Werkt voor provincies

Bijlage 14 - API kaders en richtlijnen

8 maart 2023
Versie Definitief

Werkt voor provincies

Colofon

Document informatie	
Titel	Bijlage 14 - API kaders en richtlijnen
Auteur	J.P. Kloosterman
Versie	Definitief
Status	Definitief
Datum	8 maart 2023
Bestandsnaam	*API kaders en richtlijnen*
ISO Document	Kaders en richtlijnen API's
(ISO) Proces	Gebruik van API's

	Naam	*Rol*
Documenteigenaar	J.P. Kloosterman	Information Security Officer
Proceseigenaar	J.P. Kloosterman	Information Security Officer
Procesverantwoordelijk	J.P. Kloosterman	Information Security Officer

Versiebeheer/wijzigingshistorie				
Versie	*Status*	*Datum*	*Beschrijving*	*Auteur*
0.1	Concept	3-2-2023	Eerste opzet	J.P. Kloosterman
0.2	Review	2-3-2023	Reviewcommentaar ver-werkt	J.P. Kloosterman
1.0	Defini-tief	2-3-2023	Definitieve versie	J.P. Kloosterman

Managementsamenvatting

BIJ12 heeft een belangrijke gegevensuitwisselingstaak en daarvoor worden Web Services ingezet. Web services moeten stabiel en veilig zijn. Daarvoor is dit document opgesteld waarin de kaders en richtlijnen zijn vastgelegd om zo goed én zo veilig mogelijk Web Services aan te bieden.

Zaken die organisatorisch geregeld moeten worden, staan niet in dit document. Niet alles is met techniek op te lossen. Er zullen afspraken en keuzes op het gebied van beschikbaarheid, integriteit, vertrouwelijkheid, archivering, beheer, onderhoud en verantwoordelijkheden gemaakt moeten worden.

Wat we de afgelopen drie jaar ervaren hebben, is dat de wereld heel snel kan veranderen. Daarom moeten de Web Services continu gemonitord en periodiek getest, gescand en bijgeschaafd worden. Patchmanagement speelt hier een hele belangrijke rol, want niet bijblijven is een risico die BIJ12 zich niet kan permitteren.

Inhoudsopgave

1 Inleiding

Application Programming Interfaces (API's) kunnen gegevens beschikbaar stellen en uitwisselen tussen verschillende informatiesystemen. Voor BIJ12 zijn deze essentieel om te gebruiken en voor een goede werking horen kaders en richtlijnen bij.

In dit document worden de kaders en richtlijnen voor API's beschreven. De beschreven kaders en richtlijnen zijn dwingend. Wanneer een implementatie hiervan afwijkt, moet het altijd vooraf worden besproken en zal er een afwijkingsverklaring opgesteld moeten worden.

Afwijkingen zijn altijd tijdelijk en alleen onder uitzonderlijke voorwaarden mogelijk, zijn altijd goed onderbouwd en er zijn uitvoerbare afspraken gemaakt die ook zijn vastgelegd.

2 Kaders

Kaders die hier gehanteerd worden, zijn:

1. Een Web Service is veilig[1];
2. Een Web Service wordt continu beheerd en onderhouden (patchmanagement);
3. Een Web Service wordt gemonitord en heeft een auditlog;
4. Een Web Service is gebaseerd op de *Representational State Transfer* (REST of RESTful) architectuur;
5. Elke Web Service heeft een berichtenboek en er is een wijzigingsproces vastgesteld.
6. Elke Web Service heeft een doel een eigenaar en de doelgroep is vastgelegd;
7. Elke Web Service heeft gepubliceerde gebruikersvoorwaarden;
8. Elke Web Service hanteert de OpenAPI specificatie;
9. Elke Web Service volgt de Nederlandse API Strategie I (API Strategie Algemeen) en daaruit volgend:
    a. de API Design Rules 1.0
    b. de Extensie
    c. het OAuth profiel
10. Webservices zijn vindbaar en zo nodig via een centrale locatie toegankelijk te maken.
11. Security by design methodiek wordt toegepast;
12. Indien van toepassing: privacy by design methodiek wordt toegepast.

De bovenstaande uitgangspunten moeten ervoor zorgen dat Web Services veilig, actueel en op de agenda blijven.

[1] De eisen die een Web Service veilig maken komen later in dit document aan bod.

