



Project Notificatieservices

Architectuur

Versie 1.0

Datum: juni 2022

Auteurs: Project Notificatieservices



Inhoud

Inleiding en leeswijzer	4
1 Begrippen	5
1.1 Inleiding	5
1.2 Gebeurtenis, notificeren, notificatie en event	5
1.3 Gebeurtenisgedreven en event-driven	8
1.4 Aanbieder, afnemer, intermediair	8
1.5 Producer, Consumer, Intermediary	9
1.6 CloudEvents terminologie	10
1.7 Aspecten en termen	11
2 Architectuurstijlen	13
2.1 Inleiding	13
2.2 Service-oriëntatie	13
2.3 Message-driven	14
2.4 Event-driven	16
2.5 REST	17
2.6 Domain Driven Design	18
2.7 Conclusies	20
3 Soorten gebeurtenissen en notificaties	21
3.1 Inleiding	21
3.2 Business- en systeem-gebeurtenissen	21
3.3 Reële en administratieve gebeurtenissen	23
3.4 Informatierijke en informatiearme notificaties	24
3.5 Levensgebeurtenissen	26
4 Aspecten	27
4.1 Inleiding	27
4.2 Synchrone en asynchrone communicatie	27
4.3 Push en pull mechanismen	29
4.4 Ontkoppeling	31
4.5 Non-functional aspecten	33
4.5.1 Betrouwbaarheid	33
4.5.2 Beveiliging	34
4.5.3 Privacy	35
5 Patronen en standaarden	37
5.1 Inleiding	37
5.2 Patronen	37
5.3 Standaarden	38
6 Het Publish-Subscribe patroon	40
6.1 Algemeen	40
6.2 Filteren en routeren	41
6.3 Abonneren	45
6.4 Betrouwbaarheid en schaalbaarheid	46
6.5 Uitdagingen	47
6.6 Gebruik binnen de overheid	48
7 Implementatie-patronen	49
7.1 Inleiding	49
7.2 Change Data Capture	49
7.3 Basale messaging	51
7.4 Webhook	52
7.5 Event notification	54



7.6	Event Carried State Transfer	55
7.7	Event streaming	56
7.8	Event processing	57
7.9	Event sourcing.....	58
7.10	Command/Query Responsibility Segregation	60
8	Gegevens-formaten	62
8.1	Inleiding.....	62
8.2	JSON	62
8.3	XML	63
8.4	CSV	64
8.5	Overige formaten	64
9	Transport-protocollen.....	65
9.1	Inleiding.....	65
9.2	Protocollen	65
9.3	HTTP	67
9.3.1	Webhook	67
9.3.2	Digikoppeling RESTful API profiel	67
9.3.3	SOAP	68
9.3.4	WebSocket.....	68
9.3.5	Servers-Sent Events	69
9.4	AMQP	69
9.5	FTP.....	70
9.6	SMTP	71
9.7	Protocollen en betrouwbaarheid	71
10	Standaarden	73
10.1	Inleiding.....	73
10.2	CloudEvents berichtstandaard	73
10.3	WebSub interactiestandaard	73
10.4	AsyncAPI beschrijvingsstandaard.....	75
11	CloudEvents specificaties en het GOV NL profile for CloudEvents	76
11.1	Inleiding.....	76
11.2	CloudEvents doelen.....	77
11.3	CloudEvents architectuur	78
11.4	NL GOV profile for CloudEvents	79
Bijlage 1: Relatie met basisregistraties		82
Bijlage 2: Relatie met NORA, GDI-architectuur en Forum Standaardisatie		84
Bijlage 3: Toelichting Archimate		86
Bijlage 4: Middleware		88
Bijlage 5: Domain driven design - Context Maps		89
Bijlage 6: Handreikingen notificatieservices		90

Documentbeheer

Versie	Datum	Auteur	Toelichting
0.x	23-4-2021 1-6-2022	Ad Gerrits (VNG)	Conceptversies ter review door projectleden en community
1.0	30-6-2021	Projectteam Notificatieservices	Definitieve versie.



INLEIDING EN LEESWIJZER

Dit document beschrijft een aantal architectuuraspecten die van belang zijn bij het ontwerpen en uitvoeren van gebeurtenisgedreven werken in algemene zin en notificeren in het bijzonder. Het document maakt deel uit van de producten die zijn opgeleverd binnen Project Notificatieservices. Een project dat in opdracht van het Ministerie van Binnenlandse Zaken in 2021 en 2022 is uitgevoerd met als opdrachtnemer VNG Realisatie. Bij de uitvoering waren verschillende overheidsorganisaties, uitvoeringsorganisaties en marktpartijen betrokken. Informatie over zaken zoals aanleiding voor het project, doel en scope zijn elders beschreven.

Onderdeel van de opdracht was om 'een overheidsbreed bruikbare berichtstandaard te ontwikkelen voor notificeren van afnemers over plaatsgevonden relevante gebeurtenissen'. Naast de ontwikkelde [concept-berichtstandaard](#) zijn tijdens het project veel nieuwe inzichten en ervaringen opgedaan. Deze worden voor een deel beschreven in deze notitie. De tekst bevat, blauw gemarkeerd, ook concrete aanbevelingen die zijn bedoeld om organisaties te helpen bij het zetten van stappen op weg naar meer en beter gebeurtenisgedreven werken.

Voor het lezen van dit document wordt een zekere mate van architectuurn kennis en affiniteit met IT verondersteld. Er wordt hier gebruik gemaakt van Engelstalige, wat meer technisch-georiënteerde, begrippen waarmee wordt aangesloten bij wat wereldwijd gebruikelijk is. Voor de meeste architectuurbeschrijvingen gebruiken we de internationale [Archimate](#)-architectuurbeschrijvingstaal¹.

Dit document kent de volgende hoofdstukken:

- | | |
|---|--|
| 1 Begrippen | Beschrijving van begrippen die worden gebruikt. |
| 2 Architectuurstijlen | Architectuurstijlen die bruikbaar zijn bij het ontwerp van notificatieprocessen. |
| 3 Soorten gebeurtenissen en notificaties | Beschrijving van te onderscheiden soorten gebeurtenissen en hoe daarmee omgegaan kan worden. |
| 4 Diverse aspecten | Verschillend Aspecten waar bij notificeren aandacht aan moet worden geschonken. |
| 5 Patronen | Patronen bruikbaar voor event-driven werken. |
| 6 Het Publish-Subscribe patroon | Beschrijving van het belangrijkste patroon voor notificeren. |
| 7 Standaarden | Bruikbare standaarden voor notificeren via het Publish-Subscribe patroon. |
| 8 Implementatie-patronen | Bruikbare implementatiepatronen |
| 9 Gegevens-formaten | Bruikbare gegevensformaten. |
| 10 Transport-protocollen | Toepasbare protocollen met voor- en nadelen. |
| 11 CloudEvents standaarden en het GOV NL Profile for CloudEvents | Verschillende CloudEvents-specificaties en de daarop gebaseerde concept berichtenstandaard voor de Nederlandse overheid. |

De hoofdstukken zijn bedoeld om in volgorde te lezen maar het is ook mogelijk om bij interesse in een bepaald onderwerp slechts een of enkele hoofdstukken te lezen.

¹ Bijlage 3 'Archimate legenda' beschrijft de binnen diagrammen gebruikte elementen en relaties.



1 BEGRIPPEN

1.1 INLEIDING

Bij het uitwisselen van Informatie over gebeurtenissen en notificatieprocessen worden de grenzen van organisaties, domeinen en disciplines overschreden. Om overheidsbreed meer en beter event-driven te kunnen (samen)werken zijn afspraken en standaarden noodzakelijk. Een gezamenlijk begrippenkader maakt daar onderdeel van uit. Dit hoofdstuk beschrijft een aantal begrippen die bij notificeren een belangrijke rol spelen. Voor sommige van die begrippen geldt dat er nu verschillende termen voor worden gebruikt. Andersom geldt dat één begrip soms in verschillende betekenissen wordt gebruikt.

Binnen project Notificatieservices is een aparte notitie, '[Introductie van notificeren](#)', opgeleverd die beschrijft wat wordt bedoeld met belangrijke begrippen zoals 'gebeurtenis', 'notificeren' en 'notificatie'. Dit hoofdstuk bouwt hierop voort en licht de betekenis binnen architectuur en meer technische toepassingen toe. Doel hiervan is om binnen de Nederlandse overheid steeds meer van dezelfde taal gebruik te gaan maken bij communiceren over event-driven oplossingen.

Gebruik de overeengekomen terminologie bij het ontwerpen van event-driven oplossingen

Binnen architectuur maken we onderscheid tussen een bedrijfs-, applicatie- en technologielaag. Iedere laag kent eigen elementen en relaties daartussen. De bedrijfslaag kent bijv. 'diensten', 'bedrijfsprocessen' en 'actoren', de applicatielaag 'services', 'interfaces' en 'applicatiecomponenten'. Businessvertegenwoordigers en bedrijfsarchitecten zijn vooral geïnteresseerd in de bedrijfslaag. Informatiearchitecten en ontwikkelaars meer in de applicatielaag. We sluiten zoveel mogelijk aan bij wat qua terminologie gebruikelijk is bij de doelgroep. Begrippen binnen de bedrijfslaag worden daarom bijv. veelal aangeduid met een Nederlandse term terwijl we binnen de applicatielaag vaker met Engelse termen werken.

Gebruik terminologie die past bij de doelgroep en architectuurlaag

Voor het maken van architectuurbeschrijvingen en diagrammen gebruiken we de architectuurbeschrijvingstaal Archimate. Een wereldwijd gebruikte standaard voor beschrijving van architectuur. Archimate wordt (ook) gebruikt binnen overheidsbreed bruikbare referentiearchitecturen zoals de Nederlandse Overheid Referentie Architectuur (NORA) en Generieke Digitale Infrastructuur Architectuur (GA). Met behulp van Archimate zijn nauwkeuriger beschrijvingen te maken dan met alleen 'blokken en pijlen'. De bijlage 'Archimate elementen' beschrijft kort de belangrijkste elementen uit de Archimate-taal en hun betekenis binnen de context van event-driven werken.

1.2 GEBEURTENIS, NOTIFICEREN, NOTIFICATIE EN EVENT

Onder een '*gebeurtenis*' verstaan we: 'een representatie van een voorval'.

Binnen de context van event-driven werken gaat het om *voorvallen die hebben plaatsgevonden*. Een gebeurtenis is dus *een feit* dat tot bepaalde veranderingen heeft geleid die niet ongedaan kunnen worden gemaakt. Hooguit kan er, via een 'compenserende actie', een nieuwe gebeurtenis plaatsvinden waardoor een eerdere toestand hersteld kan worden. Hierbij passend geldt dat voor de omschrijving van een gebeurtenis gebruik wordt gemaakt van de voltooid verleden tijd (bijv. "vergunning (is) aangevraagd").

Formuleer gebeurtenissen in de voltooid verleden tijd



Gebeurtenissen zijn van een bepaald type: het *gebeurtenistype*. Goed onderscheiden en omschrijven van gebeurtenistypes is cruciaal om te zorgen dat verstrekte gebeurtenisgegevens door afnemers goed worden begrepen en gebruikt. Onderscheiden en omschrijven van gebeurtenistypen moet doordacht gebeuren en is niet eenvoudig. Reden waarom hiervoor een aparte handreiking '[Gebeurtenistypes definiëren](#)' is gemaakt. Bij het verstrekken van gegevens over plaatsgevonden gebeurtenissen wordt normaalgesproken aangegeven welk gebeurtenistype het betreft².

Onderscheid en omschrijf gebeurtenistypes zorgvuldig

Met '*notificeren*' bedoelen we: 'Het door een aanbieder aan afnemers verstrekken van gegevens over binnen zijn domein plaatsgevonden gebeurtenissen.' 'Verstrekken van informatie' kan op veel verschillende manieren plaatsvinden. Bijvoorbeeld:

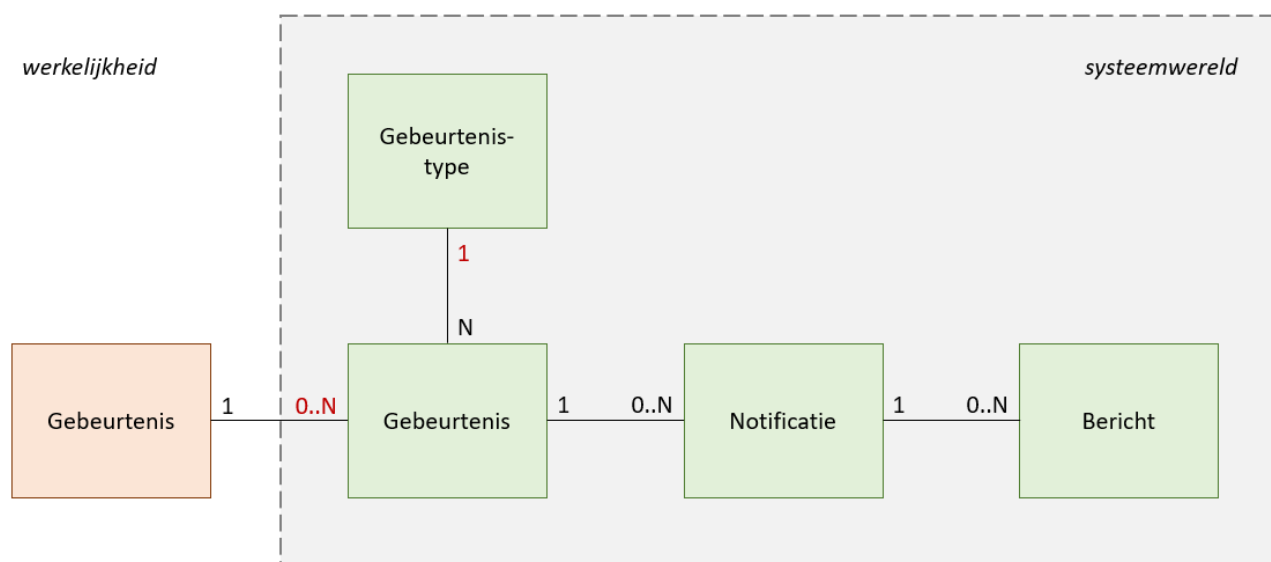
- Op initiatief van een aanbieder of van een afnemer ('push en pull').
- Met gebruik van verschillende gegevensformaten en protocollen ('JSON/XML', 'HTTP/AMQP').
- Terwijl aanbieder en afnemer wel of niet gelijktijdig actief zijn ('synchroon en asynchroon').
- Via verschillende soorten interactie tussen aanbieder en afnemer ('interactiepatronen').

De definitie spreekt over '*gegevens*' en niet '*informatie*' omdat wat een aanbieder verstrekt feiten over plaatsgevonden gebeurtenissen zijn. Afnemers ontleen daar, mogelijk verschillende, betekenis aan en maken er voor hen relevante '*informatie*' van die binnen hun omgeving is te verwerken.

'*Notificatie*' is als zelfstandig naamwoord afgeleid van 'notificeren' en kan worden vertaald als: 'bekendmaking'. Met 'notificaties' bedoelen we: '*de door een aanbieder aan afnemers verstrekte gegevens over binnen zijn domein plaatsgevonden gebeurtenissen*'.

De meest voorkomende, maar niet enige, manier om notificaties te verstrekken is om gebruik te maken van '*berichten*'. Een bericht is: 'Een gegevensrecord dat een berichtensysteem kan verzenden via een berichtenkanaal'³.

Onderstaande afbeelding toont hoe bovenstaande begrippen zich tot elkaar verhouden:



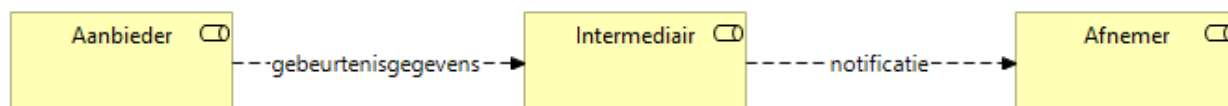
² Uitzondering hierop zijn situaties waarin het voor afnemers van gebeurtenisgegevens altijd duidelijk is om wat voor gebeurtenistype het gaat. Bijv. als het gegevens van een sensor betreft die de actuele temperatuur meet.

³ Bron: [Enterprise Integration Patterns, Gregor Hohpe and Bobby Woolf](#)



Het diagram maakt onderscheid tussen gebeurtenissen zoals ze zich in de werkelijkheid voordoen en representaties daarvan in de 'systeemwereld'. Een gebeurtenis binnen de systeemwereld kent een aantal beschrijvende kenmerken (waaronder het betreffende gebeurtenistype). Voor zo'n gebeurtenis kunnen notificaties ('bekendmakingen') aan afnemers worden verstrekt in de vorm van berichten.

Bij notificeren kunnen naast 'aanbieders' en 'afnemers' ook 'intermediairs' betrokken zijn⁴. Zij hebben de rol van 'bemiddelaar' tussen aanbieders en afnemers van notificaties. Zowel bij het verstrekken van gegevens van aanbieder naar intermediair als van intermediair naar afnemer zouden we kunnen spreken over 'notificeren'. Voor de duidelijkheid reserveren we 'notificeren' voor de activiteit waarbij gebeurtenisgegevens aan afnemers worden verstrekt. Het verstrekken van gegevens door aanbieders aan intermediairs duiden we eenvoudigweg aan als 'verstrekken van gebeurtenisgegevens'.



Reserveer de termen 'notificeren' en 'notificatie' voor de verstrekking van gegevens aan afnemers

Voor processen binnen de applicatielaag gebruiken we internationaal gebruikelijke Engelse termen en de daarbij behorende betekenis. Waar we binnen de bedrijfslaag spreken over 'gebeurtenis', 'aanbieder', 'afnemer' en 'intermediair', spreken we binnen de applicatielaag over 'event', 'Producer', 'Consumer' en 'Intermediary'. We sluiten hiermee, zowel qua aanduiding als betekenis, aan bij de definities binnen de CloudEvents-specificatie⁵:

An "event" is a data record expressing an occurrence and its context. Events are routed from an event Producer (the source) to interested event Consumers. The routing can be performed based on information contained in the event, but an event will not identify a specific routing destination. Events will contain two types of information: the Event Data representing the Occurrence and Context metadata providing contextual information about the Occurrence. A single occurrence MAY result in more than one event.

The "Producer" is a specific instance, process or device that creates the data structure describing the CloudEvent.

A "Consumer" receives the event and acts upon it. It uses the context and data to execute some logic, which might lead to the occurrence of new events.

An "Intermediary" receives a message containing an event for the purpose of forwarding it to the next receiver, which might be another Intermediary or a Consumer. A typical task for an Intermediary is to route the event to receivers based on the information in the Context.

Bron: [CloudEvents](#)

Het centrale begrip "event" wordt gedefinieerd als "a data record expressing an occurrence and its context". Dit is vergelijkbaar met wat we eerder hebben beschreven als 'een plaatsgevonden gebeurtenis die binnen de systeemwereld door een aantal kenmerken wordt beschreven'. Het is belangrijk om daarbij te realiseren

⁴ Feitelijk beschrijven 'aanbieder', 'afnemer' en 'intermediar' de rol die organisaties of applicaties spelen. Voor de leesbaarheid duiden we de organisaties of applicaties aan met de rolomschrijving.

⁵ De Cloud-Events specificatie speelt een belangrijke rol bij de ontwikkelde berichtstandaard en wordt later uitgebreid besproken.



dat een “event” wordt beschreven als een zelfstandig object (“a data record”) *dat voor verschillende doeleinden is te gebruiken*. Events gebruiken om gestandaardiseerd applicaties te notificeren, een van de hoofddoelen van project Notificatieservices, is er daar ‘slechts’ een van. Gebruik van standaarden als CloudEvents is dus niet alleen nuttig om geautomatiseerd notificeren te standaardiseren, maar is ook meer algemeen bruikbaar om binnen de Nederlandse overheid andere vormen van event-driven werken te standaardiseren.

Gebruik de term 'event' in de betekenis zoals die door CloudEvents is gedefinieerd

1.3 GEBEURTENISGEDREVEN EN EVENT-DRIVEN

Met de term ‘gebeurtenisgedreven’ bedoelen we binnen de bedrijfslaag dat plaatsgevonden gebeurtenissen als trigger werken voor aanbieders om informatie daarover te verstrekken voor geïnteresseerde afnemers. Daarbij is sprake van een *wezenlijk ander interactiepatroon* dan wanneer afnemers op het moment dat ze informatie nodig hebben die bij een aanbieder gaan opvragen⁶.

Naast het feit dat de interactie anders verloopt geldt ook dat informatie die wordt opgevraagd meestal geen informatie bevat over plaatsgevonden gebeurtenissen maar uitsluitend gegevens bevat die *het resultaat zijn* van een of meer plaatsgevonden gebeurtenissen.

Notificeren is een bijzondere vorm van verstrekking waarbij standaardisatie nodig is om als Nederlandse overheid effectief samen te kunnen werken en (snel) gegevens uit te wisselen. Er is een aparte handreiking, [Waarom notificeren](#), gemaakt om duidelijk(er) te maken waarom gestandaardiseerd notificeren nodig is.

Zoals eerder toegelicht sluiten we voor de applicatielaag aan bij de daar gebruikelijke terminologie. In plaats van ‘gebeurtenisgedreven’ spreken we binnen de applicatielaag over ‘event-driven’.

1.4 AANBIEDER, AFNEMER, INTERMEDIAR

Bij notificatieprocessen zijn minimaal twee rollen van belang: een 'aanbieder' en een 'afnemer' van gegevens. Afhankelijk van de context worden vaak andere termen voor deze rollen gebruikt. Onderstaande tabel geeft een aantal in gebruik zijnde combinaties weer.

Rol 1	Rol 2	Context
(Dienst)aanbieder	(Dienst)afnemer	Dienstgerichte benaderingen
Verstrekker	Ontvanger	Algemeen
Leverancier	Klant	Algemeen
Provider	Consumer	Informatietechnologie
Producer	Consumer	Informatietechnologie
Publisher	Subscriber	Gebruik van ‘Publish Subscribe’ patroon.
Houder Landelijke Voorziening	Afnemer	Digilevering
Subject	Observer	Softwareontwikkeling
Sender	Receiver	Informatietechnologie

⁶ Het opvragen van informatie via een ‘request/reply patroon’ is momenteel sterk in opmars binnen de overheid vanuit het uitgangspunt “opvragen bij de bron”. Dit gebeurt nu steeds vaker via services die werken conform de REST-architectuurstijl (zie paragraaf 2.5).

In lijn met de wens om bij doelgroepen passende terminologie te hanteren gebruiken we binnen de bedrijfslaag en voor communicatie naar brede doelgroepen de Nederlandse termen '(dienst)aanbieder' en '(dienst)afnemer'. Hiermee sluiten we aan bij de dienstgerichte benadering en terminologie van overheidsbrede architecturen als de Nederlandse Overheid Referentie Architectuur (NORA) en de GDI-Architectuur (GA) en bij het taalgebruik van business-vertegenwoordigers. Voor de beschrijving van architectuur binnen de applicatielaag gebruiken we de daar gebruikelijke termen 'Producer' en 'Consumer'. Voor de bemiddelende rol binnen notificatieprocessen spreken we binnen de bedrijfslaag over 'Intermediair' en binnen de applicatielaag over 'Intermediary'.

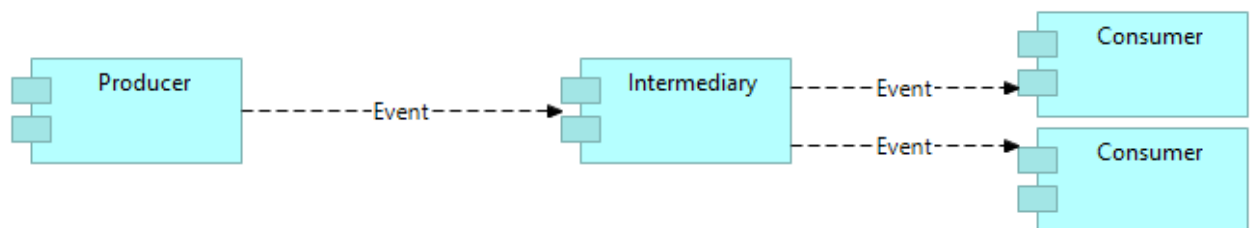
1.5 PRODUCER, CONSUMER, INTERMEDIARY

The "Producer" is a specific instance, process or device that creates the data structure describing the CloudEvent.

A "Consumer" receives the event and acts upon it. It uses the context and data to execute some logic, which might lead to the occurrence of new events.

An "Intermediary" receives a message containing an event for the purpose of forwarding it to the next receiver, which might be another Intermediary or a Consumer. A typical task for an Intermediary is to route the event to receivers based on the information in the Context.

bron: [CloudEvents](#)



Figuur 1 Basispatroon waarbij een Producer gebruik maakt van een Intermediary voor het verstrekken van notificaties aan Consumers

Toelichting:

- De rol van 'Intermediary' kan door een aparte, intern of extern aanwezige, applicatie worden ingevuld. Afhankelijk van de context kan de rol wel of niet apart worden benoemd.
- De pijlrichting binnen diagrammen geeft alleen aan dat er events vanuit een Producer naar Consumers stromen. Het zegt niets over zaken als het daarbij gebruikte mechanisme (bijv. 'push' of 'pull'), gegevensformaat (bijv. JSON of XML) of transportprotocol (bijv. HTTP of AMQP).

Om misverstanden te voorkomen onderscheiden we bij communicatie naar brede doelgroepen 'gebeurtenisgegevens' (die een Producer aan een Intermediary verstrekt) en 'notificaties' (die een



Producer of Intermediary aan Consumers verstrekt). Binnen de applicatielaag spreken we echter in beide gevallen over (verstrekken van) 'events' ('a data record').

Redenen hiervoor zijn dat:

- Binnen ketens van Producer tot Afnemer van meerdere bemiddelende applicaties sprake kan zijn waarbij zij meerdere rollen vervullen. Bijv. een Intermediary die events verstrekt aan een volgende Intermediary.
- Consumers zelf ook de rol van Producer of Intermediary kunnen gaan spelen. Bijv. als een Consumer een ontvangen event verstrekt aan andere Consumers of Intermediaries.
- Standaardisatie wordt bevorderd door binnen notificatieprocessen op verschillende momenten gebruik te maken van dezelfde standaarden. Bijv. door Producers events conform dezelfde standaard te laten verstrekken aan Intermediaries als dat Intermediaries dit doen richting Consumers⁷.

1.6 CLOUDEVENTS TERMINOLOGIE

Een van de hoofddoelen van project Notificatieservices was om een overheidsbreed bruikbaar berichtformaat voor notificeren te ontwikkelen. Bij het maken daarvan is voortgebouwd op de [CloudEvents specificatie](#) die is ontwikkeld door de [Serverless Working Group](#) van de [Cloud Native Computing Foundation](#).

CloudEvents is a specification for describing event data in common formats to provide interoperability across services, platforms and systems. – bron: [CloudEvents](#)

We maken zo veel mogelijk gebruik van de [CloudEvents terminologie](#). Zowel om de voor de Nederlandse overheid bedoelde berichtenstandaard te beschrijven als om event-driven werken in het algemeen te beschrijven.

Occurrence	An "occurrence" is the capture of a statement of fact during the operation of a software system. This might occur because of a signal raised by the system or a signal being observed by the system, because of a state change, because of a timer elapsing, or any other noteworthy activity. For example, a device might go into an alert state because the battery is low, or a virtual machine is about to perform a scheduled reboot.
Event	An "event" is a data record expressing an occurrence and its context. Events are routed from an event Producer (the source) to interested event Consumers. The routing can be performed based on information contained in the event, but an event will not identify a specific routing destination. Events will contain two types of information: the Event Data representing the Occurrence and Context metadata providing contextual information about the Occurrence. A single occurrence MAY result in more than one event.
Producer	The "Producer" is a specific instance, process or device that creates the data structure describing the CloudEvent.
Source	The "source" is the context in which the occurrence happened. In a distributed system it might consist of multiple Producers . If a source is not aware of CloudEvents, an external Producer creates the CloudEvent on behalf of the source.
Consumer	A "Consumer" receives the event and acts upon it. It uses the context and data to execute some logic, which might lead to the occurrence of new events.
Intermediary	An "Intermediary" receives a message containing an event for the purpose of forwarding it to the next receiver, which might be another Intermediary or a Consumer . A typical task for an Intermediary is to route the event to receivers based on the information in the Context .
Context	Context metadata will be encapsulated in the Context Attributes . Tools and application code can use this information to identify the relationship of Events to aspects of the system or to other Events.
Data	Domain-specific information about the occurrence (i.e. the payload). This might include information about the occurrence, details about the data that was changed, or more. See the Event Data section for more information.

⁷ Een event dat een Producer verstrekt aan een Intermediary hoeft niet noodzakelijkerwijs dezelfde inhoud te hebben als het event dat vervolgens door de Intermediary aan Consumers wordt verstrekt. Ook wanneer dit niet het geval is kan van dezelfde standaarden gebruik worden gemaakt.



Event Format	An Event Format specifies how to serialize a CloudEvent as a sequence of bytes. Stand-alone event formats, such as the JSON format , specify serialization independent of any protocol or storage medium. Protocol Bindings MAY define formats that are dependent on the protocol.
Message	Events are transported from a source to a destination via messages. A "binary-mode message" is one where the event data is stored in the message body, and event attributes are stored as part of message meta-data. A "structured-mode message" is one where the event is fully encoded using a stand-alone event format and stored in the message body.
Protocol	Messages can be delivered through various industry standard protocol (e.g. HTTP, AMQP, MQTT, SMTP), open-source protocols (e.g. Kafka, NATS), or platform/vendor specific protocols (AWS Kinesis, Azure Event Grid).
Protocol Binding	A protocol binding describes how events are sent and received over a given protocol. Protocol bindings MAY choose to use an Event Format to map an event directly to the transport envelope body, or MAY provide additional formatting and structure to the envelope. For example, a wrapper around a structured-mode message might be used, or several messages could be batched together into a transport envelope body.

Bron: [CloudEvents terminology](#)

1.7 ASPECTEN EN TERMEN

Bij ontwerp en inrichting van notificatieprocessen moeten op een aantal *aspecten* keuzes worden gemaakt. Deze paragraaf beschrijft kort enkele van die aspecten⁸:

Push	Bij gebruik van push-mechanismen worden notificaties bezorgd bij Consumers (die daarvoor in staat moeten zijn om ze conform afspraken te ontvangen).
Pull	<p>Bij gebruik van pull-mechanismen stelt een Producer notificaties beschikbaar die op een zelf te bepalen moment door Consumers opgehaald.</p> <p>Let op: Binnen de Nederlandse overheid worden de termen meestal gebruikt in bovenstaande betekenis. CloudEvents hanteert een afwijkende definitie van 'pull' als mechanisme waarbij een Consumer het initiatief neemt tot gegevensuitwisseling, daarvoor een verbinding opzet waarmee notificaties door een Consumer worden opgehaald <i>of door een Producer verstrekt</i>. In onze termen zouden we daarbij spreken over een combinatie van push- en pull mechanismen.</p>
Synchroon	<p>Bij synchrone (<i>technische</i>) <i>communicatie</i> tussen applicaties blijft de initiërende applicatie wachten op een antwoord van de ontvangende partij. Een voorbeeld hiervan is het Request-Reply patroon dat wordt toegepast bij services die werken volgens de REST-stijl.</p> <p>Bij synchrone (<i>functionele</i>) <i>interactie</i> van applicaties zijn applicaties gelijktijdig actief tijdens het uitwisselen van gegevens. Bijv. als een Producer eenmalig probeert om een verbinding te maken met een Consumer om een notificatie te verstrekken.</p>
Asynchroon	<p>Bij asynchrone (<i>technische</i>) <i>communicatie</i> tussen applicaties wacht de initiërende applicatie niet op de ontvangende applicatie (en kan dus doorgaan met andere activiteiten)⁹. Als er al een antwoord komt wordt dit op een later moment, via een nieuwe verbinding, naar de initiërende partij verstuurd.</p> <p>Bij asynchrone (<i>functionele</i>) <i>interactie</i> kunnen applicaties gegevens uitwisselen zonder dat ze gelijktijdig actief hoeven te zijn. Gegevens worden daarvoor als</p>

⁸ Keuzes per aspect en daarbij geldende voorkeuren worden in latere hoofdstukken beschreven.

⁹ Er kan wel sprake zijn van een 'technische ontvangstbevestiging' waarbij de ontvanger via een 'acknowledged' (ACK) uitsluitend aangeeft dát een bericht is aangekomen.



autonoom object behandeld en kunnen bijv. worden bewaard en opnieuw verstrekt als een applicatie tijdelijk niet actief is.

Opmerking: Om asynchrone (functionele) interactie te realiseren kan zowel gebruik worden gemaakt van asynchrone als synchrone (technische) communicatie tussen applicaties.

Informatiearm	Met 'informatiearm' wordt bedoeld dat er zeer beperkt gegevens over plaatsgevonden gebeurtenissen worden verstrekt ('dat gegevens') op basis waarvan Consumers aanvullende informatie kunnen opvragen (iets dat vaak nodig is om tot verwerking over te gaan). Een voorbeeld hiervan is een notificatie die uitsluitend gegevens bevat waaruit is te concluderen dát er een bepaald type gebeurtenis heeft plaatsgevonden (al dan niet met betrekking tot een bepaald object).
Informatierijk	Met 'informatierijk' wordt bedoeld dat er een rijke set aan gegevens wordt verstrekt over plaatsgevonden gebeurtenissen ('dat en wat gegevens') op basis waarvan Consumers direct over kunnen gaan tot verwerking. Een voorbeeld hiervan is een notificatie die alle voor Consumers relevante gegevens over een plaatsgevonden gebeurtenis bevat.
Ontkoppeling	De mate van ontkoppeling van applicaties geeft aan in welke mate ze los van elkaar kunnen functioneren. Ontkoppeling kan op een aantal aspecten betrekking hebben (bijv. technologie, tijd, plaats). Het is wenselijk dat binnen notificatieprocessen applicaties zo veel mogelijk van elkaar ontkoppeld zijn.

Naast genoemde aspecten spelen bij het ontwerpen en realiseren van notificatieoplossingen uiteraard ook 'non-functional requirements' een rol zoals: betrouwbaarheid, beveiligbaarheid, gebruiksgemak en performance. Binnen project Notificatieservices zijn deze aspecten, mede omdat ze vaak sterk contextafhankelijk zijn, slechts beperkt belicht.



2 ARCHITECTUURSTIJLEN

2.1 INLEIDING

Voor het ontwerpen en uitvoeren van notificatieprocessen kan gebruik worden gemaakt van verschillende architectuurstijlen. Iedere stijl kent een aantal fundamentele uitgangspunten met betrekking tot hoe processen en systemen worden ontworpen. Dit hoofdstuk beschrijft een aantal bewezen architectuurstijlen en hoe ze bij het ontwerpen van notificatieprocessen zijn toe te passen. Doel is om bij het ontwerpen van notificatieprocessen kennis te hebben van verschillende stijlen om die binnen de eigen context effectief toe te kunnen passen.

Architectuurstijlen zijn bruikbaar binnen verschillende architectuurlagen, zoals bedrijfs-, applicatie- en technologielaag. Bij het ontwerpen van oplossingen spelen meerdere architectuurlagen een rol. Voor notificatieoplossingen zijn bijv. vaak afspraken nodig op bedrijfsniveau en worden overeengekomen standaarden toegepast op applicatie- en technologie-niveau.

2.2 SERVICE-ORIËNTATIE

Service-oriëntatie is een binnen de overheid bekende architectuurstijl. Daarbij is sprake van aanbieders en afnemers van diensten. Voor de levering van diensten worden afspraken gemaakt tussen aanbieder en afnemers. Binnen de bedrijfslaag wordt, in de taal van businessvertegenwoordigers, gesproken over 'diensten'. Ook de NORA stelt het begrip 'dienst' centraal: "Een afgebakende prestatie van een persoon of organisatie (de dienstverlener), die voorziet in een behoefte van haar omgeving (de dienstafnemer(s))".

Binnen de applicatie- en technologielaag spreken we, in lijn met internationale terminologie, over 'services'. Kenmerkend voor services is:

- *Een service is virtueel: de afnemer heeft geen weet van de implementatie van de dienst. De dienst is onafhankelijk van de afnemer. Scheiding van verantwoordelijkheid is expliciet vastgesteld. Voordeel hiervan is onafhankelijkheid van veranderingen van afnemer en leverancier. Voldoen aan het contract staat immers centraal;*
- *Een service is gestandaardiseerd: er is slechts één implementatie aanwezig van een verantwoordelijkheid. Voordeel is het rationaliseren van de standaard. Standaardiseren=massa, massa=kassa. De dienst wordt 'commodity' en eenvoudig te vervangen door een andere leverancier of goedkoper door een efficiëntieslag;*
- *Een service is modulair (vervangbaar) en compositioneerbaar. Standaard is niet flexibel. Flexibiliteit wordt bereikt door combineren (componeren) van standaards tot een nieuwe standaard;*
- *Een service is abstract: generiek, niet afgestemd voor 1 specifieke afnemer, maar op een doelgroep van afnemers;*
- *Losgekoppeld: afnemer en leverancier zijn maximaal onafhankelijk van implementatie van beide. Elke service is daarom autonoom. Er bestaat geen directe link of relatie tussen verschillende services. Services zijn zich ook niet van elkaar bewust. – Bron: [Wikipedia](https://en.wikipedia.org/wiki/Service_(business))*

De NORA beschouwt service-oriëntatie als de leidende architectuurstijl voor de Nederlandse overheid. Naast de brede toepasbaarheid ervan is een belangrijk argument dat de stijl zeer geschikt is om binnen de overheid 'interoperabiliteit' te realiseren:

Interoperabiliteit is het vermogen van organisaties (en hun processen en systemen) om effectief en efficiënt informatie te delen met hun omgeving – bron: [NORA](https://nora.nl/)



Notificatieprocessen spelen zich af binnen een divers landschap van organisaties en applicaties waar interoperabiliteit nodig is op verschillende aspecten. Het NORA-Vijflaagsmodel onderscheidt hiervoor een vijftal lagen. (Ook) bij notificeren moet aandacht zijn voor alle vijf de lagen om te zorgen dat er binnen de overheid voldoende interoperabiliteit kan ontstaan.

