MyData Architecture Framework

Consenting Specification

v. 2.0

MyData Architecture Framework defines the operations and APIs between the Operational Roles (Operator, Service, 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 Consenting and consent record management over their lifetime.

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

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 Framework release 2.0 available at
https://github.com/mydata-sdk/mydata-docs/tree/master/architecture_specs.

Consent [lawful basis] as defined in GDPR 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. 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. With regard to specific categories of personal data, automated decisions, including profiling, and transfers of personal data to third countries (i.e. outside the EU) in certain circumstances 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 Service to a specific Sink Service, 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 digital-form 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 and in the service.

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 2.0:

- Updated flow diagrams
- Added Consent Record for Consenting with a single Service, formerly uncovered use case

1.4 Formats

In MyData Architecture 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. Dynamic Consent for MyData

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

2.1 Consent Record

When Account Owner issues a consent, it is documented in a ***Consent Record (CR)***. A Consent Record is a manifestation of legally valid consent that contains unambiguous data processing policies accepted by the Account Owner at the point of 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 policing data processing within a service, the Account Owner creates a single CR for the related service.

When authorising data transfer from a specific Source to a specific Sink, the Operator 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 is authorised to be accessed. The Sink's CR can also include the permissions for data processing.

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's service description contains a description of all data a service can provision or process. Account Owner creates (through the Operator implementation) 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.

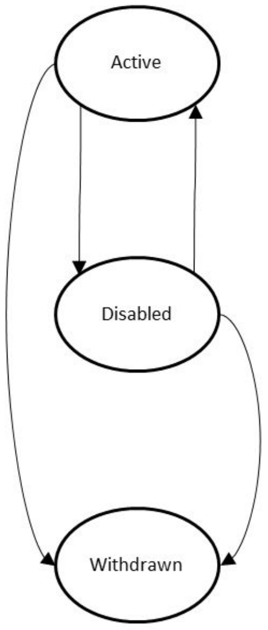


Figure 2.1: Consent lifecycle

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.

Services 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. Services 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 Consenting Transactions

This section describes consenting specific transactions needed to fulfill the use cases requiring account owner's consent. 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.

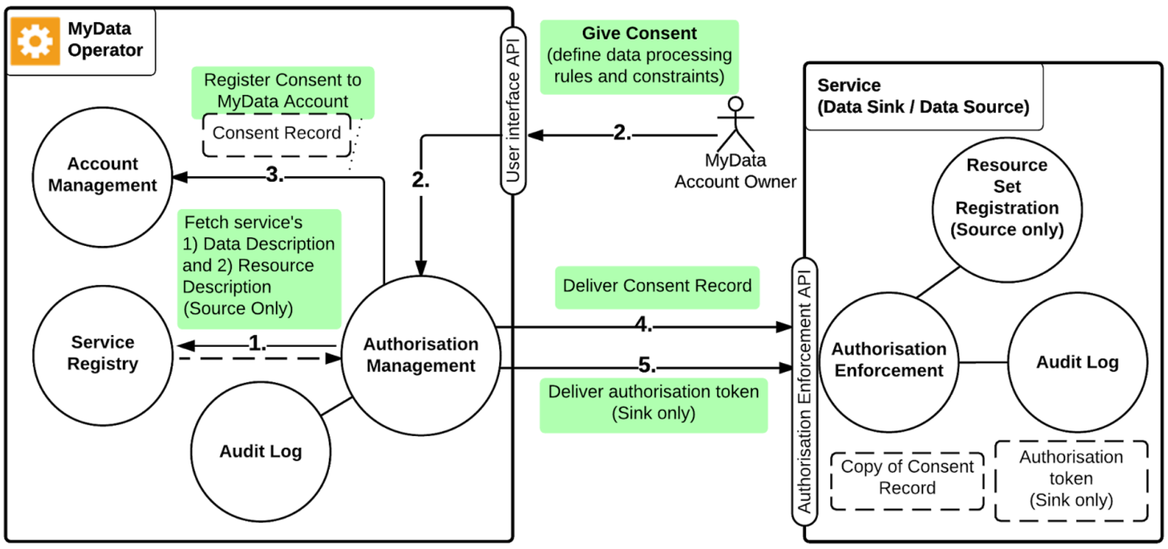
Service descriptions contains the minimum requirements services have for particular types of consents. It is up to the operator to enforce that the created consents meet the minimum requirements.

3.1 Account Owner Issues Consent

Prerequisites: There is a s´Service Link between Account Owner and service(s)

Process: Account Owner issues consent (for data processing within a service) or pair of consents (for data transfer 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. In case of the 3rd party re-use the 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. https://github.com/mydata-sdk



MyData Operator
Register Consent to MyData Account
Consent Record
Account Management
3.
2.
Fetch service's 1) Data Description and 2) Resource Description (Source Only)
Service Registry
1.
Authorisation Management
Audit Log
User interface API
Give Consent (define data processing rules and constraints)
2.
MyData Account Owner
Deliver Consent Record
4.
5.
Deliver authorisation token (Sink only)
Authorisation Enforcement API
Service (Data Sink / Data Source)
Resource Set Registration (Source only)
Authorisation Enforcement
Audit Log
Copy of Consent Record
Authorisation token (Sink only)

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

3.1.1 Consenting

After a service has been linked at MyData Account, the service can be authorised to process data by conducting Consenting step. The consenting results in a Consent Record where the Account Owner gives an explicit consent for the service to process specified data according to terms defined in the Consent Record. The contents of a Consent Record is depicted in *section 4.2*.

