Proof of concept

Surf stimuleringsregeling Uitwisselen onderwijsaanbod-data

Technische Universiteit Eindhoven

|  |  |
| --- | --- |
| **Auteur** | R.v.d. Slot |
| **Versie** | 1.0.0 |
| **Datum** | 3 december 2019 |

Inhoud

[Inleiding 2](#_Toc26435625)

[Oplossing 2](#_Toc26435626)

[Canonicals 3](#_Toc26435627)

[Canonical Classes 7](#_Toc26435628)

[Service Provider Zijde 7](#_Toc26435629)

[Osiris.Cursus WCF service 7](#_Toc26435630)

[Osiris.Cursus BizTalk component 9](#_Toc26435631)

[Consumer zijde 12](#_Toc26435632)

[BizTalk pipelines 12](#_Toc26435633)

[Receive pipeline 12](#_Toc26435634)

[Send pipeline 13](#_Toc26435635)

[IIS Webapplicatie 14](#_Toc26435636)

[Afmontage BizTalk 18](#_Toc26435637)

# Inleiding

Dit document beschrijft de ontwikkeling en uitwerking van de middleware oplossing voor de

Open Onderwijs API (OOAPI) endpoint door de Technische Universiteit Eindhoven (TUe).

De OOAPI specificatie is een uitwerking door Surf met als basis de European Credit Transfer System (ECTS) Course Catalog. De specificatie beschrijft de uitwisselingsstructuur van door universiteiten en hogescholen aangeboden cursussen en opleidingen op een generieke manier. Het achterliggende doel hiervan is het bieden van een platform waarin studenten verschillende cursussen en opleidingen van de aangesloten instituten kunnen vergelijken en van daaruit een keuze al dan niet selectie kunnen maken.

De bron van de gegevens over de cursussen en opleidingen van de TUe vormt de Osiris applicatie geleverd door Caci. De Osiris applicatie wordt door meerdere opleidings instituten gebruikt en is in de loop van het traject door Caci voorzien van een endpoint, die op zichzelf volledig aansluit op de OOAPI specificatie. De Osiris oplossing van de TUe kan hiermee in feite ook direct aangesloten worden op de OOAPI gateway van Surf. De architectuur regels van de TUe laten echter een directe koppeling van een externe partij als Surf met de Osiris omgeving van de TUe niet toe. Alle externe afnemers en leveranciers van gegevens dienen aan te sluiten op de middleware oplossing van de TUe (Microsoft BizTalk met WCF koppelingen).

# Oplossing

De positie van de middleware is er een waar een afnemer (consumer) applicatie een verzoek doet voor informatie en wacht op het resultaat, terwijl de middleware deze benodigde informatie weer direct ophaalt bij een leverancier (service provider) applicatie.

Leggen we deze tegen de referentie architectuur van de TUe, dan komen we tot onderstaand schema.

.

OOAPI architectuur oplossing

Binnen de OOAPI architectuur oplossing wordt de consumer zijde (links) via canonical berichten aan de Service Provider zijde (rechts) gekoppeld. De consumer zijde plaatst hierbij een canonical bericht op de messagebus, die via een subscription van SendPorten op deze berichten wordt opgepakt. Omgekeerd plaatst de Service Provider zijde de resultaten weer als canonical berichten op de messagebus. De middleware (BizTalk) zorgt er via een zogenaamde CorrelationToken weer voor dat dit resultaat bij de juiste consumer aanvraag wordt afgeleverd.

Zoals volgens normaal gebruik wordt de oplossing ook hier in drie deel oplossingen (applicaties) opgebouwd :

1. Canonicals (TUe.Domain.CDM.OOAPI.ESB)
2. Consumer zijde (Tue.SurfOOAPI.ESB)
3. Service Provider zijde (TUe.Osiris.Cursus)

