MyData Authorisation Specification

Notice

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

1. Introduction

This document specifies MyData Authorisation and its management over its lifetime.

This document is part of the MyData architecture release 1.2.1 The reader is assumed to be familiar with the 'MyData Architecture - Consent Based Approach for Personal Data Management' and with the parallel technical specification documents document available at https://hiit.github.io/mydata-stack/.

Known deficiencies in this release: limited error handling and HTTP error messages. Support for granting permission for a service to (further) process data it already has. These will be part of the next release of this document.

1.1 Terms and Definitions

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

1.2 Terminology

Key terminology used in this specification is defined in the Glossary of MyData Architecture - Consent Based Approach for Personal Data Management release 1.2.1 available at https://hiit.github.io/mydata-stack/.

Consent [lawful basis] is one of the grounds for lawfulness of processing personal data and it is given by the Data Subject for one or more specific purposes (Arts 6-7 GDPR). Consent means "any freely given, specific, informed and unambiguous indication of the data subject's wishes" by which agreement to processing of their personal data is signified, either by statement or clear affirmative action (Art. 4 (8) GDPR). With regard to specific categories of personal data (Art. 9), automated decisions, including profiling (Art. 22), and transfers of personal data to third countries (i.e. outside the EU) in certain circumstances (Art. 49) consent must also be explicit.

Consent Record (CR) documents the permission the Account Owner has granted to a specific service. For authorising data processing within a service, the Account Owner creates a single Consent Record for the related service. For authorising data transfer from a specific Source to a specific Sink, the Account Owner creates a pair of Consent Records (one for the Source and one for the Sink). The Source's CR defines, what data can be provisioned to the specified Sink, and the Sink's CR defines, how the data can be accessed. The Sink's CR can also include the permissions for data processing. A Consent Record is a manifestation of legally valid Consent and makes it technically feasible to change or withdraw the consent dynamically. Consent Records are stored in the MyData Account.

Consent Status Record (CSR) is a record MyData Operator sends to a service when status of a consent changes. Service MUST store these records for future use.

1.3 Diff/Changes

In release 1.2.1:

- timestamp format corrected
- MyData Consent Record details corrected

1.4 Formats

In MyData Architecture, 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. MyData Consent Model

In the current version of MyData architecture, all data processing is based on consents, one of the possible legal bases for processing personal data. Support for other legal bases is deferred to a later architecture release.

It is characteristic to consents that they are always issued by the Account Owner (data subject), who can also change or withdraw the consent at will. It should be noted that consent does not legitimise all sorts of processing activities, nor can it negate obligations stemming from general principles related to processing of personal data.

2.1 MyData Consent Record

When Account Owner issues a consent, in MyData Architecture it is documented in a ***MyData Consent Record (CR)***. A Consent Record is a manifestation of legally valid Consent and makes it technically feasible to change or withdraw the consent dynamically. CRs are stored in the MyData Account and at the related service. Source and Sink MUST verify the signature of the CR and reject it, if it is not signed with one of the Account Owner's keys defined in the related Service Link Record.

For authorising data processing within a service, the Account Owner creates a single CR for the related service. For authorising data transfer from a specific Source to a specific Sink, the Account Owner creates a pair of CRs (one for the Source and one for the Sink). Then, the Source's CR defines, what data can be provisioned to the specified Sink, and the Sink's CR defines, how the data can be accessed. The Sink's CR can also include the permissions for data processing.

Note: The CR constructed in MyData Architecture differs in certain key ways from the Kantara Initiative's CR (KI-CR), which this specification includes as a sub-structure. The differences stem from the supported use case in MyData Architecture: e.g. CR's status can be adjusted dynamically and it defines the exact authorised resources at the Source. Also, CR is signed by the Account Owner and not Data Controller. Deeper integration of KI-CR as a machine-readable record for MyData Architecture CRs is deferred to a later architecture release.

2.2 Resource Sets

CRs refer to Account Owner's data using Resource Sets. A ***Resource Set*** defines a specific (subset) of the Account Owner's data that a particular Source provisions or processes. Source can provision subsets of the original data set in different ways by defining different Resource Sets. Service Registry contains a description of all data a service can provision or process. Account Owner creates a Resource Set while creating a consent by selecting a set of data for processing.

Resource Set is identified using ***Resource Set ID***, rs_id. rs_id is composed of a *URI* that identifies the Source and a *resource key* unique within that Source, which makes it a globally unique identifier. It MUST be used only in one consent at the time. This means that while the same Resource Set can be used in more than one consent, each must use a different rs_id. URI or the resource key must not leak

out any specific information about the user or data but only provide reference to the Source e.g. com.example.a3h413h4b13h41.

2.3 Consent Lifecycle

A Consent Record is a signed, immutable object, so the status of the consent is indicated by a separate ***Consent Status Record (CSR)***. CRs have three valid states:

- *Active*, Account Owner's data can be processed according to rules set in consent
- *Disabled*, Data processing is temporarily not allowed
- *Withdrawn*, Data processing is permanently not allowed

When issued, CR is in *Active* state. Account Owner can also withdraw CR (from any other state) at will by setting its state to *Withdrawn*. A withdrawn consent cannot be reactivated, a new consent must be created in it's place, if processing is to continue. Finally, Operator sets a CR to *Disabled* state whenever there is reason to think the CR is no longer valid, e.g. when the Service Link between the related service and Account Owner is removed, or Service behaves suspiciously. Operator must record internally the reason(s) for disabling the CR. A disabled consent can be re-activated or withdrawn by the Account Owner or by the Operator depending on the reason for disabling. All states and allowed transitions are shown in Figure 2.1.

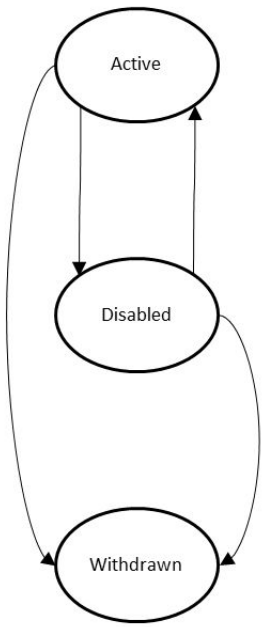

Figure 2.1: Consent lifecycle

When Account Owner changes the status of the CR given to a Sink, the status of the CR given to related Source MUST also be updated accordingly to ensure that Source doesn't provide any data to the Sink while Sink is not allowed to receive new data. If Account Owner changes the status of the CR given to Source, it is sufficient to notify the Source only.

When Sink-related Service Link is changed, both the CR given to the Sink and the related CR(s) given to Source(s) are set to *Disabled* state. If Source-related Service Link is changed, it is sufficient to put Source CR to *Disabled* state.

2.4 Consent Status Record

Account Owner or Operator can modify a CR's status by issuing *Consent Status Records*, which are stored at the at the Operator and sent to the related service. Services MUST maintain local copies of CSRs.

