# Enexis HAL implementatie

Implementatie van RESTful API's o.b.v. de HAL specificatie

**Wouter Meijers** wouter.meijers@enexis.nl
**Ton Donker** ton.donker@enexis.nl

**Versie: 1.0**
**8 april 2021**

ENEXIS GROEP

# Change history

| Version | Changes | Document |
|---|---|---|
| 20201113 | Eerste release voor review met Tony Sloos; | HAL-Implementatie-ECDM-20201113.pptx |
| 20201207 | Introductie van ResourceRef, Filter expressies middels '_filter' parameter, kleine collecties toegelicht; | HAL-Implementatie-ECDM-20201207.pptx |
| 20210126 | Aanpassingen n.a.v. review Tony Sloos;<br>Put op een resource alleen voor overschrijven;<br>Link attributen herzien: 'name', 'title'. Toevoeging 'id';<br>IdentificationScheme (MRID def) toegevoegd. | HAL-Implementatie-ECDM-20210126.pptx |
| 20210403 | Nieuw document met focus op HAL/HATEOAS aspecten | HAL-Implementatie-Enexis-20210403.pptx |

# RESTful API's – Statement

*What needs to be done to make the REST architectural style clear on the notion that hypertext is a constraint? In other words, if the engine of application state (and hence the API) is not being driven by hypertext, then it cannot be RESTful and cannot be a REST API. Period.*

Source: **REST APIs must be hypertext-driven** by Roy Fielding.

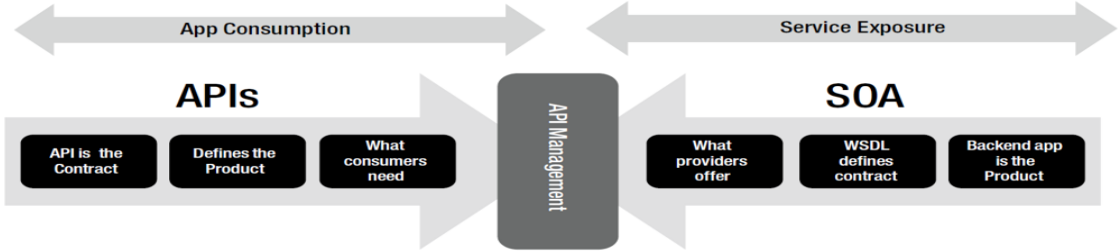
# RESTful API's Introductie - HAL

- ✓ Er is momenteel geen W3C standaard voor het implementeren van hypermedia controls, maar wereldwijd gezien zijn HAL en JSON API de meest populaire implementatietechnieken[1];
- ✓ HAL is geadopteerd als 'best-practice' voor implementatie van hypermedia controls door Kennisplatform API's t.b.v. NL overheid (Geonovum e.d.)[2] en NL overheids API's zijn HAL-compliant[3];
- ✓ De HAL specificatie bevindt zich (nog) steeds in de status van RFC draft[4];
- ✓ In vergelijking met andere specificaties[5] biedt media type HAL een aantal onderscheidende pluspunten:
  - Simplicity en leesbaarheid
  - Navigeerbaarheid (cross API's – context switch) en hiërarchisch (nesting)
  - Multi format: XML en JSON
  - Incorporatie van documentatie ('curies')
  - Representatie op maat (met '_fields' en '_expand' parameters)
  - Actieve community[6] en internationale implementatie (o.a.: Amazon, Microsoft, Visa, Paypal)

1. https://sookocheff.com/post/api/on-choosing-a-hypermedia-format/
2. https://www.geonovum.nl/themas/kennisplatform-apis & https://iplo.nl/digitaal-stelsel/aansluiten/standaarden/api-en-uri-strategie/
3. Voorbeelden: https://developer.overheid.nl/, https://www.vngrealisatie.nl/producten/haal-centraal, https://commonground.nl/
4. https://tools.ietf.org/html/draft-kelly-json-hal-08
5. Overzicht van verschillende hypermedia formats: https://restapilinks.com/hypermediaformats/
6. https://groups.google.com/g/hal-discuss/

# RESTful API's Introductie – HAL *vervolg*

- ✓ Enexis ziet de toegevoegde waarde van het 'representatie op maat' argument en dit document is een detail uitwerking van deze functionaliteit middels de '_fields' en '_expand' parameter. RESTful API's zijn ontworpen met de behoefte van de consumer in gedachten:




- ✓ Aandachtspunten m.b.t. HAL zijn:
  - Nog niet vastgesteld als standaard (zoals voor JSON-LD wel het geval is[1])
  - Omdat HAL vnl. *resource-driven* is en niet *action-driven* (zoals Siren[2]) zijn uitbreidingen/extensies benodigd voor ondersteuning HATEOAS (zie volgende slide)

1. https://www.w3.org/TR/json-ld/
2. https://sookocheff.com/post/api/on-choosing-a-hypermedia-format/

# RESTful API's Introductie - HATEOAS

- ✓ HATEOAS is het principe dat een client interacteert via hypermedia die dynamisch wordt toegewezen. Een REST-client heeft zodoende geen voorkennis over hoe te communiceren met een bepaalde toepassing of server buiten een algemeen begrip van hypermedia. Vergelijk het met 'browsen op het WWW'. Het HATEOAS principe volgt de definitie van Hypertext als: *tekst met links die de client in staat stelt om vervolgacties uit te voeren*;
- ✓ Volgens Roy Fielding is ondersteuning van HATEOAS een verplichte constraint voor een RESTful API[1]. De meningen hierover zijn echter verdeeld en een pragmatische insteek lijkt het meest gangbaar. Het Kennisplatform API's definieert HATEOAS als een 'optionele constraint' en er zijn momenteel geen API's te vinden op https://developer.overheid.nl/ die HATEOAS daadwerkelijk ondersteunen;
- ✓ Implementatie van HATEOAS is momenteel 'work in progress', hoewel er inmiddels verschillende internationale API's zijn ontwikkeld die HATEOAS 'in zekere mate' ondersteunen[2];
- ✓ Met betrekking tot HATEOAS lijk de markt lijkt de laatste jaren steeds meer in ontwikkeling te komen als gevolg van de wereldwijde adoptie van de REST architectuur stijl;
- ✓ Succes van HATEOAS is ook afhankelijk van de ontwikkeling van hypermedia clients, die de links op éénduidige wijze kunnen interpreteren;

1. https://roy.gbiv.com/untangled/2008/rest-apis-must-be-hypertext-driven
2. https://restapilinks.com/hateoas-2/

# RESTful API's Introductie – HATEOAS *vervolg*

- ✓ Nu HAL als 'best practice media type' is vastgesteld, zijn in het verlengde hiervan een aantal marktontwikkelingen gaande die de implementatie van HAL/HATEOAS mogelijk maken:
  - Extensie van HAL met 'method'. Implementatie voorbeeld: Paypal[1]
  - Link relaties die de acties weergeven. Implementatie voorbeeld Amazon AWS[2]. Eventueel in combinatie met 'curies'[3]
  - HAL-FORMS[4]: recente extensie op HAL en kan gezien worden als een implementatie van HTML formulieren. Eventueel in combinatie met 'curies'[5]
- ✓ Het standaard HTTP commando OPTIONS, waarmee de actions op een bepaalde resource kunnen worden opgevraagd vindt nauwelijks opvolging in de markt[6]. OPTIONS geeft ook geen informatie over de attributen in de payload;
- ✓ Conclusie: er wordt dus volop gekeken naar de HAL specificatie en navenante aanpassingen om HATEOAS mogelijk te maken. HAL lijkt dus ook in dit opzicht een toekomstbestendige keuze;

1. https://developer.paypal.com/docs/api/reference/api-responses/#hateoas-links
2. https://docs.aws.amazon.com/apigateway/api-reference/
3. https://developer.avid.com/ctms/hal_intro.html
4. https://rwcbook.github.io/hal-forms/
5. https://github.com/hateoas-forms/spring-hateoas-example
6. http://zacstewart.com/2012/04/14/http-options-method.html



# RESTful API’s – Introductie

- ✓ De kracht van RESTful API’s komt alleen dan daadwerkelijk naar voren als de ***aanroepende partij*** in verregaande vorm ***controle*** kan uitoefenen op de informatie die zij ***wil ontvangen*** en waarbij de ***leverende bron*** informatie levert die de aanroepende partij ***helpt*** bij het uitoefenen van die controle:
  - ➢ Aanroepende partij kan resource informatie ***selectief opvragen***, of juist ***onderdrukken***;
  - ➢ Aanroepende partij kan door resource collecties ”***bladeren***”, zowel voor- als achteruit als selectief;
  - ➢ Aanroepende partij kan, in een resource collectie, resources ***vinden*** met specifieke content;
  - ➢ Aanroepende partij kan resultaten ***filteren*** en/of ***sorteren*** op diverse criteria;
  - ➢ Leverende bron ***helpt*** de aanroepende partij door context-gevoelige ***hyperlinks*** te leveren waarmee de aanroepende partij ***meer gedetailleerde*** of ***aanvullende*** informatie kan opvragen en/of op dynamische wijze ***acties*** kan uitvoeren. We introduceren hiermee “***een beetje HATEOAS functionaliteit***”;
- ✓ De verzameling van middelen waarop RESTful API’s de aanroepende partij helpen bij het ***interpreteren*** van ***informatie*** en het ***bepalen*** van ***vervolgacties*** op basis van resource representaties, noemen we ***hypermedia controls***;



# RESTful API's – Introductie

- ✓ Binnen de scope van dit document onderscheiden we de volgende ***hypermedia controls***:
  - ➢ De RESTful API kan per resource ***hyperlinks*** teruggeven naar ***gerelateerde informatie en/of uit te voeren acties.*** Hiervoor maken we gebruik van de ***Hypertext Application Language*** (*HAL*);
  - ➢ De ***scope*** van een leesoperatie kan worden beïnvloed middels de '*_expand*' parameter. Standaard zal de RESTful API alleen ***links*** naar geassocieerde- en/of sub-resources teruggeven; middels '*_expand*' geef je aan dat je in plaats van de link toegang wil krijgen tot de ***inhoud*** van die resources;
  - ➢ De verzameling van ***state properties*** (*attributen*) die, *binnen de geselecteerde scope*, door een leesoperatie worden opgehaald is te beïnvloeden middels de '*_fields*' en de '*_exclude*' parameters. Met '*_fields*' geef je op ***alleen*** geïnteresseerd te zijn in de genoemde properties, met '*_exclude*' geef je aan bepaalde properties juist ***niet*** te willen ontvangen;
  - ➢ ***resource collecties kunnen*** het navigeren door verzamelingen resources (***paginering***) alsmede ***zoeken*** en ***sorteren*** van verzamelingen resources ondersteunen. Hiertoe zijn een aantal specifieke stuur-parameters gedefinieerd ('*_limit*', '*_page*', '*_from*', '*_before*', '*_after*', '*_sort*', '*_find*', '*_filter*') en kunnen navigatie links worden teruggegeven ('*first*', '*last*', '*next*', '*prev*' en '*item*');



# RESTful API's – Leeswijzer

- ✓ Voornoemde concepten zijn in dit document verder uitgewerkt. Zie:
  - ➢ Resource en geassocieerde resource voor de definitie van de resource class;
  - ➢ Sub-resource voor de definitie van een sub-resource;
  - ➢ Resource collectie voor de definitie van de resource collectie, inclusief zoeken, pagineren en sorteren;
  - ➢ Gegevensgroep en Attribuut beschrijven de *state properties* van de resource;
  - ➢ Gereserveerde namen beschrijft alle speciale parameters en andere gereserveerde namen zoals '*_expand*', '*_fields*', '*_exclude*', '*first*', '*next*', etc.
  - ➢ In Eager- en Lazy Loading beschrijven we middels voorbeelden hoe, door gebruik van voornoemde parameters, het gedrag van een lees-response is te sturen volgens de wensen van de aanroepende partij;

# RESTful API’s

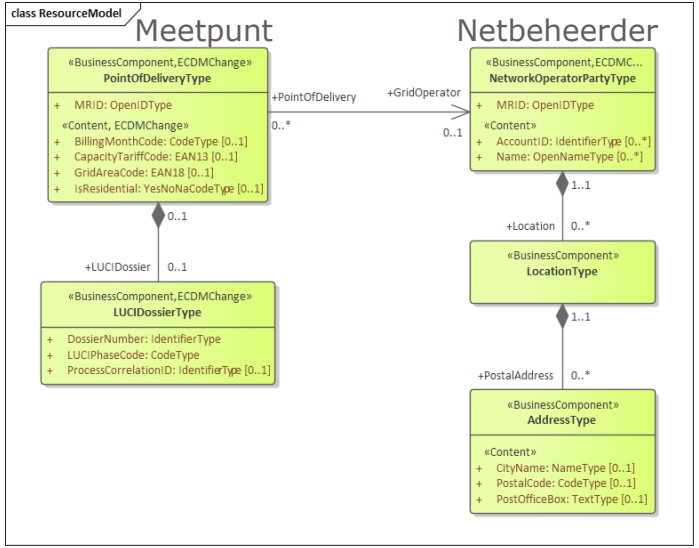
1. **Resource en geassocieerde resource;**
2. Sub-resource;
3. Resource collectie;
4. Gegevensgroep;
5. Attribuut;
6. Gereserveerde namen;
7. Eager- en Lazy loading;

# RESTful API's – Resource definitie

Als basis voor de implementatie van HAL/HATEOAS dienen we als eerste een aantal definities op te stellen van de relevante componenten die we binnen het berichtmodel van een willekeurige RESTful API kunnen onderscheiden, te weten:

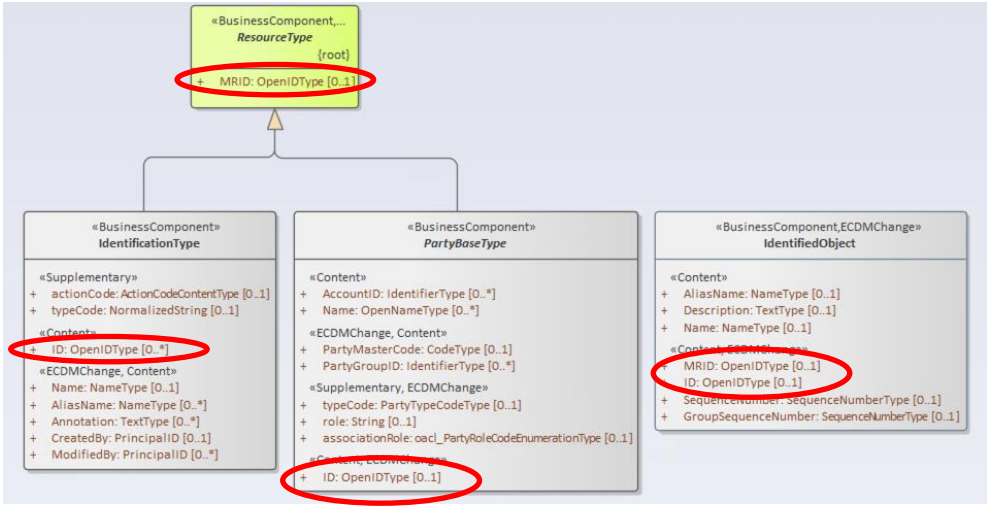
1. *Resource & geassocieerde resource*;
2. *Sub-resource*;
3. *Resource collectie;*
4. *Gegevensgroep*;
5. *Attribuut*;

Nevenstaand ***resource model*** zal gebruikt worden ter illustratie. Het toont een tweetal ***resources*** (*Meetpunt* en *Netbeheerder*) met een paar relaties.



