# Autogiro – Creditor APIs - Specification

AUG-CRED-SPEC

Version: 1.0
08.02.2023

**TLP: WHITE**

Acc. To the Norwegian defined Traffic-Light-Protocol (Bits-TLP Traffic-Light-Protocol)

As long as copyright is respected, information marked **TLP: WHITE** may be distributed without restrictions.


| **Bits AS** | | | |
|---|---|---|---|
| Postaddress:<br>Postboks 26<br>0205 OSLO | Visiting address:<br>Hansteensgt 2<br>OSLO | Phone:<br>+47 23 28 45 10<br>E-mail:<br>post@bits.no | Org.nr.:<br>NO 916 960 190 |

# 1 Table of Contents

# 2 Document Information

## 2.1 Document History

| Version | Status | Date | Editor |
|---|---|---|---|
| 1.0 | First release | 08.02.2023 | K. Holm |

## 2.2 Change Log

| Version | Changes |
|---|---|
| | |

## 2.3 Reference Documents

| Short name/name | Document | Source |
|---|---|---|
| API-SPEC | API Specification (OpenAPI 3.0)<br>https://bitsnorge.github.io/AutoGiro-Creditor-APIs/ | Bits |
| HTTP Message Signatures | draft-ietf-httpbis-message-signatures-13<br>https://datatracker.ietf.org/doc/html/draft-ietf-httpbis-message-signatures-13 | IETF |

## 2.4 Definitions

| Term | Definition |
|---|---|
| Mandate | A mandate in the Autogiro system (fullmakt) |
| Fullmaktsregisteret | The central infrastructure administering the register of mandates (fullmakter) and facilitates the communication between Autogiro participating banks and creditors. |

## 2.5 Terminology

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 RFC 2119.

## 2.6 Latest version of the document

Latest version of this document can be obtained from https://www.bits.no/document/autogiro-creditor-apis-specification

## 2.7 Traffic Light Protocol (TLP)

Bits AS uses TLP in accordance with «FIRST – TLP Standard Definitions and Usage Guidance». (https://www.first.org/tlp) and (http://www.bits.no/tlp)

| | | |
|---|---|---|
| **TLP:WHITE**<br>  <br>Disclosure is not limited. | Sources may use **TLP:WHITE** when information carries minimal or no foreseeable risk of misuse, in accordance with applicable rules and procedures for public release. | Subject to standard copyright rules, **TLP:WHITE** information may be distributed without restriction. |

# 3 Introduction

Creditors have agreement with a bank to use the Autogiro service. All communication between the creditor and the operator of Autogiro fullmaktsregister that is done by creditor is on behalf of the creditors bank. To know which payers should be billed using Autogiro, the creditor creates a mandate based on an agreement with the debtor. Creditors using Autogiro as a service for their customers have traditionally only been able to retrieve and create these mandates via a Batch Interface or manually via mail. In order to support and standardise on HTTP + REST based APIs, Bits and Mastercard as the operator of Autogiro fullmaktsregisteret have now developed APIs for creditors to maintain mandate information in fullmaktsregisteret.

## 3.1 Document purpose

The purpose of this document is to provide technical documentation for the API interfaces to be used by creditors using Autogiro based on an agreement with their bank. The technical documentation is meant to guide the reader with regards to developing the solution. This document does not comment on rules in effect governing the use of these APIs nor does it address billing or costs associated with using the APIs.

## 3.2 Audience

The audience of this document is organisations using Autogiro as a payment option for their customers based on an agreement with their bank and Mastercard as the operator of Autogiro fullmaktsregisteret. The main focus of this document covers technical aspects for use of the APIs and as such is aimed at technicians.

# 4 Specification

## 4.1 Security

### 4.1.1 Authentication

The mechanism used to authenticate the creditor will be Mutual-TLS or MTLS. The authentication will use Organisation-validated certificates in accordance with eIDAS regulations for a certificate on level NCP or better for the creditor. Fullmaktsregisteret will authenticate itself using a qualified QWAC TLS certificate issued to the operator of Fullmaktsregisteret.

Test certificates issued by issuers on the verified issuer list for QC eSeal and QWAC certificates are valid certificates for use in the test environment. Only real production ready certificate issued to the organisation using the APIs will be allowed in production.

Both participants must only accept valid certificates that have not expired when authenticating the opposite party, this must be done either manually (by manually installing the certificates) or by using the CAs OCSP or CRL-list.

### 4.1.2 Transport Security (TLS)

Mutual-TLS will also be the mechanism to ensure transport security. Only TLS version 1.2 or later is allowed. For TLS 1.2 only cipher suites allowed for use in TLS 1.3 is allowed.

### 4.1.3 Integrity protection and non-repudiation

Every request made to Fullmaktsregisteret must be signed and every response from Fullmaktsregisteret will be signed. The signatures must be created in accordance with a standard for message signatures *[HTTP Message Signatures]*. It is mandatory for the creditor to validate the signature of any message. Fullmaktsregisteret will use an organisation-validated certificate issued to the operator of Fullmaktsregisteret to create the signature. The organisation must similarly use a valid organisation-validated certificate as described in chapter 4.1.1 to create their signatures.

Signatures to and from Fullmaktsregisteret will include important header elements in addition to “derived-components” as explained in the standard for signing HTTP messages. The signatures must contain the following components:

- **@request-target** – The full request-target for the request that this signature is attached to.
- **@status** – *(Response only)* The HTTP status code of the response
- **X-Request-ID** – Identifier of the request.
- **Client-Name –** Common name of the client making this request
- **Requester-Merchant** – (Request only) Name of the Merchant making this request, if the communication with Fullmaktsregisteret is outsourced, this must still be the name of the merchant this message originates from.
- **Content-Digest** – Contains a hash of the message body of the response. This will be hashed using SHA-256. (NB the “/mandate/{mandate_request_identification}” will omit this field).
- **@signature-params** – Contains information about how the signature was created. The signature-params component will contain a list of all the components used to create this signature, in addition to information about how the signature was created. The signature-params list will for signatures from Fullmaktsregisteret contain the following:
    - @request-target (a derived component)
    - @status (response only, a derived component)
    - x-request-id
    - requester-merchant
    - content-digest
    - created: UNIX-timestamp of when the signature was created
    - alg: The algorithm that was used to create this message. Always “rsa-pss-sha512”
    - keyid: An x5t thumbprint of the certificate that was used to create the signature.

Chapter 5 lists what signature components should be used for each request-type. Examples of signatures are available in Appendix 1 .

## 4.2 Common headers

All APIs have a set of header elements that are common between them. These all act in the same way and as such are explained here:

- X-Request-ID: Unique identifier of the request, must be assigned by the requestor and should be unique within a timespan of one week. If a message is received with the same X-Request-ID as pervious request, it must be treated as a duplicate.
- Requester-Merchant: Name of the Merchant making this request, if the communication with Fullmaktsregisteret is outsourced, this must still be the name of the merchant this message originates from.
- Client-Name: Name of the direct technical participant that sent the message, if the communication with Fullmaktsregisteret is outsourced this will be the name of the party that this has been outsourced too.

## 4.3 Domains

All APIs from Fullmaktsregisteret will be available on the following domains:

- Test: https://payment-api-test.nets.no/autogiro-creditor-api
- Production: https://payment-api.nets.no/autogiro-creditor-api

## 4.4 Error handling

### 4.4.1 Error response

In cases where the request from creditor to Fullmaktsregisteret causes an error, this will result in an error response from Fullmaktsregisteret. The APIs all handle error responses in the same manner, except for when for whatever reason the message is intercepted and responded to by the gateway. If an error is caught at the gateway this will result in a simple HTTP status code response of either 401, 403 or 404.

In cases where an error is handled by Fullmaktsregisteret itself, the request will be responded to with an error message and accompanying HTTP status code. The error response comes in the following format:

```
{
  "errorCode": "AUG-001",
  "errorMessage": "Invalid request",
  "timestamp": "2018-02-05T12:54:12"
}
```

All error responses will contain an application specific error code, a pre-defined error message and a timestamp. The defined error codes and error messages are listed below:

- 'AUG-001' - Invalid request (e.g. mandatory input missing)
- 'AUG-002' - Invalid input (e.g. provided input doesn't have) correct values
- 'AUG-003' - Method is not allowed
- 'AUG-004' - Unsupported media type
- 'AUG-005' - Client is not authorized, when provided customer unit id not configured or not provided through gateway
- 'AUG-006' - Client does not have access
- 'AUG-007' - Invalid credit account
- 'AUG-008' - Missing credit agreement
- 'AUG-009' - Invalid Debtor account
- 'AUG-010' - Invalid payment reference
- 'AUG-011' - Invalid period
- 'AUG-012' - Invalid mandate type
- 'AUG-013' - Mandate already exist
- 'AUG-014' - Invalid Debtor & creditor (Same creditor and debtor)
- 'AUG-015' - Invalid amount limit
- 'AUG-016' - Mandate not found
- 'AUG-017' - Internal error
- 'AUG-018' - Signature could not be verified

### 4.4.2 Message repetition

All creditors should follow these rules for the repetition of messages. This describes how messages should be repeated for all APIs in the solution.

A request is considered timed-out if a creditor does not receive an answer after 20 seconds. After a message has timed out the message may be repeated. The message can be repeated with an interval following this formula where *n* is the number of the repetition:

$$This\ intervall\ in\ seconds(n) = (n-1)^3 + 30$$

This will give a repetition sequence of:

| Repetition | Wait time(seconds) | Time passed since original request (includes 20 second wait for response) |
|---|---|---|
| 1 | 30 | 50 seconds |
| 2 | 31 | 1 minute and 41 seconds |
| 3 | 38 | 2 minutes and 39 seconds |
| 4 | 57 | 3 minutes and 56 seconds |
| 5 | 94 | 5 minutes and 50 seconds |

This may continue until the total time since the original request has passed 6 minutes (5 repetitions), after this the request should be considered a failure and manual investigation shall begin.

## 4.5 API Specification

In addition to this document the APIs are defined in an OpenAPI 3.0 specification. Which can be found here:

https://bitsnorge.github.io/AutoGiro-Creditor-APIs/

# 5 Use cases

Below follows a list of possible use-cases and their deviations. The list of use cases is complete, but they do not contain all possible deviations. The deviations illustrated should be viewed together with the API specification to inform the developer how to program their interface with the APIs.

## 5.1 Create a mandate (POST /mandate)

This API is used to create a single mandate in Fullmaktsregisteret.

The creditor may use this API to create a new mandate between a creditor and a debtor. The request uses the standard headers and the message body contains a mandate object, which will be used to add the mandate to the registry.

