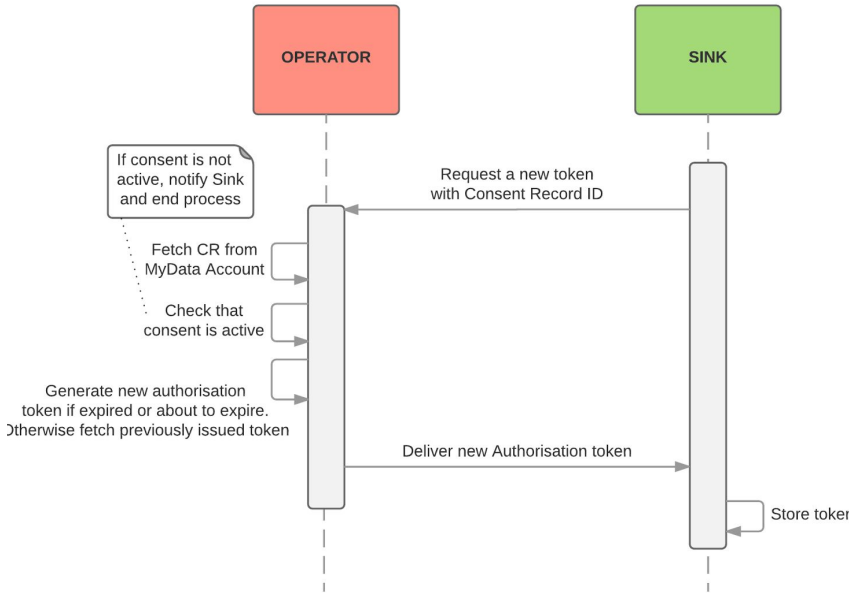


Figure 3.1: Sink request a new Authorisation token from Operator

3.2 Sink Requests Data from Source

Sink MUST ensure that the consent associated with the resource to be requested is active and contains sufficient rights for intended use before making a request to protected resource. Furthermore the Sink MUST ensure that the time of the impending request is within Authorisation token's validity period. Request to a protected resource MUST be signed using as defined in [draft-ietf-oauth-signed-http-request]. Data request transaction is depicted in Figure 3.2.

Prerequisites: Sink has a valid Authorisation Token and Consent Record

Process:

- Sink makes a data request to Source. The request MUST be signed as defined in [draft-ietf-oauth-signed-http-request].
- Source receiving the data request validates the request, provided token and the related Consent Record. Request MUST be validated as defined in section 3.2.2.
- Source either denies or grants access to requested resources based on outcome of the validation.

Outcome: Based on the validation process conducted by Source, Sink either receives the data it requested or receives an error message.