# RESTful API's – Resource definitie

- ***In ECDM is een resource gedefinieerd als een UML Class in een berichtmodel die uniek identificeerbaar is middels een MRID (Master Resource IDentifier) attribuut.***
- ***Een resource dient te allen tijde direct- en zelfstandig adresseerbaar te zijn middels een Top-Level URL in "zijn" API***;
- ***Een Top-Level URL is een URL die de resource direct adresseert en die niet afhankelijk is van een andere resource;***




Omdat in ECDM de OAGIS basistypen zijn afgeleid van het nieuwe root-object *ResourceType* hebben alle 'identificeerbare' OAGIS classes tegenwoordig de mogelijkheid om een ***MRID*** te kiezen en daarmee de class in een berichtmodel tot **resource** te promoveren (in CIM bestond deze identifier altijd al).

**Door in *SchemaComposer* het MRID te selecteren ontstaat "automatisch" een resource!**

De API structuur dient dit uiteraard wel te eerbiedigen (m.a.w. de resource dient als URL terug te komen in de API).
Om te voorkomen dat CIM classes nu per definitie altijd een resource worden is in CIM het **ID** attribuut toegevoegd waarmee we in een berichtmodel een CIM class kunnen "degraderen" tot een ***gegevensgroep***.

# RESTful API's – Resource definitie

- ✓ Een resource maakt impliciet of expliciet deel uit van een ***resource collectie*** waarvan de naam in het algemeen gelijk is aan de ***meervoudsvorm*** van de resourcetype naam (de ***resource*** Netbeheerder zit in ***resource collectie*** Netbeheerders);
- ✓ De schema generator zal, bij het parsen van het berichtmodel, alle classes met een '*MRID*' attribuut als resource bestempelen en ze daarmee impliciet (volgens HAL) voorzien van een '*_links'* en een '*_embedded'* sectie (zie HAL standaard);
- ✓ Elke resource dient *in ieder geval* vanuit "*een*" API benaderbaar te zijn middels een Top-Level URL. ***Idealiter bestaat een 1:1 relatie tussen API en resource*****,** m.a.w. de resource kent ***precies één*** Top-Level URL!

  `https://api.enexis.nl/marktpartijen/v1/netbeheerders/8712423014022`

  → Deze ***Top-Level URL*** vraagt de Netbeheerder resource op, op basis van zijn MRID (de EAN13 code van de beheerder);

  In deze context noemen we de Netbeheerder resource ook wel de ***primaire resource*** voor deze API omdat dit de direct geadresseerde resource is.

# RESTful API's – Resource definitie

- ✓ Als ***alternatief mag*** een resource tevens voorkomen als ***geassocieerde resource*** in relatie met andere resources, maar dit is dan ***uitsluitend*** een ***alternatieve adressering*** die relevant is in de context van de ***associërende resource***:

  `https://api.enexis.nl/aansluitingen/v1/meetpunten/871687110001345494/netbeheerders`

  → Identificeert de collectie van Netbeheerder resources die behoren bij de Meetpunt resource met het gespecificeerde MRID (EAN18). In deze context fungeert de Netbeheerder resource als een ***geassocieerde resource*** van de Meetpunt resource;

- ✓ Een geassocieerde resource wordt in die rol ***altijd indirect*** (dus via de associërende resource) benaderd;
- ✓ Als een resource acteert in de rol van ***geassocieerde resource*** dan ***kan*** het zijn dat in ***die context*** slechts een ***beperkt aantal attributen*** van de resource beschikbaar worden gesteld;
- ✓ Voor een ***geassocieerde resource moet*** in ***ieder geval*** een link attribuut (de "***base***" link) beschikbaar worden gesteld waarmee de ***volledige details*** van die resource zijn op te vragen (deze link verwijst naar de API waar de ***geassocieerde resource*** is te vinden in de rol van ***primaire resource***). Dit zijn ***altijd absolute links***:

```
{
  "_links": {
    "base": {"href": "https://api.enexis.nl/marktpartijen/v1/netbeheerders/8712423014022"}
```

- ✓ Daarnaast dient de geassocieerde resource ***altijd*** een ***MRID*** attribuut te bezitten;

# RESTful API's – Resource definitie

Twee Top-Level URL's:

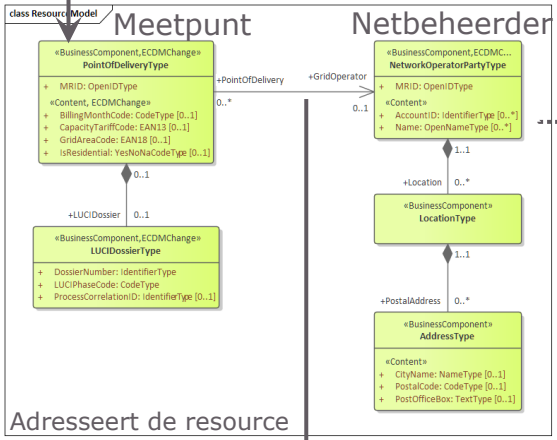
`https://api.enexis.nl/marktpartijen/v1/netbeheerders/8712423014022`

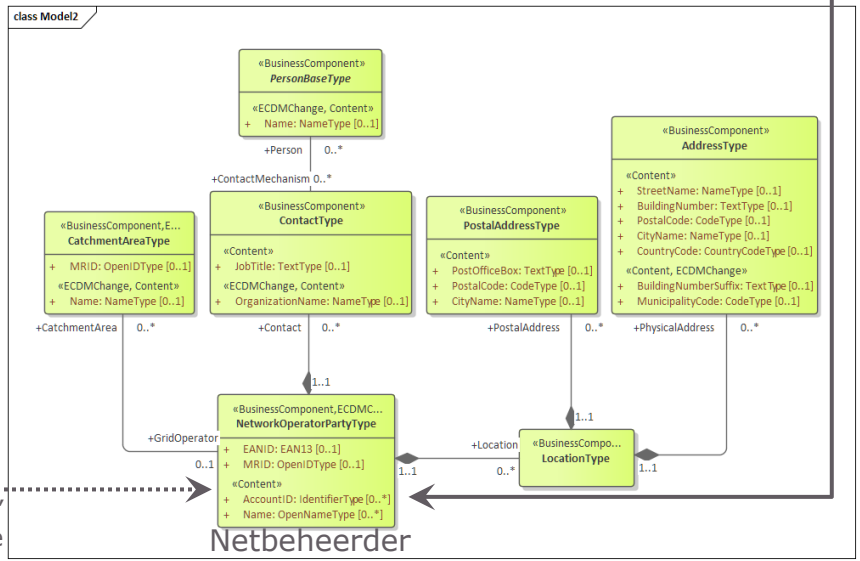
`https://api.enexis.nl/aansluitingen/v1/meetpunten/871687110001345494`

Adresseert de resource 'Meetpunt' als ***primaire*** resource

Adresseert de resource 'Netbeheerder' als ***primaire*** resource

resource in rol van **geassocieerde** resource kent minder attributen dan in rol van **primaire** resource




Adresseert de resource 'Netbeheerder' als **geassocieerde** resource (van de resource 'Meetpunt')

**Geassocieerde** resource bevat '**base**' link naar **primaire** resource

`https://api.enexis.nl/marktpartijen/v1/netbeheerders/8712423014022`

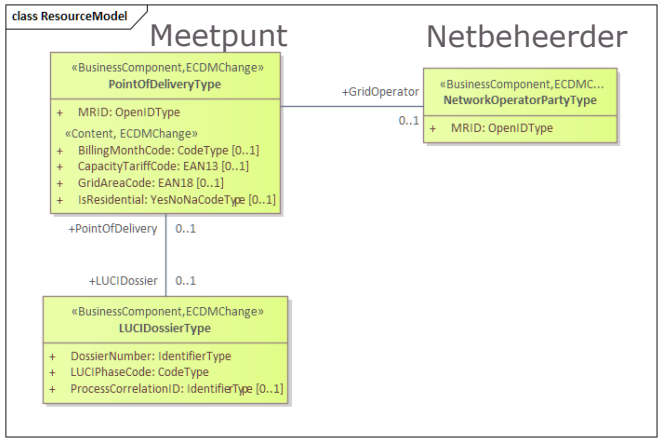
`https://api.enexis.nl/aansluitingen/v1/meetpunten/871687110001345494/netbeheerders/8712423014022`

# RESTful API's – Resource definitie

- ✓ Het kan voorkomen dat van een geassocieerde resource ***uitsluitend*** het MRID attribuut aanwezig is en verder ***geen*** andere attributen en/of associaties. We spreken dan van een ***resource reference***;
- ✓ De associërende resource stelt in dat geval ***uitsluitend*** een ***link*** beschikbaar naar de API waar deze geassocieerde resource optreedt als primaire resource;
  - ➢ Als het niet mogelijk is om een externe link aan te bieden dan mag de geassocieerde resource ***niet*** als zodanig worden gemodelleerd (en vervalt de resource tot een ***gegevensgroep*** met alleen een ***ID*** attribuut);
- ✓ De resource reference kan ***niet*** worden geëxpandeerd (er is immers niets te expanderen);

In het voorbeeld hiernaast is *PointOfDelivery.Gridoperator* een ***resource reference.***
Deze zal uitsluitend tot uitdrukking komen in een ***link*** vanuit PointOfDelivery, b.v.:

```
{
  "_links": {
    "self": {……},
    "GridOperator": {
       "href": "……/marktpartijen/v1/netbeheerders/8712423014022"}
  }
}
```



# RESTful API's – Resource definitie

- ✓ Voor operaties op een resource (met uitzondering van de 'create' die via de collectie loopt), dient de URL altijd de unieke sleutel (***MRID***) van de betreffende resource te specificeren:

  `https://api.enexis.nl/marktpartijen/v1/netbeheerders/8712423014022`

  - ➢ De operatie '*GET*' ***haalt exact één*** resource op met gespecificeerde MRID;
  - ➢ De operatie '*PUT*' ***overschrijft*** de gegeven resource met gespecificeerde MRID;
  - ➢ De operatie '*DELETE*' ***verwijdert*** de resource met gespecificeerde MRID;
  - ➢ De operatie '*PATCH*' voert een ***gedeeltelijke update*** uit op de resource met gespecificeerde MRID;

# RESTful API's – Resource definitie

- ✓ Een ***MRID*** is een ***opaque datatype***. Met andere woorden: de ontvanger mag ***geen*** aannames doen over het ***formaat*** of de ***betekenis***, anders dan dat de MRID een gegeven resource op unieke wijze kan identificeren binnen een ***gegeven bron***;
- ✓ In het geval dat de MRID een waarde representeert die voor de ontvanger betekenis dient te hebben dan dient hiervoor een ***separaat***, expliciet, attribuut ***naast*** MRID te worden gebruikt;
  - Voorbeeld: voor een aansluiting geldt de EAN code van de aansluiting typisch als MRID. Echter, omdat de ontvanger niet mag aannemen dat dit zo is (en het zou, afhankelijk van de bron, kunnen wijzigen), wordt de EAN daarnaast als expliciet attribuut opgenomen.
- ✓ Om te kunnen bepalen wat de (functionele) bron is voor een gegeven MRID bezit ***elk*** MRID attribuut een ***supplementary attribuut*** '*identificationScheme*' waarmee, middels een URI, de MRID wordt geclassificeerd. Met behulp van het *identificationScheme* en een ***conversie API*** is het tevens mogelijk om een gegeven MRID te '*her-classificeren*' om zo b.v. aanvullende resource informatie uit een andere bron te lezen (waar voor de gegeven resource dan ook een andere MRID waarde bij hoort).