Source and Sink MUST verify the signature of the CSR and reject status changes that are not signed with one of the keys contained in Service Link Record. Source and Sink MUST also verify the CSRs form an uninterrupted chain by verifying that the prev_record_id field always equals previous CSR's id (record_id field) or NULL for the first CSR.

If the service can not verify the CSR, it MUST stop data processing based on that CR and update the status of this CR by requesting the related Consent Status Record from the Operator.

2.5 Consent Record Verification and Validation

CR MUST be verified when service receives it from the Operator.
Verification steps consist of:

1. verify CR is issued by authorised party
2. verify CR has not been corrupted
3. verify that is well-formed and contains all mandatory information
4. verify associated Consent Status Record

CR is verified if, and only if all the verification steps are passed.

CR MUST be validated every time when the data is processed based on the consent.
Consent is valid if:

1. Time of use is between not before and not after timestamps; AND
2. Consent status in latest Consent Status Record is Active

It is assumed that the Operator will deliver Consent Status Records immediately to relevant services. If the service is uncertain whether it has the latest CSR it can request it from the Operator.

3 MyData Authorisation Transactions

This section describes MyData Authorisation transactions. There are four main transactions:

- Issuing consent
- Withdrawing consent
- Modifying consent status
- The related SSR changing to *Removed*.

Modifications to an existing consent are implemented by withdrawing the existing consent and issuing a new consent based on modifications made by Account Owner.

Service Registry contains the minimum requirements of sinks have for particular types of consents. It is up to the UI to enforce that the created consents meet the minimum requirements.

3.1 Account Owner Issues Consent

Prerequisites: There is a Service Link between Account Owner and Service

Process: Account Owner issues consent (for data processing within a service) or pair of consents (for data connection between Source and Sink) at the Operator.

Outcome: Consent Record(s) that contain all required information [for both technical & legal perspective]. Consent Record(s) are distributed to relevant parties. Sink can then request the authorisation token from Operator before a data request can take place. For detailed management and lifecycle of authorisation token see MyData Data Connection specification.

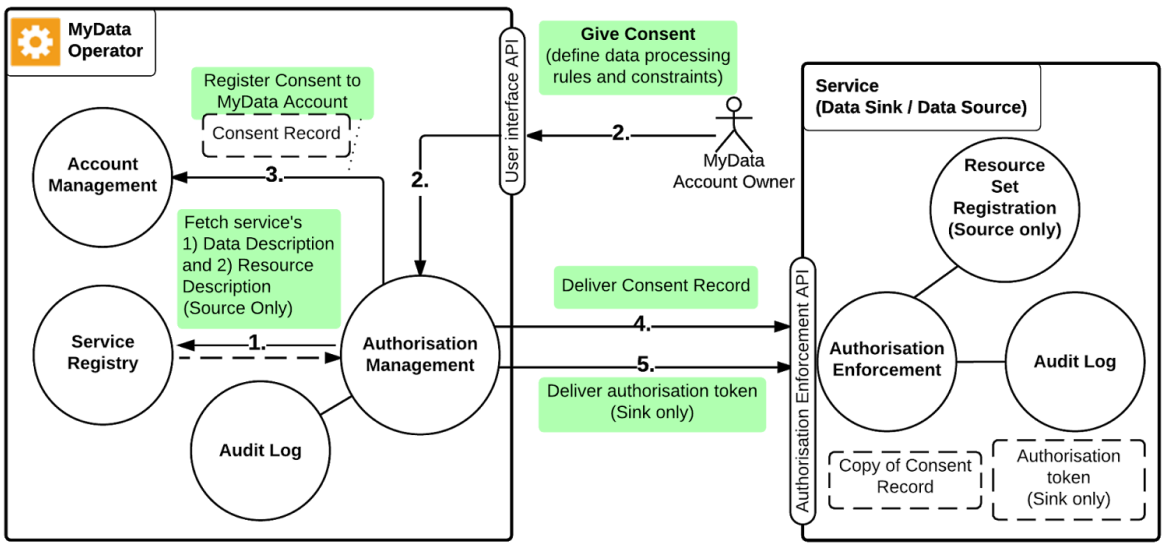

Figure 3.1: Authorisation flow. For details of step 5, creation and delivery of authorisation token, see MyData Data Connection specification.

After a Source and a Sink have been linked at MyData Account, the Sink can be authorised to access data on the Source by conducting MyData Authorisation step. The Authorisation results in a pair of Consent Records. In the first, Account Owner gives an explicit consent for particular Sink to access a specific Resource Set on the Source (depicted in the Source's Consent Records). In the second, Account Owner also authorises Sink to process the specified data according to terms defined in the Sink's Consent Records. The contents of a Consent Records are depicted in *section 4.1*. So both Source and Sink have their own Consent Records, which contain role specific information necessary in establishing a Data Connection between Source and Sink.

Upon creation of Consent Record, Operator delivers both Records to the corresponding services as shown in *Figure 2* (Sink's Record is delivered to Sink and Source's Record is delivered to Source).

