Guidelines for NL-GOV profile CloudEvents

Project Notificatieservices

Logius Handreiking
Werkversie 28 oktober 2022

Deze versie:
https://logius-standaarden.github.io/CloudEvents-NL-Guidelines/

Laatst gepubliceerde versie:
https://publicatie.centrumvoorstandaarden.nl/notificatieservices/CloudEvents-NL-Guidelines/

Laatste werkversie:
https://logius-standaarden.github.io/CloudEvents-NL-Guidelines/

Redacteurs:
Gershon Jansen (VNG Realisatie)
Ad Gerrits (VNG Realisatie)
Edwin Wisse (Logius)

Auteur:
Werkgroep Berichtenstandaard (Project Notificatieservices)

Doe mee:
GitHub Logius-standaarden/CloudEvents-NL-Guidelines
Dien een melding in
Revisiehistorie
Pull requests

Samenvatting

Abstract

Status van dit document

Dit is een werkversie die op elk moment kan worden gewijzigd, verwijderd of vervangen door andere documenten. Het is geen door het Technisch Overleg goedgekeurde consultatieversie.

Inhoudsopgave

Samenvatting

Status van dit document

1. Guideline for the use of the HTTP Protocol Binding for CloudEvents
1.1 Summary with points for attention
1.2 Recommendations
1.3 Examples
1.4 Normative References

2. Guideline for the use of the JSON Event Format for CloudEvents

2.1 Summary with points for attention
2.2 Recommendations
2.3 Examples
2.4 Normative References

3. Guideline for the use of the Webhook pattern for CloudEvents
3.1 Summary with points for attention
3.1.1 Delivering notifications
3.1.2 Authorization
3.1.3 Abuse protection
3.2 Recommendations
3.3 Examples
3.4 Normative References

4. Conformiteit

A. Index
A.1 Begrippen gedefinieerd door deze specificatie
A.2 Begrippen gedefinieerd door verwijzing

1. Guideline for the use of the HTTP Protocol Binding for CloudEvents

The CloudEvent-NL message format can be used when using different formats and protocols and patterns. In order to be able to use the GOV-NL profile properly in practice, agreements must also be made when and in what way a certain format, protocol or pattern is used.

The Serverless Working Group has described how the HTTP protocol can be used in a standardized way in combination with the CloudEvents message format: HTTP Protocol Binding for CloudEvents. The specification defines how the elements defined in the CloudEvents specification are to be used in HTTP 1.1 requests and response messages.

Similar to what happened in the NL GOV profile for the CloudEvents specification, this guideline makes recommendations about the use of the HTTP specification within the context of the Dutch government. These are intended to make use of the specification more unambiguous and to work towards standardisation in the long term.

1.1 Summary with points for attention

- Events can be transferred with all standard or application-defined HTTP request methods that support payload body transfers. Events can be also be transferred in HTTP responses and with all HTTP status codes that permit payload body transfers.
- This specification defines three content modes for transferring events:
  - structured (required): HTTP message body contains both event data and metadata attributes; an appropriate event format is used ((non-batching) JSON is the only event format that MUST be supported);
  - batched (optional): HTTP message body contains event data and metadata attributes from multiple events; an appropriate event format is used.
  - binary (required): HTTP message body contains event data as-is; event attributes mapped to HTTP-headers; HTTP `Content-Type` header declares the media type.
- Received `Content-Type` header value is:

    - `application/cloudevents(+xxxx)`: structured mode (`xxx` denotes the used event format)
    - `application/cloudevents-batch`: batched mode
    - Otherwise: binary mode (`Content-Type` header declares the media type.)
- Structured content mode:
    - The HTTP message body MUST contain both event data and metadata attributes.
    - The HTTP Content-Type header MUST be set to the media type of an event format (E.g. `application/cloudevents+json; charset=UTF-8`)
    - The chosen event format defines how all attributes, and data, are represented in the HTTP message body.
    - Implementations MAY include the same HTTP headers as defined for the binary mode (E.g.`ce-id`).
- Batched content mode:
    - The HTTP message body MUST contain both event data and metadata attributes.
    - The HTTP Content-Type header MUST be set to the media type of an event format (E.g. `application/cloudevents-batch+json; charset=UTF-8`)
    - The chosen event format defines how a batch of events and all event attributes, and data, are represented.
    - All batched CloudEvents MUST have the same specversion attribute. Other attributes MAY differ, including the datacontenttype attribute. The batch MAY be empty.
- Binary content mode:
    - All CloudEvents context attributes, including extensions, MUST be mapped to HTTP headers with the same name as the attribute name but prefixed with `ce-`.
    - The value for each HTTP header is constructed as described in thespecification

At the moment there are *no additional agreements* about the use of the specification within the Dutch government.