  Voorbeeld van een EAN MRID classificatie:

```
"MRID": {"content": "87168711000142784 4",
         "_identificationScheme": "urn:x-enexis:ecdm:schemes:mrid:connections:ap:e"}
```

# RESTful API's – Resource definitie

Samengevat:

1. Een resource in ECDM is een (UML) class in een berichtmodel dat in het bezit is van een ***MRID*** (*Master Resource Identifier*) attribuut;
2. Een resource dient ***altijd*** benaderbaar te zijn middels een ***Top-Level URL*** waarin de naam van de bijbehorende *resource collectie* terugkomt (i.h.a. de resource naam in meervoud) en waarbinnen de resource gezocht kan worden, of direct opgehaald op basis van zijn MRID:

   `https://api.enexis.nl/marktpartijen/v1/netbeheerders/8712423014022`

3. In een gegeven RESTful API kan een resource optreden als een ***geassocieerde resource*** (in het voorbeeld is Meetpunt de "*associërende*" resource en Netbeheerder de "*geassocieerde*" resource):

   `https://api.enexis.nl/aansluitingen/v1/meetpunten/871687110001345494/netbeheerders/8712423014022`

4. Een geassocieerde resource ***kan*** minder attributen bevatten dan de primaire resource;
5. Een geassocieerde resource wordt normaalgesproken ***alleen*** via een ***link*** teruggegeven naar de cliënt maar kan expliciet in de response worden opgenomen middels de '*_expand*' parameter (***tenzij*** er sprake is van een ***resource reference;*** die kan ***niet*** worden geëxpandeerd);
6. Een geassocieerde resource bevat eventueel een link ('***base***' URL) waarmee de primaire resource kan worden opgehaald;

# RESTful API’s

1. Resource en geassocieerde resource;
2. **Sub-resource;**
3. Resource collectie;
4. Gegevensgroep;
5. Attribuut;
6. Gereserveerde namen;
7. Eager- en Lazy loading;

# RESTful API's – Sub-resource definitie

- ***Een sub-resource is een UML Class in een berichtmodel die uniek identificeerbaar is middels een MRID (Master Resource Identifier) attribuut, doch uitsluitend relevant is in de context van een "eigenaar" resource → een sub-resource heeft een "part-of" relatie met zijn eigenaar;***
- ***Een sub-resource is adresseerbaar middels een URL die altijd de eigenaar dient te omvatten***;

✓ sub-resources kunnen dus ***niet*** op zichzelf bestaan:

`https://api.enexis.nl/meter-registratie/v1/slimme-meters/E000900000149/registers/1.8.0`

→ Vraag de Register sub-resource op van de meter met MRID E000900000149, op basis van de register MRID (de OBIS code van het register);

✓ Meter registers kunnen niet op zichzelf bestaan, ze zijn afhankelijk van de meter waar ze een onderdeel van zijn en ze zullen dan ook ***nooit*** via een Top-Level URL te benaderen zijn. Een sub-resource dient altijd volledig te zijn beschreven in de context van zijn eigenaar en de '*self*' link dient dan ook ***altijd*** de ***eigenaar*** te bevatten (zie voorbeeld hierboven);

✓ De MRID van de sub-resource ***is uitsluitend uniek in de context van zijn "eigenaar" resource***;

# RESTful API's – Sub-resource definitie

- ✓ De ***volledige definitie*** van een sub-resource is altijd te vinden in ***dezelfde API*** waar ook de eigenaar resource in is gedefinieerd, immers, de sub-resource is volledig afhankelijk van die eigenaar!
- ✓ Als de parent van de sub-resource wordt uitgevraagd zal de sub-resource ***alleen*** als een ***link*** worden teruggegeven (deze link zal ook de parent omvatten);

```
GET https://api.enexis.nl/meter-registratie/v1/slimme-meters/E000900000149 →

"_links": {
  "self": {"href": ……},
  "Registers": {"href": "https://...../slimme-meters/E000900000149/registers"}
```

- ✓ Middels de '*_expand*' parameter kan de sub-resource als onderdeel van de parent worden teruggegeven (in plaats van de link);
- ✓ Als de sub-resource een ***collectie*** is (zoals bij registers, er zijn er altijd meer dan 1), dan zal er standaard altijd een ***link naar de Collectie*** terug worden gegeven ***in plaats van*** de individuele items in de collectie. Als je in plaats daarvan de individuele items wilt hebben kan je de collectie expanderen in een lijst ***item objecten*** door parameter *'_expand=<collectie-naam>'* mee te geven:

```
GET https://api.enexis.nl/meter-registratie/v1/slimme-meters/E000900000149?_expand=Registers
```

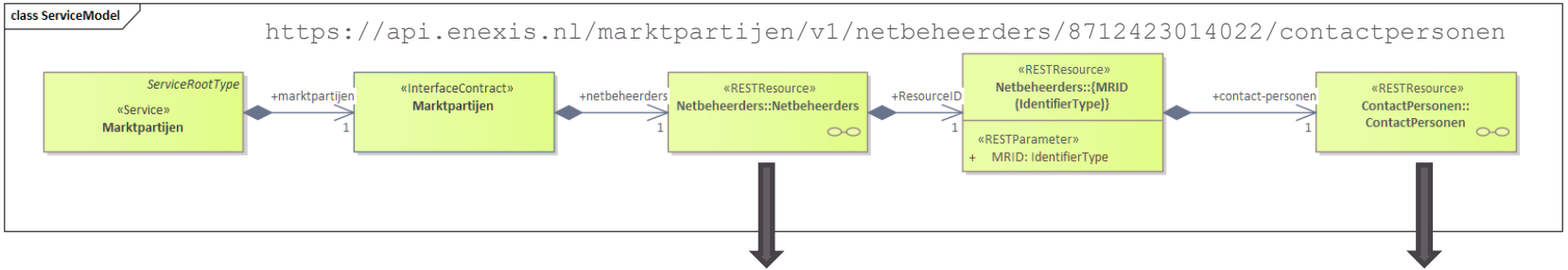
# RESTful API's

1. Resource en geassocieerde resource;
2. Sub-resource;
3. **Resource collectie;**
4. Gegevensgroep;
5. Attribuut;
6. Gereserveerde namen;
7. Eager- en Lazy loading;



# RESTful API's – Resource collectie definitie

- ✓ Een resource maakt deel uit van een ***resource collectie*** waarvan de naam in het algemeen gelijk is aan de ***meervoudsvorm*** van de resource type naam (de ***resource*** Netbeheerder zit in ***resource collectie*** Netbeheerders);
- ✓ Een RESTful API heeft te allen tijde één of meer ***top-level resources*** (de resource direct "*na*" het versienummer van de API) en dit zijn ***altijd*** (impliciete) ***resource collecties***;
- ✓ Een resource kan vervolgens associaties bezitten met andere resources. Als de cardinaliteit van deze associatie groter is dan 1 ontstaan hier ***geassocieerde resource collecties***;

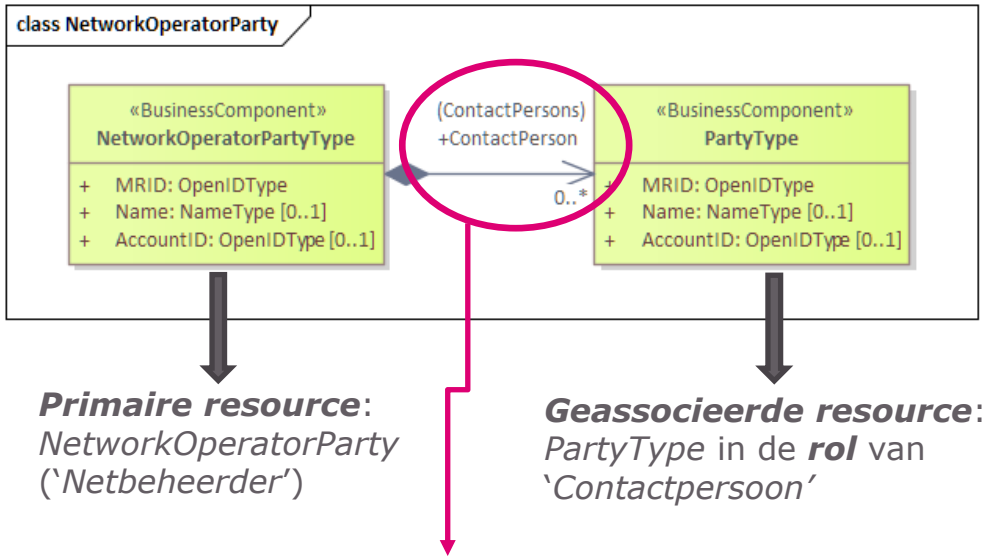



***Top-level resource collectie***:
*Netbeheerders* collectie

***Geassocieerde resource collectie***:
*Contactpersonen* collectie

# RESTful API's – Resource collectie definitie

- ✓ In het resource model zien we de ***collectie*** terug in de vorm van een ***associatie*** tussen de *Netbeheerder* resource en de *Contactpersoon* resource met een ***cardinaliteit > 1***:



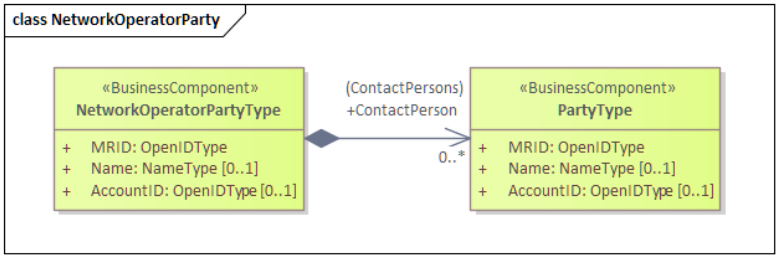

- ✓ Omdat de standaard rol naam in het model hier enkelvoudig is (‘*ContactPerson*’) dienen we een ***Alias*** toe te voegen om de naam van de collectie in de meervoudsvorm te krijgen;
- ✓ Hierdoor is de feitelijke naam van de ***geassocieerde collectie*** van contactpersonen dus ‘*ContactPersons*’;
- ✓ Dit is dan ook de naam die je moet gebruiken als je naar de collectie verwijst vanuit een parameter:

`‘_expand=ContactPersons’`

# RESTful API's – Resource collectie definitie

✓ We onderkennen een tweetal typen collecties:

1. ***Expliciete collecties***; Deze zie je expliciet terug in het resource model in de vorm van een associatie tussen twee resources, waarbij de cardinaliteit van deze associatie groter is dan 1 (zie figuur voor een voorbeeld); Deze collectie bevraag je via de eigenaar, '*NetworkOperatorParty'*. Er komt dan een link terug naar de collectie met naam '*ContactPersons*'. Deze ***link*** implementeert op zijn beurt een ***impliciete collectie*** (zie hieronder). Je kan de collectie ook expanderen middels parameter '`_expand=ContactPersons`'. Het KAN zijn dat dit een foutmelding geeft bij grote collecties.




2. ***Impliciete collecties;*** Deze zie je niet in het resource model maar in de URL:
   `https://api.enexis.nl/marktpartijen/v1/netbeheerders/8712423014022/contactpersonen`

   Impliciete collecties gedragen zich als een gewone resource, ze hebben attributen en je kan ze bevragen. Omdat het implementatie-technisch ***niet mogelijk*** is om automatisch de meervoudsvorm van de elementen in de collectie te bepalen, zullen impliciete collecties altijd de naam '*Items*' gebruiken om de elementen in de collectie te adresseren.

   De pagina's hierna gaan in op de functionaliteit en gedrag van ***impliciete collecties***;

# RESTful API's – Resource collectie definitie

- ***Een (impliciete) resource collectie is een gespecialiseerde resource die toegang geeft tot (verzamelingen van-) resources (en/of sub-resources);***
- ***Een resource collectie bevat resources van hetzelfde type (of eventueel gespecialiseerde varianten van dat type);***
- ***Een resource collectie maakt het mogelijk om resources te zoeken en te creëren;***

`https://api.enexis.nl/marktpartijen/v1/netbeheerders`
→ Representeert de resource collectie van Netbeheerder resources;

`https://api.enexis.nl/meter-registratie/v1/meters`
→ Representeert de resource collectie van Meter resources. Deze kan zowel Slimme Meters als Conventionele Meters bevatten (beide specialisaties van Meter);

- ✓ Een resource collectie bevat typisch de operaties '*POST*' (creëer nieuwe resource in de collectie) en '*GET*' (zoek in de collectie);
- ✓ De operatie '*PUT*' ***vervangt de gehele collectie*** door de meegezonden resources;
- ✓ De operatie '*DELETE*' ***verwijdert de gehele collectie*** (alle resources worden verwijderd);
- ✓ De operatie '*PATCH*' ***overschrijft*** (indien bestaand) of ***voegt toe*** (indien niet bestaand) de meegezonden resources;

# RESTful API's – Resource collectie definitie

- ***Doelstelling is om een resource collectie qua gedrag zoveel mogelijk consistent te laten zijn met elke andere resource;***

✓ Dit impliceert dat de collectie '*_expand*', '*_fields*' en '*_exclude*' dient te ondersteunen;

✓ Dit impliceert ook dat de collectie ***attributen*** bezit die de cliënt kan uitvragen, de collectie "*metadata*":

- '*TotalCount*', '*RemainingCount*', '*ReturnedCount*' en '*MoreIndicator*' leveren informatie over het aantal elementen in de collectie;
- De HAL standaard beschrijft niet op een formele wijze hoe een collectie zich dient te gedragen, maar de algemene Internet opinie lijkt te zijn dat een collectie te allen tijde een geëxpandeerde lijst van resources representeert. Dus, als uitzondering op het gedrag van "reguliere" resources, levert de collectie de items (lijst van beheerde resources) als een "auto-expanded" lijst;

✓ Zoals elke resource kunnen de attributen worden onderdrukt met '*_exclude*' en selectief worden gekozen met '*_fields*';

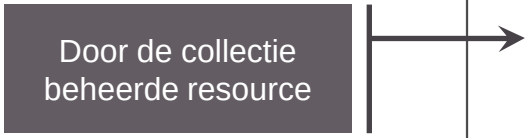
✓ Voor de ***items*** in de collectie kunnen we '*_expand*' gebruiken om sub-resources en/of geassocieerde resources te expanderen;

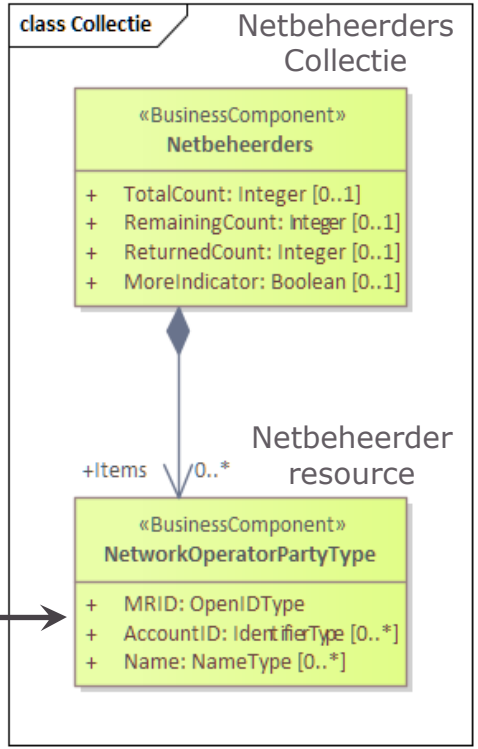
✓ De volgende slides gaan hier in meer detail op in;