## Canonicals

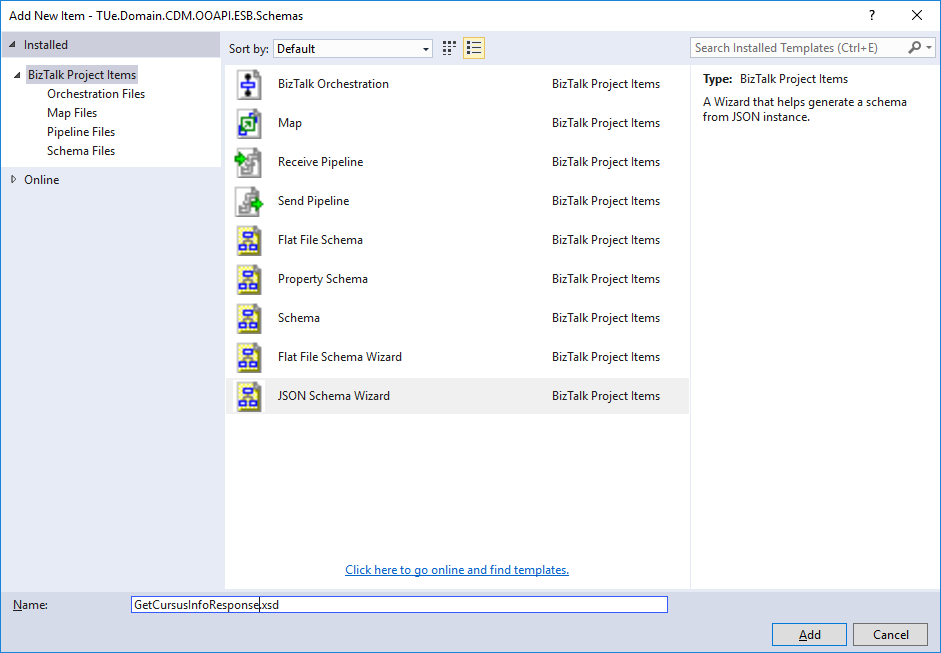
De canonical schema’s zijn over het algemeen generieke schema’s die alle relevante attributen van een functioneel domein van een organisatie bevatten. Hierbij is het niet echt van belang of aanleverende of afnemende partijen (applicaties) deze attributen ook aanleveren of afnemen. De canonical schema’s vormen een bepaalde werkelijkheid van de organisatie en hebben in eerste instantie een functionele opzet. Canonical schema’s in functionele zin worden opgesteld al dan niet geaccordeerd door de dataarchitect van een organisatie.

Binnen het kader van de proof of concept zou het te ver gegaan zij om een compleet traject op te starten om tot een aangepaste set van canonicals ten behoeve van de uitwisseling van cursus informatie te komen. We zijn daarom direct uitgegaan van de uit te wisselen gegevens, omdat vanuit het OOAPI voortraject reeds een uitwisselingsstructuur van cursusgegevens opgesteld is. Hierbij zijn de relevantie van attributen geëvalueerd. De berichtenstructuur is OOAPI dan ook een soort van overkoepelend canonical model.

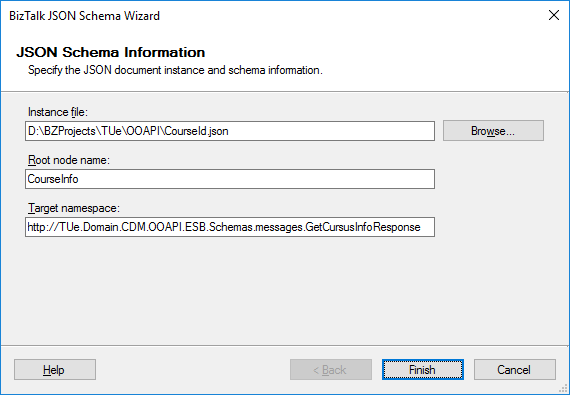
Voor de canonical schema’s is eerst in Visual Studio 2015 een nieuw BizTalk project gedefinieerd met de naam “TUe.Domain.CDM.OOAPI.ESB”. Binnen dit project is volgens goed gebruik een nieuwe oplossing (Solution) aangemaakt met de naam “TUe.Domain.CDM.OOAPI.ESB.Schemas” waar binnen alle (canonical) schemas worden gedefinieerd.

Binnen OOAPI specificatie van Surf zijn de berichten uitwisselingen in het JSON formaat gedefinieerd. BizTalk werkt op zichzelf met XML schema’s en daarom zijn de JSON schema’s eerst binnen BizTalk geïmporteerd via de JSON Schema Wizard. De JSON voorbeeld bestanden (de voorbeelden inclusief voorbeeld data) zijn hiervoor opgehaald en op de BizTalk ontwikkelserver geplaatst, omdat de import wizard ze als file moet kunnen inlezen.

Vanuit de “TUe.Domain.CDM.OOAPI.ESB.Schemas” hebben de uit te wisselen schema’s een voor een met “Add New Item” en “JSON Schema Wizard” aangemaakt.



Na het invullen van een naam voor het schema, kunnen we het opgehaalde JSON bestand selecteren en moeten we nog een zelf te kiezen “Root node name” invoeren.

