whitehorses

member of the human network

WHITEBOOK

Rest Services in de Oracle Service Bus

Juli 2021

Auteur:
Maurik-Jan Veenman
INTEGRATIESPECIALIST

Inleiding

Bij mijn huidige klant worden diverse koppelingen gelegd vanuit de OSB naar omliggende systemen. Daarbij wordt een breed palet aan protocollen gebruikt. Een aantal daarvan zijn REST services, waarbij van verschillende technieken gebruik wordt gemaakt. Omdat ik merkte dat de Oracle documentatie over dit onderwerp niet uitblinkt in leesbaarheid, en ook bij de bekende bloggers vaak alleen het standaard Oracle oefenrecept wordt behandeld, leek het me een goed idee om de verschillende mogelijkheden en hun toepassing op een rijtje te zetten. En zo ontstond dit Whitebook!

Wat is REST?

REST staat voor Representational State Transfer. Het is een manier om een interface naar een programma te beschrijven. Je kunt erin vastleggen welke methode van het programma je wilt aanroepen, welke in- en uit-parameters er zijn en deze vullen met gegevens. Dit kun je vastleggen in een WADL of een Swagger document. Een WADL (Web Application Description Language) is vergelijkbaar met een WSDL bij een SOAP interface. Belangrijk voordeel ten opzichte van SOAP: HTTP Verbs en betere feedback.
Er zijn verschillende Verbs gedefinieerd, waarmee alle CRUD acties mogelijk worden gemaakt.

GET haalt de gezochte gegevens op van de service. Dit kan resulteren in een HTTP 200 response (object gevonden) met het object in de response body (in XML of JSON vorm). Wanneer het object niet wordt gevonden ontvang je een 404 (object niet gevonden), en je krijgt een 400 (bad request) als je vraag onjuist gedefinieerd is.

POST voegt een nieuw object toe aan de verzameling objecten van de service. Als dat succesvol is gebeurd, ontvang je een 201 response (created) met een locatieverwijzing naar het object. Mocht het niet resulteren in een URL die naar het object wijst, dan ontvang je een 200 (ok) of een 204 (no content).

PUT wordt gebruikt voor het updaten van een bestaand object. Als het object niet bestaat, creëert het een nieuw object. Ook hier kun je weer de antwoorden krijgen als bij de POST variant (201 (gecreëerd), 200 of 204 (beiden voor updated)).

DELETE verwijdert een object van de verzameling. Mogelijke responses zijn hier 200 (ok), 202 (accepted) voor als de opdracht in de queue is gezet en 204 (no content) voor als de actie wel is uitgevoerd, maar er geen identifier teruggegeven wordt.

PATCH doet ook een update, maar in tegenstelling tot een PUT opdracht (die dient om een object volledig te vervangen voor iets nieuws) dient PATCH voor het gedeeltelijk aanpassen van het object (bijvoorbeeld alleen het telefoonnummer van het object ‘persoon’). Ook hier krijg je weer 200 (ok), 204 (no content) of 404 (object niet gevonden) terug.

Welke mogelijkheden van parameter overdracht naar REST calls zijn er, en wanneer worden ze gebruikt?

Ik ben er tot nu toe drie tegengekomen, en weet eerlijk gezegd niet of dit alle mogelijkheden zijn. Gezien het feit dat je de combinatie van gebruik van de verschillende onderdelen van de opbouw van de aanroep kunt variëren, betekent dat er meer mogelijkheden zijn.

Je hebt de 'klassieke Oracle-methode' met query parameters, die worden meegegeven in de aanroep naar de Business Service. Deze wordt meestal uitgelegd in de oefenrecepten. Deze methode kan gebruikt worden wanneer je de interface aanroept met een URL, en apart de query parameters meegeeft in de aanroep.