3 Richtlijnen

Veilige Web Services voldoen aan de volgende richtlijnen:

1. Het garanderen van **vertrouwelijkheid** van Web Service bericht door middel van adequate versleuteling;
2. Het garanderen van **integriteit** van Web Service bericht wordt door middel van handtekeningen/signatures;
3. Het garanderen van **beschikbaarheid** wordt door middel van DDoS mitigerende maatregelen en data replicatie toegepast + redundantie;
4. **Authenticatie** en **Autorisatie** van personen via SAML en OAuth;
5. **Netwerk beveiliging** VPN/access list/ Reverse proxy worden toegepast waar mogelijk. Eventueel toepassen van een Web Application Firewall om onveilige berichten te filteren. Altijd end-to-end-encryptie.

Bij het gebruik van Web Services moeten de volgende zaken beschreven, vastgelegd en geïmplementeerd zijn:

1. Afwijzen van transacties;
2. Veilig verstrekken van inloggegevens;
3. Tegengaan van misbruik van (verborgen) kanalen;
4. Handelingsperspectief van gecompromitteerde diensten;
5. Voorkomen van het verspreiden van malware;
6. Dienst ontoegankelijk maken (Denial of services aanvallen) tegengaan;
7. Incorrecte diensten implementatie tegengaan door Secure Software methoden en technieken te gebruiken;
8. Vooraf is bepaald hoe de web service moet presteren en hier wordt op gemonitord en gerapporteerd.

4 Bijlage A: TLS Cipher Suite

Hieronder staan de minimale instellingen voor het gebruik van TLS 1.2 of TLS 1.3:

Omschrijving	**Waarden/instellingen**
Key Exchange	RSA, Diffie-Hellman, ECDH, SRP, PSK
Authentication	RSA, DSA, ECDSA
Bulk Ciphers	RC4, 3DES, AES
Message Authentication	HMAC-SHA256, HMAC-SHA1, HMAC-MD5

Minder sterke instellingen worden **nooit** gebruikt!
TLS 1.0 en 1.1. worden **niet** gebruikt!

Geen enkele versie van SSL wordt gebruikt!

5 Bijlage B: API's en de standsaarden op de Pas-toe-of-leg-uit-lijst (PTOLU-lijst) (bron: forum standaardisatie)

Vanuit de overheid zijn de volgende standaarden van toepassing:

Standaard	Invloed REST	Uitleg
Aquo-standaard	Ja	Gegevensverzameling, -vastlegging en -uitwisseling voor het beheer van waterkeringen, oppervlaktewater en afvalwaterzuivering. Dit zou ook via een RESTful API kunnen en met JSON formaten. Overigens worden er ook bestandsformaten via JSON aangeboden
BWB	Ja, gebruikt SOAP/WSDL/ XML	Elektronische verwijzing naar (delen van) geconsolideerde wetten en regelingen met het doel om deze met anderen te delen. De BWB web-services is gebaseerd op WSDL en SOAP.
CMIS	Ja, ondersteunt al REST	Het toegankelijk maken van ongestructureerde gegevens in content CMS'enDMS'en. Oorspronkelijk op SOAP gebaseerd, maar er zijn drie protocolbindingen gedefinieerd. 1 voor SOAP, 1 voor Atompub (RESTful XML) en 1 voor JSON. Overigens moet voor CMIS altijd aanvullende afspraken worden gemaakt .
Digikoppeling	Ja	Geautomatiseerde gegevensuitwisseling tussen informatiesystemen voor sectoroverstijgend berichtenverkeer,
DKIM	Nee	Het faciliteren van het vaststellen van organisatorische herkomst van e-mail. Geen specifieke relatie met REST.
DNSSEC	Nee	Het registreren en in DNS publiceren van internet-domeinnamen ('signing'). Hierdoor is validatie van het domeinnaam mogelijk. Geen specifieke relatie met REST.
E-portfolio NL	Nee	Het uitwisselen van informatie over de ontwikkelingsvoortgang van een individu, Is wel gebaseerd op XML en de IMS specificatie. Deze kan echter via Restful principe of SOAP worden uitgewisseld.
ECLI	Nee	Identificatie ter citatie van gepubliceerde rechterlijke uitspraken. Zegt niks over de protocolbindingen.
EML_NL	Nee	De definitie en uitwisseling van kandidaatgegevens en uitslaggegevens bij verkiezingen welke onder de Nederlandse Kieswet vallen. Uitwisseling kan alleen als XML bestand en niet als JSON, maar zegt niks over de hoe het uitgewisseld moet worden.

