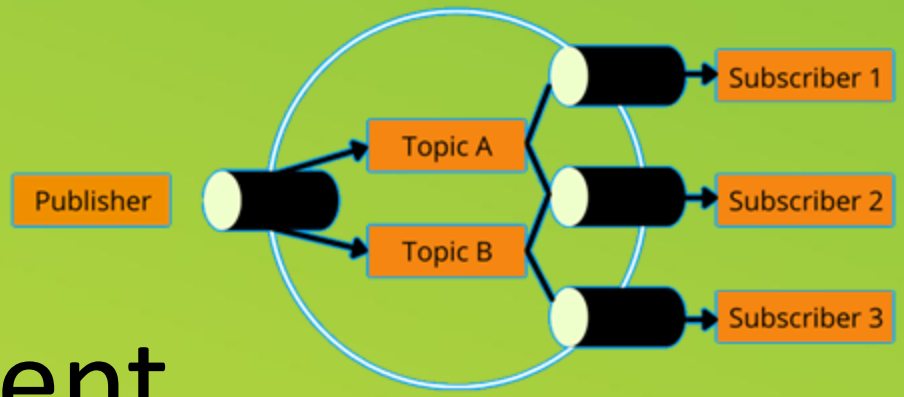
CoPSA



Visiedocument
Event Driven Architecture

Onderzoek naar de mogelijkheden van **Event Driven Architecture**
in de gegevensuitwisseling.

Versie 1.0
Vastgesteld in de CoP SA 9, 17-01-2024

mfbas

CoPSA

Deel 1: Presentatie van de belangrijkste bevindingen

Van de CoP SA Werkgroep “Visie op Event Driven Architectuur”

mffbas

Inleiding bij dit visiedocument

- Directe aanleiding: **IC 284A** (optimalisatie stamdata bericht secundair deel) en **IC 284B** (optimalisatie business service opvragen gegevens primair deel meetinrichting).
- Daarnaast behoeft aan meer inzicht in **business toepassingsmogelijkheden** van moderne (vaak technisch toegepaste) architectuurpatronen zoals Event Driven Architecture
- Onderzoek door **CoP SA WG** in Q2 – Q4 2023, met review rondes en vaststelling CoP SA 9, 17 januari 2024
- Bevindingen, conclusies en aanbevelingen vastgelegd in dit visiedocument
- Dient als **richtlijn** voor het bepalen of EDA van toepassing is als oplossingsrichting
- Dient daarnaast als **beoordelingskader** bij te realiseren oplossingen, zodat de juiste vragen gesteld kunnen worden

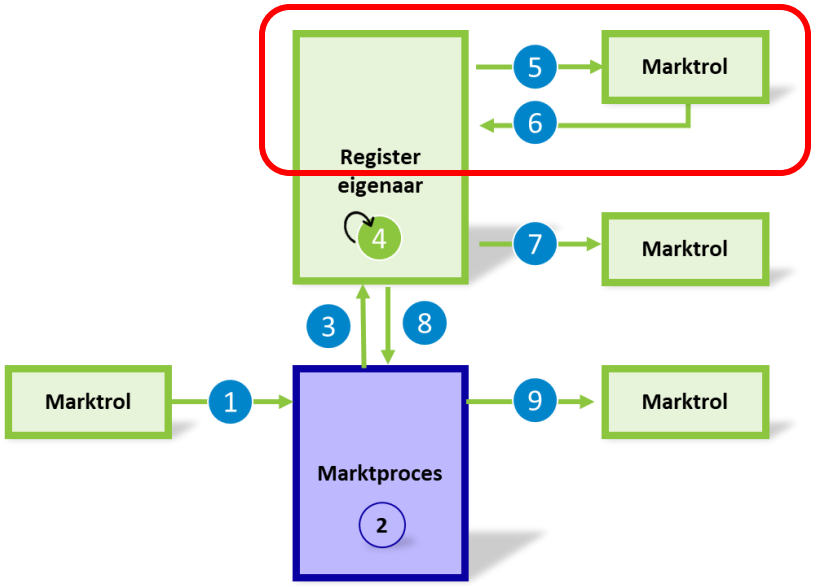
CoPSA

Event Driven Architecture (of EDA)
is een ontwerp-patroon voor toepassingen
dat wordt gebruikt wanneer het wenselijk is
om zo snel mogelijk
op een 'gebeurtenis' te kunnen reageren.

Wij zien voornamelijk toepassingen
wanneer een register wordt geupdate
(naar aanleiding van een gebeurtenis in de buitenwereld
of een marktproces)
en een marktrol daarnaar moet handelen.

mffbas

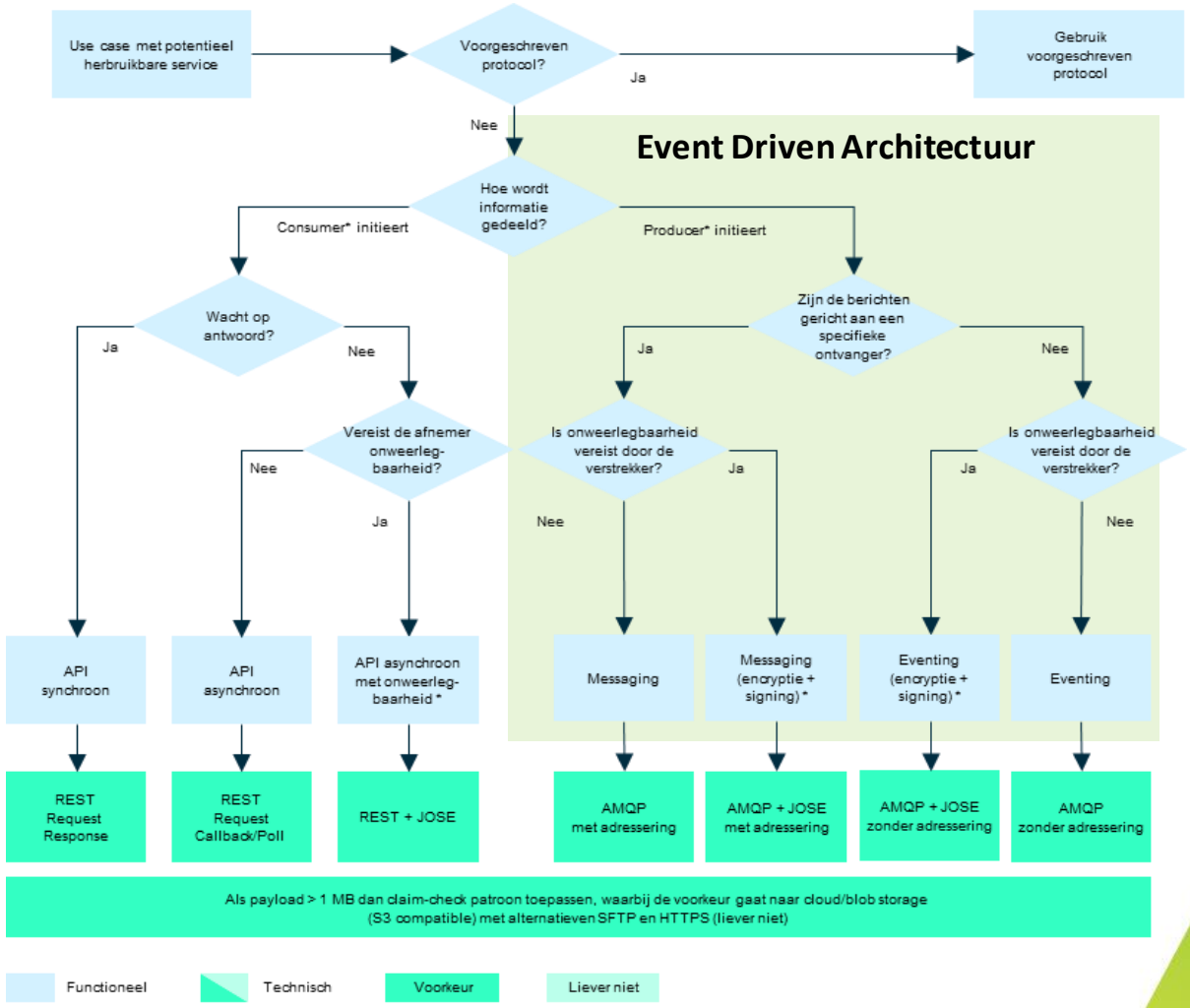
Waar(om) Event Driven Architectuur



1. EDA draagt bij aan het **snel kunnen reageren** op belangrijke gebeurtenissen zodat organisaties vroegtijdig maatregelen kunnen nemen;
2. Er kan vergaande **ontkoppeling** van producer en consumers worden gerealiseerd waardoor ze volledig onafhankelijk van elkaar kunnen functioneren;
3. Er is een *one-to-many* relatie waarbij het aantal consumers **eenvoudig uitbreidbaar** is;
4. Sluit aan bij **hedendaagse Cloud platform technologie** waarbij er steeds meer mogelijkheden komen om goede en betaalbare event-driven oplossingen te realiseren;
5. Sluit aan bij **visie van de NL overheid**: Project Notificatieservices (2022) wil het mogelijk maken dat Nederlandse overheidsorganisaties vaker en beter gebeurtenisgedreven ('event driven') werken;