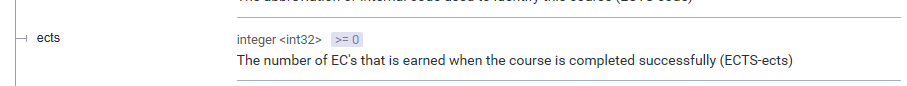


De target namespace is gebaseerd op de gekozen schema naam en voldoet daarmee prima.

Nadat het schema is aangemaakt, zullen we deze nog moeten nalopen. De door BizTalk gegenereerde schema worden aangemaakt volgens een stelsel van standaardregels. Deze regels die kunnen leiden tot interpretatie die afwijkt van de werkelijkheid.

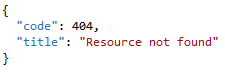


Zo is in bovenstaande gegenereerde schema het veld ‘maxNumberStudents’ volgens de interpretatie een “unsigned byte”. Dit komt doordat in het voorbeeld een getal stond dat binnen een “unsigned byte” past.



Echter binnen de OOAPI specification is deze als “int32” gedefinieerd. Hierop moet het schema handmatig gecorrigeerd worden om uiteindelijk datacorruptie of excepties te voorkomen.

Op deze manier hebben we vrij eenvoudig alle resultaat canonical schema’s gedefinieerd, die vanuit de Service Provider terug naar de Consumer worden gezonden. Aan deze canonicals zijn uiteindelijk nog twee velden ( “code” en “title” ) toegevoegd, waarmee we ze ook kunnen gebruiken om foutmeldingen terug te sturen. Dit is inherent aan de specificatie waar in plaats van het antwoord bericht ook een foutmelding aan de consumer kan worden aangeboden in de vorm :



Uiteindelijk hebben we voor de volgende resultaat berichten (endpoints) een canonical schema nodig :

/

/institution

/courses

/courses/{courseId}

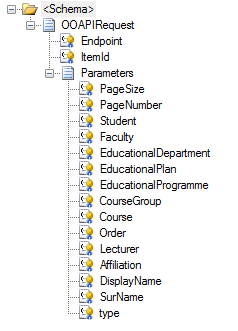
/course-offerings

/course-offerings/{courseOfferingId}

/educational-programmes

/educational-programmes/{educationalProgrammeId}

Voor de verzoeken vanuit de consumer hebben we voor een enkel generiek canonical schema gekozen.



In het schema zijn velden opgenomen voor het gewenste endpoint en de eventueel daarbij gewenste item. Verder zijn er alle gedefinieerde url parameters opgenomen van de Service Provider. In feite is dit geen echt canonical schema. Het is een compromis om Restful verzoeken van de Consumer naar de Service Provider te kunnen aanleveren. De attributen in het schema zijn allemaal promoted properties (MessageContextPropertyBase), zodat er later binnen BizTalk op gefilterd kan worden.

### Canonical Classes

Voor de uitwisseling via de WCF services aan de Service Provider zijde maken we ook gebruik van de canonical modellen. Hiervoor hebben we CSharp classes nodig. Deze kunnen met de hand worden geherdefinieerd, maar het is eenvoudiger om deze via Xsd2Code te genereren :

Xsd2Code.exe" "Courses.xsd" "TUe.Domain.CDM.OOAPI.ESB. Messages.Classes" "Courses.cs" /platform Net40 /collection List /dc+ /xa+ /eit+

Bovenstaand voorbeeld levert een “Courses.cs” gebaseerd op het “Courses.xsd” canonical schema van BizTalk met als namespace Tue.Domain.CDM.OOAPI.ESB.Messages.Classes. Alle vanuit de canonical schema’s gegenereerde .cs files zijn aan een nieuw CSharp class library project “TUe.Domain.CDM.OOAPI.ESB.Classes” toegevoegd, zodat daar vanuit de WCF aan gerefereerd kan worden.

## Service Provider Zijde

De Service Provide zijde bestaat weer uit twee componenten. De eerste is de WCF service, die de verzoeken vanuit BizTalk omzet naar een aanroep van de provider applicatie (Osiris) en het resultaat weer teruglevert aan de BizTalk. Het tweede component is het BizTalk component dat bestaat uit een aantal Send Ports met filters en een mapping om het resultaat naar de canonical schemas om te zetten.

### Osiris.Cursus WCF service

Vanuit Visual Studio hebben voor de WCF service een nieuw project gegenereerd van het type WCF Service application met de naam “TUe.Osiris.Cursus.Services”. Vanuit de solution legden we een referentie aan naar de solution van de Canonical Classes project (TUe.Domain.CDM.OOAPI.ESB.Classes.csproj), zodat deze classes binnen het WCF project beschikbaar komen.