Applicaties in de rol van Producer, Intermediary of Consumer kennen verschillende taken en verantwoordelijkheden. Om goed samen te kunnen werken moeten die helder benoemd en van elkaar gescheiden worden ('separation of concerns'). Bij voorkeur worden ze maximaal van elkaar ontkoppeld ('high cohesion, low coupling') zodat ze onafhankelijk van elkaar kunnen functioneren.

Services stellen functionaliteit beschikbaar voor gebruik via toegangspunten: 'Application Programming Interfaces' (APIs). Er zijn verschillende stijlen om APIs te ontwikkelen, ieder met eigen voor- en nadelen. Binnen de overheid is wordt met name de REST-stijl steeds vaker toegepast. Het binnen project Notificatieservices ontwikkelde [NL GOV profile for CloudEvents](#) kan (ook) bij gebruik van deze stijl worden gebruikt.



Gebruik service-oriëntatie als architectuurstijl bij het ontwerpen van notificatieoplossingen

2.3 MESSAGE-DRIVEN

Vrijwel alle applicaties moeten tegenwoordig samenwerken en gegevens uitwisselen met andere applicaties. Voor de benodigde 'integratie van applicaties' zijn verschillende stijlen bruikbaar.

Bekende integratiestijlen zijn:

- Gebruik van bestanden of databases:
De aanbieder biedt hierbij een bestand of database aan dat beschikbaar wordt gesteld voor gebruik door afnemers. Op basis van afspraken over inhoud en formaat kunnen afnemers gegevens lezen en gebruiken. Beschikbaarstelling kan via delen of (deel) kopieën. Bij uitwisseling van kopieën kan gebruik worden gemaakt van bestandsformaten zoals CSV, JSON en XML, push- of pull-mechanismes, protocollen zoals FTP, HTTP en RPC.
- Remote Procedure Call:
Afnemers roepen hierbij een door de aanbieder beschikbaar gestelde service aan die bijvoorbeeld bepaalde gegevens naar de afnemer retourneert. Het meest gebruikte transportprotocol daarbij is HTTP. Het meest gebruikte berichtprotocol SOAP. Gangbare berichtformaten zijn XML en JSON. gRPC is een framework voor RPC-communicatie waarbij hoge performance nodig is.

Beide stijlen zijn tot op zekere hoogte bruikbaar voor notificeren maar kennen ook een aantal, soms doorslaggevende, nadelen bij integratievraagstukken. Bijv. wanneer:

- Processen moeten worden getriggerd.
- Gegevens realtime beschikbaar moeten komen.
- Sprake is van veel diverse organisaties en applicaties.
- Het belangrijk is om applicaties van elkaar ontkoppeld te houden.

De meest geschikte en in praktijk vaakst toegepaste integratiestijl is 'messaging'. Aanbieders verzenden daarbij gegevens naar afnemers in de vorm van berichten ('messages'):



A message is a data record that the messaging system can transmit through a message channel. – bron: Enterprise Integration Patterns, G. Hohpe and B. Woolf

Een groot voordeel van een bericht ten opzichte van een set gegevens die via een tijdelijke verbinding worden uitgewisseld is dat een bericht geschikt is om als zelfstandig object te functioneren. Het kan bijv. een of meerdere keren worden verzonden, kan tijdelijk of permanent worden opgeslagen en kan worden voorzien van verschillende soorten metagegevens. Bijv. gegevens die nodig zijn voor het kunnen routeren naar de juiste Consumers of meta-informatie over inhoudelijke informatie in het bericht.

Gebruik bij het ontwerp van notificatieprocessen messaging als integratiestijl

Omdat bij notificatieprocessen vaak veel verschillende applicaties gegevens met elkaar moeten uitwisselen is het wenselijk dat interactie asynchroon kan verlopen. Vanwege het feit dat berichten als zelfstandige pakketjes zijn te transporteren en bewaren is de messaging-stijl hiervoor zeer geschikt.

De communicatie tussen betrokken applicaties kan daarbij zowel synchroon ('A wacht op antwoord van B') als asynchroon ('A wacht niet op antwoord van B') plaatsvinden. Iets dat ook de keuze beïnvloedt van welk transportprotocol gebruik gemaakt wordt. Asynchrone communicatie zorgt weliswaar voor (nog) meer ontkoppeling maar het is uiteindelijk contextafhankelijk welk type communicatie, en daarbij behorende transportprotocollen, de voorkeur verdient.

Overweeg toepassing van zowel synchrone als asynchrone communicatie tussen applicaties

Bij berichtuitwisseling wordt vaak gebruik gemaakt van gespecialiseerde (middleware)applicaties om de rol van Intermediary te vervullen. Dit type applicaties kan bijv. zorgen voor tussentijds opslaan van berichten om te voorkomen dat ze kwijtraken of berichten bufferen om als 'schokdemper' te fungeren als het tempo waarin berichten beschikbaar komen voor een afnemer te hoog is ('throttling'). Met gebruik van gespecialiseerde voorzieningen is een flexibeler en betrouwbaarder berichtuitwisseling te realiseren. Een belangrijk bijkomend voordeel is dat hierdoor bedrijfsapplicaties worden ontlast van dit soort taken waardoor ze zich kunnen focussen op hun kernfuncties ('separation of concerns').

Gebruik gespecialiseerde voorzieningen voor berichtenverkeer

Voor het samenstellen van berichten kan gebruik worden gemaakt van verschillende gegevensformaten. Ieder formaat heeft zijn eigen karakteristieken en toepassingsgebied. Voorbeelden van veelgebruikte gegevensformaten zijn JSON, XML en CSV. Met name JSON wordt steeds vaker toegepast voor gegevensuitwisseling, binnen en buiten de overheid. Binnen de Nederlandse API-strategie bestaat het voornemen om JSON als voorkeurformaat bij gebruik van APIs te benoemen ('[APIs receive and send JSON](#)').

Gebruik JSON als gegevensformaat

Voor transport van berichten kan gebruik worden gemaakt van verschillende transportprotocollen. Ieder protocol heeft zijn eigen karakteristieken en toepassingsgebied. Het ene protocol is bijv. geschikt om betrouwbare uitwisseling te ondersteunen maar is lastig in gebruik, terwijl een ander protocol minder betrouwbaarheid kan bieden maar gemakkelijker is in gebruik.

Bij protocollen is onderscheid te maken in breed toepasbare protocollen zoals HTTP en specifiek voor messaging ontwikkelde protocollen zoals [AMQP](#), [STOMP](#), [XMPP](#) en [MQTT](#). Een voordeel van een breed toepasbaar protocol als HTTP is dat veel partijen het kennen en er al ervaring mee hebben. Een nadeel is dat het, in vergelijking met voor messaging ontworpen protocollen, niet over bepaalde benodigde functionaliteit beschikt (en er dus aanvullend maatwerk nodig is om die te realiseren). Het is daarom raadzaam om bij het



ontwerpen van oplossingen niet zondermeer gebruik te maken van wat het meest bekend is, maar te bezien welk protocol binnen de gegeven context het meest geschikt is.

Weeg af welk protocol binnen de gegeven context het meest geschikt is

2.4 EVENT-DRIVEN

In een steeds dynamischer en digitalere wereld, waarin alles met elkaar verbonden is, wordt van organisaties en applicaties verwacht dat er met actuele informatie wordt gewerkt¹⁰. Na het plaatsvinden van relevante gebeurtenissen moeten gegevens daarover snel en betrouwbaar beschikbaar komen en verwerkt. Event-driven architectuur is daarvoor bij uitstek geschikt.

Event-driven architecture (EDA) is a software architecture paradigm promoting the production, detection, consumption of, and reaction to events. – bron: Wikipedia

De event-driven architectuurstijl kan worden gezien als een aanvulling op de servicegerichte- en messaging-stijl. Waar klassieke service-oriëntatie zich veelal richtte op statische vraag-gedreven bedrijfsprocessen is bij event-driven architectuur sprake van een dynamisch netwerk van services die elkaar continu op de hoogte houden van relevante plaatsgevonden gebeurtenissen. 'Events' zijn de trigger om gegevens te publiceren en alle partijen die daar belang bij kunnen hebben snel van actuele informatie te voorzien.

De in de voorgaande paragraaf beschreven messaging-stijl wordt vaak toegepast bij gegevensuitwisseling tussen actoren die elkaar kennen en uniek adresseerbaar zijn ('point-to-point'). De event-driven stijl wijkt hiervan af: de hoofdtak van een Producer is het publiceren van events. Die events kunnen vervolgens op verschillende manieren, bijv. met behulp van een Intermediary, worden verstrekt aan nul, 1 of meerdere Consumers. Consumers die een Producer, anders dan bij point-to-point communicatie, voor technische uitwisseling niet noodzakelijkerwijs hoeft te kennen¹¹. Producers en Consumers zijn hierbij in technische zin dus volledig ontkoppeld.

Gebruik de event-driven architectuurstijl om (snel) gegevens over plaatsgevonden gebeurtenissen te kunnen verstrekken aan Consumers

Omdat gebeurtenissen zich in allerlei vormen voordoen is de event-driven architectuurstijl bruikbaar binnen hele verschillende contexten, zoals:

- Interactieve software die moet reageren op uitgevoerde handelingen door een gebruiker (bijv. met behulp van toetsenbord of muis).
- Apparaten binnen het Internet of Things die elkaar van informatie voorzien (bijv. een sensor die overschrijding van een signaalwaarde constateert)
- Microservices die elkaar notificeren (bijv. een bestelcomponent die bekend maakt dat er een nieuwe bestelling is geplaatst).

¹⁰ Hoewel 'actueel' een relatief begrip is geldt in zijn algemeenheid dat er steeds hogere eisen aan actualiteit worden gesteld. Consumenten verwachten bijv. snel een betrouwbaar antwoord op vragen waardoor bedrijfsprocessen over actuele ('near realtime') gegevens moeten kunnen beschikken.

¹¹ Uiteraard is het bij uitwisseling van bijv. vertrouwelijke gegevens wel nodig dat een Producer voorafgaand aan verstrekking weet welke Consumers het betreft.



Project Notificatieservices richtte zich met name op het verbeteren en standaardiseren van geautomatiseerd notificeren van (bedrijfs)applicaties¹². Vaak toegespitst op situaties waarin opgetreden gebeurtenissen leidden tot wijziging van bepaalde brongegevens die relevant (kunnen) zijn voor Consumers.

Voor effectiever event-driven werken binnen de overheid is het belangrijk om te beseffen dat de event-driven architectuurstijl veel breder toepasbaar dan 'slechts' het doorgeven van brongegevens-wijzigingen. Andere event-driven toepassingen die meerwaarde kunnen bieden zijn bijv. het bewaren van events in volgorde van ontvangst ('log based') waardoor nieuwe soorten analyses, selectieve tijdgebonden opvragingen of 'replay van events' mogelijk worden. Een andere mogelijkheid is om events te gaan bewaren en als primaire bron te zien en traditioneel gegevensgericht gebruik van stateful objecten via 'views' te faciliteren. Deze manier van kijken naar events is een paradigmashift ten opzichte van de gegevensgerichte manier waarop de overheid nu meestal werkt. De handreiking '[Introductie van notificeren](#)' beschrijft meer in detail wat een event-driven werkwijze inhoudt en aan meerwaarde voor de overheid kan hebben. In algemene zin geldt dat de event-driven stijl geschikt is in situaties waarin gebeurtenissen een belangrijke rol spelen.

Gebruik de event-driven architectuurstijl als gebeurtenissen een belangrijke rol spelen

De event-driven architectuurstijl is bij uitstek geschikt autonoom functionerende applicaties (of 'componenten') via events met elkaar te laten communiceren. Binnen cloud-omgevingen is deze manier van samenwerken al langer gebruikelijk (bijv. bij gebruik van microservices of serverless functies). Binnen traditionele omgevingen worden in plaats daarvan nog vaak (1 op 1) maatwerkkoppelingen gemaakt. Communiceren via events zorgt ervoor dat Producers, Intermediaries en Consumers deel gaan uitmaken van een dynamisch netwerk waarin events het middel zijn om applicaties te laten samenwerken.

Gebruik events om autonome applicaties met elkaar te laten communiceren

Binnen de overheid worden oplossingen veelal data- en proces gedreven ontworpen en wordt er in lijn daarmee gewerkt. De event-driven architectuurstijl wordt nog weinig toegepast. Effectief gebruik van deze stijl vereist een fundamenteel andere manier van denken en ontwerpen van oplossingen. Aanpassingen in cultuur en technologie duren, afgaande op ervaringen in andere sectoren, minimaal 5 tot 10 jaar. Rekening houdend met de grote diversiteit en verschillende niveaus van volwassenheid binnen de overheid onderscheiden we verschillende niveaus waarop event-driven architectuur is toe te passen. De notitie '[Waarom notificeren?](#)' beschrijft deze niveaus meer in detail. Hoofdstuk 7 'Implementatiepatronen' beschrijft patronen die op verschillende niveaus bruikbaar zijn.

Bepaal op welk niveau event-driven kan worden gewerkt en kies passende implementatiepatronen

2.5 REST

De Representational State Transfer (REST) architectuurstijl is ontwikkeld voor het ontwerpen van software binnen gedistribueerde hypermedia ('gelinkte') systemen. Voor de communicatie tussen applicaties gelden daarbij [zes eisen](#). In de praktijk voldoen systemen vaak maar aan een aantal van deze zes eisen en worden daarom vaak aangeduid als 'RESTful'¹³. Het fundamentele concept binnen REST is de 'resource'. Alle informatie die benoemd kan worden is te beschouwen als een resource. Een 'server' ondersteunt acties voor

¹² We gebruiken het begrip 'applicatie' omdat dit gebruikelijk is bij traditionele gegevensuitwisseling binnen de overheid. Binnen andere contexten wordt bijv. gesproken over '(applicatie)componenten' of 'microservices'.

¹³ Het [Richardson Maturity Model](#) onderscheidt vier niveau's om de mate waarin API's REST-conform zijn te beschrijven.



‘clients’ om resources te benaderen. Daarbij wordt vrijwel altijd gebruik gemaakt van het HTTP-protocol en de daarin onderscheiden methoden (bijv. ‘GET’, ‘POST’, ‘UPDATE’, ‘DELETE’).

In de praktijk wordt binnen notificatieprocessen vaak gebruik gemaakt van zogenaamde 'webhooks'. Er wordt dan een notificatie verstrekt door een (service-)API aan te roepen op een door een Consumer opgegeven endpoint. Daarbij wordt dan gebruik gemaakt van de POST methode van het HTTP-protocol. Een Consumer kan zijn API ontwikkelen volgens de REST-stijl. Bijv. door een endpoint te definiëren als 'https://example.com/notificaties' en POST-requests te zien als trigger voor het ontstaan van 'nieuwe instanties van de resource notificatie'.

Er kan om valide redenen voor worden gekozen om gebruik te maken van de REST-stijl (bijv. omdat die bekend is en er brede ervaring mee is). Het is daarbij belangrijk te realiseren dat het niet noodzakelijk is om notificatieverstrekking aan afnemers te realiseren. Bijv. door een ander protocol dan HTTP te gebruiken of door een HTTP-POST te doen zonder dat die REST-conform is.

Maak bij notificeren gebruik van de REST-stijl als dit binnen de context voordelen biedt

Opmerking: Bij andere aspecten van notificeren is gebruik van de REST-stijl meer voor de hand liggend. Bijv. voor het geautomatiseerd maken, bijwerken en verwijderen van abonnementen op notificaties. ‘Abonnementen’ zijn daarbij te zien als resource die geraadpleegd en bewerkt kunnen worden via de binnen REST gebruikelijke HTTP-methodes.

2.6 DOMAIN DRIVEN DESIGN

Bij notificatieprocessen wordt informatie uitgewisseld tussen applicaties die gebruikt worden binnen verschillende ‘taakgebieden’ of ‘domeinen’. Ieder domein kent zijn eigen taal, processen, gegevens en applicaties. Om uitwisseling van gegevens effectief te laten verlopen zijn op verschillende gebieden afspraken en standaarden nodig. Producers nemen in notificaties gegevens op over gebeurtenissen die binnen hun domein hebben plaatsgevonden. Voor Consumers moet goed zijn beschreven wat de precieze betekenis daarvan is. Pas dan kunnen zij notificatiegegevens goed interpreteren en bijv. vertalen naar begrippen die binnen hun eigen domein worden gebruikt.

Domain driven design is een stijl waarbij domeinen worden gemodelleerd met als uitgangspunt de kennis van domein-experts. Business-vertegenwoordigers en softwareontwikkelaars gebruiken daarbij dezelfde taal (‘ubiquitous language’) met domeinbegrippen. Er wordt onderkend dat binnen grotere systemen, waar bijv. meerdere afdelingen of organisaties deel van uitmaken, slechts tot op zekere hoogte standaardisatie van taal en modellen mogelijk is. Men acht het ook niet wenselijk om wat inherent verschillend is geforceerd eenvormig te maken.

*“In an ideal world, we would have a single model spanning the whole domain of the enterprise.
... Total unification of the domain model for a large system will not be feasible or cost-effective”
– bron: Domain Driven Design, Eric Evans*

Binnen verschillende domeinen worden verschillende talen gesproken. Vanwege het taakspecifieke karakter vaak om goede redenen. Waar het ene domein het over een ‘klant’ heeft kan in een ander domein de term ‘cliënt’ worden gebruikt. Als twee domeinen het over een ‘klant’ hebben kan daar iets heel anders mee worden bedoeld. Domain Driven Design benadrukt het belang van een goede afbakening van wat tot een domein behoort (‘bounded context’). Van alle begrippen, inclusief binnen het domein plaatsvindende gebeurtenissen, moet volkomen duidelijk zijn wat de precieze betekenis is.

Domain driven design is voor notificeren extra bruikbaar als stijl omdat wordt onderkend hoe belangrijk het is om verschillende soorten domeingebeurtenissen in kaart te brengen en beschrijven:



Model information about activity in the domain as a series of discrete events. Represent each event as a domain object. These are distinct from system events that reflect activity within the software itself, although often a system event is associated with a domain event, either as part of a response to the domain event or as a way of carrying information about the domain event into the system. A domain event is a full-fledged part of the domain model, a representation of something that happened in the domain. Ignore irrelevant domain activity while making explicit the events that the domain experts want to track or be notified of, or which are associated with state change in the other model objects – bron: Domain-Driven Design Reference – Definitions and Pattern Summaries – Eric Evans

'Event storming' is een op domain driven design gebaseerde techniek waarbij binnen het domein plaatsvindende gebeurtenissen, geformuleerd in business-taal, het uitgangspunt vormen ('events first'). De techniek is zeer geschikt als eerste stap in het ontwerpen van event-driven oplossingen. Het leidt tot een fundamenteel andere manier van ontwerpen dan bij proces- of datagerichte ontwerpmethoden. Daarbij bestaat bijv. het risico bestaat dat veel informatie over gebeurtenissen verloren gaat omdat ze vertaald worden in termen van CRUD-mutaties¹⁴.

Overweeg gebruik van event storming om uitgaande van domeingebeurtenissen event-driven oplossingen te ontwerpen

Gelet op de grote diversiteit binnen het overheidslandschap moet bij het inrichten van notificatieprocessen veel aandacht worden geschonken aan verschillen tussen domeinen. Er moet een balans worden gezocht tussen de wens om te standaardiseren en het accepteren van, met goede redenen aanwezige, inhoudelijke verschillen. Domain Driven Design als stijl kan daarbij behulpzaam zijn. Bijv. om te bepalen binnen welk domein bepaalde gegevens en functionaliteit (en dus ook notificaties daarover) thuishoren en om notificatiegegevens uit een Producer-domein goed te vertalen naar begrippen binnen Consumer-domeinen.

Overweeg gebruik van Domain Driven Design voor notificatieprocessen die domeingrenzen overschrijden

Om goede notificatieoplossingen te kunnen ontwerpen moet rekening worden gehouden met de relatie tussen domeinen. Afhankelijk van het soort relatie kunnen dan passende maatregelen worden getroffen:

- Hebben domeinen een gelijkwaardige relatie dan kunnen bepaalde gegevens bijv. gezamenlijk worden gedefinieerd en gebruikt ('Shared Kernel').
- Is de ene partij van de ander afhankelijk maar hebben beide belang bij een goede uitwisseling dan spelen overleg, afspraken en standaarden een belangrijke rol ('Customer-Supplier').
- Is de aanbieder volledig bepalend dan zullen afnemers zich daarnaar moeten schikken ('Conformist') en binnen hun eigen omgeving maatregelen treffen (bijv. via een 'Anticorruption Layer').

Bijlage 5 '[Domain driven design: Context Maps](#)' beschrijft een aantal patronen die bruikbaar zijn om de relatie tussen betrokken partijen te beschrijven en daarbij passende maatregelen in kaart te brengen.

Overweeg gebruik van Domain Driven Design om relaties tussen partijen te beschrijven en daarbij passende oplossingen te ontwerpen

¹⁴ 'CRUD' is een afkorting uit de informatica die staat voor de vier basisoperaties die op duurzame gegevens (meestal een database) uitgevoerd kunnen worden: Create (Toevoegen), Read (Opvragen), Update (Wijzigen), Delete (Verwijderen).



2.7 CONCLUSIES

Voor effectief event-driven werken en geautomatiseerd notificeren van applicaties is gebruik van meerdere bewezen architectuurstijlen mogelijk:

- De servicegeoriënteerde architectuurstijl blijft een belangrijke stijl, ook voor event-driven werken.
- Messaging is ook voor event-driven werken het meest geschikt als integratiestijl.
- De event-driven stijl is bij uitstek geschikt voor event-driven werken en notificeren.
- De REST-architectuurstijl is niet ontworpen voor event-driven werken, maar kan wel geschikt zijn omdat er veelvuldig gebruik van wordt gemaakt.
- Domain Driven Design is bruikbaar om goed inzicht te krijgen in domeinevents, relaties tussen domeinen en het ontwerpen van daarbij passende notificatieoplossingen.

Gebruik bewezen architectuurstijlen bij het ontwerpen van notificatieprocessen

Bij de keuze voor gebruik van een stijl spelen situationele factoren een grote rol. Afhankelijk van requirements en aspecten zoals aanwezige kennis, ervaring en mogelijkheden van betrokken organisaties bepalen sterk mee welke stijl(en) het meest geschikt zijn.

Gebruik stijlen die geschikt zijn binnen de specifieke context

Het is uiteraard ook mogelijk om gebruik te maken van een combinatie aan stijlen. Wanneer voor notificeren gebruik wordt gemaakt van een Intermediary kan voor de communicatie tussen aanbieder en intermediair bijv. gebruik worden gemaakt van een andere stijl dan voor de communicatie tussen Intermediary en Consumers. Voor afnemers geldt dat er, zeker bij grote aantallen, vaak verschillen zijn in aanwezige voorkeuren en mogelijkheden. Gebruik van verschillende stijlen kan dan wenselijk of zelfs noodzakelijk zijn.

Gebruik meerdere stijlen bij diversiteit in betrokken applicaties

3 SOORTEN GEBEURTENISSEN EN NOTIFICATIES

3.1 INLEIDING

Er zijn verschillende manieren om gebeurtenissen en notificaties te typeren en onderscheiden. Het is belangrijk dat voor alle betrokkenen duidelijk is over wat voor soort gebeurtenissen we het hebben, hoe we ze noemen en wat de precieze betekenis is. Dit hoofdstuk beschrijft een aantal verschillende typen gebeurtenissen die van belang zijn om binnen de architectuur te onderscheiden.

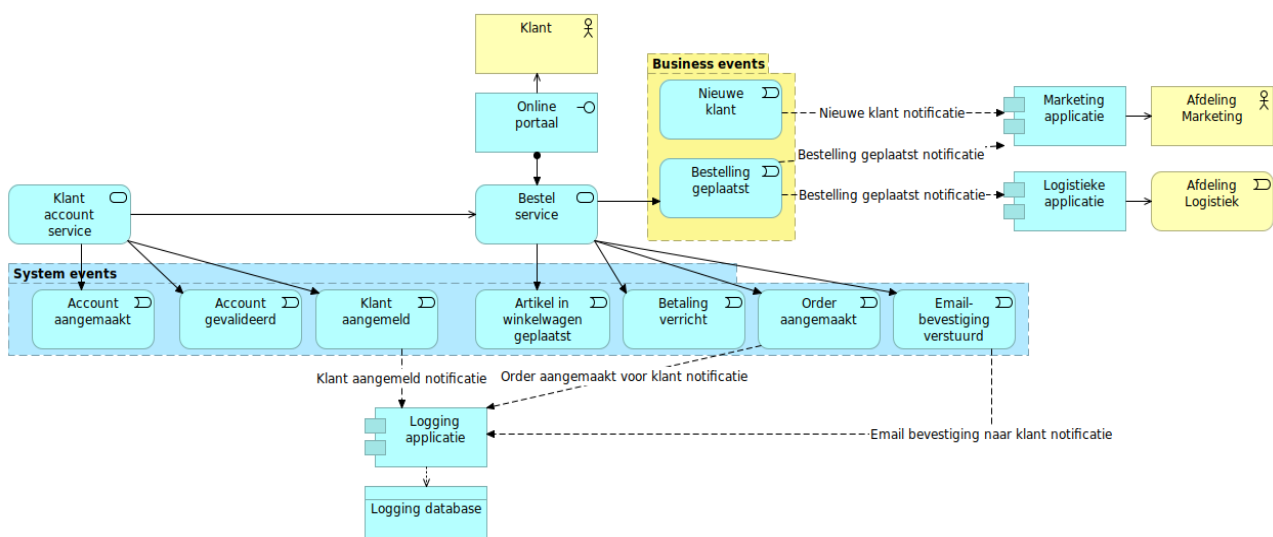
De notitie '[Introductie van notificeren](#)' beschrijft op een laagdrempelige manier een aantal soorten gebeurtenissen die binnen de overheid van belang zijn.

3.2 BUSINESS- EN SYSTEEM-GEBEURTENISSEN

Een *business-gebeurtenis* (of 'bedrijfsgebeurtenis') is een gebeurtenis die plaatsvindt binnen een bepaald businessdomein. Zij heeft binnen het domein een duidelijke betekenis en wordt uitgedrukt in de taal van business-vertegenwoordigers. Bijvoorbeeld: 'order geplaatst' of 'factuur verzonden'. Dit type gebeurtenissen vinden plaats, ongeacht of en hoe ze met behulp van software en technologie worden vastgelegd.

Een *systeem-gebeurtenis* is een gebeurtenis die plaatsvindt binnen software- of registratiesystemen. Het kan een neerslag zijn van een business-gebeurtenis en bijvoorbeeld leiden tot het creëren of wijzigen van bepaalde gegevens. Zo kan de business-gebeurtenis 'order geplaatst' leiden tot systeem-gebeurtenissen als 'order created' en 'order items created'. Systeem-gebeurtenissen zijn dus meer technisch van aard en hebben met name betekenis voor mensen die bijv. informatiemodellen of software ontwikkelen. Notificaties over systeem-gebeurtenissen bevatten vaak gegevens over binnen registraties opgeslagen gegevensobjecten en hun kenmerken.

Eén business-gebeurtenis leidt vaak tot meerdere systeem-gebeurtenissen. Eén transactie kan bijv. leiden tot het creëren van meerdere gerelateerde gegevensobjecten ('entiteiten'). Onderstaand diagram illustreert hoe bij een online geplaatste bestelling door een nieuwe klant sprake is van een tweetal business-gebeurtenissen ("Nieuwe klant" en "Bestelling geplaatst") en zeven onderscheiden systeem-gebeurtenissen.



Het diagram laat zien hoe voor de bedrijfsvoering, hier bij de afdelingen Marketing en Logistiek, met name notificaties over business-gebeurtenissen relevant zijn (en dus niet de onderliggende systeem-gebeurtenissen).



Om goed te kunnen notificeren over business-gebeurtenissen moet zijn overeengekomen welke gebeurtenissen relevant zijn en hoe ze moeten worden geformuleerd. Daarbij gelden algemene uitgangspunten zoals gebruik van de voltooid verleden tijd. Maar een belangrijk aspect is ook om gebruik te maken van de taal uit het business-domein. Dit zorgt voor eenduidigheid en herkenbaarheid en leidt tot goede communicatie tussen business-vertegenwoordigers, domeinexperts en IT'ers.

Benoem en specificeer business-gebeurtenissen in de taal van business-vertegenwoordigers

Bepalen welke soorten gebeurtenissen er allemaal binnen een domein plaatsvinden, hoe ze moeten worden genoemd en wat de precieze betekenis is, is niet eenvoudig. Er is daarom een aparte handreiking '[Gebeurtenistypes definiëren](#)' gemaakt die o.a. ondersteunt bij:

- Vaststellen welke soorten business-gebeurtenissen zich binnen een domein voordoen.
- Beslissen voor welke gebeurtenistypes notificaties verstrekt moeten worden.
- Beschrijven van gebeurtenistypes zodat Consumers die daarover notificaties ontvangen de precieze betekenis daarvan kennen.

Op dit moment vindt systeemontwikkeling binnen de overheid vooral data-gedreven plaats. Bijvoorbeeld door informatiemodellen te maken die gegevens en hun onderlinge relaties beschrijven. Op basis daarvan worden dan bijv. elementaire operaties gedefinieerd waarmee gegevens actueel worden gehouden (bijv. via services die CRUD-operaties uitvoeren). Bij deze manier van werken wordt binnen registraties vaak weinig of geen informatie vastgelegd over gebeurtenissen die de aanleiding waren voor gegevenswijziging. Op basis van aanwezige registratiegegevens kunnen daarom uitsluitend notificaties over systeem-gebeurtenissen worden verstrekt. Deze kunnen nuttig zijn voor systeemtechnische doelen zoals het binnen een andere omgeving repliceren van gegevens uit bronregistraties. De verwachting is dat de behoefte aan notificeren over systeem-gebeurtenissen zal afnemen als brongegevens vaker realtime opvraagbaar en er meer event-driven met notificaties wordt gewerkt.

In de meeste situaties geldt dat Consumers notificaties willen ontvangen over (inhoudelijke) business-gebeurtenissen. Daarvoor is het nodig om meer dan nu gebruikelijk is binnen projecten aandacht te besteden aan wat er binnen een domein *gebeurt*: gebeurtenissen moeten 'first-class citizens' worden. Er moet (ook) vroegtijdig 'gebeurtenis-analyse' plaatsvinden. Domeinexperts en IT'ers moeten daarvoor samen vroegtijdig in kaart brengen welke triggers leiden tot business-gebeurtenissen, wie daarvoor verantwoordelijk is, welke gegevens er betrokken zijn, etc. (bijv. via een methode als '[event storming](#)'). Dit betekent een fundamentele verandering ('paradigmashift') van werkwijze bij het ontwerpen en realiseren van oplossingen.

Analyseer business-gebeurtenissen om event-driven oplossingen te ontwerpen

Bij nieuwe doelen en een andere werkwijze zijn andere denkwijzen, patronen, methoden en technieken dan gebruikelijk nodig. Dit document beschrijft een aantal voorbeelden hiervan zoals de Domain Driven Design stijl, het Publish-Subscribe patroon en de event-storming methode. Qua techniek kan het betekenen dat het wenselijk is om tot dan tot niet gebruikte gegevensformaten, protocollen en (middleware)voorzieningen worden onderzocht en ingezet. Passend bij een paradigmashift zal op meerdere gebieden met een open blik moeten worden gezien hoe optimale event-driven oplossingen zijn te ontwerpen.

Onderzoek en gebruik nieuwe denkwijzen, patronen, methoden en technieken bij het ontwerpen van event-driven oplossingen

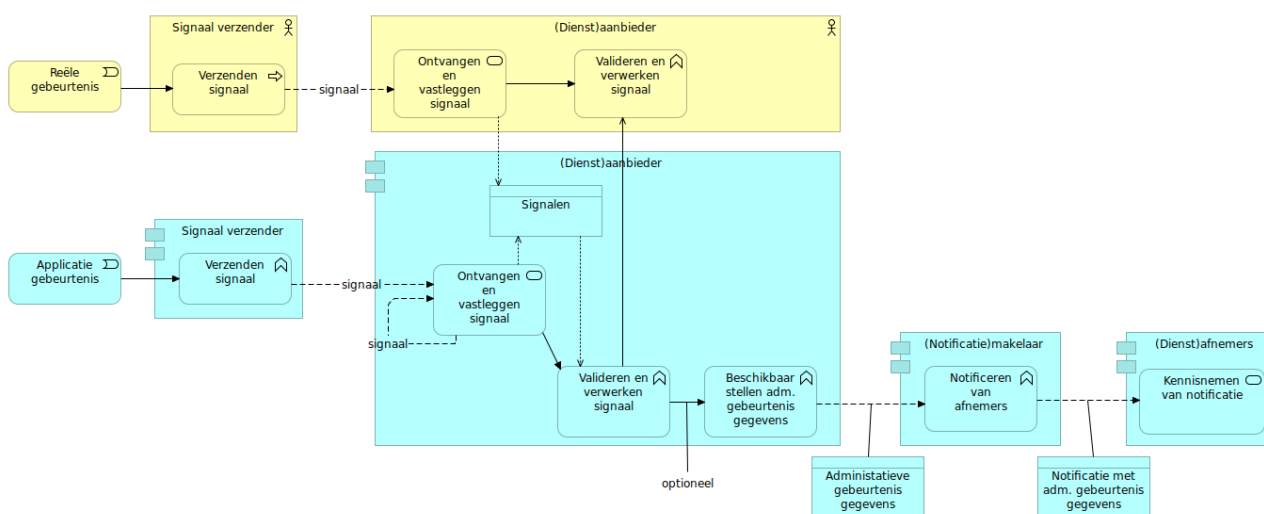
Een belangrijk voordeel van een event-driven mindset is dat het denken en werken in kokers of silo's minder wordt. Want notificaties zijn niet alleen nuttig binnen het eigen domein maar ook om de uitwisseling van

informatie met andere domeinen te verbeteren. Via notificeren zijn partijen vaker en sneller van actuele informatie te voorzien en ontstaat een groeiend netwerk waarin event-driven wordt samengewerkt. Vroegtijdig aandacht besteden aan mogelijke meerwaarde van notificaties voor partijen buiten het eigen domein bevordert interoperabiliteit.

Besteed vroegtijdig aandacht aan meerwaarde van notificaties voor partijen buiten het eigen domein

3.3 REËLE EN ADMINISTRATIEVE GEBEURTENISSEN

Het woord 'gebeurtenis' wordt in het dagelijks spraakgebruik gebruikt om een voorval of feit dat zich afspeelt in de werkelijkheid te beschrijven. We noemen dit 'een reële gebeurtenis'. Informatie hierover kan op allerlei manieren beschikbaar komen en moet bijv. eerst op geldigheid worden gecontroleerd voordat er waarde aan kan worden ontleend. Onderstaand diagram toont een voorbeeld waarbij enkele processtappen nodig zijn voordat gegevens over een plaatsgevonden reële gebeurtenis als notificatie aan Consumers kan worden verstrekt.



Figuur 2: Voorbeeldproces waarbij reële gebeurtenissen kunnen leiden tot notificaties aan Consumers

Een 'signaal verzender' kan een signaal geven dat er een reële gebeurtenis heeft plaatsgevonden. Dit wordt gezien als een 'bewering' die nog moeten worden gevalideerd voordat verwerking plaats mag vinden. Zowel het verzenden van signalen als het valideren kan zowel door mensen als applicaties gebeuren. Bijv. door een balie medewerker die een klant vraagt om bepaalde bewijsstukken te tonen of door een applicatie die binnen een eFormulier ingevulde gegevens controleert. Na geslaagde validatie wordt het ontvangen signaal verwerkt binnen bedrijfs- en applicatieprocessen en vindt er een 'administratieve gebeurtenis' plaats. Die leidt vaak tot creatie of wijziging van gegevens in een bronregister¹⁵. Een plaatsgevonden reële gebeurtenis wordt hierbij dus 'vertaald' naar een administratieve gebeurtenis die leidt tot creatie, wijziging waarbij of verwijdering van bepaalde brongegevens¹⁶.

Naast externe gebeurtenissen waarover in de vorm van een signaal informatie binnenkomt kunnen ook interne administratieve gebeurtenissen leiden tot nieuwe signalen. Zo kan er binnen de eigen organisatie, en binnen dezelfde applicatie, dus ook event-driven worden gewerkt waarbij (systeem)gebeurtenissen leiden

¹⁵ We beschrijven hier de gebruikelijke manier van werken. Het hoofdstuk 'Implementatiepatronen' beschrijft (ook) patronen waarbij dit fundamenteel anders verloopt zoals het 'Event sourcing' patroon.

¹⁶ De termen "reel" en "adminsitratief" zijn verwant maar niet identiek aan de termen "materieel" en "formeel". Die worden gebruikt aan te duiden of het gegevens betreft over wat zich in de fysieke of in de administratieve werkelijkheid heeft afgespeeld (bijv. een datum die het moment van plaatsvinden van een gebeurtenis aanduidt of een datum die het moment van administratieve vastlegging daarvan aanduidt).



tot notificaties die door in- of externe processen zijn te verwerken. Event-driven mechanismen zijn dus op allerlei niveaus aanwezig.

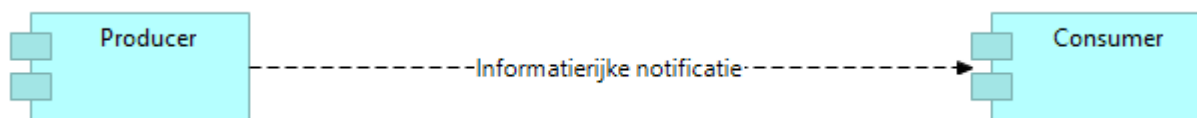
De manier waarop reële gebeurtenissen binnen applicaties worden vertaald in administratieve gebeurtenissen bepaalt in sterke mate hoe Consumers via notificaties worden geïnformeerd over plaatsgevonden reële gebeurtenissen. Op dit moment gaat bij die vertaling nog vaak veel informatie over gebeurtenissen verloren. Bijv. omdat alleen het effect ervan in de vorm van gewijzigde gegevens constateerbaar is. Bij andere vormen van vastlegging zoals [event sourcing](#) wordt (ook) contextuele gebeurtenisinformatie bewaard waardoor betekenisvollere notificaties aan Consumers zijn te verstrekken.