# RESTful API's – Resource collectie definitie

- ✓ Een resource collectie is een ”***impliciete resource***” (zonder MRID) met als ***attributen*** metadata over de collectie als geheel en een *verzameling van door die collectie beheerde resources*.
- ✓ Om de (mogelijk grote) collectie te bevragen kan gebruik worden gemaakt van “***paginering***” waarmee ***per request een gedeelte*** van de beheerde resources is op te vragen;
- ✓ Een resource collectie kent daarnaast een impliciet attribuut ‘***Items’*** die de lijst met ***geassocieerde resources*** representeert. De cardinaliteit hiervan is (uiteraard) 0..n (zie plaatje hiernaast);
- ✓ Collecties kunnen speciale operaties zoals ***filteren***, ***sorteren*** en ***vinden*** ondersteunen;



# RESTful API's – Resource collectie definitie

- ✓ Items zijn typisch op te vragen middels een '*GET*' op de collectie (al dan niet in combinatie met filter parameters);
- ✓ Als de verzameling filter parameters te groot of te complex is, gebruiken we '*POST*' als alternatief en geven we de parameters door middels een, willekeurig complexe, HTTP body datastructuur;
- ✓ Items werkt (bij grote collecties) in combinatie met "*paginering"* om door de collectie te kunnen bladeren;
- ✓ Het teruggeven van items kan worden ***onderdrukt*** middels `'_exclude=Items'`. Hierdoor krijg je feitelijk ***alleen*** de collectie ***metadata*** (dus de attributen '*TotalCount*', '*RemainingCount*', etc.);
- ✓ Het is ook mogelijk om de items te onderdrukken door, middels `'_fields'`, alleen de ***metadata attributen*** op te vragen;
- ✓ Andersom kan je ook de metadata onderdrukken door expliciet '`_fields=Items`' te specificeren, waarmee je dus ***alleen*** de ***collectie items*** opvraagt;



# RESTful API's – Resource collectie definitie

- ✓ Binnen een collectie wordt de ***lijst met items*** gezien als ***impliciet onderdeel*** van de collectie (hij is immers per definitie al geëxpandeerd tot een soort van gegevensgroep). Dat houdt in dat je de naam ervan ***niet*** hoeft te specificeren om attributen, onderliggende gegevensgroepen en/of geassocieerde resources te benaderen. Er is slechts één zo'n lijst dus de naam is eenduidig;
- ✓ Als er sprake is van ***onderliggende gegevensgroepen***, ***geassocieerde-*** en/of ***sub-resources***, dan moeten die ***wel*** volledig worden gespecificeerd, relatief ten opzichte van de lijst;
- ✓ De '*_fields*' parameter is nuttig om uit de lijst met items alleen specifieke attributen te filteren zodat de response niet te groot wordt. In tegenstelling tot '*normale*' resources hoeft dit voor de items in de lijst ***niet*** te worden gecombineerd met '`_expand=Items`' omdat de item lijst ***per definitie*** in expanded vorm wordt teruggegeven;

  `GET https://api.enexis.nl/medewerkers/v1/ander-personeel?_fields=AccountID,Name`
  → Retourneert per element uit de lijst met collectie items alleen de attributen '*AccountID*' en '*Name*'.

  `GET https://api.enexis.nl/medewerkers/v1/ander-personeel?_expand=Afdeling&_fields=Afdeling.Name`
  → Expandeert de geassocieerde resource '*Afdeling*' en retourneert per element uit de lijst met collectie items alleen het attribuut '*Name*' van die geassocieerde '*Afdeling*' resource.

# RESTful API's – Resource collectie definitie

- ✓ '*GET*' Operaties op een resource collectie ***kunnen*** informatie teruggeven waarmee we door de collectie kunnen manoeuvreren ("***paginering***"). Deze informatie komt dan terug in de vorm van een set links:

```
{
  "_links": {
    "self": {"href": "https://api.enexis.nl/medewerkers/v1/ander-personeel?_page=7&_limit=10"},
    "first": {"href": "https://api.enexis.nl/medewerkers/v1/ander-personeel?_page=1&_limit=10"},
    "last": {"href": "https://api.enexis.nl/medewerkers/v1/ander-personeel?_page=20&_limit=10"},
    "next": {"href": "https://api.enexis.nl/medewerkers/v1/ander-personeel?_page=8&_limit=10"},
    "prev": {"href": "https://api.enexis.nl/medewerkers/v1/ander-personeel?_page=6&_limit=10"},
    "item": {"href": "https://api.enexis.nl/medewerkers/v1/ander-personeel/{mrid}",
             "templated": true}
```

- ✓ Een resource collectie bestaat uit 0 – n "***pagina's***" met resources. Elke pagina bevat een aantal resources waarbij filter parameter '*_limit*' door de cliënt moet worden meegegeven om het aantal terug te geven resources per pagina te specificeren. Het voorbeeld hieronder specificeert een pagina grootte van 20 resources per response bij het opvragen van de 10'e pagina:

```
GET https://api.enexis.nl/medewerkers/v1/ander-personeel?_page=10&_limit=20
```

- ✓ De waarde van '*_limit*' moet ***per request*** worden meegegeven, de server zal hem ***niet onthouden (stateless)!***



# RESTful API's – Resource collectie definitie

- ✓ Parameter '*_limit*' is ***verplicht*** in ***combinatie*** met '*_page*', beiden ***mogen gezamenlijk*** worden weggelaten;

  De response hangt in dat laatste geval af van de collectie en van de server implementatie. Bij grote collecties ***kan*** het zijn dat een ***error 400*** ("*Syntax fout*") wordt teruggegeven. Bij kleine collecties ***kan*** het zijn dat de ***gehele*** collectie wordt teruggegeven. De RESTful API documentatie dient dit gedrag nader te specificeren;
- ✓ Een onrealistisch hoge waarde van '*_limit*' dient een **error 422** ("*Semantische fout*") op te leveren;

  '*_limit*' is in ieder geval onrealistisch hoog indien geldt dat '*_limit*' x '*resource size*' > 8 MByte. Met andere woorden: bij "kleine" resources laat je een hoger maximum toe dan bij "grote" resources. De server bepaalt uiteindelijk wat als realistische waarde voor '*_limit*' wordt toegestaan want, los van de omvang, kan ook de tijd die het kost om het gevraagde aantal resources te verzamelen een rol spelen;
- ✓ De cliënt kan expliciet een bepaalde pagina opvragen middels de '***_page***' parameter. Deze representeert een index in de totale lijst van pagina's met grootte '*_limit*'. Indien niet gespecificeerd geldt **default pagina 1** (de ***eerste pagina*** heeft ***altijd*** nummer 1);
- ✓ Navigeren ***voorbij het einde*** van de collectie zal altijd een ***response 204*** ("*Niets gevonden*") opleveren;
- ✓ De '*self*' link in een **collectie** bevat **altijd** de index van de ***huidige pagina*** in de collectie;

  Typisch bevat '*self*' de URL die is gebruikt voor de RESTful API aanroep. Als het een collectie betreft en er is ***geen expliciete*** '*_page*' gespecificeerd, zal in '*self*' de URL ***inclusief*** '`_page=1`' worden gezet, wat de default is;

# RESTful API's – Resource collectie definitie

- ✓ De pagineringslinks bevatten, naast de '*_page*' parameter, ook de '*_limit*' parameter waarmee de request is ingediend. De aangegeven pagina's gelden namelijk ***uitsluitend*** voor de gespecificeerde '*_limit*'!
- ✓ De teruggegeven pagina nummers zullen door de server ***per request afzonderlijk*** worden berekend voor de ***gegeven*** waarde van '*_limit*' uit het ***ontvangen request*** met de ***aanname*** dat alle voorgaande requests ***diezelfde*** '*_limit*' hebben gehanteerd. Om die reden is het aan te bevelen om binnen een gegeven sessie de waarde van '*_limit*' voor elke request constant te houden;
- ✓ Een collectie ***kan*** additionele attributen bevatten in de response die informatie geven over de collectie:
  - ➢ ***TotalCount*** = Totaal aantal resources in de resource collectie op moment van de request (kan variëren per request bij dynamische collecties);
  - ➢ ***RemainingCount*** = Totaal aantal resources minus het aantal resources dat al is teruggegeven (merk op dat de server dit 'stateless' kan bepalen op basis van het gevraagde pagina nummer en het aantal resources per pagina);
  - ➢ ***ReturnedCount*** = Het aantal resources dat is teruggegeven in de ***huidige response*** (kan kleiner of gelijk zijn aan '***_limit***');
  - ➢ ***MoreIndicator*** = Geeft aan of er nog meer resources zijn om op te halen, zonder expliciete aantallen te noemen. Deze indicator is 'true' zolang we nog niet de laatste pagina hebben opgevraagd;
- ✓ ***RemainingCount*** en ***MoreIndicator*** zijn ***mutual-exclusive***: een RESTful API gebruikt de een of de ander;



# RESTful API's – Resource collectie definitie

- ✓ Omdat *TotalCount* en *RemainingCount* ”*dure*” operaties (in executie tijd) kunnen zijn, zullen niet alle collecties ze ondersteunen! Het wordt aanbevolen om wel ***altijd*** *ReturnedCount* en *MoreIndicator* te implementeren omdat dit de cliënt kan helpen om efficiënt door de collectie te lopen (*ReturnedCount* is een extra bescherming om te zien of de ontvangen response compleet is en *MoreIndicator* helpt bij het bouwen van een simpele ”*uitvraag-lus*” om de hele collectie in te lezen);
- ✓ Om ook toegang te hebben tot de collectie metadata als, voor welke reden dan ook, de response payload niet toegankelijk is of er geen gebruik wordt gemaakt van hypermedia controls, dient specifieke collectie metadata tevens beschikbaar te zijn in de vorm van ***HTTP Header Parameters***:

| Header parameter: | Verplicht / Optioneel | Toelichting: |
|---|---|---|
| X-Pagination-Page | Verplicht | Nummer van de huidige pagina. |
| X-Pagination-Limit | Verplicht | Maximaal aantal resultaten per pagina. |
| X-Pagination-Count | Optioneel | Totaal aantal pagina’s in de collectie (bij gegeven X-Pagination-Limit). |
| X-Total-Count | Optioneel | Totaal aantal elementen in de collectie. |



# RESTful API's – Resource collectie definitie

- ✓ De navigatie links van een collectie zijn ***gestandaardiseerd***:
  - ➢ '*first*': Levert de ***eerste pagina*** in de collectie voor een gegeven '*_limit*';
  - ➢ '*last*': Levert de ***laatste pagina*** in de collectie voor een gegeven '*_limit*';
  - ➢ '*next*': Levert de ***direct opvolgende pagina*** in de collectie voor een gegeven '*_limit*';
  - ➢ '*prev*': Levert de ***direct voorgaande pagina*** in de collectie voor een gegeven '*_limit*';
  - ➢ '*item*': Levert het ***exacte item*** uit de collectie met de gegeven MRID. Dit is een ***template URI*** waarbij de ***variabele*** `{mrid}` moet worden vervangen door de MRID van het gevraagde item;
- ✓ De links zijn **dynamisch** en worden **alleen teruggegeven** als er ook informatie bij hoort (b.v. op de laatste pagina van de collectie is er geen '*next'* en ook geen '*last'*, op de eerste pagina geen '*prev'* en ook geen '*first'*, etc.).



# RESTful API's – Resource collectie definitie

Hieronder een ***voorbeeld*** van de response op een ***initiële*** uitvraag van een collectie waarbij de cliënt per request ***maximaal 2 items*** terug wil krijgen (bij weglaten van '*_page*' uit de request geldt als ***default altijd expliciet "`_page=1`"***):

GET https://api.enexis.nl/marktpartijen/v1/netbeheerders?_limit=2:


```
{
    ”TotalCount": 7,
    ”RemainingCount": 5,
    ”ReturnedCount": 2,
    "_embedded": {
        ”Items": [
            {
                "MRID": "8712423014022",
                "Name": "Enexis",
                "_links": {
                    "self": {"href": "https://api.enexis.nl/marktpartijen/v1/netbeheerders/8712423014022"}
                }
            },
            {
                "MRID": "8716892000005",
                "Name": "Stedin",
                "_links": {
                    "self": {"href": "https://api.enexis.nl/marktpartijen/v1/netbeheerders/8716892000005"}
                }
            }
        ]
    },
    "_links": {
        "self": {"href": "https://api.enexis.nl/marktpartijen/v1/netbeheerders?_page=1&_limit=2"},
        "next": {"href": "https://api.enexis.nl/marktpartijen/v1/netbeheerders?_page=2&_limit=2"},
        "last": {"href": "https://api.enexis.nl/marktpartijen/v1/netbeheerders?_page=4&_limit=2"},
        ”item”: {“href”: “https://api.enexis.nl/marktpartijen/v1/netbeheerders/{mrid}”, “templated”: true}
    }
}
```

# RESTful API's – Resource collectie definitie

- ✓ In een ***"dynamische" collectie***, d.w.z. een collectie waarbinnen willekeurig resources kunnen bijkomen en afvallen tussen opvolgende '*GET*' requests, kan in plaats van pagina's ook worden gekozen voor adresseren op basis van de ***primaire sleutel*** (het MRID) van de resources:

```
{
  "_links": {
    "self": {"href": "https://api.enexis.nl/eventregister/v1/meter-events?_from=234"},
    "first": {"href": "https://api.enexis.nl/eventregister/v1/meter-events?_from=10"},
    "last": {"href": "https://api.enexis.nl/eventregister/v1/meter-events?_from=341231"},
    "next": {"href": "https://api.enexis.nl/eventregister/v1/meter-events?_after=234"},
    "prev": {"href": "https://api.enexis.nl/eventregister/v1/meter-events?_before=234"},
    "item": {"href": "https"//api.enexis.nl/eventregister/v1/meter-events/{mrid}",
             "templated": true}
```