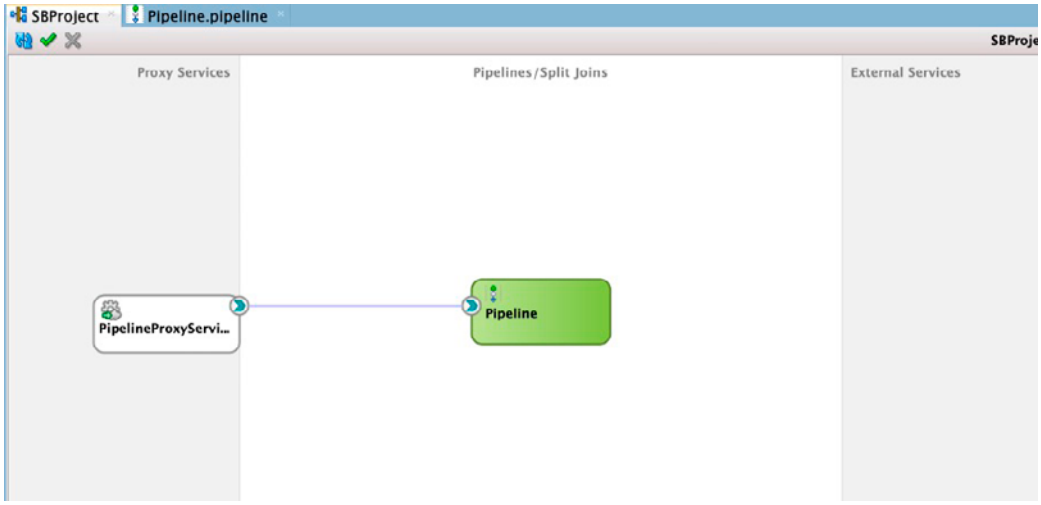
Daarnaast had ikzelf in mijn project behoefte aan een methode, waarbij een van de parameters zich midden in de URL van de aanroep bevond (dit is dan een path parameter). Dit werkte eigenlijk alleen maar goed als je vervolgens ook de 'template' methode voor parameteroverdracht gebruikt.

Als derde methode wordt ook de 'template' methode gebruikt. Hier wordt echter een nXSD transitie gebruikt om de body van het request op te bouwen. Als extra wordt ook het meegeven van een API key in de aanroep behandeld. Dit is een vooraf met elkaar afgesproken code die de aanroepende partij mee moet geven voor de beveiliging van de service, zodat niet iedereen de service aan kan roepen.

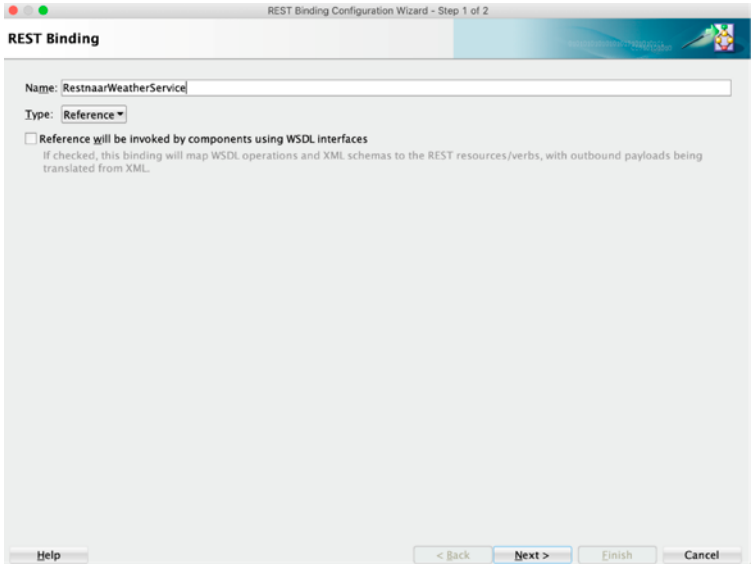
Hoe gaat het in zijn werk?

Variant 1, de 'klassieke Oracle-methode'

Normaliter maak je aan het einde van je pipeline een routering (of ergens halverwege een callout) naar de Business Service die de REST Service aanroept, en geef je de query parameters mee om het specifieke antwoord wat je zoekt te verkrijgen. Om deze Business Service en het meegeven van de query parameters voor elkaar te krijgen doe je het volgende: In je Service Bus projectoverzicht maak je aan de rechterkant een 'external service' aan van het type REST door deze in je project te slepen:



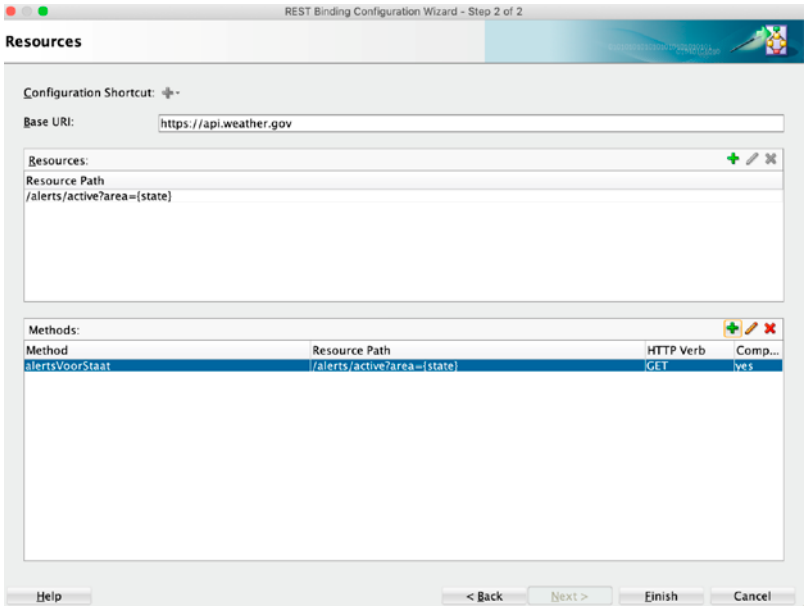
Er wordt nu een configuration wizard gestart. Geef op de eerste tab de service een zinvolle naam:



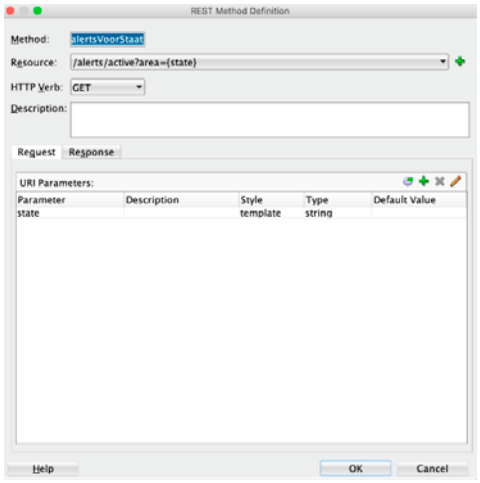
Laat het type op 'reference' staan.

We gaan voor dit voorbeeld een koppeling leggen naar de alerts van de openbare weerservice van Amerika. Dit is de URL waar we naar toe moeten: https://api.weather.gov/alerts/active?area={state}

Deze wordt verdeeld over onderstaande tab ingevuld. Bij base URL geef je de website zelf aan: https://api.weather.gov, bij de resources het relatieve pad naar de methode die je aan wilt roepen (alerts, die wij op een volgend tabblad 'alertsVoorStaat' hebben genoemd)

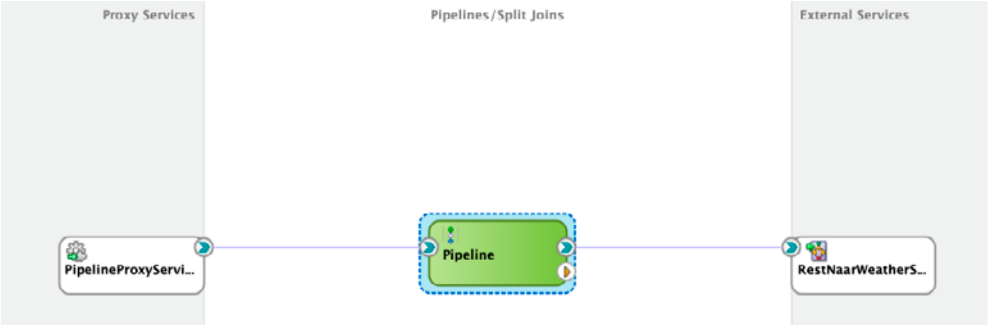


Op het method tabblad geven we de methode een eigen naam, en kunnen we de mee te geven parameter opgeven. Deze wordt dan runtime gesubstitueerd in de {state} query variabele in de URL:

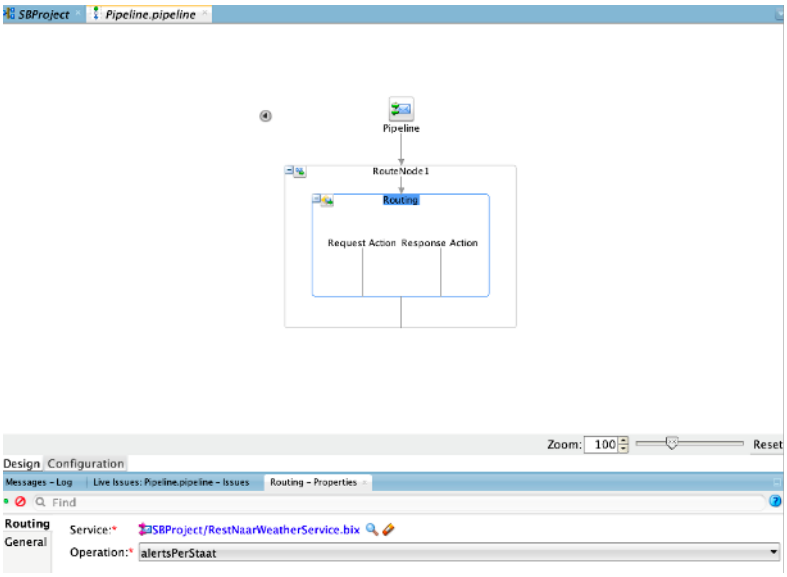


We zijn nu klaar met het doorlopen van de configuration wizard. We hebben nu een Business Service gecreëerd en een WADL waarmee de URL en parameters worden gedefinieerd.