3.4 INFORMATIERIJKE EN INFORMATIEARME NOTIFICATIES

Met 'informatierijk' bedoelen we dat er binnen een notificatie een rijke set aan gegevens wordt verstrekt over een plaatsgevonden gebeurtenis. Consumers kunnen normaal gesproken met deze gegevens direct overgaan tot verwerking en hoeven geen aanvullende informatie op te vragen. We spreken over 'informatierijk notificeren' en 'informatierijke notificaties'.

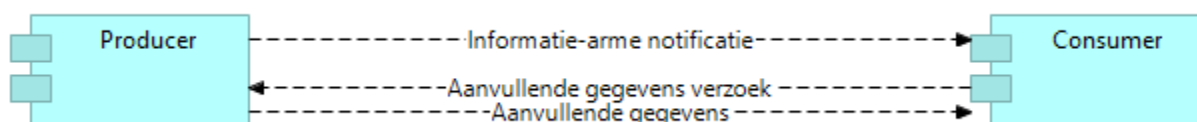
Informatierijke notificaties geven niet alleen informatie dát zich een bepaalde gebeurtenis heeft voorgedaan maar ook wát dit aan gevolgen had. Bijv. door in de notificatie gegevens op te nemen over de plaatsgevonden gebeurtenis (bijv. de reden waarom een gebeurtenis plaatsvond) of over gegevensobjecten die als gevolg van de gebeurtenis zijn gewijzigd (bijv. dat het kenmerk 'status' van een object is gewijzigd)¹⁷.

Vanuit afnemerperspectief bestaat informatierijk notificeren uit een informatiestroom waarbij een Producer een notificatie verstrekt aan de Consumer.



Onder 'informatiearme notificaties' verstaan we notificaties waarin slechts (zeer) beperkt informatie is opgenomen. Consumers kunnen na ontvangst constateren dát zich een bepaalde gebeurtenis heeft voorgedaan maar zullen voordat tot verwerking kan worden overgegaan normaal gesproken eerst aanvullende gegevens moeten opvragen. In de notificatie kan daarvoor bijv. een link zijn opgenomen die verwijst naar een service-API waar de benodigde aanvullende gegevens zijn op te vragen ('[Claim Check patroon](#)')¹⁸.

Vanuit afnemerperspectief bestaat informatiearm notificeren uit een patroon waarbij 3 informatiestromen zijn te onderscheiden:



'Informatiearm' en 'informatierijk' zijn relatieve begrippen. Informatiearme notificaties kunnen meer informatie bevatten dan alleen dát zich een bepaald type gebeurtenis heeft voorgedaan. Bijv. omdat afnemers (ook) moeten weten op welk(e) object(en) een gebeurtenis betrekking had om te kunnen bepalen of en zo ja, welke aanvullende opvraging(en) nodig zijn. Hierdoor ontstaat een mengvorm van 'arm' en 'rijk' die soms wordt aangeduid met de term: 'hybride eventing'.

¹⁷ Op deze manier informatie overdragen naar consumers wordt ook aangeduid met de term 'Event-Carried State Transfer'.

¹⁸ Minimaal informatie verstrekken via notificaties wordt ook aangeduid met de term 'Event Notification'.



Een aandachtspunt daarbij is dat naarmate er op verzoek van Consumers meer gegevens in notificaties worden opgenomen er een steeds sterkere koppeling tussen aanbieder en afnemers ontstaat. Om ongewenste koppeling tegen te gaan is het daarom wenselijk om bij gebruik van informatiearm notificeren inhoudelijke gegevens zo beperkt mogelijk te houden.

Neem in informatiearme notificaties alleen minimaal benodigde gegevens op

De keuze voor gebruik van informatiearme of informatierijke notificaties is van verschillende factoren afhankelijk. In zijn algemeenheid geldt dat informatiearm notificeren lastiger is toe te passen omdat er aanvullende inspanningen nodig zijn bij zowel Consumer als Producer¹⁹. In situaties waarin informatierijk notificeren voldoet kan het vanwege de eenvoudiger implementatie daarom de voorkeur verdienen. Bijv. in situaties waarin een beperkte set openbare gegevens wordt verstrekt aan Consumers met beperkte mogelijkheden voor wie opvraging van aanvullende gegevens lastig is.

Notificeer informatierijk als dit eenvoudig en verantwoord kan

Er zijn echter legio situaties, zeker binnen de overheid, waarin informatiearm notificeren de voorkeur kan verdienen. Bijv. als er sprake is van verstrekking van vertrouwelijke gegevens waarbij dataminimalisatie nodig is. Verstrekte notificaties bevatten dan minimaal of geen inhoudelijke gegevens maar bijv. alleen metagegevens waaruit is te concluderen dat zich een bepaald type gebeurtenis heeft voorgedaan. Voor het opvragen van aanvullende vertrouwelijke gegevens kan dan vaak gebruik worden gemaakt van al bestaande services die beschikken over authenticatie- en autorisatiefunctie. Deze kunnen dan worden hergebruikt om te borgen dat alleen gegevens aan Consumers worden verstrekt waartoe zij zijn geautoriseerd. Informatiearm notificeren kan daarmee een geschikte vorm van notificeren zijn om verstrekking van informatie in overeenstemming met privacywetgeving uit te voeren.

Overweeg informatiearm notificeren bij de verstrekking van vertrouwelijke gegevens

Een andere reden om informatiearm te notificeren is dat er grote hoeveelheden gegevens moeten worden verstrekt en levering daarvan binnen notificatieberichten tot problemen leidt. Bijv. omdat in gebruik zijnde middleware en afnemersservices niet in staat zijn om berichten van grote omvang te verwerken. Informatiearm notificeren is dan geschikt omdat de verstrekking van de grote hoeveelheid gegevens na notificatie plaats kan vinden via een ander, daarvoor geschikt, kanaal (bijv. via een streaming connectie of gebruik van protocollen zoals FTP)²⁰.

Overweeg informatiearm notificeren bij de verstrekking van grote hoeveelheden gegevens

Zowel bij informatierijk als informatiearm notificeren kunnen zich bijzondere situaties voordoen waarbij passende maatregelen nodig zijn. Een voorbeeld hiervan zijn notificaties die een correctie bevatten van eerder genotificeerde informatie. De manier waarop een ontvangen correctie moet worden verwerkt kan per Consumer verschillen en kan variëren van 'niks bijzonders doen' tot het uitvoeren van complexe, geautomatiseerde of handmatige, compenserende transacties. Bij informatiearm notificeren moet expliciet rekening worden gehouden met tijdigheid van gebeurtenis en notificatie. Idealiter wordt bij opvraging van aanvullende gegevens rekening gehouden met het tijdstip waarop de notificatie betrekking had en kunnen

¹⁹ Omdat het in de praktijk meestal het geval is gaan we er hier gemakshalve van uit dat de partij in de rol van Producer ook de partij is bij wie aanvullende gegevens zijn op te vragen. In de praktijk kan het ook voorkomen dat aanvullende gegevens bij een of meer andere partijen zijn op te vragen.

²⁰ Binnen messaging wordt zo'n patroon aangeduid als het '[claim-check pattern](#)'. Ontvangers kunnen na ontvangst van een eerste minimaal bericht via een ander, voor grote hoeveelheden gegevens geschikt, kanaal ('out of band') gegevens ophalen.



de op dat moment geldende gegevens worden geleverd. Veel systemen zijn daartoe echter niet in staat en leveren gegevens zoals die gelden op het moment van opvraging (die mogelijk afwijken van de gegevens ten tijde van notificeren). Het moet duidelijk zijn voor afnemers als dit zich voor kan doen zodat zij, afhankelijke van context en noodzaak, daar eventueel rekening mee houden bij verwerking.

Verschaf duidelijkheid over tijdigheid rondom gebeurtenissen, notificaties en opvragingen zodat Consumers waar nodig passende maatregelen kunnen nemen

3.5 LEVENSGEBEURTENISSEN

Met 'levensgebeurtenissen' worden binnen de overheid belangrijke gebeurtenissen bedoeld die zich tijdens het leven van een persoon kunnen voordoen. Bijvoorbeeld de geboorte van een kind, een huwelijk of het vinden of verliezen van een baan. Levensgebeurtenissen worden vaak gebruikt als zoekingang om mensen die te maken hebben met een bepaalde levensgebeurtenis informatie op maat aan te bieden (zie bijv. de voorbeelden in dit [overzicht](#)) of om burgers en ondernemers een zo gebruiksvriendelijk mogelijke 'klantreis' aan te bieden.

Levensgebeurtenissen kunnen gevolgen kunnen hebben op allerlei gebieden. Het feit dat een persoon is verhuisd of 18 jaar is geworden kan bijvoorbeeld relevant zijn voor bedrijfsprocessen en registraties in verschillende domeinen. Binnen de context van project Notificatieservices zien we levensgebeurtenissen als *reële gebeurtenissen* die binnen meerdere domeinen kunnen leiden tot *business-gebeurtenissen* (en daarmee samenhangende *systeem-gebeurtenissen*). Ze vormen een belangrijke categorie gebeurtenissen omdat binnen de overheid vaak meerdere diensten moeten worden uitgevoerd naar aanleiding van het optreden ervan.

Binnen de overheid bestaat al lange tijd de wens is om burgers en ondernemers die een levensgebeurtenis meemaken een 'integrale klantreis' aan te bieden waarbij meerdere overheidsorganisaties betrokken kunnen zijn. Dit blijkt in de praktijk niet eenvoudig realiseerbaar omdat de Nederlandse overheid gelaagd is opgebouwd en gedistribueerd werkt. Onderlinge samenwerking is, ook via automatisering, vaak lastig te realiseren waardoor de burger vaak zelf stap voor stap bij alle betrokken organisaties langs moet. Event-driven werken en gestandaardiseerd notificeren is bij uitstek geschikt om als Nederlandse overheid effectiever samen te kunnen werken. Organisaties kunnen daarbij hun kerntaken autonoom blijven uitvoeren, maar elkaar wel via notificaties snel attenderen. Iedere betrokken organisatie kan dan in actie kan komen als dat nodig is waardoor de klantreis van betrokken burger of bedrijf aanzienlijk is te verbeteren.

4 ASPECTEN

4.1 INLEIDING

Dit hoofdstuk beschrijft een aantal aspecten waar bij het inrichten van notificatieprocessen rekening mee moet worden gehouden. Het gaat daarbij om ‘kwaliteitsaspecten’ die los staan van de inhoud van notificaties maar wel belangrijk zijn om notificatieprocessen betrouwbaar uit te kunnen voeren. Het gaat daarbij bijv. om aspecten zoals veiligheid en betrouwbaarheid van uitvoering en aspecten zoals aanpasbaarheid en schaalbaarheid die duurzaam gebruik mogelijk maken.

Voor ieder aspect geldt dat er in de praktijk keuzemogelijkheden zijn met eigen voor- en nadelen. Er worden aanbevelingen gedaan met betrekking tot te maken keuzes maar daarbij geldt dat contextuele factoren andere keuzes kunnen rechtvaardigen.

Sommige van de hier genoemde aspecten worden in een apart hoofdstuk meer in detail beschreven.

4.2 SYNCHRONE EN ASYNCHRONE COMMUNICATIE

Communicatie verloopt *synchroon* of *asynchroon*. Bij synchrone communicatie vindt de communicatie tussen verschillende actoren *direct en gelijktijdig* plaats. Daarbij wordt een sessie opgezet die in stand blijft tot hij expliciet wordt verbroken. Een bekend voorbeeld hiervan is een telefoongesprek.

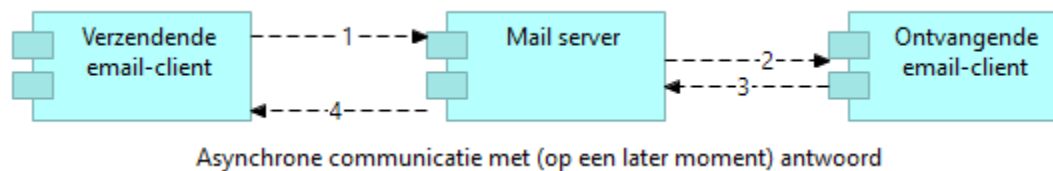


Synchrone communicatie waarbij partijen gelijktijdig actief moeten zijn

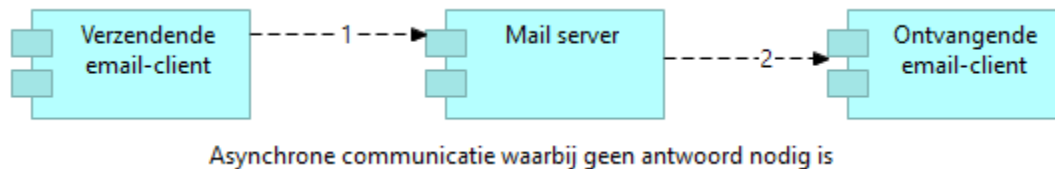
Bij een synchroon notificatieproces verstuurt de aanbieder een notificatie naar een of meer afnemers en wacht op de antwoorden van afnemers. Aan de hand van het al dan niet ontvangen antwoord kan een aanbieder bijv. concluderen er nog vervolgacties nodig zijn. Wanneer er binnen een bepaalde tijd geen antwoord of een foutmelding komt moet hij bepalen wat hij gaat doen. Hij kan bijvoorbeeld besluiten om notificaties met een bepaald tijdsinterval en een maximaal aantal pogingen opnieuw te gaan versturen.

Synchrone communicatie is relatief eenvoudig te implementeren maar kent ook nadelen omdat er ongewenste afhankelijkheden tussen applicaties kunnen ontstaan. Wanneer een van de betrokken applicaties tijdens het communicatieproces niet goed functioneert of niet actief is stopt het proces ('domino-effect'). Naarmate meer applicaties met een bepaalde betrouwbaarheid zijn betrokken neemt de betrouwbaarheid van het totale proces steeds sneller af (bijv. $99\% * 99\% * 99\% = 97\%$). Door onvoldoende ontkoppeling kan bij synchrone communicatie dus een situatie ontstaan waarin het moeilijk is om betrouwbaar te notificeren. Zeker in een gedistribueerd landschap met veel verschillende organisaties en applicaties, zoals bij de overheid vaker het geval is, zullen regelmatig verstoringen optreden ("Everything fails, all the time" - Werner Vogels (CTO Amazon)). Er is dan correctieve actie nodig waarvoor speciale functionaliteit nodig is bij Producer en/of Intermediary en/of Consumers (bijv. retries of gebruik van volgnummers).

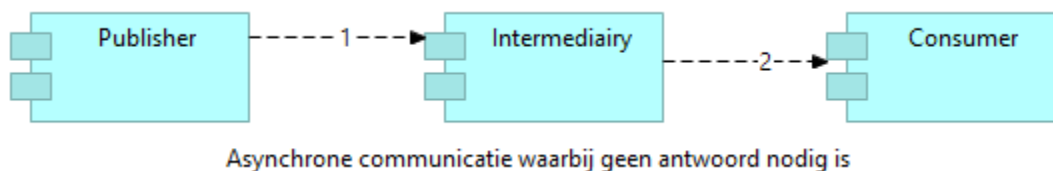
Bij asynchrone communicatie hoeven, in tegenstelling tot bij synchrone communicatie, applicaties niet gelijktijdig actief te zijn. Informatie wordt dan in 'pakketjes' getransporteerd en kan bijv. worden bewaard als aflevering tijdelijk niet mogelijk is. Applicaties worden hierdoor meer ontkoppeld wat de robuustheid van gegevensuitwisseling ten goede komt. Een bekend voorbeeld van asynchrone uitwisseling is briefpost of email: verzender en ontvanger kunnen daarbij op verschillende, zelf te bepalen, momenten via onafhankelijke kanalen berichten verzenden en ontvangen.



Bij 1-richting verkeer, waarbij de verzender geen antwoord van de ontvanger hoeft te hebben, is het patroon nog eenvoudiger. Een voorbeeld hiervan is de verzending van nieuwsbrieven via email.



Bij notificatieprocessen wordt dit patroon het vaakst toegepast. In hoofdstuk 6, 'Het Publish-Subscribe patroon' wordt gebruik van dit patroon verder toegelicht.



Functioneel asynchroon werkende systemen²¹ maken gebruik van 'store-and-forward' technieken om tijdverschillen te compenseren zodat applicaties op verschillende momenten informatie kunnen verwerken. Omdat applicaties onafhankelijk van elkaar kunnen functioneren wordt het mogelijk om notificatieprocessen flexibel, robuust en schaalbaar in te richten. Een aanbieder van gebeurtenisgegevens kan bijv. zijn taak, het beschikbaar stellen van informatie, autonoom uitvoeren zonder afhankelijk te zijn van het verwerken van notificaties door afnemers²².

Asynchronous communication is fundamentally a pragmatic reaction to the problems of distributed systems. Sending a message does not require both systems both systems to be up and available at the same time. Furthermore, thinking about the communication in an asynchronous manner forces developers to recognize that working with a remote application is slower and prone to failure, which encourages design of components with high cohesion (lots of work locally) and low adhesion (selective work remotely). - Enterprise Integration Patterns

Binnen de context van project Notificatieservices richten we ons op notificatieprocessen waarbij 1-richting verkeer plaatsvindt ('one-way-messaging'). Aanbieders hebben daarbij geen inhoudelijke antwoorden nodig van afnemers (wat bijv. wel het geval is bij gedistribueerde transacties waarbij consistentie nodig is). Als er al antwoorden nodig zijn betreft het uitsluitend 'technische ontvangstbevestigingen' ('ACK'). Op basis daarvan kan de applicatie die notificeert bijv. concluderen dat zijn taak succesvol is afgerond en (afhankelijk van de afspraken) de notificatie kan verwijderen. De verantwoordelijkheid voor verwerking van de notificatie ligt volledig bij de afnemer.

²¹ 'Asynchroon werken' kan zowel betrekking hebben op functionaliteit als techniek (bijv. i.r.t. transportprotocollen). We richten ons hier op functionele aspecten.

²² Afhankelijk van de inrichting van het notificatieproces kan de aanbieder een ontvangstbevestiging ('ACK') willen ontvangen. Is dat niet nodig dan volstaat voor de aanbieder het eenvoudige 'fire and forget' patroon.



Gelet op de voordelen die asynchrone berichtuitwisseling tussen applicaties biedt verdient het de voorkeur om hiervan gebruik te maken bij notificatiesprocessen.

Maak bij notificatieprocessen gebruik van (functioneel) asynchrone berichtuitwisseling

Voor notificeren kan, net als voor andere vormen van asynchroon plaatsvindende berichtuitwisseling, gebruik worden gemaakt van gespecialiseerde voorzieningen. Daarmee is het mogelijk om op verschillende manieren berichten betrouwbaar te ontvangen, bewaren en verstrekken. Bijv. door gebruik te maken van message queues zodat direct na ontvangst een technische ontvangstbevestiging kan worden verstuurd en specifieke functionaliteit zorgt voor optimale bewaring en verstrekking. Met de opkomst van Cloud-voorzieningen komt er steeds meer keuze in voorzieningen om de rol van Intermediary ('event broker') in te vullen en asynchrone berichtuitwisseling mogelijk te maken. Notificaties kunnen bijv. tussentijds worden opgeslagen om fouten te kunnen herstellen of notificaties langer beschikbaar te houden. Er kunnen hoge doorvoersnelheden van gegevens mee worden gerealiseerd of de intermediary kan juist als 'schokdemper' fungeren voor afnemers voor wie het tempo waarin gegevens beschikbaar komen te hoog is.

That is what asynchronous messaging infrastructure does, but for your apps.

- *Decoupling: A system handling work behind a messaging infrastructure can be running at capacity and yet not be overwhelmed and can even be down while the messaging infrastructure still accepts messages on its behalf.*
- *Delivery: You can entrust over your messages and the messaging system will try its best not lose them. It will then attempt to deliver them to the right parties and will retry as often as necessary.*
- *Buffering: A messaging infrastructure is generally great at accepting bursts of messages at once and organizing them for later retrieval. The retrieval can then occur at the pace that your application can handle. That is also called load-leveling.*
- *Network-Bridging: Messaging infrastructures can often be attached to multiple networks, allowing information to pass between applications in those networks without there being IP-level connectivity between them.*

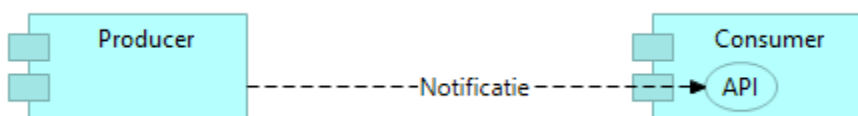
Bron: [Asynchronous Messaging and Eventing Resources – Why would I care?, C.Vasters](#)

In lijn met het 'seperation of concerns' principe geldt dat, met uitzondering van zeer eenvoudige notificatieprocessen, het snel wenselijk is om voor de uitvoering van notificatieprocessen gebruik te maken van gespecialiseerde voorzieningen. Vaak aangeduid met de termen: 'message oriented middleware' en 'event-driven middleware'.

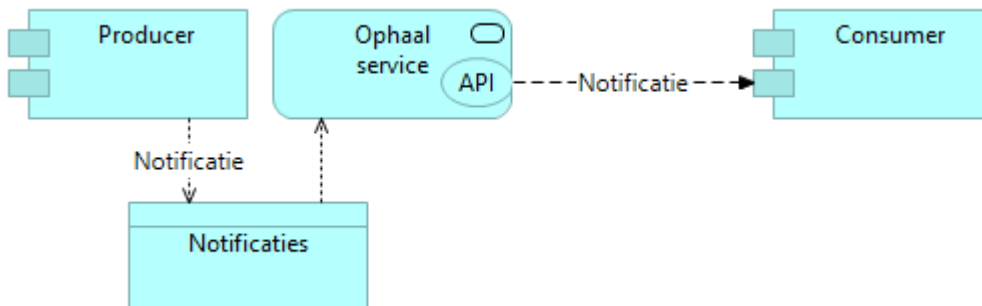
Gebruik message- of event-oriented middleware voor ondersteuning van asynchroon berichtenverkeer

4.3 PUSH EN PULL MECHANISMEN

Met 'push mechanismen' bedoelen we mechanismen waarbij notificaties bij afnemers worden afgeleverd. Dit kan bijv. wenselijk zijn als afnemers meteen genotificeerd willen worden als zich bepaalde gebeurtenissen hebben voorgedaan. De afnemer moet hiervoor uiteraard in staat zijn om notificaties te ontvangen. Bijv. via een service die een Application Programming Interface (API) kent waar een bericht is af te leveren.



Met 'pull mechanisme' bedoelen we mechanismen waarbij de aanbieder notificaties op een afgesproken manier beschikbaar stelt maar afnemers zelf notificaties gaan ophalen. Dit kan bijv. wenselijk zijn als afnemers niet in staat zijn om op een betrouwbare manier notificaties te ontvangen. De aanbieder moet hierbij dus afnemers faciliteren om notificaties op te kunnen halen. Bijv. door een service te bieden waar via een API notificaties zijn op te vragen.



Binnen de CloudEvents-community speelt 'push' of 'pull' niet zo'n belangrijke rol omdat men zich vooral richt op de inhoud van events (en niet zozeer de manier waarop interactie plaatsvindt). Bij de in ontwikkeling zijnde standaard voor abonneren wordt wel een onderscheid tussen 'push' en 'pull' gemaakt maar 'pull' heeft daarbij een andere betekenis dan hiervoor beschreven. Ophalen van aanvullende gegevens door een Consumer ziet men niet als een 'pull patroon' maar men ziet dit als het gangbare request-reply patroon waarmee opvragingen worden gedaan. Men spreekt daar over 'pull' als een Consumer een sessie initieert met een Producer/Intermediary waarna gedurende die sessie events richting Consumer kunnen stromen. Dat kan zowel op initiatief van de Producer/Intermediary als op initiatief van de Consumer gebeuren. Protocollen zoals AMQP en MQTT ondersteunen dit type interactie.

Uiteraard zijn bij het ontwerpen van oplossingen ook combinaties van 'push' en 'pull' mogelijk. Zoals eerder beschreven bij 'informatie-arm notificeren' wordt er eerst een (informatie-arm) notificatiebericht verstrekt aan Consumers ('push') waarna die zelf aanvullende informatie gaan ophalen ('pull').

Binnen de event-driven wereld zijn push-mechanismen het meest gebruikelijk²³. Onder andere omdat Consumers hiermee snel na het plaatsvinden van een gebeurtenis op de hoogte zijn te stellen (en tijdigheid vaak een belangrijk aspect is).

Gebruik een push-mechanisme als afnemers snel over notificaties moeten beschikken

Gebruik van push-mechanismen stelt zowel eisen aan aanbieder als afnemers. Aanbieders moeten bijvoorbeeld om kunnen gaan met verstoringen bij afnemers, waardoor afleveren niet mogelijk is en op een later moment opnieuw moet plaatsvinden. Afnemers moeten, in het ideale geval 24x7, notificaties kunnen ontvangen. Tussen aanbieder en afnemers moeten daarom altijd goede afspraken bestaan. Afhankelijk van de benodigde kwaliteit van notificeren kunnen die beperkt of heel vergaand zijn. Afspraken kunnen bijv. gaan over hoe moet worden omgegaan met uitzonderingssituaties (bijv. als een afnemer notificaties in de verkeerde volgorde ontvangt) of opgetreden fouten (bijv. als een afnemer naar aanleiding daarvan notificaties opnieuw wil ontvangen).

Zorg voor goede afspraken tussen Producer en Consumers bij gebruik van push-mechanismes

²³ Sommigen zien push-mechanismen als het enige mechanisme waarbij met recht gesproken mag worden over 'event-driven'.



Hoewel een pull-mechanisme zoals hier beschreven binnen de event-driven wereld vaak niet wordt gezien als een goed mechanisme voor event-driven werken kunnen ze in bepaalde situaties, ook binnen de overheid, wel nuttig zijn. Er kan daarmee vergaande ontkoppeling van Producer en Consumers worden gerealiseerd waardoor ze volledig onafhankelijk van elkaar kunnen functioneren. Een Producer zet op een zelfgekozen moment notificaties klaar die Consumers vervolgens op door hen te bepalen momenten kunnen ophalen. In vergelijking met een push-mechanisme is dit relatief eenvoudig te realiseren en soms de enige manier om een werkende oplossing te realiseren (bijv. als Consumers niet in staat zijn om betrouwbaar notificaties te ontvangen). Het kan ook bijkomende voordelen bieden zoals het faciliteren van vaker dan 1 keer notificaties ophalen (binnen de termijn dat de aanbieder ze beschikbaar stelt). Uiteraard kleven er ook nadelen aan. Bijv. het feit dat Consumers niet snel na het plaatsvinden van gebeurtenissen daarvan op de hoogte worden gesteld en dat het niet efficiënt is als alle Consumers regelmatig zelf moeten gaan kijken of er nieuwe notificaties zijn ('pollen').

Hoewel gebruik van push-mechanismen voor notificeren voordelen kent en gebruikelijker is kunnen pull-mechanismen in bepaalde situaties dus toch hun functie hebben. Binnen project Notificatieservices is geconstateerd dat organisaties binnen de overheid hele verschillende niveaus van volwassenheid kennen waar het gaat om event-driven werken en werken met IT-voorzieningen. Voor partijen waarbij het niet haalbaar is om push-gebaseerde oplossingen te realiseren zien we pull-gebaseerde oplossingen daarom als een alternatief bruikbaar mechanisme.

Gebruik pull-mechanismes voor notificatieoplossingen als gebruik van push-mechanismen niet mogelijk is

4.4 ONTKOPPELING

Bij notificatieprocessen binnen een gedistribueerd en heterogeen landschap is het belangrijk dat organisaties en applicaties zoveel mogelijk van elkaar ontkoppeld zijn en onafhankelijk kunnen functioneren.

Losgekoppeld: afnemer en leverancier zijn maximaal onafhankelijk van implementatie van beide. Elke service is daarom autonoom. Er bestaat geen directe link of relatie tussen verschillende services. – bron: Wikipedia

Ontkoppeling is bij een aantal aspecten rondom notificeren mogelijk. Of en in welke mate ontkoppeling nodig en mogelijk is op een bepaald aspect hangt uiteraard af van de specifieke context. Voor onderstaande aanbevelingen geldt daarom dat ze gezien moeten worden als algemene aanbevelingen.

- **Tijd** – Bij notificeren hoeven niet alle betrokken applicaties gelijktijdig actief te zijn. Dit vereist een vorm van 'bufferen' waarbij de Producer geen onmiddellijk antwoord nodig heeft. Bijv. door gebruik van asynchrone communicatie en queues waarin berichten tijdelijk worden bewaard.

Communiceer asynchroon tussen Publisher en Consumers om tijdonafhankelijk te functioneren

- **Plaats** – Betrokken applicaties moeten zich op verschillende plaatsen kunnen bevinden (bijv. in een on-premise en in een Cloud omgeving). Het moet mogelijk zijn om applicaties ergens anders te plaatsen. Bijv. via gebruik van zoekfuncties die naar de actuele plaats van applicaties verwijzen.
- **Tempo**: Aanbiedende applicaties die in hoog tempo notificaties verstrekken mogen geen last hebben van afnemende applicaties die notificaties niet in dat tempo kunnen verwerken. Bijv. door het tempo waarin verstrekt wordt te beperken ('throttling').

Zorg dat Producer of intermediair het tempo waarin notificaties worden verstrekt kan aanpassen aan de mogelijkheden van Consumers



- **Versie** – Notificeren moet blijven functioneren als de inhoud van notificaties in de loop der tijd wijzigt. Bijv. door tijdelijke meerdere versies van notificaties te verstrekken of door gebruik te maken van versionering van notificaties op basis waarvan Consumers kunnen bepalen of en hoe ze notificaties verwerken.

Zorg dat wijzigingen van notificaties beheersbaar verlopen en Consumers de gelegenheid hebben om ze te blijven verwerken

- **Kardinaliteit** – Aanbiedende applicaties hoeven geen rekening te houden met hoeveel en welke Consumers notificaties afnemen. De Producer volstaat met het publiceren van informatie en kan bijv. het verstrekken van notificaties aan Consumers beleggen bij een Intermediary (al dan niet van een andere organisatie).

Zorg dat Producers bij het publiceren van informatie geen rekening hoeven te houden met hoeveel en welke Consumers notificaties ontvangen

- **Vindbaarheid** – Aanroep van services en het verstrekken van meta-informatie (bijv. over gebeurtenistypen of semantiek van berichten) vindt plaats via logische namen en adressen die vindbaar zijn in catalogi.

Gebruik verwijzingen en catalogi om te zorgen dat services en meta-informatie vindbaar zijn

- **Technologie** – Betrokken applicaties kunnen van verschillende technologieën gebruik maken door het gebruik van overeengekomen standaarden.

Gebruik standaarden om gebruik van verschillende technologie mogelijk te maken

- **Transportprotocol** – Voor notificaties kunnen verschillende protocollen worden gebruikt die ieder hun eigen toepassingsgebied kennen. Ondersteun dit bijv. door van een Intermediary gebruik te maken die via protocolconversie notificaties via meerdere protocollen richting Consumers kan verstrekken.

Zorg dat bij notificeren gebruik van verschillende transportprotocollen mogelijk is

- **Gegevensformaat** – Voor notificeren is van meerdere gegevensformaten gebruik te maken met ieder hun eigen voor- en nadelen. Het vaak toegepaste JSON-formaat is bijv. compact en goed leesbaar terwijl het XML-formaat meer mogelijkheden voor beschrijving en validatie biedt maar tot grotere berichten leidt. Maak bijv. gebruik van een Intermediary om berichten van het ene naar het andere formaat te kunnen vertalen.

Zorg dat bij notificeren gebruik van verschillende gegevensformaten mogelijk is

- **Interface** – Door gebruik van interfaces wordt de interne werking van Producer en Consumers onzichtbaar gemaakt en kunnen ze eenduidig met elkaar communiceren. Gebruik bij voorkeur bekende interfaces (bijv. REST-APIs) en biedt zo nodig meerdere interfaces.

Biedt Consumers bekende en waar nodig verschillende interfaces



In de praktijk zullen de mogelijkheden van betrokken organisaties en applicaties sterk mede bepalen voor welke aspecten, en in welke mate, ontkoppeling mogelijk is. In zijn algemeenheid geldt voor notificeren dat maximale ontkoppeling van Producer, intermediair en Consumers wenselijk is.

Streef bij notificatieprocessen naar maximale ontkoppeling van betrokken applicaties

Om ontkoppeling te kunnen realiseren zijn afspraken en het gebruik van standaarden nodig. Bij voorkeur standaarden die vaak gebruikt worden zodat er veel kennis over beschikbaar is en partijen relatief eenvoudig deel kunnen nemen aan notificatieprocessen.

Gebruik bekende breed gebruikte standaarden voor notificeren

Realiseren van maximale ontkoppeling vereist vaak speciale maatregelen. Het is daarom al snel wenselijk om daarbij gebruik te maken van gespecialiseerde voorzieningen (bijv. om de rol van Intermediary in te vullen). Daarmee kan worden voorkomen dat bij Producers steeds meer (complex) maatwerk nodig is en zij zich kunnen beperken tot Producer-taken.

Realiseer ontkoppeling door gebruik van gespecialiseerde componenten in de rol van intermediair

4.5 NON-FUNCTIONAL ASPECTEN

In systems engineering and requirements engineering, a non-functional requirement (NFR) is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviours. They are contrasted with functional requirements that define specific behavior or functions. The plan for implementing functional requirements is detailed in the system design. The plan for implementing non-functional requirements is detailed in the system architecture, because they are usually architecturally significant requirements. – bron: Wikipedia

Niet-functionele eisen zijn kwaliteitseisen die gesteld worden aan een systeem, maar niet direct bijdragen aan het te behalen functionele doel. Net als voor andere geautomatiseerde processen geldt ook voor notificatieprocessen dat oplossingen bijv. betrouwbaar, veilig en effectief werken. Hoe dit het beste kan worden gegarandeerd is sterk contextafhankelijk. Deze paragraaf beperkt zich daarom tot een drietal aspecten waarbij voor ieder aspect geldt dat het zeer belangrijk is maar binnen de context van project Notificatieservices niet volledig was uit te werken. Binnen de CloudEvents-community is vanwege de diversiteit aan mogelijkheden om aan non-functionele eisen te voldoen besloten om dit niet te gaan standaardiseren.

4.5.1 BETROUWBAARHEID

De context bepaalt wat het gewenste of vereiste betrouwbaarheidsniveau is waarmee notificaties voor afnemers beschikbaar moeten komen. Hoewel er situaties denkbaar zijn waarin een afnemer het geen probleem vindt om af en toe een notificatie te missen zal er bij gegevensuitwisseling tussen overheidsorganisaties vaak een hoog betrouwbaarheidsniveau nodig zijn. Denk bijvoorbeeld aan notificaties over plaatsgevonden [levensgebeurtenissen](#) of transacties waar rechtskracht aan kan worden ontleend.

Realisatie van hoge betrouwbaarheid vereist speciale maatregelen zoals het gebruik van daarvoor geschikte standaarden (bijv. Digikoppeling) en voorzieningen (bijv. speciale 'event-oriented middleware'). Daarmee wordt het mogelijk om garanties te bieden dat berichten gegarandeerd worden bezorgd en goed wordt omgegaan met uitzonderings- of foutsituaties. Denk daarbij aan zaken zoals het tijdelijk bewaren van notificaties ('buffering'), het vaker opnieuw proberen af te leveren van notificaties ('retry') of het markeren van notificaties als 'niet af te leveren' (bijv. via 'deadletter-queues').



Als wordt gewerkt met stijlen, standaarden of voorzieningen die inherent geen hoge betrouwbaarheid kennen zullen zowel Producers als Consumers speciale maatregelen moeten treffen. Bijvoorbeeld door in applicaties controles in te bouwen of zich fouten hebben voorgedaan tijdens berichtuitwisseling of door als Producer notificaties van volgnummers te voorzien en van alle Consumers te verwachten dat ze bij ontvangst controleren op compleetheid en volgordeijkheid.

Naarmate het vereiste betrouwbaarheidsniveau toeneemt wordt dit (heel) snel (heel) complex. In zijn algemeenheid geldt dat het onwenselijk is om dit type functionaliteit binnen bedrijfsapplicaties op te gaan nemen.

Probeer maatwerkcode bij Producers en Consumers om betrouwbaarheid te realiseren te vermijden

In zijn algemeenheid geldt dat voor het vergroten van betrouwbaarheid van gegevensuitwisseling het beste gebruik kan worden gemaakt van standaarden en voorzieningen die functionaliteit bevatten voor hoge betrouwbaarheid. Bij transportprotocollen geldt bijv. dat sommige protocollen specifieke functionaliteit bevatten voor het omgaan met verstoringen terwijl zo'n functionaliteit bij andere protocollen volledig ontbreekt²⁴.

Gebruik standaarden en voorzieningen die functionaliteit bevatten om hoge betrouwbaarheid te garanderen

4.5.2 BEVEILIGING

Voor de overheid geldt de BIO als algemeen normenkader voor informatiebeveiliging. Om risico's in beeld te krijgen en te kunnen managen kan bijv. een toets worden gedaan om het juiste 'basisbeveiligingsniveau' te bepalen en te onderzoeken of beveiligingsniveau 2 voldoende is²⁵.

Beveiligingsniveaus bouwen op elkaar voort en bevatten een aantal controls, verplichte overheidsmaatregelen en een verantwoordings- en toezichtregime. Na het bepalen van het BBN moet worden bepaald welke controls moeten worden doorlopen en per control moet op basis van een risicoafweging worden bepaald welke aanvullende maatregelen nodig zijn om aan de beveiligingsdoelstellingen te voldoen. Met name wanneer er bij een notificatieproces vertrouwelijke en kritische informatie wordt uitgewisseld is het cruciaal om integraal te kijken naar beveiligingsmaatregelen bij zowel Producers, Intermediaries als Consumers.