Scope EDA en koppeling met Integratiepatronen Ketensamenwerking

- In het document ‘Integratiepatronen Ketensamenwerking’ wordt een beslisboom gepresenteerd voor integratiepatronen t.a.v. (minimaal de) gegevensuitwisseling tussen Netbeheerders onderling en het centrale CMF landschap.
- Aan de rechterzijde bevinden zich de patronen die passen bij een gegevensuitwisseling waarbij de Producer het informatiedelen initieert.
- **Deze patronen behoren in de definitie van dit visiedocument tot de scope van Event Driven Architecture**



Bron: “Integratiepatronen Ketensamenwerking” Netbeheer Nederland, juni 2022

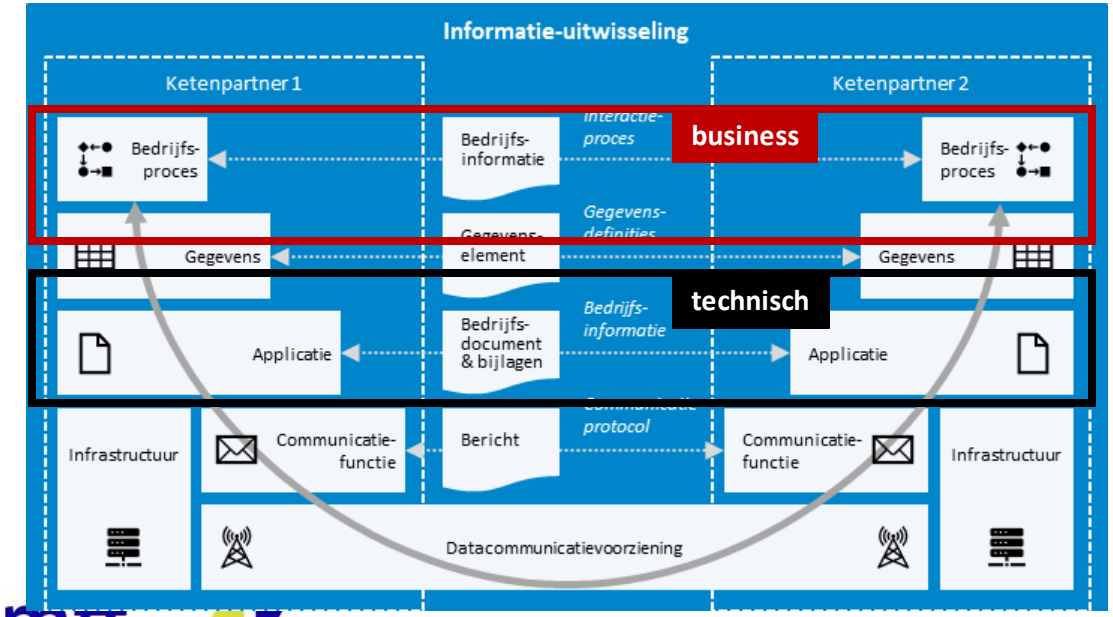
Toelichting Producer en Consumer

- Producer en Consumer zijn rollen bij een integratie use case
- Producer is de partij die de data ter beschikking stelt in de context van de use case
- Consumer is de partij die de data consumeert in de context van de use case

Een Event Driven Architectuur is ofwel Business event ofwel Technisch event gedreven

Business events in de context van EDA

Deze verwijzen naar concrete bedrijfsgebeurtenissen of -signalen die **relevant zijn voor zakelijke processen, besluitvorming en activiteiten** binnen een organisatie. Deze gebeurtenissen zijn meestal gerelateerd aan de bedrijfsactiviteiten en doelstellingen van een organisatie. Het is belangrijk voor de organisatie dat deze in staat is snel te reageren op een dergelijke gebeurtenis.



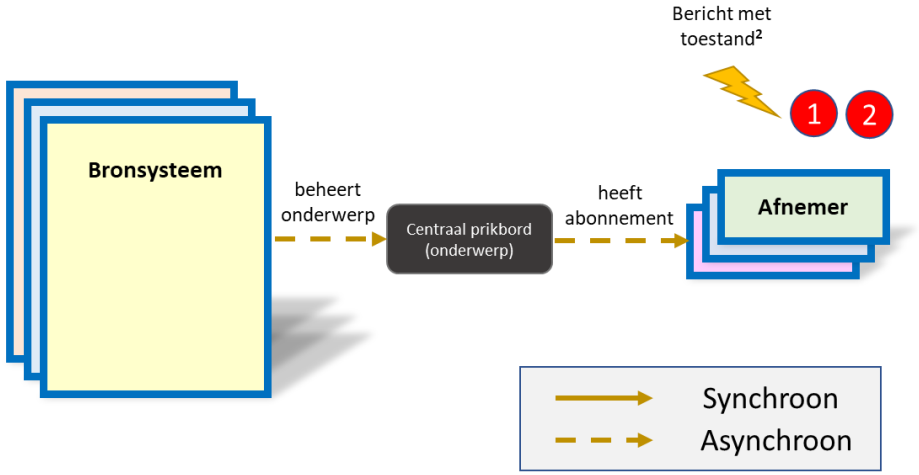
Technische events in de context van EDA

Dit zijn technische gebeurtenissen die zich **voordoen in de technische infrastructuur en systemen van een organisatie**. Deze events hebben betrekking op de werking en het beheer van technologie, zoals netwerken, servers, sensoren en apparaten. Vaak gaat het om vele (duizenden, miljoenen) kleine signalen of status updates die geen directe betekenis hebben voor de business, maar waarbij het wel nodig is om deze te monitoren en wanneer nodig snel te kunnen reageren (denk aan een systeemstoring of een security incident).

Een Event Driven Architecture kent ofwel informatie-rijke events ofwel informatie-arme events

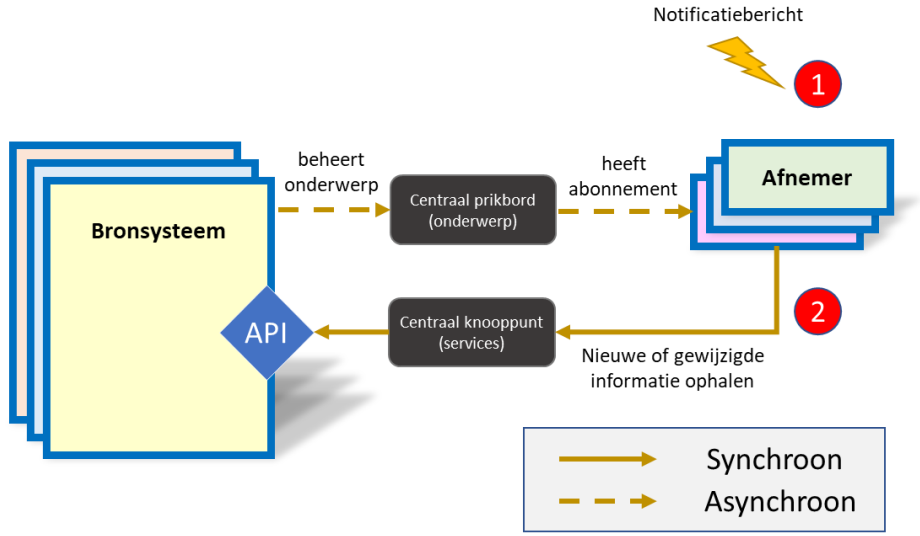
1. Informatie-rijke variant

In het informatie-rijke patroon van EDA wordt naast de notificatie ook de betreffende informatie zelf meegestuurd, in een bericht (ook wel 'event-carried state transfer' patroon genaamd). Dit lijkt eenvoudiger te implementeren, maar vereist wel additionele afspraken over de gegevens in het event.