Upon creation of Consent Record, Operator delivers the record to the corresponding service as shown in *Figure 2*.

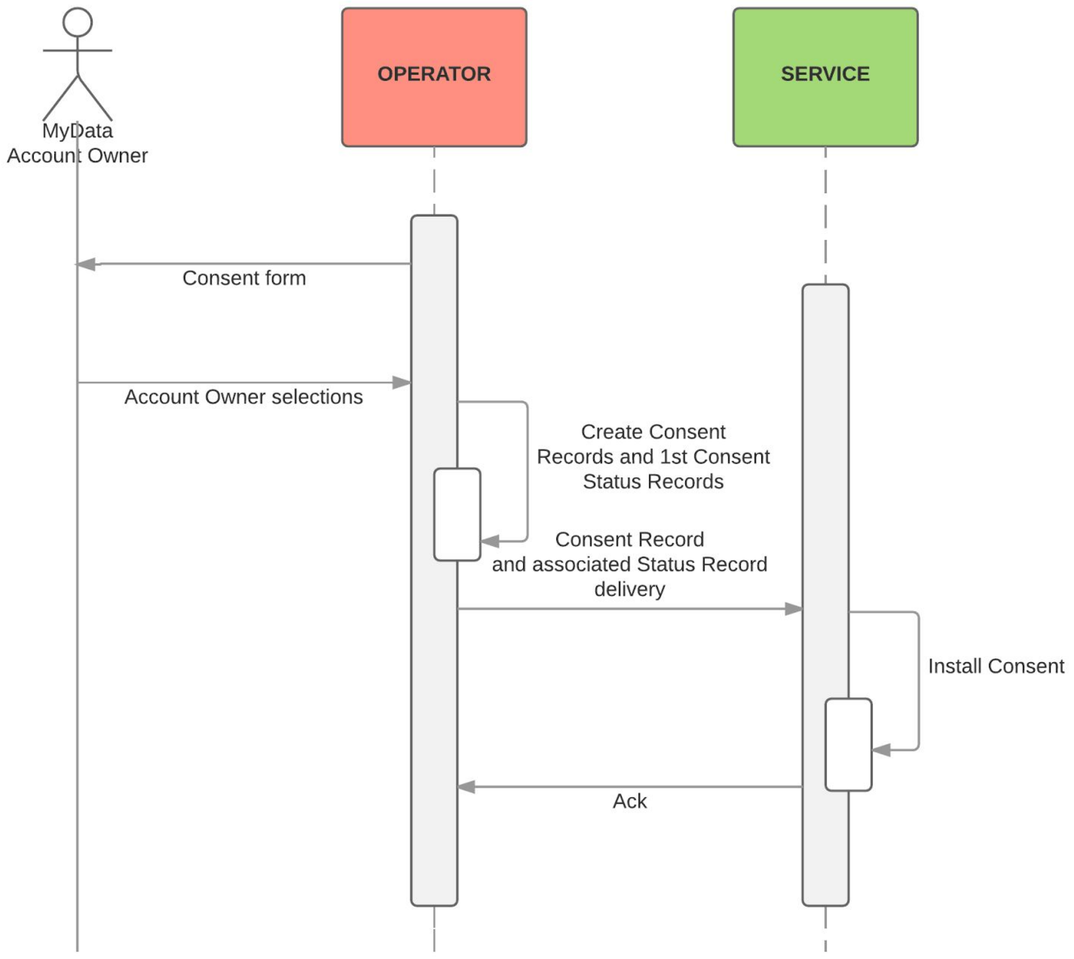

MyData
Account Owner
OPERATOR
SERVICE
Consent form
Account Owner selections
Create Consent Records and 1st Consent Status Records
Consent Record and associated Status Record delivery
Install Consent
Ack

Figure 3.2: Consent creation and delivery in consenting case

3.1.2 Third party re-use

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 a two-party Consenting step. The step 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 4.3*. 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 3* (Sink's Record is delivered to Sink and Source's Record is delivered to Source).