Gebruik bij notificeren de BIO als normenkader voor informatiebeveiliging

Om te zorgen dat uitwisseling van gegevens bij notificeren veilig genoeg verloopt moet zo vroeg mogelijk in het ontwerpproces aandacht worden geschonken aan beveiligingsaspecten. Ook hier geldt dat met name de aanwezigheid van gevoelige gegevens het belangrijk maakt om na te denken over welke gegevens in welke vorm in notificaties opgenomen worden.

²⁴ In het hoofdstuk over 'protocollen' wordt beschreven welke protocollen meer en minder geschikt zijn om hoge betrouwbaarheid te realiseren.

²⁵ *Basisbeveiligingsniveau 1 (BBN-1)*: het niveau waaraan alle overheidssystemen minimaal moeten voldoen. Denk daarbij aan volgen van wet- en regelgeving, de AVG en algemene beheersmaatregelen. Op dit niveau gaat het over openbare en niet-gevoelige informatie. *Basisbeveiligingsniveau 2 (BBN-2)*: hierbij ligt de focus op bewuste bescherming van informatie. Op dit niveau wordt vertrouwelijke informatie verwerkt en kunnen incidenten leiden tot commotie. De BIO gebruikt BBN-2 als uitgangspunt. *Basisbeveiligingsniveau 3 (BBN 3)*: er vindt actieve bescherming van vertrouwelijke en kritische informatie plaats. Het betreft zeer gevoelige informatie waarbij verlies ervan zeer grote impact heeft. Aansluiting op een infrastructuur van BBN 3 is vereist om informatie te kunnen verwerken op deze infrastructuur.



Daarbij is onderscheid te maken in gegevens die als context ('meta') gegeven of als payload ('inhoudelijk') gegeven worden aangemerkt. Met name wanneer gebruik wordt gemaakt van gespecialiseerde Intermediaries mogen context-attributen niet-versleuteld zijn om te kunnen worden verwerkt (bijv. voor filtering en routing). Wanneer een gevoelig gegeven als context-attribuut nodig is (bijv. BSN) is het extra van belang om goede afspraken over verwerking te maken (bijv. in de vorm van een Verwerkersovereenkomst).

Maak afspraken over verwerking als gevoelige gegevens als onversleuteld context-attribuut worden gebruikt

Voor attributen binnen het payload-gedeelte van een bericht geldt dat ze zodanig versleuteld kunnen worden dat alleen Consumers (en dus geen Intermediaries) gegevens kunnen lezen. Iets dat aanvullende afspraken vereist tussen Producer en Consumers. Waar mogelijk biedt versleuteling van inhoudelijke gegevens binnen het payload-gedeelte van een bericht extra garantie dat gegevens niet door onbevoegden worden ingezien.

Versleutel binnen notificatieprocessen waar mogelijk het payload-gedeelte van berichten

Een andere reden om bij verwerking van berichten binnen notificatieprocessen geen gebruik te maken van payload-gegevens is dat daarmee (te veel) complexiteit wordt vermeden. Payload-gegevens zijn immers bestemd voor Consumers en kunnen op heel veel manieren worden gestructureerd en van betekenis worden voorzien. Bij veel gespecialiseerde producten die de rol van Intermediary kunnen vervullen wordt gebruik van payload-gegevens voor bijv. filtering en routing daarom niet ondersteund. Ook voor de CloudEvents standaard geldt dat payload-gegevens worden gezien als 'black box' die niet bruikbaar is voor zaken als routing en filtering.

Gebruik binnen notificatieprocessen payload-gegevens niet voor routing en filtering

Vroegtijdig meenemen van beveiligingsaspecten is nodig om bij het ontwerp van notificatieoplossingen de juiste keuzes te kunnen maken. In relatie tot beveiliging en vertrouwelijkheid van gegevens kan bijvoorbeeld worden besloten om informatiearm in plaats van informatierijk te gaan notificeren.

Neem beveiliging vroegtijdig mee bij het ontwerp van notificatieprocessen ('security by design')

Notificatieprocessen zijn te beschouwen als een bijzondere vorm van processen waarbij gegevens tussen applicaties worden uitgewisseld. Daarom zijn gangbare mechanismen voor beveiliging van gegevensuitwisseling ook toe te passen bij notificeren.

Er kan bijv. gebruik worden gemaakt van:

- Beveiligde verbindingen (bijv. via 1- of 2-zijdig TLS authenticatie).
- Speciale certificaten (bijv. PKI-overheid).
- Besloten netwerken (bijv. Diginetwerk).
- Standaarden die verschillende aspecten omvatten (bijv. Digikoppeling).

Gebruik bij notificeren gangbare beveiligingsmechanismen voor gegevensuitwisseling

4.5.3 PRIVACY

Wanneer bij notificeren op personen herleidbare gegevens betrokken zijn moet worden gezorgd dat voldaan wordt aan privacywetgeving (bijv. de AVG). Een voorbeeld hiervan is om te zorgen dat Intermediaries en Consumers geen gegevens kunnen inzien als daarvoor doelbinding ontbreekt. Dit kan betekenen dat er minder functionaliteit is te leveren dan wenselijk is (bijv. omdat bepaalde gegevens niet als contextgegevens



mogen worden gebruikt). Bij het ontwerpen van notificatieprocessen waarbij privacygevoelige gegevens zijn betrokken, wat al snel het geval is bij gegevens over personen, moet daarom vanaf de start aandacht zijn voor privacyaspecten. Zo kan bijv. vroegtijdig worden bepaald welk type oplossingen en technieken geschikt zijn.

Neem privacyaspecten vroegtijdig mee bij ontwerp van notificatieprocessen ('privacy by design')

Borgen van privacy bij notificeren vereist maatregelen op verschillende niveaus. Er moeten op bedrijfsmatig niveau bijv. afspraken over gebruik worden gemaakt tussen Producers, Intermediaries en Consumers. Op applicatieniveau moet worden gezorgd dat notificatieberichten niet meer informatie bevatten dan noodzakelijk en toegestaan is. Op technologieniveau geldt dat voorzieningen robuuste en veilige uitwisseling kunnen faciliteren.

Neem maatregelen op verschillende niveaus voor het borgen van privacy

Privacy borgen bij notificatieprocessen is een te uitgebreid onderwerp om hier te bespreken. Binnen enkele opgestelde handreikingen die zijn gemaakt tijdens project Notificatieservices wordt er beperkt aandacht aan besteed. Gelet op het belang ervan is het een van de onderwerpen waarbij het wenselijk is dat het als vervolg op project Notificatieservices apart wordt uitgewerkt.



5 PATRONEN EN STANDAARDEN

5.1 INLEIDING

Dit hoofdstuk beschrijft de betekenis en rol van patronen en standaarden voor het ontwerpen van goede notificatieoplossingen.

Onder een '(ontwerp)patroon' verstaan we:

A design pattern is the re-usable form of a solution to a design problem – bron: Wikipedia

Onder een 'standaard' verstaan we:

Een document dat een set van regels bevat die beschrijven hoe mensen materialen, producten, diensten, technologieën, taken, processen en systemen dienen te ontwikkelen en beheren' (bron: [NORA](#))

5.2 PATRONEN

Door bij het ontwerpen van notificatieoplossingen gebruik te maken van bewezen patronen en standaarden wordt gebruik gemaakt van opgedane kennis en ervaring en wordt realiseren van interoperabiliteit binnen de overheid beter mogelijk. Omdat de ervaring met gebeurtenisgedreven werken binnen de Nederlandse overheid nog vrij beperkt is, is het des te belangrijk om maximaal gebruik te maken van wat er wereldwijd aan patronen en standaarden beschikbaar is.

Er is een grote verscheidenheid aan patronen en standaarden. Patronen kunnen bijv. van elkaar verschillen qua:

- **Doel:** patronen zijn voor verschillende doeleinden bruikbaar. We focussen hier op patronen die behulpzaam zijn voor het ontwerpen van oplossingen voor event-driven samenwerken in het algemeen en voor notificeren in het bijzonder.
- **Aard:** het ene type patroon beschrijft hoe interactie tussen applicaties plaats kan vinden terwijl een ander patroon een concrete implementatievorm beschrijft.
- **Abstractieniveau:** een patroon beschrijft uitwisseling op logisch niveau terwijl een andere patroon dit op fysiek niveau doet.
- **Modulariteit:** een patroon kan meerdere patronen combineren tot een nieuw patroon of een patroon kan juist een nadere uitwerking van een bestaand patroon zijn.

In de praktijk zijn verschillende oplossingen mogelijk om een probleem op te lossen. Kennis en ervaring daarmee in de praktijk heeft geleid tot bewezen bruikbare '(ontwerp)patronen' die als startpunt ('sjabloon') zijn te gebruiken om binnen vergelijkbare situaties goede oplossingen te ontwerpen.

Binnen de context van event-driven werken en notificeren interacteren Producers, Intermediaries en Consumers. De manieren waarop dat gebeurt zijn (ook) herkenbaar binnen andere contexten. Reden waarom bepaalde algemene basispatronen (ook) bruikbaar zijn binnen de context van event-driven werken. Enkele bekende basispatronen zijn:

- **Observer:** *Define a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically*



- **Mediator:** Define an object that encapsulates how a set of objects interact. Mediator promotes loose coupling by keeping objects from referring to each other explicitly, and it lets you vary their interaction independently.
- **Chain of Responsibility:** Avoid coupling the sender of a request to its receiver by giving more than one object a chance to handle the request. Chain the receiving objects and pass the request along the chain until an object handles it.
- **Command:** Encapsulate a request as an object, thereby letting you parameterize clients with different requests, queue or log requests, and support undoable operations.

bron: *Design Patterns: Elements of Reusable Object-Oriented Software* – Gamma, Helm, Johnson, Vlissides, 1994

Voor de specifieke vorm van interactie die plaatsvindt bij notificeren is het Publish-Subscribe patroon verreweg het meest bekende en toegepaste patroon. Het patroon is te zien als een uit deze basispatronen opgebouwd interactiepatroon voor notificeren²⁶.

Het patroon beschrijft in de kern hoe:

- Aanbieders/Producers gegevens publiceren over plaatsgevonden gebeurtenissen ('Publish') en
- Afnemers/Consumers zich abonneren ('Subscribe') om na het publiceren van gegevens over bepaalde gebeurtenissen notificaties te ontvangen.

We zien het Publish-Subscribe patroon op dit moment (ook) voor de Nederlandse overheid als meest bruikbare patroon voor ontwerp en standaardisatie van notificatieoplossingen.

De term 'Publish-Subscribe patroon' wordt in de praktijk overigens verschillend gebruikt. Soms wordt bijv. gesteld dat om te kunnen spreken over een Publish-Subscribe patroon aan een bepaalde voorwaarde moet zijn voldaan. Bijv. dat de intermediair-rol door een aparte applicatie moet zijn ingevuld, dat gegevensuitwisseling asynchroon moet verlopen of dat notificaties altijd actief naar Consumers moeten worden gebracht ('push-mechanisme'). We zien dit soort aanvullende voorwaarden niet als voorwaarde om te kunnen spreken over 'gebruik van het Publish-Subscribe patroon' maar als bewezen en in de praktijk vaak toegepaste implementatiekeuzes. Gelet op het uitgangspunt om als Nederlandse overheid maximaal aan te sluiten bij wat wereldwijd gebruikelijk is, is het wenselijk om hier bij het ontwerpen van oplossingen bij aan te sluiten.

Het volgende hoofdstuk beschrijft hoe het Publish-Subscribe patroon in algemene zijn is toe te passen voor notificeren. Het hoofdstuk daarna, 'Implementatie-patronen' beschrijft hoe het Publish-Subscribe patroon in de praktijk op een aantal verschillende manieren is toe te passen.

5.3 STANDAARDEN

Net zoals op andere gebieden geldt ook bij event-driven werken en notificeren dat standaardisatie productontwikkeling vereenvoudigt en versnelt en bijdraagt aan compatibiliteit en interoperabiliteit.

²⁶Als een Publisher zelf notificaties verstrekt aan een Consumer is sprake van het 'observer-patroon'. Wordt dit gecombineerd met het 'mediator-patroon' om ontkoppelde interactie te ondersteunen dan is sprake van het Publish-Subscribe patroon. "Mediator (305) and Observer (326) are competing patterns. The difference between them is that Observer distributes communication by introducing Observer and Subject objects, whereas a Mediator object encapsulates the communication between other objects." – bron: *Design Patterns – Elements of Reusable Object-Oriented Software* – Gamma, Helm, Johnson, Vlissides, 1994

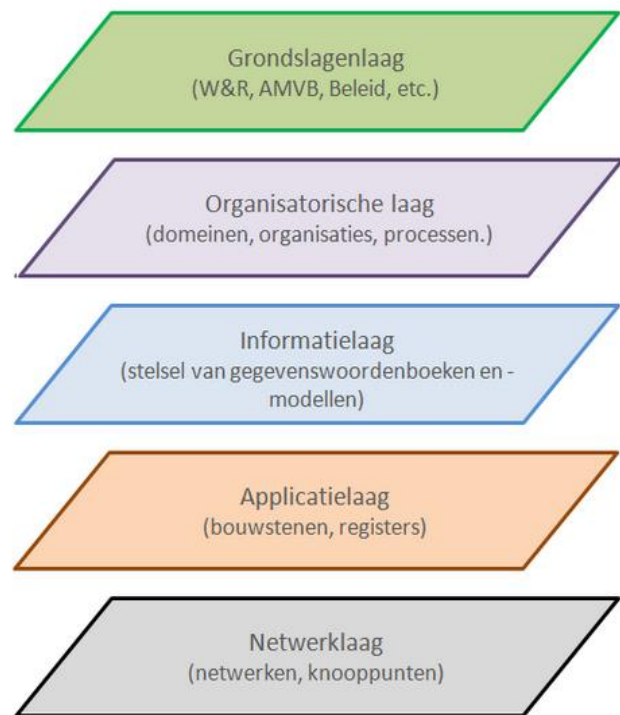
Als computersystemen gegevens uitwisselen, dan moeten zij de afspraken en vaste processen hanteren. Doen ze dat niet, dan kunnen er fouten ontstaan, kan het systeem misbruikt worden of wordt verkeerde data overgenomen. – Bron: [Forum Standaardisatie](#)

Voor event-driven werken is standaardisatie op verschillende niveaus nodig zodat betrokken organisaties en applicaties effectief en efficiënt kunnen samenwerken. Binnen alle vijf de lagen van het hiernaast getoonde NORA-vijflaagsmodel zijn afspraken en standaarden nodig om als overheid goed event-driven te kunnen werken en elkaar van bruikbare notificaties te voorzien.

Project Notificatieservices heeft enkele handreikingen opgeleverd waarin extra aandacht is voor wat in het model aangeduid wordt als 'grondslagen' en 'organisatie' (bijv. de handreiking [Randvoorwaardelijke aspecten](#)). Binnen dit architectuurdocument richten we ons met name op de informatie-, applicatie- en (beperkt) netwerklagen.

In volgende hoofdstukken wordt voor een aantal soorten standaarden beschreven hoe ze bruikbaar zijn bij event-driven werken en notificeren. Het gaat daarbij om standaarden voor:

- Gebeurtenisberichten (bijv. CloudEvents).
- Gegevensformaten (bijv. JSON).
- Transport protocollen (bijv. HTTP).
- Interactie (bijv. Webhook).



Figuur 3: NORA vijflaagsmodel



6 HET PUBLISH-SUBSCRIBE PATROON

6.1 ALGEMEEN

Bij notificatieprocessen is er sprake van een specifieke vorm van interactie tussen Publishers, Intermediaries en Consumers²⁷. We zien het Publish-Subscribe patroon als het meest bruikbare patroon voor het ontwerpen van notificatieoplossingen binnen de overheid. O.a. omdat het:

- Een bekend, bewezen en vaak toegepast patroon is voor event-driven werken en notificeren.
- Geschikt is om gegevens over plaatsgevonden gebeurtenissen te delen tussen ontkoppelde Publishers en een onbeperkt aantal (evt. bij Publishers onbekende) Consumers.
- Daarbij verschillende gegevensformaten, transportprotocollen en implementatiepatronen zijn te gebruiken.
- Brede technische ondersteuning kent in de vorm van standaarden (bijv. AsyncAPI), hulpmiddelen voor systeemontwikkeling (bijv. System Development Kits) en gespecialiseerde applicaties ('middleware') die de rol van Intermediary kunnen vervullen (bijv. event broker applicaties).

Gebruik het Publish-Subscribe patroon bij het ontwerpen van notificatieoplossingen

Net als voor andere patronen geldt ook voor het Publish-Subscribe patroon dat het geen recept is dat klakkeloos gevolgd kan worden. Bij gebruik van het patroon moeten altijd specifieke, bij de context passende, ontwerpkeuzes worden gemaakt²⁸.

In lijn met het uitgangspunt om zoveel mogelijk aan te sluiten bij wat wereldwijd gebruikelijk is nemen we de Wikipedia-omschrijving van het 'Publish-Subscribe pattern' als uitgangspunt om af te spreken wat we er binnen de overheid onder verstaan.

In software architecture, publish-subscribe is a messaging pattern where senders of messages, called Publishers, do not program the messages to be sent directly to specific receivers, called subscribers, but instead categorize published messages into classes without knowledge of which subscribers, if any, there may be. Similarly, subscribers express interest in one or more classes and only receive messages that are of interest, without knowledge of which Publishers, if any, there are. – bron: [Wikipedia](#)

Opmerkingen:

- Waar wordt gesproken over 'Subscribers' mag hier worden gelezen 'Consumers'.
- Binnen de overheidscontext geldt dat Publishers en Subscribers vaak op de hoogte zullen zijn van elkaars betrokkenheid omdat er, bijv. bij uitwisseling van vertrouwelijke gegevens, voorafgaand aan notificatie een overeenkomst nodig is.

Zoals in het hoofdstuk Architectuurstijlen is beschreven verdient messaging als integratiestijl de voorkeur voor notificeren. Ook als gebruik wordt gemaakt van het Publish-Subscribe patroon. In de diverse praktijk van de Nederlandse overheid zullen zich ook situaties voordoen waarin messaging niet bruikbaar is. Bijv. als

²⁷ Situaties waarin aanbieders op de hoogte stellen van plaatsgevonden gebeurtenissen via 1 op 1 koppelingen vallen hier buiten.

²⁸ Het hoofdstuk 'Implementatiepatronen' beschrijft een aantal patronen die voortbouwen op het Publish-Subscribe patroon maar verschillende kenmerkende implementatiekeuzes kennen.



organisaties niet in staat zijn om via messaging betrouwbaar gegevens uit te wisselen maar dit wel kan via bijv. file-transfer.

Publish-Subscribe: Gebruik bij voorkeur messaging als integratiestijl voor notificeren

Een belangrijk onderdeel van de Wikipedia-definitie is dat Publishers en Consumers maximaal van elkaar worden ontkoppeld. Publishers kunnen zich dan focussen op het publiceren van relevante beschreven gebeurtenisinformatie, Consumers op het specificeren van hun interesses en het zorgen dat notificaties aan hen zijn te verstrekken²⁹. Voor een gedistribueerd en divers landschap als de Nederlandse overheid kan niet genoeg benadrukt worden hoe belangrijk ontkoppeling tussen Producers en Consumers is om te kunnen komen tot goede notificatieoplossingen. Eerder is in '[paragraaf 4.4 Ontkoppeling](#)' beschreven op welke aspecten ontkoppeling is te realiseren.

Publish-Subscribe: Ontkoppel Producers en Consumers maximaal

Waar het gaat om *organisaties* is binnen de overheid vaak een bepaalde mate van koppeling nodig omdat aanbiedende en afnemende organisatie afspraken moeten maken over het notificatieproces. Bijvoorbeeld over de mate waarin en manier waarop vertrouwelijke gegevens (mogen) worden verstrekt of over de garanties waarmee verstrekking plaats moet vinden. Ook dan blijft gelden dat naar ontkoppeling moet worden gestreefd. Bijv. om te zorgen dat organisaties niet wederzijds afhankelijk worden, het mogelijk te maken dat met andere partners of applicaties is te werken en om te zorgen dat organisaties zich tot hun kerntaken kunnen beperken. Dit gebeurt bijv. door de verantwoordelijkheid voor correct gebruik van notificaties volledig bij afnemende partijen te beleggen (waar bijv. via audits kan worden gecontroleerd of gebruik verantwoord plaatsvindt). Een andere maatregel is het beleggen van specialistische intermediair-taken bij een derde partij waardoor een aanbiedende organisatie (deels) ontkoppeld wordt van afnemende organisaties. Een binnen de overheid bekend voorbeeld daarvan is Logius die via de voorziening Digilevering bemiddelt tussen aanbieders van basisregistratiegegevens en afnemende organisaties die notificaties willen ontvangen over bepaalde wijzigingen binnen basisgegevens.

Publish-Subscribe: Ontkoppel aanbiedende en afnemende organisaties maximaal van elkaar

6.2 FILTEREN EN ROUTEREN

Binnen het domein van een publisher vinden heel veel soorten gebeurtenissen plaats. Er moet worden gezorgd dat die veelheid aan gebeurtenissen uiteindelijk leidt tot verstrekking van goed bruikbare notificaties aan Consumers. Acties daarvoor zijn bijv. om in beeld te brengen van welke soorten gebeurtenissen er sprake is binnen een domein, voor welke daarvan notificaties gewenst zijn en op welke manier die het beste verstrekt kunnen worden. Dit is niet eenvoudig. Reden om hiervoor een aparte handreiking, '[Definiëren van gebeurtenis-types](#)', te maken die dit proces meer in detail beschrijft.

Notificatieprocessen moet zodanig zijn ontworpen dat aan iedere Consumer de notificaties worden verstrekt die hij mag en wil ontvangen. Voor het selecteren van de juiste notificaties voor Consumers binnen het totaal aan notificaties gebruiken we de term 'filteren'.

In the Publish-Subscribe model, subscribers typically receive only a subset of the total messages published. The process of selecting messages for reception and processing is called filtering. There are two common forms of filtering: topic-based and content-based.

²⁹ Zijn bij notificeren vertrouwelijke gegevens betrokken dan zullen betrokken partijen ook afspraken over verstrekking moeten maken.



- In a **topic-based** system, messages are published to "topics" or named logical channels. Subscribers in a topic-based system will receive all messages published to the topics to which they subscribe. The publisher is responsible for defining the topics to which subscribers can subscribe.
- In a **content-based** system, messages are only delivered to a subscriber if the attributes or content of those messages matches constraints defined by the subscriber. The subscriber is responsible for classifying the messages. – bron: [Wikipedia](#)

Bij topic-based filtering worden berichten door de publisher als het ware in verschillend gelabelde postvakken geplaatst. Het creëren van topics door Producers kan op basis van verschillende criteria gebeuren. Er kunnen bijv. topics worden aangemaakt per:

- Objecttype (bijv. 'vergunningen', 'subsidies', zaken).
- Gebeurtenistype (bijv. 'vergunningAangevraagd', 'vergunningVerleend').
- Gegevensbron (bijv. 'zakenRegistratie', 'documentenRegistratie').
- Beoogde Consumer (bijv. 'gemeenteX', 'waterschapY').
- etc.

Het is sterk contextafhankelijk welk type topics het meest wenselijk zijn.

Publish-Subscribe: Gebruik topic-based filtering voor het op maat verstrekken van notificaties

Topics kunnen gelaagd worden opgebouwd. Vaak door ze hiërarchisch te structureren (bijv. 'gemeentex.vergunningen.parkervergunning.verleend'). Hierdoor worden de mogelijkheden voor routing en filtering vergroot en kan meer op maat worden genotificeerd. Er is hierbij ook een relatie met de binnen een event aanwezige metadata. Een binnen CloudEvents gedefinieerd attribuut als (event)'type' is (ook) bedoeld om te beschrijven om wat voor event het gaat en kan dus vergelijkbaar gelaagd worden opgebouwd. Het is afhankelijk van meerdere factoren hoe topics het beste kunnen worden gedefinieerd. De (ook toekomstige) filterbehoeften van Consumers spelen daarbij een belangrijke rol maar ook de mogelijkheden van Intermediaries (bijv. als voor filtering gebruik wordt gemaakt van serverless functies die alleen kunnen werken met metadata in een bericht). Het is dus van belang om vroegtijdig na te denken over het definiëren van topics om (ook) in toekomstige behoeften van Intermediaries en Consumers te voorzien.

Publish-Subscribe: denk vroegtijdig en goed na over het definiëren van topics

Gebruik van topics wordt in de praktijk vaak toegepast. Dikwijls met behulp van specialistische applicaties in de rol van intermediair voor zowel routeren als filteren. In lijn met het 'separation of concerns' principe geldt dat het gelet op het specialistisch karakter het al snel wenselijk wordt om voor filteren en routeren van events gebruik te maken van bewezen middleware voorzieningen. De technische ontkoppeling die daarmee wordt gerealiseerd kan eventueel worden uitgebreid met organisatorische ontkoppeling door technisch, en eventueel ook functioneel beheer, van dit type voorziening te beleggen bij een externe partij.

Publish-Subscribe: Gebruik bij voorkeur bewezen middleware voor filter- en routeer-functionaliteit

De toevoeging 'bij voorkeur' geeft aan dat er situaties kunnen zijn waarin het exclusief beleggen van filtering bij een Intermediary niet wenselijk of toegestaan is. Bij inzet van generieke middleware moet bijv. worden gewaakt voor het toevoegen van domein-specifieke bedrijfslogica om te kunnen bepalen welke Consumers welke notificaties moeten krijgen. Het verdient dan de voorkeur dat een Producer dit type filtering uitvoert. Voorbeelden waarbij dit speelt zijn:



- Notificaties waarin vertrouwelijke persoonsgegevens zijn betrokken waarvoor bij Consumers doelbinding moet bestaan om ze te mogen ontvangen. Bepalen aan welke Consumer wat mag en moet worden verstrekt kan zo complex worden dat de Producer de aangewezen partij is om de eerste filtering te doen. Daarna kunnen notificaties worden gepubliceerd via topics zoals hierboven beschreven. Waarbij in extreme gevallen op een bepaald topic zich slechts enkele, of zelfs 1, Consumer(s) mogen abonneren (en eventueel zelf nog filters toepassen).
- Notificaties waarbij hele specialistische gegevens moeten worden gefilterd. Bijvoorbeeld als het gaat om geografische gegevens waarbij voor afgebakende gebieden selecties moeten worden gemaakt (bijv. van objecten die binnen bepaalde gebiedsgrenzen liggen). Dit type filtering is vaak maar beperkt mogelijk bij gebruik van gangbare middleware-voorzieningen en kan daarom het beste bij in dit type informatie gespecialiseerde Producers plaatsvinden.

Publish-Subscribe: Laat domein-specifieke filtering uitvoeren door Producers

Net zoals voor Producers geldt, geldt ook voor Consumers dat er gevallen zijn waarin het wenselijk is dat filtering door hen zelf wordt uitgevoerd. Soms is het bijv. niet mogelijk of wenselijk om alle wensen met betrekking tot ontvangst van notificaties via op te geven filtercriteria te realiseren. In die situaties kan een Consumer zelf ontvangen notificaties gaan filteren. Bijv. om de bij verschillende soorten ontvangen notificaties de juiste interne processen te starten of om bepaalde notificaties, in- of extern, te verstrekken aan andere Consumers (waarmee de rol van Consumer verandert in de rol van Producer). Een voordeel van filtering door Consumers is dat een Intermediary niet hoeft te voorzien in zeer specifieke, vaak slechts voor een enkele Consumer, benodigde filterfunctionaliteit.

Publish-Subscribe: Laat consumer-specifieke filtering uitvoeren door Consumers

Bij toepassing van het Publish-Subscribe patroon wordt vaak verondersteld dat filtering en routing exclusief plaatsvindt door een Intermediary. Zoals hierboven beschreven geldt binnen de overheidscontext dat het belangrijk is hier niet zondermeer van uit te gaan. Het is wenselijk om vroegtijdig na te denken over welke typen filtering nodig zijn en waar die het beste plaats kunnen vinden. Doel is om zowel Producers, Intermediaries als Consumers de taken te laten uitvoeren die bij hen horen ('separation of concerns').

Publish-Subscribe: Bepaal vroegtijdig welke soorten filtering nodig zijn en waar die het beste plaats kunnen vinden

Binnen de Wikipedia-definitie wordt 'topic-based filtering' onderscheiden van 'content-based filtering'. In het laatste geval wordt voor filtering en routing binnen gebeurtenisberichten aanwezige attribuutwaarden vergeleken met voor iedere Consumer aanwezige selectiecriteria. Het is cruciaal hierbij onderscheid te maken in:

- 'context-attributen': attributen met meta-informatie die specifiek bedoeld zijn voor zaken als filtering en routing zodat de juiste Consumers de juiste notificaties ontvangen, en
- 'payload-attributen': attributen die de feitelijk plaatsgevonden gebeurtenis beschrijven zodat Consumers ze correct kunnen interpreteren en gebruiken.

Context-attributen zijn bruikbaar om events te filteren en routeren om de juiste notificaties aan de juiste Consumers te verstrekken. Het op de CloudEvents-specificatie gebaseerde NL GOV profile for CloudEvents schrijft voor dat uitsluitend gebruik wordt gemaakt van context-attributen voor filtering en routing. Niet van payload-gegevens. Hierdoor wordt vergaande standaardisatie en verwerking door een veelheid aan Intermediaries mogelijk.

Publish-Subscribe: Gebruik context-attributen voor content-filtering



'Payload-attributen' zijn specifiek bedoeld voor Consumers om informatie te krijgen over een plaatsgevonden gebeurtenis. Vanwege complexiteit van gebruik daarvan voor filteren en routeren is het onwenselijk ze daarvoor te gebruiken. In bepaalde situaties is het ook niet toegestaan (bijv. als het vertrouwelijke gegevens betreft) of is het onmogelijk (bijv. als inhoud versleuteld is en niet kan worden gelezen). Payload-gegevens worden door een Intermediary bij voorkeur beschouwd als een 'black box met gegevens' die moet worden overgebracht van Producer naar Consumers.

Publish-Subscribe: Gebruik geen payload-attributen voor content-based filtering

Het Publish-Subscribe patroon wordt in de praktijk zowel gebruikt in situaties waarin geen autorisatie van afnemers nodig is ('alle afnemers mogen alles ontvangen') als in situaties waarin autorisatie wel van belang is ('afnemers mogen alleen krijgen waar ze recht op hebben'). Bij notificeren binnen de overheid zijn vaak vertrouwelijke, bijv. op personen herleidbare, gegevens betrokken. Producers kunnen niet altijd beoordelen of Consumers recht hebben op bepaalde notificaties. Afnemers geven in de praktijk binnen bepaalde gebeurtenistypes vaak zelf aan voor welke objecten (bijv. personen) zij notificaties mogen en willen ontvangen. Bij deze vorm van 'objectfiltering' ligt de verantwoordelijkheid voor het in overeenstemming met doelbinding filteren en verstrekken van notificaties expliciet bij de afnemer. Controle hierop kan bijv. via periodieke audits worden gedaan.

Publish-Subscribe: Laat Consumers waar nodig zelf objecten filteren om in overeenstemming met doelbinding notificaties te kunnen verstrekken

Naast het wegfilteren van bepaalde notificaties is 'attribuut-filtering' een andere manier om te zorgen dat Consumers niet meer gegevens ontvangen dan waar ze recht op hebben. Bij het samenstellen van notificaties uit beschikbare attributen wordt dan rekening wordt gehouden met voor een specifieke afnemer(s) geldende autorisatieregels. Naarmate het aantal verschillende soorten Consumers toeneemt wordt het steeds lastiger om op deze manier te filteren.

Publish-Subscribe: Filter waar nodig attributen om in overeenstemming met doelbinding notificaties te kunnen verstrekken

Los van de precieze taakverdeling tussen de organisatie in de rol van Producer en die in de rol van intermediair is de Producer eindverantwoordelijk voor notificatieverstrekking. Voor attribuutfiltering waarbij domeinkennis nodig is zal die vaak door de Producer worden uitgevoerd. Bijv. omdat alleen de Producer over de benodigde domeinkennis beschikt en het zo helder is wie waarvoor verantwoordelijk is. Een Producer kan naar aanleiding van één gebeurtenis bijv. verschillende soorten notificaties samenstellen en via verschillende topics publiceren. De functie van Intermediary kan dan beperkt blijven tot meer technische functies om notificaties volgens afnemerwensen te verstrekken (bijv. om door Consumers ingestelde filters toe te passen of notificaties in een aangepast tempo te verstrekken ('throttling')).

Publish-Subscribe: Beleg attribuutfiltering op basis van autorisatie bij Producers

Opmerkingen:

- Attribuutfiltering is hiervoor met name beschreven als middel om bij notificeren te kunnen voldoen aan doelbindingseisen. Uiteraard kan attribuutfiltering ook worden toegepast om 'op maat' notificaties aan afnemers te verstrekken. Afnemers kunnen dan zelf specificeren welke attributen ze binnen notificaties opgenomen willen zien.
- Waar we hier spreken over 'attributen' geldt dat het hierbij niet uitsluitend over enkelvoudige attributen hoeft te gaan maar er ook sprake kan zijn van geneste attributen. Naarmate de structuur



van een attribuut complexer wordt, wordt het ook lastiger om op basis van autorisatieregels correcte filtering toe te passen³⁰.

6.3 ABONNEREN

Subscribers may register for specific messages at build time, initialization time or runtime. – bron: [Wikipedia](#)

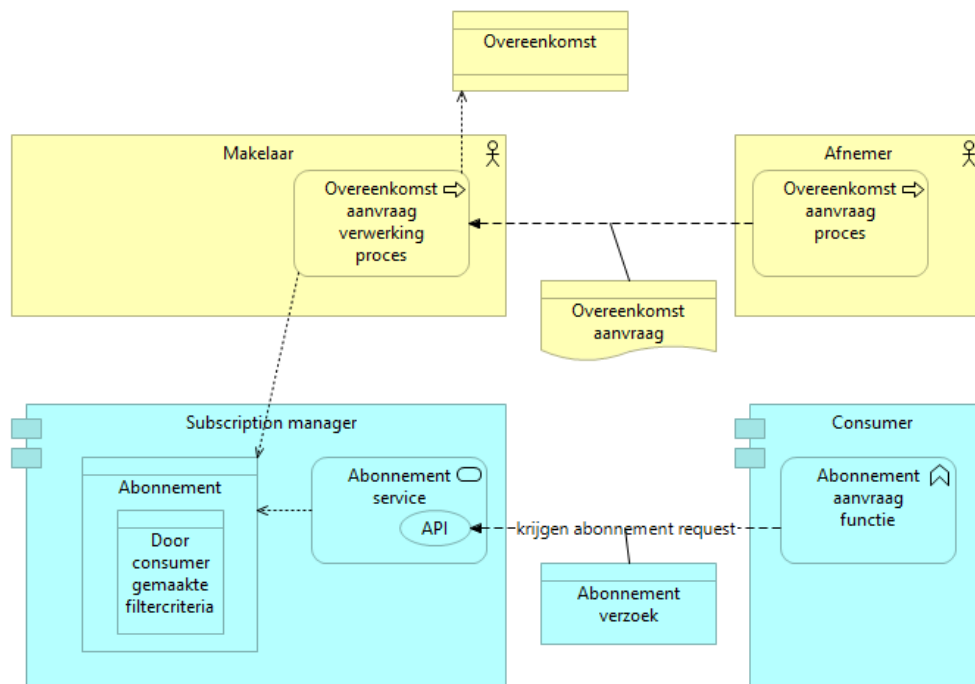
Consumers sluiten een abonnement af om notificaties te ontvangen wanneer zich bepaalde soorten gebeurtenissen hebben voorgedaan. Een abonnement kan op verschillende momenten worden afgesloten. Binnen de overheidscontext geldt vaak dat er eerst aan een aantal formele voorwaarden moet worden voldaan voordat een abonnement is af te sluiten. Bijv. omdat er eerst een [gegevensleveringsovereenkomst](#) moet zijn afgesloten. Het formeel bekrachtigen van een aangevraagd abonnement vindt dan vaak handmatig plaats.

Er zijn echter ook situaties waarin een abonneerproces, of delen daarvan, geautomatiseerd kan verlopen. Bijv. als een aanbieder openbare gegevens beschikbaar stelt en iedereen die dat wil zich daarop kan abonneren of als een Consumer rechten zijn toegekend om zelf abonnement gegevens te kunnen wijzigen. Voor het afsluiten, tussentijds wijzigen en beëindigen van een abonnement zijn dan services aanwezig die door Consumers zijn aan te roepen. Consumers kunnen dan bijv. opgeven voor welk type gebeurtenissen zij notificaties willen ontvangen, welke filtercriteria daarbij gelden en waarheen notificaties moeten worden gestuurd. In een steeds dynamischer wereld is het wenselijk om abonneerprocessen zoveel mogelijk geautomatiseerd te laten verlopen.

Publish-Subscribe: automatiseer abonneerprocessen voor het kunnen ontvangen van notificaties door Consumers zoveel mogelijk

Tijdens het project Notificatieservices is herhaaldelijk de behoefte uitgesproken aan een standaard voor geautomatiseerd abonneren. Op basis van een aantal use cases is geconcludeerd dat standaardiseren van (geautomatiseerd) abonneren een complex vraagstuk is. Zeker binnen een overheidscontext waar vaak vertrouwelijke of privacygevoelige gegevens worden uitgewisseld moet zowel op bedrijfs- als applicatieniveau zijn gegarandeerd dat aan alle benodigde voorwaarden is voldaan (bijv. om rechtmatig handelen te kunnen verantwoorden).

³⁰ Binnen de CloudEvents standaard is om deze reden besloten om binnen context-attributen geen nesting toe te staan. Bij attribuut-filtering door een producer kan zich, afhankelijk van de onderliggende informatievoorziening, die situatie wel voordoen.