Trek nu in het OSB projectoverviewscherm een lijntje van het pijltje van de REST Service naar de pipeline:

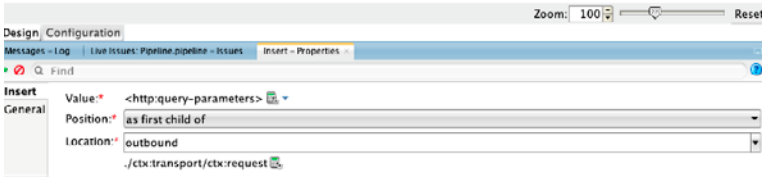


Hierdoor wordt een routeringactie in de pipeline toegevoegd:



De routering verwijst naar de Business Service van de REST Service, en specifiek naar de methode ‘alertsPerStaat’. Prima dus. Nu moeten we nog de query parameter meegeven.

Voeg hiervoor een ‘insert’ actie toe in de routering:



Je voegt hiermee aan de ‘outbound properties’ een query parameter toe, waarin je de waarde van de staat meegeeft aan de ‘state’ variabele. Dit doe je door de volgende value toe te voegen in bovenstaande scherm:

```
<http:query-parameters>
    <http:parameter name="state" value="{$inbound/ctx:transport/ctx:request/
http:query-parameters/http:parameter[@name ='staat']/@value}"/>
</http:query-parameters>
```

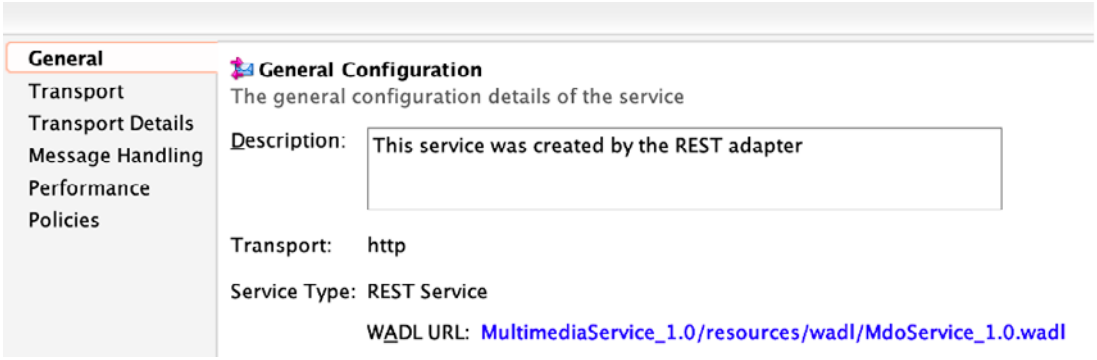
Hiermee wordt de waarde uit de in de pipeline binnenkomende ‘staat’ parameter in de ‘state’ query parameter van de Business Service gestopt.

Nu ben je klaar. De service zal nu de alerts voor de in de query parameter opgegeven staat teruggeven als hij wordt aangeroepen.

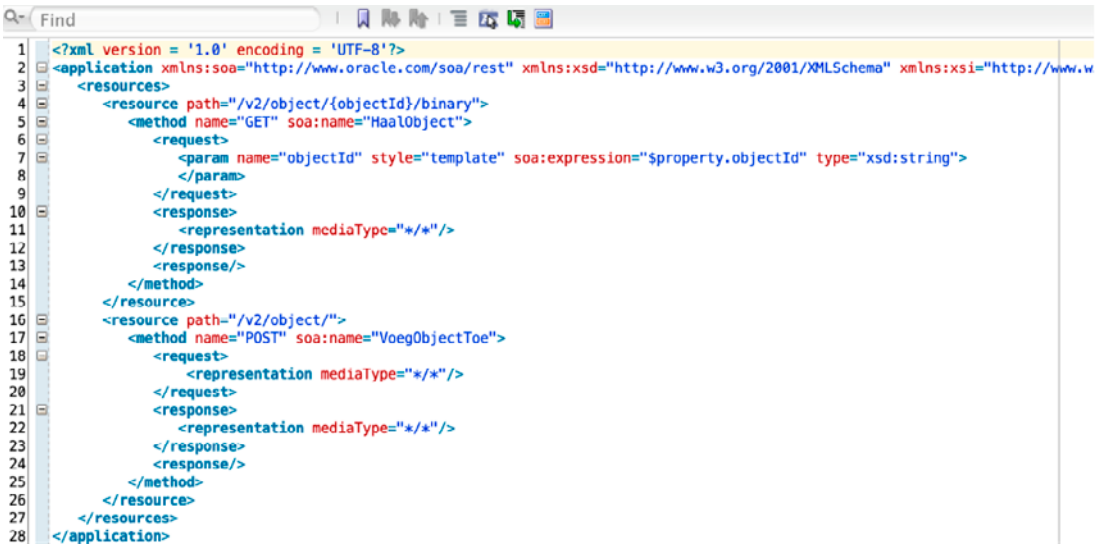
Variant 2, de path parameter methode'