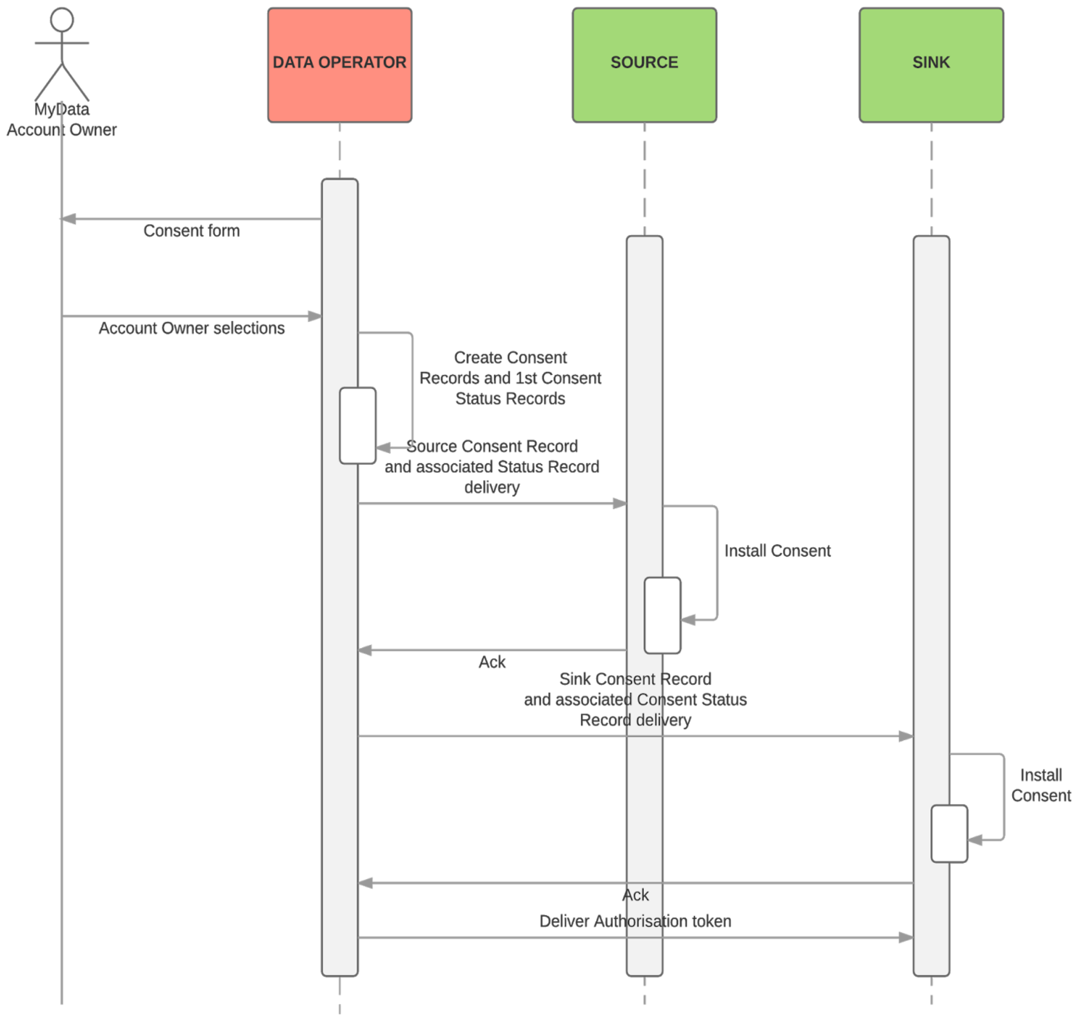

DATA OPERATOR
SOURCE
SINK
MyData Account Owner
Consent form
Account Owner selections
Create Consent Records and 1st Consent Status Records
Source Consent Record and associated Status Record delivery
Install Consent
Ack
Sink Consent Record and associated Consent Status Record delivery
Install Consent
Ack
Deliver Authorisation token

Figure 3.3: Consent creation and delivery in 3rd party re-use case

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.

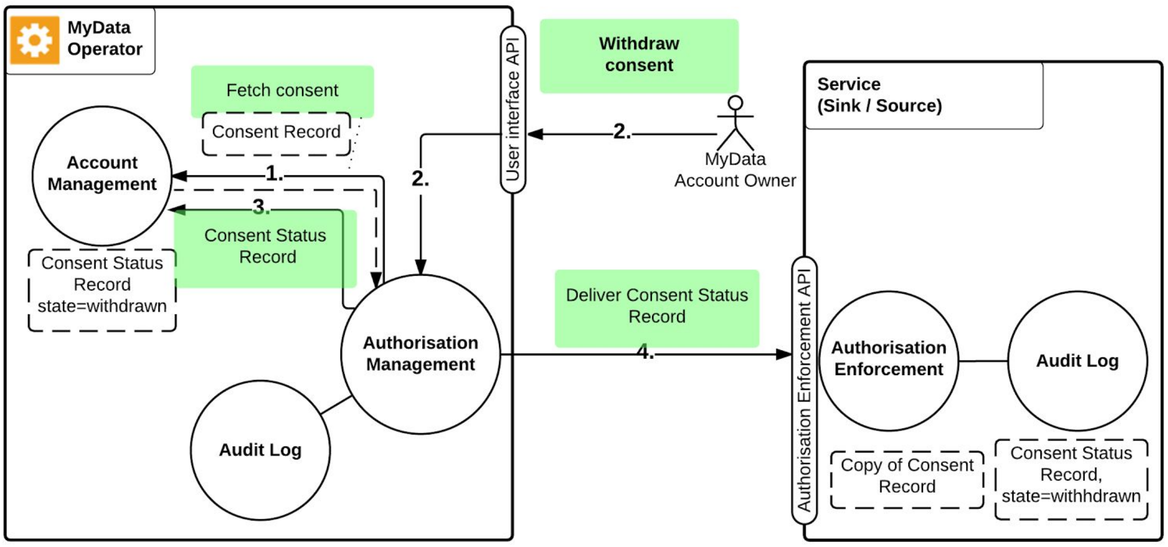


Figure 3.4: Consent withdrawal

3.2.1 Consenting case

Account Owner may withdraw the consent issued to a service at will, at which point the affected service MUST be informed about withdrawal as soon as possible. Upon receiving such notification from the Operator, service MUST immediately stop processing.

