Logius Standard - Draft

FSC - Properties extension

Logius Standard
Draft March 09, 2026

This version:
https://logius-standaarden.github.io/fsc-properties/

Latest published version:
https://gitdocumentatie.logius.nl/publicatie/fsc/properties/

Latest editor's draft:
https://logius-standaarden.github.io/fsc-properties/

Editors:
VNG Realisatie (VNG)
Logius (Logius)

Authors:
Eelco Hotting (Hotting IT), Email
Ronald Koster (PhillyShell), Email
Henk van Maanen (AceWorks), Email
Niels Dequeker (ND Software), Email
Edward van Gelderen (vanG IT), Email
Pim Gaemers (Apily), Email

Participate:
GitHub Logius-standaarden/fsc-properties
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 *MAY* and *MUST* 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 specifies an extension to the Federated Service Connectivity (FSC) Core specification, introducing the ability to include custom properties in grants. The extension defines a flexible mechanism for adding key-value pairs to grants, which are then propagated through access tokens and made available to services.

Key features of this extension include:

1. A `properties` object within grants, allowing for arbitrary JSON key-value pairs.
2. Integration of the `properties` object into the grant hash calculation process.
3. Inclusion of properties in access tokens as a new `prp` claim.
4. Guidelines for Manager, Inway, and Outway behavior when handling properties.
5. Security considerations, including size limitations and sensitive data handling.

This extension maintains backwards compatibility with the FSC Core specification while enabling more detailed and flexible grant definitions. It opens up possibilities for application-specific metadata and configuration options within the FSC ecosystem, enhancing the capabilities of federated service connectivity implementations.

1. Architecture

This section describes how the property object fits into the existing FSC architecture.

1.1 Grant Properties

Each grant in a Contract can now contain an optional property object. This object is included in the grant's data and is part of the content that is hashed when creating the grant hash and contract content hash.

The property object can contain one or multiple keys, with values of any valid JSON datatype. This flexibility allows for a wide range of use cases and future extensibility.

Example of a ServiceConnectionGrant with a property object:

```
{
  "data": {
    "type": "GRANT_TYPE_SERVICE_CONNECTION",
    "service": {
      "peer_id": "00000000000000000001",
      "name": "example-service"
    },
    "outway": {
      "peer_id": "00000000000000000002",
      "public_key_thumbprint": "3a56f2e9269ac63f0d4394c46b96539da1625b6a9
```

```
    },
    "properties": {
      "max_requests_per_minute": 100,
      "tags": ["api", "v1", "beta"],
      "custom_metadata": {
        "department": "IT",
        "project": "FSC Integration"
      }
    }
  }
}
```

2. Specifications

This section provides detailed specifications for implementing the property object extension.

2.1 Property Object

2.1.1 Structure

The property object must be a valid JSON object. It can contain any number of key-value pairs, where:

1. Keys must be strings
2. Values can be of any valid JSON datatype (string, number, boolean, object, array, or null)

2.1.2 Grant

The property object *MAY* be added to a grant object and *MUST* have the key 'properties'.

§ 2.1.3 Inclusion in Grant Hash

When calculating the grant hash, the property object must be included in the hashing process. This ensures that the properties are part of the immutable grant definition.

To include the property object in the grant hash:

1. Canonicalize the JSON property object as described in RFC 8785
2. Hash the canonicalized JSON string using the hash algorithm described in contract.content.algorithm
3. Add the hash to the grantBytes array as described in the FSC Core specification.
4. Continue with the hashing process as described in the FSC Core specification.

§ 2.1.4 Validation

When validating a grant with properties:

1. The Manager must ensure that the `properties` field, if present, is a valid JSON object.
2. The Manager should not impose any restrictions on the content of the property object beyond it being valid JSON, as the meaning and use of properties are application-specific.

§ 2.1.5 Security Considerations

1. The size of the `properties` object should be limited to prevent excessive data transfer and storage. It is recommended to implement a size limit of 1MB for the serialized `properties` object.

2. Sensitive information like secrets or other private information should not be stored in the `properties` object, as it may be visible to Services and potentially logged or stored in various systems.

3. Implementers should be aware that the contents of the `properties` object are unsanitized. For example, they should consider sanitizing the data before showing it in user interfaces to prevent XSS injections or other security vulnerabilities.

§ 2.1.6 Backwards Compatibility

This extension is backwards compatible with the FSC Core specification. Systems that do not implement this extension will ignore the `properties` field in grants and the `prp` claim in access tokens.

§ 2.1.7 Future Considerations

1. Standard property keys and their meanings could be defined in future versions to promote interoperability between different implementations.

§ A. References

§ A.1 Normative references

[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

[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

↑