### 5.1.1 Signature elements in use for the request:

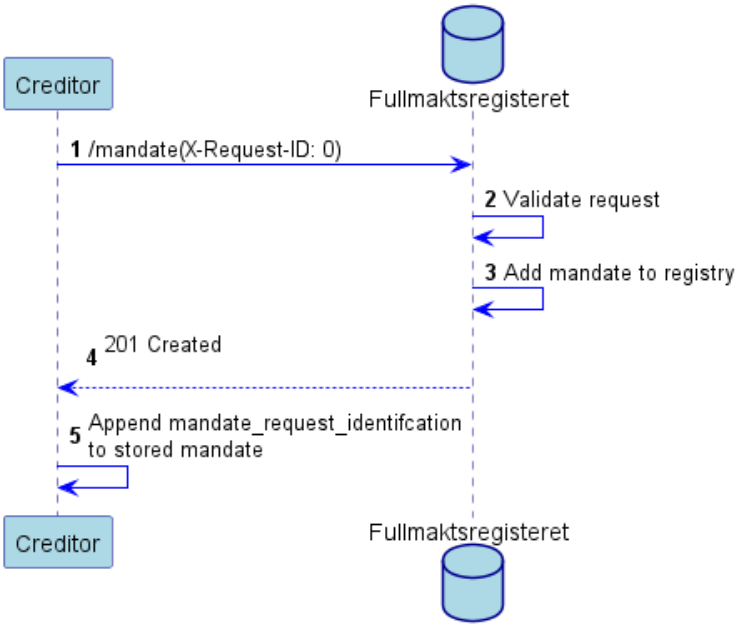
- **@request-target**
- **@method**
- **@authority**
- **X-Request-ID**
- **Client-Name**
- **Requester-Merchant**
- **Content-Digest**
- **@signature-params**
    - @request-target
    - @method
    - @authority
    - x-request-id
    - client-name
    - requester-merchant
    - content-digest
    - created
    - alg
    - keyid

### 5.1.2 Signature elements in use for the response:

- **@request-target**
- **@status**
- **X-Request-ID**
- **Client-Name**
- **Content-Digest**
- **@signature-params**
    - @request-target
    - @status
    - x-request-id
    - client-name
    - content-digest
    - created
    - alg
    - keyid

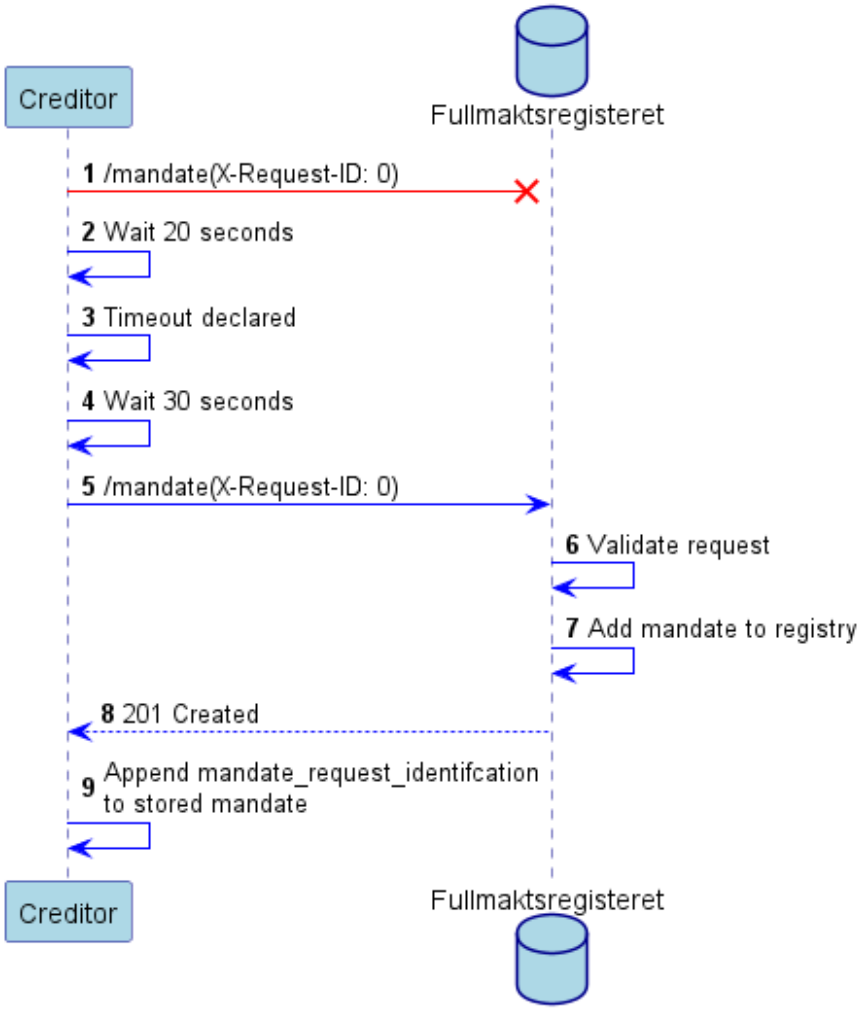
### 5.1.3 Sequence: Normal situation

In a normal situation the creditor calls the API to create a mandate in the registry of Fullmaktsregisteret. Note that it will take two days after the mandate has been created before it can be used.



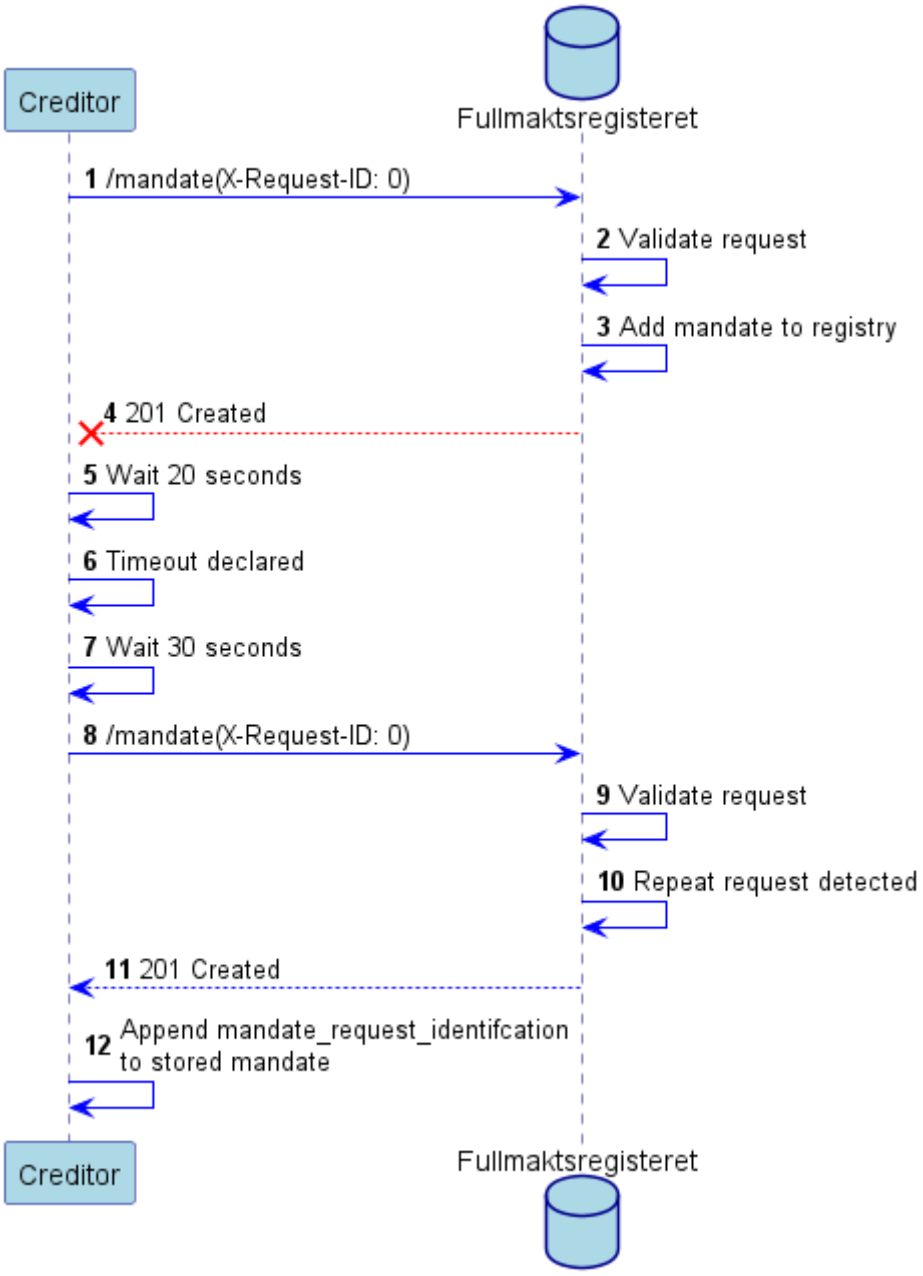
### 5.1.4 Sequence: Deviation – Creditor request does not reach Fullmaktsregisteret

If the creditor receives no reply from Fullmaktsregisteret within the specified time for message repetition the creditor must send the request again. For a repeat message the same X-Request-ID must be used:



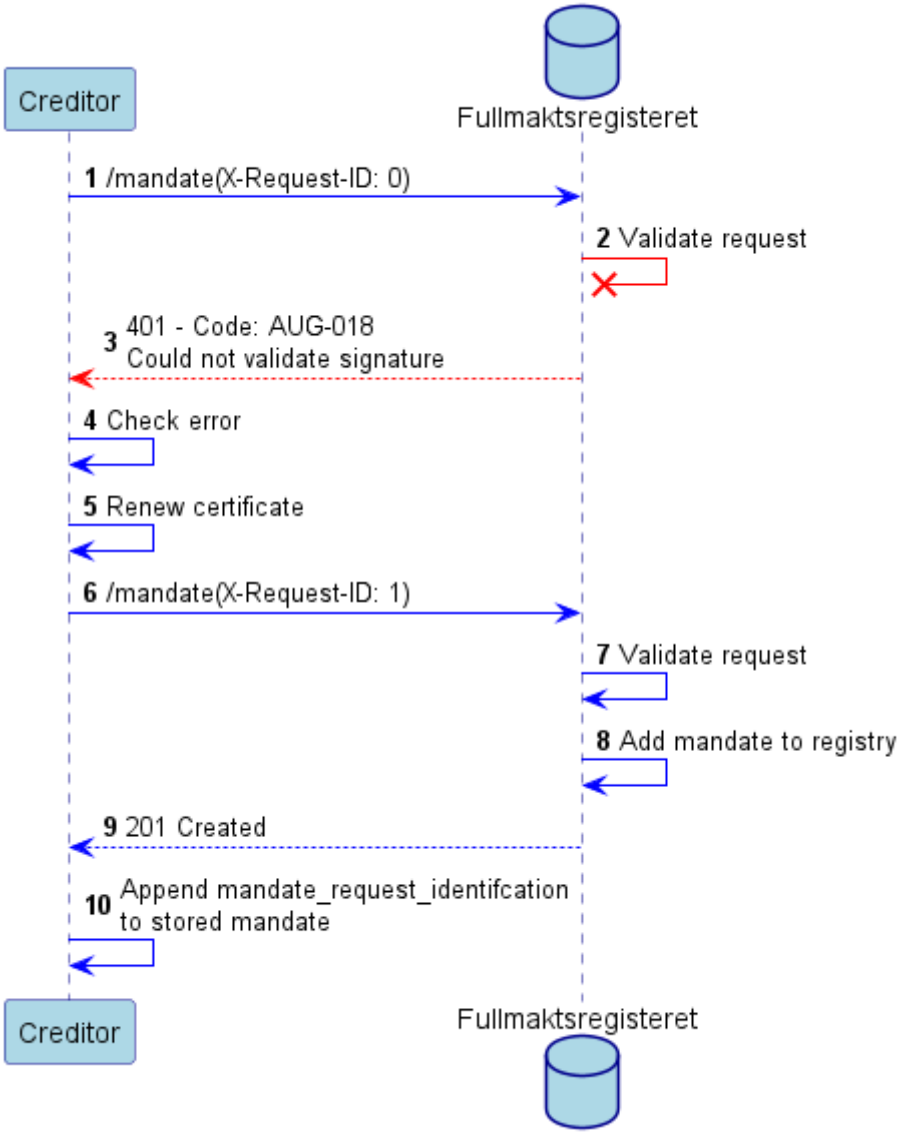
### 5.1.5 Sequence: Deviation – Creditor receives no reply from Fullmaktsregisteret