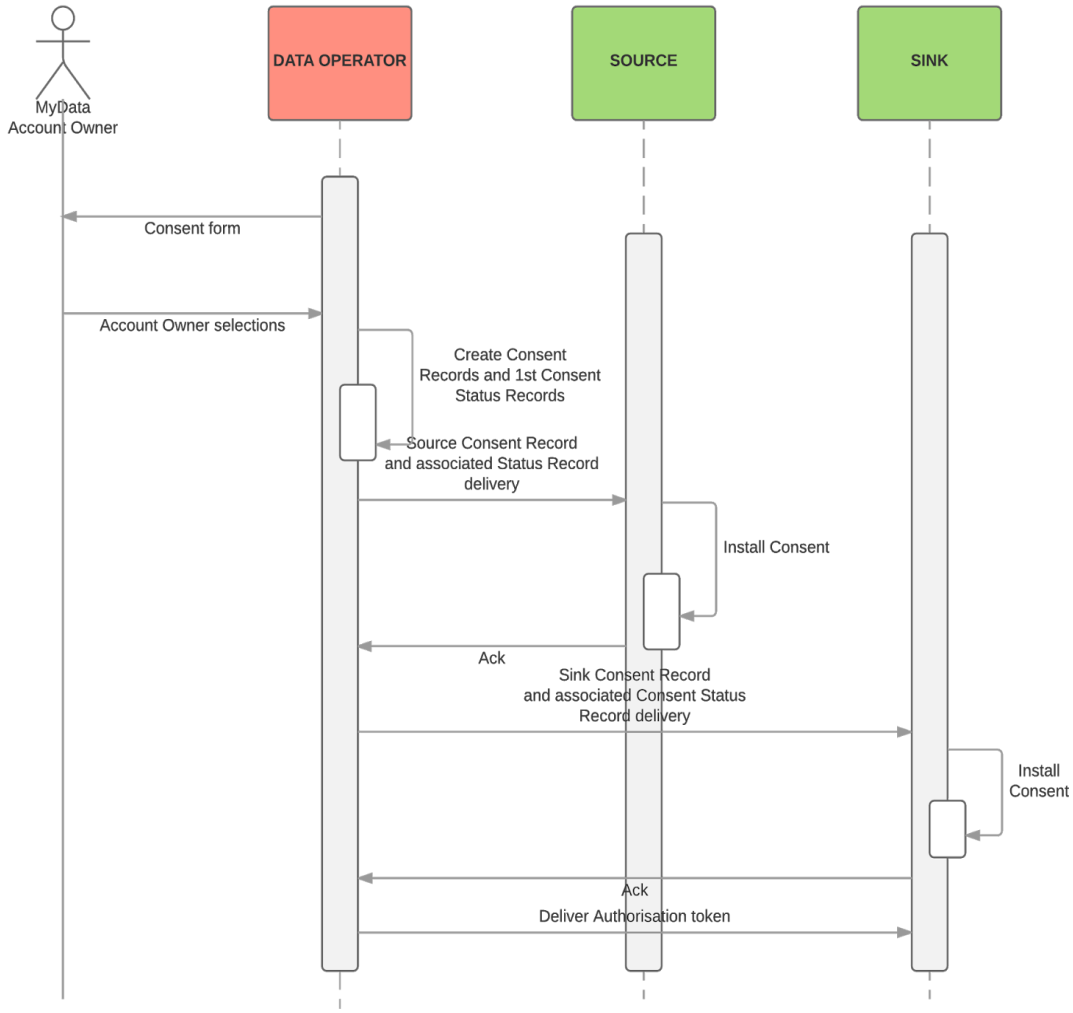

Figure 3.2: Consent creation and delivery

3.2 Account Owner Withdraws Consent

Prerequisites: Consent or pair of consents exists

Process: Account Owner withdraws previously issued consent or consent pair thus disabling further data processing.

Outcome: Consent Records are in *Withdrawn* state and further data processing based on the consent cannot legally happen.

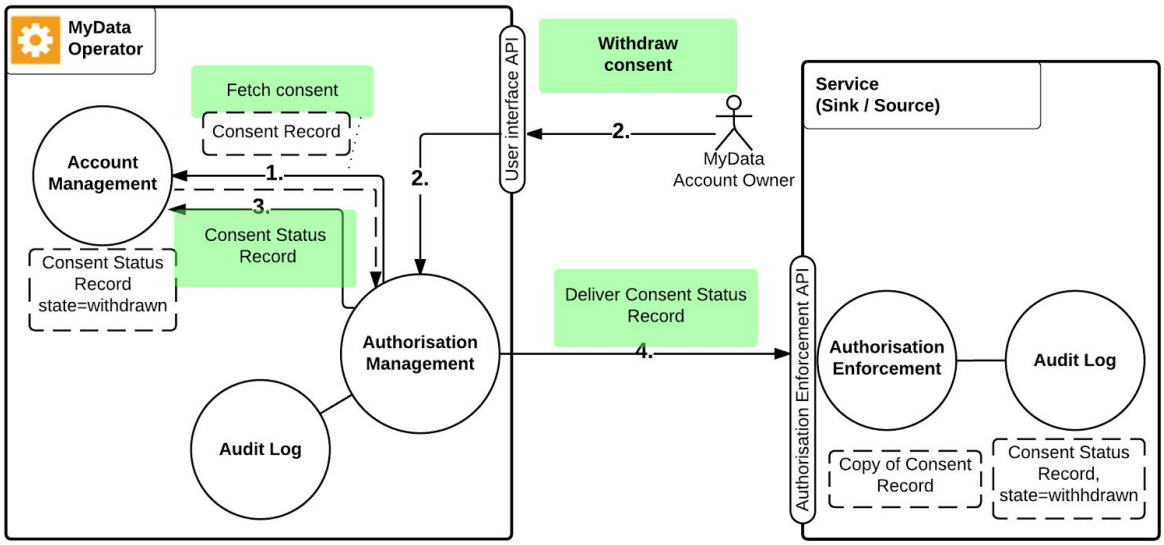

Figure 3.3: Consent withdrawal

Account Owner may withdraw the consent issued to a Sink or Source at will, at which point the affected Source and Sink MUST be informed about withdrawal as soon as possible. Upon receiving such notification from the Operator, Source MUST immediately stop processing request made by the affected Sink. For example, if Account Owner withdraws a consent, Operator notifies Source about the change and the Source will reject further requests that concern data defined in the withdrawn consent.

If a Source is processing request at the time it receives a consent withdrawal notification, it checks if the information presented in the notification concerns the request-under-processing, and stops processing the request if necessary.

Consent withdrawal process is shown in Figure 3.4.