Voor ieder resultaatbericht (endpoint) wordt nu binnen het project een class gegenereerd, waarin een [DataMember] wordt opgenomen, die een instantie krijgt naar het bijbehorende class uit het gerefereerde canonical class (uit TUe.Domain.CDM.OOAPI.ESB.Classes) en die een MessageBodyMember namespace krijgt, die overeenkomt met de name space uit het BizTalk canonical project (TUe.Domain.CDM.OOAPI.ESB.Schemas).



Voorbeeld van class definitie

Aan de hand van deze nieuw gegenereerde classes zijn de WCF interfaces gedefinieerd.



In een nieuw solution (TUe.Osiris.Cursus) zijn de implementaties van de interfaces (operationalContracts) uitgewerkt.



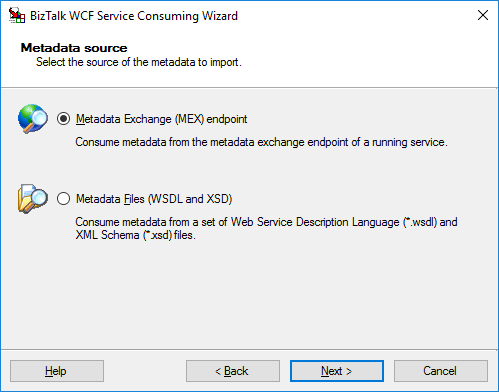
Voor Osiris betekende dit dat vanuit het canonical verzoek (GenericRequest) de URL naar Osiris geconstrueerd diende te worden en dat de JSON resultaat naar het canonical formaat moest worden omgezet. Doordat het door Osiris aangeleverde resultaat overeenkomt met de canonical, konden we volstaan met een eenvoudige serialization van de JSON, nadat deze omsloten is door de root nodes van het gedefinieerde xml schema. Indien de serialization faalt, wordt er een nieuw fout bericht gegenereerd en een paar impliciet gegenereerde nodes verwijdert (door deze null te verklaren).

### Osiris.Cursus BizTalk component

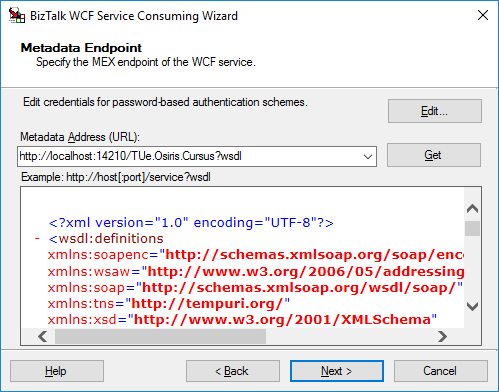
Nadat de WCF service gestart is, kan het BizTalk component worden gemaakt. De draaiende WCF service is benodigd om via de wsdl de BizTalk schema’s te genereren.

Binnen Visual Studio 2015 is hiervoor een nieuw project gemaakt met de naam “TUe.Osiris.Cursus”. Binnen dit project zijn volgens goed gebruik twee solutions aangemaakt. De eerste voor de BizTalk schema’s “TUe.Osiris.Cursus.Schemas” en de tweede voor de translations “TUe.Osiris.Cursus.Maps”. De Maps solution heeft een reference naar de Schema’s solution en naar de de Canonical ( TUe.Domain.CDM.OOAPI.ESB.Schemas ) solution gekregen.

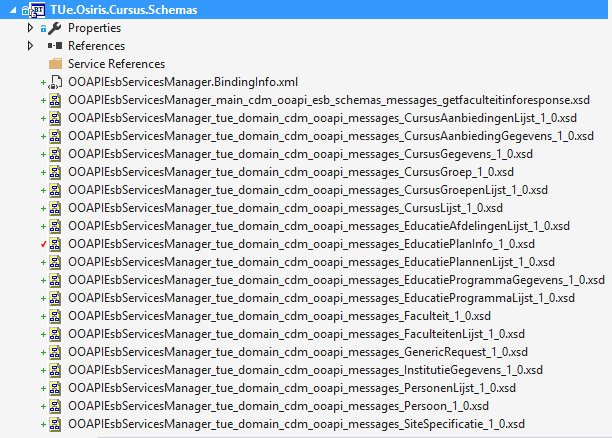
De schema’s benodigd voor de uitwisseling tussen de canonical schema’s en de WCF service zijn vanuit de wsdl van de WCF service gegenereerd. Vanuit de schema’s solution is via “Add Generated Items” de “Consume WCF Service” gestart.