- ✓ Merk op dat de '*self*' link in dit geval semantisch moet worden geïnterpreteerd als "*de eerste resource die in deze response is teruggegeven heeft een MRID van 234*"!
- ✓ In plaats van een pagina nummer geven we hier het ***relevante MRID*** op, op basis waarvan we willen lezen. Onderstaand voorbeeld geeft maximaal 100 events terug, startend ***direct na*** het event met MRID = 234:

```
GET https://api.enexis.nl/eventregister/v1/meter-events?_after=234&_limit=100
```



# RESTful API's – Resource collectie definitie

- ✓ De links bevatten hier geen '*_limit*' component aangezien de indices '*absoluut*' zijn. Gegeven de volgende request met mogelijke response:

```
GET https://api.enexis.nl/eventregister/v1/meter-events?_after=230&_limit=10
```

```
{
  "_links": {
    "self": {"href": "https://api.enexis.nl/eventregister/v1/meter-events?_from=234"},
    "first": {"href": "https://api.enexis.nl/eventregister/v1/meter-events?_from=10"},
    "last": {"href": "https://api.enexis.nl/eventregister/v1/meter-events?_from=341231"},
    "next": {"href": "https://api.enexis.nl/eventregister/v1/meter-events?_after=247"},
    "prev": {"href": "https://api.enexis.nl/eventregister/v1/meter-events?_before=234"}
```

- ✓ In geval van ***sleutel-gebaseerde paginering*** hebben de link namen een ***afwijkende betekenis***:
  - '*self*' geeft hier aan dat het ***eerste item in de response set*** een MRID heeft van 234. Kennelijk bestaan 231, 232 en 233 niet (meer);
  - '*first*' adresseert hier het ***eerste item* in de collectie**, dat kennelijk een MRID van 10 heeft;
  - '*last*' identificeert het ***laatste item in de collectie;***
  - '*next*' geeft hier aan dat om de ***volgende set*** responses op te vragen, je moet starten ***na*** item 247. Dit MRID is altijd het ***laatste*** MRID dat je in de ***huidige response*** hebt teruggekregen;
  - '*prev*' geeft hier aan dat om de ***voorgaande set*** responses op te vragen, je moet starten ***voor*** item 234. Dit MRID is altijd het ***eerste*** MRID dat je in de ***huidige response*** hebt teruggekregen;

# RESTful API's – Resource collectie definitie

- ✓ Merk op dat, alhoewel de '*self*' link ***lijkt*** op een '*GET*' van event 234, dat niet zo is want je geeft een ***set met resources*** terug en ***niet*** een individuele resource. Wat hier feitelijk staat is: "*voor deze response staat de index binnen de collectie nu te wijzen naar de resource met MRID 234*".
- ✓ Nog een subtiel verschil tussen de verschillende parameters:

  `GET https://api.enexis.nl/eventregister/v1/meter-events?_from=234&_limit=100`
  → Haalt (maximaal) 100 meter-event resources op, ***startend met MRID = 234***;

  `GET https://api.enexis.nl/eventregister/v1/meter-events?_after=234&_limit=100`
  → Haalt (maximaal) 100 meter-event resources op, ***startend met het eerstvolgende MRID dat na MRID = 234 komt***. Dat ***kan*** dus 235 zijn, maar kan ook best 257 zijn als er tussenin een aantal resources zijn verwijderd.
- ✓ Met andere woorden: '*_from=XXX*' plaatst de index in de collectie met resources ***op*** de resource met MRID 'XXX' terwijl '*_after=XXX'* de index plaatst op de ***eerstvolgende*** resource die in de collectie voorkomt ***NA*** 'XXX'. Tot slot zet '*_before=XXX'* de index op de resource ***direct voorafgaande*** aan 'XXX'.
- ✓ Specificeren van '*_before=XXX'* zal ook betekenen dat de response "***achteruit***" gaat:

  `GET https://api.enexis.nl/eventregister/v1/meter-events?_before=234&_limit=100`
  → Haalt (maximaal) 100 meter-event resources op, ***voorafgaande*** aan het event met MRID = 234 (alle events hebben een MRID dat ***kleiner*** is dan 234);



# RESTful API's – Resource collectie definitie

## Zoeken in een collectie:

- ✓ De eenvoudigste vorm om een collectie te doorzoeken is gebruik te maken van een '*GET*' op de collectie in combinatie met één of meerdere ***filterparameters***. Deze parameters dienen ***exact*** overeen te komen met ***attributen*** van de ***resources*** in de collectie waarop gezocht kan worden;
- ✓ Gebruik parameter '*_filter*' om een komma-gescheiden lijst van expressies op te geven. Let op dat elke expressie "*URL-encoded*" moet zijn om geen fouten te krijgen bij de parsing:

```
GET https://api.enexis.nl/marktpartijen/v1/netbeheerders?_limit=20&_filter=Location.Region%3DZuid-Nederland
```

In bovenstaande expressie representeert '*%3D*' het '=' teken dat hier niet direct voor mag komen;

# RESTful API's – Resource collectie definitie

## Zoeken in een collectie:

✓ Het is mogelijk om ***conditionele expressies*** samen te stellen door middel van "***Left Hand Size Brackets***" notatie.

Ook hier geldt dat, omdat de expressie onderdeel is van de '*_filter*' parameter, deze "*URL-encoded"* moet zijn:

| Operator: | Notatie: | URL encoded: | Voorbeeld: |
|---|---|---|---|
| < | [lt] | %5Blt%5D | GET …/marktpartijen/v1/netbeheerders?_limit=20?_filter=ActiveSince%5Blt%5D%3D2020-01-01 |
| > | [gt] | %5Bgt%5D | GET …/marktpartijen/v1/netbeheerders?_limit=20?_filter=ActiveSince%5Bgt%5D%3D2020-01-01 |
| <= | [lte] | %5Blte%5D | GET …/marktpartijen/v1/netbeheerders?_limit=20?_filter=ActiveSince%5Blte%5D%3D2020-01-01 |
| >= | [gte] | %5Bgte%5D | GET …/marktpartijen/v1/netbeheerders?_limit=20?_filter=ActiveSince%5Bgte%5D%3D2020-01-01 |
| != | [not] | %5Bnot%5D | GET …/marktpartijen/v1/netbeheerders?_limit=20?_filter=Location.Region%5Bnot%5D%3DZuid-Nederland |

Door de parameter meerdere malen op te nemen is een bereik te selecteren:

```
GET …/v1/netbeheerders?_limit=20?_filter=ActiveSince%5Bgte%5D%3D2018-01-01%26ActiveSince%5Blt%5D%3D2020-01-01
```



# RESTful API's – Resource collectie definitie

## Zoeken in een collectie:

- ✓ Als er veel en/of complexe zoek criteria nodig zijn is gebruik van '*GET*' niet praktisch omdat daar ***geen request body*** aan meegegeven kan worden;
- ✓ Voor ***complexe filters*** maken we dan ook gebruik van '*POST*' waarbij de ***request body*** de data structuur bevat die we als ***filter*** willen gebruiken.
- ✓ De rest van het mechanisme blijft echter ongewijzigd: we hebben nog steeds de speciale parameters als '*_expand*', '*_limit*', '*_exclude*' en '*_fields*' en de response is nog steeds een geëxpandeerde lijst met resources;
- ✓ De grens tussen gebruik van '*GET*' en '*POST*' ligt ***rond de 5 filter parameters*;** bij meer dan 5 parameters (waarbij we de speciale parameters niet meetellen), verdient het aanbeveling om 'POST' te overwegen;
- ✓ ***Conditionele expressies ('_filter')*** kunnen ***niet*** worden gebruikt in combinatie met een ***request-body*** gebaseerd filter;



# RESTful API's – Resource collectie definitie

## Zoeken in een collectie:

- ✓ Een filter zal alleen die resultaten teruggeven waarvan de attribuutwaarden ***exact*** overeen komen met de de waarde zoals in het filter gespecificeerd;
- ✓ Filteren is alleen praktisch voor relatief eenvoudige gevallen (‘*exact match*’);
- ✓ ***Vinden*** maakt het mogelijk om, middels een ***vrije-tekst zoekmachine***, de collectie te doorzoeken op basis van ***expressies***;
- ✓ ***Vinden*** gebruikt de parameter ‘*_find*’ om een expressie op te geven waarvan we graag een match willen vinden;
- ✓ Het resultaat kunnen we dan nog filteren en sorteren;

  `…/marktpartijen/v1/netbeheerders?_find=aardwarmte&_filter=status%3Dactief&_sort=FoundingDate`

  → Zoek in de collectie van netbeheerders naar beheerders waarin het woord ‘*aardwarmte*’ voorkomt, waarvan de status ‘*actief*’ is en sorteer het resultaat op oplopende oprichtingsdatum;
- ✓ De door ‘*_find*’ ondersteunde syntax is ***afhankelijk*** van de gebruikte ***zoekmachine***. Dit kan b.v. Elasticsearch zijn;
- ✓ Vinden ondersteunt in het algemeen een veel ***krachtiger*** en ***flexibeler*** syntax dan filteren en maakt complexe expressies mogelijk;

# RESTful API's – Resource collectie definitie

**Sorteren van een collectie:**

- ✓ De response van een zoekopdracht op een collectie is een lijst met links of een lijst met geëxpandeerde objecten. Deze lijsten kunnen we ***sorteren*** middels parameter '*_sort*';
- ✓ Argument(en) aan '*_sort*' is een (lijst van) response attributen waarop gesorteerd moet worden. Deze attributen ***moeten voorkomen in de response objecten***;
- ✓ Sortering vindt "***van links naar rechts***" plaats op deze lijst, dus eerst volgens eerste attribuut, daarbinnen volgens tweede attribuut, etc.;
- ✓ Sortering vind standaard plaats in ***oplopende volgorde*** (dus alfabetisch, stijgende datum, oplopende nummering, etc.). Door voor de veldnaam een "–" te plaatsen wordt de volgorde ***omgedraaid***:

  Parameter '`_sort=-Priority&CreationDate`' zal dus sorteren van hoge- naar lage prioriteit en daarbinnen op creatie datum van oudste naar nieuwste;



# RESTful API's – Resource collectie definitie

**Kleine collecties:**

- ✓ Voor eenvoudige gevallen waar de resource collectie uit slechts een gering aantal elementen bestaat (minder dan 10) is er geen behoefte aan de overhead van alle collectie ‘features’ zoals paginering en metadata;
- ✓ In dat soort gevallen zal de collectie ***uitsluitend*** de ‘*_embedded*’ sectie met de elementen en een ‘*_links*’ sectie met een ‘*self*’ link (en eventueel een ‘*item*’ link) bevatten;
- ✓ Zie het voorbeeld hiernaast;


```
{
  “_embedded”: {
    ”Items”: [{
      "MRID": "1.8.1",
      "MeasurementUnitCode": "KWH",
      "MeasurementDirectionCode": "LVR",
      "Multiplier": "1.0",
      "_links": {
      "self": {"href": "https:/api.enexi.../slimme-meters/E000900000149/registers/1.8.1”}
      }
    },
    {
      "MRID": "1.8.2",
      "MeasurementUnitCode": "KWH",
      "MeasurementDirectionCode": "LVR",
      "Multiplier": "1.0”,
      "_links": {
        "self": {"href": "https:/api.enexi.../slimme-meters/E000900000149/registers/1.8.2”}
      }
    }]
  },
  “_links”: {
    “self”: {“href”: “https://api.enexi.../slimme-meters/E000900000149/registers”},
  }
}
```

# RESTful API's

1. Resource en geassocieerde resource;
2. Sub-resource;
3. Resource collectie;
4. **Gegevensgroep;**
5. Attribuut;
6. Gereserveerde namen;
7. Eager- en Lazy loading;



# RESTful API's – Gegevensgroep definitie

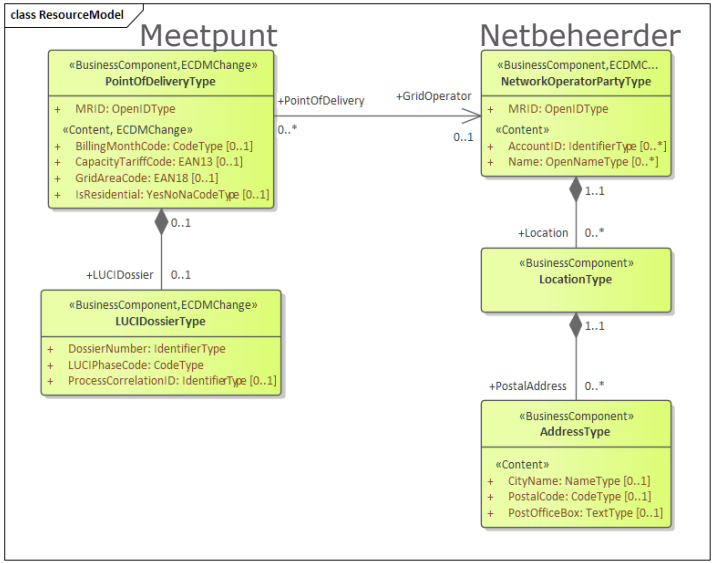
- ***Een gegevensgroep is een UML Class in een berichtmodel dat GEEN MRID attribuut bezit (het MAG een ID attribuut bezitten);***
- ***Een gegevensgroep dient te worden gezien als een samengesteld attribuut van de Class waarmee hij is geassocieerd;***
- ***Een gegevensgroep kan zijn geassocieerd met andere gegevensgroepen;***
- ***gegevensgroepen zijn NOOIT via een URL te benaderen, ze bestaan uitsluitend in de context van de eigenaar;***
- ***Een (verzameling van) gegevensgroep(en) is/zijn geassocieerd met minimaal één resource (of geassocieerde resource of sub-resource);***
- ***gegevensgroepen zijn expliciet op te vragen middels de '_fields' parameter of te onderdrukken via de '_exclude' parameter;***
- ***De '_expand' parameter heeft GEEN invloed op gegevensgroepen;***

# RESTful API's – Gegevensgroep definitie

Stel dat nevenstaand model het berichtmodel van een ***Meetpunt RESTful API*** voorstelt, dan is PointOfDelivery (Meetpunt) de ***primaire resource*** (voor deze RESTful API);

Deze heeft GridOperator (Netbeheerder) als ***geassocieerde resource*** en bezit daarnaast de ***gegevensgroep*** LUCIDossier.

