

Logboek Dataverwerkingen Extensie Objecten



Logius Praktijkrichtlijn
Werkversie 17 december 2025

Deze versie:

<https://logius-standaarden.github.io/logboek-extensie-object/>

Laatst gepubliceerde versie:

<https://logius-standaarden.github.io/logboek-extensie-object/>

Laatste werkversie:

<https://logius-standaarden.github.io/logboek-extensie-object/>

Redacteurs:

Nil Barua ([Logius](#))

Tim van der Lippe ([Logius](#))

Auteurs:

Frank Terpstra ([Geonovum](#))

Niels Hoffmann ([Geonovum](#))

Doe mee:

[GitHub Logius-standaarden/logboek-extensie-object](#)

[Dien een melding in](#)

[Revisiehistorie](#)

[Pull requests](#)

Dit document is ook beschikbaar in dit niet-normatieve formaat: [PDF](#)



Dit document valt onder de volgende licentie:

[Creative Commons Attribution 4.0 International Public License](#)

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 TO goedgekeurde consultatieversie.

Inhoudsopgave

Status van dit document

Conformiteit

Samenvatting

- 1. Feedback en Issues**
- 2. Logboek Dataverwerkingen Extensie Objecten**
 - 2.1 Doel en nut
 - 2.2 Extensie (geo)objecten
 - 2.2.1 Resource
 - 2.2.2 Trace
 - 2.3 Gebruiksscenario's
 - 2.3.1 Verwerking van alleen objectgegevens
 - 2.3.2 Verwerking van objectgegevens met een relatie naar een persoonsgegevens
 - 2.3.3 Verwerking van objectgegevens in combinatie met persoonsgegevens
- A. Referenties**
 - A.1 Normatieve referenties

§ 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.

Samenvatting

404: Not Found

§ 1. Feedback en Issues

Dit onderdeel is niet normatief.

We moedigen gebruikers aan om meldingen of suggesties aan te maken via [GitHub](#). Mocht dit niet mogelijk zijn, dan kunt u ook een e-mail sturen naar api@logius.nl.

§ 2. Logboek Dataverwerkingen Extensie Objecten

De overheid wil voor burgers en bedrijven zo transparant mogelijk zijn in de omgang met hun gegevens. Daarom is het bij de informatieverwerking in datasets belangrijk om voor elke mutatie of raadpleging vast te leggen wie deze actie wanneer uitvoert, en waarom. Deze herleidbaarheid speelt zowel een rol in het kader van de wetgeving op het gebied van privacy als ook het streven naar openheid en transparantie bij de overheid. Voor een optimale samenwerking over organisaties en bronnen heen is voor deze logging een algemene standaard nodig.

Deze extensie beschrijft de technische specificaties hoe de Logboek Dataverwerkingen standaard kan worden toegepast voor het loggen van objectgegevens.

§ 2.1 Doel en nut

De [*logboek dataverwerkingen*] standaard specificeert hoe het verwerken van gegevens van personen en niet-natuurlijke personen gedaan moet worden. Er zijn diverse scenario's waarbij het wenselijk is om wel logging toe te passen, maar waar geen sprake is van persoonsgegevens. Of waar aanvullend aan persoonsgegevens ook gegevens over objecten gelogd dienen te worden. Deze extensie beschrijft hoe dit geïmplementeerd kan worden.

§ 2.2 Extensie (geo)objecten

Het loggen van (geo)objecten is een bijzonder ruim en daarmee flexibel onderwerp. Er zijn zeer veel mogelijke scenario's, afhankelijk van de gebruikte data, het type analyse of algoritme en het gewenste detailniveau.

We nemen een paar uitgangspunten op om de scope te verduidelijken.

1. Input/output poorten conform het idee van 'Dataproducten' als uitgangspunt voor de afbakening van het proces wat wordt gelogd.
2. API specificatie van een proces (als het goed is gelijk aan 1.)
3. Voor het loggen van gegevens binnen een trace maken we gebruik van het idee van 'Dataproducten' zoals die in de [DPROD](#) ontologie gepositioneerd worden.

In de Logboek dataverwerkingen standaard wordt de OTLP standaard aanbevolen om de [interface](#) naar het logboek mee te implementeren.

De OTLP standaard kent een aantal categorieën om telemetrie vast te leggen. Voor de Logboek dataverwerkingen standaard wordt de [traces](#) categorie gebruikt. Op het 'hoogste' niveau kent de standaard ook nog het concept [resource](#).

Bij het opzetten van een Logboek Dataverwerkingen interface kan er dus informatie op het niveau van resource vastgelegd worden. Dit is typisch informatie over het systeem waar de betreffende logging vandaan komt.

De traces zijn vervolgens de individuele verwerkingen die door het betreffende systeem gedaan worden.

§ 2.2.1 Resource

```
service.name = Logical name of the service
service.instance.id = The string ID of the service instance
```

Overeenkomstig de Opentelemetry specificatie.

Elk individueel Dataproduct/proces krijgt een eigen naam en id. Dus als er meerdere algoritmes/processen op een server zijn geïmplementeerd moet de `service.name` op het niveau van het individuele product/proces geïntanceerd worden.