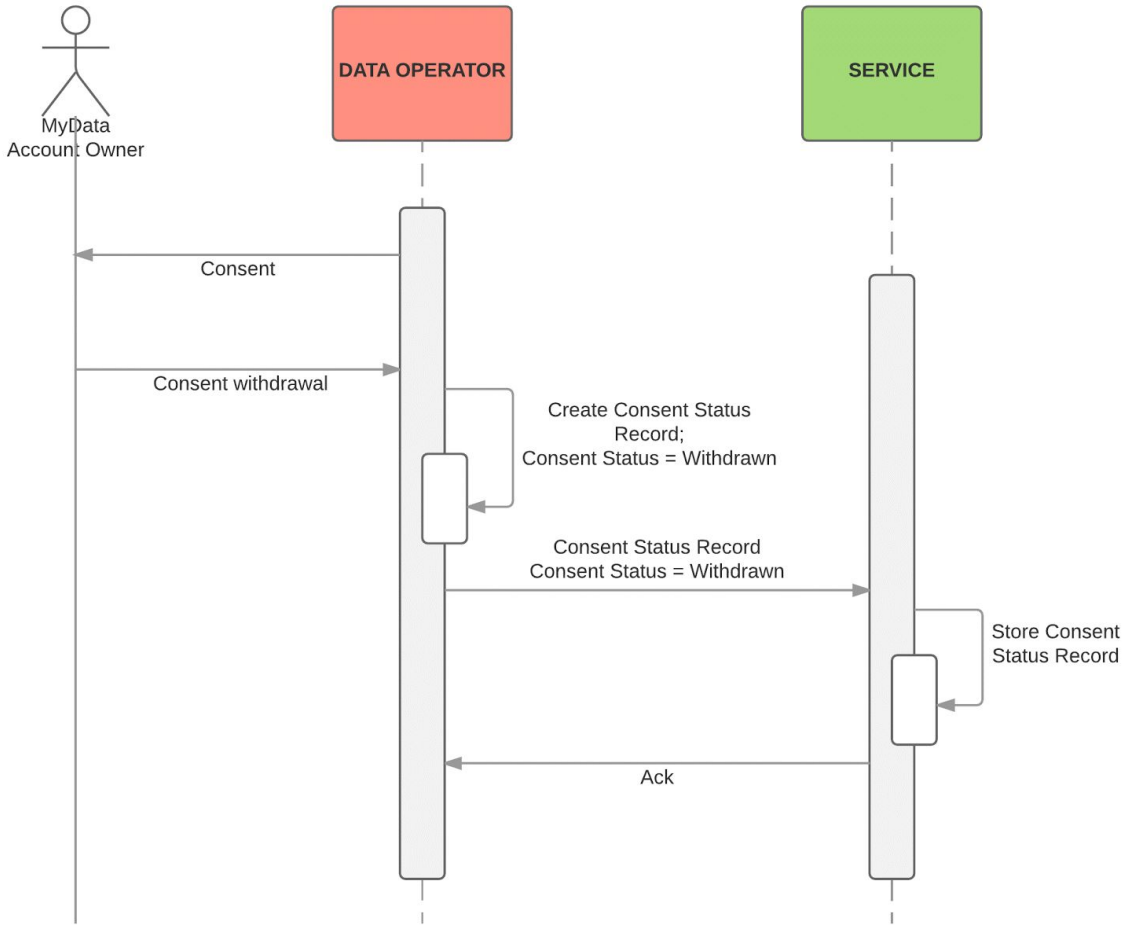

Figure 3.4: Consent withdrawal related consent status change propagation flow

3.3 Consent Status Change

Prerequisites: Consent or pair of consents exists

Process: Account Owner changes the consent status at Operator.

Outcome: Consent status is changed and data processing is either enabled or disabled based on the new status.

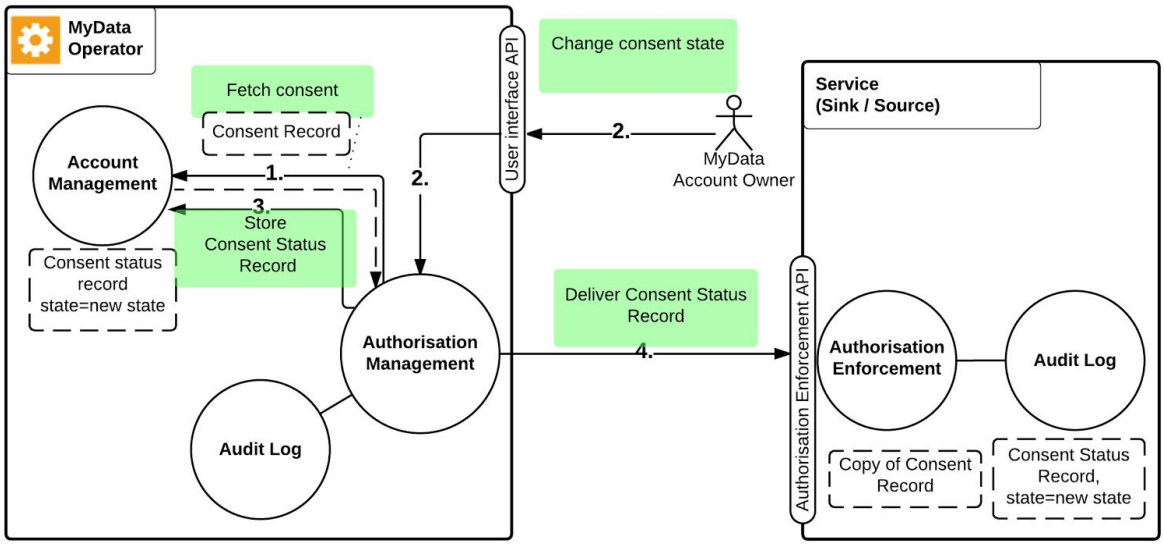

Figure 3.5: Consent status change flow

Account Owner may disable or re-activate the consent given to a Sink or Source at any time, at which point the affected Source and Sink MUST be informed about these changes as soon as possible. Source MUST immediately act and change its processing of request made by Sink, according to information presented in the notification. For example, if Account Owner disables a consent, Operator notifies Source about the change and the Source will reject further requests that concern data defined in the withdrawn consent.

If a Source is processing a request at the time it receives a consent change notification, it checks if the information presented in the notification concerns the request-under-processing, and updates its request processing if necessary.

The consent status change flow is shown in Figure 3.6.