Op zijn beurt kent GridOperator een ***gegevensgroep*** Location en Location kent een ***gegevensgroep*** PostalAddress;



# RESTful API's – Gegevensgroep definitie

Als we voor voorgaand voorbeeld een Meetpunt opvragen:

```
GET https://api.enexis.nl/aansluitingen/v1/meetpunten/871687110001345494
```

Dan verwachten we de volgende response:

```
{
  "MRID": "871687110001345494",
  "BillingMonthCode": "12",
  "CapacityTariffCode": "54934595312311",
  "GridAreaCode": "154376512000342112",
  "IsResidential": true,
  "LUCIDossier": {
      "DossierNumber": "1223a.09",
      "LUCIPhaseCode": "Closed",
      "ProcessCorrelationID": "443"
  },
   "_links": {
      "self": {"href": "https://api.enexis.nl/aansluitingen/v1/meetpunten/871687110001345494"},
      "GridOperator": {"href": "https://api..../meetpunten/871687110001345494/netbeheerders/7110001345494"}
   }
}
```



# RESTful API's – Gegevensgroep definitie

- ✓ We zien in de response de (verplichte) links sectie met daarin een '*self*' link naar de ***Meetpunt resource*** alsmede de link naar de Netbeheerder als ***geassocieerde resource***; De Netbeheerder link verwijst hier naar een URL binnen de ***Meetpunt API*** waar gegevens over de Netbeheerder kunnen worden opgevraagd;
- ✓ Er komen ***geen verdere details*** voor Netbeheerder voor in de response;
- ✓ De response bevat verder WEL de individuele attributen van het Meetpunt zoals voor de API gedefinieerd, inclusief de attributen van ***gegevensgroep*** LUCIDossier;
- ✓ Als we in plaats van de ***link*** naar de Netbeheerder de gedefinieerde ***attributen*** van de Netbeheerder terug willen krijgen dan kunnen we gebruik maken van de '*_expand*' parameter:

```
GET https://api.enexis.nl/aansluitingen/v1/meetpunten/871687110001345494?_expand=GridOperator
```

- ✓ Nu krijgen we de response als getoond op de ***volgende slide***;
- ✓ In de response zien we nu dat de attributen die voor Netbeheerder zijn gedefinieerd ***als onderdeel van het berichtmodel van de Meetpunt API*** worden meegestuurd binnen een "***embedded***" sectie;
- ✓ Omdat ***Netbeheerder*** een ***resource*** is, heeft hij ook weer een eigen '*links*' sectie waar de '*base*' link is te vinden die toegang geeft tot de ***primaire*** resource definitie voor Netbeheerders (zie hier hoe je deze link specificeert in ECDM);
- ✓ Netbeheerder heeft in deze context een tweetal ***geneste gegevensgroepen*** die worden meegegeven in de response.
- ✓ We spreken af dat de structuur ***altijd*** is: eerst Attributen, dan e.v.t. de Embedded sectie en tot slot de Links sectie;

# RESTful API's – Gegevensgroep definitie


```
{
   "MRID": "871687110001345494",
   "BillingMonthCode": "12",
   "CapacityTariffCode": "54934595312311",
   "GridAreaCode": "154376512000342112",
   "IsResidential": true,
   "LUCIDossier": {
      "DossierNumber": "1223a.09",
      "LUCIPhaseCode": "Closed"
      "ProcessCorrelationID": "443"
   },
   "_embedded": {
      "GridOperator": {
         "MRID": "8712423014022",
         "AccountID": "341",
         "Name": "Enexis",
         "Location": {
            "PostalAddress": {
               "CityName": "Den Bosch",
               "PostalCode": "1234AA",
               "PostOfficeBox": "23"
            }
         },
         "_links": {
            "self": {"href": "https://api.enexis.nl/aansluitingen/v1/meetpunten/871687110001345494/netbeheerders/8712423014022"},
            "base": {"href": "https://api.enexis.nl/marktpartijen/v1/netbeheerders/8712423014022}
         }
      }
   },
   "_links": {
      "self": {"href": "https://api.enexis.nl/aansluitingen/v1/meetpunten/871687110001345494"}
   }
}
```

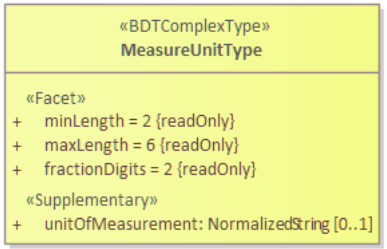
# RESTful API's

1. Resource en geassocieerde resource;
2. Sub-resource;
3. Resource collectie;
4. Gegevensgroep;
5. **Attribuut;**
6. Gereserveerde namen;
7. Eager- en Lazy loading;



# RESTful API's – Attribuut definitie

- ***Een Attribuut is een enkelvoudig (niet samengesteld) kenmerk van een UML class;***
- ***Een Attribuut heeft een naam en een classifier (type);***
- ***Een classifier kan eventueel supplementary attributen bezitten waarmee metadata aan de classifier kan worden toegevoegd (zoals eigenaar, grootheden, classificatie, etc.);***
- ***Een classifier kan eventueel facet attributen bezitten waarmee beperkingen en/of controles op waardebereik zijn af te dwingen;***




- ✓ Het model hiernaast toont de classifier '*MeasureUnitType*';
- ✓ Deze bezit een drietal Facets:
  - '*minLength'*: waarde moet minimaal 2 digits bevatten;
  - *'maxLength'*: waarde mag maximaal 6 digits bevatten;
  - *'fractionDigits'*: er mogen maximaal 2 digits achter de komma staan;
- ✓ Daarnaast bezit de classifier nog een supplementary attribuut '*unitOfMeasurement*', dat kan worden gebruikt om de meet-grootheid mee aan te geven;

# RESTful API's – Attribuut definitie

- ✓ Een attribuut definitie kan nu zijn: `GrossWeight: MeasureUnitType [0..1]`
- ✓ De ECDM Schema Generator zal dit vertalen naar de correcte syntax, wat betekent voor RESTful API's een vertaling naar JSON Schema ('*fractionDigits*' zie je hier niet terug omdat JSON Schema dit niet ondersteunt):

```
"GrossWeight": {
  "title": "MeasureUnitType",
  "description": "Gross item weight in Kg.",
  "type": "object",
  "additionalProperties": false,
  "properties": {
    "content": {
      "title": "MeasureUnitDecimal",
      "example": "12.4",
      "type": "number",
      "minLength": 2,
      "maxLength": 6
    },
    "_unitOfMeasurement": {
      "title": "NormalizedString",
      "example": "KG",
      "type": "string"
    }
  }
},
```

# RESTful API's – Attribuut definitie

- ✓ Attributen worden altijd in request- en response berichten meegegeven als ***onderdeel*** van de UML class ***waarbinnen ze zijn gedefinieerd***;
- ✓ De cliënt kan ***expliciet*** een keuze maken uit de attributen die hij ***terug wil krijgen*** met de '*_fields*' parameter;
- ✓ De cliënt kan ***expliciet*** kiezen om bepaalde attributen ***niet terug te krijgen*** met de '*_exclude'* parameter;
- ✓ ***Optionele attributen*** die ***geen waarde*** hebben worden ***niet*** uitgewisseld (ook niet als ze, middels '*_fields*', expliciet zijn opgevraagd);
- ✓ ***Verplichte attributen*** die ***geen waarde*** hebben dienen een foutmelding op te leveren (afhankelijk van request- of response).
- ✓ Een attribuut dat moet worden ***verwijderd*** (in een update operatie) krijgt als waarde '*null*';



# RESTful API's

1. Resource en geassocieerde resource;
2. Sub-resource;
3. Resource collectie;
4. Gegevensgroep;
5. Attribuut;
6. **Gereserveerde namen;**
7. Eager- en Lazy loading;



# RESTful API's – Gereserveerde namen

Binnen de ECDM API omgeving bestaan een aantal “gereserveerde” parameter namen met een bijzondere betekenis.
*Deze parameters beginnen **altijd** met een “_” om ze eenduidig te onderscheiden van “normale” URL parameters!*

1. ***Filters (eager- en lazy loading):***
   - ✓ *_expand*: Neem het berichtmodel van de geassocieerde- en/of sub-resources waarvan ***de (gekwalificeerde) naam*** is opgegeven ***op in de response scope*** (***in plaats van een link*** naar de resource terug te geven);
   - ✓ *_fields*: Neem, uit de ***response scope***, ***alleen die*** gegevensgroepen en/of attributen en/of resources op waarvan ***de (gekwalificeerde) naam*** is opgegeven;
   - ✓ *_exclude*: Laat de gegevensgroepen en/of attributen en/of resources waarvan ***de (gekwalificeerde) naam*** is opgegeven ***weg*** uit de response;

   ‘*_fields*’ en ‘*_exclude*’ zijn ‘*mutual excl*usive’: je gebruikt de een, ***of*** de ander maar ***nooit*** samen;

   *De **gekwalificeerde naam** bevat **alle** voorgaande resource namen/associaties die je moet doorlopen om bij de betreffende property uit te komen, relatief t.o.v. de primaire resource, b.v.: GridOperator.Location.PostalAddress.CityName.*

# RESTful API's – Gereserveerde namen

- ✓ ***Onderscheid*** tussen '*_expand*' enerzijds en '*_fields*' / '*_exclude*' anderzijds is belangrijk voor de begripsvorming. Deze parameters hebben zeer duidelijk ***verschillende betekenissen***!
- ✓ Met '*_expand*' kies je de ***scope*** van het response model door de ***inhoud*** van resources ***expliciet*** op te nemen in die response. Van alle niet-expliciet in '*_expand*' genoemde resources komt ***alleen een link terug;***
  Uitzonderingen zijn de ***primaire resource*** en de ***lijst van items*** in een ***collectie,*** beiden worden ***altijd*** in "*expanded*" vorm teruggegeven;

  De response scope bepaalt welke attributen en/of gegevensgroepen direct ***beschikbaar*** zijn in de response. De '*_expand*' parameter heeft, anders gezegd, tot doel om voor die betreffende response ***geassocieerde resources om te zetten in gegevensgroepen***;
- ✓ Met '***_****fields'* of '*_exclude'* kies je ***welke*** attributen en/of gegevensgroepen en/of (links naar-) geassocieerde resources je wel- of juist niet wilt ontvangen. Als je dus ***bepaalde attributen*** uit een geassocieerde- of sub-resource wilt selecteren (dus een ***bredere scope*** wilt hebben), dan dien je deze ***altijd***, middels '*_expand*', tot ***gegevensgroep*** te verklaren zodat deze attributen daadwerkelijk ***onderdeel*** worden van de ***scope***;

  De '*_fields*' en '*_exclude*' parameters moet je zien als ***filters*** op de inhoud van de ***scope***.



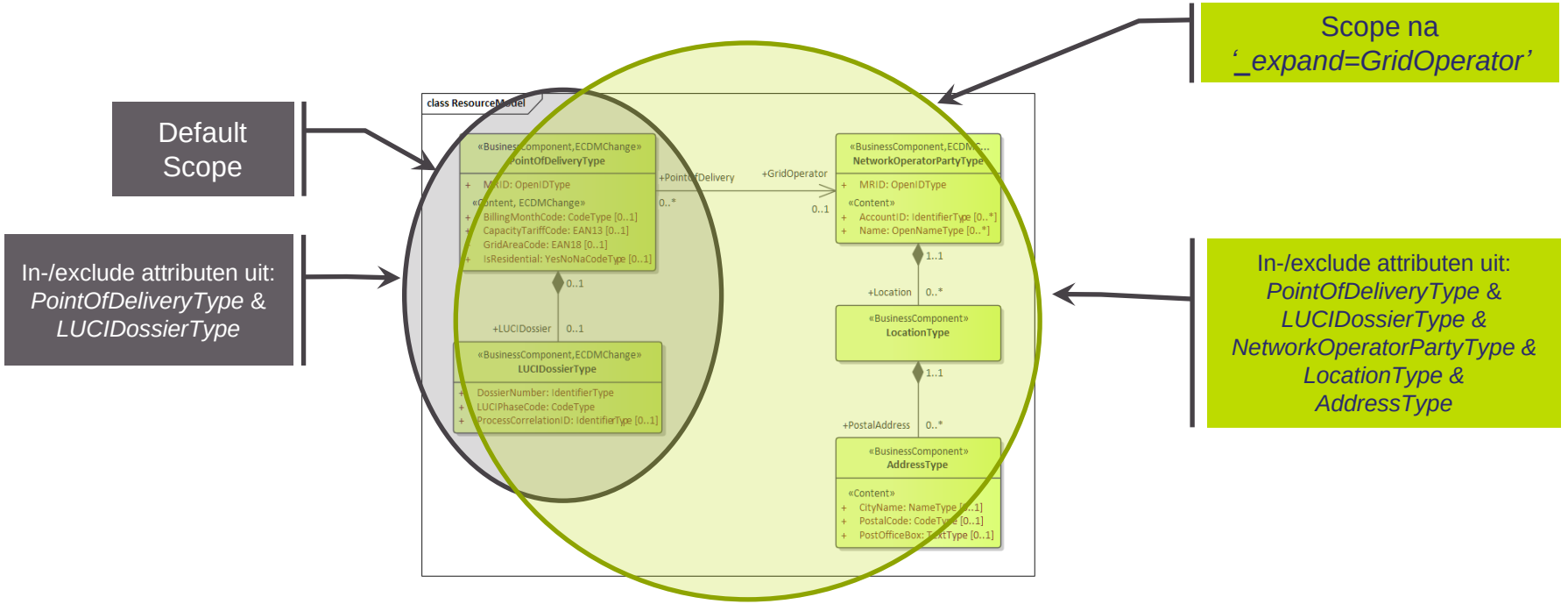
# RESTful API's – Gereserveerde namen

- ✓ Als je ***geen*** '*_fields*' of '*_exclude*' gebruikt krijg je automatisch ***alle attributen*** en ***gegevensgroepen*** terug die verplicht zijn en/of een waarde hebben ***en*** die ***onderdeel*** zijn van de ***response scope*** (verplichte attributen zouden altijd een waarde moeten hebben).