Geo Standaarden	Ja, zie uitleg hierboven	Uitwisseling van geografische informatie tussen organisaties, waarbij de ruimtelijke dimensie van significant belang is
IFC	Ja,	Uitwisseling in het kader van bouwwerkinformatiemodellen. IFC kent al een RESTful API
IPv6 en IPv4	Nee	Voor communicatie van computernetwerken over organisatiegrenzen heen tussen organisaties, individuele eindgebruikers, apparaten, diensten en sensoren. Zegt niks over REST of SOAP.
JCDR	Nee	Identificatie van geconsolideerde decentrale regelgeving en een gestandaardiseerde manier om hiernaar elektronisch te verwijzen zegt niks over REST of SOAP.
NEN-ISO/IEC 27001 / 2	Nee	Specificeren van eisen voor het vaststellen, implementeren, uitvoeren, controleren, beoordelen, bijhouden en verbeteren van een gedocumenteerd Information Security Management System. Is een organisatorische standaard en zegt in principe niks over de techniek.
NL LOM	Nee	Metadatering van content die ontsloten wordt ten behoeve van educatieve doeleinden. Maakt wel gebruik van XML schema's, maar zegt niks over het aanbieden en uitwisselen.
NTA 9040	Ja,	De standaard is van toepassing voor overheden die expliciet met ondernemingen hebben afgesproken een Ondernemingsdossier in te zetten voor de informatie-uitwisseling met ondernemingen. De standaard heeft een relatie, schrijft namelijk expliciet een WSDL voor.
OAI-PMH	Ja	Het vraaggestuurd aanbieden en ophalen van verzamelingen metadata uit bibliotheken met (digitale) documenten of andere objecten. Een protocol gebaseerd op het REST principe.
ODF 1.2	Nee	Uitwisseling van reviseerbare documenten en zegt niks over de manier van uitwisselen.
OWMS	Ja	Metadateren van publieke overheidsinformatie op internet. Is syntax neutraal en net zoals OAI-PMH gebaseerd op de internationale Dublin Core standaard. Sluit goed aan op LinkedData wat goed samen gaat met het RESTful API.
PDF 1.7, A1 en A2	Nee	Het uitwisselen, publiceren en archiveren van niet- of beperktreviseerbare documenten. Zegt niks over de manier van uitwisselen.
SAML	Ja	Federatieve (web)browser-based single-sign-on (SSO) en single-signoff. Dat wil zeggen dat een gebruiker na eenmalig inloggen via zijn browser toegang krijgt tot verschillende diensten. Voor de relatie zie uitleg hierboven.

Semantisch model ehttps://lijsten.forumstandaardisatie.nl/open-standaard/semantisch-model-e-factureren factureren	Nee	De verzending en ontvangst van elektronische facturen door organisaties die deelnemen aan het economisch verkeer in Nederland. Is techniek neutraal.
SETU-standaard	Ja	Is een XML standaard voor elektronische berichtenuitwisseling rondom de bemiddeling/inhuur van flexibele arbeidskrachten. Zegt niet iets over de manier van uitwisselen, kan dus zowel via een RESTful API als via SOAP verbindingen.
SIKB0101 & SIKB0102	Ja	Uitwisselen van onderzoeksgegevens over de milieuhygiënische kwaliteit van de bodem. Voor SIKB0101 is er een specifiek SOAP webservice voorgeschreven, geen gebaseerd op REST.
SKOS	Ja	Het in een gestructureerde vorm op het Web publiek beschikbaar stellen van gegevenswoordenboeken. Is een typisch standaard die goed aansluit op REST, zijn ook RESTful APIs voor SKOS.

SPF	Nee	Het controleren of een e-mailserver gerechtigd is om namens een domeinnaam e-mail te mogen verzenden. Geen specifieke relatie met REST.
STOSAG	Ja	Digitaal container- en pasmanagement voor afval en grondstoffen, maakt gebruik van SOAP/WSDL en XML.
StUF	Ja	Uitwisseling en bevraging van basisgegevens die behoren tot een aantal wettelijk vastgestelde basisregistraties, zoals Personen (GBA), Adressen (BRA), Gebouwen (BGA), Kadaster (BRK), Nieuw Handelsregister (NHR) en Waarde Onroerende Zaken (WOZ). Zie uitleg hierboven.
TLS	Nee	Het met behulp van certificaten beveiligen van de verbinding (op de transportlaag) tussen client- en serversystemen of tussen serversystemen onderling. Geen specifieke relatie met REST.
VISI	Ja	Formele communicatie tussen partijen in de bouwsector, zowel grondweg en waterbouw, de burger & utiliteitsbouw als de installatiebranche. Heeft richtlijnen met betrekking tot uitwisseling gebaseerd op SOAP.
WDO Datamodel	Misschien	Gegevensuitwisseling tussen het bedrijven dien bij grensoverschrijdend goederenverkeer betrokken zijn. Gaat via XML berichten, maar niks terug te vinden over SOAP of RESTful.