1.2 Recommendations

- One SHOULD use the HTTP Protocol Binding for CloudEvents.
- There are no agreements to deviate from the standard in any part.
- Given the fact that many organizations still have little experience with standardized exchange of events, we recommend a useful relatively simple mechanism that consumers are already familiar with:
    - One SHOULD use structured content mode.
    - One SHOULD use the JSON (non batched) event format
    - When using structured mode one SHOULD NOT depend on the usage of HTTP-headers that replicate context-attributes in the event.
- If the above advice is followed, when notifying via webhooks 1 message will contain JSON-structured data about 1 event that has occurred.

1.3 Examples

Structured content mode: HTTP POST request with a JSON event format encoded event:

```
POST /myresource HTTP/1.1
Host: webhook.example.com
Content-Type: application/cloudevents+json; charset=utf-8
Content-Length: nnnn

{
  "specversion": "1.0",
  "type": "nl.overheid.zaken.zaakstatus-gewijzigd",
  "source": "urn:nld:oin:00000001823288444000:systeem:BRP-component",
  "id": "f3dce042-cd6e-4977-844d-05be8dce7cea",
   ... further attributes omitted ...
  "data": {
        ... application data ...
  }
}
```

Batched content mode: HTTP POST request with two JSON event format encoded events:

```
[
    {
        "specversion": "1.0",
        "type": "nl.overheid.zaken.zaakstatus-gewijzigd",
        "source": "urn:nld:oin:00000001823288444000:systeem:BRP-component",
        "id": "f3dce042-cd6e-4977-844d-05be8dce7cea",
        ... further attributes omitted ...
        "data": {
                ... application data ...
        }
    },
    {
        "specversion": "1.0",
        "type": "nl.overheid.zaken.zaakstatus-gewijzigd",
        "source": "urn:nld:oin:00000001823288444000:systeem:BRP-component",
        "id": "1ca55552-bc4a-4f5d-8cc8-8106e3e883c1",
        ... further attributes omitted ...
        "data": {
                ... application data ...
        }
    }
]
```

Binary content mode: HTTP POST request with context attributes mapped to HTTP-headers:

```
POST /myresource HTTP/1.1
Host: webhook.example.com
ce-specversion: 1.0
ce-type: nl.overheid.zaken.zaakstatus-gewijzigd
ce-id: f3dce042-cd6e-4977-844d-05be8dce7cea
ce-source: urn:nld:oin:00000001823288444000:systeem:BRP-component
... further attributes omitted ...
Content-Type: application/json; charset=utf-8
Content-Length: nnnn

{
    ... application data ...
}
```

1.4 Normative References

- HTTP Protocol Binding for CloudEvents - Version 1.0.1

-->

2. Guideline for the use of the JSON Event Format for CloudEvents

The CloudEvent-NL message format can be used when using different formats and protocols and patterns. In order to be able to use the GOV-NL profile properly in practice, agreements must also be made when and in what way a certain format, protocol or pattern is used.

The Serverless Working Group has described how the JavaScript Object Notation (JSON) Data Interchange Format can be used in a standardized way in combination with the CloudEvents message format: JSON Event Format for CloudEvents.

Similar to what happened in the NL GOV profile for the CloudEvents specification, this guideline makes recommendations about the use of the JSON specification within the context of the Dutch government. These are intended to make use of the specification more unambiguous and to work towards standardisation in the long term.

The JSON Format for CloudEvents specification defines how events are expressed in JSON. The JSON syntax is not a specification of a complete data interchange. Meaningful data interchange requires agreement between a producer and consumer on the semantics attached to a particular use of the JSON syntax. What JSON does provide is the syntactic framework to which such semantics can be attached.

2.1 Summary with points for attention

- Each CloudEvents event can be wholly represented as a JSON object and MUST use the media type `application/cloudevents+json`.
- The CloudEvents JSONSchema for the spec can be used to validate events in JSON.
- Unset attributes MAY be encoded to the JSON value of null. When decoding attributes and a null value is encountered, it MUST be treated as the equivalent of unset or omitted. OPTIONAL not omitted attributes MAY be represented as a null JSON value.
- The runtime data type of the `data' content inside the JSON object can be either:
  - a textual JSON value with member name `data`
  - a binary JSON string expression containing the Base64 encoded binary value with member name `data_base64`
- So if a `data_base64` member is present it indicates that its value is Base64 encoded binary data.
- JSON Batch Format can be used to batch several CloudEvents into a single JSON document.
  - The document contains a JSON array filled with CloudEvents in the JSON Event format.
  - Media type `application/cloudevents-batch+json` MUST be used.
  - The JSON Batch Format MUST NOT be used when only support for the JSON Format is indicated.