§ 2.2.2 Trace

Definitie van een [Dataproduct](#): *A rational, managed, and governed collection of data, with purpose, value and ownership, meeting consumer needs over a planned life-cycle. A data product may have input and output ports, code and metadata.*

Open Data Mesh beschrijft een [dataprodukt](#) als: *It's the smallest unit that can be independently deployed and managed in a data architecture (i.e. architectural quantum). It is composed of all the structural components that it requires to do its function: the metadata, the data, the code, the policies that govern the data and its dependencies on infrastructure.*

Dit sluit goed aan op het abstractieniveau van wat we willen loggen met Logboek Dataverwerkingen. De beschreven structuur hieronder legt deze structurele componenten vast.

Afhankelijk van het gekozen niveau wordt er alleen gelogd op het niveau van het Dataproduct (niveau 1), Op het niveau van de datasets (of tabellen) en eventueel de features (rijen in de tabellen) (niveau 2), of zelfs de attributen en de waarden binnen de features (niveau 3).

```
dpl.objects.algorithm_id
dpl.objects.dataproduct_id
dpl.objects.dataset [
    dataset_id
    dataset_def
    dataset_port
    feature [
        feature_id
        feature_def
        feature_port

        feature_attribute [
            attribute_name
            attribute_value
            attribute_def
        ]
    ]
]
```

| attribute | Niveau | beschrijving |
|----------------------------|--------|--|
| dpl.objects.algorithm_id | 1 | verwijzing naar het register van het betreffende algoritme. uri naar uniek identificeerbaar algoritme |
| dpl.objects.dataproduct_id | 1 | uri naar een catalogus met de dataproduct metadata |
| dpl.objects.dataset | 2a | lijst met datasets (input en/of output van het dataproduct) |
| dataset_id | 2a | unieke id van de dataset |
| dataset_def | 2a | uri naar de definitie/metadata van de dataset (catalog_record/dcat_ap_nl) |
| dataset_port | 2a | input/output dataset, dit wordt in de log vastgelegd omdat het niet noodzakelijk uit de metadata in de catalogus afgeleid kan worden |
| feature | 2b | lijst met features: |
| feature_id | 2b | unieke id van het feature |
| feature_def | 2b | uri naar een definitie van het feature |
| feature_port | 2b | input/output feature, dit wordt in de log vastgelegd omdat het niet noodzakelijk uit de metadata in de catalogus afgeleid kan worden |
| feature_attribute | 3 | lijst van attributen van het feature |

| attribute | Niveau | beschrijving |
|-----------------|--------|---|
| attribute_name | 3 | unieke identifier van het attribuut |
| attribute_value | 3 | waarde van het attribuut in de specifieke verwerking / logregel |
| attribute_def | 3 | verwijzing naar de metadata van het attribuut |

Afhankelijk van het detailniveau wordt er gedetailleerder gelogd. Voor de hogere niveaus geldt dat de gegevens van het lagere niveau ook gelogd worden.

Voor niveau 2 (kolomniveau) geldt dat er op gehele dataset gelogd kan worden (2a), of dat er specifiek aangegeven kan worden welke features in een dataset gebruikt zijn (2b).

NOOT

Er is expliciet gekozen voor de naamgeving `dpl.objects.algorithm_id` om een duidelijk onderscheid te maken met `dpl.core.processing_activity_id`. `processing_activity_id` is altijd een verwijzing naar een verwerkingsregister van persoonsgegevens. `algorithm_id` is altijd de verwerking naar een register specifiek voor data verwerkingen. Dit kan het algoritme register zijn, maar zou ook een ander register kunnen zijn.

§ 2.3 Gebruiksscenario's

In dit kopstuk worden de verschillende use cases van de extensie beschreven.

§ 2.3.1 Verwerking van alleen objectgegevens

Bijvoorbeeld: Remote Sensing gebiedsclassificatie op basis van AI beeldherkenning.

In dit geval wordt alleen de `dpl.objects` namespace gebruikt op het gewenste detailniveau. `data_subject_id` en `processing_activity_id` uit de core namespace worden niet gebruikt.

§ 2.3.2 Verwerking van objectgegevens met een relatie naar een persoonsgegeven

Bijvoorbeeld: Maaidata analyse, remote sensing beelden analyseren of percelen wel/niet gemaaid zijn.

In dit geval wordt alleen de `data_subject_id` uit de core namespace gebruikt in combinatie met de `dpl.objects` namespace op het gewenste detailniveau.

§ 2.3.3 Verwerking van objectgegevens in combinatie met persoonsgegevens

Bijvoorbeeld: Aanvraag kapvergunning

In dit geval wordt zowel de core namespace gebruikt voor de verwijzing naar een verwerkingsregister en de `dpl.objects` namespace met een verwijzing naar een algoritme en andere relevante gegevens.

§ A. Referenties

§ A.1 Normatieve referenties

[logboek dataverwerkingen]

Logboek Dataverwerkingen. Logius. URL: <https://logius-standaarden.github.io/logboek-dataverwerkingen/>