If the creditor receives no reply from Fullmaktsregisteret within the specified time for message repetition the creditor must send the request again. For a repeat message the same X-Request-ID must be used:



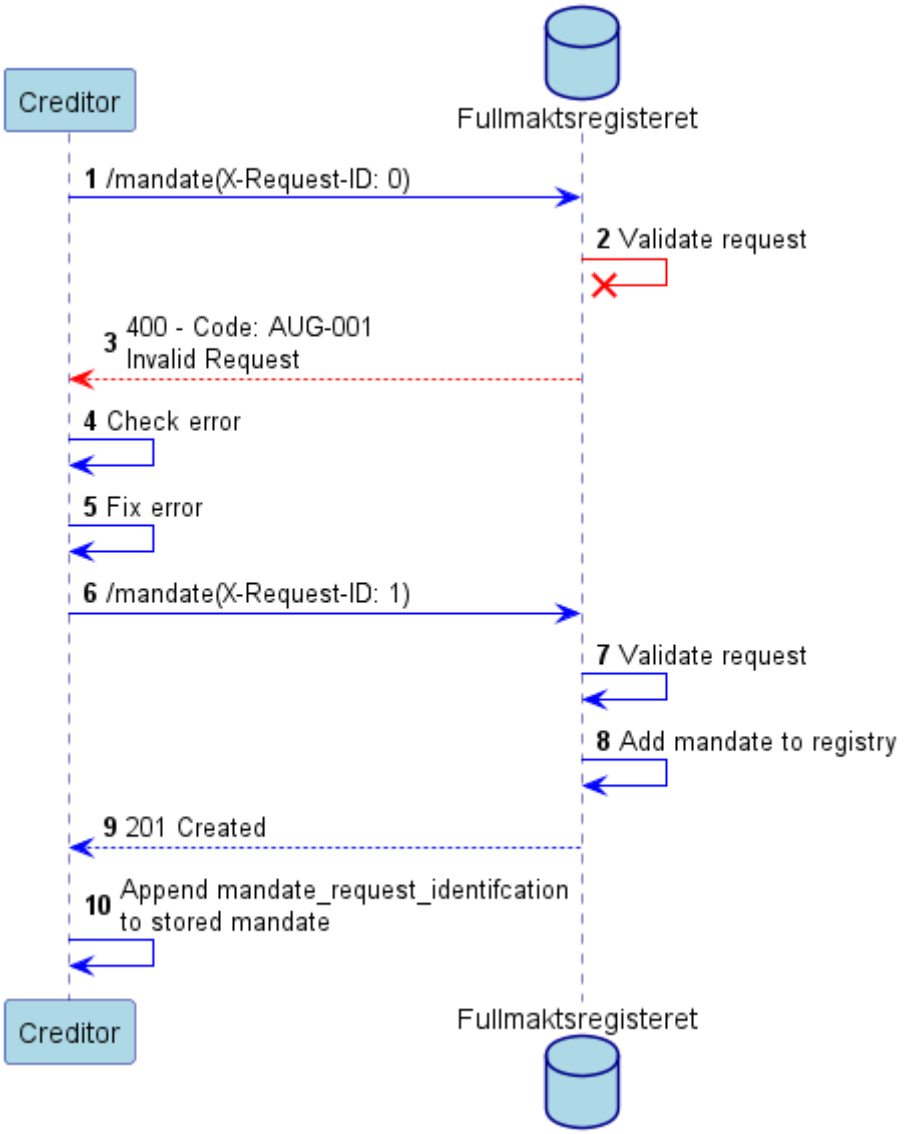
### 5.1.6 Sequence: Deviation – A signature could not be validated

If Fullmaktsregisteret detects an error with the request it will return an error to the creditor. In such cases the creditor must resolve their error before attempting another request. In such cases a new X-Request-ID must also be used. If for instance Fullmaktsregisteret rejects the request based on the fact that the creditor certificate that was used to create the signature was expired, the creditor must attempt the message again, after renewing its certificate, and using a new X-Request-ID.



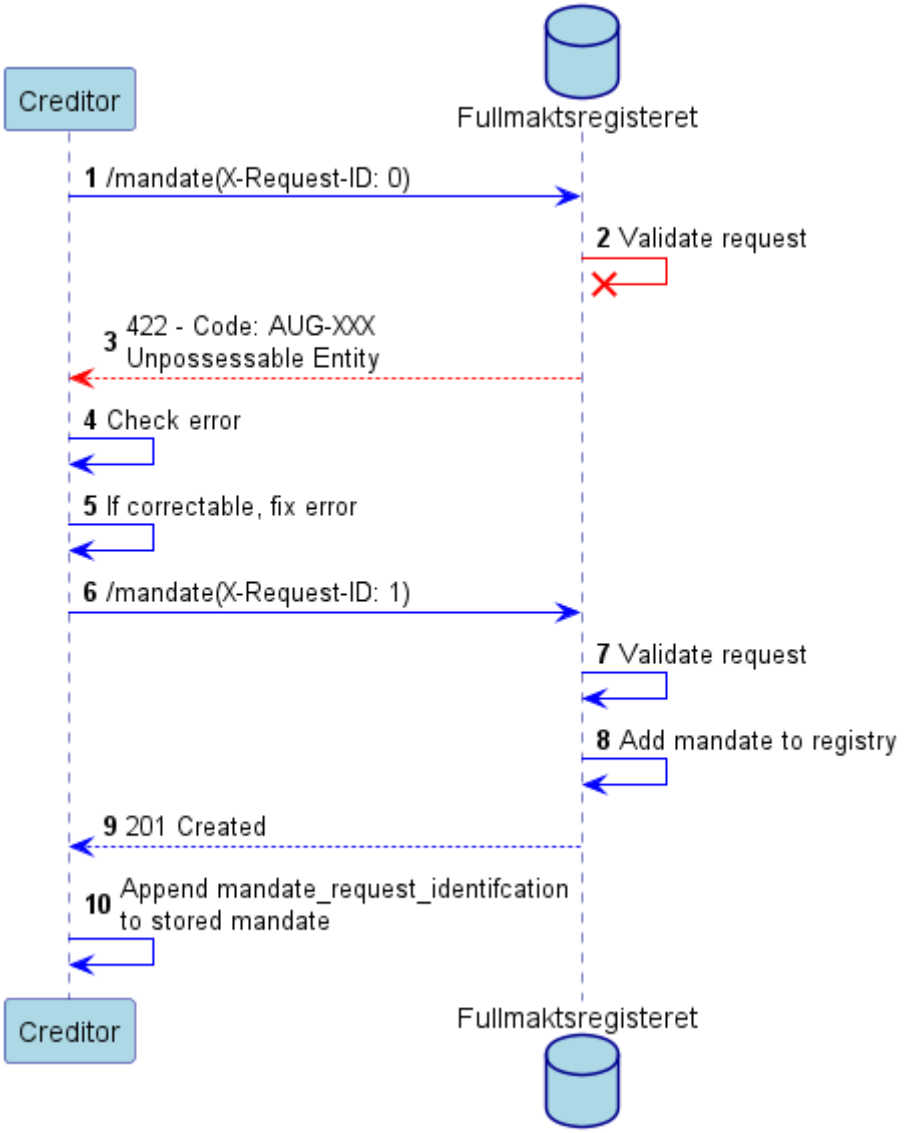
### 5.1.7 Sequence: Deviation – Technical error with message formatting or missing content

If the request from the creditor is malformed so that Fullmaktsregisteret is unable to process the request an error will be returned to the creditor. If information that is essential is missing from either the headers or the message itself so that Fullmaktsregisteret is unable to interpret the request a similar error will be returned. In cases where this occur manual investigation and error correction should be initiated by the creditor.



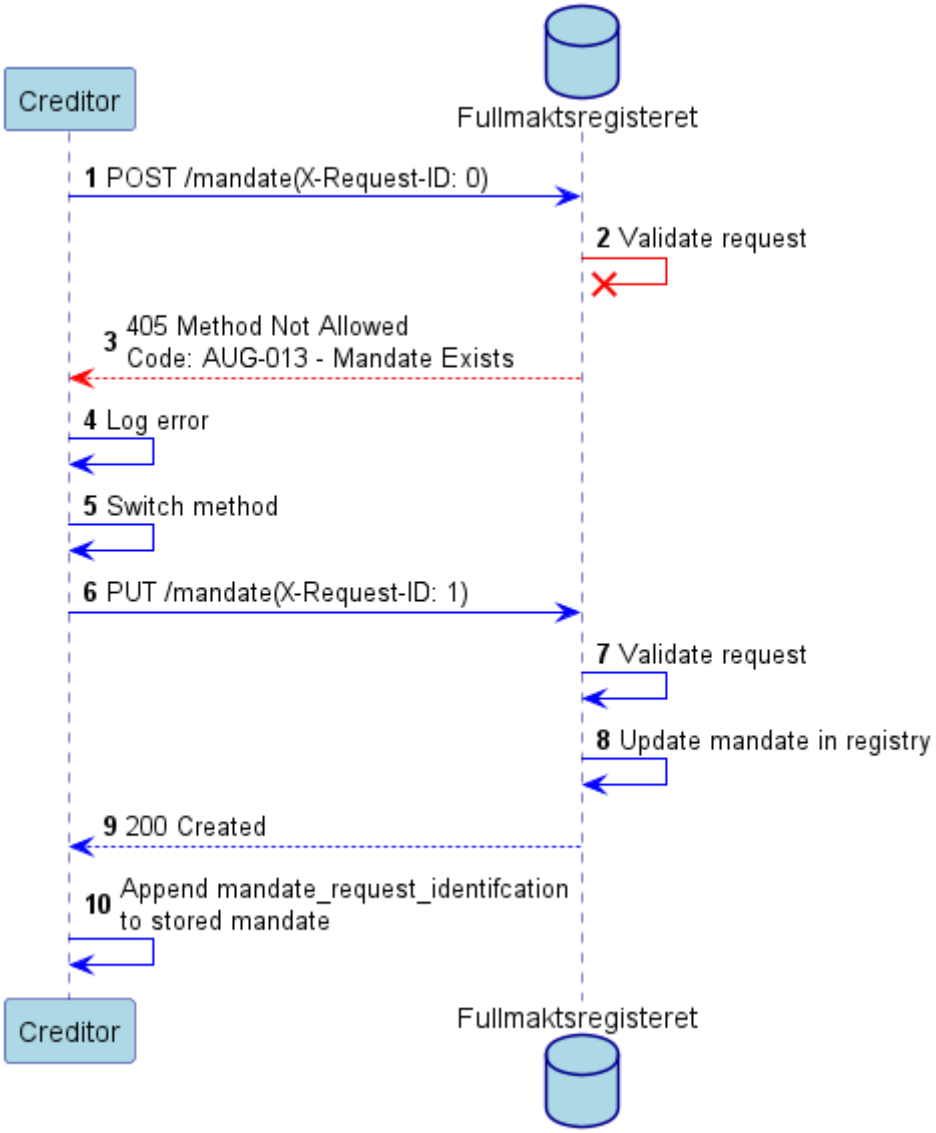
### 5.1.8 Sequence: Deviation – Invalid mandate

If the request from the creditor is valid and can be interpreted but the mandate itself is invalid due to logical or business-related reasons Fullmaktsregisteret will respond with an error. The error response will always be a part of a 422 “Unpossessable entity” response, but the error code in the response will give more detailed information about what caused the error.