2.2 Recommendations

- One SHOULD use the JSON Event Format for CloudEvents.
- There are no agreements to deviate from the standard in any part.
- If applicable then one SHOULD use JSON as primary event format:
    - JSON is the only event format that MUST be supported when HTTP structured content mode is used.
    - JSON will be the primary format for APIs as formulated within the API strategie voor de Nederlandse overheid - Extensies.

2.3 Examples

Event with an attribute 'geheimnummer' having a null value and a 'data' attribute with a JSON value:

```json
{
  "specversion": "1.0",
  "type": "nl.overheid.zaken.zaakstatus-gewijzigd",
  "source": "urn:nld:oin:00000001823288444000:systeem:BRP-component",
  "subject": "123456789",
  "id": "f3dce042-cd6e-4977-844d-05be8dce7cea",
  "time": "2021-12-10T17:31:00Z",
  "nlbrpnationaliteit": "0083",
  "geheimnummer": null,
  "dataref": "https://gemeenteX/api/persoon/123456789",
  "sequence": "1234",
  "sequencetype": "integer",
  "datacontenttype": "application/json",
  "data": {
    "bsn": "1234567789",
    "naam": "Jan Jansen",
    "gecontroleerd": "ja"
  }
}
```

Event with both the attribute 'datacontenttype' and 'data_base64' and a JSON string expression with Base64 encoded binary data as value of the 'data_base64' attribute:

```json
{
  "specversion": "1.0",
  "type": "nl.overheid.zaken.zaakstatus-gewijzigd",
  "source": "urn:nld:oin:00000001823288444000:systeem:BRP-component",
  "id": "f3dce042-cd6e-4977-844d-05be8dce7cea",
  "datacontenttype": "application/vnd.apache.thrift.binary",
  "data_base64": "YWFwIG5vb3QgbWllcw=="
}
```

Event with only the attribute 'data_base64' and a JSON string expression with Base64 encoded binary data as value of the 'data_base64' attribute:

```json
{
  "specversion": "1.0",
  "type": "nl.overheid.zaken.zaakstatus-gewijzigd",
  "source": "urn:nld:oin:00000001823288444000:systeem:BRP-component",
  "id": "f3dce042-cd6e-4977-844d-05be8dce7cea",
  "data_base64" : "YWFwIG5vb3QgbWllcw=="
}
```

2.4 Normative References

- JSON Event Format for CloudEvents - Version 1.0.1

-->

3. Guideline for the use of the Webhook pattern for CloudEvents

The CloudEvent-NL message format can be used when using different formats and protocols and patterns. In order to be able to use the GOV-NL profile properly in practice, agreements must also be made when and in what way a certain format, protocol or pattern is used.

The Serverless Working Group has described how the Webhook pattern can be used in a standardized way in combination with the CloudEvents message format: HTTP 1.1 Web Hooks for Event Delivery.

"Webhooks" are a popular pattern to deliver notifications between applications and via HTTP endpoints. Applications that make notifications available, allow for other applications to register an HTTP endpoint to which notifications are delivered. In spite of pattern usage being widespread, there is no formal definition for Web Hooks. The CloudEvents specification now provides such a definition for use with CNCF CloudEvents (but is considered generally usable beyond the scope of CloudEvents).

Similar to what happened in the 'NL GOV profile for the CloudEvents' specification, this guideline makes recommendations about the use of the drafted CloudEvents specification within the context of the Dutch government. These are intended to make use of the specification more unambiguous and to work towards standardisation in the long term.

3.1 Summary with points for attention

The 'HTTP 1.1 Web Hooks for Event Delivery' specification prescribes rules constraining the use and handling of specific HTTP methods and headers and defines:

1. a HTTP method by how notifications are delivered by the sender
2. an authorization model for event delivery to protect the delivery target
3. a registration handshake that protects the sender from being abused for flooding arbitrary HTTP sites with requests. For each of the mechanisms it is described which agreements apply to both sender and receiver. The specification

3.1.1 Delivering notifications

- A delivery request MUST use a HTTP POST request via HTTPS.
- A delivery response MUST have the appropriate status code:
  - 200 OK of 200 Created if delivery had been accepted and processed and response carries a payload
  - 201 Created of 204 No Content when accepted and processed but carries no payload
  - 202 Accepted if accepted but not yet processed or processing status is unknown
  - 410 Gone if delivery target has been retired

- 429 Too Many Requests when exceeding a request rate limit and MUST include the Retry-After header.
- 415 Unsupported Media Type wehen notification format is not understood.
- other error status codes apply as specified inRFC7231.