Consent withdrawal process for both consenting and 3rd party re-use cases is shown in Figure 3.5.

3.2.2 3rd party re-use case

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 3.5.

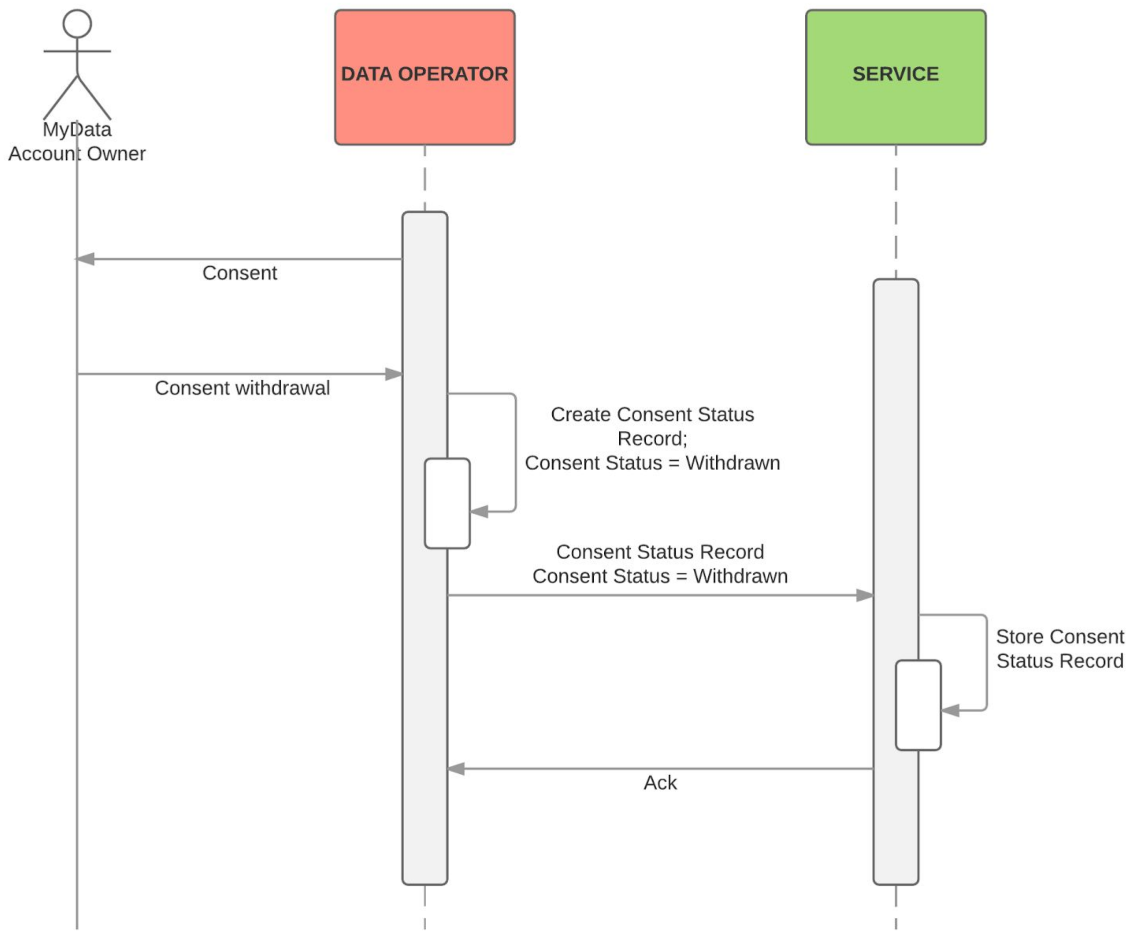

MyData Account Owner
DATA OPERATOR
SERVICE
Consent
Consent withdrawal
Create Consent Status Record; Consent Status = Withdrawn
Consent Status Record Consent Status = Withdrawn
Store Consent Status Record
Ack

Figure 3.5: 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.

