Logius Standard - Draft

API Design Rules Module: Signing

Logius Standard
Draft December 15, 2025

This version:
https://logius-standaarden.github.io/API-mod-signing/

Latest published version:
https://logius-standaarden.github.io/API-mod-signing/

Latest editor's draft:
https://logius-standaarden.github.io/API-mod-signing/

Editor:
Logius Standaarden (Logius)

Author:
Peter Haasnoot (Logius)

Participate:
GitHub Logius-standaarden/API-mod-signing
File an issue
Commit history
Pull requests

This document is also available in these non-normative format: PDF

Status of This Document

This is a draft that could be altered, removed or replaced by other documents. It is not a recommendation approved by TO.

Table of Contents

§ Conformance

As well as sections marked as non-normative, all authoring guidelines, diagrams, examples, and notes in this specification are non-normative. Everything else in this specification is normative.

The key words *MUST* and *SHOULD* in this document are to be interpreted as described in BCP 14 [*RFC2119*] [*RFC8174*] when, and only when, they appear in all capitals, as shown here.

Abstract

This document is part of the *Nederlandse API Strategie*.

The Nederlandse API Strategie consists of a set of distinct documents.

Status	Description & Link
Informative	Inleiding NL API Strategie
Informative	Architectuur NL API Strategie
Informative	Gebruikerswensen NL API Strategie
Normative	API Design Rules (ADR v2.0)
Normative	Open API Specification (OAS 3.0)

Status	Description & Link
Normative	NL GOV OAuth profiel
Normative	Digikoppeling REST API koppelvlak specificatie
Normative module	GEO module v1.0

Before reading this document it is advised to gain knowledge of the informative documents, in particular the Architecture.

This ADR Module contains the requirements for ADR-HTTP Message and payload signing with *JAdES digital signatures*.

This module is based on the *ISA² IPS REST API Profile v1.0 section 5.2.2 Message And Payload Level Security*.

§ 1. Introduction

This module specifies the use of JAdES signatures for HTTP message and payload siging. The module is directly based on the *ISA² IPS REST API Profile v1.0* (which was a result of the REST API Pilot project for eDelivery)

§ 1.1 Use Cases

This module is applicable when there is a need for assurance of end to end message integrity and authenticity between client application and server application. In the context of HTTP messages signing header elements (for example HTTP operation (GET/POST/UPDATE/DELETE) and resource path / parameters, or timestamps) together with payload provides an extra level of protection against manipulation of the message.

In a complex IT landscape the path between client and server can go over several intermediary components/systems in which case end to end integrity and authenticity can be especially relevant. (In this case TLS is terminated in each step on the path and does not protect the http-message in transport fully end to end).

§ 1.2 JWS detached signatures

This module enforces the use of JWS detached signatures following the HTTP Headers Mechanism of the ETSI ESI JAdES specification [*JAdES*].

This structure is enforced for the following reasons:

- JWS, being a simple JSON Structure, can be supported by clients in a light context, while specifications like the ETSI ESI ASIC containers are more difficult to do.
- JWS in detached form does not change the payload structure, meaning that a client not supporting the validation of signature can continue to operate as if there was no signature applied.
- JWS Detached can be transported using an HTTP header, making its presence unintrusive and easily transportable.

§ 2. Normative Design Rules

§ 2.1 Summary

Design rules are technical rules, which should be tested automatically.

List of technical rules

- /signing/algorithm: Support the RSASSA-PSS Algorithm
- /signing/payload: Apply JAdES detached signatures for payload signing
- /signing/message: Apply JAdES HttpHeaders Mechanism for message-level security

§ 2.2 Cryptographic Algorithms

> Technical
>
> **/signing/algorithm**: Support the RSASSA-PSS Algorithm

Statement

The RSASSA-PSS Algorithm *MUST* be supported, with a key length of at least 256 bits. The value `PS256` for the `alg` parameter *MUST* be used as defined in [*RFC7518*].

§ 2.3 Payload signing

Technical

/signing/payload: Apply JAdES detached signatures for payload signing

Statement

Payload signing ensures the integrity and authenticity of the payload part of the message. When payload signing is considered, the Detached JSON Web Signatures following the JAdES specification [*JAdES*] *MUST* be applied with the following restrictions:

- The JWS content (Data to be Signed) *MUST* be detached from the signatures as defined in [*RFC7515*] Appendix F.
- The signed SigD parameter object *MUST* be present in the JWS headers, denoting the use of the JAdES detached header profile.
- The value of the `mId` parameter *MUST* be set to `http://uri.etsi.org/19182/HttpHeaders`.
- The `pars` array of the SigD *MUST* contain only the element `digest`, denoting that for the calculation of the signature only the digest of the HTTP payload must be taken into account, according to [*RFC3230*].
- The `alg` parameter *MUST* be set to the correct value depending on the algorithm used (see 2.2 Cryptographic Algorithms).

The JWS structure shall be carried in HTTP header field named `Payload-Signature`. The header field can be used in both requests and responses. The header field *MUST* not appear more than once in a message; if a message contains multiple `Payload-Signature` header fields, the receiver *MUST* consider the signature invalid.

§ 2.4 Message signing

The Introduction section of [*RFC9421*] details why message integrity and authenticity are critical to the secure operation of many HTTP/REST applications.

Technical

/signing/message: Apply JAdES HttpHeaders Mechanism for message-level security

Statement

When Message-Level Security is considered, the HttpHeaders Mechanism of the JAdES Specification [*JAdES*] *MUST* be used, with the following restrictions applied.