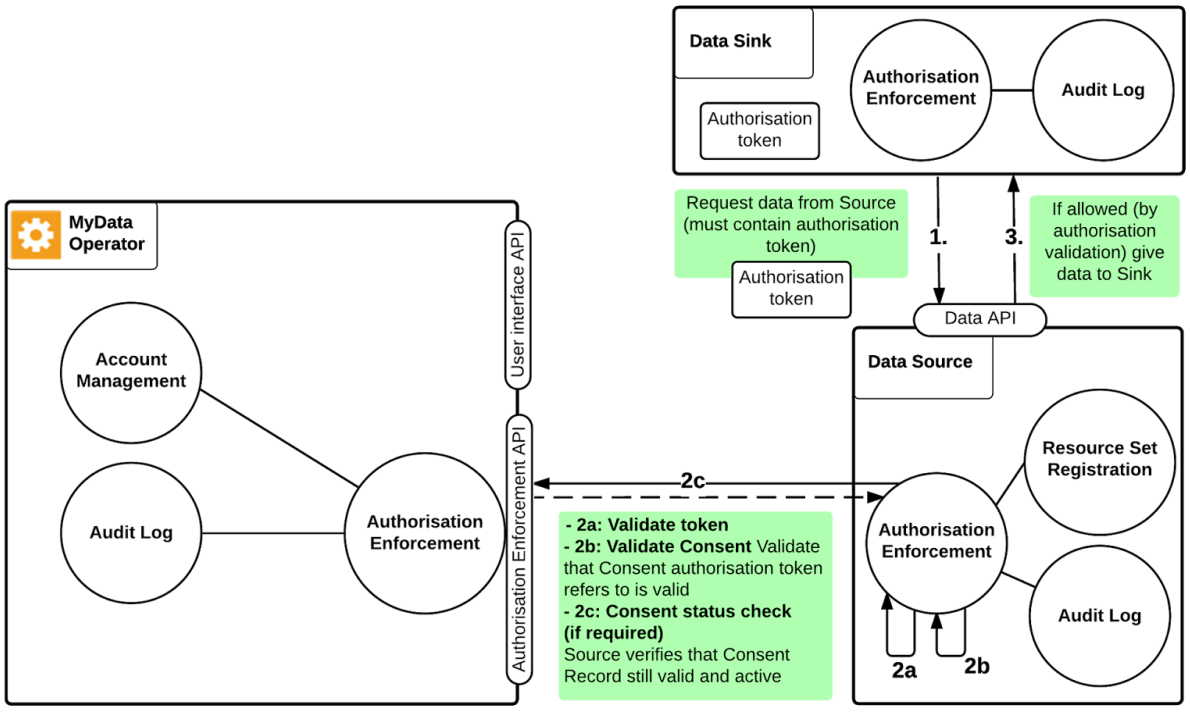


Figure 3.2: Data transfer flow

3.2.1 Request Creation

Before making Data Request Sink MUST ensure that the consent associated with the resource to be requested is valid and contains sufficient rights for intended use before making a request to protected resource. This entails following steps.

Process:

- Consent Record is Valid if
  - Current time is between Consent's not before and not after timestamps
  - Consent status in latest Status Record is Active
  - Depending on policy, the status may be validated either locally or using policy enforcement endpoint provided by Operator.
- Consent Record contains sufficient purposes for intended use
- Consent Record contains authorisation for intended protected resource

Data Request MUST be signed, signed object MUST contain Authorisation Token value and timestamp, and MAY contain other values as defined in [draft-ietf-oauth-signed-http-request].

3.2.2 Request Validation

For Source to grant access to the requested data, it MUST validate the Sink's authorisation to access that data. This entails validating the token, Data Request and the related Consent Record as detailed in the following sections.

Process:

- verify there exists an associated Consent Record with provided Consent Record ID
- verify the token signature with the token issuer key contained in the Consent Record
- verify token's validity period includes the time of data request
- verify the request signature as defined in [draft-ietf-oauth-signed-http-request] with the proof-of-possession key contained in the Consent Record
- verify that request was made to resource authorised by the Consent Record. Exact verification MAY vary depending API structure but MUST include at least verification that token's audience includes the URL to which the data request was made.
- Verify that the associated Consent Record is Valid
    - Depending on policy the status may be validated either locally or using policy enforcement endpoint provided by the Operator.
    - Consent Record is Active if
        - Current time is between not before and not after timestamps
        - Consent status in latest Status Record is Active

Outcome:

- If all verifications pass grant access to requested data.
- Otherwise, deny the data request with appropriate error message.

4. Authorisation Token

This section defines the structure of the authorisation token, which is a *signed JSON Web Token* as described in [RFC7519].

4.1 Authorisation Token Details

Table 4.1. Structure of Authorisation Token's payload

<table>
<tr><th>KEY</th><th>TYPE</th><th>DESCRIPTION</th></tr>
<tr><td>iss</td><td>String</td><td>Unique ID of the authorisation server that issued the JWT</td></tr>
<tr><td>cnf</td><td>Object</td><td>
<table>
<tr><th>KEY</th><th>TYPE</th><th>DESCRIPTION</th></tr>
<tr><td>kid</td><td>String</td><td>Key ID of Public key of the token user (Sink). Used to verify data request</td></tr>
</table>
</td></tr>
<tr><td>aud</td><td>Array of Strings</td><td>Exhaustive list of URL patterns to which data requests are allowed with this token.</td></tr>
<tr><td>exp</td><td>Integer</td><td>The expiration time of the token or after which the token MUST NOT be accepted.</td></tr>
<tr><td>nbf</td><td>Integer</td><td>The time before which the token MUST NOT be accepted.</td></tr>
<tr><td>iat</td><td>Integer</td><td>The time at which the JWT was issued.</td></tr>
<tr><td>jti</td><td>String</td><td>The "jti" (JWT ID) claim provides a unique identifier for the JWT.</td></tr>
<tr><td>cr_id</td><td>String</td><td>Source's Consent Record ID linked to the token.</td></tr>
</table>

Token MUST be signed with the issuer's private key as defined in [RFC7515].

The JWS header MUST contain 'kid' field identifying token issuer's key pair used to sign the token.

4.1.1 Authorisation token payload

```
{
  "iss":String,
  "cnf": {
    "kid":String
  },
  "aud": [
    String
  ],
  "exp": Integer,
  "nbf": Integer,
  "iat": Integer,
  "jti": String,
  "cr_id": String
}
```

5. Data Transfer APIs

Data Transfer has been implemented as a service in MyData SDK. Related developer API documentation and detailed flow diagrams clarifying the implementation are linked and documented in this section.