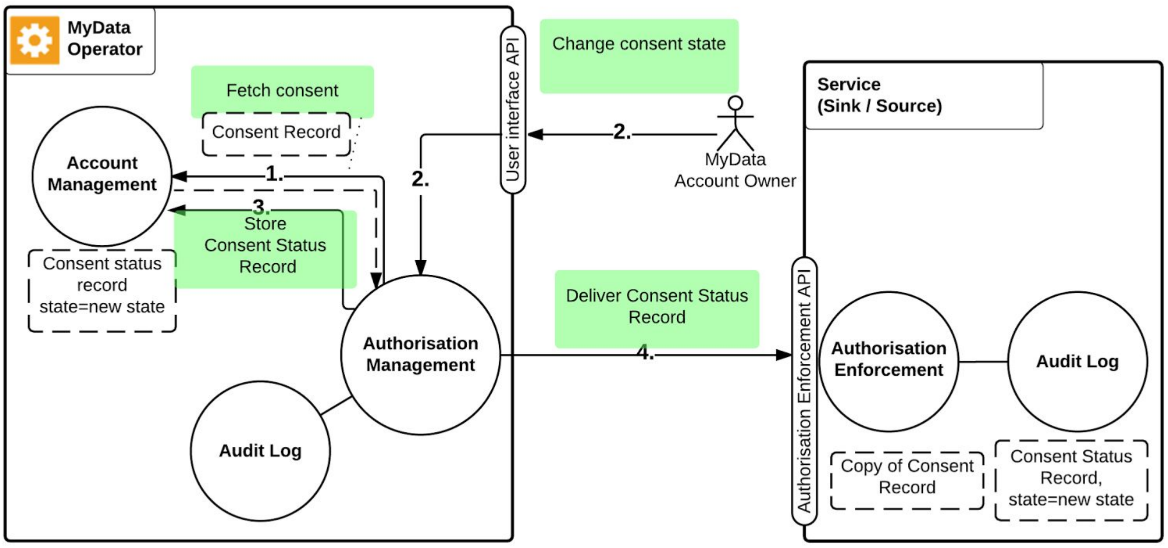


Figure 3.6: Consent status change flow

3.3.1 Consenting case

Account Owner may disable or re-activate the consent given to a service any time, at which point the affected service MUST be informed about these changes as soon as possible. Service MUST immediately act and change its data processing according to information presented in the notification. For example, if Account Owner disables a consent, Operator notifies the service about the change and the service will stop data processing.

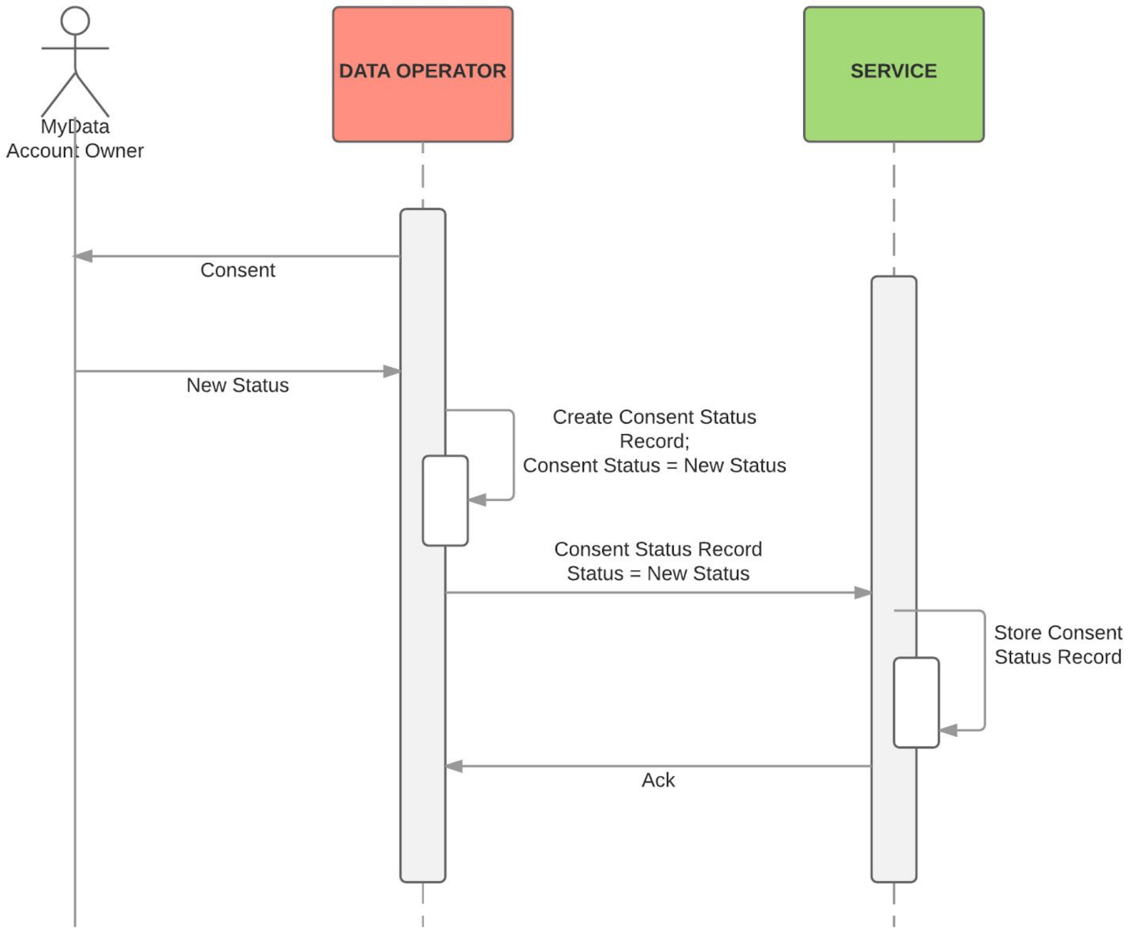
The consent status change flow for both consenting and 3rd party re-use cases is shown in Figure 3.7.

3.3.2 3rd party re-use case

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 3.7.



MyData Account Owner
DATA OPERATOR
SERVICE
Consent
New Status
Create Consent Status Record; Consent Status = New Status
Consent Status Record Status = New Status
Store Consent Status Record
Ack

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 *withdrawn* state. If the service is re-linked the Account Owner has to give new consents.

Outcome: All affected consents in *Withdrawn* stater.