Binnen de Wizard is de “Metadata Exchange (MEX) endpoint” gekozen.

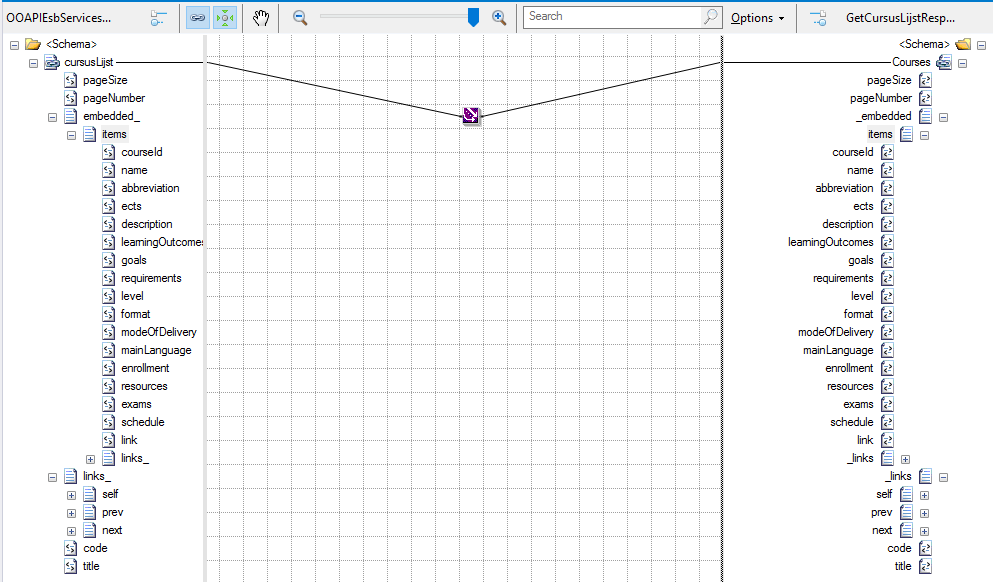


Binnen de Metadata Address dient de URL van de WebService ingevuld inclusief “?wsdl”. Daarna is met “Get” de service informatie opgehaald en zijn na “Next” alle schema’s aan het BizTalk schemas project toegevoegd.



Nadat de WCF schema’s zijn toegevoegd, zijn de mappings in de mappings solution aangemaakt. Voor iedere endpoint is er een mapping voor het resultaat naar de canonical en voor het verzoek is er een enkele mapping van de canonical naar de WCF service.

De mappings zelf zijn in de simpelste vorm. Door de gekozen opzet met canonicals en de daarmee afgeleide WCF schema’s zijn de schema’s ten aanzien van de velden identiek. Het verschil zit hem in de xml namespace (en de rootnodes). Binnen de map dienen dan ook alleen deze te worden vertaald en hiervoor maken we gebruik van een ‘bulk copy’ op de rootnode.



Bulk copy translation

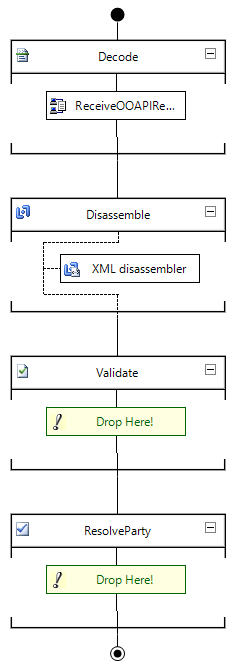
## Consumer zijde

De consumer zijde bestaat uit een BizTalk solution met een receive en een send pipeline en een IIS Webapplication.

### BizTalk pipelines

### Receive pipeline

De receive pipeline zorgt voor de vertaling van de GET verzoeken naar het BizTalk request schema.



Binnen de Receive pipeline hebben we twee pipelinecomponenten. De Disassemble phase in voorzien van de standaard BizTalk XML disassembler en heeft als doel om het juiste message type op het bericht te plaatsen en daarmee ook de properties te promoten, die we eerder in de canocical als promoted properties hebben gedefinieerd.

Voor de Decode stage hebben we een maatwerk pipeline component toegevoegd, die als taak heeft om het GET verzoek om te zetten naar de canonical xml structuur. Hiermee lossen we het probleem om dat de GET request geen body heeft en daarmee normaliter een leeg bericht aanlevert ( BizTalk verwijdert lege berichten weg zonder er iets mee te doen).