5.1 Interfaces of Different Actors

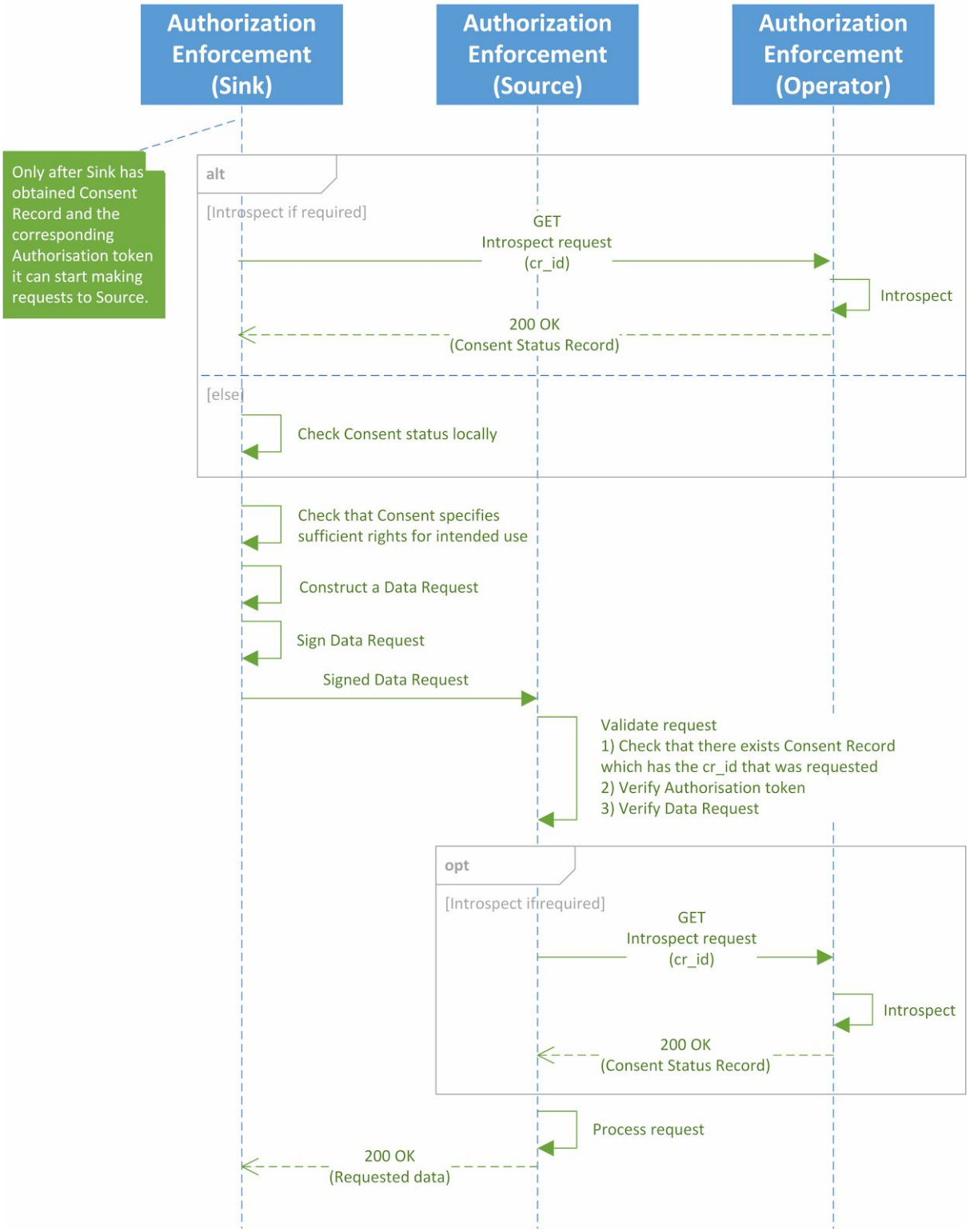
Operator MUST provide an Authorisation Enforcement API for Sources and Sinks, through which they can request consent introspection. Sink can also request Authorisation Tokens using Operator's Authorisation Enforcement API.

Note: Source has the freedom to choose the design of its endpoint and API resources as long as the documentation provided in the Service Description (see *Service Descriptions*) describes this sufficiently. Sink receives the endpoint details as part of the authorisation process.

5.2 API Specification

An API for handling Data Transfers on a Sink and Source (implemented at MyData at SDK) is available at: https://github.com/mydata-sdk/mydata-docs/tree/master/api_specs

5.3 Detailed Flow



6. References

[RFC2119] Bradner, S, "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
[RFC7515] Jones, M., Bradley, J., Sakimura, N. “JSON Web Signature (JWS)”, RFC 7515, May 2015
[RFC7519] Jones, M., Bradley, J., Sakimura, N. “JSON Web Token (JWT)”, RFC 7519, May 2015
[draft-ietf-oauth-signed-http-request] Bradley, J., Tschofenig,H. “A Method for Signing HTTP Requests for OAuth draft-ietf-oauth-signed-http-request-03”, Internet-Draft, August 08, 2016