  ***Alleen*** een '*_expand*', ***zonder*** '*_fields*' of '*_exclude*' zal dus de geëxpandeerde resource ***in zijn geheel opnemen*** in de response;
- ✓ De ***volgorde*** van properties binnen een '*_fields*', '*_exclude*' of '*_expand*' parameter is ***niet relevant*** en heeft geen directe relatie met de volgorde binnen het resource Model;
- ✓ Met parameter '`_expand=all`' forceer je dat ***alle*** geassocieerde- en/of sub-resources binnen het resource model worden geëxpandeerd, zonder expliciet alle namen te hoeven opgeven;
- ✓ Met parameter '`_fields=all`' forceer je dat ***alle*** attributen die ***onderdeel zijn van de huidige scope*** worden teruggegeven. Dit heeft feitelijk hetzelfde effect als '*_fields*' weglaten;



# RESTful API's – Gereserveerde namen




*Effect van '_expand' op de **response scope***

# RESTful API's – Gereserveerde namen

- ✓ In voorgaande figuur bestaat een ***associatie*** tussen de *resource PointOfDelivery* en de *resource NetworkOperator. PointOfDelivery* is '*eigenaar*' van deze associatie die zich manifesteert als een ***attribuut*** van *PointOfDelivery* met ***naam*** *GridOperator*;
- ✓ Omdat het attribuut als ***classifier*** een *geassocieerde resource* bezit, zal dit, zonder extra maatregelen, te allen tijde als een ***HAL link*** worden teruggegeven;

  Met andere woorden: '`_fields=GridOperator`' zal uitsluitend een HAL link opleveren en ***geen*** van de ***attributen*** van *NetworkOperator* is beschikbaar in de response;
- ✓ De parameter '`_expand=GridOperator`' zorgt er feitelijk voor dat *NetworkOperator* zich gaat ***gedragen*** als een ***gegevensgroep***: de attributen worden nu toegankelijk en er zal ***geen*** HAL link meer worden gegenereerd voor deze resource. Zonder verdere parameters krijg je nu in de response dus ***alle attributen*** van *NetworkOperator* terug;
- ✓ Met '*_fields*' kunnen we ***attributen filteren***; ***alleen*** wat is vermeld in '*_fields*' zal in dat geval worden geretourneerd. Dit impliceert dat, op het moment dat je '*_fields*' gaat gebruiken in ***combinatie*** met '*_expand*' ***en*** je wilt de ***geëxpandeerde resource*** terug krijgen, je dus de resource naam zal ***moeten*** opgeven aan '*_fields*':

  ```
  '_fields=MRID,BillingMonthCode,GridOperator&_expand=GridOperator'
  ```



# RESTful API's – Gereserveerde namen

2. ***Pagineren (alleen geldig bij zoeken in resource collecties waar de response een collectie van resources bevat):***

   ✓ *_limit:* Bepaalt het ***maximaal aantal terug te geven*** resources, dit bepaalt de "***pagina grootte***";

   ✓ *_page:* Als '*_limit*' het ***aantal*** resources in een ***enkele response*** definieert (een "***pagina***"), dan is '*_page*' het ***volgnummer*** van die pagina (bij een constante waarde van '*_limit*'). Dus _page = 1 is de ***eerste set*** van '*_limit*' resources en _page = 4 is de 4'e set van '*_limit*' resources;

   ✓ *_from:* Zet de "*virtuele index"* in de collectie ***op de resource*** met gegeven sleutel (MRID);

   ✓ *_before:* Zet de "*virtuele index"* in de collectie op de resource ***direct voorafgaand*** aan de resource met gegeven sleutel (MRID);

   ✓ *_after:* Zet de "*virtuele index"* in de collectie op de resource ***direct volgend*** op de resource met gegeven sleutel (MRID);

   Met de '***_page'*** parameter kan je eenvoudig door ***aaneengesloten lijsten*** met resources bladeren. Als de response lijst dynamisch gedrag vertoont kan je beter gebruik maken van ***resource-gebaseerd*** pagineren met '*_from*', '*_before'* en '*_after*'.

   ***Niet elke collectie zal pagineren ondersteunen!***



# RESTful API's – Gereserveerde namen

3. ***Sorteren (van lijsten resultaten):***

   ✓ *_sort:* Sorteer het resultaat in ***oplopende volgorde*** op basis van het genoemde attribuut (of lijst attributen).

   Plaats een “-” teken voor de attribuutnaam om voor dat specifieke attribuut een ***aflopende volgorde*** te selecteren;

   ‘`_sort=Status.PriorityCode`’ sorteert resultaat in oplopende prioriteit;

   ‘`_sort=-Status.PriorityCode`’ sorteert resultaat in aflopende prioriteit;

   ‘`_sort=-Status.PriorityCode,CreationDate`’ sorteert eerst op aflopende prioriteit en daarbinnen op oplopende creatiedatum;

   ***Niet elke collectie zal sorteren ondersteunen!***



# RESTful API's – Gereserveerde namen

4. ***Vinden (van resultaten die overeenkomen met een specifieke zoek-opdracht):***

   ✓ *_find:* Zoek in de collectie van resources naar die resources die de tekst-string(s) bevatten die voldoen aan de opgegeven zoek expressie;
   De syntax van '*_find*' is afhankelijk van de gebruikte ***zoekmachine,*** b.v. Elasticsearch;

   ***Niet elke collectie zal vinden ondersteunen!***



# RESTful API's – Gereserveerde namen

5. ***Filteren (van resultaten die matchen op gespecificeerde attribuut expressies):***

   ✓ *_filter:* *Geeft alleen die resultaten terug waarvan de attribuutwaarden overeenkomen met de in het filter gespecificeerde waarden*;

   Attributen moeten als ***gekwalificeerde naam*** worden opgegeven en de attributen moeten (uiteraard) onder exact die naam in het berichtmodel te vinden zijn;

   Omdat de filter expressies onderdeel zijn van de parameterstring ***moet*** deze als een ”*URL-encoded*” string zijn opgemaakt.

   Voorbeeld:

   ```
   GET …/v1/netbeheerders?_limit=20?_filter=ActiveSince%5Bgte%5D%3D2018-01-01%26ActiveSince%5Blt%5D%3D2020-01-01
   ```

***Niet elke collectie zal filteren ondersteunen!***



# RESTful API's – Gereserveerde namen

5. ***Standaard links (HAL '_links' sectie):***

   ***De definities van de standaard link namen voor navigatie zijn afhankelijk van de gekozen navigatie vorm;***

   a. ***Navigeren binnen collectie op basis van pagina's:***

   - ✓ '*self*': levert een link naar de ***momentele pagina*** binnen de huidige collectie voor een gegeven '*_limit*';
   - ✓ '*first*': levert de ***eerste pagina*** in de huidige collectie voor een gegeven '*_limit*';
   - ✓ '*last*': levert de ***laatste pagina*** in de huidige collectie voor een gegeven '*_limit*';
   - ✓ '*next*': levert de ***direct opvolgende pagina*** in de huidige collectie voor een gegeven '*_limit*';
   - ✓ '*prev*': levert de ***direct voorgaande pagina*** in de huidige collectie voor een gegeven '*_limit*';
   - ✓ '*item*': levert het ***exacte item*** uit de huidige collectie met de gegeven MRID. Dit is een ***template URI*** waarbij de ***variabele*** `{mrid}` moet worden vervangen door de MRID van het gevraagde item.



# RESTful API's – Gereserveerde namen

***b. Navigeren binnen collectie op basis van MRID:***

- ✓ '*self*': verwijst naar de ***eerste resource in de lijst van resources*** uit het huidige response bericht;
- ✓ '*first*': verwijst naar de ***eerste resource*** in de huidige collectie;
- ✓ '*last*': verwijst naar de ***laatste resource*** in de huidige collectie;
- ✓ '*next*': verwijst naar de ***eerstvolgende resource*** in de huidige collectie;
- ✓ '*prev*': verwijst naar de ***direct voorgaande resource*** in de huidige collectie;
- ✓ '*item*': levert het ***exacte item*** uit de huidige collectie met de gegeven MRID. Dit is een ***template URI*** waarbij de ***variabele*** `{mrid}` moet worden vervangen door de MRID van het gevraagde item;

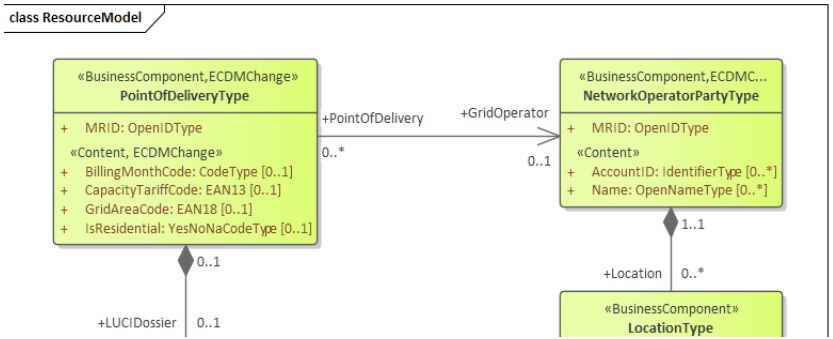


# RESTful API's – Gereserveerde namen

*c. **Links binnen resources:***

- ✓ ‘*self*’: verwijst naar de ***huidige resource*** binnen ***dezelfde*** RESTful API;
- ✓ ‘*base*’: *alleen voor geassocieerde resources*; verwijst naar de RESTful API waar de betreffende resource als ***primaire resource*** is gedefinieerd;
- ✓ ‘*<Rolnaam>*’: wordt binnen een resource gebruikt om de ***relatie*** met een ***geassocieerde resource*** of een ***sub-resource*** aan te geven waarbij de ***associatie rol-naam*** uit het ***resource model*** als link ‘*rel*’ optreedt. Onderstaand voorbeeld geeft aan hoe de *Netbeheerder*’ resource is weergegeven als een link relatie in de ‘*Meetpunt*’ resource:

```
"_links": {
  "GridOperator": {"href": "https://..../v1/meetpunten/871687110001345494/netbeheerders/711000134549 4"}
}
```

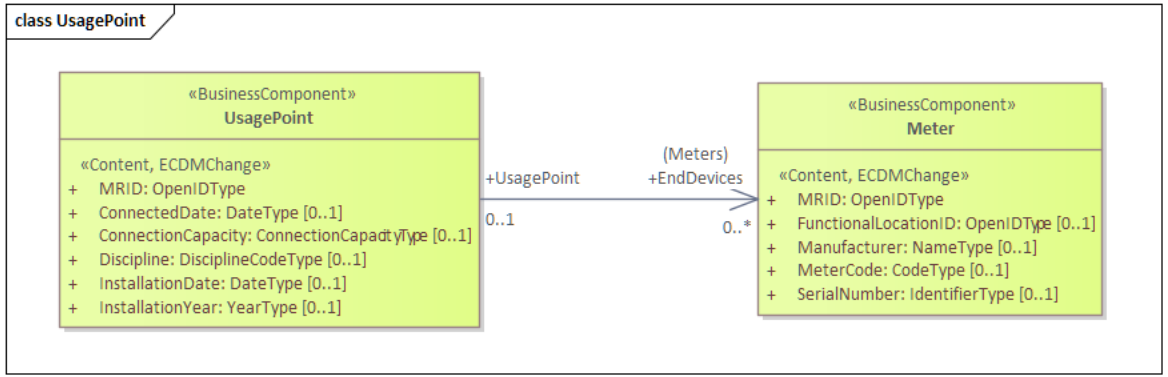


# RESTful API's – Gereserveerde namen

***c. Links binnen resources:***

✓ '*<Rolnamen>*': als de associatie een ***collectie*** representeert (cardinaliteit > 1) dan gebruiken we de meervoudsvorm van de geassocieerde entiteit om tot de naam van de collectie te komen. Onderstaand voorbeeld geeft aan hoe de collectie van '*Meter*' resources is weergegeven als een link relatie in de '*Meetpunt*' resource (de ***rol-naam*** in het model is hier '*EndDevices*' en we hebben een Alias gedefinieerd om de collectienaam in vast te leggen):

```
"_links": {
  "Meters": {"href": "https://...nl/aansluitingen/v1/meetpunten/871687110001345494/meters"}
}
```



# RESTful API's

1. Resource en geassocieerde resource;
2. Sub-resource;
3. Resource collectie;
4. Gegevensgroep;
5. Attribuut;
6. Gereserveerde namen;
7. **Eager- en Lazy loading;**