Maak net als bij de klassieke methode een Business Service aan, die verwijst naar de aan te roepen REST Service. We gaan nu in de aangemaakte WADL de URL aanpassen om de path parameter (die zich midden in de URL bevindt) te kunnen beïnvloeden.

In de Business Service geef je bij 'Transport' de URL van de REST Service op, en bij 'General' verwijs je naar de WADL:



In de WADL zelf neem je de parameter op (regel 7), en voegt hem toe aan de URL van de aan te roepen service (regel 4):



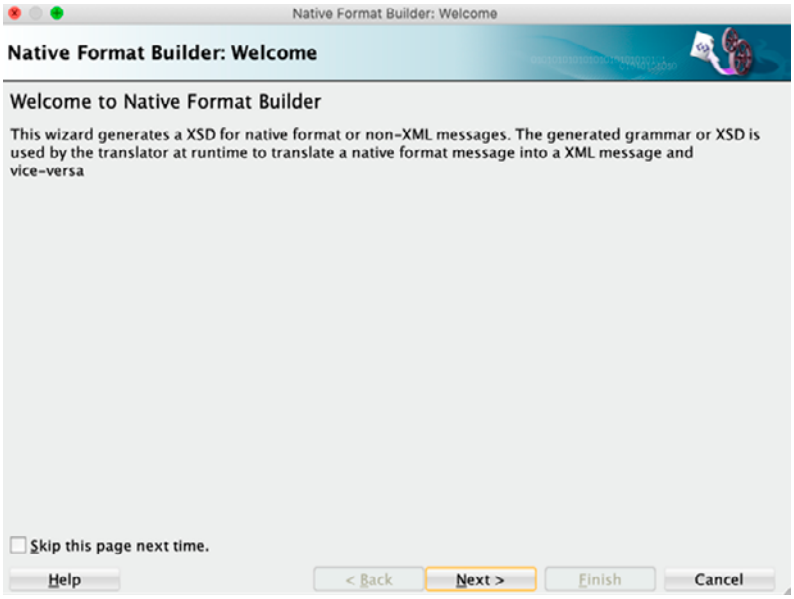
In je pipeline maak je een service callout naar de Business Service die je hierboven hebt gecreëerd, en hierin neem je een insert op waarmee je de waarde meegeeft aan de outbound properties:

Hierdoor wordt de waarde van de parameter 'objectid', bijvoorbeeld '123x' gesubstitueerd in de URL van de aangeroepen service: ~/v2/object/123x/binary, en daarmee het juiste object opgehaald.

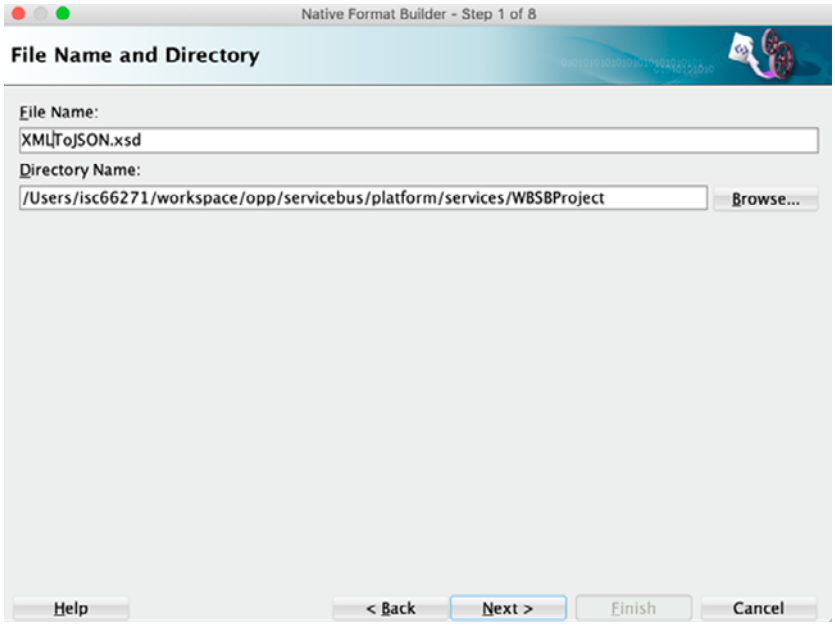
Variant 3, de 'nXSD methode'