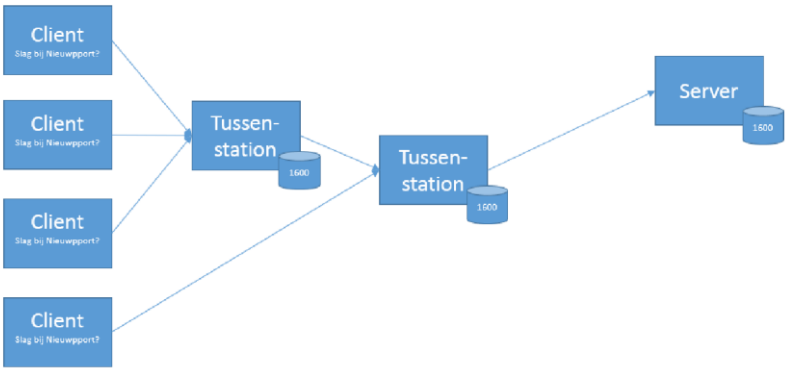
Webrichtlijnen	Nee	Voor de toegankelijkheid van webgebaseerde informatie-, interactie-, transactie- en participatiediensten. Geen specifieke relatie met SOAP of REST

XBRL en Dimensions	Ja	Elektronisch verkeer dat te kenmerken is als verantwoordingsverkeer waarin financiële informatie de kern vormt. Gaat over XML berichten en wordt met name gebruikt in combinatie met SOAP.

6 Bijlage C: REST architectuur (bron: forum standaardisatie)

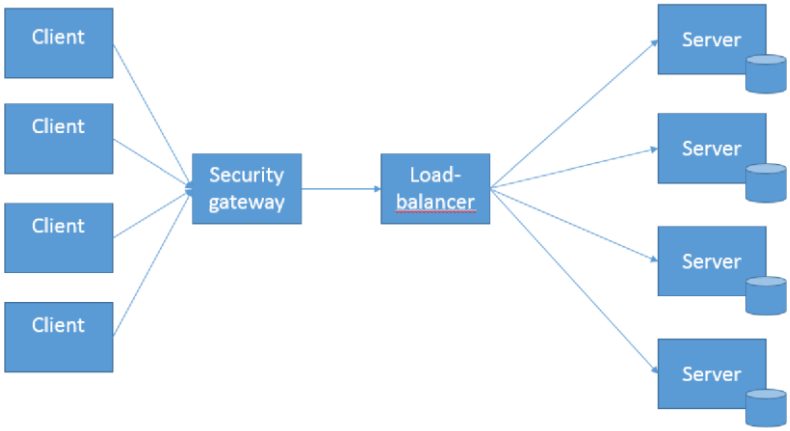
REST bereikt zijn belangrijkste eigenschappen (prestaties, schaalbaarheid en simpele interfaces) door een aantal beperkingen (contraints) op te leggen:

- **Expliciet onderscheid tussen Client en Server**. Zodat beide zich goed op hun taak kunnen richten. De client stelt vragen aan de server en houdt zelf bij waar het mee bezig is, waar in het proces de vraag zich bevindt (state) en wat het aan een gebruiker presenteert (user interface).
  De server beantwoordt de client en houdt zich bezig met dataopslag. Hoe de server data opslaat hoeft de client niet te weten, alleen dat hij doet en dat het op de server op te vragen is. Deze duidelijke scheiding verhoogt de schaalbaarheid. Je kan Eenvoudig het aantal clients en servers uitbreiden zonder dat dit grote impact heeft op de prestaties.
- **Stateless:** De server houdt niets bij over waar de verschillende clients waar het mee communiceert mee bezig zijn. De server hoeft alleen de vragen van het huidige moment te beantwoorden en verder nergens rekening mee te houden. Dit bevordert de prestaties.
- **Cacheable:** Bij antwoorden die een server geeft kan het expliciet aangeven of deze cacheable zijn. Veelvoorkomende vragen kunnen zo veel sneller uit cache beantwoord worden. Dit verhoogt de prestaties en schaalbaarheid enorm.