2. Informatie-arme variant

In het informatie-arme patroon van EDA wordt alleen een notificatie van een wijziging gepost, en haalt de partij de informatie zelf later op via een API. Hierbij is wel een API architectuur nodig voor het ophalen van de gegevens bij de bron.



The Event-Carried State Transfer pattern | by Oskar uit de Bos | ITNEXT

De vier gezichten van EDA | Smals Research

Aanbeveling CoP SA WG: adopteer Event Driven Architecture als een waardevol patroon in de optimalisatie van gegevensuitwisseling in het energiedomein.

Bedenk daarbij wel dat door toepassing van EDA de regie op het proces verschuift.

Van Orkestratie:

- Centrale dirigent: In orkestratie is er een centrale controller of dirigent die de stappen en volgorde van acties beheert. Deze controller coördineert het gedrag van deelnemers.
- Gedirigeerd: Deelnemers voeren acties uit zoals aangegeven door de centrale controller. Ze hebben minder autonomie en volgen een gecoördineerd plan.
- Meer controle: Er is een strikte controle over de volgorde en timing van acties, wat nuttig kan zijn voor complexe workflows.
- Voorbeeld: Stel je voor dat een centrale applicatie alle stappen in een orderverwerkingsproces beheert. Het verzendt instructies naar verschillende subsystemen om specifieke taken uit te voeren, zoals voorraadcontrole, facturering en verzending.

Naar Choreografie:

- Samenwerking tussen deelnemers: In choreografie werken systemen of partijen samen zonder een centrale controller of dirigent. Ze communiceren direct met elkaar op basis van vooraf gedefinieerde regels.
- Zelfsturend: Elke deelnemer weet wat zijn volgende stap is op basis van berichten die hij ontvangt. Ze zijn autonoom en onafhankelijk.
- Minder controle: Er is geen strikte controle over de volgorde van acties, waardoor het flexibel is maar ook complex kan worden.
- Voorbeeld: Stel je een proces voor waarbij verschillende applicaties in een e-commerce systeem samenwerken. Ze sturen berichten naar elkaar om een bestelling te verwerken zonder een centrale controller.

Next steps

- Inzetten als **beoordelingskader** bij te realiseren oplossingen, zodat de juiste vragen gesteld kunnen worden
- Inzetten als **richtlijn** voor het bepalen of EDA van toepassing is als oplossingsrichting
- Concreet: toepassen in de uitwerking van IC 284 A en B → reeds gestart
- Waar nodig: verdere kaders en richtlijnen voor de implementatie van EDA, wanneer daar aanleiding c.q. behoefte voor is

Dank U!

Vragen?

CoPSA

Deel 2: Visiedocument

Zoals opgesteld door de CoP SA Werkgroep
“Visie op Event Driven Architectuur”

mffbas

Versie en wijzigingsgeschiedenis

Versie	Datum	Auteur	Wijzigingen
0.3	29-06-2023	Martijn van Glabbeek	Aanpassingen structuur document n.a.v. feedback werkgroep
0.4	30-06-2023	Werkgroep	Aanpassingen n.a.v. doorloop document in werkgroep
0.5	03-07-2023	Martijn van Glabbeek	Aanvullingen na review, t.b.v. het creëren van een review document voor de CoP SecureArchitecture
0.6	07-07-2023	Werkgroep	Versie voor review door CoP
0.7	01-09-2023	Werkgroep	Aanpassingen n.a.v. review commentaar CoP; titel document gewijzigd. Zie apart document 'review commentaar' voor de exacte verwerking daarvan.
0.8	17-10-2023	Werkgroep	Aanpassingen n.a.v. final check op versie 0.7. Procedure voor vaststelling besproken: eerst de WG, dan naar de CoP SA, dan toelichten in BOSO/BODO. Geen vaststelling in ALV MFF, alleen ter informatie.
0.9	09-11-2023	Werkgroep	Versie ter vaststelling in de CoP SA meeting 8
0.91	02-01-2023	Martijn van Glabbeek	Presentatie t.b.v. CoP SA 9
1.0	18-01-2024	Martijn van Glabbeek	Definitieve versie op basis van de vaststelling in de CoP SA 9, waarin presentatie en toelichting zijn gecombineerd

Inhoud van dit document

1. Aanbeveling: adopteer Event Driven Architecture als waardevol patroon in de optimalisatie van gegevensuitwisseling in het energiedomein

Dit visiedocument toont de resultaten van het onderzoek van de CoP SA werkgroep naar de mogelijke toepassing van EDA in de gegevensuitwisseling (systeemprocessen en datadelen).

Het toepassen van EDA kent verschillende voordelen:

- *Verbeterde efficiëntie*: door gegevens event gedreven te delen, kunnen partijen in de keten de processen beter op elkaar laten aansluiten, wat leidt tot efficiëntie en een betere gebruikservaring voor de klant
- *Flexibiliteit en schaalbaarheid*: EDA maakt het gemakkelijker om nieuwe functionaliteiten en diensten toe te voegen zonder grote wijzigingen in bestaande systemen.

Uiteraard zijn er ook nadelen bij het toepassen van EDA:

- *Complexiteit*: de implementatie van EDA kan complex zijn en vereist aandacht voor dataverwerking, synchronisatie en fouttolerantie.
- *Kosten*: het opzetten van een EDA-infrastructuur en het trainen van personeel kan initieel meer investeringen met zich meebrengen dan alternatieve patronen.

Conclusie en Aanbevelingen:

- De toepassing van EDA biedt aanzienlijke voordelen voor de gegevensuitwisseling in het energiedomein.
- Het is een effectieve business samenwerkingsvorm voor organisaties met ontkoppelde waardestromen die snel willen inspelen op een business gebeurtenis in de keten.
- Daarnaast is EDA een effectief technisch patroon voor use cases waarbij een data consument een technisch event snel wil kunnen analyseren op benodige acties.
- **Deze visie biedt een basis voor verdere uitwerking (bijvoorbeeld een richtlijn of een concrete implementatie) om de voordelen van Event Driven Architecture volledig te benutten.**

2. Referenties en leeswijzer bij dit document

Referenties

- Dit document bevat de Visie op de toepassing van Event Driven Architecture van de Community of Practice Secure Architecture (CoP SA) verbonden aan MFFBAS.
- Dit document is nauw gerelateerd aan het overkoepelende document Visie op Informatie Uitwisseling van de CoP Secure Architecture.
- Daarbij maken we maximaal gebruik van de inzichten en het uitzoekwerk door de NL Overheid, Notificatie aanpak en Logius CloudEvents
- Dit document bouwt voort op het document 'Integratie Architectuur Marktfacilitering' van de TC NEDU en zoekt aansluiting bij het document 'CMF – Integratiepatronen – Ketensamenwerking v1' van Netbeheer Nederland.
- Dit document gebruikt inzichten op datadelen en datamanagement uit het rapport Data Governance van de Topsector Energie.
- Omdat de ontwikkelingen in de energiemarkt niet stilstaan, moet dit document ook worden gezien als een snapshot in een zich steeds ontwikkelend inzicht in de eisen die worden gesteld aan een goede informatie-voorziening.

Leeswijzer

- Allereerst gaat dit document in op de noodzaak voor de verbetering van gegevensuitwisseling in de energiesector.
- Daarna beschrijft het document de doelstellingen en uitgangspunten bij het inrichten van de toekomstige gegevensuitwisseling.
- Vervolgens wordt beschreven hoe een Event Driven Architecture bijdraagt aan de doelstellingen van de toekomstige gegevensuitwisseling.
- De kern van het document is het visie-statement hoe de CoP SA de toepassing van EDA ziet in de toekomst.
- Vervolgens wordt beschreven wanneer EDA het beste toegepast kan worden en daarna hoe EDA toegepast zou kunnen worden in een actuele voorbeeld-casus.
- In de bijlagen worden belangrijke begrippen geduid en nuttige brondocumenten aangehaald.
- *Let wel dat in dit visie-document geen (mogelijke) implementatie wordt beschreven of voorgesteld. Indien tot implementatie wordt overgegaan, kan dit document wel dienen als leidraad.*

Stand van zaken: project Notificatie Services | VNG

3. Aanleiding: de ontwikkelingen in het energiedomein maken vernieuwing van de gegevensuitwisseling in de energiesector noodzakelijk

De markt vraagt om vernieuwing van gegevensuitwisseling

- Directe aanleiding voor deze visie zijn de issues IC284A en IC284B, die met de bestaande middelen niet optimaal op te lossen zijn, en vragen om een nieuwe manier van inrichten
- Een betere gegevensuitwisseling tussen marktpartijen is cruciaal om de uitdagingen van de energiemarkt in het geheel en de energietransitie in het bijzonder aan te pakken.
- Het stelt hen in staat om efficiënter samen te werken, flexibeler te reageren op vraag en aanbod, te innoveren en transparanter te opereren.
- Doordat partijen meer en meer gebruik maken van moderne (cloud-) oplossingen, zijn legacy oplossingen een belemmering voor partijen om in te stappen en goed te integreren met de gegevensuitwisseling in de energiemarkt
- Ook de nieuwe energiewet is ook een trigger voor vernieuwing: deze vraagt om meer gegevensuitwisseling tussen meer marktrollen over meer soorten gegevens.
- Moderne architectuur-patronen, zoals Event Driven Architecture, kunnen een belangrijke rol in spelen in de verbetering van de bestaande informatie-voorziening.