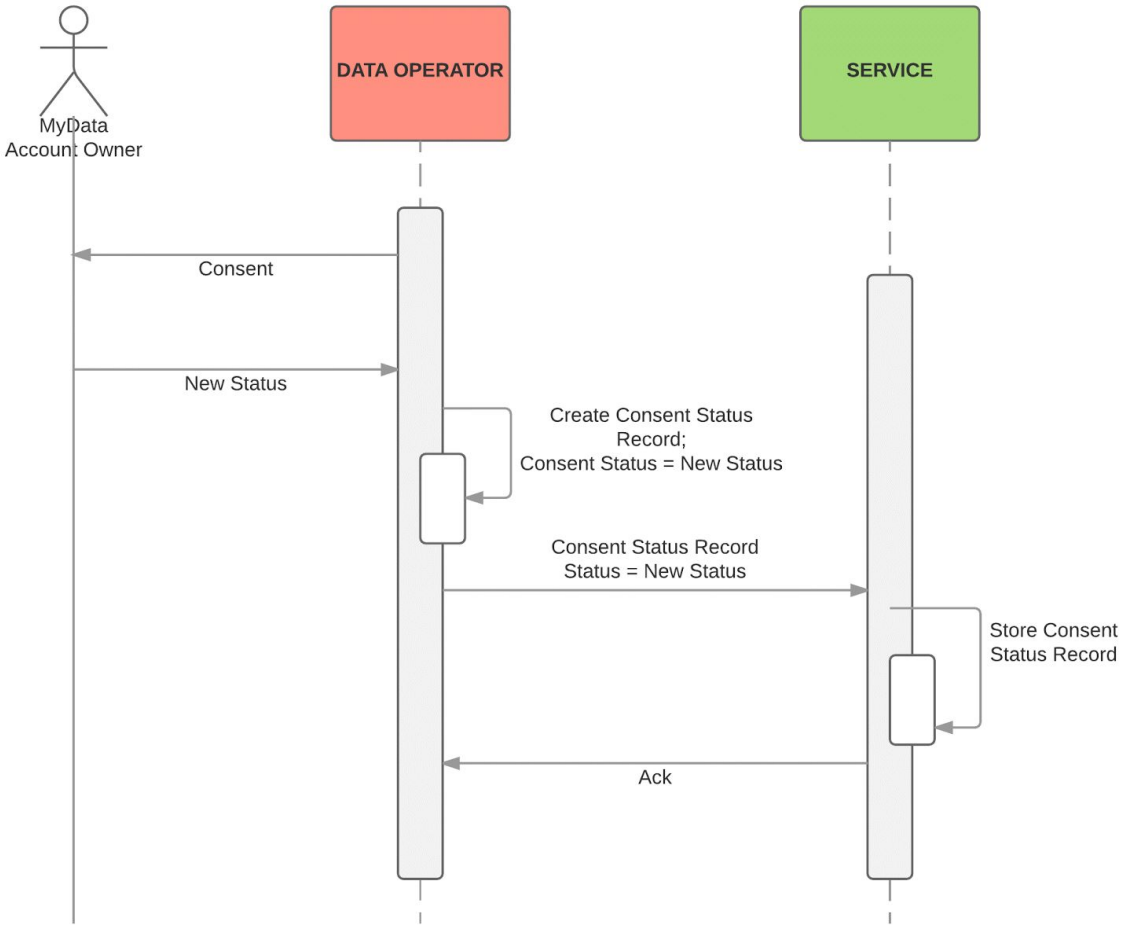

Figure 3.6: Consent status change related propagation flow

3.4 The related Service Link Status changing to Removed

Prerequisites: Consent or pair of consents exist

Process: When the related SSR is changed to *Removed* due to service unlinking or Service removal, Operator MUST set all affected consents to *Disabled* state. Consents remain in *Disabled* state until service is re-linked and the Account Owner has either re-activated or withdrawn the consent.

Outcome: All affected consents initially in *Disabled* state, and after acceptance in state set by the Account Owner.

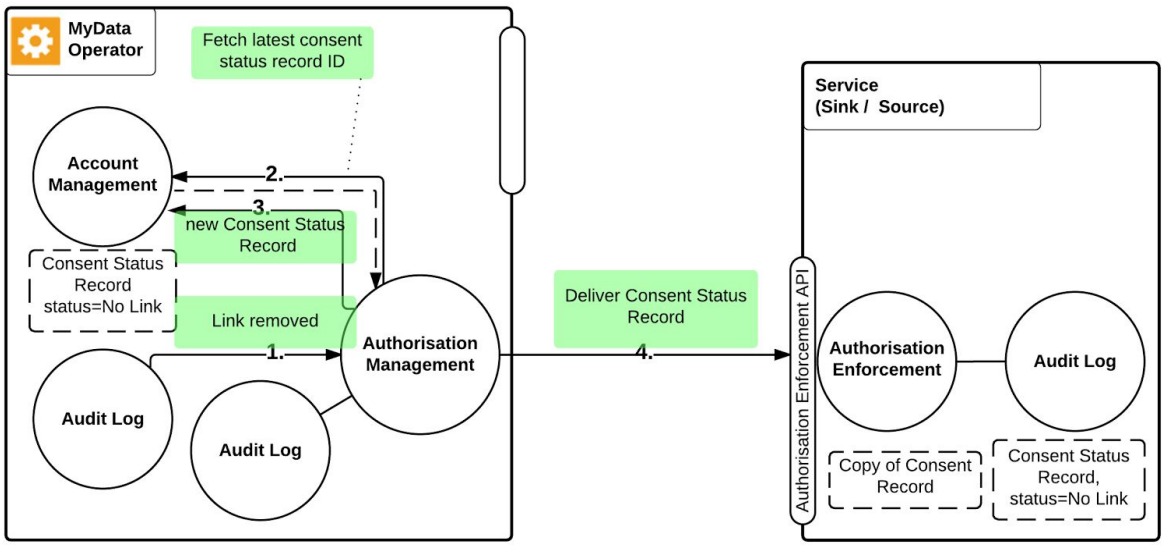

Figure 3.7: Link removal related flow

Service can be unlinked or removed from the Service Registry at any time, at which point the affected Source and Sink MUST be informed as soon as possible.

Upon receiving such notification from the Operator, Source MUST immediately stop processing request made by Sink. If a Source is processing request at the time it receives a service unlinking notification, it checks if the information presented in the notification concerns the request-under-processing and stops its request processing if necessary.