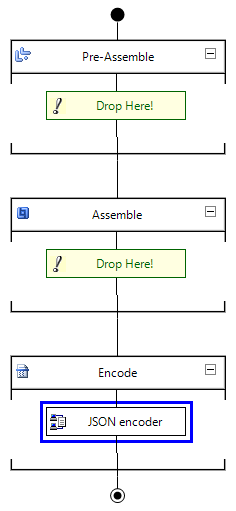
Onderstaande code is die van de “Execute” overload functie. Dit is ook de enige code die aan het pipeline component template is toegevoegd. Binnen de code is de xml van de canonical gedefinieerd met Endpoint “Root” en ItemId “geen”. Deze waarde gebruiken we om niet aangeleverd te detecteren.

De context properties, die we hier gebruiken zijn door de “BtsHttpUrlMapping” gezet, welke in het volgende hoofdstuk zijn beschreven.



### Send pipeline

De send pipeline zorgt voor de vertaling van de canonical berichten naar het JSON formaat, die binnen de OOAPI specificatie beschreven staan. Hiervoor wordt gebruikt gemaakt van de standaard door BizTalk meegeleverde JSON encoder, waarbij de “Remove Outer Envelope” op True gezet is om de root node van het BizTalk schema niet mee te nemen in het resultaat.

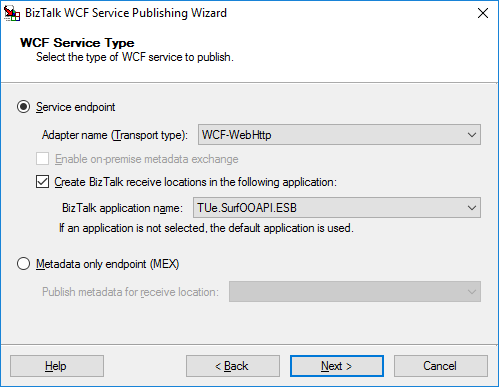


BizTalk Send Pipeline

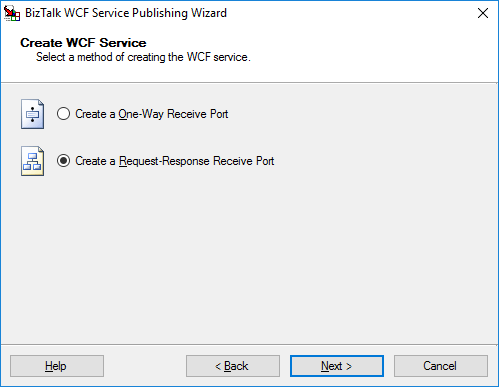
### IIS Webapplicatie

De IIS Webapplicatie is een template applicatie, die via de BizTalk WCF Service Publishing Wizard is aangemaakt. Deze applicatie zorgt ervoor dat een Web verzoek dat binnenkomt op IIS bij BizTalk wordt aangeboden.

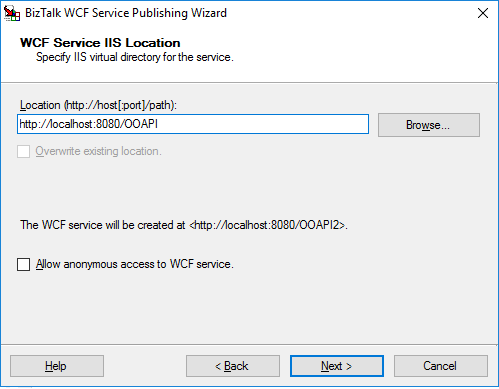
Voordat de wizard wordt gestart, zijn de twee pipelines reeds op de BizTalk server gedeployed binnen de BizTalk applicatie “TUe.SurfOOAPI.ESB”



Na het starten van de wizard is de adapter op “WCF-Web-Http” transport gezet en laten we de Wizard receive locations toevoegen aan de “TUe.SurfOOAPI.ESB” applicatie, waarin we de twee pipelines hebben gedeployed.