### 5.1.9 Sequence: Deviation – Mandate already exists

If the mandate already exists, this is considered an error if the endpoint is called using the “POST” method. Updates must be performed using the “PUT” method. If the creditor creates a new mandate with conflicting data for a mandate that already exist this will result in an error from Fullmaktsregisteret.



## 5.2 Update existing mandate (PUT /mandate)

The "/mandate" endpoint is used both to create and update existing mandates, the only difference between updating and creating is the method used. Creating new mandates uses the "POST" method, while updating existing mandates uses the "PUT" method. For this reason, many of the normal cases and deviations will be the same for updating and creating. All scenarios are therefore not covered in this chapter, please refer to chapter 5.1.1, 5.1.2, 5.1.3, 5.1.4, 5.1.5 and 5.1.6 for a more comprehensive description of different scenarios (Note 5.1.7 is unapplicable for the update functionality).

### 5.2.1 Signature elements in use for the request:

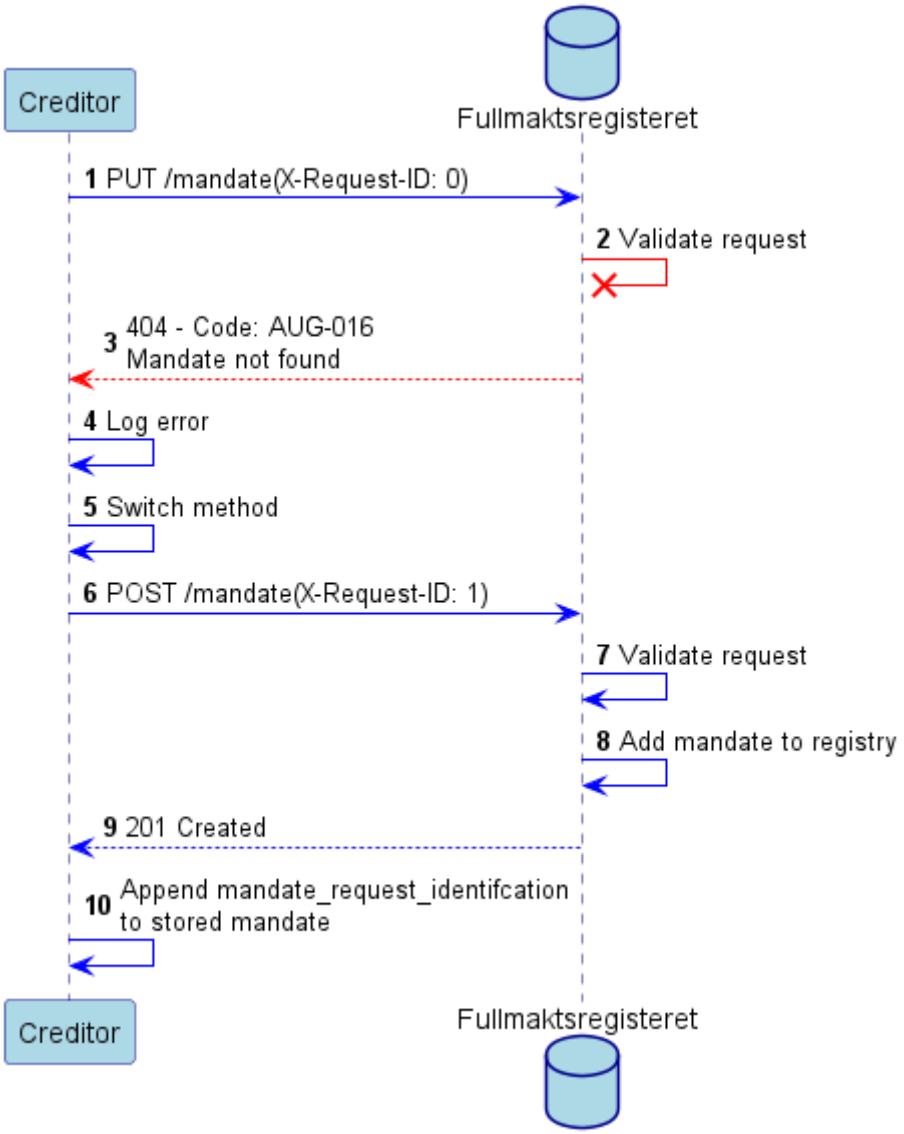
- **@request-target**
- **@method**
- **@authority**
- **X-Request-ID**
- **Client-Name**
- **Requester-Merchant**
- **Content-Digest**
- **@signature-params**
    - @request-target
    - @method
    - @authority
    - x-request-id
    - client-name
    - requester-merchant
    - content-digest
    - created
    - alg
    - keyid

### 5.2.2 Signature elements in use for the response:

- **@request-target**
- **@status**
- **X-Request-ID**
- **Client-Name**
- **Content-Digest**
- **@signature-params**
    - @request-target
    - @status
    - x-request-id
    - client-name
    - content-digest
    - created
    - alg
    - keyid

### 5.2.3 Sequence: Deviation – Mandate not found

If Fullmaktsregisteret is unable to find a mandate correlating with the information provided in the request, this will result in an error from Fullmaktsregisteret.



## 5.3 Delete a mandate (DELETE /mandate/{mandate_request_identification})

There are two ways of deleting mandates stored in Fullmaktsregisteret using APIs. The first endpoint uses the DELETE method on the same URL as the update and create endpoint ended by the "mandate_request_identifaction" field representing a previously created mandate. The "mandate_request_identification" is a field that is returned from Fullmaktsregisteret when creating or updating a mandate. (the second endpoint for deletion is described in chapter 5.4)

### 5.3.1 Signature elements in use for the request:

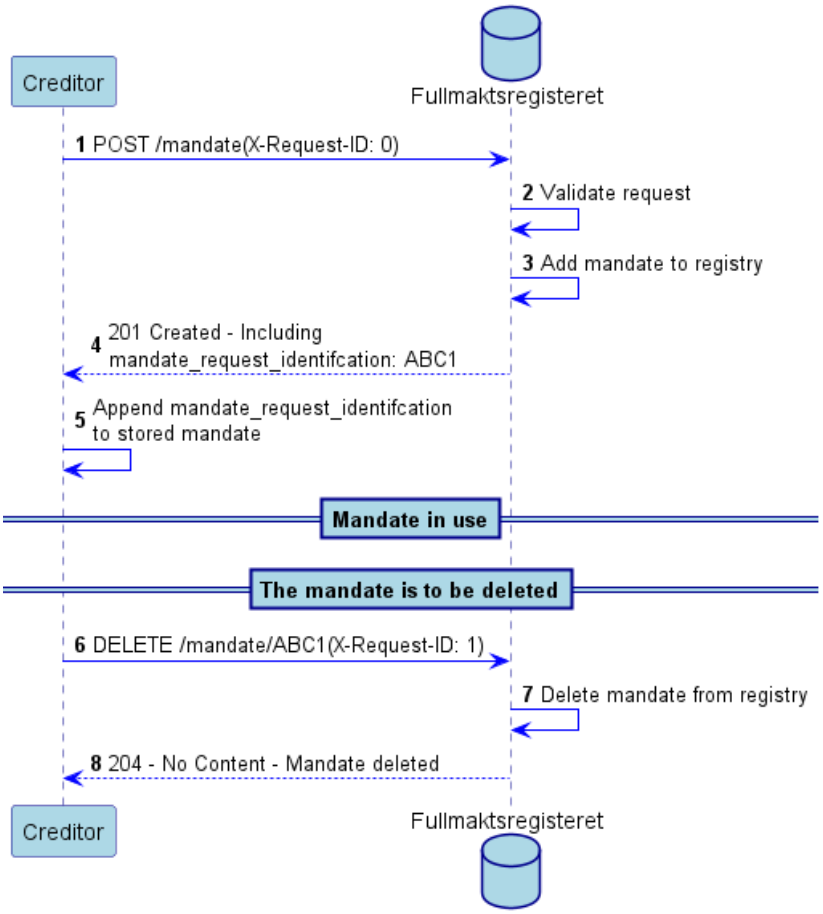
- **@request-target**
- **@method**
- **@authority**
- **X-Request-ID**
- **Client-Name**
- **Requester-Merchant**
- **@signature-params**
    - @request-target
    - @method
    - @authority
    - x-request-id
    - client-name
    - requester-merchant
    - created
    - alg
    - keyid

### 5.3.2 Signature elements in use for the response:

- **@request-target**
- **@status**
- **X-Request-ID**
- **Client-Name**
- **@signature-params**
    - @request-target
    - @status
    - x-request-id
    - client-name
    - created
    - alg
    - keyid

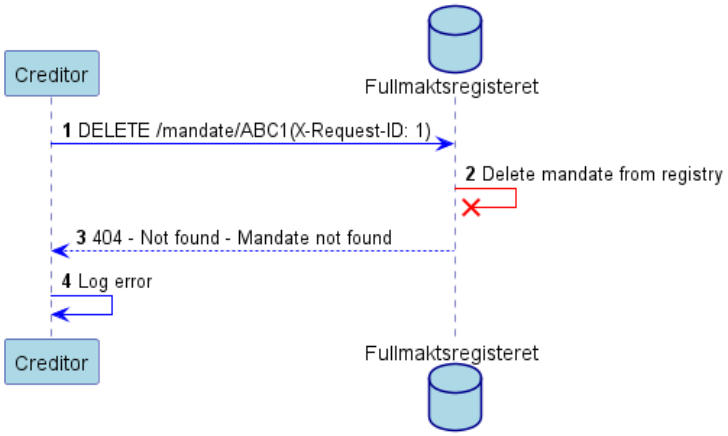
### 5.3.3 Sequence: Normal Situation – Creating and Deleting a mandate

In order to demonstrate the use of the “mandate_request_identification”, this flow includes the flow for creating a mandate in addition to deleting it.