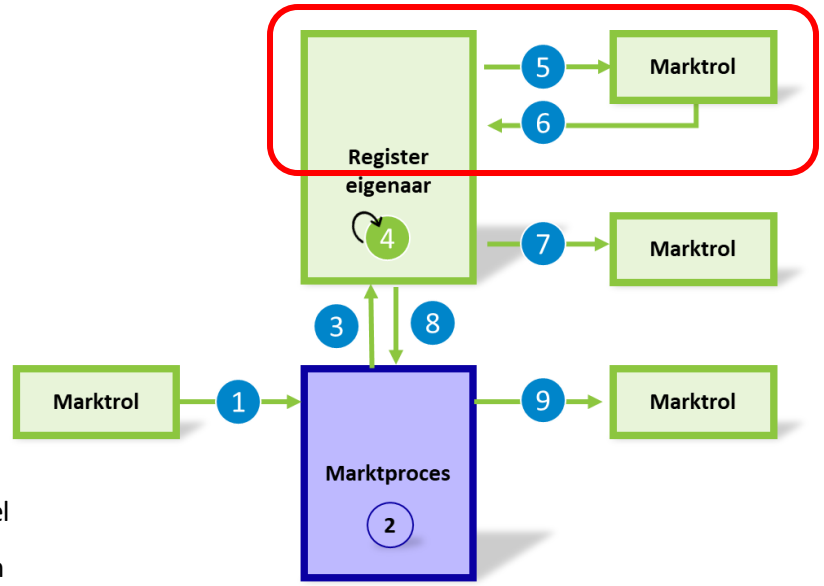
De huidige situatie kent echter belemmeringen

- De huidige inrichting van de gegevensuitwisseling is deels nog gebaseerd op patronen en technieken uit de begintijd van de gegevensuitwisseling in de energiesector.
- Dit kan leiden tot inertie. Wanneer de focus vooral ligt op het gemak van hergebruik en de kortere termijn investering, wordt gewenste, zelfs noodzakelijke verbetering van de gegevensuitwisseling niet bereikt.
- Het is immers makkelijker om bestaande patronen en technieken te blijven toepassen dan om nieuwe manieren te adopteren: vanwege de bestaande kennis, hergebruik van bestaande systemen, en de lagere kosten op de korte termijn daarvan.
- We zien daarom de noodzaak om nieuwe manieren van gegevensuitwisseling te stimuleren, door een goede duiding van wat die manieren zijn en hoe deze bijdragen aan de doelstellingen. En door breed geaccepteerde richtlijnen te ontwikkelen voor de toepassing hiervan.
- Dit document, dat zich richt op Event Driven Architecture speelt samen met de bovenliggende Visie op gegevensuitwisseling een rol in het uitdragen daarvan.

4. Toepassingsgebied van EDA en de relatie met architectuur principes

Toepassingsgebied EDA

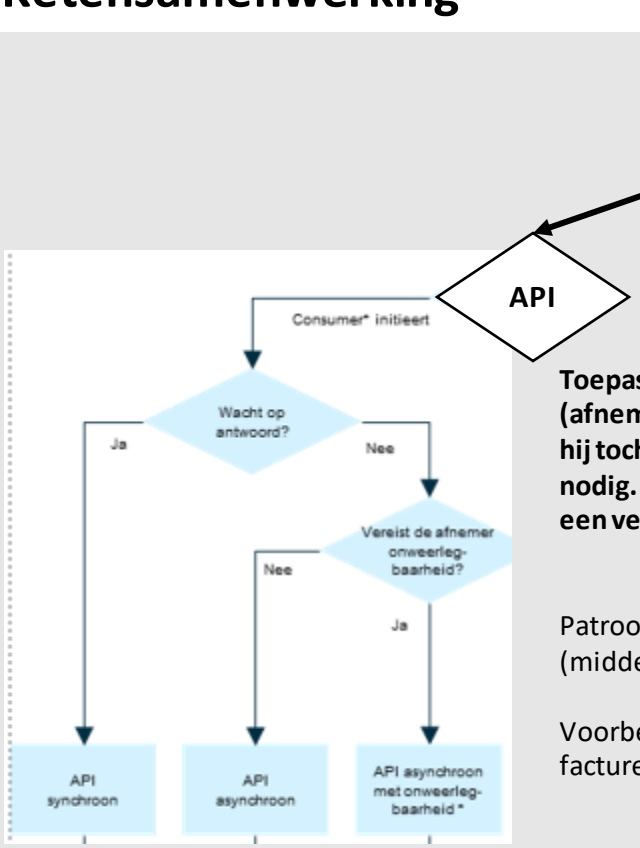
- Event Driven Architecture wordt in onze ogen met name toegepast in de gegevensuitwisseling tussen een register (eigenaar) en een partij met een bepaalde marktrol.
- We hanteren in dit visiedocument een brede definitie van Event Driven Architecture.
- In een Event Driven Architectuur worden acties en communicatie tussen verschillende componenten van een systeem getriggerd door gebeurtenissen (events). Deze gebeurtenissen kunnen variëren van gebruikersacties tot interne systeemgebeurtenissen.
- Daaronder verstaan we zowel de business als de technische event driven architectuur, zowel informatierijk als informatie-arm, en zowel gebaseerd op eventing (broadcast, gericht aan alle geabonneerde afnemers, zonder adressering) als gebaseerd op messaging (narrowcast, één of meer specifieke ontvangers, dus met adressering).



Architectuur principes relevant voor EDA

- In dit document geven we onze visie op de toepassing van zowel de business als technische verschijningsvorm van EDA.
- Wanneer het EDA patroon wordt toegepast, is het van belang dat de implementatie wordt getoetst tegen alle architectuur principes.
- Het inzetten van EDA dient onderzocht te worden in situaties waar een bepaalde marktrol behoefte heeft om direct op de hoogte te worden gebracht van een business gebeurtenis, omdat hij daar direct wil kunnen reageren.
- Hierbij blijft het principe dat er één bron van data is gewaarborgd.
- Om technische redenen kan er wel sprake zijn van replicatie, maar die replicatie kan/mag nooit leidend zijn in (vervolg-) processen c.q. voor andere (markt-) actoren.
- Deze zullen de bron weer (opnieuw) moeten bevragen.

5. Visualisatie toepassing van verschillende patronen, op basis van de Integratiepatronen Ketensamenwerking



Toepassing: De consumer (afnemer) heeft voor een actie die hij toch al zou nemen, actuele data nodig. Hij initieert op dat moment een verzoek om actuele data.

Patroon: berichten opvragen (middels een API).

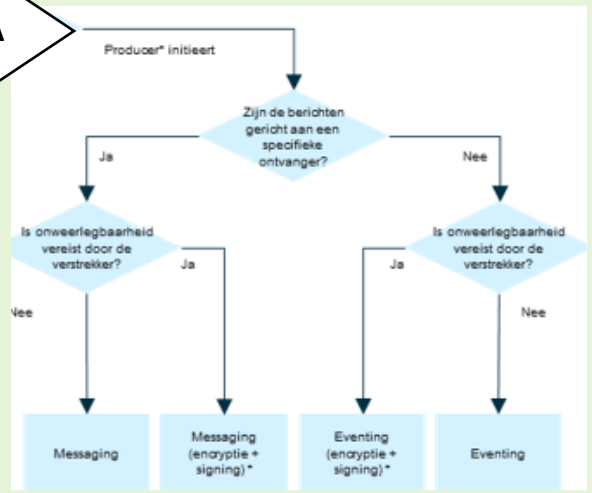
Voorbeeld use case: maandelijkse facturering (niet verder uitgewerkt)

EDA

Toepassing: De consumer moet direct actie ondernemen die hij zonder ingelicht te worden over een gebeurtenis, niet direct zou kunnen ondernemen, en heeft daarbij de gegevens van de producer (verstrekker) nodig.