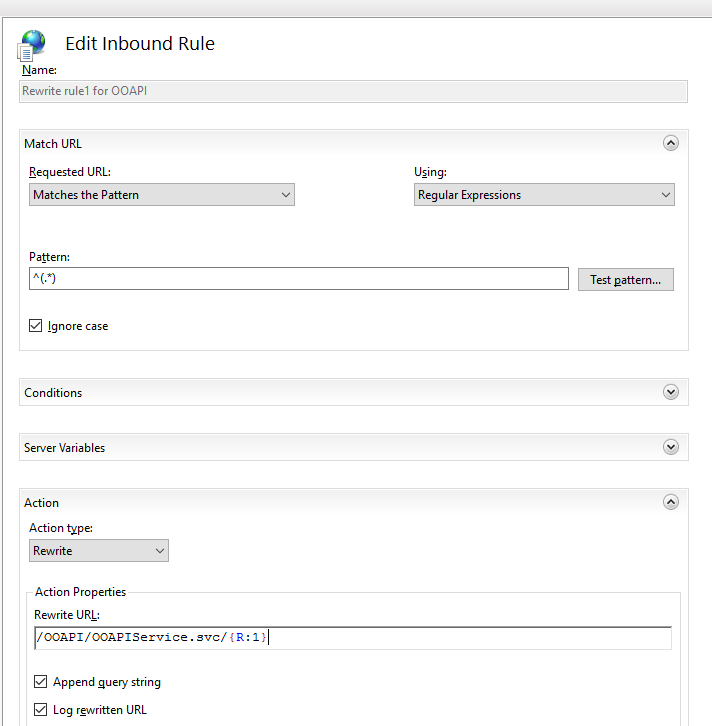


De method van de WCF Service is op “Request-Response” port gezet, omdat we een verzoek vanuit Surf ontvangen en hier direct ook een antwoord op terug dienen te geven.



Als laatste is in de Wizard de locatie aangegeven binnen IIS vanwaar de webservice benaderd dient te worden.

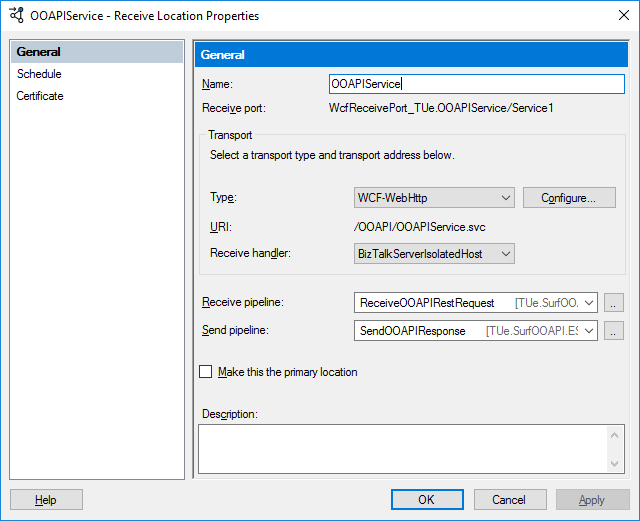
Volgens bovenstaand voorbeeld wordt de root url voor de Surf gateway : “https://host:port/OOAPI/OOAPIService.svc”.

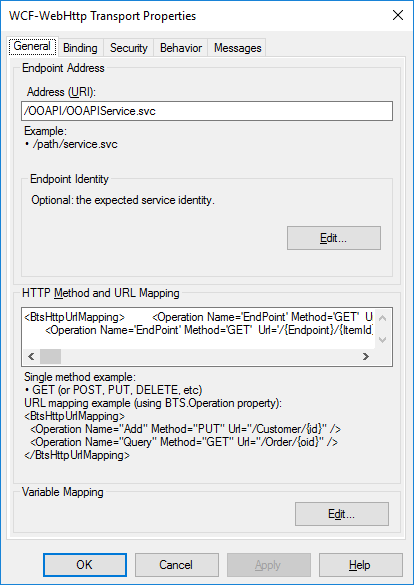


Met behulp van bovenstaande “Rewrite Rule” (rewrite rules diene apart geinstalleerd te worden) hebben de url verder kunnen normaliseren tot :

“https://host:port/OOAPI”.

Binnen het BizTalk console is er hierdoor een nieuwe Receive Port en Receive Location aangemaakt. Binnen de Receive location zijn de twee pipelines toegevoegd.





Binnen de HTTP Method hebben we onderstaande mapping toegevoegd. Dit zorgt ervoor dat alle endpoints, items en parameters binnen BizTalk ‘promoted’ worden.

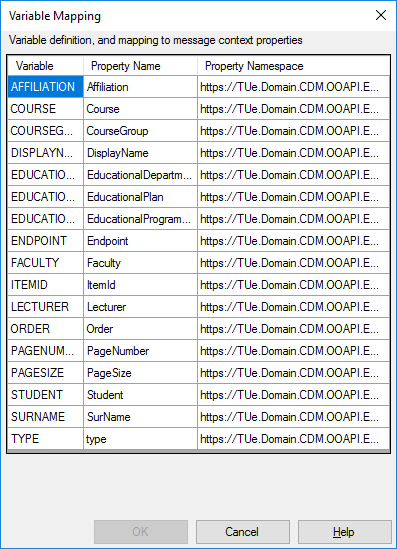
<BtsHttpUrlMapping>

<Operation Name='EndPoint' Method='GET' Url='/{Endpoint}?PageSize={PageSize}&amp;PageNumber={PageNumber}&amp;student={Student}&amp;faculty={Faculty}&amp;educationalDepartment={EducationalDepartment}&amp;educationalPlan={EducationalPlan}&amp;educationalProgramme={EducationalProgramme}&amp;courseGroup={CourseGroup}&amp;course={Course}&amp;Lecturer={Lecturer}&amp;order={Order}&amp;surName={SurName}&amp;displayName={DisplayName}&amp;affiliation={Affiliation}&amp;type={type}' />