In OSB 12c bestaat nu de mogelijkheid om direct van een XML bericht naar native JSON (of andere gewenste formaten) te converteren. Hierbij wordt dan ook de SOAP body verwijderd. Dit gaat als volgt:

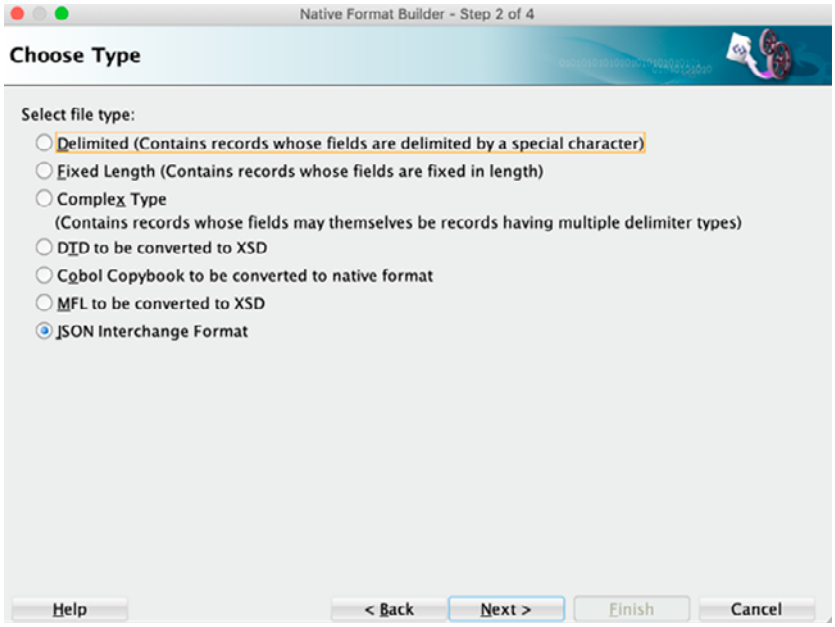
Maak eerst een nXSD schemabestand aan. Hierin leg je vast hoe het native formaat er uit moet komen te zien:



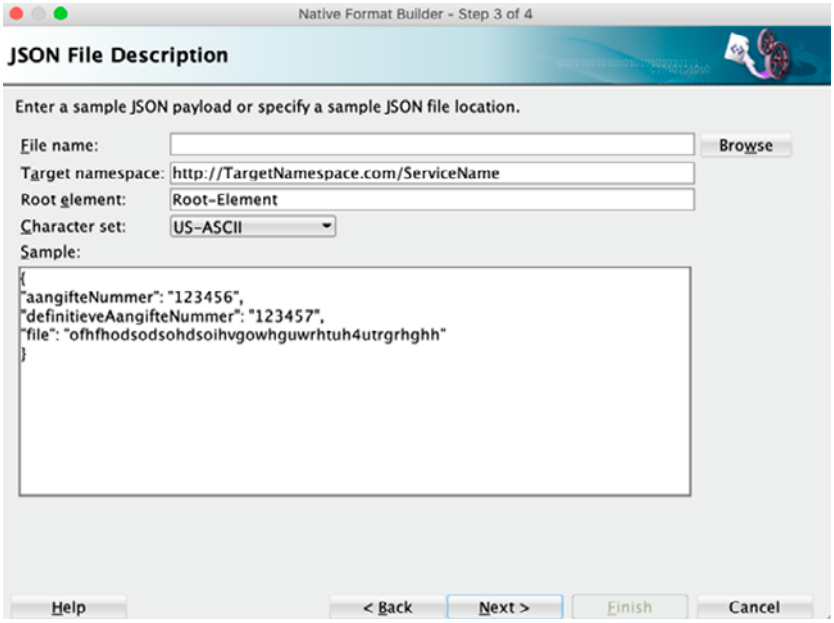
Geef je schema een naam:



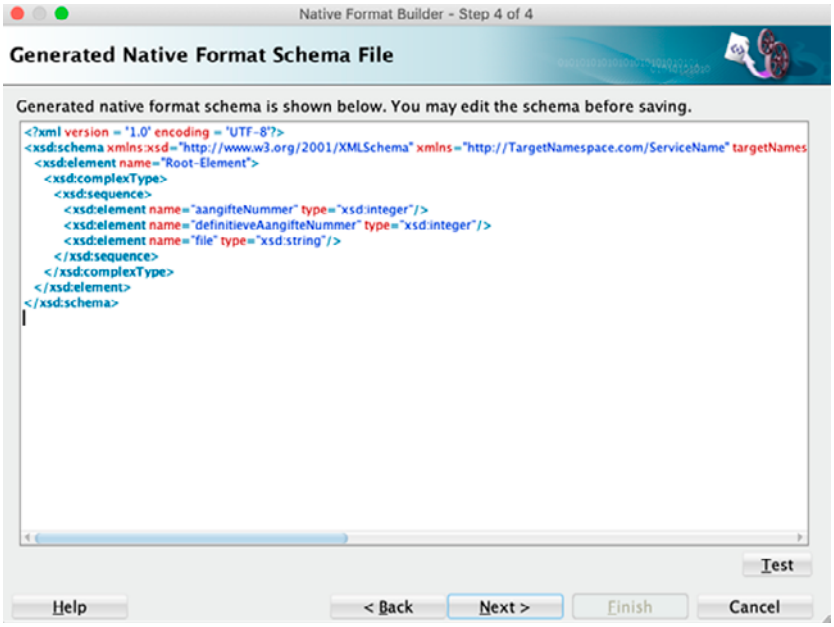
Selecteer het JSON formaat (zoals je ziet kun je ook heel veel andere datavormen kiezen):



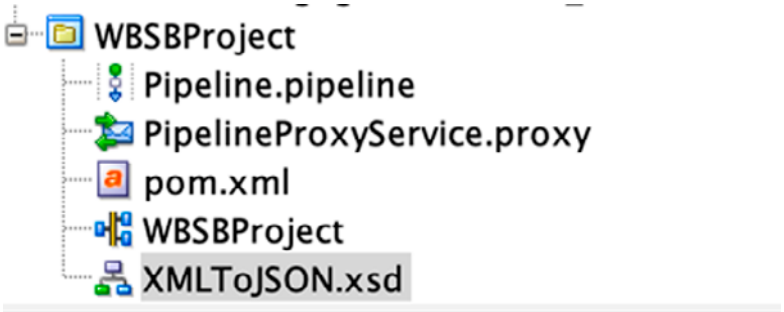
Geef een voorbeeld van hoe je JSON bericht eruit moet zien:



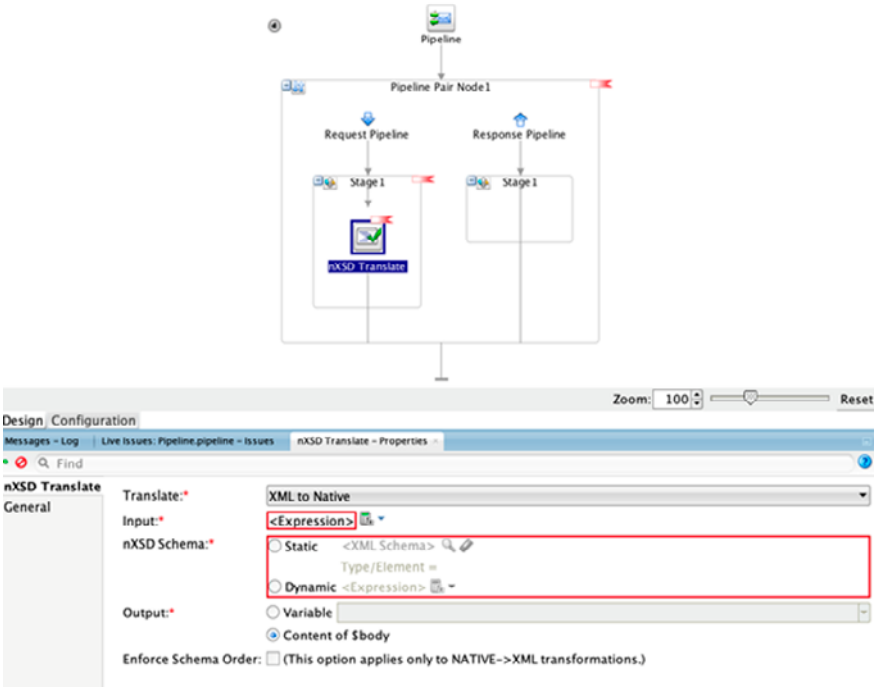
De native builder genereert vervolgens het schema voor je:



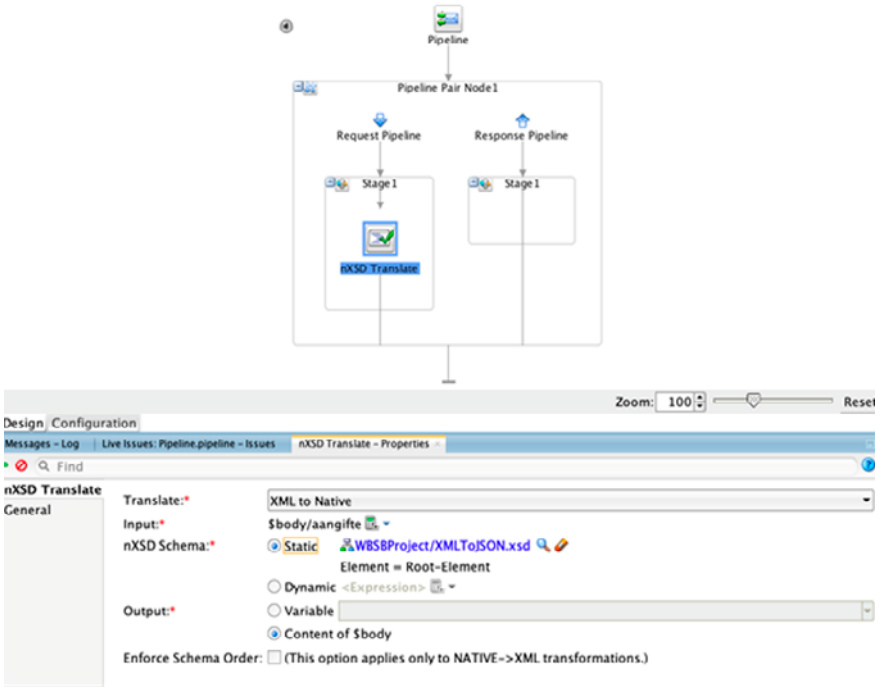
Sluit de builder af en check waar je file staat:



Nu ga je de nXSD transformatie maken. Sleep een nXSD blokje in je pipeline:



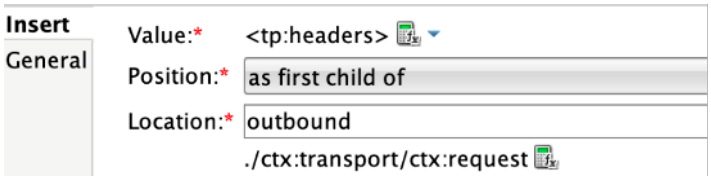
Laat de translatie op XML to Native staan, want dat is wat je nu nodig hebt, maar andersom kan dus ook. Kies bij input je binnenkomende bericht waarin zich de bronvelden bevinden. Kies voor nXSD schema het in de vorige stap gecreëerde schema, en laat de output op 'Content of $body' staan, want dat is wat je mee wilt geven. De service verwacht een berichtbody met daarin het JSON bericht:



De translatie is nu af, en er kan gedeployed en getest worden!

X-API-Key meegeven

Regelmatig worden REST Services beveiligd door de aanroepende partij een vooraf afgesproken API-Key mee te laten geven, zodat niet iedereen de service aan kan roepen. De locatie waar de key wordt meegegeven in de properties luistert nauw. Als je het op de volgende manier doet werkt het goed. Neem in de routingactie van de pipeline een insert op met de volgende inhoud:



Expression:

```
<tp:headers>
  <tp:user-header name="X-API-Key" value="{$xApiKey}"/>
  <http:Content-Type>application/json</http:Content-Type>
</tp:headers>
```

In dit voorbeeld komt de waarde van de API-Key uit een variabele (xApiKey). Deze variabele wordt door Maven tijdens de deployment gevuld met een waarde uit de property file van de desbetreffende omgeving. Zo kun je zonder last-minute aanpassingen de deployment doen op alle omgevingen van een OTAP-straat. Mocht je daar geen behoefte aan hebben, dan kun je in plaats van de xApiKey parameter hier ook gewoon de waarde opnemen.

Conclusie

REST Services kunnen uiteenlopende manieren hebben om aangeroepen te worden. Met bovengenoemde methodes is echter iedere manier van aanroepen te maken vanuit de OSB.

Bronnen en verder lezen

Oracle documentatie over REST Services (Service Bus 12c):
https://docs.oracle.com/middleware/1221/osb/develop/GUID-C346DF7D-041D-4E10-BE1C-451F50719106.htm#OSBDV89235

soapui.org/docs/rest-testing/understanding-rest-parameters

https://blog.sysco.no/osb/javascript/json/rest/Call-REST-API-in-Service-Bus-12.2.1.2-JSON-Request-Response/

https://blogs.oracle.com/adapters/invoke-a-rest-api-protected-with-an-api-key-using-oracle-integration-cloud-service

https://pebblesofosb.blogspot.com/2017/06/osb-12c-binary-message-type.html

https://kanifnathkolhe.wordpress.com/2019/05/26/json-to-xml-using-nxsd-in-osb-12c/
https://www.postman.com/ superfijne testtool om goed vast te stellen hoe het bericht er uit moet zien voor een goede aanroep van de API service.