Patroon: notificaties ontvangen (middels een message of event).

Voorbeeld use case: wijziging primair deel meetinrichting



6. Bij de beoordeling of een gegevensuitwisseling in aanmerking komt voor EDA, moet altijd een afweging van de voor- en nadelen plaatsvinden

Voordelen EDA algemeen

1. Dienstverlening: signalen komen snel terecht op de juiste plaats. Dit biedt kansen om snel te reageren en klanten beter van dienst te zijn;
2. Verwachting: we leven in een samenleving waarin snel reageren de norm is. EDA draagt bij aan het real-time reageren zodat organisaties vroegtijdig maatregelen kunnen nemen;
3. Ontkoppeling: er kan vergaande ontkoppeling van producer en consumers worden gerealiseerd waardoor ze volledig onafhankelijk van elkaar kunnen functioneren;
4. Robuustheid: door de ontkoppelde en gedistribueerde architectuur werken fouten niet door in andere systemen en zijn ze beter op te vangen;
5. Schaalbaarheid: applicaties werken autonoom waardoor (horizontaal) op- en afschalen van capaciteit mogelijk is;
6. Interoperabiliteit: via notificeren zijn partijen vaker en sneller van actuele informatie te voorzien en ontstaat een groeiend netwerk waarin event-driven wordt samengewerkt;
7. Uitbreidbaarheid: events kunnen geconsumeerd worden door meerdere consumers. Er is een *one-to-many* relatie waarbij het aantal consumers eenvoudig uitbreidbaar is;
8. Recovery: een event broker die streaming diensten aanbiedt maakt het mogelijk events uit het verleden opnieuw af te spelen (‘replay’);
9. Cloud technologie: sluit aan bij hedendaagse Cloud platform technologie waarbij er steeds meer mogelijkheden komen om goede en betaalbare event-driven oplossingen te realiseren;
10. NL overheid: sluit aan bij visie van de NL overheid: Project Notificatieservices (2022) wil het mogelijk maken dat Nederlandse overheidsorganisaties vaker en beter gebeurtenisgedreven (‘event driven’) werken;

Nadelen EDA algemeen

- EDA is geen silver-bullit. Er moet goed worden nagedacht of de situatie inderdaad vraagt om een business event driven patroon, danwel een technisch event driven patroon, danwel een combinatie daarvan.
- Daarnaast vereist EDA een hogere IT-volwassenheid van participerende organisaties;
- De noodzaak blijft om heldere afspraken te maken en standaarden toe te passen om te zorgen dat betrokken organisaties en applicaties goed kunnen samenwerken;
- Zowel bij producers, intermediaries als consumers zijn maatregelen nodig om goed om te gaan met fouten en verstoringen bij gegevensuitwisseling om het vereiste betrouwbaarheidsniveau te garanderen
- De verschuiving van orkestratie naar choreografie heeft implicaties, bijvoorbeeld dat (keten) processen lastiger te testen zijn (zie volgende slide).

7. Een Event Driven Architecture kent twee verschijningsvormen: business of technisch, die elkaar kunnen versterken

Business event-driven architectuur en technische event-driven architectuur zijn twee concepten die verband houden met het ontwerp en de organisatie van systemen, maar ze hebben verschillende focusgebieden en doelen. In essentie draait het bij business event-driven architectuur om het verbeteren van bedrijfsprocessen en besluitvorming, terwijl technische event-driven architectuur gericht is op het creëren van robuuste en adaptieve technische systemen. Het is ook mogelijk om beide concepten te combineren, waarbij technische gebeurtenissen de bedrijfsprocessen aansturen, en de technische architectuur de basis legt voor het vastleggen en verwerken van business events.

1. Business Event Driven Architecture

Business Event-Driven Architectuur: een samenwerkingsvorm

- Focus op Bedrijfsprocessen: Bij een business event-driven architectuur draait alles om het sturen en optimaliseren van bedrijfsprocessen. Het richt zich op het begrijpen en reageren op de gebeurtenissen die van invloed zijn op de bedrijfsactiviteiten.
- Bedrijfslogica: Deze architectuur is gericht op het vastleggen en verwerken van bedrijfsgerelateerde gebeurtenissen, zoals bestellingen, transacties, klantinteracties en marktveranderingen.
- Besluitvorming: Business events kunnen de trigger zijn voor belangrijke beslissingen en acties in het bedrijf. Het doel is om bedrijfsprocessen wendbaarder te maken door automatische reacties op gebeurtenissen mogelijk te maken.
- Bedrijfsintegratie: Business event-driven architecturen kunnen worden gebruikt om verschillende bedrijfssystemen en applicaties naadloos te integreren, waardoor realtime gegevensuitwisseling mogelijk wordt.

2. Technische Event Driven Architecture

Technische Event-Driven Architectuur: een technisch patroon

- Focus op Technische Infrastructuur: Technische event-driven architectuur richt zich op het bouwen van flexibele en schaalbare technische systemen die reageren op gebeurtenissen in de technische omgeving.
- Infrastructuur Gebeurtenissen: Dit type architectuur richt zich op technische gebeurtenissen zoals foutmeldingen, systeemstatussen, netwerkproblemen en servergebeurtenissen.
- Systeemintegratie: Technische event-driven architecturen zijn vaak gericht op het verbinden van verschillende technische componenten en microservices om een efficiënte en betrouwbare verwerking van gebeurtenissen mogelijk te maken.
- Schaalbaarheid en Betrouwbaarheid: Dit soort architecturen is vaak ontworpen om schaalbaarheid en betrouwbaarheid te bieden, waarbij systemen automatisch reageren op veranderingen in de technische omgeving, zoals verhoogde belasting of uitval van componenten.

8. Business Event Driven Architecture is een proces-ontwerp patroon

Definitie Business Event Driven Architecture

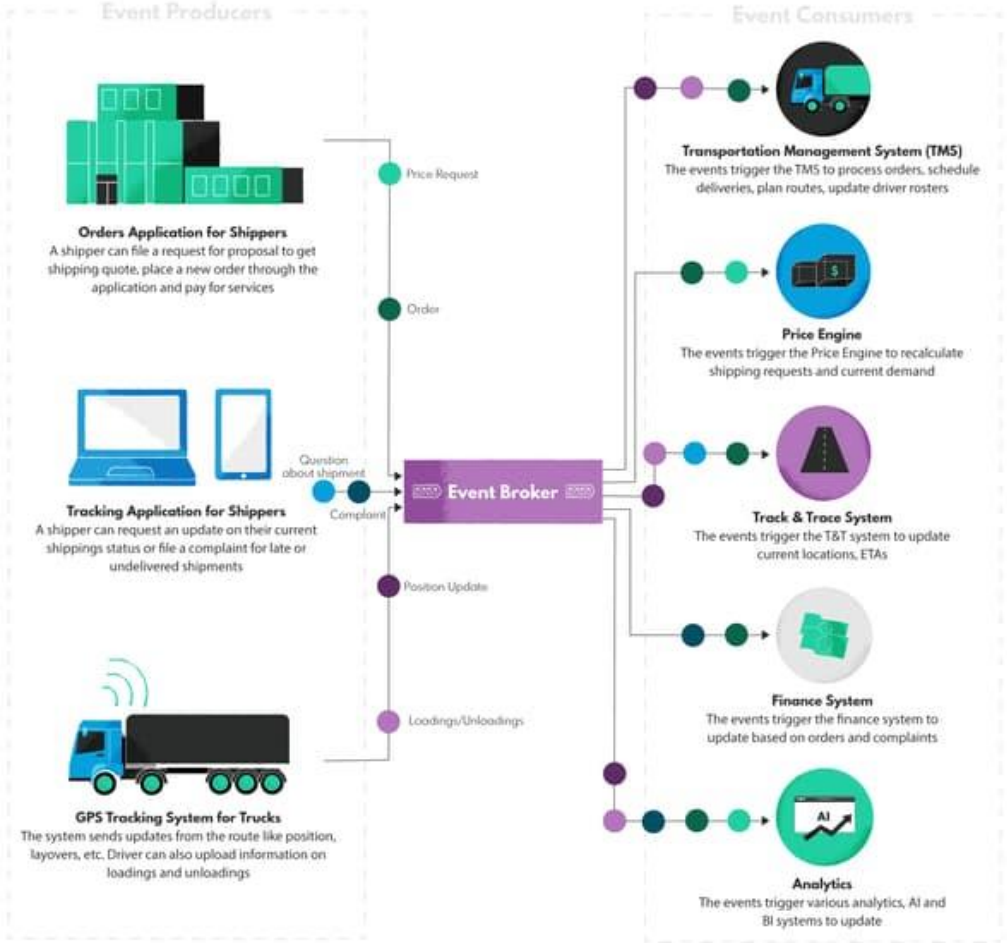
- Business gedreven EDA is een manier om business processen en interacties te ontwerpen die afhankelijk zijn van zakelijke gebeurtenissen om acties te activeren.
- Door het ontwerp te baseren op deze triggerende gebeurtenissen, krijgt u meer flexibiliteit en de mogelijkheid om nieuw gedrag toe te voegen.
- Voordelen zijn een verbeterde responsiviteit en time-to-market.
- EDA introduceert ook complexiteit. De gebruiker moet ervoor zorgen dat de gebeurtenissen die als trigger fungeren, kernbedrijfsgebeurtenissen zijn: zaken die belangrijk zijn voor uw activiteiten.
- EDA is het meest geschikt voor ondernemingen die moeten opschalen en snel moeten reageren op veranderingen in hun omgeving.
- Hoewel het gemakkelijker is om gebeurtenismechanismen voor greenfield-systemen te bouwen, is het perfect mogelijk om ze te introduceren met behulp van uw bestaande infrastructuur.