<Operation Name='EndPoint' Method='GET' Url='/{Endpoint}/{ItemId}?PageSize={PageSize}&amp;PageNumber={PageNumber}&amp;student={Student}&amp;faculty={Faculty}&amp;educationalDepartment={EducationalDepartment}&amp;educationalPlan={EducationalPlan}&amp;educationalProgramme={EducationalProgramme}&amp;courseGroup={CourseGroup}&amp;course={Course}&amp;Lecturer={Lecturer}&amp;order={Order}&amp;surName={SurName}&amp;displayName={DisplayName}&amp;affiliation={Affiliation}&amp;type={type}' />

<Operation Name='Endpoint' Method='GET' Url='' />

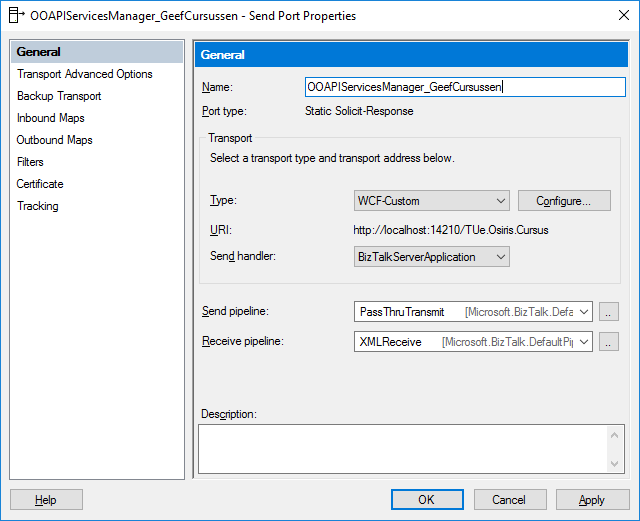
</BtsHttpUrlMapping>



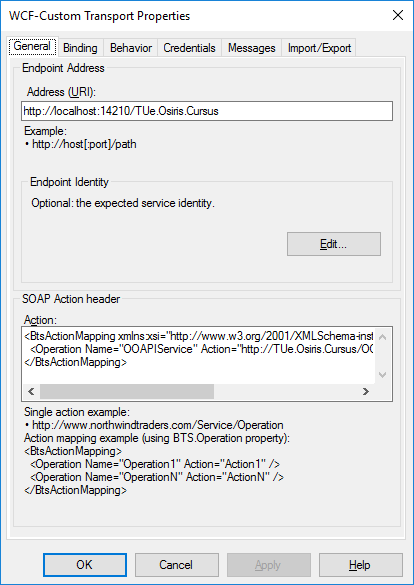
Variabele mapping

## Afmontage BizTalk

De laatste activiteit die uitgevoerd is, zijn de Send ports binnen de TUe.Osiris.Cursus applicatie binnen het BizTalk console. Deze Send ports nemen een subscription op de berichten die via de “TUe.SurfOOAPI.ESB” applicatie op de messagebus worden geplaatst.



Per endpoint is er een “static solid send-response” sendport gemaakt van het type WCF-Custom.



De URL naar de WCF-service is ingevoerd en bij de Action de verwijzing naar de specifieke service in de “TUe.Osiris.Cursus.Services”. Bijvoorbeeld :

<BtsActionMapping xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<Operation Name="OOAPIService" Action="http://TUe.Osiris.Cursus/OOAPIEsbServices/GeefCursussen" />

</BtsActionMapping>

Met een filter naar het te verwerken verzoek bericht. Het filter heeft daarbij in de vorm :

“BTS.MessageType == http://TUe.Domain.CDM.OOAPI.Messages/GenericRequest/1.0#OOAPIRequest And

TUe.Domain.CDM.OOAPI.ESB.Schemas.PropertySchema.Endpoint == courses And

BTS.Operation == OOAPIService And

TUe.Domain.CDM.OOAPI.ESB.Schemas.PropertySchema.ItemId == geen“