Standaardiseren van abonneren vereist daarom apart aandacht en tijd en zou in een vervolgproject verder uitgewerkt kunnen worden. Een nuttige bron van informatie daarvoor is de specificatie voor geautomatiseerd abonneren die binnen de CloudEvents-community wordt ontwikkeld: de [CloudEvents Subscriptions API](#). Bij het ontwerpen van oplossingen kan worden gezien of onderdelen van deze concept-standaard bruikbaar zijn.

Publish-Subscribe: raadpleeg de CloudEvents Subscriptions API voor bij het ontwerpen van voorzieningen voor abonneren

Binnen het project Notificatieservices is in samenwerking met VNG Realisatie een beproeving gedaan waar geautomatiseerd abonneren onderdeel van uitmaakte. Een bestaande API-standaard is daarbij aangepast in lijn met de concept CloudEvents abonneerstandaard. Op Github is documentatie beschikbaar over:

- De ontwikkelde nieuwe (concept) [API-specificatie](#).
- Een [presentatie](#) waarin een aantal uitdagingen worden beschreven m.b.t. abonneren en daarin opgeven van filtercriteria.

6.4 BETROUWBAARHEID EN SCHAAALBAARHEID

Bij gegevensuitwisseling in een gedistribueerd gegevenslandschap kunnen om tal van redenen fouten optreden. Het is niet eenvoudig om robuuste en betrouwbare notificatieoplossingen te maken. Zoals eerder beschreven is het daarom al snel wenselijk voor de rol van Intermediary gebruik te maken van daarvoor ontworpen software. Daarmee kunnen dan bijv. berichten tijdelijk worden bewaard ('buffering') en kunnen optredende fouten goed worden afgehandeld. Publisher en Consumers worden daardoor van elkaar ontkoppeld en kunnen zich beide richten op hun kernfunctionaliteit.

Publish-Subscribe: gebruik gespecialiseerde applicaties om de rol van intermediair in te vullen

Bij gebruik van het Publish-Subscribe patroon kunnen verschillende implementatiekeuzes worden gemaakt. Een belangrijk aandachtspunt daarbij is om Producer en Consumers zoveel mogelijk van elkaar te



ontkoppelen. Bijv. op de aspecten: tijd, plaats, tempo, kardinaliteit, technologie, protocol en formaat (zie de paragraaf [Ontkoppeling](#)). Zeker als het aantal Consumers toeneemt is het belangrijk dat de ontkoppeling maximaal is.

Publish-Subscribe: maak implementatiekeuzes die Producer en Consumers maximaal ontkoppelen

Schaalbaarheid: Publish-Subscribe provides the opportunity for better scalability than traditional client-server, through parallel operation, message caching, tree-based or network-based routing, etc. – bron: [Wikipedia](#)

Het Publish-Subscribe patroon is heel geschikt om schaalbaarheid te kunnen realiseren zodat er geen problemen optreden als het aantal berichten of het aantal Consumers (sterk) toeneemt. Naast al langer bekende middleware-applicaties leveren tegenwoordig ook alle grote Cloud platforms services die bij gebruik van het Publish-Subscribe patroon de rol van intermediair vervullen. Daarbij is bijv. automatisch op- en afschalen mogelijk en kunnen notificatieprocessen, ook bij zeer grote aantallen afnemers en berichten snel en robuust worden uitgevoerd.

Publish-Subscribe: gebruik applicaties in de rol van intermediair die goed schaalbaar zijn

6.5 UITDAGINGEN

The most serious problems with Publish-Subscribe systems are a side-effect of their main advantage: the decoupling of publisher from subscriber. – bron: [Wikipedia](#)

Ontkoppeling en werken met een gedistribueerd applicatielandschap is om veel redenen wenselijk maar kent ook de nodige uitdagingen. Een daarvan is de noodzaak om heldere afspraken te maken en standaarden toe te passen om te zorgen dat betrokken organisaties en applicaties goed kunnen samenwerken. De kunst daarbij is om te standaardiseren op die aspecten die duidelijke meerwaarde in de praktijk opleveren zonder dat standaardiseren een doel op zich wordt. Binnen project Notificatieservices is er om die reden voor gekozen om als eerste te streven naar standaardisatie van het berichtformaat waarmee gegevens over plaatsgevonden gebeurtenissen worden uitgewisseld. Aansluitend bij de uitgangspunten van CloudEvents is dit beperkt tot een minimale set aan generieke kenmerken en worden domein-specifieke keuzes bewust overgelaten aan domeinexperts.

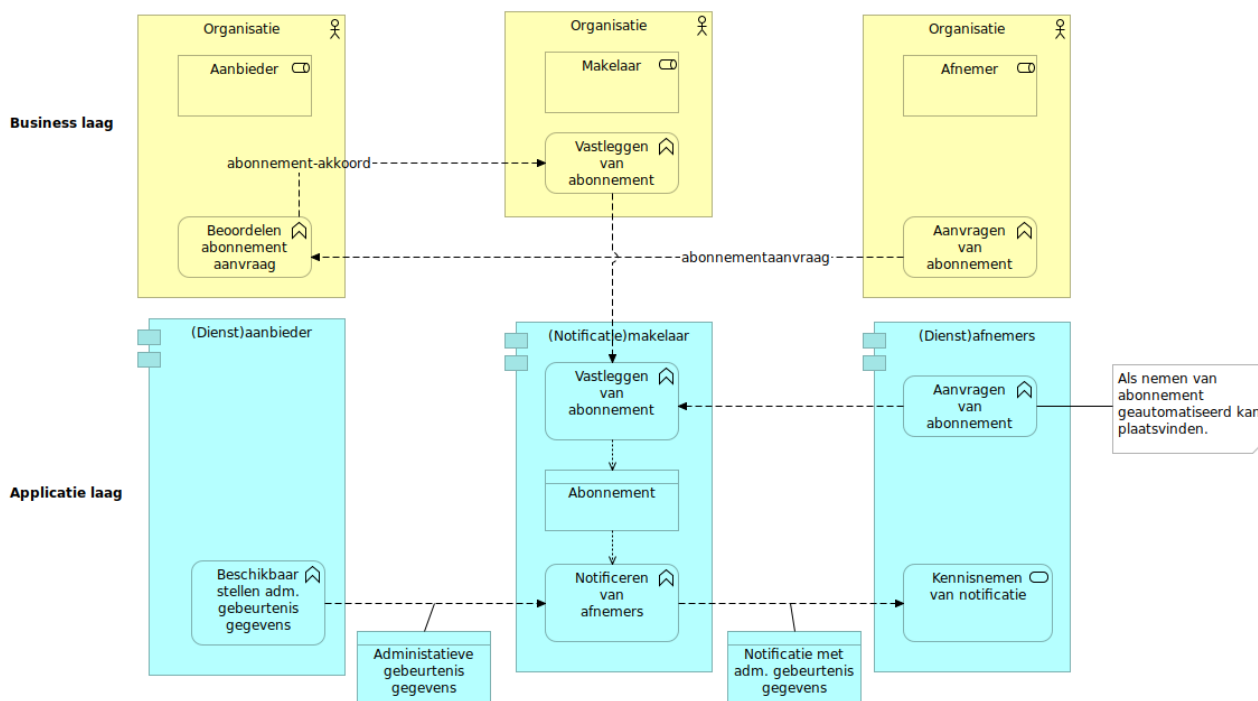
Publish-Subscribe: standaardiseer alleen wat meerwaarde heeft en laat domein specifieke standaardisatie over aan domeinen

Een tweede uitdaging is dat er bij gegevensuitwisseling tussen verspreid aanwezige applicaties fouten en verstoringen zullen optreden: applicaties zijn tijdelijk niet beschikbaar, er treden netwerkstoringen op, berichten raken verminkt of kwijt, berichten komen in de verkeerde volgorde binnen, applicaties raken overbelast, etc.. Zowel bij Producers, Intermediaries als Consumers zijn maatregelen nodig om daar goed mee om te kunnen gaan. Naarmate het vereiste betrouwbaarheidsniveau toeneemt wordt gebruik van geschikte technologie daarbij steeds belangrijker. Reden om bij de invulling van de cruciale rol van intermediair gebruik te maken van applicaties naast de functionele kwaliteiten (bijv. routing en filtering) ook beschikken over non-functional kwaliteiten om bijv. optredende fouten en storingen goed af te kunnen handelen.

Publish-Subscribe: gebruik Intermediaries die fouten en storingen goed kunnen afhandelen

6.6 GEBRUIK BINNEN DE OVERHEID

Zoals eerder beschreven is het Publish-Subscribe patroon (ook) heel geschikt om binnen de Nederlandse overheid toe te werken naar meer gebeurtenisgedreven werken en geautomatiseerd notificeren. We zien het als een 'neutraal patroon' waarbij allerlei implementatiekeuzes mogelijk zijn. Zowel op business- als applicatieniveau.



Figuur 4: Notificatieprocessen op business- en applicatieniveau

Op businessniveau kunnen bijv. verschillende taakverdelingen tussen Producer, Intermediary en Consumer worden gebruikt waarbij bepaalde taken binnen of buiten de eigen organisatie zijn te beleggen. Gelet op de diversiteit binnen het overheidslandschap zullen er op dit vlak met goede redenen verschillende keuzes worden gemaakt.

Ook op applicatieniveau geldt dat contextuele factoren een belangrijke rol spelen bij het maken van keuzes. De aanbevelingen in dit hoofdstuk zijn daarom gebaseerd op best practices maar zullen altijd binnen de specifieke context op hun bruikbaarheid moeten worden getoetst. Overeenkomstig de manier waarop binnen de overheid wordt omgegaan met standaarden kunnen ze, afhankelijk van organisatie-eigen keuzes, worden beschouwd als 'pas-toe-of-leg-uit' of 'aanbevolen'.



7 IMPLEMENTATIE-PATRONEN

7.1 INLEIDING

Dit hoofdstuk beschrijft een aantal implementatiepatronen die bruikbaar zijn bij het ontwerpen van event-driven oplossingen. Notificeren van Consumers is daar een onderdeel van en wordt veelal ondersteund door een specifieke implementatie van het Publish-Subscribe patroon. De in dit hoofdstuk beschreven patronen hebben een verschillend karakter. Voor allemaal geldt echter dat ze kunnen bijdragen aan realisatie van goede event-driven oplossingen.

Voor de patronen in de eerste helft van dit hoofdstuk ('Change Data Capture', 'Basale messaging', 'Webhook', 'Event Notification', 'Event Carried State Transfer') geldt dat ze een duidelijke relatie hebben met notificeren. Toepassing van die patronen is (ook) mogelijk door gebruik te maken van binnen de overheid bekende stijlen als service-oriëntatie en REST en van vaker gebruikte voorzieningen zoals message brokers, message queues en relationele databases. Stijlen en voorzieningen die vaak worden toegepast bij oplossingen voor het opvragen van (bron)gegevens.

Voor de patronen in de tweede helft van het hoofdstuk ('event streaming', 'event processing', 'event sourcing' en 'CQRS') geldt dat ze meer doelen hebben dan alleen notificeren van afnemers. Gebruik van die patronen betekent een fundamentele verandering ('paradigmashift') ten opzichte van hoe oplossingen nu worden ontworpen, gerealiseerd en gebruikt. Een essentieel verschil is bijv. dat het gebruikelijke vastleggen van objecten met attributen vervangen wordt door het vastleggen van gebeurtenissen en die vastlegging te beschouwen als bronregistratie. Vastgelegde gebeurtenissen beschouwen als 'single source of truth' in plaats van gegevensobjecten die worden geactualiseerd is een fundamenteel andere zienswijze en vereist gebruik van andere patronen en voorzieningen.

De mate waarin dit laatste type patronen bruikbaar zijn wordt sterk bepaald door de aanwezige ambities, behoeften, kennis en mogelijkheden van betrokken organisaties en applicaties. Het aantal organisaties binnen de overheid dat min of meer op deze manier werkt is nog zeer beperkt. Voor organisaties die nog weinig ervaring hebben met event-driven werken geldt dat kennisname van deze patronen nuttig is om er eventueel op langere termijn stapsgewijs gebruik van te kunnen maken. Een 'winstwaarschuwing' is daarbij wel op zijn plaats omdat gebruik ervan, zeker in het begin, niet gemakkelijk is en geen 'silver bullet' is maar ook (andersoortige) haken en ogen kent.

Ieder patroon wordt afzonderlijk beschreven. In de praktijk kunnen patronen ook worden gecombineerd tot een nieuw samengesteld patroon ('compound pattern'). Hoewel implementatiepatronen duidelijk richting geven aan hoe een oplossing eruit moet komen te zien zijn ze nog geen recept dat klakkeloos kan worden gevolgd. Ook hier geldt dat contextuele factoren een belangrijke rol spelen bij het bepalen van welk patroon op welke manier het beste is te gebruiken.

7.2 CHANGE DATA CAPTURE

Doel	Gebeurtenissen kunnen publiceren terwijl applicaties en registraties data-georiënteerd zijn.
Probleem	De meeste in gebruik zijnde applicaties en registraties binnen de overheid zijn gericht op het bijhouden van gegevens zodat de actuele status van objecten bekend is. Naar aanleiding van gebeurtenissen worden gegevens weggeschreven zonder de context van de achterliggende plaatsgevonden gebeurtenis te bewaren. Het is dan niet of beperkt mogelijk om events met gebeurtenisinformatie te publiceren.
Oplossing	Leidt op basis van plaatsgevonden gegevenswijzigingen plaatsgevonden gebeurtenissen af.



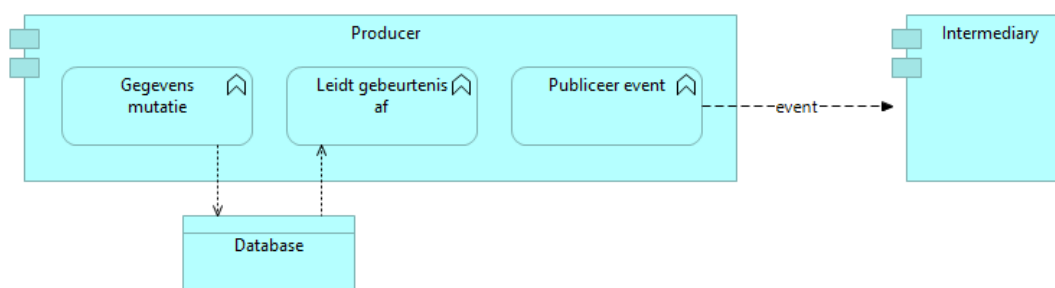
Gevolgen

- Naar aanleiding van systeem-gebeurtenissen (bijv. CRUD-mutaties) zijn events te publiceren.
- Om op basis van gegevensmutaties bedrijfsgebeurtenissen te kunnen publiceren moet een 'vertaalslag van data naar gebeurtenis' plaatsvinden. Afhankelijk van de context is dat mogelijk met een verschillende mate van betrouwbaarheid.

Voorbeeld

- Op basis van aanmaak van een nieuw gegevensobject (bijv. in de vorm van een rij in een databasetabel) kan een event 'ObjectAangemaakt' worden gepubliceerd (systeem-gebeurtenis).
- Op basis van het vullen van een attribuut DatumOverlijden met een datum kan een event worden afgeleid en gepubliceerd met als type 'PersoonOverleden' (business-gebeurtenis).
- Voor meer informatie over systeem- versus business-gebeurtenis zie [Business- en systeem-gebeurtenissen](#).

Werking



Implementatie Implementatie is op verschillende manieren mogelijk. Enkele mechanismen die bruikbaar zijn om naar aanleiding van een gegevensmutatie een business-event af te leiden en publiceren:

- Gebruik van triggers op databasetabellen die reageren op het plaatsvinden van bepaalde typen gegevensbewerkingen (bijv. toevoegen, wijzigen of verwijderen).
- Periodiek scannen van data waarbij op basis van bepaalde statusgegevens (bijv. versienummers of timestamps) is af te leiden dat een bepaald soort gebeurtenis heeft plaatsgevonden.
- Gebruik van specialistische software die kan worden geconfigureerd om op basis van bepaalde type mutaties specifieke events te publiceren.

Opmerkingen

- Gelet op het feit dat de huidige applicaties en registraties binnen de overheid veelal data-georiënteerd zijn ingericht zal dit patroon de komende jaren vaak bruikbaar zijn om applicaties en registraties (meer) geschikt te maken voor event-driven werken. De handreiking '[Introductie van notificeren](#)' beschrijft meer in detail hoe vanuit verschillende startsituaties meer event-driven werken is te realiseren.
- Het periodiek leveren van 'was/wordt mutaties' is te zien als een specifieke vorm van het Change Data Capture patroon. Meestal betreft het informatierijke notificaties met zowel de status van objecten vóór als de status van een object ná plaatsgevonden mutaties³¹. Het doel is vaak om objectgegevens binnen de eigen

³¹De mate waarin berichten informatierijk zijn kan verschillen. In minimale vorm bevat een bericht alleen was-wordt gegevens van gewijzigde objectgegevens maar een bericht kan ook alle gegevens van het gewijzigde object bevatten (of zelfs meer als een object wordt gezien als deel van een meer omvattend object).



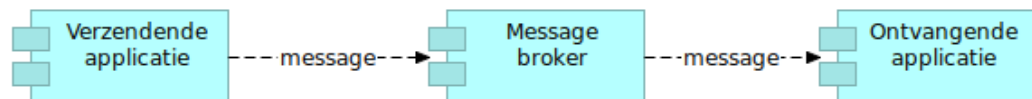
omgeving binnen een acceptabele termijn te kunnen actualiseren (bijv. als externe gegevens zijn gerepliceerd) en niet om actie te ondernemen direct na het optreden van gebeurtenissen. Voor dat doel moeten direct na het optreden van gebeurtenissen notificaties worden gestuurd met informatie over de gewijzigde status van objecten ('Event-Carried State Transfer').

- Ook als de gegevenshuishouding meer event-driven is ingericht kan het Change Data Capture patroon bruikbaar zijn om in specifieke behoeften te voorzien. Bepaalde wijzigingen kunnen bijv. als 'gebeurtenisstroom' worden aangeboden aan voorzieningen die ze, eventueel aangepast, doorleveren aan Consumers (bijv. om bepaalde analyses te maken of trends te signaleren).

7.3 BASALE MESSAGING

Doel	Gegevens via berichten flexibel, betrouwbaar en asynchroon kunnen uitwisselen tussen applicaties.
Probleem	Uitwisselen van gegevens tussen applicaties is lastig als ze op bepaalde aspecten van elkaar verschillen (bijv. qua technologie, locatie, beschikbaarheid).
Oplossing	Verpak gegevens in zelfstandig te verwerken berichten die zijn te bewaren en op verschillende momenten zijn te verwerken.
Gevolgen	Betrokken applicaties worden van elkaar ontkoppeld en kunnen los van elkaar functioneren.
Voorbeeld	Een bestelapplicatie stuurt na het plaatsen van een bestelling via een message broker een bericht naar een interne logistieke applicatie en naar een financiële applicatie in een SaaS omgeving waar financiële afhandeling plaatsvindt.

Werking



Implementatie	<ul style="list-style-type: none">- Ontkoppel verzender en ontvanger door gebruik te maken van een 'message-broker applicatie' die als intermediair fungeert voor verzender en ontvanger. De intermediair kan bijv. tijdelijk berichten bewaren en later opnieuw proberen af te leveren als een ontvanger tijdelijk niet bereikbaar is.
Opmerkingen	<ul style="list-style-type: none">- De reden om te spreken over 'basale messaging patronen' is dat de Message-driven architectuurstijl (zie hoofdstuk 'Architectuurstijlen') veel verschillende patronen kent. We bedoelen hier de basispatronen die in de praktijk vaak worden toegepast.- Er bestaan veel messaging-patronen om aan context-specifieke eisen te kunnen voldoen. Voor een uitgebreid overzicht daarvan verwijzen we naar https://www.enterpriseintegrationpatterns.com/patterns/messaging/.- Messaging als patroon is geschikt voor allerlei vormen van gegevensuitwisseling tussen applicaties. Specifiek voor notificeren geldt dat Publishers berichten gegevens over plaatsgevonden gebeurtenissen in de vorm van berichten kunnen publiceren en Intermediaries berichten kunnen bewaren en als notificatie aan Consumers kunnen verstrekken.



- Messaging heeft als doel om een bericht van A naar B te brengen. Berichten worden hooguit tijdelijk bewaard voor systeemtechnische doelen (bijv. gegarandeerde aflevering) en berichten geïsoleerd van elkaar verwerkt³².
- Gelet op hoe de meeste applicaties en registraties binnen de overheid zijn ontworpen zal messaging in basale vorm een veel gebruikt patroon zijn om te notificeren. Daarnaast zullen ook traditionele patronen zoals bestandsuitwisseling en databasekoppelingen in gebruik blijven. Binnen project Notificatieservices beschouwen we dat type patronen als geschikt om op een laagdrempelige manier stapsgewijs meer event-driven te kunnen gaan werken.
- Er zijn veel, vaak al tientallen jaren bestaande, applicaties die de rol van ‘message broker’ kunnen vervullen. Naast lokaal te installeren applicaties wordt in toenemende mate gebruik gemaakt van applicaties die vanuit een Cloud-omgeving worden aangeboden³³.

7.4 WEBHOOK

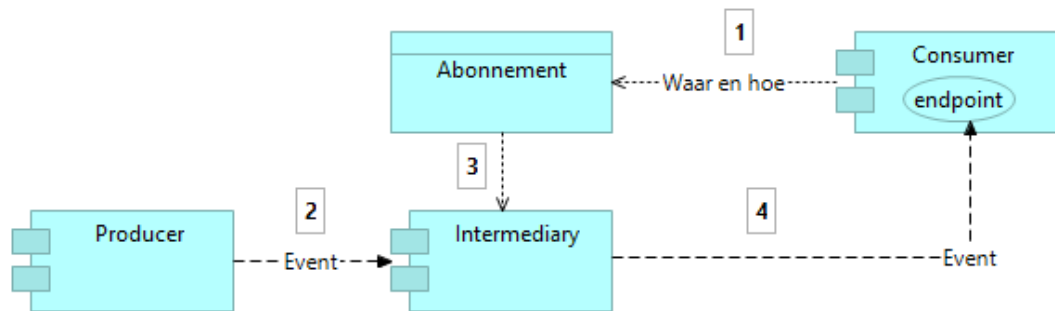
Doel	Verstrekken van events op een door Consumers gespecificeerde plaats en manier.
Probleem	Consumers hebben verschillende mogelijkheden en voorkeuren om events te kunnen ontvangen.
Oplossing	Laat Consumers zelf, bij abonneren, specificeren op welke plaats en op welke manier ze events willen ontvangen en gebruik die informatie bij verstrekking.
Gevolgen	<ul style="list-style-type: none">- Consumers zijn verantwoordelijk voor specificatie waar en hoe ze events willen ontvangen (bijv. op een opgegeven endpoint dat via HTTP bereikbaar is en waar via een POST methode notificaties zijn te verstrekken).- Consumers moeten in staat zijn events te ontvangen en daarbij te voldoen aan de afspraken m.b.t. verstrekking (bijv. 24x7 bereikbaar zijn of na een HTTP-Post conform afspraken een HTTP-response geven).
Voorbeeld	<ul style="list-style-type: none">- Een Consumer ontwikkelt een webservice die in staat is om een event te ontvangen via een HTTP POST-methode. Hij meldt bij de intermediair die voor verstrekking van events zorgt de locatie ('endpoint') waar de service zich bevindt waar de Intermediary via een HTTP POST-methode events kan verstrekken.

³²Met ‘geïsoleerd van elkaar vewerkt’ bedoelen we dat berichten niet worden beschouwd als onderdeel van een betekenisvolle stroom van berichten zoals het geval is bij patronen zoals ‘Even streaming’ en ‘Event sourcing’.

³³“Message broker” kan in deze context ook betekenen dat er sprake is van een combinatie van componenten (bijv. een aparte message queue applicatie) die samen de benodigde functionaliteit realiseren (ook vaak aangeduid als ‘de enterprise servicebus’). Het gaat hier om de kernfunctionaliteit om berichten te kunnen ontvangen, bewaren en af te leveren.



Werking

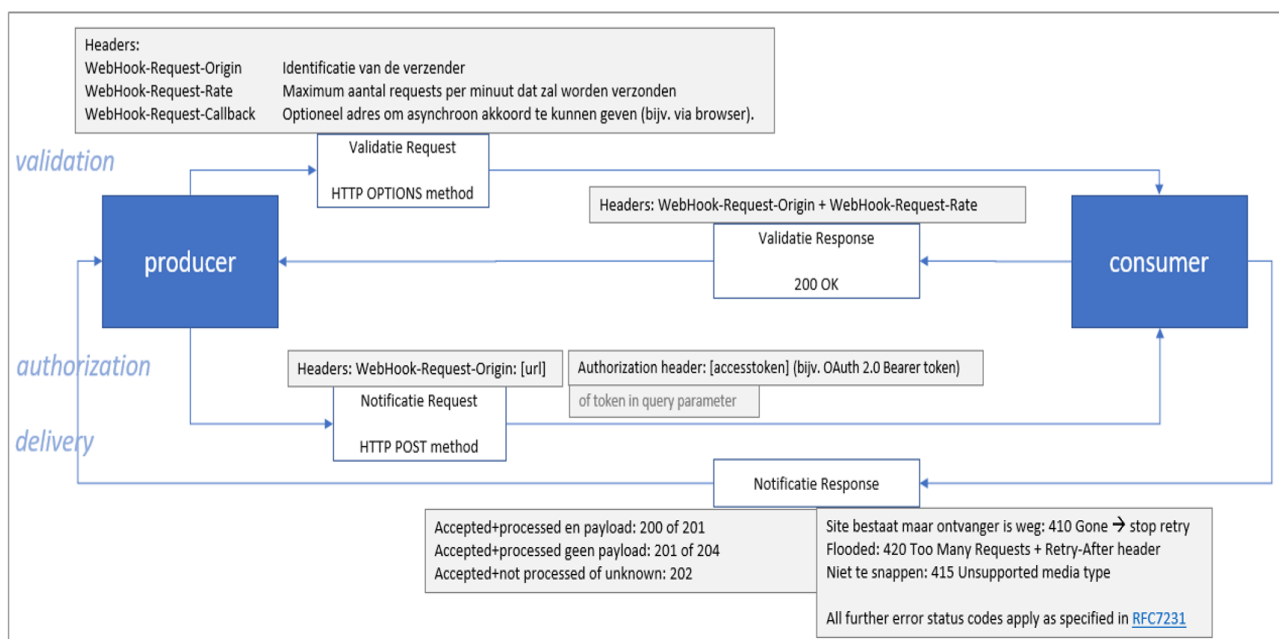


Implementatie

- Ontkoppel Consumer en Intermediary door Consumers te laten bepalen waar en hoe events worden verstrekt (binnen de mogelijkheden die de Intermediary biedt).

Opmerkingen

- Webhook is een vaak toegepast patroon en wordt [breed ondersteund](#). Mede door de bekendheid ermee is het in veel situaties een relatief eenvoudig patroon om toe te passen.
- Voor geautomatiseerd abonneren door Consumers wordt vanwege de bekendheid ermee en de relatieve eenvoud van gebruik vaak gewerkt met een REST-API die CRUD-operaties ondersteunt voor aanmaken, raadplegen, bewerken en verwijderen van abonnementen.
- Hoewel webhook als patroon vaak wordt gebruikt is het niet gestandaardiseerd. Reden waarom binnen de CloudEvents-community hiervoor een standaard is ontwikkeld: [HTTP 1.1 Web Hooks for Event Delivery](#). Hierin zijn onder andere keuzes gemaakt hoe applicaties met elkaar communiceren voor o.a. validatie van endpoints en het geven van feedback na ontvangst van een notificatie. Onderstaande afbeelding toont globaal welke afspraken binnen deze standaard zijn gemaakt over de interactie tussen Producers (of Intermediaries) en Consumers.
- De binnen project Notificatieservices in samenwerking met VNG Realisatie opgestelde [API-specificatie](#) voor notificeren past dit patroon toe.





7.5 EVENT NOTIFICATION

In het hoofdstuk 'Aspecten' is ingegaan op informatiearm en informatierijk notificeren. Bij informatiearm notificeren is 'Event Notification' het toegepaste patroon.

Doel Verstrekken van minimale gegevens over plaatsgevonden gebeurtenissen in notificaties om Consumers op hoogte te stellen dat een bepaalde gebeurtenis heeft plaatsgevonden.

Probleem In bepaalde situaties moeten notificaties worden verstrekt waarin minimaal informatie aanwezig is. Bijv. omdat het onwenselijk of niet toegestaan is om (alle) gegevens over een plaatsgevonden gebeurtenis via notificaties aan Consumers te verstrekken of omdat het om zodanige hoeveelheden gegevens gaat dat het verstrekken via berichten op praktische problemen stuit.

Oplossing Neem in notificaties alleen de voor afnemers minimaal benodigde gegevens op en neem een verwijzing op naar een service waar aanvullende gegevens zijn op te vragen.

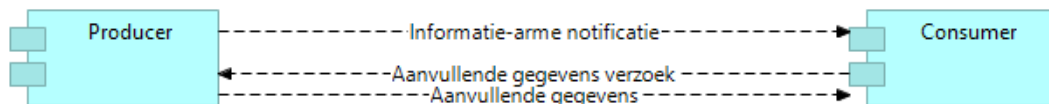
Gevolgen

- Consumers ontvangen minimale gegevens op basis waarvan zij moeten kunnen beslissen of er aanvullende gegevens moeten worden opgehaald. Afhankelijk van de context volstaat het om alleen metagegevens over de gebeurtenis op te nemen of worden ook beperkt inhoudelijke gebeurtenisgegevens opgenomen.
- Producers moeten, bij voorkeur permanent, een service aanbieden waarmee afnemers aanvullende gegevens kunnen ophalen. Voor gebruik van de service moet bij de aard van de op te vragen gegevens passende authenticatie, autorisatie en beveiliging zijn geregeld.
- De door een Producer aangeboden opvraagservice moet 'tijdgebonden opvraging' van aanvullende informatie ondersteunen als het voor Consumers niet volstaat is om bij opvraging altijd de meest actuele gegevens te ontvangen (maar bijv. gegevens te ontvangen die geldig waren op het moment van notificatie).
- Producers moeten, zeker bij verstrekking van vertrouwelijke gegevens, zowel loggen aan welke afnemers notificaties zijn verstrekt als welke afnemers welke aanvullende gegevens hebben opgevraagd.

Voorbeeld

- Een afnemer ontvangt een informatiearme notificatie dat een nieuwe productaanvraag is ingediend en gaat met behulp van een opgenomen link de betreffende aanvraaggegevens ophalen.
- Een afnemer ontvangt een notificatie dat een bepaald type gebeurtenis heeft plaatsgevonden waarbij een object betrokken was waarmee de afnemer een relatie heeft. Naar aanleiding daarvan vraagt hij aanvullende gegevens over het object op.

Werking



Implementatie

- Producers moeten een service bieden die Consumers kunnen aanroepen voor het verkrijgen van aanvullende informatie.
- Producers kunnen bij aanroep van de service voor aanvullende informatie authenticatie- en autorisatieregels toepassen om te bepalen welke aanvullende gegevens aan welke Consumers worden verstrekt. Hiermee kan mogelijk gebruik worden gemaakt van bestaande opvraagservices en kan complexe filtering op basis van



autorisatie bij notificeren worden vermeden.

- Opmerkingen**
- In de praktijk wordt 'event notification' soms gezien als voorwaarde om met recht te mogen spreken over 'notificeren'. Binnen de context van project Notificeren zien we 'notificeren' ruimer en vatten het op als het verstrekken van informatie over plaatsgevonden gebeurtenissen waarbij ook andere patronen kunnen worden toegepast.
 - Het patroon is gebaseerd op het 'claim check' of 'message reference' patroon dat bij messaging wordt toegepast. Berichten worden daarbij opgesplitst in een 'claim check' en een 'payload'. De ontvanger ontvangt de claim-check die hij bij een service kan inwisselen voor de daarbij behorende payload-gegevens.

7.6 EVENT CARRIED STATE TRANSFER

In het hoofdstuk 'Aspecten' is ingegaan op informatiearm en informatierijk notificeren. Bij informatierijk notificeren is het gebruikte patroon 'Event Carried State Transfer'.

Doel Verstrekken van alle voor afnemers relevante gegevens over plaatsgevonden gebeurtenissen in notificaties.

Probleem In bepaalde situaties moeten notificaties worden voorzien van alle relevante gegevens voor Consumers. Bijv. omdat het wenselijk of niet anders mogelijk is. Het kan daarbij bijv. gaan om een beperkte set openbare gegevens of de Publisher wil of kan geen service bieden voor Consumers om na ontvangst van een notificatie aanvullende gegevens op te vragen.

Oplossing Neem in notificaties alle voor afnemers relevante gebeurtenisgegevens op.

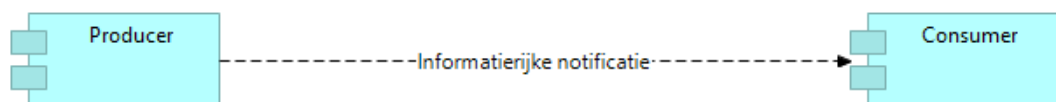
Gevolgen

- Consumers ontvangen alle relevante gegevens naar aanleiding van een gebeurtenis en kunnen direct na ontvangst tot verwerking daarvan overgaan.
- Producers kunnen volstaan met het verstrekken van informatierijke notificaties.

Voorbeeld

- Een afnemer ontvangt een informatierijke notificatie van een nieuwe productaanvraag en kan meteen tot verwerking daarvan overgaan.
- Een afnemer ontvangt een notificatie dat een bepaald type gebeurtenis heeft plaatsgevonden inclusief de actuele gegevens van een betrokken object waarmee de afnemer een relatie heeft en kan meteen tot verwerking overgaan.

Werking



Implementatie

- De implementatie is relatief eenvoudig ten opzichte van het event notification patroon omdat alle betrokken gegevens tegelijkertijd worden verstrekt. Het patroon zorgt voor vergaande ontkoppeling tussen Producer en Consumer. Producers zijn klaar nadat ze notificaties hebben gepubliceerd en (laten) verstrekken. Consumer-applicaties kunnen snel en betrouwbaar over (kopie-)brongegevens beschikken omdat er geen externe opvraging nodig is.

Opmerkingen

- Dit patroon wordt vaak toegepast als synchronisatiemechanisme om bij Consumers lokale kopieën ('replica's') van bij de Producer aanwezige brongegevens te maken en bij te houden.
- Bij het opbouwen van lokale kopieën moet rekening worden gehouden met het feit dat



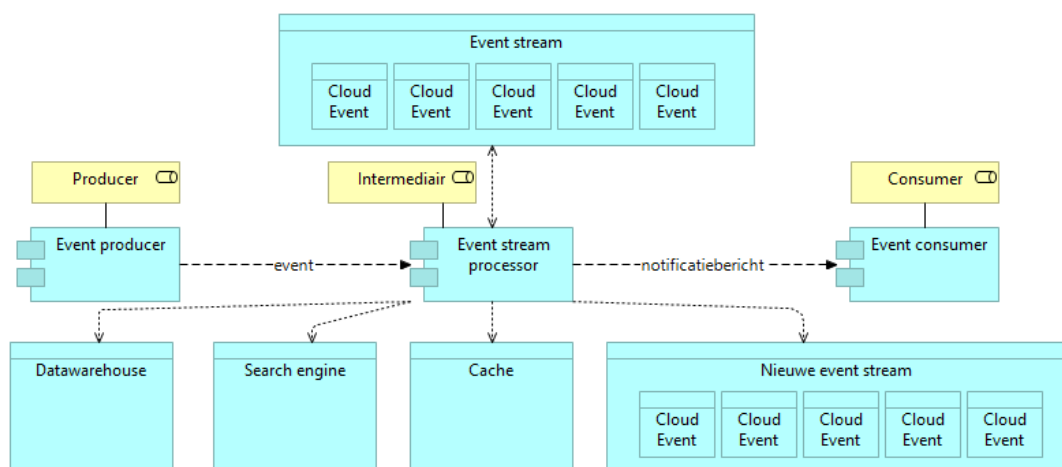
gegevens 'achter kunnen lopen' op gegevens in de authentieke bronregistratie. In hoeverre dit een probleem vormt is afhankelijk van de context.

- Net zoals bij ieder ander patroon kunnen er tijdens gebruik fouten optreden waarvoor speciale maatregelen moeten worden getroffen (bijv. opnieuw leveren van berichten als de Consumer bepaalde berichten niet heeft ontvangen of er een nieuwe replica moet worden opgebouwd).

7.7 EVENT STREAMING

Doel	Berichten met gebeurtenisinformatie in samenhang ('als stream') kunnen gebruiken voor verschillende doeleinden.
Probleem	De intrinsieke waarde die gebeurtenisberichten, individueel en als verzameling, hebben gaat verloren als ze na verwerking niet worden bewaard.
Oplossing	Beschouw elkaar opvolgende berichten met informatie over gebeurtenissen als waardevol en bewaar ze op een zodanig manier dat ze bruikbaar blijven voor verschillende doelen ³⁴ .
Gevolgen	<ul style="list-style-type: none">- Gebeurtenissen worden beschouwd als 'single source of truth'; de status van objecten is afleidbaar van verzamelingen bewaarde gebeurtenissen.- Gebeurtenisberichten moeten met dezelfde zorgvuldigheid worden behandeld en bewaard als gebruikelijk is bij objectgegevens in bronregisters.
Voorbeeld	<ul style="list-style-type: none">- Door realtime berichten over plaatsvindende gebeurtenissen in samenhang te bezien kunnen patronen en trends worden ontdekt waarop passend kan worden gereageerd (bijv. voor het constateren van beveiligingsincidenten of op basis van een patroon fraude te detecteren).- Bewaren biedt mogelijkheden om gebeurtenisberichten opnieuw 'af te spelen' op basis waarvan bepaalde conclusies zijn te trekken (bijv. ten behoeve van audit-doeleinden of om de oorzaak van opgetreden fouten te vinden).- Naast grofmazige business-events kunnen ook fijnmazig events worden gestreamed. Bijv. door monitoren van realtime events als gevolg van het surfgedrag van een websitebezoeker kunnen maatregelen worden getroffen om klantreis te optimaliseren (bijv. door hulp aan te bieden of op bepaalde gerelateerde diensten te wijzen).