Event-driven architecture | Thoughtworks

What Is Event-Driven Architecture? Everything You Need to Know (hubspot.com)

Key Benefits of Event-Driven Architecture for Your Business (nexocode.com)

Visualisatie werking Business EDA (voorbeeld)

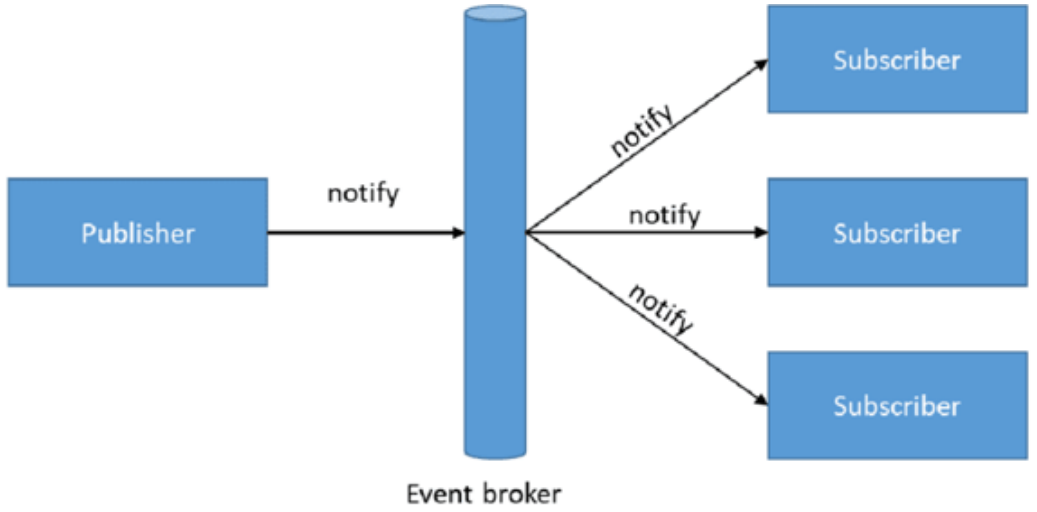


9. Technische Event Driven Architecture is een applicatie-integratie patroon

Definitie Technische Event Driven Architecture

- Een technische event-gedreven architectuur is een applicatie-integratie architectuur-patroon gebaseerd op het *publish-subscribe* mechanisme, waarbij de focus ligt op het communiceren van veranderingen tussen verschillende systemen.
- In een event-gedreven architectuur worden *events* (notificaties) geproduceerd door de *publisher* wanneer er belangrijke veranderingen plaatsvinden binnen een bepaald domein (proces, register etc).
- Een *subscriber* abonneert zich op deze *events* (of een deel daarvan). Wanneer deze het *event* ontvangt, kan deze hierop reageren.
- De *events* worden gedistribueerd via een *event broker*.
- **We maken daarbij onderscheid tussen *messages*, die specifiek zijn geadresseerd voor één of meer ontvangers, en *events*, die niet zijn geadresseerd, en dus** kunnen worden ontvangen door iedereen die geïnteresseerd is.
- Dit kan op meerdere manieren worden geïmplementeerd, maar dat is buiten scope van dit document.
- We onderscheiden ook twee varianten van *events* of notificaties: informatie-rijk en informatie-arm; deze worden toegelicht op de volgende sheets.

Visualisatie werking Technische Event Driven Architecture

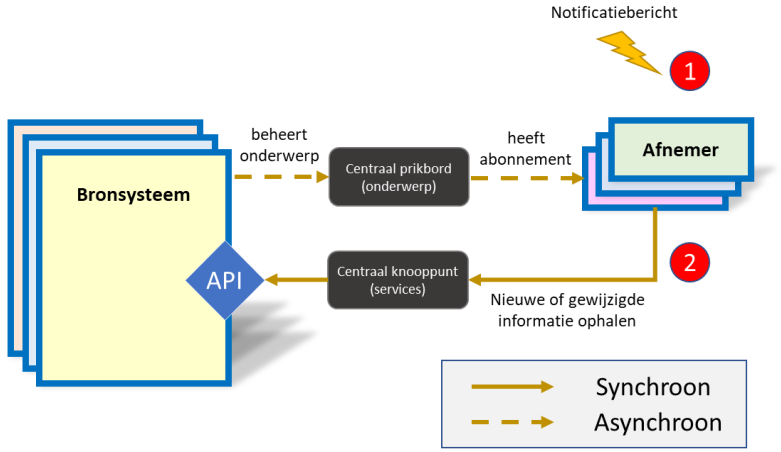


NL GOV profile for CloudEvents | Logius

Advanced Event Broker. An event mesh for connected enterprises | Solace

10. Toelichting Informatie-arme EDA variant

Uitgangspunten



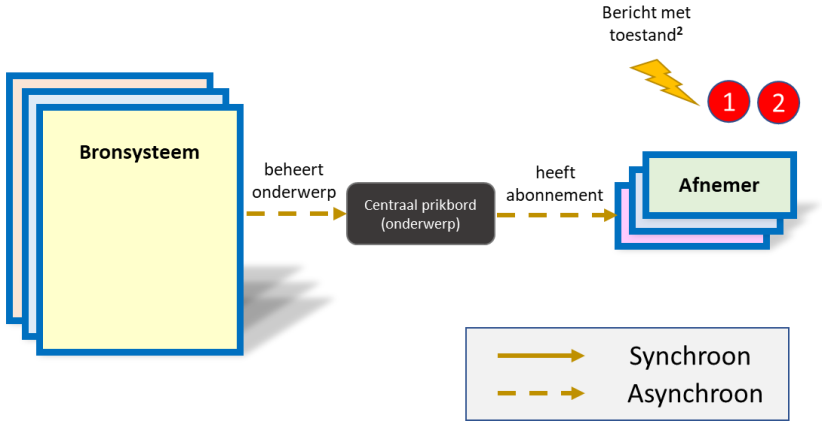
- De bronsystemen zijn via een API raadpleegbaar
- De afnemers hebben toegang tot de API's via het centrale knooppunt
- De bronsystemen en de afnemers hebben toegang tot een centraal prikbord

Toelichting

- Dit patroon hanteert een scheiding van informatie en meta-informatie (informatie over informatie). Afnemers abonneren zich via een centraal prikbord op een onderwerp en worden genotificeerd over alle relevante gebeurtenissen in de context van dit onderwerp.
- De notificatie (1) verwijst alleen naar wijzigingen in het bronsysteem en is in die zin informatiearm. De afnemer wordt geacht zelf actie te ondernemen om de relevante informatie (wijziging / nieuwe status) op te halen bij de bron.
- Wanneer nieuwe of gewijzigde informatie in het bronsysteem beschikbaar is, wordt dit in de context van een vooraf bepaald onderwerp kenbaar gemaakt via het centrale prikbord.
- Afnemers die op dit onderwerp zijn geabonneerd ontvangen hiervan een notificatie(bericht). Vervolgens gebruiken ze de inhoud van zo'n notificatie (meta-informatie) om het bronsysteem gericht te bevragen en de nieuwe of gewijzigde informatie op te halen. De afnemer bepaalt zelf het moment waarop dit gebeurt maar ook welke informatie nodig is.