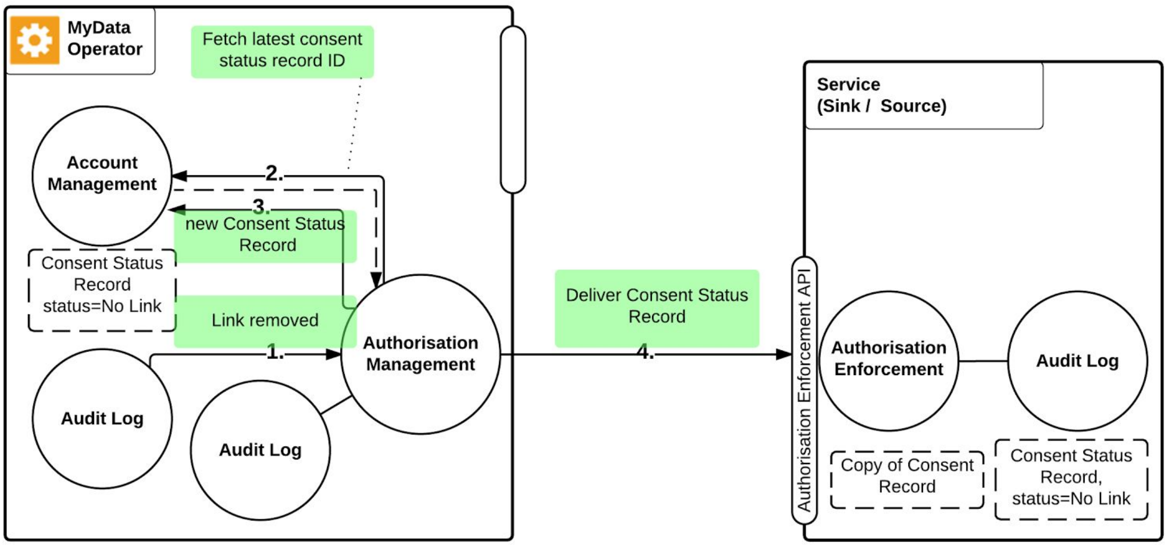

MyData Operator
Fetch latest consent status record ID
Account Management
2.
3.
new Consent Status Record
Consent Status Record status=No Link
Link removed
1.
Authorisation Management
Audit Log
Audit Log
Deliver Consent Status Record
4.
Authorisation Enforcement API
Service (Sink / Source)
Authorisation Enforcement
Audit Log
Copy of Consent Record
Consent Status Record, status=No Link

Figure 3.8: Link removal related flow

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

In 3rd party re-use case Source MUST immediately stop processing request made by Sink upon receiving such notification from the Operator. 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.