Werking



³⁴De periode dat berichten worden bewaard kan uiteraard verschillen. Incidenteel kan bewaring ook permanent plaatsvinden zoals dat bij het Event Sourcing patroon gebeurt.



Het Publish-Subscribe patroon voor notificeren is te implementeren via het event streaming patroon. Maar het levert daarnaast veel andere mogelijkheden op om informatie over gebeurtenissen vast te leggen en gebruiken. Naast de hierboven getoonde voorbeelden van 'datawarehouse', 'search engine' en 'cache' is ook weergegeven hoe er ook allerlei soorten nieuwe streams zijn te maken voor verschillende doeleinden. Hierdoor kan een netwerk aan streams ontstaan.

Implementatie Het event streaming patroon wordt vaak gebruikt als het om grote hoeveelheden data gaat die snel verwerkt moeten worden en voor langere tijd bewaard moeten worden. Soms is daarbij ook functionaliteit voor bewerkingen of analyse nodig. Gebruik van het event streaming patroon vereist gebruik van gespecialiseerde applicaties of 'platforms' met Apache Kafka als bekendste voorbeeld. Kafka kent een 'Stream API' waarmee Producers en Consumers flexibel en betrouwbaar gebruik kunnen maken van streams om grote hoeveelheden gegevens te verwerken.

Opmerkingen

- Wanneer gegevens over gebeurtenissen geen metadata bevatten spreken we over 'data' en 'data streams' (bijv. bij sensoren die alleen een pakket ruwe data leveren). Als er wel contextgegevens aanwezig spreken we over 'events' en 'event streams' (bijv. bij gebeurtenisgegevens die op basis van metakenmerken moeten worden geïnterpreteerd om verwerkt te kunnen worden)³⁵.
- Zowel data- als event-streams kunnen waardevol zijn. De aanwezigheid van goede metagegevens verhoogt de bruikbaarheid van berichten (bijv. omdat daarmee routing en filtering mogelijk is zonder inhoudelijke kenmerken in te hoeven zien).
- Metadata en inhoudelijke berichtdata moeten herkenbaar worden onderscheiden. Bijvoorbeeld door metadata in berichten op te nemen als 'header data' en inhoudelijke gegevens als 'payload data' (bijv. via aparte secties binnen een bericht of bij protocollen die dit ondersteunen in aparte frames).
- Het event streaming patroon is opgenomen omdat het is te zien als een specifieke implementatie van het Publish-Subscribe patroon. Gelet op de eisen die gebruik van het patroon stelt ligt gebruik van het patroon niet voor de hand als er alleen eenvoudige notificatieberichten nodig zijn. Naarmate een hoger volwassenheidsniveau van event-driven werken wordt bereikt, en er steeds meer events komen, wordt het aantrekkelijker om met event streaming te gaan werken.

7.8 EVENT PROCESSING

Doel	Het interpreteren en verwerken van gebeurtenisgegevens voor een bepaald doel of Consumer.
Probleem	Gebeurtenisgegevens zoals ze aangeleverd worden door een Producer zijn niet altijd voor alle doelen of Consumers geschikt.
Oplossing	Interpreteer en verwerk gebeurtenisgegevens zodanig dat ze geschikt worden voor gebruik.
Gevolgen	Gegevens over gebeurtenissen worden op verschillende manieren geïnterpreteerd en verwerkt en kunnen tot afgeleide gegevens leiden (bijv. qua inhoud of vorm). Bijv. door gegevens te transformeren of te combineren.

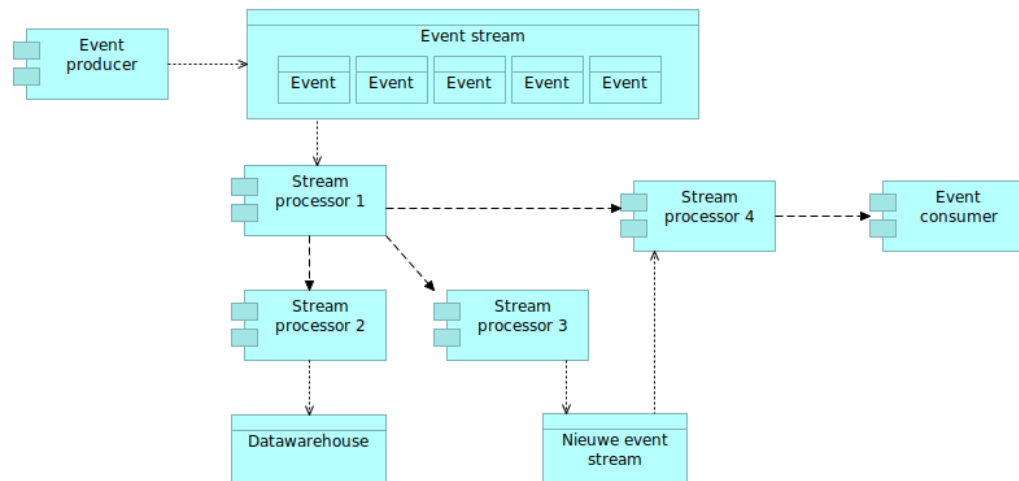
³⁵In de praktijk wordt dit onderscheid niet altijd gemaakt en wordt ook bij ruwe datastromen gesproken over 'eventstreams'.



Voorbeeld

- Het filteren van beschikbare gebeurtenisgegevens voor verstrekking aan afnemers is een veelvoorkomende vorm van event processing.
- Beschikbare gebeurtenisgegevens worden gefilterd en omgevormd tot gegevens die binnen een specifieke omgeving bruikbaar zijn (bijv. een datawarehouse of een cache waarin het laatst ontvangen bericht over een bepaald object staat).

Werking



Bij de beschrijving van event streaming is al een voorbeeld getoond van hoe 'event processing' plaats kan vinden binnen een streaming platform als Kafka. In bovenstaand diagram is weergegeven hoe voor event processing in plaats van met 1 applicatie/platform steeds vaker gebruik wordt gemaakt van aparte componenten die (vaak via events) met elkaar communiceren. Een voorbeeld daarvan is het gebruik van serverless functies binnen Cloud omgevingen waarbij iedere functie specifieke verwerkingsfunctionaliteit levert.

7.9 EVENT SOURCING

Doel

Duurzaam kunnen gebruiken van gegevens over plaatsgevonden gebeurtenissen.

Probleem

Als gebeurtenissen alleen worden gebruikt om objecten in een bronregister te creëren, wijzigen of verwijderen maar niet worden bewaard gaat veel informatie verloren en is event-driven werken niet optimaal te ondersteunen.

Oplossing

Bewaar gebeurtenissen onveranderlijk en duurzaam in een bronregister ('event store') en benoem dit als authentieke bron ('systems of record').

Gevolgen

- Een bronregister bestaat niet meer uit vastgelegde gerelateerde objecten met attribuutwaarden ('data-store') maar uit bewaarde gebeurtenisgegevens in een 'event-store'.
- Op basis van vastgelegde gebeurtenisgegevens is te herleiden welke gebeurtenissen in welke volgorde hebben plaatsgevonden ('event-log').
- Bij gebruik van een eventstore worden nieuwe gebeurtenisgegevens ('log based') toegevoegd en daarna niet meer gewijzigd of verwijderd. Vastgelegde gegevens representeren feiten die onveranderlijk zijn ('immutable').
- De actuele waarden van attributen van objecten kan worden bepaald door vastgelegde events in de eventstore te raadplegen en 'af te spelen'. Er kan ook gebruik worden



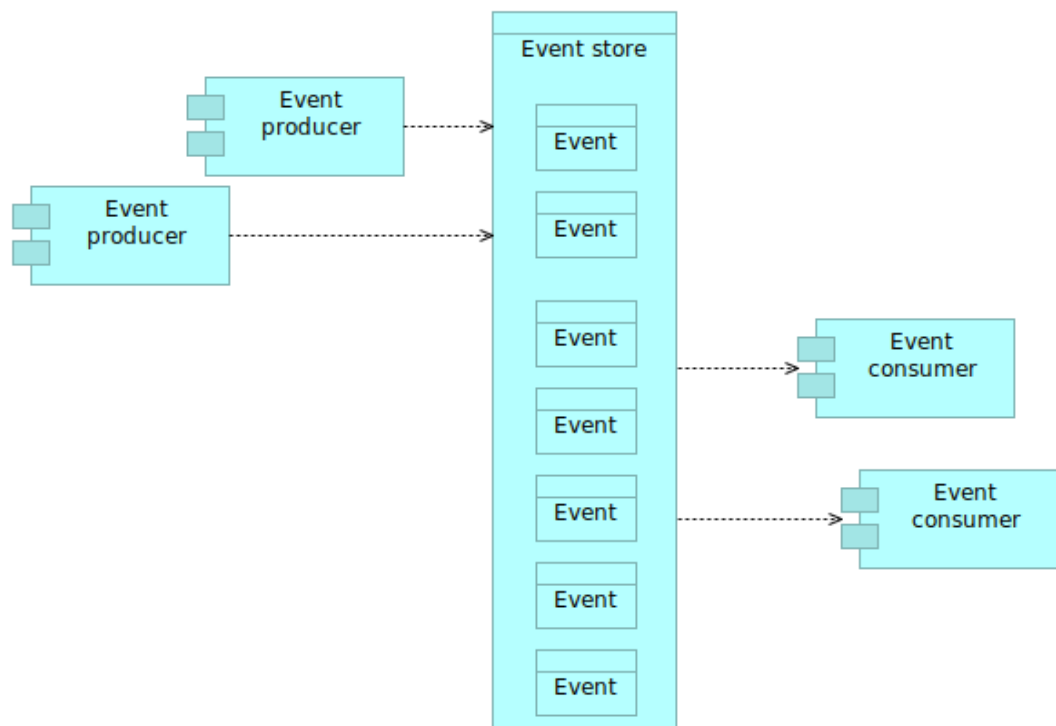
gemaakt van afgeleide views waarin actuele gegevens op de gebruikelijke manier zijn op te vragen (via het hierna beschreven CQRS-patroon).

- Het wordt mogelijk om bij gebruik van informatiearme notificaties via links te verwijzen naar de gebeurtenis waarop de notificatie betrekking heeft.
- Het wordt mogelijk om historische bevestigingen te doen omdat alle gebeurtenisinformatie, inclusief tijdstip van optreden en registratie worden bewaard (wat nieuwe mogelijkheden biedt voor bijv. audits of voor specifieke Consumer-behoefte).
- Verschillende toestanden, waaronder de actuele toestand, zijn te reconstrueren (bijv. om een read-only registratie op te bouwen of om maatwerk-views met alleen bepaalde soorten gebeurtenissen op te bouwen).

Voorbeeld

- Als een klant op een website producten toevoegt aan en verwijdert uit zijn winkelmandje kan voor iedere actie een 'event' in de eventstore worden opgeslagen. Bij het afrekenen is aan de hand van de opgeslagen events te bepalen wat de actuele inhoud van het winkelmandje is en wat hij moet betalen. Het betalen zelf wordt daarna ook als event in de eventstore opgeslagen. De transactie is daarmee afgerond en kan als event van het type 'AankoopGedaan' worden opgeslagen.
- De afdeling Logistiek heeft zich geabonneerd op events van het type 'AankoopGedaan' en ontvangt daarover notificaties. In de eventstore is vervolgens op te zoeken welke artikelen zijn aangekocht en verzonden moeten worden. De verzending van de artikelen wordt als nieuw event in de eventstore vastgelegd (op basis waarvan een klant bijv. online kan zien dat de bestelde artikelen zijn verzonden).
- De afdeling Marketing kan op basis van de vastgelegde events analyses doen van het koopgedrag van klanten (bijv. hoe vaak en welke artikelen uit winkelmandjes worden verwijderd), van de tijd tussen betaling en verzending, etc..

Werking



Implementatie - Een eventstore is te implementeren met gebruik van traditionele databases (bijv. een relationele- of nosql-database). Het betekent echter een heel ander soort gebruik dan



waarvoor dit type databases zijn ontworpen³⁶. Er zijn ook databases die speciaal zijn ontworpen om als eventstore te worden gebruikt (bijv. EventStoreDB).

- Een event store is primair bedoeld om gebeurtenissen goed in een bronregistratie vast te leggen. Vanuit het streven naar ontkoppeling kan het onwenselijk zijn om Consumers direct gebruik te laten maken van de eventstore. Voor afnemers kunnen aanvullende voorzieningen worden ontwikkeld (waarmee bijv. om afnemervriendelijke read-models worden gevoed).
- Voor het bepalen van grenzen, niveau van detail, wijze van vastlegging en verstrekking naar afnemers uit andere domeinen wordt vaak gebruik gemaakt van inzichten uit de Domain Driven Design architectuurstijl (zie [Domain Driven Design](#)).

- Opmerkingen**
- Het is belangrijk om bij event sourcing onderscheid te maken in ‘business events’ (herkenbaar voor domeinexperts) en ‘system events’ (herkenbaar voor IT’ers). Afnemers zullen meestal behoefte hebben aan voor hen herkenbare business events. In de handreiking [‘Gebeurtenistypes definiëren’](#) is beschreven via welke stappen bruikbare business-events kunnen worden benoemd.
 - Afhankelijk van de context is event sourcing relatief eenvoudig of zeer complex toe te passen. Event sourcing is met name geschikt voor stabiele omgevingen waarin vastgelegde events gegarandeerd onveranderlijk (‘immutable’) zijn en waarin business logica niet vaak wijzigt. Is dat wel het geval dan kan gebruik van event sourcing snel (heel) complex worden.
 - Werken met een eventstore in plaats van een traditionele database betekent een grote omschakeling voor de organisatie die verantwoordelijk is voor het bronregister. Afhankelijk van of en hoe de voorzieningen voor Consumers zijn ingericht kan dit ook voor Consumers het geval zijn. Op dit moment is de kennis en ervaring met event sourcing binnen de overheid nog zeer beperkt.

7.10 COMMAND/QUERY RESPONSIBILITY SEGREGATION

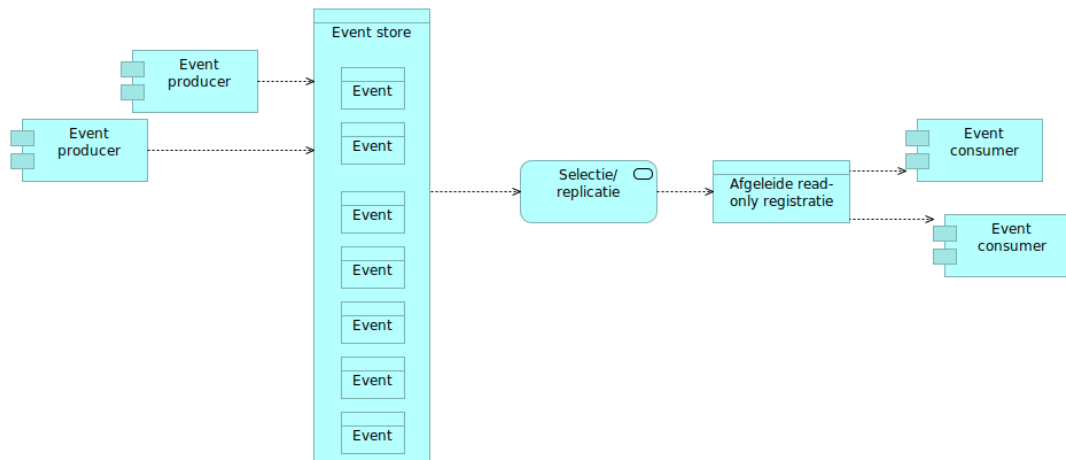
Doel	Het wijzigen en het lezen van (gebeurtenis)gegevens via verschillende voorzieningen ondersteunen om beide te kunnen optimaliseren.
Probleem	Het combineren van schrijven en lezen van gegevens leidt tot suboptimale oplossingen.
Oplossing	Gebruik aparte voorzieningen voor het muteren en het lezen van gegevens.
Gevolgen	Zowel de voorzieningen voor muteren als voor lezen kunnen zich richten op hun kernfunctie en los van elkaar worden geoptimaliseerd. Bij muteren worden gegevens bijvoorbeeld als gebeurtenis in een event source of als databaseobject vastgelegd terwijl voor lezen eenvoudig bruikbare objecten worden aangeboden (bijv. via maatwerk read-only views). Het patroon wordt vaak gebruikt in combinatie met event sourcing om gegevens afnemervriendelijk beschikbaar te stellen.
Voorbeeld	Applicatie A legt via het event sourcing patroon een gebeurtenis met een hoog niveau van detail vast en in een eventstore. De event-gegevens, of een deel daarvan, worden

³⁶In een traditionele database worden wijzigingen in een logfile verzameld maar vormen de databaseobjecten het bronregister. In een eventstore is het precies andersom. Reden waarom het ook wel wordt aangeduid als: [‘Turning the database inside out’](#).



gerepliceerd naar een aparte registratie in een vorm die eenvoudig, snel en op maat raadplegen mogelijk maakt. Afnemers die informatie over gebeurtenissen willen opvragen, bijv. na notificatie, kunnen via een aan te roepen service gegevens uit de 'read only' registratie ophalen.

Werking



- Implementatie** - Voor implementatie is gebruik te maken van verschillende mechanismen (bijv. selectie, filtering, replicatie). Afhankelijk van de mogelijkheden en wensen van afnemers kunnen verschillende keuzes, met verschillende 'read models', worden gemaakt. Gebeurtenissen kunnen bijvoorbeeld worden 'vertaald' naar herkenbare objecten met attributen binnen een relationele- of nosql-database. Uitgaande van 1 bronregister met gebeurtenissen kunnen meerdere op maat gemaakte raadpleegvoorzieningen worden gerealiseerd. Voorzieningen kunnen zowel de actuele als historische status van objecten bevatten.
- Bij gebruik van het CQRS-patroon kunnen te wijzigen en te raadplegen gegevens logisch of fysiek worden gescheiden. Logisch scheiden kan gebeuren door binnen 1 database gebruik te maken van verschillende schema's (bijv. door voor raadplegen gebruik te maken van views waarmee gegevens uit meerdere tabellen worden samengevoegd en eventueel getransformeerd). Fysiek scheiden kan door voor raadplegen een aparte opslagvoorziening te gebruiken die, continu of periodiek, wordt bijgewerkt na mutaties. In dit geval moet rekening worden gehouden met het feit dat geraadpleegde gegevens niet altijd gegarandeerd actueel zijn (wat bijv. lastig kan zijn bij het uitvoeren van transacties waarbij verschillende typen gegevens nodig zijn).
- Opmerkingen** - Bij het event sourcing patroon is al opgemerkt dat het belangrijk is om te zorgen voor voldoende ontkoppeling tussen event store eigenaar en afnemers. Het CQRS-patroon is hiervoor een bruikbaar en vaak toegepast patroon. Net als voor event-sourcing geldt dat het een verandering betekent qua ontwerp en realisatie van oplossingen ten opzichte van hoe dat nu meestal gebeurt.



8 GEGEVENS-FORMATEN

8.1 INLEIDING

Om de uitwisseling van informatie via een netwerk of communicatiesysteem goed te laten verlopen zijn verschillende (gegevens)formaten beschikbaar. Bij het inrichten van notificatieprocessen kan, ook nadat is gekozen voor een bepaald patroon, gebruik worden gemaakt van verschillende formaten. Afhankelijk van de keuze van een formaat kunnen bijvoorbeeld wel of niet metagegevens aan berichten worden toegevoegd (bijv. voor validatie of routing). Dit hoofdstuk beschrijft kort enkele voor notificeren gangbare formaten en voor- en nadelen daarvan.

8.2 JSON

JSON (JavaScript Object Notation is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays (or other serializable values). – Bron: [Wikipedia](#)

JSON is een gegevensformaat waarmee eenvoudig gegevens in gestructureerde vorm zijn te representeren. JSON is (ook) een geschikt formaat voor het opnemen van gegevens over plaatsgevonden gebeurtenissen binnen notificaties.

```
{ "werknemers": [
  { "voorNaam": "Jan", "achterNaam": "Jansen" },
  { "voorNaam": "Anna", "achterNaam": "Smits" },
  { "voorNaam": "Peter", "achterNaam": "Geurts" }
]}
```

Figuur 5: Voorbeeld-bestand in JSON-formaat

JSON wordt steeds vaker gebruikt voor gegevensuitwisseling bij nieuwe ontwikkelingen. Binnen de Nederlandse API Strategie is men voornemens om JSON te benoemen als het voorkeurformaat voor uitwisseling van gegevens via API's.

Een belangrijke reden waarom JSON steeds vaker wordt gebruikt in plaats van het 'rijkere' XML is dat het minder complex en daardoor eenvoudiger in gebruik is³⁷. Rijkere functionaliteit dan JSON basaal biedt kan worden gerealiseerd door gebruik te maken van [JSON-schema](#) voor beschrijving van structuur en gegevens of van [YAML](#) voor configuratiebeschrijvingen. JSON wordt breed ondersteund door allerlei programmeertalen en middleware-voorzieningen.

Vanwege de brede ondersteuning, relatieve eenvoud en voldoende functionaliteit zien we JSON als het voorkeurformaat om binnen events meta- of context-gegevens op te nemen (bijv. voor routeren en filteren).

Gebruik JSON voor beschrijving van context-gegevens binnen events

Welk formaat het meest geschikt is voor verstrekking van inhoudelijke gegevens over plaatsgevonden gebeurtenissen ('payload-gegevens') is contextafhankelijk. Er kan bijv. voor een tekst- of binair formaat worden gekozen, met of zonder gebruik van een schema.

Om dezelfde redenen als bij contextgegevens geldt ook voor inhoudelijke gegevens dat voor niet-binaire formaten geldt dat waar JSON voldoet het de voorkeur geniet.

³⁷ Er zijn meer redenen waarom JSON vaak de voorkeur krijgt, maar het doel is hier niet om JSON te vergelijken met XML.



Gebruik JSON als voor tekstuele inhoudelijke gegevens binnen events

Voor eenvoudige gevallen volstaat het vaak om als Producer de inhoud van gebeurtenisgegevens goed te beschrijven en beschikbaar te stellen voor Consumers. Naarmate uit te wisselen informatie complexer wordt gebruik van een schema wenselijker. Consumers kunnen daarmee gebeurtenis-gerelateerde gegevens beter interpreteren en geautomatiseerd verwerken.

Neem een verwijzing naar een JSON-schema op in events als interpretatie van gebeurtenisgegevens complex is

Opmerking: In de CloudEvents berichtspecificatie en het daarop gebaseerde NL GOV profile for CloudEvents wordt in voorbeelden meestal JSON als formaat gebruikt. Het is van belang te benadrukken dat naast JSON ook gebruik van andere gegevensformaten mogelijk is.

8.3 XML

Extensible Markup Language (XML) is een standaard van het World Wide Web Consortium voor de syntaxis van formele opmaaktalen waarmee men gestructureerde gegevens kan weergeven in de vorm van platte tekst. Deze presentatie is zowel machineleesbaar als leesbaar voor de mens. Het XML-formaat wordt gebruikt om gegevens op te slaan (zoals in het OpenDocument-formaat) en om gegevens over het internet te versturen. – Bron: [Wikipedia](#)

XML is een gegevensformaat waarmee gegevens nauwkeurig in gestructureerde vorm zijn te representeren. XML is daarmee (ook) bruikbaar als formaat voor notificeren.

```
<employees>
  <employee>
    <voorNaam>Jan</voorNaam>
    <achterNaam>Jansen</achterNaam>
  </employee>
  <employee>
    <voorNaam>Anna</voorNaam>
    <achterNaam>Smits</achterNaam>
  </employee>
  <employee>
    <voorNaam>Peter</voorNaam>
    <achterNaam>Geurts</achterNaam>
  </employee>
</werknemers>
```

Figuur 6: Voorbeeld-bestand in XML-formaat

XML wordt vaak gebruikt voor de uitwisseling van gegevens via internet. Er zijn honderden documentformaten die de XML-syntaxis gebruiken en het is de taal die gebruikt wordt binnen het SOAP-protocol waarmee webservices communiceren. Er zijn ook veel berichtstandaarden die op basis van XML zijn ontwikkeld (bijv. [HL7](#) en [StUF](#)).

Tegenwoordig wordt vaker gebruik gemaakt van JSON dan van XML. Redenen daarvoor zijn o.a. dat JSON (veel) eenvoudiger is in gebruik en tot compactere berichten leidt. In zijn algemeenheid geldt dat als gegevensstructuren voldoende lichtgewicht zijn het de voorkeur verdient om ze met JSON te beschrijven (eventueel voorzien van een schema, wat het lichtgewicht karakter al minder lichtgewicht maakt...).

In de praktijk zullen XML en daarop ontwikkelde formaten hun waarde blijven houden en gebruikt blijven worden binnen notificatieprocessen. Eventueel in combinatie met JSON, waarbij voor eenvoudige contextgegevens bijv. JSON wordt gebruikt en voor meer complexe gebeurtenisbeschrijving XML.



```
{
  "specversion" : "1.0",
  "type" : "een.event.type",
  "source" : "urn:uuid:123e4567-e89b-12d3-a456-426614174000",
  "subject" : "123",
  "id" : "123-456-789",
  "time" : "2021-04-05T17:31:00Z",
  "datacontenttype" : "text/xml",
  <data xsi:type="xs:string">{ "salutation": "Good Morning", "text": "hello world" }</data>
}
```

Figuur 7: Voorbeeld met JSON-contextdata en XML inhoudelijke gebeurtenisdata

8.4 CSV

Het CSV-formaat is een eenvoudig en oud databaseformaat. Het bestaat enkel uit tekstgegevens, waardoor het gemakkelijk geïmplementeerd (lezen en/of schrijven) kan worden en een brede verspreiding kent. Waarden worden in principe gescheiden door komma's, en regels door het nieuweregelteken. – Bron: [Wikipedia](#)

CSV is een al compact formaat om gegevens gestructureerd uit te wisselen. Het is echter zeer beperkt in het beschrijven van data en relaties. Er kunnen bijv. geen hiërarchische en andere soorten relaties tussen elementen mee worden beschreven.

CSV is minder flexibel dan bijv. JSON en XML en is om die reden minder geschikt voor event-driven werken en notificeren. De reden om het hier toch te noemen dat er binnen de Nederlandse overheid nog regelmatig gebruik wordt gemaakt van CSV-bestanden voor gegevensuitwisseling. We beschouwen CSV daarom niet als voorkeursformaat maar als een formaat dat in bepaalde situaties toch nuttig kan zijn. Bijv. als betrokken partijen niet in staat zijn om JSON (of XML) te gebruiken maar wel notificatieberichten in CSV-formaat kunnen produceren of verwerken.

```
voorNaam, Achternaam
Jan, Jansen
Anna, Smit
Peter, Geurts
```

Figuur 8: Voorbeeld-bestand in CSV-formaat

8.5 OVERIGE FORMATEN

We hebben ons hier beperkt tot enkele formaten die om verschillende redenen van belang zijn voor notificatieprocessen binnen de overheid. Er zijn uiteraard veel meer gegevensformaten die gebruikt kunnen worden bij event-driven werken en notificeren.

Het GOV NL profile for CloudEvents is een berichtstandaard die het gebruik van verschillende gegevensformaten toestaat. De CloudEvents-community heeft voor de meest gangbare formaten gespecificeerd hoe gebruik van de CloudEvents-specificatie daar moet plaatsvinden. Geadviseerd wordt om met name van deze gangbare formaten gebruik te maken bij het ontwerpen van nieuwe oplossingen. Zeker wanneer gebruik wordt gemaakt van de CloudEvents berichtspecificatie, of het voor de Nederlandse overheid ontwikkelde profiel daarvan. Hiermee wordt verdergaande standaardisatie gerealiseerd waarmee de interoperabiliteit van oplossingen wordt vergroot.

Gebruik de CloudEvents gegevensformaat-specificaties

Op dit moment, juni 2022, zijn een drietal formaat-specificaties vastgesteld (JSON, AVRO, Protobuf) en is een vierde (XML) in ontwikkeling. Voor de actuele stand van zaken zie: [CloudEvents Documents](#).



9 TRANSPORT-PROTOCOLLEN

9.1 INLEIDING

Om de uitwisseling van informatie via een netwerk of communicatiesysteem goed te laten verlopen zijn verschillende (transport)protocollen ontwikkeld. Bij het inrichten van notificatieprocessen kan, ook nadat is gekozen voor een bepaald patroon, gebruik worden gemaakt van verschillende protocollen. In aanvulling op de inhoudelijke berichttekst kan bijv. opslag-, routerings- en bezorgingsinformatie wordt toegevoegd vóór verzending en verwijderd van het bericht voordat het wordt afgeleverd bij de ontvangende applicatie.

Dit hoofdstuk beschrijft een aantal protocollen en de bruikbaarheid ervan voor notificeren.

9.2 PROTOCOLLEN

Een netwerkprotocol is een protocol, een afgesproken communicatiewijze, voor netwerkcomponenten. Door het toepassen van een standaard protocol, kunnen componenten van verschillende leveranciers met elkaar gegevens uitwisselen.

Een netwerkprotocol bevat afspraken omtrent:

- *Identificatie van de verschillende communicerende componenten.*
- *Het onderhandelen over het tot stand komen van de communicati.,*
- *De betekenis van de over en weer gezonden gegevens en informatieblokken.*
- *Het afbreken van de communicatiestroom.*

– bron: [Wikipedia](https://nl.wikipedia.org/wiki/Netwerkprotocol)

Voor het beschrijven van de uitwisseling tussen applicaties wordt vaak gebruik gemaakt van een lagenmodel (bijv. het OSI- of TCP/IP lagenmodel) om verschillende typen functionaliteit te beschrijven. Binnen iedere laag kunnen verschillende protocollen worden gebruikt. Dit hoofdstuk beperkt zich tot een korte beschrijving van een aantal veelgebruikte protocollen voor gegevensuitwisseling binnen de ‘applicatielaag’³⁸. Doel daarvan is om bij het inrichten van notificatieprocessen op basis van de aanwezige eisen te kunnen bepalen welk protocol, of protocollen, het geschiktst zijn om te gebruiken. Binnen protocollen is onderscheid te maken in standaardprotocollen (bijvoorbeeld HTTP, AMQP, MQTT, SMTP), open source protocollen (bijvoorbeeld Kafka, NATS) en platform- en leverancier-specifieke protocollen (AWS Kinesis, Azure Event Grid).

Een belangrijk aspect is welke mogelijkheden een protocol kent om beschrijvende metadata en inhoudelijke payload-gegevens op te nemen. Onderstaande tabel toont in de eerste kolom een aantal ‘applicatielaag’ protocollen. De tweede kolom toont een aantal protocollen of standaarden die voortbouwen op het vaak gebruikte HTTP-protocol. Niet alle weergegeven protocollen zijn even geschikt voor notificatieprocessen zoals die binnen de scope van project Notificatieservices vallen. Het feit dat sommige protocollen gedurende langere tijd 2-wegverkeer kunnen ondersteunen is daarbij bijv. vaak niet nodig als gebruik wordt gemaakt van functioneel asynchrone berichtuitwisseling. Bij andere type notificatieprocessen, die mogelijk in het vervolg op dit project aandacht gaan krijgen, kan dat type functionaliteit echter wel van belang zijn.

³⁸Binnen het OSI-lagenmodel de ‘Applicatielaag’ binnen het TCP/IP lagenmodel de ‘Applicatie of Proces laag’



Protocol	'Standaard'	Opmerking
HTTP(S) ³⁹		Het Hypertext Transfer Protocol (HTTP) is een applicatielaag protocol voor gedistribueerde, collaboratieve hypermedia-informatiesystemen en de basis van datacommunicatie voor het World Wide Web.
	HTTP Polling	Bij HTTP polling wacht de cliënt op antwoord van de server om nieuwe informatie vragen via Short Polling (weinig gebruikt), Long Polling (vraag en blijf wachten) of Periodic Polling (om de zoveel tijd vraag herhalen).
	HTTP Streaming / Server-Sent Events	Bij HTTP Streaming is er een langdurige verbinding waarbij de server na het optreden van events gegevens pusht totdat de cliënt de verbinding verbreekt. Server-Sent Events is een (in ontwikkeling zijnde) standaard hiervoor.
	Webhooks	Door een afnemer gedefinieerde HTTP callback die een aanbieder gebruikt om via een HTTP POST methode notificatieberichten af te leveren.
	Digikoppeling RESTful API profiel	REST is een architectuurstijl die een aantal algemene eisen stelt aan hoe communicatie tussen cliënt en server plaatsvindt. Er zijn veel manieren om, min of meer, volgens de REST eisen te communiceren. Digikoppeling RESTful API profiel , dat is gebaseerd op de REST-API Design Rules standaard, is een standaard die meer uitgewerkt RESTful uitwisselen specificeert.
	SOAP/WS-ReliableMessaging	RPC-Protocol waarmee SOAP berichten met hoge zekerheid, ook bij storingen, worden uitgewisseld tussen gedistribueerde applicaties. Communicatieprotocol-neutraal maar maakt meestal gebruik van HTTP. Een van de Digikoppeling koppelvakstandaarden .
	SOAP/ebMS	Op SOAP gebaseerd pakket-, routerings- en transportprotocol voor uitwisselen van op ebXML gebaseerde B2B-documenten. Communicatieprotocol-neutraal maar maakt meestal gebruik van HTTP. Een van de Digikoppeling koppelvakstandaarden .
	WebSockets	Het WebSocket protocol upgrade een standaard HTTP request/response naar een efficiënte permanente full-duplexverbinding waarmee, zo lang als een sessie duurt, cliënt en server elkaar op elk moment berichten kunnen sturen. In tegenstelling tot HTTP biedt WebSocket full-duplex communicatie.
AMQP		Het Advanced Message Queuing Protocol is een open standaard messaging protocol dat is ontworpen om een breed scala aan berichtentoepassingen en communicatiepatronen efficiënt te ondersteunen.
MQTT		Message Queue Telemetry Transport (MQTT) is ontworpen als lichtgewicht berichtenprotocol voor gebruik via het Publish-Subscribe patroon. Het wordt vaak gebruikt binnen IOT-omgevingen waarin, ook hele kleine, apparaten zoals sensoren gebeurtenissen signaleren en in de rol van aanbieder informatie daarover verstrekken.
(S)FTP		File Transfer Protocol om bestanden uit te wisselen via een op TCP/IP gebaseerd netwerk zoals het internet. M.b.v. het TLS protocol is versleuteling

³⁹De hyperlinks in de kolom 'Protocol' verwijzen allemaal naar Wikipedia.



Protocol	'Standaard'	Opmerking
		en veilige identificatie van de server mogelijk. FTP kan bijv. worden gebruikt om notificaties in de vorm van een tekstbestanden op door de afnemers opgegeven locatie te plaatsen.
SMTP(S)		Simple Mail Transfer Protocol voor verzenden en ontvangen van email. Hoewel het protocol zich meer leent voor menselijke afnemers van notificaties (buiten scope van het project) kan het ook worden gebruikt om notificaties via mailberichten met bijlages te versturen die geautomatiseerd kunnen worden verwerkt.
Platform-specifieke protocollen		Met name binnen grote Cloud omgevingen wordt vaak gebruik gemaakt van eigen protocollen. De voordelen van gebruik ervan kunnen opwegen tegen het nadeel van lock-in die ontstaat dus kan het binnen bepaalde situaties wenselijk zijn om ook gebruik van dit type protocollen te overwegen.

De volgende paragrafen lichten een aantal protocollen toe die binnen de context van project Notificatieservices een belangrijke rol kunnen spelen.

9.3 HTTP

HTTP is een synchroon request/reply protocol dat de basis van datacommunicatie voor het World Wide Web vormt. In documenten opgenomen hyperlinks verwijzen daarbij naar andere informatiebronnen. Binnen de Nederlandse overheid wordt het HTTP-protocol intensief gebruikt voor het synchroon opvragen van gegevens. Deze paragraaf beschrijft een aantal standaarden die HTTP als basis gebruiken die bruikbaar kunnen zijn voor het realiseren van notificatieoplossingen.

9.3.1 WEBHOOK

Webhooks are "user-defined HTTP callbacks". They are usually triggered by some event, such as pushing code to a repository or a comment being posted to a blog. When that event occurs, the source site makes an HTTP request to the URL configured for the webhook. Users can configure them to cause events on one site to invoke behavior on another. ... The format is usually [JSON](#). The request is done as an [HTTP POST](#) request. – bron: [Wikipedia](#)

Webhooks zijn een middel waarmee Publishers via een HTTP POST-methode notificatieberichten bij Consumers afleveren. Het is een relatief eenvoudig patroon om via een push-mechanisme notificaties te verstrekken en wordt in de praktijk vaak toegepast⁴⁰. Webhooks maken gebruik van het HTTP-protocol en de daar gebruikelijke methoden voor het uitvoeren van een aantal typen operaties. Voor meer informatie over het webhook-patroon zie [Webhook](#).

9.3.2 DIGIKOPPELING RESTFUL API PROFIEL

Aansluitend op de trend binnen de overheid om steeds vaker gebruik te maken van de REST-stijl voor gegevensuitwisseling tussen applicaties is de Digikoppeling standaard uitgebreid met een [RESTful API profiel](#) dat zich o.a. volledig conformeert aan alle (normatieve) eisen van de [REST API Design Rules](#) standaard.

⁴⁰Zie bijv. [100 Webhook Implementations](#) waarin 100 omgevingen worden beschreven waarin gebruik wordt gemaakt van webhooks.



- Het Digikoppeling Rest API profiel is gericht op Machine-to-Machine (M2M) interactie via een gesloten dienst tussen overheidspartijen volgens de algemene uitgangspunten van de Digikoppeling standaard en is wat betreft functionele toepassing vergelijkbaar met het Digikoppeling WUS profiel.
- Het Digikoppeling REST-API profiel conformeert zich volledig aan het normatieve deel van de API Design Rules standaard. Hierin zijn eisen beschreven met betrekking een aantal aspecten die een rol spelen bij de ontwikkeling en gebruiken van services.
- Een aantal regels uit de API Design Rules extensies zijn verplicht of aanbevolen. Hierbij gaat het met name over beveiliging, autorisatie en foutafhandeling.
- In versie 1.0 van het Digikoppeling REST API profiel wordt signing en encryptie niet ondersteund. In toekomstige versies van het profiel zal hier wel invulling aan worden gegeven.