In bovenstaand plaatje wordt caching geïllustreerd. Stel een heleboel clients willen weten wanneer de slag bij nieuwpoort was. Het antwoord is iedere keer hetzelfde en daardoor zeer geschikt voor caching. De eerste keer dat de vraag wordt gesteld wordt het antwoord bij de server opgehaald. Daarna kunnen tussenstations die (voor de gebruiker niet zichtbaar) gebruikt worden in het transport dit antwoord opslaan in hun eigen cache. Ze kunnen de volgende clients die dezelfde vraag stellen daarmee veel sneller antwoord geven. De server wordt daarbij ontzien zodat hij meer capaciteit over heeft om moeilijke vragen snel te verwerken.

- **Layered system:** Een client kan niet zien of hij rechtstreeks met een server praat of dat er tussenpartijen tussen zitten zoals load balancers, (security) gateways etc… welke worden gebruikt om de schaalbaarheid te vergroten en welke prestaties en schaalbaarheid verbeteren door caching toe te passen.



In het plaatje hierboven zie je een layered system uitgewerkt. Iedere van de clients stelt zijn vraag maar heeft geen idee dat er eigenlijk 4 servers, een loadbalancer en een security gateway betrokken zijn. Hij wordt gewoon doorgeleid naar 1 van de 4 servers en krijgt daar antwoord van zonder dat hij iets van de infrastructuur hoeft te weten. Zo kan je de implementatie van een webservice heel makkelijk opschalen naar meer capaciteit.

- **Code on demand (optioneel):** Een server kan naar een client code sturen om lokaal uit te voeren bijvoorbeeld applets, of java script. Dit hoeft niet toegepast te worden (het is optioneel).

6.1.1 *Uniforme interfaces*

De interfaces waarmee communicatie tussen componenten plaatsvindt is zoveel mogelijk uniform. Hierbij wordt er gezorgd voor een ontkoppeling tussen client en server. Daarvoor zijn er een aantal ontwerp principes voor interfaces ○ **Identificatie van resources**

Resources, bijvoorbeeld data worden in berichten teruggegeven als URIs die verwijzen naar de resource zelf. De representatie van resources zoals teruggeven van de server is daarbij volledig onafhankelijk van de interne representatie die een server gebruikt. Dus een server kan data opslaan in een SQL database maar teruggeven aan de cliënt in JSON, XML of een ander formaat dat de Client graag wil hebben. De daadwerkelijk resource en wat de Client krijgt is zo ontkoppeld. Dit zorgt dat er gemakkelijk en flexibel met uitwisselingsformaten omgegaan kan worden. ○ **Manipulatie van resources**

Een client die in het bezit is van een representatie van een resource(een stukje XML of JSON gekregen van de server) heeft daarmee alle informatie die nodig is om deze resource te manipuleren: veranderen of verwijderen. Een client kan snel en eenvoudig die handelingen verrichten die hij wil verrichten en hoeft hiervoor niet eerst pagina's met documentatie door te spitten waarin toegestane functionaliteit staat beschreven. ○ **Zelf beschrijvende berichten**

Ieder bericht bevat in zichzelf genoeg informatie zodat duidelijk is hoe dit bericht verwerkt kan worden. Het bericht geeft bijvoorbeeld zelf aan dat het in JSON formaat staat en met een JSON parser verwerkt kan worden.

6.1.2 ○ *Hypermedia as the engine of application state*

Het klinkt ingewikkeld maar wat hier wordt bedoeld is simpel. Client machines die met REST interfaces communiceren gedragen zich net zo als iemand die over HTML pagina's (een vorm van hypermedia) navigeert met een browser. Je kan alleen van de ene pagina naar de andere (van de ene state naar de andere) komen door het volgen van een hyperlink. Je neemt als client aan dat de server je alle informatie biedt die je nodig hebt om ergens te komen. Alles is expliciet en je

hoeft er niet vanuit te gaan dat er meer mogelijk is dan dat wat je gepresenteerd wordt. Je wordt door de server stap voor stap via hyper links door de mogelijke pagina’s(states) geleid.