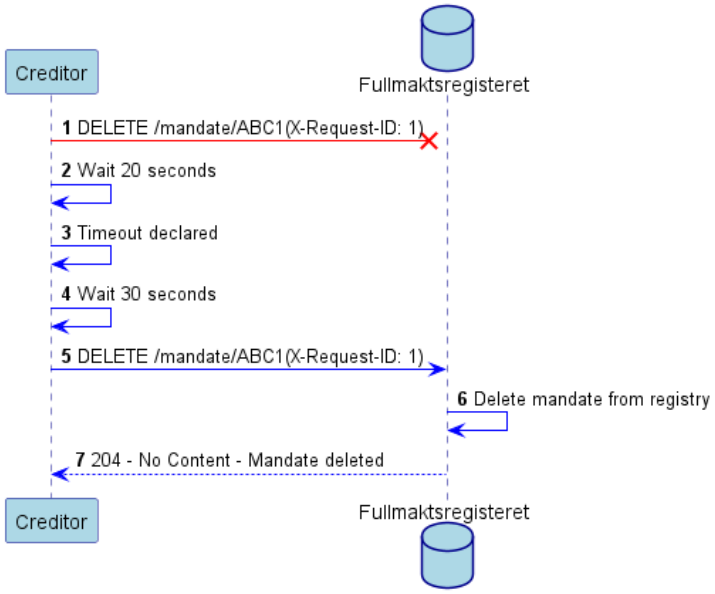
### 5.3.4 Sequence: Deviation – Mandate not found

If a “mandate_request_identification” does not correspond to a mandate, Fullmaktsregisteret will respond with an error if a delete with that id is attempted:



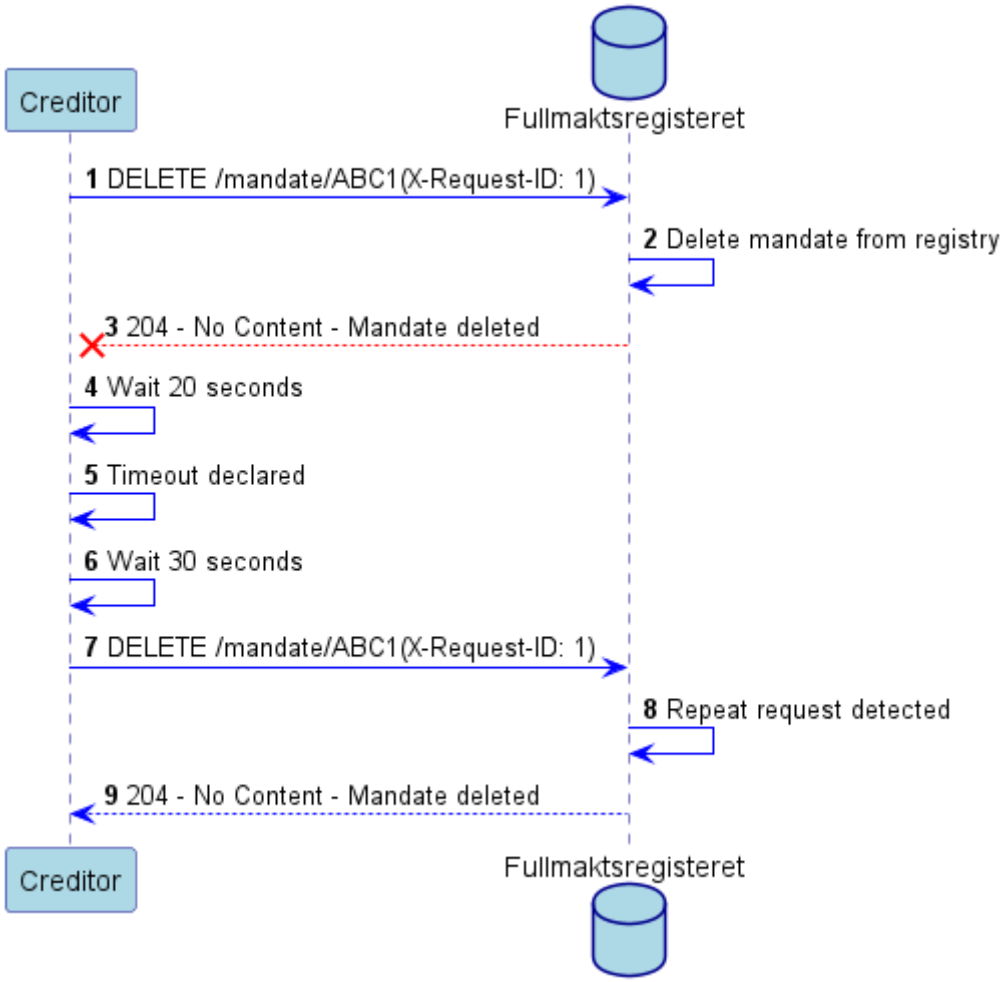

### 5.3.5 Sequence: Deviation – The request did not reach Fullmaktsregisteret

If the request does not reach Fullmaktsregisteret, the creditor must wait the allotted time specified in chapter 4.4.2 and then repeat the request.



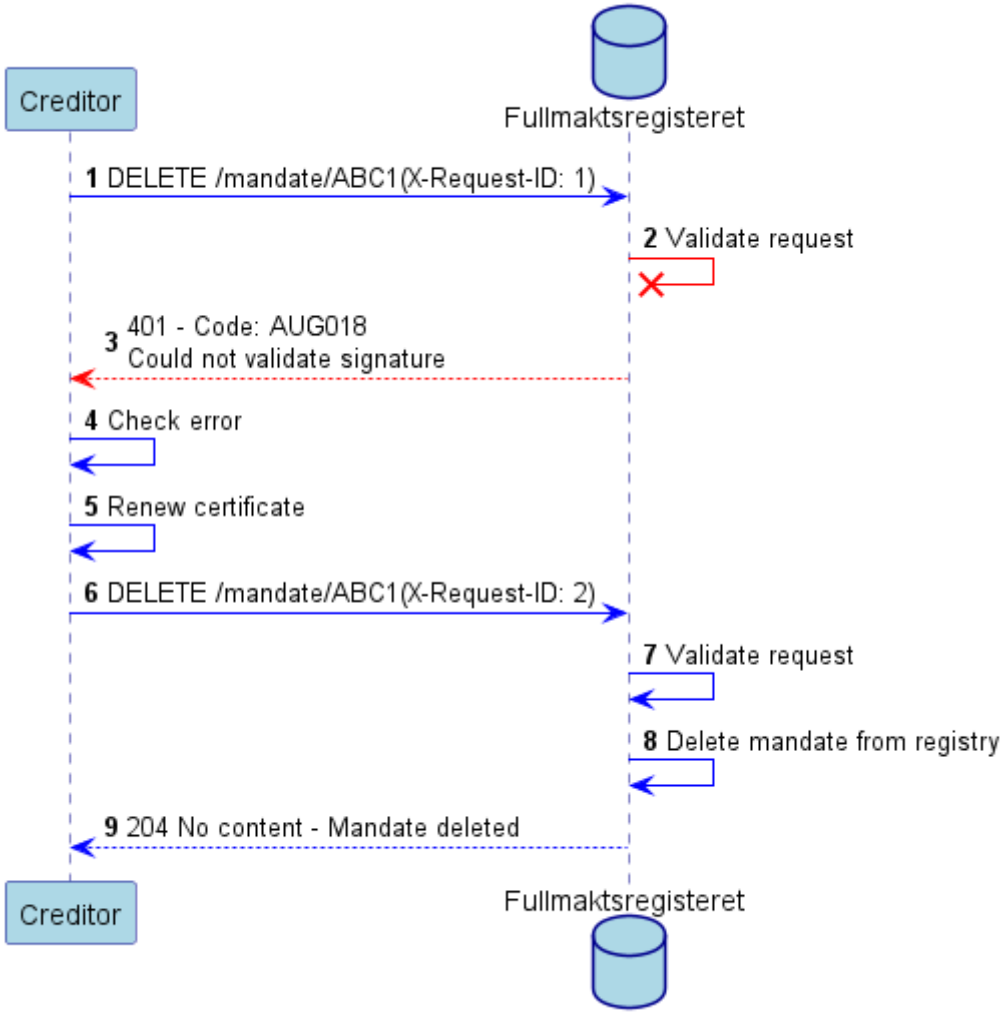
### 5.3.6 Sequence: Deviation – The response did not reach the creditor

If the creditor does not receive a response within the allotted time specified in chapter 4.4.2, the creditor must repeat the message. If the request is a duplicate of a previous request, Fullmaktsregisteret will repeat the response.



### 5.3.7 Sequence: Deviation – Signature not verified

If Fullmaktsregisteret detects an error with the request it will return an error to the creditor. In such cases the creditor must resolve their error before attempting another request. In such cases a new X-Request-ID must also be used. If for instance Fullmaktsregisteret rejects the request based on the fact that the creditor certificate that was used to create the signature was expired, the creditor must attempt the message again, after renewing its certificate, and using a new X-Request-ID.



## 5.4 Delete a mandate (POST /mandate/delete)

The other way of deleting a mandate using the API interface, is with a POST on the "/mandate/delete" endpoint. This endpoint allows the creditor to delete mandates where the creditor either don't know the "mandate_request_identification" or where the mandate has no such identifier assigned. The endpoint expects the creditor to provide a request body, containing other identifiers which can be used to uniquely identify the mandate.

### 5.4.1 Signature elements in use for the request:

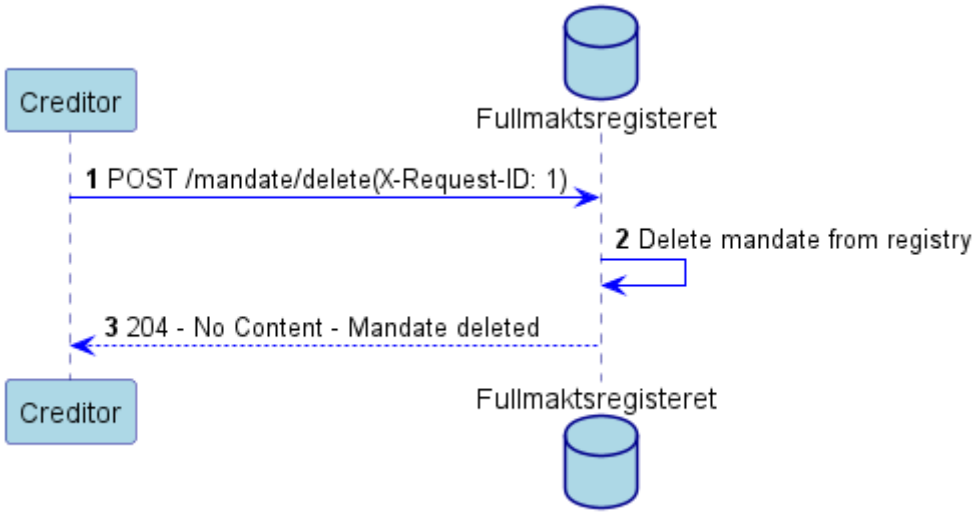
- **@request-target**
- **@method**
- **@authority**
- **X-Request-ID**
- **Client-Name**
- **Requester-Merchant**
- **Content-Digest**
- **@signature-params**
    - @request-target
    - @method
    - @authority
    - x-request-id
    - client-name
    - requester-merchant
    - content-digest
    - created
    - alg
    - keyid

### 5.4.2 Signature elements in use for the response (only provided on successful requests):

- **@request-target**
- **@status**
- **X-Request-ID**
- **Client-Name**
- **@signature-params**
    - @request-target
    - @status
    - x-request-id
    - client-name
    - created
    - alg
    - keyid

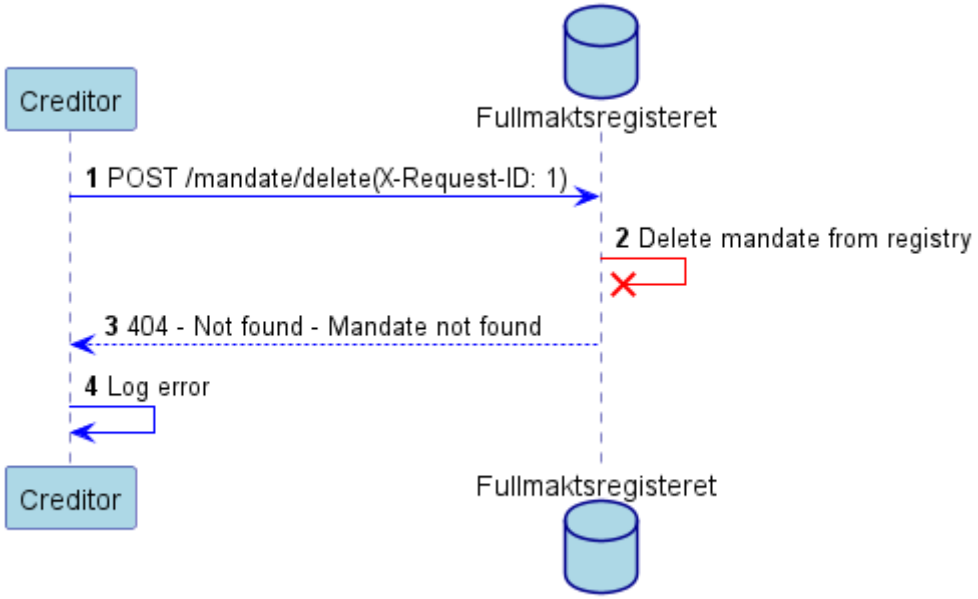
### 5.4.3 Sequence: Normal situation

In a normal situation the endpoint will be used if the creditor has no knowledge of a “mandate_request_identification” associated with the mandate they wish to delete. The creditor will then request that the mandate be deleted using other identifying information.