Gebruik van dit profiel bij notificatieprocessen die gebruik maken van de REST-stijl leidt op een aantal aspecten tot standaardisatie. In bepaalde situaties is gebruik van de Digikoppeling standaard verplicht.

Digikoppeling moet worden toegepast op alle digitale gegevensuitwisseling met behulp van gestructureerde berichten die plaatsvindt met voorzieningen die onderdeel zijn van de GDI, waaronder de basisregistraties, of die sector-overstijgend is. Geautomatiseerde gegevensuitwisseling tussen informatiesystemen op basis van NEN3610 is uitgesloten van het functioneel toepassingsgebied. – bron: [Lijst open standaarden – Forum Standaardisatie](#)

9.3.3 SOAP

Bij gebruik van het Simple Object Access Protocol stuurt een client een bericht naar een webservice van een server om daar een bepaalde procedure te laten uitvoeren ('Remote Procedure Call'). Na uitvoering daarvan stuurt de server een antwoordbericht terug naar de client. Berichten worden vrijwel altijd in XML-formaat via HTTP verstuurd⁴¹. Als hoge betrouwbaarheid van communicatie nodig is kan gebruik worden gemaakt van extensies zoals [WS-Security](#) en [WS-ReliableMessaging](#) waarmee een aantal bezorg-garanties zijn te bieden (AtLeastOnce, AtMostOnce, ExactlyOnce, InOrder). Een andere optie is om gebruik te maken van [ebMS](#) dat onder andere onderdeel uitmaakt van de [Digikoppeling standaard](#) voor het asynchroon uitwisselen van berichten.

SOAP is lang het meest gebruikte protocol geweest voor webservices (en wordt nog steeds veel gebruikt). Tegenwoordig wordt echter meestal de voorkeur gegeven aan services die volgens de REST-stijl werken. Ook als het gaat om notificeren. Een belangrijke reden voor gebruik van REST in plaats van SOAP is dat het in een aantal opzichten minder complex is en breed wordt ondersteund. Hoewel gebruik van REST-stijl services in veel situaties voldoet is het met name in situaties waarin hoge betrouwbaarheid is vereist raadzaam om gebruik van andere protocollen, zoals SOAP (evt. met extensies) te overwegen. Daarmee kan worden voorkomen dat bij gebruik van eenvoudiger protocollen waar benodigde functionaliteit ontbreekt binnen applicaties aanvullend maatwerk moet worden ontwikkeld (dat niet eenvoudig is en vaak niet leidt tot hetzelfde kwaliteitsniveau).

9.3.4 WEBSOCKET

Een WebSocket is een standaard bi-directionele TCP-socket tussen de cliënt en de server. De socket begint als een HTTP-verbinding vanuit de client en wordt vervolgens na een HTTP-handshake opgewaardeerd naar een TCP-socket. Daarna kunnen beide partijen elkaar efficiënt realtime berichten verzenden. WebSocket is ontworpen om te werken via de standaard HTTP-poorten 443 en 80 en met HTTP-firewalls en proxy's. Hoewel het protocol met name bedoeld is voor communicatie tussen webbrowsers en servers (bijv. voor mediastreaming) is het als lichtgewicht cross-platform bruikbaar wanneer er een client-server-relatie is.

⁴¹ Tegenwoordig kan SOAP ook gebruikt worden in combinatie met SMTP, FTP of JMS maar dit gebeurt zelden.



9.3.5 SERVERS-SENT EVENTS

Server-Sent Events is een uni-directioneel server push-technologie waarmee een client automatische updates van een server kan ontvangen via een HTTP-verbinding. Het protocol wordt vaak gebruikt om berichtupdates of continue gegevensstromen naar een browser te verzenden nadat de gebruiker een bepaalde URL heeft opgevraagd.

9.4 AMQP

Het Advanced Message Queuing Protocol (AMQP) is een open standaard applicatielaag-protocol (net als bijv. HTTP) ter ondersteuning van messaging. Het is ontworpen om een breed scala aan berichttoepassingen en communicatiepatronen flexibel en efficiënt te ondersteunen. Het biedt o.a. geavanceerde functionaliteit voor berichtuitwisseling⁴², authenticatie, encryptie en garanties voor berichtaflevering⁴³.

Deze paragraaf beschrijft meer in detail dan bij andere protocollen wat AMQP aan mogelijkheden biedt. Doel daarvan is om te verduidelijken dat berichtuitwisseling, ook bij notificeren, allerlei soorten eisen kan stellen waar protocollen in verschillende mate in kunnen voorzien. Een andere reden is dat AMQP binnen de overheid nog weinig wordt gebruikt maar bij een toename van event-driven werken als bewezen standaard een grotere rol zou kunnen gaan spelen.

AMQP is een binair protocol dat interoperabiliteit bevordert omdat het, zolang maar wordt voldaan aan de specificaties, door allerlei soorten applicaties gebruikt kan worden. Het is bruikbaar in heel verschillende situaties (van heel kleine schaal IOT-messaging tot hyper-scale Cloud messaging) en bij toepassing van verschillende patronen (bijv. peer-to-peer, request-reply, Publish-Subscribe). AMQP wordt ondersteund door veel berichtgeoriënteerde middlewareproducten om bijv. queing, routing, betrouwbaarheid en beveiliging te ondersteunen.

Het protocol kent een gelaagde architectuur:

- Een typesysteem
- De transport laag; een efficiënt asynchroon, binair, peer-to-peer protocol voor transport van berichten over een netwerk.
- Een gestandaardiseerd uitbreidbaar berichtformaat.
- Een set gestandaardiseerde, en uitbreidbare, messaging capabilities.

Binnen AMQP worden connecties opgezet waarbinnen meerdere sessies mogelijk zijn. Binnen een sessie kunnen meerdere 'links' (verbindingen) worden gecreëerd om berichten uit te wisselen. Zowel op sessie- als link-niveau zijn limieten in te stellen (bijv. hoe groot berichten en throughput mogen zijn en hoeveel berichten maximaal in- en uitgaand zijn te verwerken). Hiermee kan een robuuste verwerkingsomgeving worden gerealiseerd die voorkomt dat er tijdens berichtuitwisseling fouten optreden als gevolg van onvoldoende systeem- of applicatie-resources⁴⁴.

Berichten kunnen via een eigen link in 1 richting worden verzonden terwijl status- en flowcontrol-informatie via een aparte link in 2 richtingen wordt uitgewisseld. Wanneer 2-weg berichtenverkeer nodig is kunnen 2 aparte links worden aangemaakt. Met behulp van een bericht- of correlatie-id kunnen dan bijv. full-duplex of request-response uitwisselpatronen worden ondersteund.

⁴²Berichten kunnen bijv. naar afnemers worden verstuurd zonder vraag om respons maar als wel een respons nodig is kan een afnemer na verwerking van een ontvangen bericht bijv. aangeven of het bericht is 'geaccepteerd', 'afgewezen' of dat het niet te verwerken was.

⁴³Mogelijke aflevergaranties zijn dat een bericht hoogstens 1 keer wordt afgeleverd ('At-Most-Once'), minstens 1 keer wordt afgeleverd ('At-Least-Once') of precies 1 keer ('Exactly-Once') wordt afgeleverd.

⁴⁴Bedenk hierbij dat AMQP zowel gebruikt kan worden door een klein IOT-device als binnen een 'hyper-scale' cloud-serveromgeving die miljoenen berichten moet kunnen verwerken.



AMQP ondersteunt verschillende soorten berichtaflevering met een aantal garanties:

- 1 een bericht wordt hoogstens 1 keer afgeleverd ('At-Most-Once') ('Fire-and-Forget') of
- 2 minstens 1 keer afgeleverd ('At-Least-Once')⁴⁵ of
- 3 precies 1 keer ('Exactly-Once') afgeleverd.

Als sessies door storingen worden beëindigd kunnen eerder aangemaakte links worden hersteld. Daarbij is fijnmazig per bericht, afhankelijk van de status, aan te geven wat van de andere partij verwacht wordt. Op deze manier kan een onverwacht onderbroken communicatieproces worden hervat vanaf het moment waarop de verstoring optrad.

Enkele kenmerken van AMQP:

- Het kent een eigen typesysteem dat zowel schema-bound als schema-free is te gebruiken. Naast verplicht gebruik voor stuurgegevens in berichten kan het bijv. ook worden gebruikt om payload-gegevens in een gegevensformaat met beperkte functionaliteit om te zetten in AMQP gegevenstypes (JSON kent bijv. beperkte functionaliteit voor numerieke gegevens en timestamps).
- Er zijn veel mogelijkheden om notificatieprocessen flexibel en betrouwbaar in te richten. Een voorbeeld daarvan is de mogelijkheid voor afnemende applicaties om, bij de start of tussentijds, door te geven hoeveel berichten zij binnen een bepaald tijdsbestek maximaal willen ontvangen of om tussentijds een (tijdelijke) stop in te lassen. Beide zijn voorbeelden van de vele mogelijkheden aan 'flow control' dat AMQP biedt. AMQP biedt heel veel mogelijkheden om partijen met elkaar te laten communiceren en berichtuitwisseling fijnmazig 'op maat' in te richten.
- Het is ontwikkeld binnen de financiële sector, was eerder een OASIS-standaard maar is sinds 2014 ook een [ISO/IEC standaard](https://www.iso.org/standard/68811.html). Documentatie er over is te vinden op amqp.org.

Zoals eerder vermeld is AMQP als protocol interessant omdat het op een aantal aspecten ondersteuning biedt waar behoefte aan kan zijn, of ontstaan, bij event-driven werken en uitvoeren van notificatieprocessen.

9.5 FTP

Het [File Transfer Protocol](https://tools.ietf.org/html/rfc959) is een al 50 jaar bestaand protocol om bestanden uit te wisselen. Eerder is beschreven dat notificeren via bestandsuitwisseling niet de voorkeur verdient. In sommige situaties kan FTP echter wel degelijk bruikbaar zijn of zelfs de enige beschikbare optie zijn.

Project Notificatieservices heeft als doel om event-driven werken en notificeren breed binnen de overheid te stimuleren en door middel van standaardisatie te vergemakkelijken. Dit betekent dat rekening moet worden gehouden met partijen die beperkt zijn in hun mogelijkheden. FTP kan in bepaalde situaties een protocol zijn waarmee notificaties zijn te verstrekken of (bij informatiearm notificeren) aanvullende informatie wordt opgehaald.

Bij gebruik van FTP kan, net als bij het hierna besproken SMTP, met enige creativiteit ook gebruik worden gemaakt van het NL GOV profile for CloudEvents. Onder andere door binnen uit te wisselen gegevens de binnen het profiel benoemde attributen met metagegevens op te nemen zodat routing, filtering en interpretatie door Consumers op een vergelijkbare manier kan plaatsvinden als bij gebruik van gangbare protocollen zoals HTTP.

⁴⁵ De ontvanger moet hierbij feedbackinformatie geven aan de verzender over het ontvangen bericht: accepted (OK), rejected (met evt. toelichtende informatie), released (een verzoek om opnieuw verzenden van het bericht omdat er is iets misgegaan bij verwerking (bijv. een time-out)), modified (released maar met informatie die aangeeft wat er veranderd moet worden aan het bericht voor opnieuw verzenden).



9.6 SMTP

Simple Mail Transfer Protocol (SMTP) is het standaard protocol om email te verzenden en ontvangen. Net als voor FTP geldt dat gebruik ervan voor notificeren niet de voorkeur verdient maar het in bepaalde situaties wel bruikbaar kan zijn.

Gebruik van email richting mensen valt buiten de scope van project Notificatieservices omdat notificeren van mensen hele andere eisen stelt dan notificeren van applicaties. Met de nodige maatregelen kan email worden gebruikt om communicatie tussen applicaties plaats te laten vinden. Bijvoorbeeld door in berichten of in bijlages gestructureerde notificatiegegevens op te nemen (bijv. In JSON- of XML-formaat). Door afnemers ontvangen e-mails kunnen dan bijv. via een Polling mechanisme automatisch worden verwerkt. Is er sprake van vertrouwelijke gegevens in notificaties dan biedt gebruik van standaard email te weinig veiligheid en moet gebruik worden gemaakt van 'veilige email'⁴⁶.

9.7 PROTOCOLLEN EN BETROUWBAARHEID

Bij communicatie tussen applicaties is het van belang dat berichten van een Producer door Consumers met een voldoende mate van zekerheid worden ontvangen. Afhankelijk van de context kan de gewenste mate van zekerheid variëren van 'geen zekerheid nodig' tot 'volledige zekerheid vereist'.

In geval van 'geen zekerheid' volstaat een weinig betrouwbaar uitwisselmechanisme en hoeft een Producer geen actie te ondernemen als een notificatie niet aan een Consumer is te verstrekken. Wanneer volledige zekerheid nodig is gebruik van daarvoor geschikte standaarden en voorzieningen nodig. Van een Producer wordt verwacht dat hij controleert of een notificatie volgens afspraak door een Consumer is ontvangen. Is dat niet het geval dan moet hij passende actie ondernemen (bijv. door de notificatie opnieuw aan te bieden en na meerdere mislukte pogingen de afnemer daarover te informeren).

Protocollen zijn in verschillende mate geschikt om zekerheid van ontvangst te kunnen ondersteunen. Onderstaande tabel toont grofweg welke mate van zekerheid gebruik van een aantal bekende protocollen biedt.

Protocol	Mate van zekerheid	Opmerking
HTTP(S)	Matig tot Hoog	HTTP biedt zelf weinig mechanismen om zekerheid te garanderen. Met aanvullende maatregelen binnen de applicatielaag op te waarderen naar Hoog. Ondersteunt synchrone communicatie.
SMTP	Hoog	Ondersteunt asynchrone communicatie.
SOAP/WS-ReliableMessaging	Hoog	Protocol waarmee SOAP-berichten met hoge zekerheid, ook bij storingen, worden uitgewisseld tussen gedistribueerde applicaties. Maakt meestal gebruik van HTTP maar kan ook SMTP gebruiken. Ondersteunt synchrone en asynchrone communicatie.
SOAP/ebMS	Hoog	Op SOAP gebaseerd pakket-, routerings- en transportprotocol. Open standaard en als zodanig communicatieprotocol-neutraal. Maakt meestal gebruik van HTTP en kan ook SMTP gebruiken. Geschikt voor uitwisselen van op ebXML gebaseerde B2B-documenten. Ondersteunt synchrone en asynchrone communicatie.

⁴⁶Er zijn in de praktijk verschillende manieren waarop communicatie via email veilig kan verlopen. Beschrijving daarvan valt buiten de scope van dit project.



Protocol	Mate van zekerheid	Opmerking
		Een van de Digikoppeling koppelvlakstandaarden .
AMQP	Hoog	Het Advanced Message Queuing Protocol is een open standaardprotocol dat werkt op de applicatielaag, waardoor berichtoriëntatie, wachtrijen en routing mogelijk zijn. Het maakt interoperabiliteit, betrouwbaarheid en beveiliging mogelijk bij het verzenden en ontvangen van berichten tussen applicaties.
(S)FTP		Standaard netwerkprotocol om bestanden uit te wisselen via een op TCP/IP gebaseerd netwerk zoals internet. M.b.v. het TLS-protocol is versleuteling en veilige identificatie van de server mogelijk.
SMTP(S)		Protocol voor verzending en ontvangst van email.
HTTP/REST	Matig	Op te waarderen via applicatielogica bij zowel aanbieder als afnemers (zie toelichting hierna).

Protocollen die een hoge zekerheid van ontvangst kunnen bieden kunnen bijvoorbeeld garanderen dat een verzonden bericht minstens, hooguit of precies 1 keer wordt ontvangen en dat berichten altijd in volgorde van verzending aankomen. Wanneer zich fouten voordoen (bijv. wanneer een bericht niet is af te leveren of er geen 'bericht van ontvangst' ('acknowledgement') wordt ontvangen) kan automatisch passende actie worden ondernomen (bijv. het op een later moment opnieuw versturen van het bericht of het plaatsen van een bericht in een 'dead letter queue').

HTTP is een veel gebruikt protocol voor communicatie maar biedt als protocol geen zekerheid van aflevering. Ook gebruik van de REST-architectuurstijl levert geen extra functionaliteit voor zekerheid van ontvangst op. Protocollen zoals SOAP/WS-ReliableMessaging en Soap/ebMS leveren, bovenop HTTP, zulke functionaliteit wel zodat ontvangst met hoge zekerheid is te garanderen en dat is te garanderen dat een verzonden bericht minstens, hooguit of precies 1 keer is ontvangen, berichten in volgorde van verzending zijn aankomen en vaker versturen van een bericht via niet-idempotente methodes (bijv. POST) niet leidt tot ongewenste effecten. Voor dit type functionaliteit moeten speciale maatregelen worden genomen bij zowel Producer als Consumer. Bijv. door in notificaties een uniek identificerend gegeven of volgnummer op te nemen dat Consumers kunnen gebruiken om passend te reageren in fout- of uitzonderingssituaties.

Wanneer zekerheid van ontvangst belangrijk is is te overwegen om een hiervoor geschikt protocol te gebruiken (bijv. SOAP/WS-ReliableMessaging, SOAP/ebMS of AMQP). De Stelseldienst Digikoppeling omvat de ebMS-standaard die meestal wordt gebruikt voor asynchroon berichtenverkeer ('meldingen') waarbij niet direct een antwoord is te geven. De ontvanger krijgt dan een bevestiging dat zijn bericht ontvangen is en kan later eventueel een antwoord ontvangen. Applicaties identificeren zich bij gegevensuitwisseling met Digikoppeling met een speciaal PKI-overheid-certificaat. Digikoppeling is daarmee geschikt voor veilig en betrouwbaar verzenden van berichten.

Uiteraard geldt dat binnen notificatieprocessen ook meerdere protocollen gebruikt kunnen worden. Voor de communicatie tussen een Producer en een Intermediary kan bijvoorbeeld gebruik worden gemaakt van een betrouwbaar protocol zoals ebMS of AMQP terwijl een Intermediary notificaties aan Consumers verstrekt via HTTP.



10 STANDAARDEN

10.1 INLEIDING

Net zoals op andere gebieden geldt ook bij event-driven werken en notificeren dat standaardisatie productontwikkeling kan vereenvoudigen en versnellen en bijdraagt aan compatibiliteit en interoperabiliteit. Een uitgangspunt bij het streven naar standaardisatie van notificeren binnen de Nederlandse overheid is om maximaal gebruik te maken van wat er internationaal op dit vlak gebeurt.

Dit hoofdstuk beschrijft drie internationale standaarden die bij kunnen dragen aan standaardisatie. De drie standaarden hebben een verschillend karakter en eigen doelen:

- 1 De CloudEvents berichtstandaard, uitgebreid met een aantal eigen extensies, specificeert hoe gebeurtenisberichten en notificaties eenduidig zijn te beschrijven.
- 2 De WebSub interactiestandaard beschrijft hoe bij gebruik van het HTTP-protocol berichten gestandaardiseerd zijn uit te wisselen.
- 3 De AsyncAPI beschrijvingsstandaard specificeert een standaardbeschrijving voor diensten met API's die volgens het Publish-Subscribe patroon gegevens uitwisselen.

Onderstaande tabel toont een aantal kenmerken per standaard waaruit zichtbaar wordt dat het om verschillende type standaarden gaat.

	Bericht inhoud	Interactie	Be-schrijvend	Inter-nationaal	Multi protocol	Tooling beschikbaar	Brede adoptie	In ont-wikkeling
CloudEvents	X			X	X	X	X	x
WebSub		X		X				
AsyncAPI			X	X	X	X	X	x

10.2 CLOUDEVENTS BERICHTSTANDAARD

De CloudEvents berichtstandaard is ontwikkeld vanuit de behoefte om bij gebruik van verschillende Cloud voorzieningen op dezelfde manier informatie over gebeurtenissen uit te kunnen wisselen. De standaard is ontwikkeld door de Serverless Working Group van de [Cloud Native Computing Foundation](#) waarin vrijwel alle grote Cloud leveranciers zijn vertegenwoordigd (o.a. Amazon, Microsoft, Google, Alibaba). Hoewel de standaard pas enkele jaren bestaat wordt hij al vrij breed ondersteund en zijn er ook al een aantal SDK's ontwikkeld voor gebruik met de meest gangbare programmeertalen.

Op basis van de CloudEvents standaard is het [GOV NL profile for CloudEvents](#) ontwikkeld. Daarin zijn specifiek voor de Nederlandse overheid een aantal aanvullende afspraken beschreven. Het volgende hoofdstuk beschrijft CloudEvents en het Nederlandse profiel in detail.

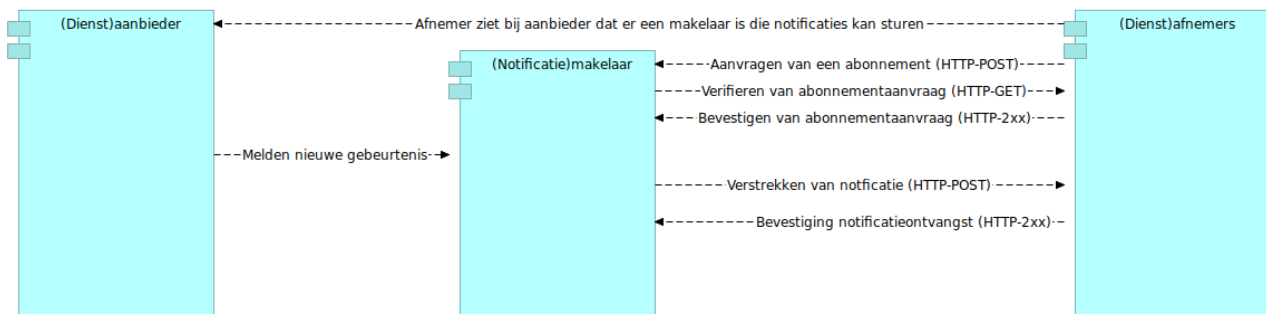
10.3 WEBSUB INTERACTIESTANDAARD

WebSub is een op HTTP gebaseerde standaard voor op het Publish-Subscribe patroon gebaseerde communicatie tussen gedistribueerde aanbieders, intermediairs en afnemers en sinds 2018 een W3C aanbeveling.

WebSub provides a common mechanism for communication between Publishers of any kind of Web content and their subscribers, based on HTTP web hooks. Subscription requests are relayed through hubs, which validate and verify the request. Hubs then distribute new and updated content to subscribers when it becomes available. – bron: [WebSub W3C Recommendation](#)

WebSub standaardiseert op het gebruik van HTTP als protocol met een aantal afspraken over hoe de interactie tussen aanbieders, intermediair en afnemers plaatsvindt. De 'happy flow' ziet er globaal als volgt uit:

- Afnemers kunnen bij een aanbieder zien dat er een of meer intermediair(s) zijn die notificaties kan/kunnen verstrekken als de aanbieder nieuws heeft over een bepaald onderwerp (via een '<link rel="hub" href="https://hub.example.com/">' in headerinfo van een webpagina).
- Om zich te abonneren (of een abonnement op te zeggen) op notificaties naar aanleiding van bepaalde gebeurtenissen doen afnemers een HTTP POST request naar een genoemde intermediair:
 - De afnemer verstrekt daarbij het onderwerp (via een URL) waarop hij zich wil abonneren, het internet(callback)adres van de service waarmee hij nieuwe notificaties wil ontvangen ('webhook') en eventueel een geheime code (en evt. een periode waarvoor het abonnement moet gelden);
 - De intermediair stuurt via een HTTP GET request een verificatiebericht naar het opgegeven (callback)adres met o.a. de opgegeven onderwerp-URL en een random tekenreeks;
 - De afnemer stuurt een positief antwoord met een HTTP-2xx code met in de body van het bericht de ontvangen random tekenreeks;
 - De intermediair accepteert het abonnementsverzoek.
- Aanbieders stellen hun intermediair(s) op een overeengekomen manier op de hoogte wanneer ze nieuws hebben. Hoe ze dit doen is geen onderdeel van de standaard. (Sommige hubs vragen bijv. om een HTTP POST request met als parameters hub.mode="publish" and hub.url=(URL van een nieuwe resource).
- De hub stuurt notificaties naar alle geabonneerde afnemers via een POST request naar de door een afnemer opgegeven callback-URL; als een afnemer bij abonnementsaanvraag een geheime code heeft opgegeven zal de intermediair notificaties daarmee digitaal ondertekenen.
- De afnemer bevestigt ontvangst van een notificatie via een HTTP-2xx om te laten weten dat de notificatie succesvol is ontvangen.



Gebruik van de WebSub standaard zorgt ervoor dat de interactie tussen actoren op een aantal aspecten gestandaardiseerd verloopt (bijv. protocol, wijze van nemen en opzeggen van abonnementen en borgen van bepaalde mate van betrouwbaarheid) waardoor er minder of zelfs geen 'maatwerkafspraken' nodig zijn.

De standaard heeft een lichtgewicht karakter en is van oorsprong ontwikkeld voor het kunnen abonneren op en pushen van berichten van nieuwssites en blogs (bijv. CNN en Blogger) naar applicaties zoals Feedly of Flipboard. Er zijn enkele intermediairs (bijv. Publish-Subscribehubbub.appspot.com en Publish-Subscribehubbub.superfeedr.com) maar de status daarvan is niet altijd even duidelijk. WebSub is ondertussen sinds 2018 een aanbevolen W3C-standaard maar lijkt nog niet breed omarmd te worden.

Gelet op de context van project Notificatieservices lijkt de standaard beperkt bruikbaar. Binnen bepaalde situaties kan de standaard, of onderdelen daarvan, bruikbaar zijn. Bijv. als het gaat om het publiceren van informatierijke events met openbare gegevens of als er behoefte is aan een laagdrempelig mechanisme om geautomatiseerd te kunnen abonneren.



Publish-Subscribe: zie of de WebSub standaard of delen daarvan zijn toe te passen in situaties waarin laagdrempelig notificeren volstaat

Een interessante ontwikkeling is hoe bij het ontwikkelen van het [Modular Open Source Identity Platform \(MOSIP\)](#) gebruik wordt gemaakt van WebSub om events te verwerken. Daarbij wordt onder andere gebruik gemaakt van [WebSubHub](#): open source software die bij gebruik van WebSub als Hub kan fungeren. Voor meer informatie hierover zie: <https://thenewstack.io/how-mosip-uses-ballerina-websubhub-for-event-driven-integration/>.

10.4 ASYNCAPI BESCHRIJVINGSSTANDAARD

AsyncAPI is een standaard om effectief met event-driven APIs te kunnen werken⁴⁷. Vergelijkbaar met de OpenAPI standaard voor REST APIs maakt de AsyncAPI standaard het onder andere mogelijk om betere documentatie te maken, geautomatiseerd broncode te genereren, gegevensvalidatie te doen en in zijn algemeenheid systeemontwikkeling te vereenvoudigen en versnellen. In maart 2021 is het initiatief onderdeel geworden van de [Linux Foundation](#).

Gelet op de veelheid aan betrokken applicaties en technologische diversiteit bij event-driven werken is standaardisatie van interfacing cruciaal. De relatief jonge AsyncAPI standaard kan daarbij een belangrijke rol spelen omdat hij generiek van aard en breed toepasbaar is, breed wordt ondersteund en het gebruik ervan wereldwijd snel toeneemt⁴⁸.

De standaard is bedoeld voor het event-driven gegevens uitwisselen tussen applicaties, meestal met gebruik van het Publish-Subscribe patroon. Gelet op het doel van project Notificatieservices om event-driven werken en notificeren binnen de overheid te standaardiseren kan de AsyncAPI standaard hierbij behulpzaam zijn⁴⁹.

De standaard specificeert onder andere de beschrijving van berichtinhoud (opbouw en attributen), locatie van applicaties, te gebruiken protocollen (bijv. AMQP, HTTP, MQTT, Kafka, WebSockets) en authenticatie- en autorisatiemechanismen (bijv. gebruikersnaam/wachtwoord, certificaten, API-keys, OAuth2).

De standaard is zowel bruikbaar in situaties waarin 1 applicatie zowel de rol van aanbieder als intermediair invult ('client-server model') als in situaties waarin dit gebeurt door verschillende applicaties ('broker-centric').

Het feit dat de standaard laagdrempelig is toe te passen, breed wordt ondersteund, steeds meer ondersteunende software als open source beschikbaar is en geautomatiseerd genereren van documentatie en code mogelijk maakt zijn redenen om van de standaard gebruik te maken.

Overweeg de AsyncAPI-standaard te gebruiken voor het beschrijven van event-driven APIs

De huidige 2.4.0 versie (juli 2022) van de AsyncAPI specificatie is te vinden op: <https://www.asyncapi.com/docs/specifications/v2.4.0>. www.asyncapi.com bevat naast een algemene beschrijving ook tutorials, voorbeelden en verwijzingen naar beschikbare tooling

⁴⁷De naam 'AsyncAPI' lijkt aan te geven dat het alleen gaat om asynchrone berichtuitwisseling. Belangrijkste doel van de standaard is echter om API's geschikt te maken om gebeurtenisgedreven te werken wat veelal synchroon verloopt maar niet daartoe beperkt hoeft te zijn. Daarom gebruiken we, net als binnen het initiatief zelf gebeurt, de term 'gebeurtenisgedreven API'.

⁴⁸Het is de snelst groeiende API-specificatie volgens een recente [enquête](#) onder ontwikkelaars, met een verdrievoudiging van het productiegebruik van 2019 tot 2020.

⁴⁹De scope van de AsyncAPI standaard is breder dan voor binnen project Notificatieservices is gedefinieerd. AsyncAPI spreekt over 'berichten' waarbij de inhoud zowel betrekking kan hebben op een plaatsgevonden 'gebeurtenis' ('event') als op een 'opdracht' ('command'). Berichten die uit te voeren opdrachten bevatten vallen nu buiten de scope van project Notificatieservices (zie 'Scope' binnen het hoofddocument).



11 CLOUDEVENTS SPECIFICATIES EN HET GOV NL PROFILE FOR CLOUDEVENTS

11.1 INLEIDING

Een van de doelen van project Notificatieservices was om voor gebruik binnen de Nederlandse overheid een berichtstandaard te ontwikkelen waarmee applicaties elkaar gaan notificeren als zich relevante gebeurtenissen hebben voorgedaan. In lijn met het voornemen om zoveel mogelijk gebruik te maken van internationaal aanwezige kennis en ervaring is gekozen om voort te bouwen op het standaardisatiewerk dat werd en wordt gedaan door de [Serverless Working Group](#) van de [Cloud Native Computing Foundation](#) onder de naam '[CloudEvents](#)'. De afgelopen jaren is een community ontstaan die werkt aan verschillende producten met als overkoepelend doel om event-driven werken te standaardiseren.

Het belangrijkste product van CloudEvents is de specificatie voor het beschrijven van gebeurtenisgegevens die we verder aanduiden als 'de CloudEvents-berichtsificatie'. Doel daarvan is om het definiëren en verstrekken van gegevens over plaatsgevonden gebeurtenissen te standaardiseren. (Ook) notificeren wordt daardoor eenduidiger en kan bij gebruik van verschillende ontwikkel- en hostingomgevingen op een vergelijkbare manier plaatsvinden.

CloudEvents wordt mede ontwikkeld door grote Cloud-providers zoals Amazon, Microsoft en Google die om vergelijkbare redenen als de Nederlandse overheid, maar dan op wereldwijde schaal, streven naar standaardisatie op dit vlak. De CloudEvents-berichtstandaard is de eerste specificatie die door hen is ontwikkeld. Op basis van de aanwezige kennis en ervaring bij de betrokken partijen is gekozen voor een 'less is more aanpak' waarbij men zich bewust beperkt tot een beperkte set beschrijvende metagegevens over gebeurtenissen ('events'). Daarmee wordt het mogelijk dat aanbieders ('Producers') gestandaardiseerd events publiceren, Intermediaries events gestandaardiseerd verwerken en bijv. in de vorm van notificaties verstrekken aan afnemers ('Consumers').

Naast afspraken over een berichtstandaard zijn er in de praktijk ook aanvullende specificaties nodig. Bijv. over hoe de berichtstandaard moet worden toegepast bij gebruik van specifieke gegevensformaten en transportprotocollen. Voor de meest gangbare formaten en protocollen is dit al gebeurd. Binnen project Notificatieservices is, onder andere voor beproevingen van het NL GOV profiel for Cloud Events, gebruik gemaakt van de specificaties voor het HTTP-protocol, het JSON-gegevensformaat en het webhook-interactiepatroon.

Voor meer gestandaardiseerd event-driven werken is het (ook) wenselijk om afspraken te maken over andere aspecten dan alleen publiceren en verstrekken van events. Binnen de CloudEvents-community zijn daarom ook werkgroepen actief die een standaard ontwikkelen voor omschrijving van gebeurtenistypes ('event-types') en voor geautomatiseerd abonneren ('subscription').

Dit hoofdstuk beschrijft achtereenvolgens:

- Een korte toelichting op de doelen van CloudEvents.
- De architectuur van CloudEvents.
- Enkele belangrijke CloudEvents specificaties.
- Het, op basis van de CloudEvents-berichtsificatie ontwikkelde, [NL GOV profile for CloudEvents](#).



11.2 CLOUDEVENTS DOELEN

Het doel van CloudEvents is om de verschillende manieren waarop event-driven wordt gewerkt te vervangen door een gestandaardiseerde werkwijze om interoperabiliteit te bevorderen. Vergelijkbaar als de NORA interoperabiliteit binnen de Nederlandse overheid wil vergroten willen Cloud providers dit doen voor de voorzieningen die zij binnen hun omgevingen aanbieden.

Why CloudEvents?

Events are everywhere, yet event Publishers tend to describe events differently.

- *Consistency: The lack of a common way of describing events means developers have to write new event handling logic for each event source.*
- *Accessibility: No common event format means no common libraries, tooling, and infrastructure for delivering event data across environments. CloudEvents provides SDKs for Go, JavaScript, Java, C#, Ruby, PHP, PowerShell, Rust, and Python that can be used to build event routers, tracing systems, and other tools.*
- *Portability: The portability and productivity we can achieve from event data is hindered overall.*

CloudEvents is a specification for describing event data in a common way. CloudEvents seeks to dramatically simplify event declaration and delivery across services, platforms, and beyond.

Bron: Cloudevents.io

De CloudEvents berichtstandaard definieert een set metadata ('attributen') over gebeurtenissen die tussen applicaties wordt overgedragen en de manier waarop de metadata in uitgewisselde berichten moeten worden opgenomen. Op basis van aanwezige metadata kunnen events gestandaardiseerd worden gefilterd en gerouteerd richting afnemende applicaties ('Consumers'). Consumers kunnen de metadata in verstrekte events gebruiken voor verwerking. Waar nodig kunnen in de vorm van 'extended attributes' eigen, bijv. domeinspecifieke, attributen worden gedefinieerd en in berichten worden opgenomen.

Naast metadata kunnen berichten, in een herkenbaar deel van het bericht, ook inhoudelijke informatie bevatten over een plaatsgevonden gebeurtenis ('payload'). Deze door een Publisher verstrekte informatie is uitsluitend bestemd voor Consumers en wordt door CloudEvents niet beschouwd als bruikbaar voor filtering en routing.

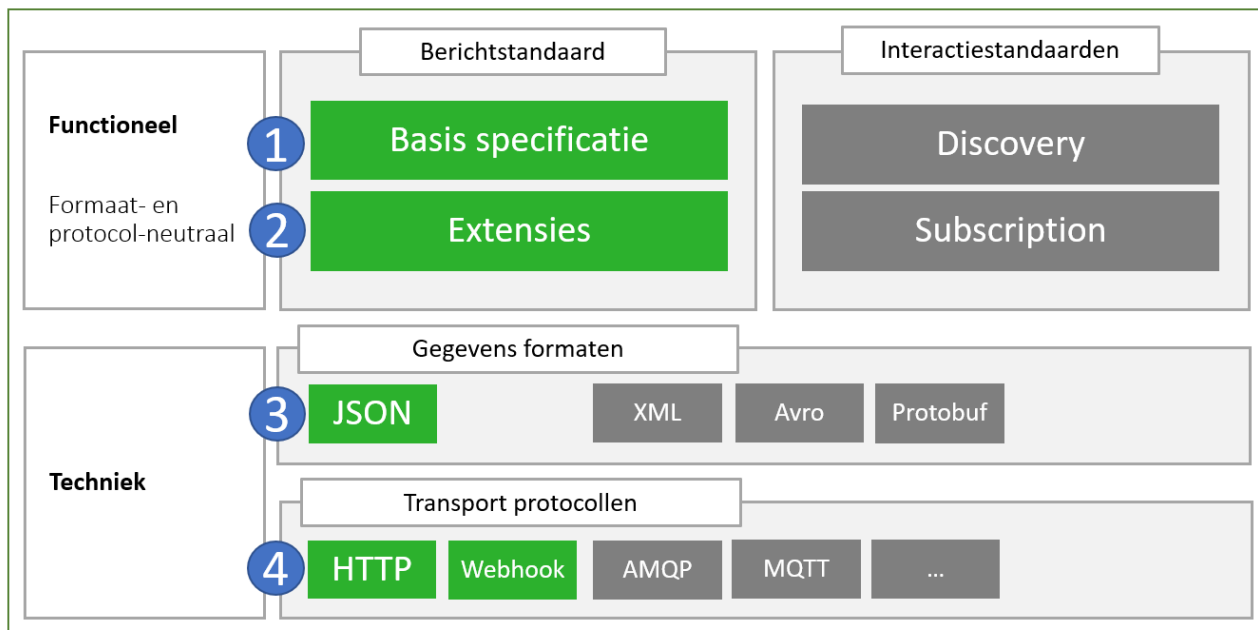
De CloudEvents-berichtsificatie is bewust heel compact gehouden en richt zich uitsluitend op het beschrijven van berichten over events er uit moeten zien. Over allerlei andere aspecten die van belang kunnen zijn bij event-driven werken spreekt de berichtsificatie zich niet uit. Bijv.:

- Wijze van authenticatie en autorisatie.
- Borgen van betrouwbaarheid en afhandeling van storingen.
- Borgen van gegevensintegriteit en vertrouwelijkheid.
- Wijze van abonneren (apart uitgewerkt).
- Wijze van publiceren van event-types en kunnen raadplegen daarvan (apart uitgewerkt).
- Toepassing bij gebruik van bepaalde formaten en protocollen (apart uitgewerkt).