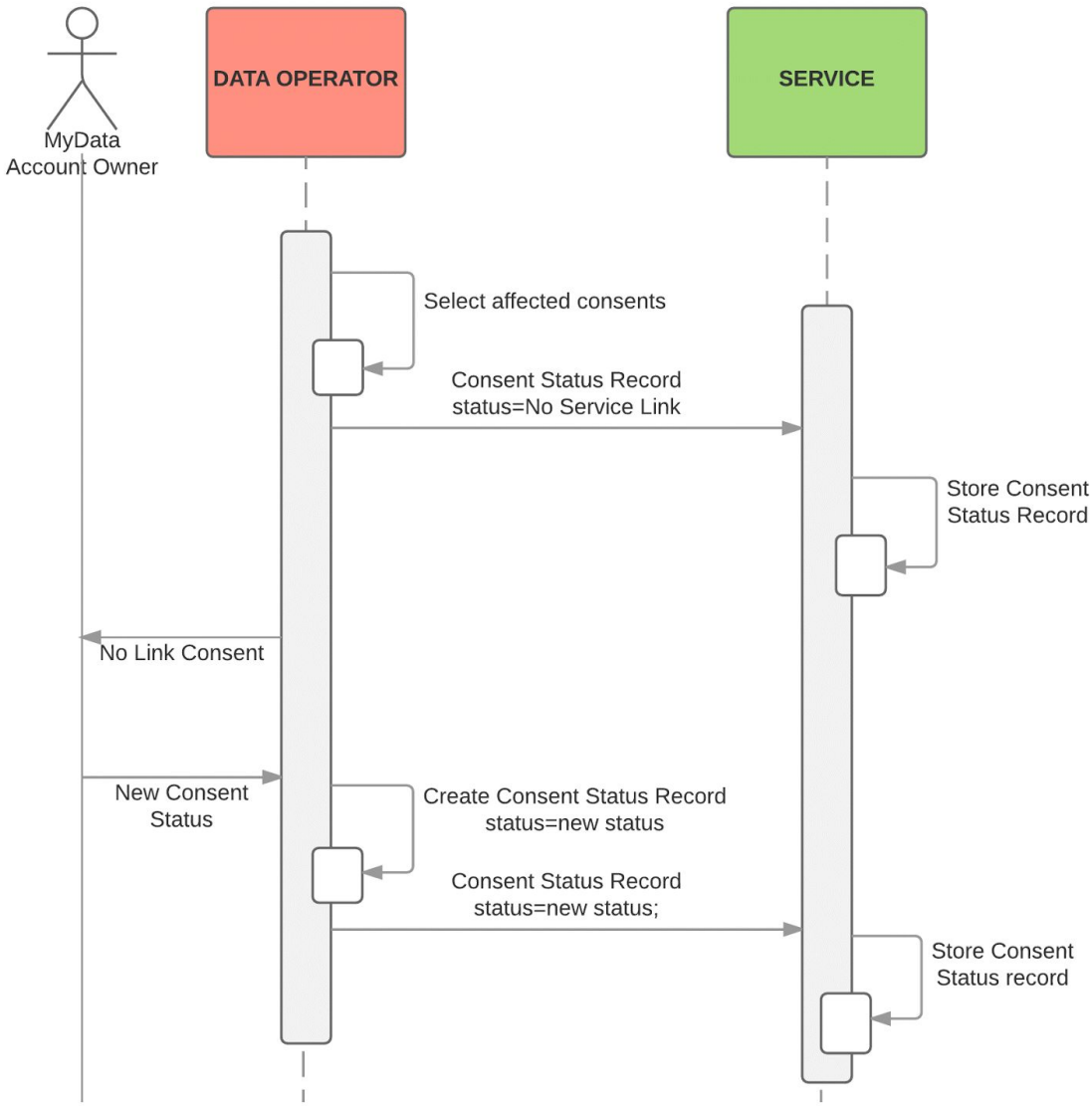

Figure 3.8: Service removal related consent status change propagation flow. This is executed for each affected consent.

4. MyData Consent Record

This section presents the structure of MyData Consent Record and Consent Status Record.

4.1. Resource Set Description

Table 4.1 Resource Set ID

rs_id	Source URI || Resource Key Source URI: URI identifying Source Resource Key: Unique key within a service identifying specific resource set

Table 4.2 Resource Set

rs_id	Resource Set ID
dataset[1..n]	dataset_id: Data Set ID from Service Data Description in Service Registry distribution_id: Distribution ID from Data Set in Service Data Description distribution_url: Host || Service Access URI || Distribution URI Host: Source's Domain from Service Instance Service Access URI: Source's Access Endpoint URI from Service Instance Distribution URI: Access URL from Distribution in Data Set

```
{
  "resource_set": {
    "rs_id": String,
    "dataset": [
```

```
      {
        "dataset_id": String,
        "distribution_id": String,
        "distribution_url": String
      }
    ]
  }
}
```

4.2 MyData Consent Record Details

MyData Consent Record consists of four parts: common part, role specific part, Kantara Initiative's Consent Receipt Mode 2 [KI-CR] part and potential extension part.

This version of architecture supports including Kantara's CR as part of the Consent Records, but use of it's contents in the architecture is not yet validated.

Table 4.2 presents a detailed structure of MyData Consent Record.

Table 4.3: Consent Record

KEY	**TYPE**	**DESCRIPTION**
COMMON PART		
version	String	Specification version number. For this release MUST be 1.2.1
cr_id	String	unique ID for the Consent Record
surrogate_id	String	the Account Owner's surrogate ID (see Service Linking document)
rs_description	Object	Resource set description
slr_id	String	Service Link Record ID
iat	Integer	Time when the consent was issued
nbf	Integer	Time: consent not valid before, OPTIONAL
exp	Integer	Time: consent not valid after, OPTIONAL
operator	String	ID of the Operator where the consent was created
subject_id	String	ID of the service to which the consent was issued
role	String	Sink **OR** Source
ROLE SPECIFIC PART		

pop_key	JWK	[SOURCE ONLY]: Public key[1] of the token user (Sink) as JWK. Used to verify the data request. JWK structure MUST contain 'kid' parameter.
token_issuer_key	JWK	[SOURCE ONLY]: Public key of the token issuer (Operator). Used by Source to verify the authorisation token. JWK structure MUST contain 'kid' parameter.
source_cr_id	String	[SINK ONLY] Identifies resource set on Source
usage_rules[1..n]	Array of Strings	[SINK ONLY] How the data can be processed, in machine readable form String [1..*]
Consent Receipt PART		
ki_cr	Object	Kantara CR, see [KI-CR]
EXTENSION PART		
extensions	Object	NONE FOR THIS RELEASE

Consent Record MUST be signed with the account owner's private key as defined in [RFC7515].

The JWS header MUST contain 'kid' field identifying account owner's key pair used to sign the Consent Record.

4.2.1 Source's Consent Record payload

```
{
  "common_part": {
    "version": "String",
    "cr_id": "String",
    "surrogate_id": "String",
    "rs_description": {
      "resource_set": {
        "rs_id": "String",
        "dataset": [
          {
            "dataset_id": "String",
            "distribution_id": "String",
            "distribution_url": "String"
          }
        ]
      }
    },
    "slr_id": "String",
    "iat": "Integer",
    "nbf": "Integer",
    "exp": "Integer",
    "operator": "String",
    "subject_id": "String",
```

[1] The Proof of Possession key received by the Operator at Sink's Service Linking, stored by the Operator since.

```
    "role": "String: Source"
  },
  "role_specific_part": {
    "pop_key": {
      "jwk": "JSON Web Key (JWK) presentation of public part of Proof-of-Possession Key"
    },
    "token_issuer_key": {
      "jwk": "JSON Web Key (JWK) presentation of public part of token issuer key"
    }
  },
  "consent_receipt_part": {
    "ki_cr": {}
  },
  "extension_part": {
    "extensions": {}
  }
}
```

4.2.2 Sink's Consent Record payload

```
{
  "common_part": {
    "version": "String",
    "cr_id": "String",
    "surrogate_id": "String",
    "rs_description": {
      "resource_set": {
        "rs_id": "String",
        "dataset": [
          {
            "dataset_id": "String",
            "distribution_id": "String",
            "distribution_url": "String"
          }
        ]
      }
    },
    "slr_id": "String",
    "iat": "Integer",
    "nbf": "Integer",
    "exp": "Integer",
    "operator": "String",
    "subject_id": "String",
    "role": "String: Sink"
  },
  "role_specific_part": {
    "usage_rules": [
      "String"
    ],
    "source_cr_id": "String"
  },
  "consent_receipt_part": {
    "ki_cr": {}
  },
  "extension_part": {
    "extensions": {}
  }
}
```

4.3 MyData Consent Status Record

Table 4.5 presents the structure of the MyData Consent Status Record.

Table 4.5 Consent Status Record

KEY	TYPE	DESCRIPTION
version	String	Version number of Service Link Status Record specification. For this release MUST be 1.2
record_id	String	Unique ID of the record
surrogate_id	String	the Account owner's Surrogate ID
cr_id	String	Unique ID of the consent
consent_status	String	Active/Disabled/Withdrawn
iat	Integer	Time when the Status Record was issued
prev_record_id	String	Link to previous Status Record ID, NULL if first Status Record

Consent Status Record MUST be signed with the account owner's private key as defined in [RFC7515].