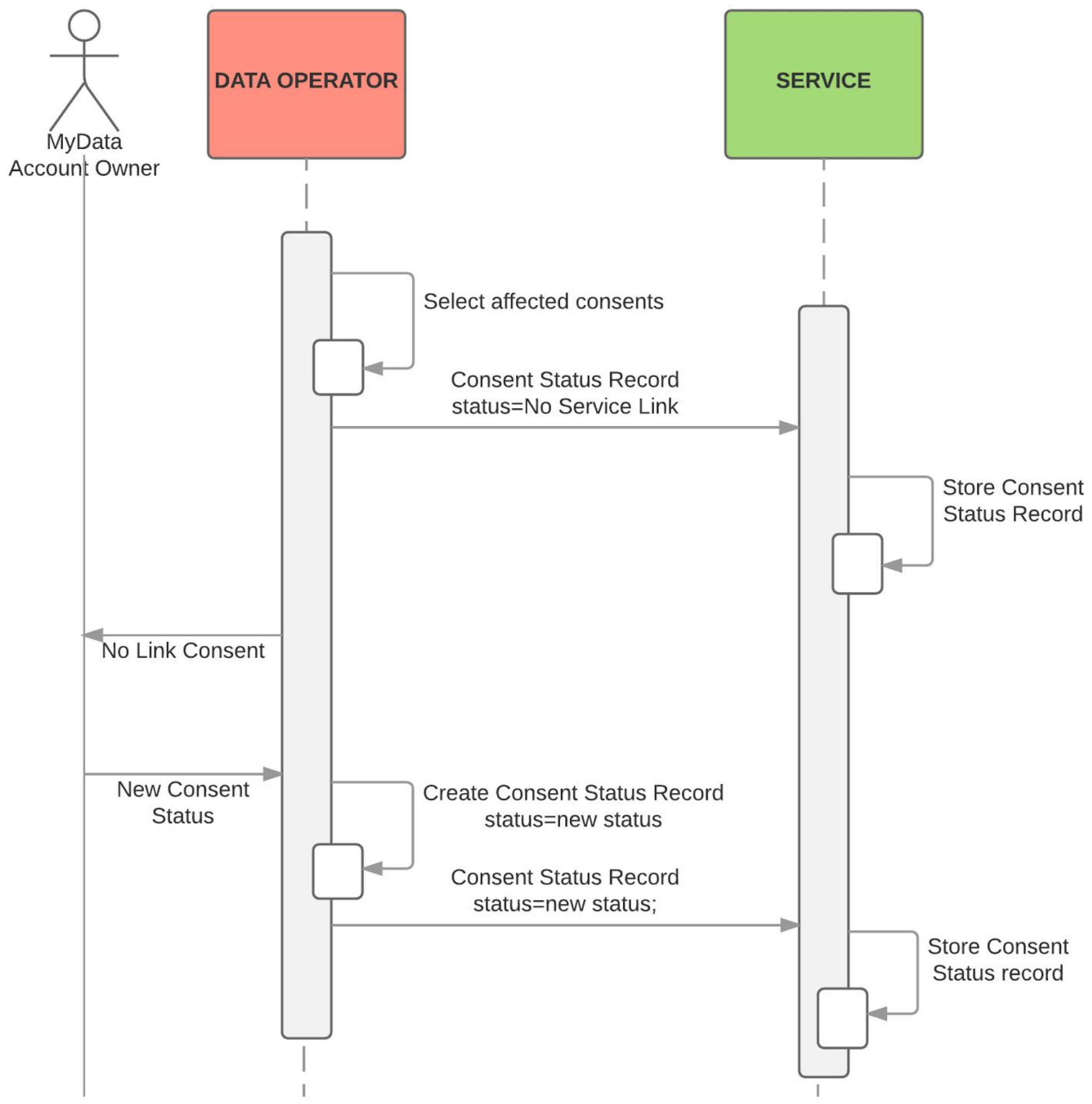


Figure 3.9: Service removal related consent status change propagation flow. This is executed for each affected consent.

4. Consent Records

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

4.1. Resource Set Description

Concepts of Resource Set and its identifying Resource Set ID were introduced in section 2.2 above. Tables 4.1 and 4.2 explain their structure and relationship with data descriptions defined in Service Description document.

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	Dataset ID (datasetId) from Service's Data Description		
	distribution_id	Distribution ID (distributionId) from Dataset's distribution inside Service's Data Description object. **Not used in Consenting CR**		
	distribution_url	Host || Service Access URI || Distribution URI **Not used in Consenting CR**		
			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
      }
    ]
  }
}
```
Figure 4.1 Sample of a resource set structure in JSON (Source offering resources for 3rd party re-use)

4.2 Consent Record for Consenting a Single Service

Table 4.4 presents a detailed structure of MyData Consent Record for Consenting case.

Table 4.4: Consent Record for consenting case

KEY	**TYPE**	**DESCRIPTION**
version	String	Specification version number. For this release MUST be 2.0
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
service_description_version	String	Service description version used when consent was issued
consent_proposal	Object	<table><tr><th>KEY</th><th>DESCRIPTION</th></tr><tr><td>url</td><td>URL pointing to consent proposal (i.e. what was shown to user)</td></tr><tr><td>hash</td><td>SHA-256 hash of consent proposal</td></tr></table>
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
usage_rules[1..n]	Array of Objects	

KEY	**Description**
purposeId	ID of the purpose
datasets	Array of dataset IDs

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 Status Record

4.2.1 Consent Record payload in Consenting a single service

```
{
  "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",
  "usage_rules": [
    String
  ]
}
```

4.3 Consent Record for 3rd party re-use

Table 4.2 presents a detailed structure of Consent Record for use in 3rd party re-use scenario.

Table 4.4: Consent Record for 3rd party re-use

KEY	TYPE	DESCRIPTION
COMMON PART		
version	String	Specification version number. For this release MUST be 2.0
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
service_description_version	String	Service description version used when consent was issued
consent_proposal	Object	**KEY** / **DESCRIPTION** url / URL pointing to consent proposal (i.e. what was shown to user) hash / SHA-256 hash of consent proposal
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.

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

source_cr_id	String	[SINK ONLY] Identifies resource set on Source
usage_rules[1..n]	Array of Objects	

KEY	**Description**
purposeId	ID of the purpose
datasets	Array of dataset IDs

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.

4.3.1 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,
    "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"
    }
  }
}
```