De paragraaf [NL GOV profile for CloudEvents](#) beschrijft de inhoud van de CloudEvents-berichtstandaard meer in detail.

11.3 CLOUDEVENTS ARCHITECTUUR

CloudEvents kent een gelaagde architectuur. Dit maakt het mogelijk om op verschillende aspecten onafhankelijk van elkaar te standaardiseren en het stelt gebruiker in de praktijk in staat om keuzes te maken die in hun situatie het beste passen.



Figuur 9: Gelaagde CloudEvents architectuur (groen=vastgesteld, grijs=in-ontwikkeling)

- 1 De *basis specificatie* definieert een aantal verplichte en optionele attributen die bruikbaar zijn om een plaatsgevonden gebeurtenis te beschrijven in de vorm van een 'event' ("a data record expressing an occurrence and its context").
- 2 *Berichtstandaard extensies* beschrijven optioneel te gebruiken attributen. Deze kunnen generiek van aard zijn (bijv. 'sequence') of domein-specifiek (bijv. 'Burgerservicenummer').
- 3 Per *gegevensformaat* zijn specificaties opgesteld hoe, bij gebruik ervan, de functionele berichtstandaard moet worden toegepast.
- 4 Per *transport-protocol* zijn specificaties opgesteld hoe, bij gebruik ervan, de functionele berichtstandaard moet worden toegepast.

De gelaagdheid maakt het mogelijk om standaardisatie van event-driven werken op verschillende aspecten zelfstandig plaats te laten vinden. Cruciaal daarbij is dat de functionele berichtspecificatie (in laag 1 en 2) volledig formaat- en protocol-neutraal is. Pas na vaststelling van de eerste versie van de 'core CloudEvent specification' zijn de technische standaarden ontwikkeld die toepassing van de functionele standaard beschrijven bij gebruik van specifieke formaten (JSON, AVRO, Protobuf, XML) en protocollen (HTTP, AMQP, MQTT, NATS, Kafka, Websockets)⁵⁰.

CloudEvents beschouwt gestandaardiseerd vastgelegde gegevens over plaatsgevonden gebeurtenissen als zelfstandige objecten ('events') die zijn te verwerken met gebruik van verschillende formaten en protocollen en door allerlei soorten voorzieningen. Het biedt daarmee de vrijheid aan betrokkenen om te kiezen voor gebruik van specifieke formaten, protocollen en voorzieningen die binnen hun context het meest geschikt worden geacht. Het is daarmee standaard die breder toepasbaar is dan standaarden waarbij al technieke-

⁵⁰ Vanuit de behoefte in de praktijk aan een standaard voor gebruik van het webhook-patroon (zie paragraaf 7.4 [Webhook](#)) is door een CloudEvents-werkgroep ook daarvoor een standaard ontwikkeld



gerelateerde keuzes zijn gemaakt (bijv. een API-standaard die een interface beschrijft waarbij gebruik wordt gemaakt van het JSON-formaat, het HTTP-protocol en de REST-architectuurstijl).

Het [NL GOV profile for CloudEvents](#) beschrijft voor de functionele lagen 1 en 2 welke aanvullende afspraken gelden voor gebruik binnen de Nederlandse overheid. In een daarin opgenomen Appendix wordt aanbevolen om gebruik te maken van een drietal aanvullende technische standaarden (laag 3 en 4):

- One SHOULD use the [JSON Event Format for CloudEvents](#) specification and pay attention to the points of attention and recommendations in the guideline [NL GOV Guideline for CloudEvents JSON](#).
- One SHOULD use the [HTTP Protocol Binding for CloudEvents](#) specification and pay attention to the points of attention and recommendations in the guideline [NL GOV Guideline for CloudEvents HTTP](#).
- One SHOULD use the [HTTP 1.1 Web Hooks for Event Delivery](#) specification and pay attention to the points of attention and recommendations in the guideline [NL GOV Guideline for CloudEvents Webhook](#).

Bron: [NL GOV profile for CloudEvents](#)

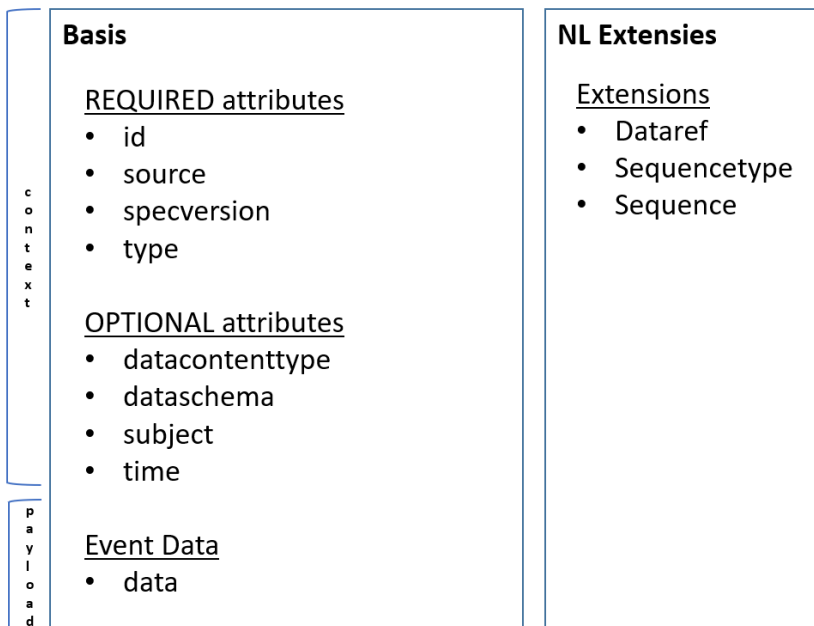
De reden om met name gebruik van de CloudEvents JSON-, HTTP- en Webhook-standaard aan te bevelen is dat ze steeds vaker worden gebruikt voor gegevensuitwisseling binnen de overheid. Onder andere het [Kennisplatform APIs](#) werkt actief aan meer en beter gebruik van APIs die hier gebruik van maken. Door niet alleen te standaardiseren op functionele maar ook op technische aspecten wordt interoperabiliteit verder vergroot en zijn in de praktijk sneller en beter event-driven oplossingen te realiseren.

11.4 NL GOV PROFILE FOR CLOUDEVENTS

CloudEvents is een algemene specificatie voor het definiëren van het formaat van gebeurtenisgegevens. Het NL GOV profile for CloudEvents bouwt hierop en vult het verder aan om gebruik binnen de Nederlandse overheid te optimaliseren. Hiermee kan worden voorkomen dat binnen de overheid op aspecten onnodig verschillende keuzes worden gemaakt. Het profiel bevat daarvoor bijv. afspraken over de naamgeving van extensie-attributen en over hoe specifieke attributen van waarden moeten worden voorzien.

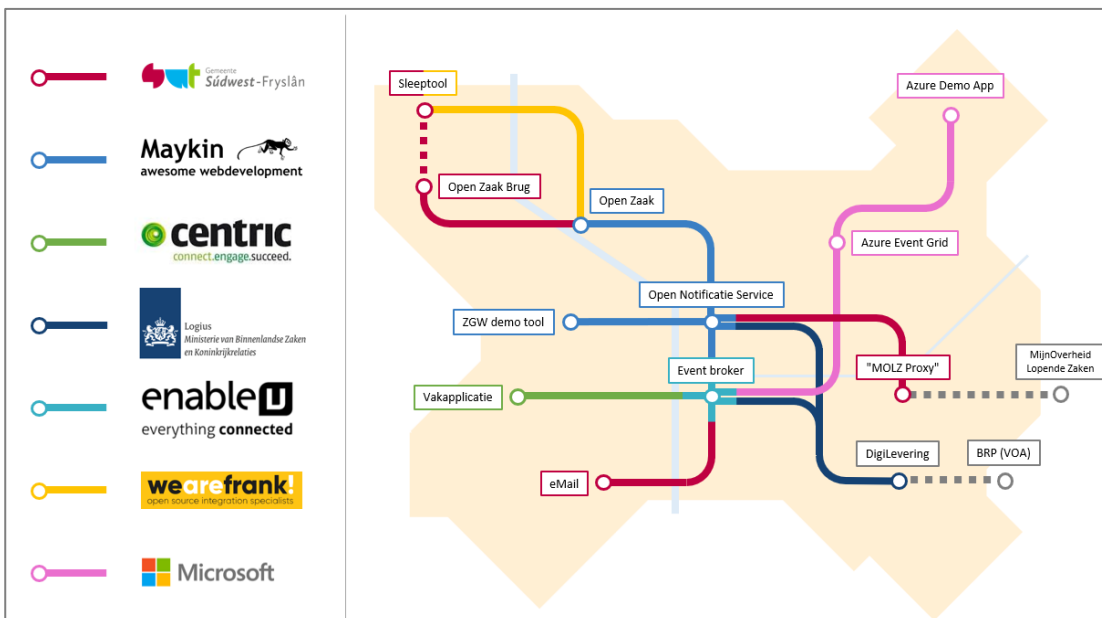
Het profiel beperkt zich tot generieke aspecten waarover overheidsbreed overeenstemming is te bereiken. Inhoudelijke aspecten, zoals het benoemen van soorten gebeurtenissen ('event-types') of het benoemen van domeinspecifieke extensie-attributen, worden bewust overgelaten aan domein-experts. Deze kunnen immers het beste, en vaak als enige, het wat en hoe voor hun domein beoordelen.

Onderstaande afbeelding toont welke event-attributen door CloudEvents zijn gespecificeerd als verplicht en optioneel en welke door het NL GOV profiel als optioneel.



Binnen het NL GOV profiel is voor ieder van de hierboven afgebeelde attributen beschreven wat de betekenis is en welke afspraken er met betrekking tot gebruik gelden. Hoewel het op het eerste oog om een heel beperkte set aan attributen gaat is hiermee al een grote mate van standaardisatie te realiseren. Tijdens project Notificatieservices uitgevoerd beproevingen van de berichtstandaard toonden aan dat dankzij gebruik van het profiel snel, werkende event-driven oplossingen waren te realiseren.

Tijdens een hackathon is door een vijftal leveranciers gedemonstreerd hoe zij zij met gebruik van het profiel in korte tijd events conform het NL GOV profiel konden produceren, filteren, routeren, verstrekken en ontvangen. Binnen enkele dagen tijd werden met gebruik van bestaande voorzieningen werkende oplossingen gerealiseerd.



Figuur 10: Betrokken leveranciers en voorzieningen bij Hackathon 'beproeving berichtstandaard'

Bijzonderheden die de brede bruikbaarheid van het NL GOV profiel en CloudEvents standaard illustreren:

- Tijdens de hackathon is ook getoond hoe voor het routeren en filteren van events relatief eenvoudig gebruik was te maken van voorzieningen van Cloud-providers. In dit geval de voorziening 'EventGrid' die aanwezig is binnen Microsoft's Azure Cloudomgeving.



- Door Logius is via een beproeving aangetoond hoe op basis van een aantal BRP-mutatiegegevens events conform NL GOV formaat waren samen te stellen. Daarbij werd gebruik gemaakt van een zogenaamde 'SDK' die is ontwikkeld om bij gebruik van een bepaalde programmeertaal, in dit geval Java, snel en eenvoudig te kunnen werken met standaard events.

De volledige specificatie van het NL GOV profile for CloudEvents is te vinden op: <https://vng-realisatie.github.io/NL-GOV-profile-for-CloudEvents>. Daarin is ook vermeld op welke manier is bij te dragen aan verdere ontwikkeling van het profiel.



BIJLAGE 1: RELATIE MET BASISREGISTRATIES

Binnen project Notificatieservices wordt nadrukkelijk gekeken naar hoe notificeren kan helpen om organisaties op de hoogte stellen van bepaalde wijzigingen binnen bronregistraties. De basisregistraties vormen daarbij een belangrijke categorie bronregistraties voor de overheid.

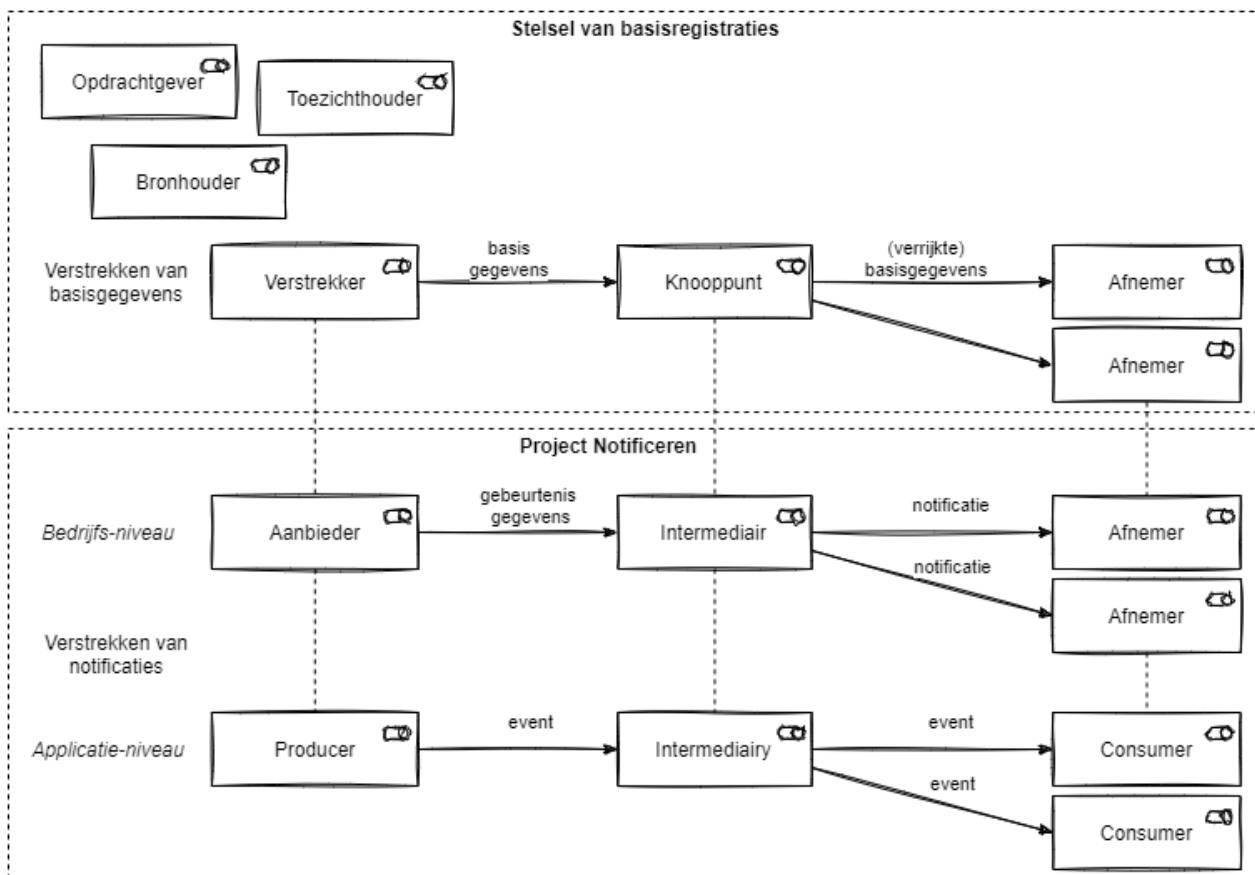
Voor beheer en gebruik van basisregistraties zijn binnen de overheid een aantal ‘stelselrollen’ benoemd. Deze bijlage beschrijft kort hoe de rollen ‘(dienst)aanbieder’, ‘(notificatie)intermediair’ en ‘(dienst)afnemer’ zich verhouden tot de stelselrollen zoals die voor de basisregistraties zijn benoemd.

Opdrachtgever	De opdrachtgever is het voor de basisregistratie verantwoordelijke ministerie, dat opdrachtgever is voor de ‘verstrekker’ (de beheerder van de landelijke voorziening). Bijvoorbeeld: het ministerie van BZK is opdrachtgever voor de Basisregistratie Personen.
Toezichthouder	De toezichthouder is de partij die er verantwoordelijk voor is dat wordt toegezien of de basisregistratie in overeenstemming met eisen, afspraken en wetgeving opereert. Een basisregistratie heeft één of meer verantwoordelijken voor toezicht. Over het algemeen is de opdrachtgever verantwoordelijk voor het toezicht op de naleving van de wettelijke bepalingen die gelden voor die basisregistratie.
Bronhouder	Een bronhouder is verantwoordelijk voor het inwinnen en bijhouden van de authentieke en niet-authentieke gegevens in een basisregistratie en voor het borgen van de kwaliteit van die gegevens (onder meer naar aanleiding van ontvangen terugmeldingen). Een basisregistratie heeft één of meer bronhouders.
Verstrekker	De verstrekker is verantwoordelijk voor het verstrekken van de gegevens aan afnemers. De verstrekker is ook verantwoordelijk voor het faciliteren van het gebruik (zoals het leveren van kennis en ondersteuning aan afnemers voor het aansluiten op de landelijke voorziening). Een basisregistratie heeft één verstrekker.
Afnemer	Een afnemer (ook wel ‘gebruiker’) is een overheidsorganisatie of private partij die gegevens afneemt van een basisregistratie voor gebruik in de eigen processen. Voor bestuursorganen met een publiekrechtelijke taak (zoals gemeenten, provincies, waterschappen en zelfstandige bestuursorganen) is het afnemen en gebruiken van relevante authentieke gegevens verplicht. ¹

Verder wordt de rol van ‘*knooppunt*’ onderscheiden: “een proactieve intermediair (intermediair) tussen de houders van bronnen (waaronder de basisregistraties) en de afnemende organisaties. Het regelt de gegevenslogistiek (integratie, conversie en distributie) en beheert de afspraken en gemeenschappelijke voorzieningen.”²

De onderscheiden stelselrollen zijn bedoeld om de *registratie, beheer en verstrekking van basisgegevens* te optimaliseren en niet specifiek het notificeren van afnemers naar aanleiding van plaatsgevonden gebeurtenissen.

Onderstaand schema toont hoe de terminologie van de basisregistraties zich verhoudt tot de terminologie zoals die binnen Project Notificatieservices wordt gebruikt.



Uitgaande van de benoemde stelselrollen geldt bij notificeren normaal gesproken dat:

- Een Verstrekker de rol van (dienst)aanbieder/Producer heeft.
- Een Knooppunt de rol van Intermediair.
- Een Afnemer de rol van (dienst)afnemer/Consumer.

In de praktijk kan een Verstrekker zowel de rol van Producer als Intermediary vervullen. Dit gebeurt bijv. als het Kadaster zelf geautomatiseerd notificaties verstrekt aan Consumers.

Het komt ook voor de Verstrekkers gebruik maken van een Intermediary die zorgt voor het geautomatiseerd verstrekken van notificaties aan Consumers. Dit is bijv. het geval als het Kadaster gegevens aanlevert bij de stelselvoorziening [Digilevering](#) die vervolgens zorgt voor het verstrekken van de juiste notificaties aan de juiste Consumers.

BIJLAGE 2: RELATIE MET NORA, GDI-ARCHITECTUUR EN FORUM STANDAARDISATIE

Nederlandse Overheid Referentie Architectuur

Op een hoger plan brengen van event-driven werken en notificeren binnen de Nederlandse overheid vereist maatregelen op verschillende niveaus. In termen van het NORA vijfslaagsmodel zijn op ieder van de daar genoemde lagen verbeteringen mogelijk. Project Notificatieservices heeft zich met name gericht op de informatie- en applicatielaag maar tijdens het project wel een aantal constatering gedaan die op andere lagen betrekking hebben. Deze zijn onder andere vastgelegd binnen diverse [handreikingen](#).

In relatie tot de NORA geldt dat event-driven oplossingen in veel opzichten bij kunnen dragen aan het binnen de (vernieuwde) NORA benoemde [kernwaarden](#) en [kwaliteitsdoelen](#). Gestandaardiseerd event-driven werken stelt de overheid in staat om sneller en beter gegevens uit te wisselen. Dit vergroot interoperabiliteit en maakt betere dienstverlening mogelijk. Het maakt het mogelijk om al lang gekoesterde wensen zoals ‘naar buiten toe optreden als 1 overheid’ of ‘verbeteren van klantreizen naar aanleiding van life-events’ te kunnen realiseren.

Voor een aantal van de [NORA Architectuurprincipes](#) geldt dat werken in lijn daarmee (ook) event-driven kunnen werken vereist (bijv. ‘[Bied de dienst proactief aan](#)’). Voor het principe ‘Informeel bij de bron’ geldt dat omschrijving nog te weinig duidelijk maakt dat overheidsorganisaties niet alleen het opvragen maar ook het *verstrekken* van brongegevens moeten gaan faciliteren⁵¹.

Generieke Digitale Infrastructuur (GDI)

Binnen de [GDI Architectuur](#) voor doorontwikkeling van de Generieke Digitale Infrastructuur, is een van de leidende principes: ‘Afspraken voor standaarden voor voorzieningen’. Project Notificatieservices heeft zowel een aantal afspraken als standaarden voorgesteld.

- **Afspraken:** zowel in deze notitie als in verschillende opgeleverde handreikingen worden adviezen gegeven en afspraken voorgesteld die helpen om als overheid effectief gebeurtenisgedreven te kunnen werken.
- **Standaarden:** er zijn meerdere standaarden opgeleverd die bijdragen aan gestandaardiseerd uitwisselen van gegevens over plaatsgevonden gebeurtenissen. Het belangrijkste product is het [NL GOV profile for CloudEvents](#): een uitbreidbaar berichtformaat waarmee 'events' op een standaard manier worden beschreven (waarmee o.a. routing en filtering eenduidig mogelijk wordt). Aanvullend zijn in de vorm van handreikingen specificaties gepubliceerd die beschrijven hoe het standaard berichtformaat moet worden gebruikt bij gebruik van specifieke protocollen, formaten en patronen: [HTTP](#), [JSON](#) en [Webhook](#). In samenwerking met de VNG is een [API-specificatie](#) ontwikkeld voor notificeren met gebruik van REST API's.



Figuur 11: NORA Vijfslaagsmodel

⁵¹ De bij het principe opgenomen ‘Stelling’ luidt: “Maak bij de dienst gebruik van gegevens die afkomstig zijn uit een bronregistratie.” Dit optimaal doen impliceert dat gegevens zowel via opvraging als via verstrekking voor gebruik beschikbaar kunnen komen.



- **Voorzieningen:** Binnen dit architectuurdocument wordt aanbevolen om van bewezen voorzieningen ('middleware') gebruik te maken om de rol van Intermediary in te vullen binnen notificatieprocessen (al dan niet voorzien van een eigen schilapplicatie). Er worden geen aanbevelingen gedaan om specifieke voorzieningen te gebruiken of om bijv. in- of externe voorzieningen te gebruiken. De Nederlandse overheid kent de stelselvoorziening Digilevering die speciaal is ontwikkeld om te notificeren over plaatsgevonden gebeurtenissen met gevolgen hebben voor basisregistratiegegevens. Daarnaast bestaan er in de markt veel gespecialiseerde producten die bruikbaar zijn om de rol van Intermediary in te vullen.

Forum Standaardisatie

[Forum Standaardisatie](#) is een adviescommissie met deskundigen uit diverse overheidsorganisaties, het bedrijfsleven en de wetenschap die voor de publieke sector open standaarden toetst, er over adviseert en gebruik ervan voorschrijft.

Het binnen project Notificatieservices ontwikkelde NL GOV Profile for CloudEvents is de eerste stap op weg naar een toekomstige toetsing⁵². Daarvoor zal de standaard moeten worden doorontwikkeld en in de praktijk verder moeten worden beproefd. Ten tijde van het einde van project Notificatieservices geldt dat het de bedoeling is om beheer en doorontwikkeling van de standaard te beleggen bij een uitvoeringsorganisatie.


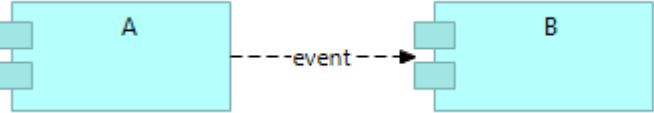
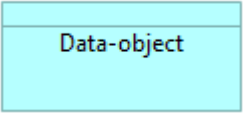
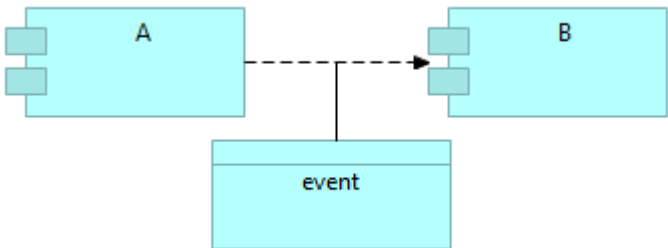
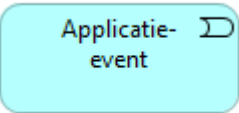

⁵² Op basis van een toetsingsadvies beslist het Overheidsbreed Beleidsoverleg Digitale Overheid (OBDO) of, en zo ja met welke mate van verplichting, de standaard wordt opgenomen binnen de [lijst Open Standaarden](#).



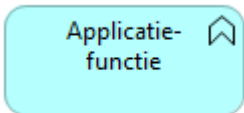

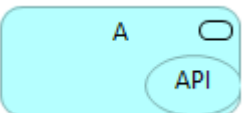
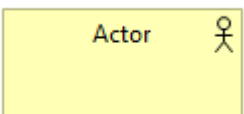
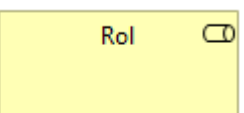
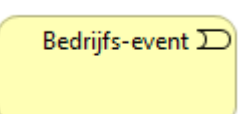
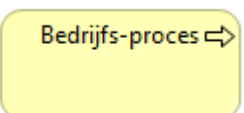
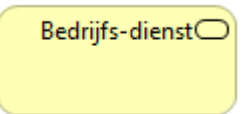



BIJLAGE 3: TOELICHTING ARCHIMATE

ArchiMate is een open en onafhankelijke beschrijvingstaal voor enterprise-architecturen. De taal wordt ondersteund door diverse modelleerhulpmiddelen voor IT-architectuur van verschillende leveranciers. De taal is gebaseerd op IEEE-standaard 1471. Sinds 2008 wordt ArchiMate als open standaard ondersteund en beheerd door The Open Group. – Bron: [Wikipedia](https://en.wikipedia.org/wiki/ArchiMate)

Deze bijlage geeft een toelichting op de elementen en relaties die worden gebruikt binnen diagrammen waarbij de beschrijvingstaal Archimate is gebruikt. Voor definities van elementen en relaties wordt verwezen naar [ArchiMate specificatie](#). Onderstaande korte toelichting is bedoeld als laagdrempelige context-specifieke toelichting voor de lezer.

	Element	Korte toelichting
1		Applicatie Kan bij notificeren de rol van Producer, intermediary of Consumer vervullen. 'Rol' wordt niet afzonderlijk gemodelleerd.
2		Flow-relatie De stippellijn met pijl geeft een 'flow' relatie weer: er stroomt event-data van applicatie A naar applicatie B (of: 'er wordt door A een event verstrekt aan B').
3		Data-object Een object waarin gestructureerde data aanwezig is. Bijv. een database, een bericht of, in termen van CloudEvents, een event.
4		Voorbeeld Applicatie A verstrekt een event aan applicatie B. Het event is hier expliciet als data-object gemodelleerd (in tegenstelling tot bij 2).
5		Applicatie-event Een gebeurtenis die een applicatie triggert om bepaalde acties uit te voeren. Applicatie-events kunnen hun oorsprong hebben buiten of binnen de applicatie.
6		Applicatieservice Een service die door een applicatie wordt geleverd. Bijv. een intermediair levert bijv. services voor routeren, filteren en verstrekken van events aan Consumers.



7		Applicatie-functie Specifieke interne functionaliteit van een applicatie.
8		Applicatie Programming Interface Een toegangspunt dat een applicatie of service biedt waar een bepaalde service beschikbaar wordt gesteld voor gebruik.
9		Voorbeeld Applicatieservice A biedt een API die via aanroep gebruik van de service mogelijk maakt.
10		Actor Een persoon of organisatie. Bijv. de organisatie die in de rol van Producer events publiceert of de organisatie aan wie in de rol van Consumer events worden verstrekt.
11		Rol De rol van een actor binnen de organisatie Bijv. in relatie tot events: 'aanbieder', 'intermediair', 'afnemer'.
12		(Bedrijfs)gebeurtenis Een gebeurtenis die tot actie op bedrijfsmatig niveau leidt (bijv. het telefonisch ontvangen van een productaanvraag).
13		(Bedrijfs)proces Een proces dat op bedrijfsmatig niveau plaatsvindt (eventueel ondersteund met applicatieservices).
14		(Bedrijfs)dienst Een dienst die door een organisatie aan de buitenwereld wordt geleverd (bijv. het verstrekken van paspoorten).
15		Bedrijfsfunctie Een interne gespecialiseerde functie binnen een organisatie (bijv. 'Financiën').
16		Flow-relatie waarbij gegevens tussen elementen stromen in de richting van de pijl.
17		Bedient-relatie waarbij een element diensten levert aan een ander element (bijv. een applicatieservice ten behoeve van de uitvoering van een bedrijfsproces).



BIJLAGE 4: MIDDLEWARE

Er zijn legio softwareproducten te koop die, met name in de rol van Intermediair, event-driven werken en notificeren kunnen ondersteunen. Bijv. door asynchrone communicatie tussen Producer en Consumers mogelijk te maken.

In veel situaties is gebruik van dit type specialistische software wenselijk om de, al snel complexe, functionaliteit voor betrouwbaar event-driven werken te realiseren. Bijkomend voordeel is dat Producers en Consumers kunnen zich dan kunnen beperken tot hun kernfunctionaliteit.

Een aantal bekende en veelgebruikte producten die hiervoor bruikbaar zijn bijvoorbeeld: Dell Boomi, TIBCO Enterprise Message Service, MuleSoft Anypoint Platform, IBM MQ, Apache Kafka, IBM Event Streams, Amazon Simple Queue Service, RabbitMQ, Azure Queue Storage, Apache RocketMQ, Ably, Solace Publish-Subscribe+. Door leveranciers worden dit type producten steeds vaker via een SaaS-model aangeboden. Organisaties kunnen daarmee (verder) worden ontzorgd en een aantal systeemtechnische taken uitbesteden.

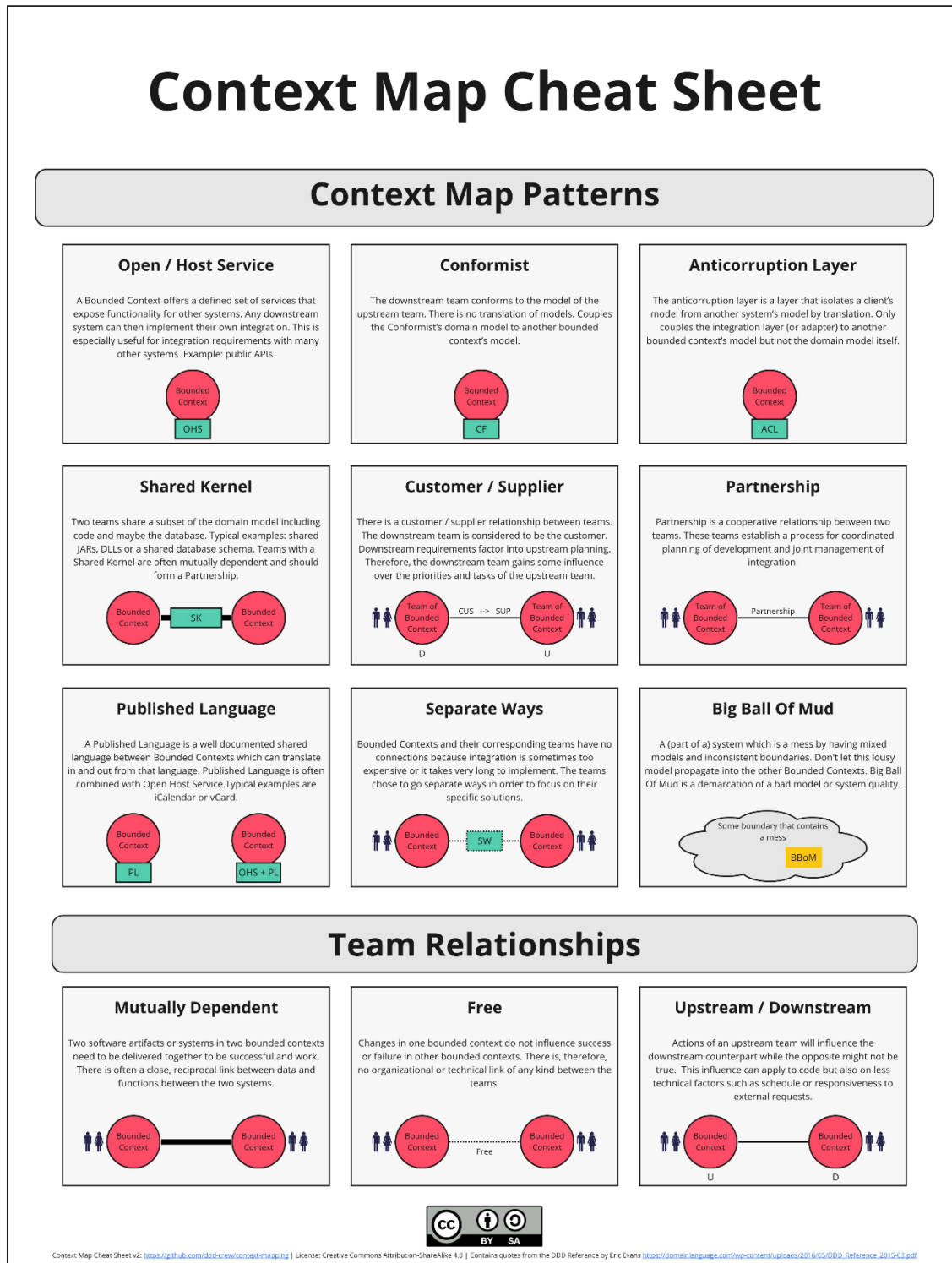
Waar eerder vaak gekozen werd voor een bepaald softwareproduct wordt, met name in Cloud omgevingen, steeds vaker gebruik gemaakt van een combinatie van producten. Vaak zijn dit leverancier-eigen producten die elkaar kunnen aanvullen en gecombineerd kunnen worden tot werkende oplossingen. Microsoft biedt bijv. producten aan als Azure Service Bus, Azure Event Hubs, Azure Event Grid, Azure WebPubSub terwijl Amazon producten kent zoals Amazon SQS, Amazon MQ, Amazon Kinesis en Amazon EventBridge.

Naast meer traditionele producten leveren cloud-leveranciers ook andere producten die meer generiek van aard zijn maar ook bruikbaar zijn voor het verwerken van events en verstrekken van notificaties. Denk bijv. aan serverless functies waarmee binnen stappen van een notificatieproces op maat functionaliteit is te realiseren. Bij gebruik van Cloud platforms wordt ontwerpen en realiseren van event-driven oplossingen steeds meer een kwestie van 'assembleren van de juiste componenten'.

Cloud platforms kennen nu al honderden services (en dit aantal groeit nog steeds) dus komen er steeds meer mogelijkheden om goede en betaalbare event-driven oplossingen te realiseren. Het is een uitdaging om voldoende overzicht en kennis te hebben van de veelheid aan producten. Gelet op het feit dat 'events overal zijn' (denk bijv. aan het Internet of Things) en te verwachten is dat ze een steeds belangrijkere rol gaan spelen binnen informatievoorziening is het wenselijk om (ook) kennis te hebben van Cloud platform producten omdat daarmee hele nieuwe mogelijkheden voor event-driven oplossingen binnen bereik komen.

BIJLAGE 5: DOMAIN DRIVEN DESIGN - CONTEXT MAPS

Context Maps zijn een hulpmiddel binnen Domain Driven Design om het contact tussen begrensde contexten en teams te beschrijven met behulp van een aantal patronen. Gebruik hiervan is behulpzaam om bij event-driven oplossingen te ontwerpen die aansluiten bij hoe organisaties of applicaties zich tot elkaar verhouden (bijv. of er sprake is van een Producer die bepalend is en Consumers zich moeten aanpassen of dat er sprake is van een meer gelijkwaardige relatie waarbij bijv. in overleg keuzes worden gemaakt over het type events dat wordt gepubliceerd).



Figuur 12: bron: <https://github.com/ddc-crew/context-mapping>



BIJLAGE 6: HANDREIKINGEN NOTIFICATIESERVICES

Hieronder staan een aantal handleidingen zoals die gemaakt zijn tijdens project Notificatieservices. De handleidingen hebben als doel om tijdens het project opgedane kennis over een bepaald onderwerp over te dragen.

Technische handleidingen

Onderstaande meer 'technische handleidingen' beschrijven hoe het [NL GOV profile for CloudEvents](#) gestandaardiseerd kan worden toegepast bij gebruik van enkele vaak gebruikte gegevensformaten, transportprotocollen en patronen:

- [JSON-gegevensformaat](#)
- [HTTP-protocol](#)
- [Webhook interactiepatroon.](#)

Functionele handleidingen

Onderstaande handleidingen beschrijven op een toegankelijke manier een aantal belangrijke aspecten van gebeurtenisgedreven werken en notificeren:

- [Introductie van notificeren](#)
- [Het waarom van notificeren](#)
- [Definiëren van gebeurtenistypes](#)
- [Randvoorwaardelijke aspecten](#)
- [Notificatiescenario's](#)