The JWS header MUST contain 'kid' field identifying account owner's key pair used to sign the Consent Status Record.

4.3.1 Consent Status Record payload

```
{
  "version": "String",
  "record_id": "String",
  "surrogate_id: "String",
  "cr_id": "String",
  "consent_status": "String",
  "iat": "Integer",
  "prev_record_id": "String"
}
```

5. Consent APIs

Authorisation 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. An interactive example will be available.

5.1 Interfaces of Different Actors

There are two main interfaces: Authorisation Management API (endpoint provided by the Operator) and Authorisation Enforcement API (to be provided by Source and Sink each)

Operator's Authorisation Management API defines methods for Sink and Source to request Consent Records (referenced using surrogate ID) and Consent Status Records (referenced using unique Consent ID and last known Status Record ID) and to verify that the service has the latest Consent Status Record.

Both Source and Sink MUST provide an Authorisation Enforcement API for Operator, through which it can deliver Consent Records and Consent Status Records.

5.2 API Specification

Authorisation management and enforcement APIs provided by the MyData SDK are documented in YAML format:
[Authorisation-Management]
[Operator-CR]

5.3 Detailed Flow

Below three figures 5.1-5.3 show details of the service link flows as currently implemented in MyData SDK.

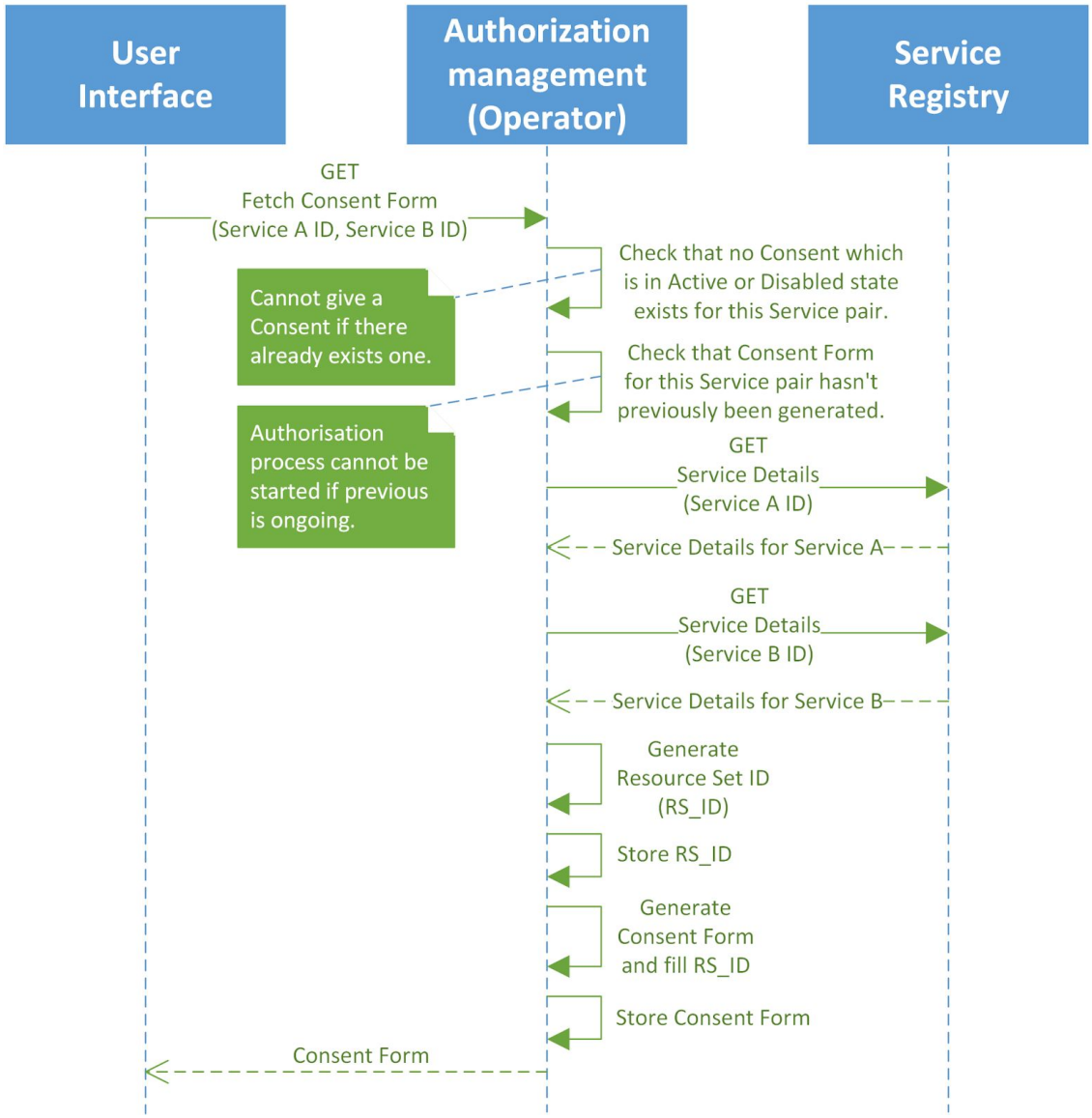

Figure 5.1: Consent form

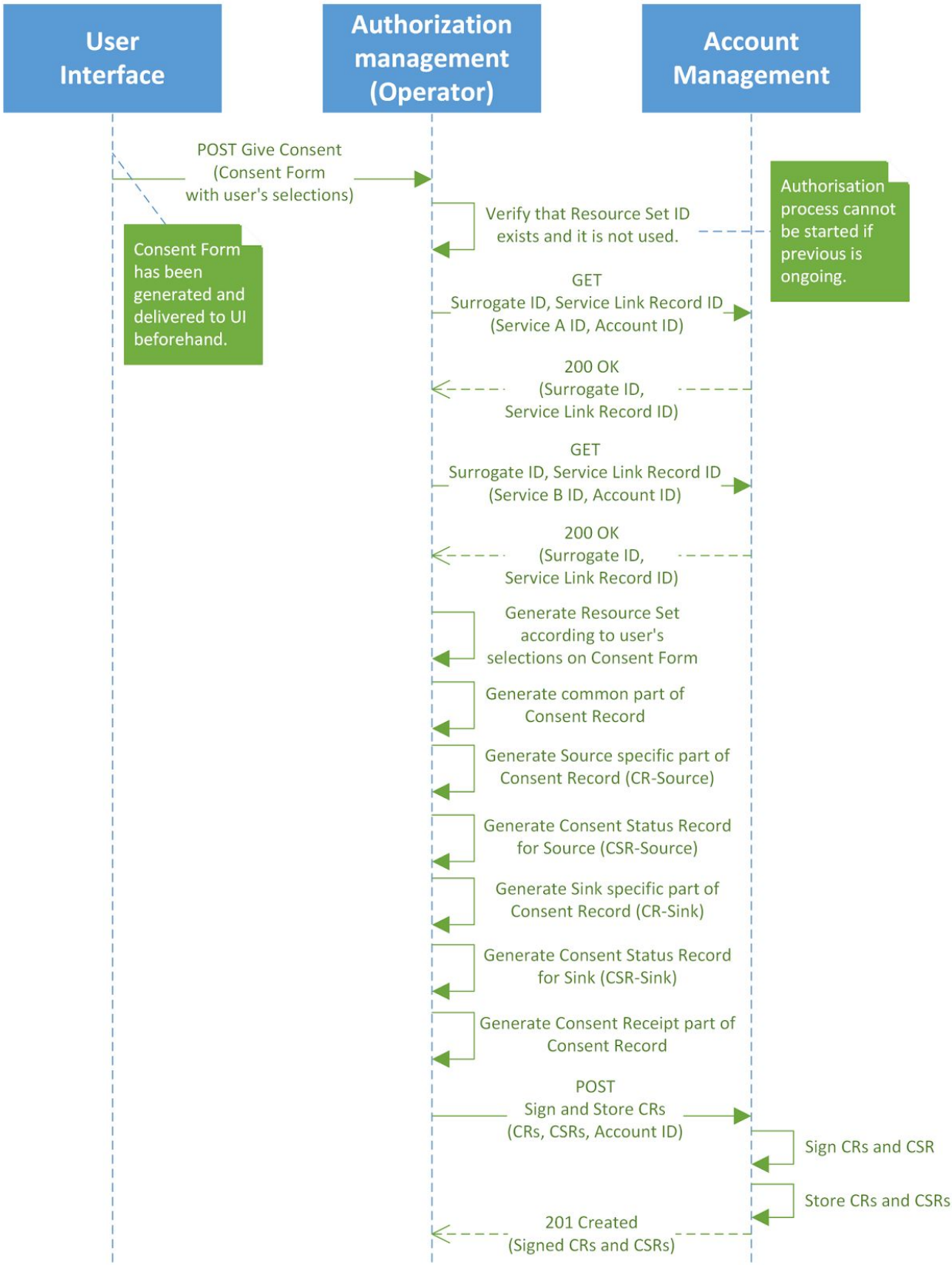

Figure 5.2: Create Consent

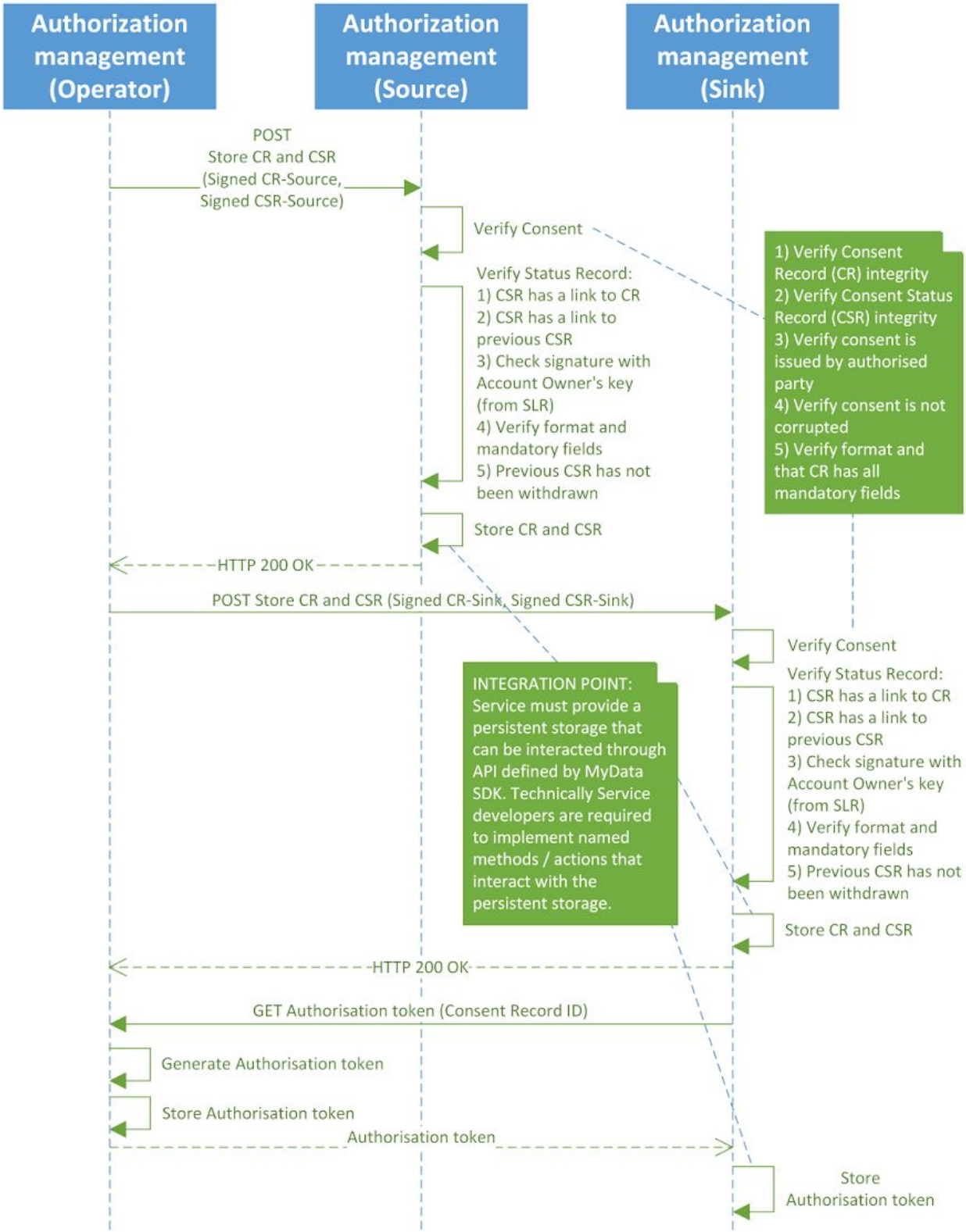

Figure 5.3: Deliver Consent

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", RFC 7515, May 2015
[RFC7519] Jones, M., Bradley, J., Sakimura, N. "JSON Web Token (JWT)", RFC 7519, May 2015
[KI-CR]: KI-CR09_2-DRAFT-2016-10-19-clean.doc
[Authorisation-Management]: http://editor.swagger.io/#/?import=https://raw.githubusercontent.com/HIIT/mydata-sdk/master/Service_Components/doc/api/swagger_Authorization_Management.yml
[Operator-CR]: http://editor.swagger.io/#/?import=https://raw.githubusercontent.com/HIIT/mydata-sdk/master/Operator_Components/doc/api/swagger_Operator_CR.yml