- The JWS content (Data to be Signed) *MUST* be detached from the signatures as defined in [*RFC7515*] Appendix F.
- The signed SigD parameter object *MUST* be present in the JWS headers, denoting the use of the JAdES detached header profile.
- The value of the mId parameter *MUST* be set to `http://uri.etsi.org/19182/HttpHeaders`.
- The pars array of the SigD *MUST* contain at least the following elements:
  - the element `(request-target)`, for containing the HTTP Request URI
  - the element `host`, for containing the host the message was submitted to, if present
  - the element `origin`, for containing the scheme, hostname, and port from which the request was initiated, if present
  - the element `content-encoding`, if present
  - the element `content-type`, if present
  - the element `content-length`, if present
  - the element `digest`, for taking into account the Digest header that contains the hash value of the HTTP payload.
- The alg parameter *MUST* be set to the correct value depending on the algorithm used (see above).

Implementations that make use of the HTTP Header fields for data representation *SHOULD* also include these header fields in the pars array. The JWS structure *MUST* be carried in HTTP header field named `Message-Signature`. The header

> field can be used in both requests and responses. The header field *MUST* not appear more than once in a message; if a message contains multiple `Message-Signature` header fields, the receiver *MUST* consider the signature invalid.

§ A. Signature Representations

This section is non-normative.

```
openapi: 3.1.0
info:
    title: JAdES Signatures
    summary: An example showcasing JAdES signatures
    description: An example showcasing JAdES signatures as JWS detached s
    termsOfService: https://domain.server.io/terms-of-service
    license:
        name: EUPL-1.2 or later
        url: https://eupl.eu/1.2/en/
    version: 1.0.0
externalDocs:
   description: The ISA² IPS REST API Core Profile
   url: https://joinup.ec.europa.eu/collection/api4dt/document/isa2-ips-r
servers:
- url: https://domain.server.io/v2
tags:
- name: DetachedPayloadSignature
  description: Operations using payload security
- name: DetachedMessageSignature
  description: Operations using message-level security
paths:
    /openapi:
        get:
            summary: Returns the OpenAPI Document for the API
        ...
        responses:
        200:
            description: ...
            content: {
               $ref: 'https://spec.openapis.org/oas/3.1/schema/2021-05-20
               ...
            }
    /certificate:
    get:
      tags:
      - DetachedMessageSignature
      summary: Get a Certificate
      securitySchemes:
        OAuth2:
           type: oauth2
        flows:
           authorizationCode:
              authorizationUrl: https://example.com/api/oauth/dialog
              scopes:
                 send:message: send a message
       ...
      responses:
      200:
         headers:
            nlgov-adr-message-sig:
               $ref: '#/components/headers/nlgov-adr-message-sig'
          description: List of Certificates
          content: { ... }
components:
   headers:
      nlgov-adr-payload-sig:
         schema:
             $ref: '#/components/schemas/JwsCompactDetached'
      nlgov-adr-message-sig:
         schema:
             $ref: '#/components/schemas/JwsCompactDetached'
   schemas:
      JwsCompactDetached:
         title: The format for the message-level and payload signature
         description: Defines the string pattern as a regular expression
         MUST be followed to represent detached JWS compact tokens
        "$id": https://raw.githubusercontent.com/isa2-api4ips/rest-api-pr
        "$schema": https://json-schema.org/draft/2020-12/schema
         type: string
         format: jws-compact-detached
         pattern: ^[A-Za-z0-9_-]+(?:(\\.\\.)[A-Za-z0-9_-]+){1}
```

§ B. Tooling

This section is non-normative.

§ B.1 Digital Signature Service

The DSS (Digital Signature Service) project is an open-source software library, aimed at providing implementation of the standards for Advanced Electronic Signature creation, augmentation and validation in line with European legislation and the eIDAS Regulation in particular.

https://ec.europa.eu/digital-building-blocks/DSS/webapp-demo/doc/dss-documentation.html

§ B.2 JWT, JWS, JWE, JWK, and JWA Implementations

Libraries implementing JWT and the JOSE specs JWS, JWE, JWK, and JWA are listed here.

https://openid.net/developers/jwt-jws-jwe-jwk-and-jwa-implementations/

§ C. References

§ C.1 Normative references

[JAdES]
JAdES digital signatures. URL: https://www.etsi.org/deliver/etsi_ts/119100_119199/11918201/01.01.01_60/ts_11918201v010101p.pdf

[RFC2119]
Key words for use in RFCs to Indicate Requirement Levels. S. Bradner. IETF. March 1997. Best Current Practice. URL: https://www.rfc-editor.org/rfc/rfc2119

[RFC3230]
Instance Digests in HTTP. J. Mogul; A. Van Hoff. IETF. January 2002. Proposed Standard. URL: https://www.rfc-editor.org/rfc/rfc3230

[RFC7515]
JSON Web Signature (JWS). M. Jones; J. Bradley; N. Sakimura. IETF. May 2015. Proposed Standard. URL: https://www.rfc-editor.org/rfc/rfc7515

[RFC7518]
JSON Web Algorithms (JWA). M. Jones. IETF. May 2015. Proposed Standard. URL: https://www.rfc-editor.org/rfc/rfc7518

[RFC8174]
Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words. B. Leiba. IETF. May 2017. Best Current Practice. URL: https://www.rfc-editor.org/rfc/rfc8174

[RFC9421]

HTTP Message Signatures. A. Backman, Ed.; J. Richer, Ed.; M. Sporny. IETF. February 2024. Proposed Standard. URL: https://www.rfc-editor.org/rfc/rfc9421

↑