4.3.2 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
  }
}
```

4.4 Consent Status Record

Table 4.5 presents the structure of the 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 2.0
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.

4.4.1 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

Consenting has been implemented as a reference service in the MyData SDK. Related developer API documentation and detailed flow diagrams clarifying the implementation are linked and documented in this section. An interactive example about 3rd party re-use scenario 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.

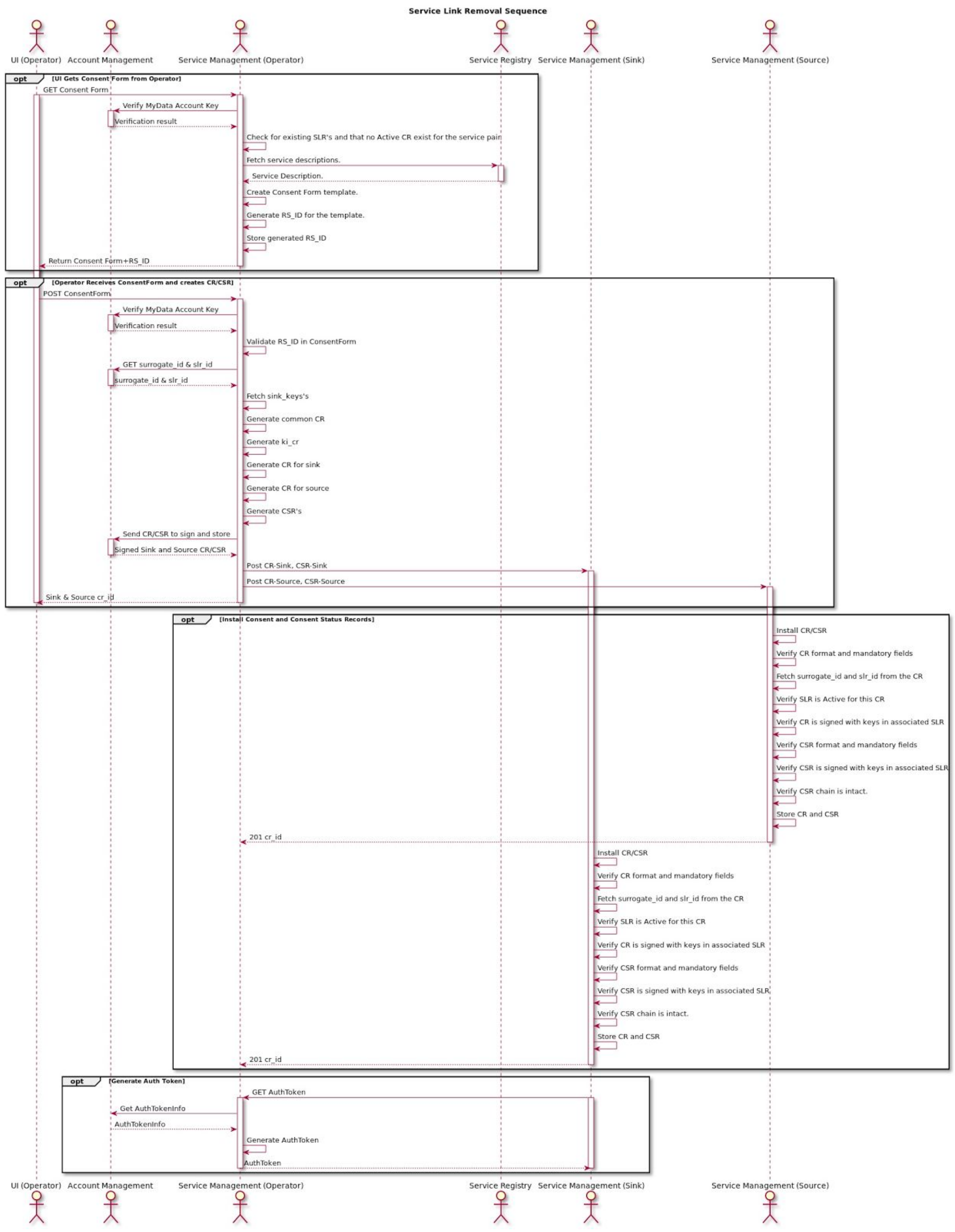
Services no matter their role (plain Service, Source or Sink) MUST provide an Authorisation Enforcement API for Operator, through which it can deliver Consent Records and Consent Status Records.

5.2 API Specification

Consent management APIs provided by the MyData SDK are documented in YAML format:
https://github.com/mydata-sdk/mydata-docs/tree/master/api_specs

5.3 Detailed Flow

Below three figures 5.1-5.3 show details of the service link flows as currently implemented in MyData SDK. Flow diagrams are available at: https://github.com/mydata-sdk/mydata-docs/tree/master/flow_diag


Service Link Removal Sequence
UI (Operator)
Account Management
Service Management (Operator)
Service Registry
Service Management (Sink)
Service Management (Source)
opt
[UI Gets Consent Form from Operator]
GET Consent Form
Verify MyData Account Key
Verification result
Check for existing SLR's and that no Active CR exist for the service pair
Fetch service descriptions.
Service Description.
Create Consent Form template.
Generate RS_ID for the template.
Store generated RS_ID
Return Consent Form+RS_ID
opt
[Operator Receives ConsentForm and creates CR/CSR]
POST ConsentForm
Verify MyData Account Key
Verification result
Validate RS_ID in ConsentForm
GET surrogate_id & slr_id
surrogate_id & slr_id
Fetch sink_keys's
Generate common CR
Generate ki_cr
Generate CR for sink
Generate CR for source
Generate CSR's
Send CR/CSR to sign and store
Signed Sink and Source CR/CSR
Post CR-Sink, CSR-Sink
Post CR-Source, CSR-Source
Sink & Source cr_id
opt
[Install Consent and Consent Status Records]
Install CR/CSR
Verify CR format and mandatory fields
Fetch surrogate_id and slr_id from the CR
Verify SLR is Active for this CR
Verify CR is signed with keys in associated SLR
Verify CSR format and mandatory fields
Verify CSR is signed with keys in associated SLR
Verify CSR chain is intact.
Store CR and CSR
201 cr_id
Install CR/CSR
Verify CR format and mandatory fields
Fetch surrogate_id and slr_id from the CR
Verify SLR is Active for this CR
Verify CR is signed with keys in associated SLR
Verify CSR format and mandatory fields
Verify CSR is signed with keys in associated SLR.
Verify CSR chain is intact.
Store CR and CSR
201 cr_id
opt
[Generate Auth Token]
GET AuthToken
Get AuthTokenInfo
AuthTokenInfo
Generate AuthToken
AuthToken
UI (Operator)
Account Management
Service Management (Operator)
Service Registry
Service Management (Sink)
Service Management (Source)

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”, 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