3.1.2 Authorization

The delivery request MUST use one of the following two methods (both of which lean on the OAuth 2.0 Bearer Token RFC6750 model):

- The access token is sent in the Authorization request header field; forOAuth 2.0 Bearer tokens, the "Bearer" scheme MUST be used.
- The access token is added to the HTTP Request URI Query component as described inURI Query Parameter.

3.1.3 Abuse protection

It must be prevented that notifications are sent to recipients who have not requested this themselves. A legitimate delivery target needs to indicate that it agrees with notifications being delivered to it. Reaching the delivery agreement is realized using a validation handshake:

- A handshake can either be executed immediately at registration time or as a "pre-flight" request immediately preceding a delivery.
- Delivery targets SHOULD support the abuse protection feature. If a target does not support the feature, the sender MAY choose not to send to the target, at all, or send only at a very low request rate.
- The *validation request* uses the HTTP OPTIONS method with header fields:
  - WebHook-Request-Origin (required): a DNS expression that identifies the sending system
  - WebHook-Request-Callback (optional): a callback URL that allows the delivery target to grant send permission asynchronously, via a simple HTTPS callback.
  - WebHook-Request-Rate (optional): a positive integer number that expresses the request rate in "requests per minute"
- The *validation respons* MUST be sent if the delivery target does allow delivery of events with header fields:
  - WebHook-Allowed-Origin (required): MUST either be the origin name supplied in the WebHook-Request-Origin header, or a singular asterisk character ('*'), indicating that the delivery target supports notifications from all origins.
  - WebHook-Allowed-Rate (depends): MUST be returned if the request contained the WebHook-Request-Rate, otherwise it SHOULD be returned; an integer number expresses the permitted request rate in "requests per minute" or asterisk when there is no rate limitation.

3.2 Recommendations

- One SHOULD use the HTTP 1.1 Web Hooks for Event Delivery.
- There are no agreements to deviate from the standard in any part.

- As described in the specification usage of an access token added to the HTTP Request URI Query component has a number of security weaknesses and therefore SHOULD NOT be used unless it is impossible to sent an access token in the Authorization request header field.
- The CloudEvents specification focuses on automated validation of intended notification applications. Within the context of the Dutch government, there can (also) be non-automated validation (e.g. by specifying endpoints in agreements between the organizations involved). In those cases it is not always necessary to perform an automated handshake before notifications may be sent.
- An automated handshake as described can take place at different moments. If automated subscription to event notification is used:
  - one SHOULD perform a handshake as described in the specification immediately at registration time
  - one MAY perform a handshake as a "pre-flight" request immediately preceding a delivery.

3.3 Examples

Validation request:

```
OPTIONS /endpoint HTTP/1.1
Host: webhook.example.com
WebHook-Request-Origin: eventemitter.example.com
WebHook-Request-Callback: https://example.com/confirm?id=12345&key=...base64..
WebHook-Request-Rate: 120
```

Validation response:

```
HTTP/1.1 200 OK
Allow: GET,POST,PUT,PATCH,DELETE,HEAD,OPTIONS
Access-Control-Allow-Origin: https://eventemitter.example.com
Access-Control-Allow-Methods: GET,POST,PUT,PATCH,DELETE,HEAD,OPTIONS
Access-Control-Allow-Headers: Content-Type
WebHook-Allowed-Origin: eventemitter.example.com
WebHook-Allowed-Rate: 120
```

Delivery request:

- HTTP structured content mode with a
- JSON event format encoded event with a
- OAuth 2.0 Bearer token

```
POST /myresource HTTP/1.1
Host: server.example.com
Authorization: Bearer mF_9.B5f-4.1JqM
WebHook-Request-Origin: eventemitter.example.com
Content-Type: application/cloudevents+json; charset=utf-8
Content-Length: nnnn

{
  "specversion": "1.0",
  "type": "nl.overheid.zaken.zaakstatus-gewijzigd",
  "source": "urn:nld:oin:00000001823288444000:systeem:BRP-component",
  "id": "f3dce042-cd6e-4977-844d-05be8dce7cea",
   ... further attributes omitted ...
}
```

Delivery response:

```
HTTP/1.1 204 No Content
```

3.4 Normative References

- HTTP 1.1 Web Hooks for Event Delivery - Version 1.0.1

4. Conformiteit

Naast onderdelen die als niet normatief gemarkeerd zijn, zijn ook alle diagrammen, voorbeelden, en noten in dit document niet normatief. Verder is alles in dit document normatief.

A. Index

A.1 Begrippen gedefinieerd door deze specificatie

A.2 Begrippen gedefinieerd door verwijzing

↑