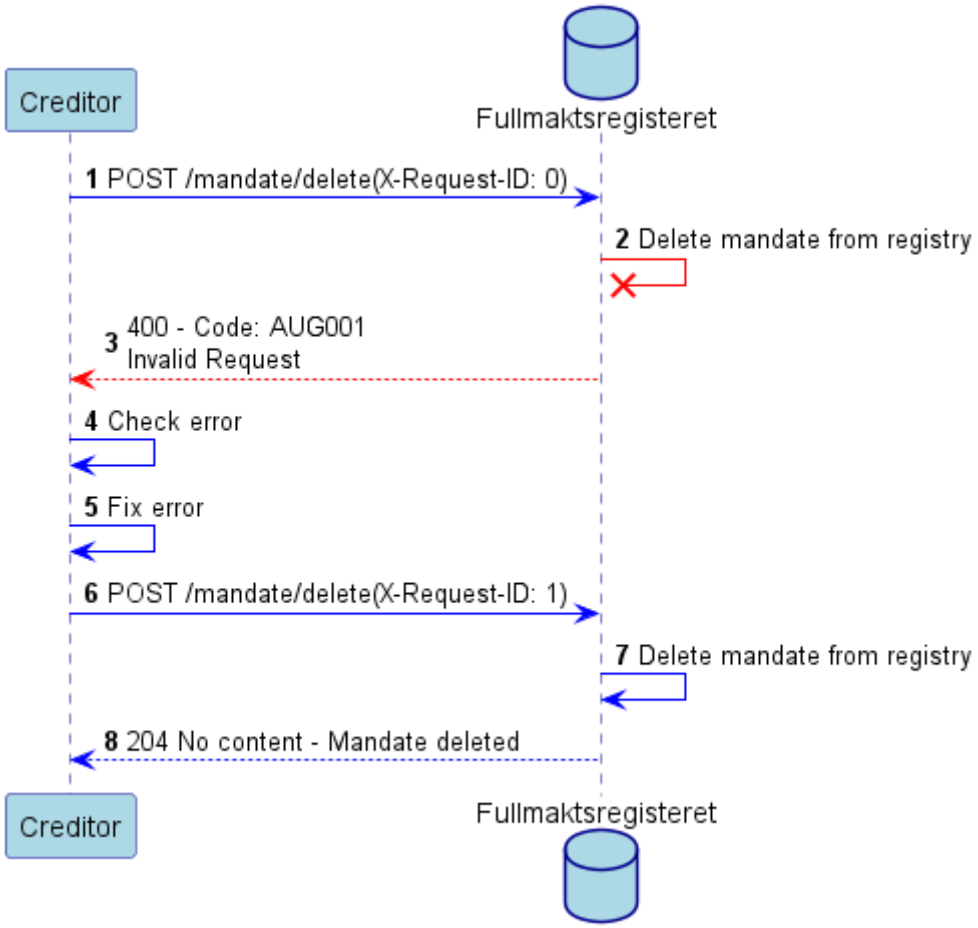

### 5.4.4 Sequence: Deviation – Mandate not found

If Fullmaktsregisteret is unable to find a mandate with the correlating identifying information supplied in the request it will respond with an error.



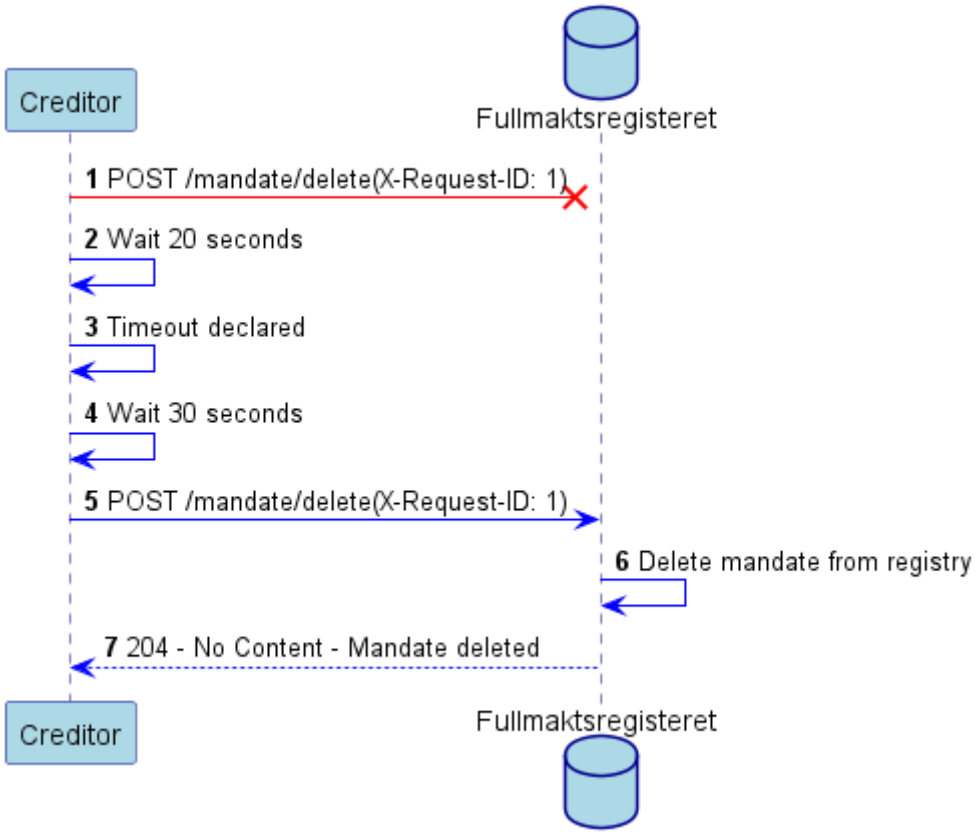
### 5.4.5 Sequence: Deviation – Mandate information format invalid

If the request to delete a mandate contains format or other technical errors that prevent Fullmaktsregisteret from interpreting the message it will respond with an error. The creditor must then fix the issue and try again.



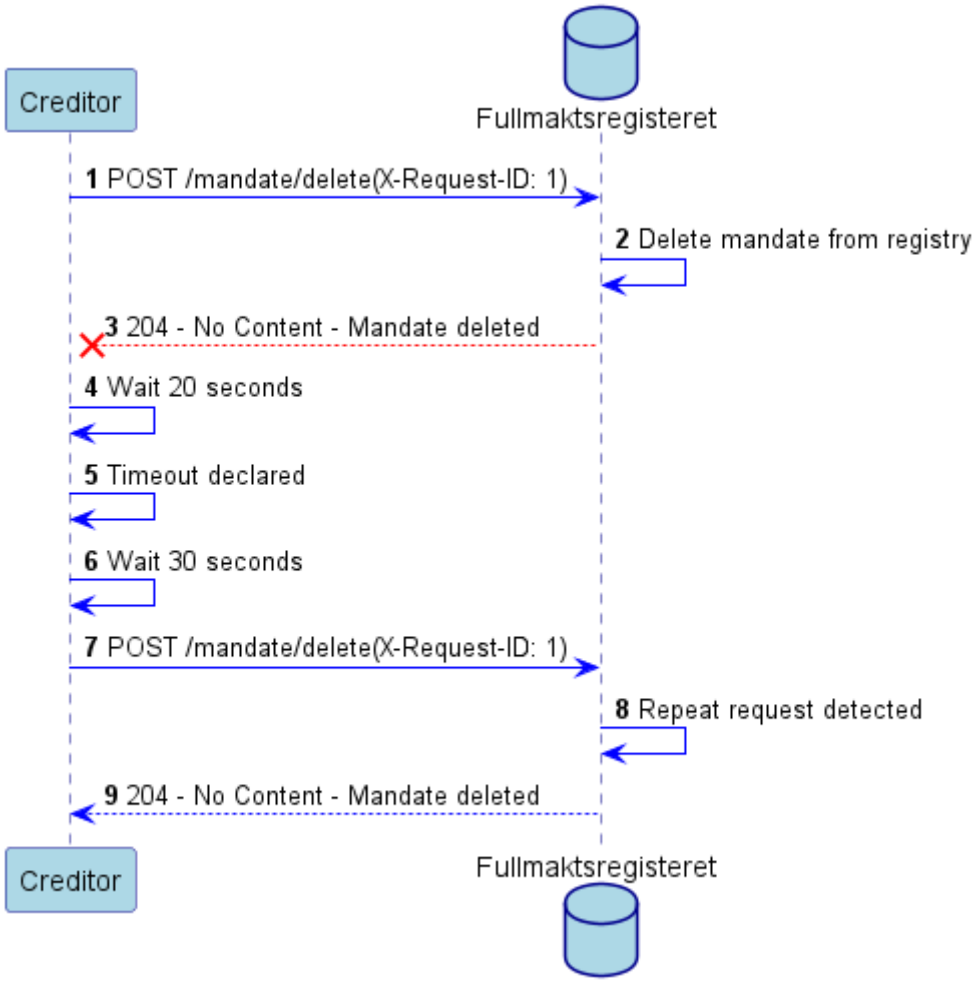
### 5.4.6 Sequence: Deviation – The request did not reach Fullmaktsregisteret

If the request does not reach Fullmaktsregisteret, the creditor must wait the allotted time specified in chapter 4.4.2 and then repeat the request.



### 5.4.7 Sequence: Deviation – The response did not reach the creditor

If the creditor does not receive a response within the allotted time specified in chapter 4.4.2, the creditor must repeat the message. If the request is a duplicate of a previous request, Fullmaktsregisteret will repeat the response.



# 6 Appendix 1 – HTTP Signatures - Examples

This document describes the process for how to create a HTTP signature in accordance with the specifications. In addition, this document includes examples that can be used to aid development. Note that these are examples only, using fake throw away keys and certificates.