11. Toelichting Informatie-rijke EDA variant

Uitgangspunten



- Per berichtenstroom is een informatiemodel gedefinieerd
- Routering naar specifieke afnemers geschiedt op basis van kenmerken in het bericht (filters)
- De bronsystemen en de afnemers hebben toegang tot een centraal prikbord

Toelichting

- Dit patroon gaat uit van een berichtenstroom met volledige toestandsinformatie.
- Afnemers abonneren zich hierop via een centraal prikbord voor een bepaald onderwerp (topic) en ontvangen na een gebeurtenis direct een notificatie(bericht) met alle relevante toestandsinformatie in de context van dit onderwerp.
- Het bericht (1) is feitelijk een envelop waarin de gewijzigde toestand(en) (2) in het bronsysteem zijn opgenomen en is in die zin informatierijk.
- De afnemer kan de relevante informatie dus direct verwerken zonder aanvullende informatie op te halen.
- Wanneer nieuwe of gewijzigde informatie in het bronsysteem beschikbaar is, worden de gewijzigde toestanden in de context van een vooraf bepaald onderwerp via het centrale prikbord gedistribueerd.
- Afnemers die op dit onderwerp zijn geabonneerd ontvangen alle relevante toestandswijzigingen in een berichtenstroom.

12. Welke variant past bij welk soort samenwerking?

Samenwerking	Patroon	Voordelen	Nadelen
Centraal ↕ **Decentraal**	**Informatie-arm**	• Informeren en verwijzen naar een centrale raadpleegfunctie is eenvoudig. • Maximale efficiëntie omdat abonnees zelf kunnen ophalen wat nodig is.	• Vereist aanvullend mechanisme om te kunnen omgaan met gemiste events. • Kan complexer worden als er state moet worden gedeeld tussen centrale en decentrale systemen.
	Informatie-rijk	• Beschikbaarheid van de volledige state (indien echt nodig) zorgt voor consistente verwerking • Beschikbaarheid van de volledige state (indien echt nodig) verbetert ook de robuustheid.	• Extra logica voor synchronisatie van state. • Intensiever beheer van uitwisselmodellen. • Significant meer bandbreedte en minder efficiëntie wanneer snapshots naar veel abonnees worden gedistribueerd.
Decentraal ↕ **Decentraal**	**Informatie-arm**	• Eenvoudige architectuur met losse koppelingen tussen decentrale systemen. • Vereenvoudigt ontwikkeling en onderhoud omdat informatiestroom is ontkoppeld. • Minimaal gebruik van bandbreedte.	• Informeren en verwijzen naar decentrale raadpleegfuncties vereist aanvullende harmonisatie van interfaces en toegang. • Vereist aanvullend mechanisme om te kunnen omgaan met gemiste events.
	Informatie-rijk	• Maakt alle informatie beschikbaar voor een consistente verwerking. • Directe overdracht van de toestand maakt decentrale raadpleeg-API's overbodig.	• Voor het opvragen los van gebeurtenis is een ander (aanvullend) mechanisme nodig. • Blijft in de kern een oplossing voor gedistribueerd data management.

13. Voorbeeld mogelijke toepassing Event Driven Architecture

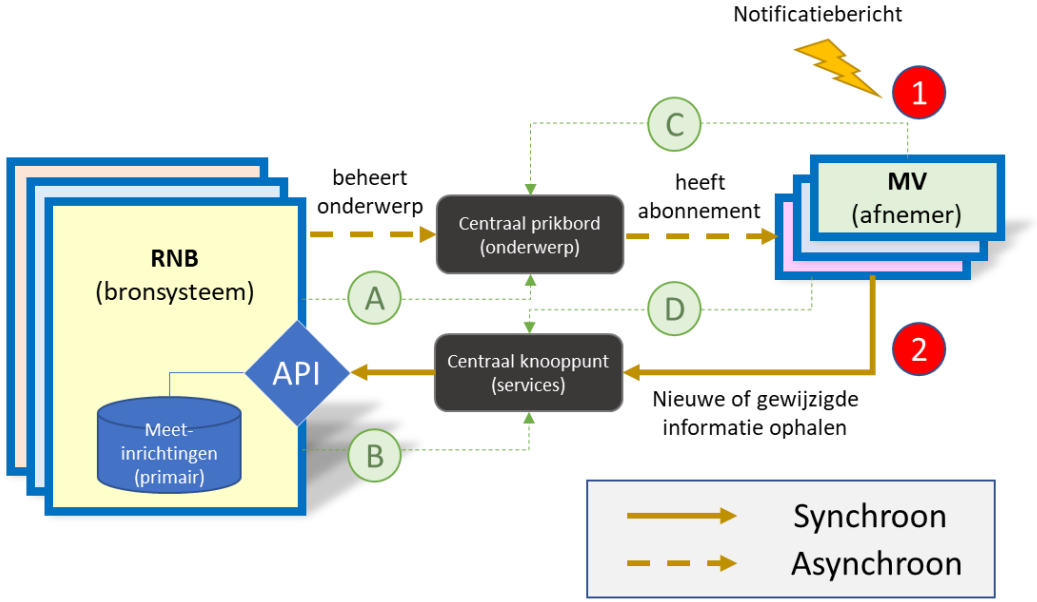
Use case: wijzigingen primair deel meetinrichting

- De Netbeheerder is verantwoordelijk voor het bijhouden van het primaire deel van de meetinrichting in een bronsysteem (decentraal register).
- **De meetverantwoordelijk moet direct bij een wijziging in het primaire deel van die wijziging op de hoogte worden gesteld, omdat deze geacht wordt daar (direct) actie op te ondernemen.**
- In de huidige situatie heeft de Meetverantwoordelijk toegang tot het PUD, en is deze continu aan het pollen, zowel om opgevraagde gegevens als spontaan door de Netbeheerder toegestuurde wijzigingen in de gegevens op te vragen
- Op basis van de visie in dit document wijzigt dit in twee separate gegevensuitwisselingen:

  (1) de Netbeheerder stuurt een event wanneer er sprake is van een wijziging, en de Meetverantwoordelijk haalt vervolgens de gewijzigde gegevens op zodat deze daar direct mee aan de slag kan, conform het EDA patroon.

  (2) de Meetverantwoordelijke haalt incidenteel of periodiek de voor zijn proces relevante gegevens op conform het API patroon.
- Omdat alleen de aangewezen Meetverantwoordelijke deze informatie mag inzien, zal hier sprake zijn van Messaging, waarin de notificatie expliciet aan de meetverantwoordelijke wordt geadresseerd.
- NB: voordat deze gegevensuitwisseling kan plaatsvinden, moet de MV zich wel abonneren (subscriben) op het juiste prikbord, en moet hij het juiste prikbord daarvoor eerst gevonden hebben.

Voorbeeld implementatie EDA patroon informatie-arm



Uitgangspunt bij deze implementatie is een informatie-arm event, en een API die door de NB ter beschikking wordt gesteld. Dit zou ook middels een informatie-rijk patroon geïmplementeerd kunnen worden.