## 6.1 Example and step-by-step walkthrough of creating a HTTP-signature

### 6.1.1 Introduction

The HTTP-Signatures specification details a specification for signing an entire HTTP message. In this chapter we will go through the step-by-step process of creating a HTTP signature. Starting with this message (example from a "/mandate" POST request to fullmaktsregisteret):

```
POST /mandate HTTP/1.1
Host: api.autogiro.no
Content-Length: 1091
X-Request-ID: 294fafb7-0e4e-4177-a5ff-ce7367c45814
Client-Name: Aarnes Badekarforhandler AS
Requester-Merchant: AB-2150
{
      "mandate": {
            "mandate_request_identification": "NOTASSIGNED",
            "type": {
                  "classification": {
                        "code": "FIXE"
                  }
            },
            "occurrences": {
                  "sequence_type": "RCUR",
                  "frequency": {
                        "type": "DAIL"
                  },
                  "duration": {
                        "from_date": "2022-08-01",
                        "to_date": "2023-08-01"
                  },
                  "first_collection_date": "2022-08-01",
                  "final_collection_date": "2023-08-01"
            },
            "tracking_indicator": false,
            "maximum_amount": {
                  "amount": "1000",
                  "currency": "NOK"
            },
            "creditor": {
                  "name": "Aarnes Badekarforhandler AS",
                  "identification": {
                        "organisation_identification": {
                              "other": {
                                    "identification": "916960190",
                                    "scheme_name": {
                                          "proprietry": "AGREEMENT_ID"
                                    }
                              }
```

```
                    }
                }
            },
            "creditor_account": {
                "identification": {
                    "other": {
                        "identification": "60013232345",
                        "scheme_name": {
                            "code": "BBAN"
                        }
                    }
                }
            },
            "debtor": {
                "name": "Ole Olsen",
                "identification": {
                    "organisation_identification": {
                        "other": {
                            "identification": "996739848",
                            "scheme_name": {
                                "proprietry": "AGREEMENT_ID"
                            }
                        }
                    }
                },
                "contact_details": {
                    "preferred_method": "MAIL"
                }
            },
            "debtor_account": {
                "identification": {
                    "other": {
                        "identification": "60013312349",
                        "scheme_name": {
                            "code": "BBAN"
                        }
                    }
                }
            },
            "mandate_reference": "01234567890",
            "xmlns": "urn:iso:std:iso:20022:tech:xsd:pain.009.001.07"
        }
}
```

### 6.1.2 The anatomy of a HTTP signature

The HTTP signature consists of multiple components, including the HTTP headers and derived components, which are data that can be derived from HTTP protocol, like the request-target URL, HTTP response code or HTTP method. Regular HTTP headers are represented normally, while derived components are represented by a name that start with "@". The components used by fullmaktsregisteretf to create a signature will be (unless indicated, all of these are mandatory):

- **@request-target** – The full request-target for the request that this signature is attached to.
- **@status** – *(Response only)* The HTTP status code of the response
- **X-Request-ID** – Identifier of the request.
- **Client-Name –** Common name of the client making this request
- **Requester-Merchant** – (Request only) Name of the Merchant making this request, if the communication with Fullmaktsregisteret is outsourced, this must still be the name of the merchant this message originates from.
- **Content-Digest** – Contains a hash of the message body of the response. This will be hashed using SHA-256. (NB the "/mandate/{mandate_request_identification}" will omit this field).
- **@signature-params** – Contains information about how the signature was created. The signature-params component will contain a list of all the components used to create this signature, in addition to information about how the signature was created. The signature-params list will for signatures from Fullmaktsregisteret contain the following:
  - @request-target (a derived component)
  - @status (response only, a derived component)
  - x-request-id
  - requester-merchant
  - content-digest
  - created: UNIX-timestamp of when the signature was created
  - alg: The algorithm that was used to create this message. Always "rsa-pss-sha512"
  - keyid: An x5t thumbprint of the certificate that was used to create the signature.

When combined for the "/mandate" POST request, the signature-param will look something like this:

```
"@signature-params": ("@request-target" "@method" "@authority" "x-request-id" "client-name" "requester-merchant" "content-digest");created=1668500614;keyid="75wGcKK8tMqzqN5qbg4bs9g5rYU";alg="rsa-pss-sha512"
```

**NB** The signature-params component is required to always come last. All components including HTTP headers are represented in lower-case and the order of the components matter! And when assigned here they cannot change for the life of the signature.

When the HTTP Message is sent the @signature-params is represented by the Signature-Input HTTP header.

### 6.1.3 Creating the digest

The Content-Digest header element is a hash of the message body. To create the digest the message body shall be hashed using SHA-256. Then it shall be Base64 encoded. A message body of:

```
{
     "mandate": {
          "mandate_request_identification": "NOTASSIGNED",
          "type": {
               "classification": {
```

```
                    "code": "FIXE"
               }
          },
          "occurrences": {
               "sequence_type": "RCUR",
               "frequency": {
                    "type": "DAIL"
               },
               "duration": {
                    "from_date": "2022-08-01",
                    "to_date": "2023-08-01"
               },
               "first_collection_date": "2022-08-01",
               "final_collection_date": "2023-08-01"
          },
          "tracking_indicator": false,
          "maximum_amount": {
               "amount": "1000",
               "currency": "NOK"
          },
          "creditor": {
               "name": "Aarnes Badekarforhandler AS",
               "identification": {
                    "organisation_identification": {
                         "other": {
                              "identification": "916960190",
                              "scheme_name": {
                                   "proprietry": "AGREEMENT_ID"
                              }
                         }
                    }
               }
          },
          "creditor_account": {
               "identification": {
                    "other": {
                         "identification": "60013232345",
                         "scheme_name": {
                              "code": "BBAN"
                         }
                    }
               }
          },
          "debtor": {
               "name": "Ole Olsen",
               "identification": {
                    "organisation_identification": {
                         "other": {
                              "identification": "996739848",
                              "scheme_name": {
                                   "proprietry": "AGREEMENT_ID"
                              }
                         }
                    }
               },
               "contact_details": {
```

```
                    "preferred_method": "MAIL"
               }
          },
          "debtor_account": {
               "identification": {
                    "other": {
                         "identification": "60013312349",
                         "scheme_name": {
                              "code": "BBAN"
                         }
                    }
               }
          },
          "mandate_reference": "01234567890",
          "xmlns": "urn:iso:std:iso:20022:tech:xsd:pain.009.001.07"
     }
}
```

Will create a digest header that looks like (note that the JSON object is compressed before it is hashed):

```
Content-Digest: sha-256=:8GPIvXEJbCWeEPAMF55OhE307xO5BzkJye+6eDfsqGk=:
```

### 6.1.4 Creating the signature input string

To form the signature input string used to create the signature parameter the sender must first gather all header elements specified in the @signature-params parameter in the order they appear and combine them. They must be combined by listing the components as they appear in the @signature-params component (in lower-case), encased in quotation marks, with a colon(":") and a space(" ") between the component name and the value. Between each component there shall be a newline ("\n") separating them. (No empty line should be provided at the bottom). Given this complete HTTP message (for a POST request on "https://api-gateway.nets.eu.no/avtalegiro-mandates/v1/listMandates/"):

```
POST /mandate HTTP/1.1
Host: api.autogiro.no
Content-Length: 1091
X-Request-ID: 294fafb7-0e4e-4177-a5ff-ce7367c45814
Client-Name: Aarnes Badekarforhandler AS
Requester-Merchant: AB-2150
{
     "mandate": {
          "mandate_request_identification": "NOTASSIGNED",
          "type": {
               "classification": {
                    "code": "FIXE"
               }
          },
          "occurrences": {
               "sequence_type": "RCUR",
               "frequency": {
                    "type": "DAIL"
               },
               "duration": {
                    "from_date": "2022-08-01",
                    "to_date": "2023-08-01"
               },
               "first_collection_date": "2022-08-01",
```

```
            "final_collection_date": "2023-08-01"
        },
        "tracking_indicator": false,
        "maximum_amount": {
            "amount": "1000",
            "currency": "NOK"
        },
        "creditor": {
            "name": "Aarnes Badekarforhandler AS",
            "identification": {
                "organisation_identification": {
                    "other": {
                        "identification": "916960190",
                        "scheme_name": {
                            "proprietry": "AGREEMENT_ID"
                        }
                    }
                }
            }
        },
        "creditor_account": {
            "identification": {
                "other": {
                    "identification": "60013232345",
                    "scheme_name": {
                        "code": "BBAN"
                    }
                }
            }
        },
        "debtor": {
            "name": "Ole Olsen",
            "identification": {
                "organisation_identification": {
                    "other": {
                        "identification": "996739848",
                        "scheme_name": {
                            "proprietry": "AGREEMENT_ID"
                        }
                    }
                }
            },
            "contact_details": {
                "preferred_method": "MAIL"
            }
        },
        "debtor_account": {
            "identification": {
                "other": {
                    "identification": "60013312349",
                    "scheme_name": {
                        "code": "BBAN"
                    }
                }
            }
        },
```

```
            "mandate_reference": "01234567890",
            "xmlns": "urn:iso:std:iso:20022:tech:xsd:pain.009.001.07"
      }
}
```

This will create a signature input string of:

```
"@request-target": /bank/getPendingPaymentsForMandate/9999
"@method": POST
"@authority": api.testno21.no
"x-request-id": 294fafb7-0e4e-4177-a5ff-ce7367c45814
"client-name": Aarnes Badekarforhandler AS
"requester-merchant": AB-2150
"content-digest": sha-256=:8GPIvXEJbCWeEPAMF55OhE307xO5BzkJye+6eDfsqGk=:
"@signature-params": ("@request-target" "@method" "@authority" "x-request-
id" "client-name" "requester-merchant" "content-
digest");created=1668500614;keyid="75wGcKK8tMqzqN5qbg4bs9g5rYU";alg="rsa-
pss-sha512”
```

### 6.1.5 Signing the signature input string

The signing-string can then be used to create the signature parameter of the message using the senders private-key. For the signing-string above signed with the Example Private Key (listed in chapter 6.3.1) using RSASSA-PSS using SHA-512, this will produce:

```
KZOuBUwVZo9QKTmKi0fJqgt2Y8TdaJqdQEX3KcNr4PNHN1jXbdIQEltyiWIUhhtiOYaYZZBmx6H
lRAPGaShOpJMSsHcKOz8G4nIyLqVx6VkFA2NE/r/vI24U516chUn0TP6y8NaHbaqS37NY6P8vjl
fadS/Uxi6bKMH44R1L4Zv+rxErHMD3s7LHtsbPuxllHt9wTAbirxmL6jzL0IUzpnKvb8k1XtuQn
VvVweyiTOwRQdSvONj3zt6f/oT9vk3sJwk+Ilxg9RnedCN4Jn4JcvNrxQpHn69wCzuqBhrqdLci
Oy0JyVQf3TrNsO8iHhzOqptfA5mge2+eCkMyATA/qA==
```

### 6.1.6 Sending the http message

Now that we have signed the http message, we can construct the complete http message. We do this by taking our original message and adding three new HTTP headers, which will house our signature, and the necessary information for our recipient to verify it.

The three HTTP headers which will be added are:

- Content-Digest: This will contain the digest that we created of the message body that was used in the signature.
- Signature-Input: This will contain the value of the "@signature-params" that was used in the signature.
- Signature: This will contain the signature.

**NB!** *[HTTP Message Signatures] supports sending multiple signatures in a single HTTP message. This is done by assigning each signature a name. This will not be used for the APIs covered by this specification. Signatures from Fullmaktsregisteret will always contain a single signature with the name "sig1". This will be added in both the Signature-Input and Signature headers.*

With this a complete and signed HTTP-message can be constructed:

```
POST /mandate HTTP/1.1
Host: api.autogiro.no
Content-Length: 1091
X-Request-ID: 294fafb7-0e4e-4177-a5ff-ce7367c45814
Client-Name: Aarnes Badekarforhandler AS
Requester-Merchant: AB-2150
Content-Digest: sha-256=:8GPIvXEJbCWeEPAMF55OhE307xO5BzkJye+6eDfsqGk=:
Signature-Input: sig1=("@request-target" "@authority" "x-request-id"
"client-name" "requester-merchant" "content-
digest");created=1664524579;keyid="75wGcKK8tMqzqN5qbg4bs9g5rYU";alg="rsa-
pss-sha512"
Signature: sig1=:
UAOuJmjdM7TYzSbI71IFa+iFgm1V/mNmkf4tnPH60rs7BBNTHNoU/AYp+l/Vxuc5Buc1WvncqUx
G1FdXR3oMNO8ZLNbO857rsUJEUJ7enau/EikEdaiFdbXHm9Vri9+rX9wlbsQFBDp7+n92AeSJyq
1qW6o84Ynsi/khK7KzNb4/S89v0/UVjHb5bpVINGxmFcd0x2BwXfAGEus1U49jrUYaxbLC9yztZ
qrluwbNpuhPf1+yXu83dU+NAkJiPEcfNdKbs6RP1qcbuAcbQbEDyRm9fwL61ZFl41ZSMgqq40o0
TQjhWkBhgR4Z1nfMOgOLeyt62E4oHrNclNhOkqPSLg==:
{
      "mandate": {
            "mandate_request_identification": "NOTASSIGNED",
            "type": {
                  "classification": {
                        "code": "FIXE"
                  }
            },
            "occurrences": {
                  "sequence_type": "RCUR",
                  "frequency": {
                        "type": "DAIL"
                  },
                  "duration": {
                        "from_date": "2022-08-01",
                        "to_date": "2023-08-01"
                  },
                  "first_collection_date": "2022-08-01",
                  "final_collection_date": "2023-08-01"
            },
            "tracking_indicator": false,
            "maximum_amount": {
                  "amount": "1000",
                  "currency": "NOK"
            },
            "creditor": {
                  "name": "Aarnes Badekarforhandler AS",
                  "identification": {
                        "organisation_identification": {
                              "other": {
                                    "identification": "916960190",
                                    "scheme_name": {
                                          "proprietry": "AGREEMENT_ID"
                                    }
                              }
                        }
                  }
```

```
            },
            "creditor_account": {
                  "identification": {
                        "other": {
                              "identification": "60013232345",
                              "scheme_name": {
                                    "code": "BBAN"
                              }
                        }
                  }
            },
            "debtor": {
                  "name": "Ole Olsen",
                  "identification": {
                        "organisation_identification": {
                              "other": {
                                    "identification": "996739848",
                                    "scheme_name": {
                                          "proprietry": "AGREEMENT_ID"
                                    }
                              }
                        }
                  },
                  "contact_details": {
                        "preferred_method": "MAIL"
                  }
            },
            "debtor_account": {
                  "identification": {
                        "other": {
                              "identification": "60013312349",
                              "scheme_name": {
                                    "code": "BBAN"
                              }
                        }
                  }
            },
            "mandate_reference": "01234567890",
            "xmlns": "urn:iso:std:iso:20022:tech:xsd:pain.009.001.07"
      }
}
```

## 6.2 Signatures on responses from Fullmaktsregisteret

Responses to request to Fullmaktsregisteret will be signed (unless the response is an error-response). These signatures will be created in the same way as demonstrated in the previous chapter but including and excluding some signature components. Notably the signatures on responses from Fullmaktsregisteret will contain a signature of the HTTP response, but they will not include all the header elements of a request.

## 6.3 Certificate and key

Below is the information that was used to generate the examples. These are just throw-away keys and self-signed certificates for demonstration purposes, feel free to test the examples, they are generated examples, and should validate (although we make no guarantees that they will, please contact the author if they are discovered to be incorrect):

### 6.3.1 Example certificate:

```
-----BEGIN CERTIFICATE-----
MIIEITCCAwmgAwIBAgIUbb2TqWjRUIpDlMMc5/OH4JnXorEwDQYJKoZIhvcNAQEL
BQAwgZ8xCzAJBgNVBAYTAk5PMQ0wCwYDVQQIDARPc2xvMQ0wCwYDVQQHDARPc2xv
MRAwDgYDVQQKDAdCaXRzIEFTMREwDwYDVQQLDAhCaXRzIE9USTElMCMGA1UEAwwc
QXZ0YWxlR2lyby1NYW5kYXRlcy1FeGFtcGxlczEmMCQGCSqGSIb3DQEJARYXa3Jp
c3RvZmZlci5ob2xtQGJpdHMubm8wHhcNMjExMDAxMDkxMDQ4WhcNMjIxMDAxMDkx
MDQ4WjCBnzELMAkGA1UEBhMCTk8xDTALBgNVBAgMBE9zbG8xDTALBgNVBAcMBE9z
bG8xEDAOBgNVBAoMB0JpdHMgQVMxETAPBgNVBAsMCEJpdHMgT1RJMSUwIwYDVQQD
DBxBdnRhbGVHaXJvLU1hbmRhdGVzLUV4YW1wbGVzMSYwJAYJKoZIhvcNAQkBFhdr
cmlzdG9mZmVyLmhvbG1AYml0cy5ubzCCASIwDQYJKoZIhvcNAQEBBQADggEPADCC
AQoCggEBAKJgKtqpb4JeqKxnuBb6yR5yRnip7+sSooi5H/og4DM3vDmM2Ly3CWfk
W/lIoEHsnVYFBb64ReQYYeJdvXrlDpqWAzp4BdQdrYzhfc0GgFP3tSA9kMKIvZNm
akfpskxK1A5/JMA/YEVWJrHxrqfKZeuRdVPKAre3uYTG1sff5ZCFhNJfCynHNc9B
u3zMV6tMJgXu7L4UTff0Uu47Ngmp6ZdM2DsgSMa7ZtWlo7tO+GuJY6QabyLGIxVd
m0qVoCeFwD7kh5gzGxOVbI03Jz0ubxjAv8BKcqA5IviCAFTRrOu328TmjCyCyEXM
rT7tQhKrfWm8Qg2ZL8oeCI9B5E6mNMECAwEAAaNTMFEwHQYDVR0OBBYEFB7O0X55
Vrr2IB0qibnsSIKj8s+9MB8GA1UdIwQYMBaAFB7O0X55Vrr2IB0qibnsSIKj8s+9
MA8GA1UdEwEB/wQFMAMBAf8wDQYJKoZIhvcNAQELBQADggEBAE9NCfKuLO2kb4r/
gNVwXNW8pDjdiJOFFfmDFHjF61taH1vkc4iElM2aSHJbhPpFEL1xc23q0zSRqyCQ
UpgVF832cGxrKhvb2SRxmyo6ZH4FWvaTn1NyiRY+NG9MJYs788jHshFW05Q96mMg
sxOhQDukxQ80wawhuGb+nj2Iv+0MlVgMeESIms2xikq8RMD6GE+WIWto3/kvKZGU
xkn7keoncAhFKVA++ZMdDHiNaM1WXOzg419x9WrMjq3IMBBC87+Hg9lUq8JmTZyp
iUQc98mkuqTkNFpdiaMfcVWGQ1XZrfUv9a8G/e6R491Bb41OKEvaAm+Bh2iwmWz/
dDm+abM=
-----END CERTIFICATE-----
```

### 6.3.2 Example private key:

```
-----BEGIN PRIVATE KEY-----
MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQCiYCraqW+CXqis
Z7gW+skeckZ4qe/rEqKIuR/6IOAzN7w5jNi8twln5Fv5SKBB7J1WBQW+uEXkGGHi
Xb165Q6algM6eAXUHa2M4X3NBoBT97UgPZDCiL2TZmpH6bJMStQOfyTAP2BFViax
8a6nymXrkXVTygK3t7mExtbH3+WQhYTSXwspxzXPQbt8zFerTCYF7uy+FE339FLu
OzYJqemXTNg7IEjGu2bVpaO7TvhriWOkGm8ixiMVXZtKlaAnhcA+5IeYMxsTlWyN
Nyc9Lm8YwL/ASnKgOSL4ggBU0azrt9vE5owsgshFzK0+7UISq31pvEINmS/KHgiP
QeROpjTBAgMBAAECggEAImJm8MLsgBj3cvrLuuIEcNQWJDsoOQlLLdS19su7bl0h
GLbAtsWz0jJDX7iHZy5p6utJWie/dRvMrpjXJQ0YWJfnuxvrcA2Q0MJ3V1FHH4DW
9CrVWryGGI6Zdvz/6rP1z9QQvj0tb8FclFXvfEyZ5JZ61/FxPeJEAN/yX4UEIeQt
7fuLTMYx4Ou6iwsYMkt5iFapIQmiyZDoMrURFWTT5fX15I5s4GIi8uxzEFC7V3J4
PoYwqbHlHCuCmoDlIPPZ0Oguboswi6CbE7NXwwSRAgFeu4uisuc2a0V5J08I4hXp
fI/+FHzxPbBnSbwrvHoTeH2XjYRaOxjeaqvDuhiGgQKBgQDPM7Jfu0hXZiLurcKf
s6wvrxUNqL/GkQZnWDv9NER8rqyCrNTk0B/wuKk5hRUbhHP4I57NjJEDk1zuj8h6
AMO0Zl5fnjDz86wkJ1JxUG9GjYJw17YAcesq30TBz8YjIuAds/YeZlhjKJgBK32G
+nCtFzqAsRv7Blg0NN/BNjcLmQKBgQDInd9fr0CB+ShUC0D/jQKnbcIMZLWr1IHK
nFspvGJpy46qShLYFlp35sO18FrHGHydeTT/CeS38rxolpTuVdC3KZ1fvs1gOEa4
Nv6oAm8Gxr6lJbRkCnuxJQKTngSivYnxcwKa7BKaolJtPqXmP1b2yoltoXsIuYh8
TFLkmfXraQKBgQCscKvMvEKyahA8b2QAITn13VI6MeyYxu17aJVUXwF4eq6belcb
rpJGdohvlHBCnHMfWhW5n3i4bxXyfLstviEhq+hyz2aSQINM+2TDZVuWbgLXAs83
g5dh8Lp6Sf7uEwJN9g2osyhwLcKDhrxLb3YCt8g6fit5OIiDulVVqVcaUQKBgBzZ
33LyUDOg8nTLvYhC7jvH5B1GKn5QrG3H+LBS4FBYrua8imM7K72MmV7D9zokwl8f
PEjlUlhF92SPK1HvU3nT6UcIuJ562WjKt+rPlsoBsQ8tEflFLK64JNu34PDKk1j+
8kP3aWGFsJb3aIJpX4dUb1kt3PTPQdqmW1F29s/pAoGAP6EW2cIzvgz/0Z8W5xGH
bZMdNvkV0FfHZ8c6O5YH4xe9p6HReqaKzEDx+/KOCN7YTCsTOrrvQNym3nJ6M01i
j09IsL/cPszz46Ss/Ngpcfj0xVGw5HxYfktoClmQI8/eGKleQmFTuh+GBisCcZuI
lBN63QJchPhYffB+j7ECTZs=
-----END PRIVATE KEY-----
```