14. Conclusies toepasbaarheid EDA

- Deze visie is ontstaan vanwege problemen met IC284A en IC284B, die niet optimaal kunnen worden opgelost met bestaande middelen en een nieuwe benadering vereisen.
- Een verbeterde gegevensuitwisseling tussen marktpartijen is cruciaal om de uitdagingen van de energiemarkt en de energietransitie aan te pakken.
- Moderne (cloud-) oplossingen worden steeds meer gebruikt, maar legacy-systemen vormen soms een obstakel voor integratie met de gegevensuitwisseling in de energiemarkt.
- Binnen de energiemarkt streeft men naar eenduidige bronnen van waarheid (registers), beheerd door registerbeheerders.
- De nieuwe energiewet zorgt voor complexere gegevensuitwisseling bijvoorbeeld vanwege meer marktrollen. Het toepassen van moderne architectuur-patronen, zoals Event Driven Architecture, kan de gegevensuitwisseling verbeteren.
- Soms is directe notificatie van wijzigingen in registers nodig, en een Event Driven Architecturepatroon biedt een efficiënte manier om deze wijzigingen snel bij de juiste partijen te krijgen.
- Dit patroon is gebaseerd op het publish-subscribe mechanisme, waarbij events worden geproduceerd en gedistribueerd via een event broker.
- Het EDA patroon wordt toegepast in situaties waarin de ontvanger (de consument) direct moet kunnen reageren op een gebeurtenis, een wijziging in de gegevens in een register.
- Het heeft verschillende voordelen, zoals snelle signaaloverdracht, real-time reacties, ontkoppeling van systemen, robuustheid, schaalbaarheid, interoperabiliteit, uitbreidbaarheid, recovery-mogelijkheden, compatibiliteit met cloudtechnologie en aansluiting bij de visie van de Nederlandse overheid.

Bijlage

Begrippen en Referenties

Begrippenlijst

Bron: https://github.com/VNG-Realisatie/notificatieservices/blob/main/docs/achtergronddocumentatie/notificatieservices_architectuur.pdf

Een van de hoofddoelen van project Notificatieservices (uitgevoerd in 2021 door VNG Realisatie in opdracht van het Ministerie van Binnenlandse Zaken) 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 sector bedoelde berichtenstandaard te beschrijven als om event-driven werken in het algemeen te beschrijven.

Bij ontwerp en inrichting van notificatieprocessen moeten op een aantal aspecten keuzes worden gemaakt. De begrippenlijst in de volgende slides geeft naast een overzicht van de CloudEvents termen en begrippen ook een korte beschrijving van enkele relevante aspecten.

Begrippenlijst – CloudEvents terminologie (1)

Occurence	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, Publisher	The "Producer" is a specific instance, process or device that creates the data structure describing the CloudEvent. Also: “Publisher”.
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, Subscriber	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. Also: “Subscriber”
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.

Begrippenlijst – CloudEvents terminologie (2)

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.
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.

Begrippenlijst – aspecten (1)

Push	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.
Pull	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.
Synchroon	Bij synchrone (technische) communicatie 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. Bij synchrone (functionele) interactie 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.
Asynchroon	Bij asynchrone (technische) communicatie tussen applicaties wacht de initiërende applicatie niet op de ontvangende applicatie (en kan dus doorgaan met andere activiteiten) . Als er al een antwoord komt, wordt dit op een later moment, via een nieuwe verbinding, naar de initiërende partij verstuurd. Bij asynchrone (functionele) interactie kunnen applicaties gegevens uitwisselen zonder dat ze gelijktijdig actief hoeven te zijn. Gegevens worden daarvoor als 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.

Begrippenlijst – aspecten (2)

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.

Type events

Type	Soort	Omschrijving
Notificatie	Informatiearm	Consumers ontvangen minimale gegevens met daarin opgenomen een verwijzing naar een service waar aanvullende gegevens zijn op te vragen;
Event-carried state transfer	Informatierijk	Consumers ontvangen alle relevante gegevens naar aanleiding van een gebeurtenis en kunnen direct na ontvangst tot verwerking daarvan overgaan. De event-carried state transfer (ECST) is onder te verdelen in 2 subtypes: 1. De ECST bevat een complete snapshot van de state (ookwel de ECST 'fat' of 'RESTful' event genoemd); 2. De ECST bevat alleen de gewijzigde velden (ookwel de ECST 'delta' event genoemd);
Mutatie	Informatierijk	Consumers ontvangen naar aanleiding van een gebeurtenis de oude en nieuwe gegevens;
Domain	Informatierijk	Private events binnen een specifieke bounded context en vanwege de Ubiquitous Language alleen relevant binnen de eigen context en niet bedoeld voor integratie met andere bounded contexten;

Belangrijke overige begrippen

Begrip	Definitie
Register	Gestructureerd geheel van gegevens die volgens bepaalde criteria toegankelijk zijn, ongeacht of dit geheel gecentraliseerd of gedecentraliseerd is, dan wel op functionele of geografische gronden is verspreid;
Register-beheerder	Partij die op grond van de artikelen 4.5, 4.6 of 4.7 of krachtens artikel 4.12 een register bijhoudt;

Bronnen en Referenties

- [1] https://aws.amazon.com/event-driven-architecture/
- [2] https://learn.microsoft.com/en-us/azure/architecture/guide/architecture-styles/event-driven
- [3] https://nordicapis.com/rest-apis-and-event-streams-taste-better-together
- https://martinfowler.com/bliki/BoundedContext.html (zie zijn 'Meter' voorbeeld)
- https://nordicapis.com/tag/event-driven-architecture/
- https://tdonker.github.io/domain-driven-design-links/
- https://github.com/VNG-Realisatie/notificatieservices project Notificatieservices VNG Realisatie
- https://martinfowler.com/articles/201701-event-driven.html
- https://serverlessland.com/event-driven-architecture/visuals/event-types
- https://itnext.io/the-event-carried-state-transfer-pattern-aae49715bb7f
- https://github.com/VNG-Realisatie/notificatieservices/blob/main/docs/achtergronddocumentatie/notificatieservices_architectuur.pdf
- https://github.com/cloudevents/spec/blob/v1.0.1/spec.md#terminology

Nadere toelichting business versus technische Event Driven Architecture

Het verschil tussen business en technische Event Driven Architecture (EDA) kan worden begrepen aan de hand van het onderscheid tussen business events en technische events. Laten we dit nader bekijken:

1. Business Event Driven Architecture:

- Business events in de context van EDA verwijzen naar gebeurtenissen of signalen die relevant zijn voor zakelijke processen, besluitvorming en activiteiten binnen een organisatie. Deze gebeurtenissen zijn meestal gerelateerd aan de bedrijfsactiviteiten en doelstellingen van een organisatie.
- Enkele voorbeelden van business events in de energiesector zijn:
- Klantaanmeldingen: Wanneer een nieuwe klant zich aanmeldt voor een energiedienst, wordt dit beschouwd als een business event. Het kan leiden tot processen zoals het genereren van een klantaccount en het instellen van facturering.
- Energiemarktfluctuaties: Veranderingen in energieprijzen, vraag en aanbod worden beschouwd als business events. Dit kan van invloed zijn op beslissingen met betrekking tot energie-inkoop en -verkoop.
- Naleving van regelgeving: Wanneer er wijzigingen zijn in de regelgeving met betrekking tot de energiesector, zoals emissievoorschriften, worden deze als business events beschouwd. Ze kunnen leiden tot aanpassingen in bedrijfsprocessen en rapportagevereisten.
- In een Business EDA-architectuur worden deze business events vastgelegd, verwerkt en verspreid om real-time inzicht te bieden in zakelijke activiteiten en om zakelijke processen en besluitvorming te sturen.

2. Technische Event Driven Architecture:

- Technische events zijn daarentegen gericht op gebeurtenissen die zich voordoen in de technische infrastructuur en systemen van een organisatie. Deze events hebben betrekking op de werking en het beheer van technologie, zoals netwerken, servers, sensoren en apparaten.
- Enkele voorbeelden van technische events in de energiesector zijn:
- Storingen in energiecentrales: Als er een storing optreedt in een energiecentrale, genereert dit technische events. Deze events kunnen betrekking hebben op parameters zoals temperatuur, druk en vermogensniveaus.
- Datastromen van slimme meters: Het uitlezen van metingen van slimme meters in realtime genereert technische events. Deze gegevens zijn essentieel voor het monitoren van het energieverbruik en de netwerkprestaties.
- Netwerkverkeer: Het vastleggen van netwerkverkeer en het identificeren van mogelijke bedreigingen en beveiligingsincidenten zijn voorbeelden van technische events in cybersecurity.
- In een Technische EDA-architectuur worden deze technische events vastgelegd, verwerkt en verspreid om de prestaties, veiligheid en betrouwbaarheid van technische systemen te bewaken en te beheren.

In essentie draait het verschil tussen Business EDA en Technische EDA om de aard van de gebeurtenissen die worden vastgelegd en gebruikt. Business EDA richt zich op gebeurtenissen met betrekking tot zakelijke activiteiten en processen, terwijl Technische EDA zich richt op gebeurtenissen die verband houden met technische infrastructuur en systemen. Het doel van beide benaderingen is echter om real-time inzicht en sturing te bieden in respectievelijk zakelijke en technische aspecten van een organisatie.