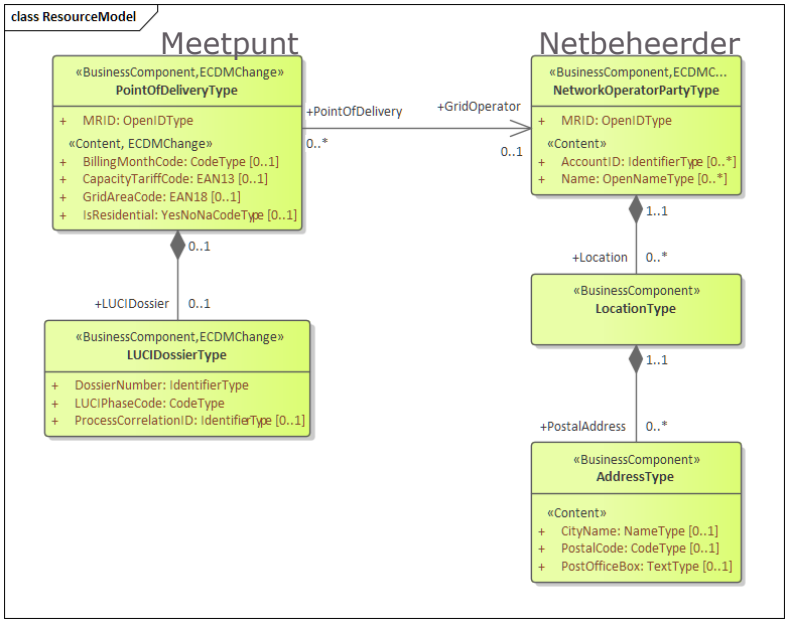


# RESTful API's – Eager- & Lazy Loading

***Voorbeelden van '_fields', '_exclude' en '_expand':***

Gegeven onze "voorbeeld" API met Meetpunt als primaire resource en Netbeheerder als geassocieerde resource dan gelden de volgende scenario's voor een 'GET' operatie:

1. Als geen verdere parameters worden opgegeven dan zal de 'GET' response ***alle attributen en gegevensgroepen*** van Meetpunt teruggeven (mits ze een waarde hebben) en tevens een ***link*** naar Netbeheerder (of een array van links als de cardinaliteit van de associatie groter is dan 1);
2. Als parameter `'_fields=MRID,GridAreaCode'` wordt meegegeven dan zal de 'GET' response ***alleen*** de attributen MRID en GridAreaCode bevatten en ***verder niets***;
3. Als parameter `'_fields=MRID,LUCIDossier'` wordt meegegeven dan zal de 'GET' response ***alleen*** het attribuut MRID bevatten, tevens ***alle attributen*** van de gegevensgroep LUCIDossier en ***verder niets***;



# RESTful API's – Eager- & Lazy Loading

4. Als parameter `'_fields=MRID,LUCIDossier.DossierNumber,LUCIDossier.ProcessCorrelationID'` wordt meegegeven dan zal de 'GET' response ***alleen*** het attribuut MRID bevatten, tevens ***alleen de attributen*** DossierNumber en ProcessCorrelationID uit LUCIDossier en ***verder niets***;
5. Als parameter `'_fields=MRID,GridOperator'` wordt meegegeven dan zal de 'GET' response ***alleen*** het attribuut MRID bevatten, tevens een ***link*** naar Netbeheerder en ***verder niets***;
6. Als parameter `'_expand=GridOperator'` wordt meegegeven dan zal de 'GET' response ***alle attributen en gegevensgroepen*** van Meetpunt teruggeven en tevens een `'_embedded'` sectie met daarbinnen ***alle attributen*** die ***binnen deze API*** voor GridOperator (Netbeheerder) zijn gedefinieerd;
7. Als parameter `'_fields=MRID&_expand=GridOperator'` wordt meegegeven dan zal de 'GET' response ***alleen*** het attribuut MRID bevatten en ***verder niets***. Reden is dat je met '*_expand*' weliswaar aangeeft dat de "***scope***" van de response ***ZOWEL*** Meetpunt ***ALS*** Netbeheerder moet zijn, maar vervolgens selecteer je middels '*_fields*' uit die scope ***uitsluitend*** 'MRID' als response waarde (de '*_expand*' heeft hier dus ***geen*** toegevoegde waarde);
8. Als parameter `'_fields=MRID,GridOperator&_expand=GridOperator'` wordt meegegeven dan zal de 'GET' response ***alleen*** het attribuut MRID bevatten en tevens een `'_embedded'` sectie met daarbinnen ***alle attributen*** die binnen deze API voor GridOperator (Netbeheerder) zijn gedefinieerd en ***verder niets***;



# RESTful API's – Eager- & Lazy Loading

9. De parameter `'_fields=MRID,GridOperator.MRID'` (**zonder** '*_expand*' definitie) dient een error 422 (semantische fout) terug te geven want je kan **alleen** velden uit een **geassocieerde resource** selecteren als je hem (middels '*_expand*') tot onderdeel van de **scope** van de response hebt verklaard. Je mag **wel** `'_fields=MRID,GridOperator'` opgeven (**zonder** '*_expand*' definitie) want daarmee selecteer je expliciet de **link** van Meetpunt naar Netbeheerder (wat feitelijk een **attribuut** is van Netbeheerder), maar dus **niet** de **inhoud** van Netbeheerder;
10. Als parameter `'_fields=MRID,GridOperator.MRID&_expand=GridOperator'` wordt meegegeven dan zal de 'GET' response **alleen** het attribuut MRID bevatten en tevens een `'_embedded'` sectie met daarbinnen **alleen** het attribuut MRID van GridOperator en **verder niets**;
11. Als parameter `'_exclude=MRID,GridOperator'` wordt meegegeven dan zal de 'GET' response **alle attributen en gegevensgroepen** van Meetpunt teruggeven (mits ze een waarde hebben) **met uitzondering van MRID** en ook **geen link** naar Netbeheerder;



# RESTful API's – Eager- & Lazy Loading

- ✓ De parameters '*_fields*' en '*_exclude*' zijn "**mutual exclusive**", je gebruikt de een of de ander maar **nooit beiden** (dat laatste dient een error 422 (semantische fout) op te leveren;
- ✓ Ook conflicterende combinaties zoals `'_exclude=GridOperator&expand=GridOperator'` zijn **illegaal** en dienen een error 422 op te leveren;
- ✓ Voor **verschillende** resources mag je ze wel combineren: `'_exclude=resourceA&_expand=resourceB'` is valide;



# RESTful API's – Eager- & Lazy Loading

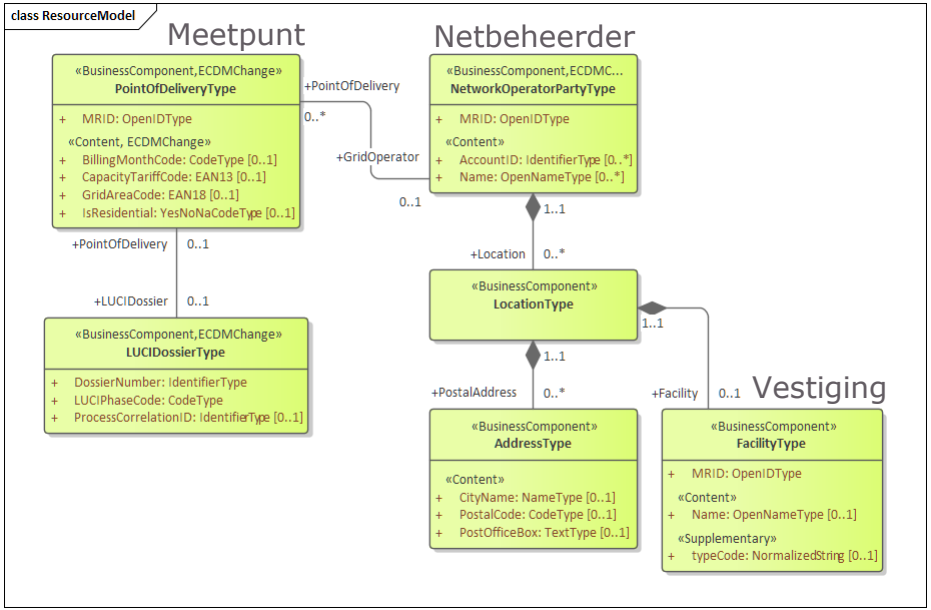
## Geneste resources:

Gegeven het volgende model waarbij **primaire resource** Meetpunt (PointOfDelivery) een **geassocieerde resource** Netbeheerder (NetworkOperatorParty) heeft die op zijn beurt (via gegevensgroep Location) een **geassocieerde resource** Vestiging (Facility) bezit:

De volgende scenario's gelden voor een 'GET' operatie:

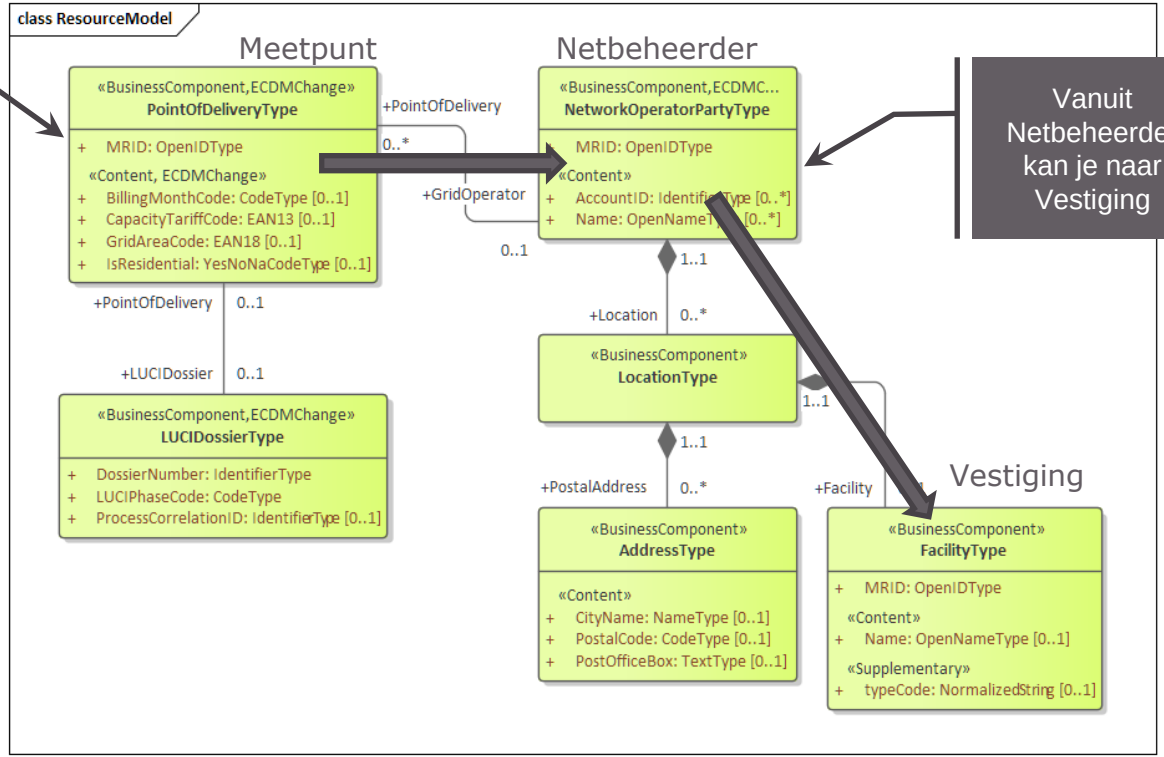
1. Als geen verdere parameters worden opgegeven dan zal de 'GET' response **alle attributen en gegevensgroepen** van Meetpunt teruggeven (mits ze een waarde hebben) en tevens een **link** naar Netbeheerder ***en verder niets*** (Vestiging is dus ***niet*** direct zichtbaar);

   De reden is dat *Vestiging* een *geassocieerde resource* is van *Netbeheerder* en dus vanuit Meetpunt ***onzichtbaar*** is. ***Regel is dat de output te allen tijde de structuur van het resource model dient te volgen!***



# RESTful API's – Eager- & Lazy Loading

Vanuit Meetpunt kan je naar Netbeheerder maar niet direct naar Vestiging




Vanuit Netbeheerder kan je naar Vestiging

# RESTful API's – Eager- & Lazy Loading

2. Als parameter "`_fields=MRID,GridOperator`" wordt meegegeven dan zal de 'GET' response **alleen** het attribuut MRID bevatten en tevens een **link** naar Netbeheerder en **verder niets**;
3. Als parameter "`_fields=MRID,GridOperator&_expand=GridOperator`" wordt meegegeven dan zal de 'GET' response **alleen** het attribuut MRID bevatten en tevens een "`_embedded`" sectie met daarbinnen **alle attributen en gegevensgroepen** die binnen deze API **voor GridOperator** (Netbeheerder) zijn gedefinieerd. Als onderdeel hiervan bevat Netbeheerder een **link** naar Vestiging;
4. Als parameter "`_expand=GridOperator.Location.Facility`" wordt meegegeven dan zal de 'GET' response **alle attributen en gegevensgroepen** van Meetpunt teruggeven (mits ze een waarde hebben) en **tevens** een "`_embedded`" sectie met daarbinnen GridOperator (Netbeheerder). Op zijn beurt heeft GridOperator ***alleen*** het attribuut Location met daarbinnen wederom een "`_embedded`" sectie met daar weer binnen **alle attributen en gegevensgroepen** die **binnen deze API** voor Facility (Vestiging) zijn gedefinieerd;

   Dus om een **geneste resource** te kunnen expanderen moet het **complete pad** naar deze resource worden opgegeven. Van de ***tussenliggende*** resources worden ***alleen*** die attributen getoond die expliciet middels '*_fields*' zijn opgevraagd, ***of*** die als '*houder*' van de te expanderen resource zijn gedefinieerd (zoals *Location* in bovengenoemd voorbeeld);
5. De parameter "`_expand=GridOperator.Location`" is **illegaal** omdat 'Location' geen resource maar een gegevensgroep representeert. Dit dient dan ook een error 422 (semantische fout) op te leveren;



# RESTful API's – Eager- & Lazy Loading

Geneste resources die “aftakken” vanuit een **gegevensgroep** (zoals in ons voorbeeld *GridOperator/Location/Facility*) worden voor wat betreft de HAL structuur geacht onderdeel te zijn van de ***eerstvolgende bovenliggende resource*** (in ons voorbeeld *GridOperator*). Dit omdat gegevensgroepen geen ‘*_links*’ en “*_embedded*” structuren ondersteunen, die zijn alleen voor de resources zelf! Om de **context** van de associaties in stand te houden en **name-clashes te voorkomen** bestaat de naam van de link uit het **volledige pad** (relatief t.o.v. de bovenliggende resource) naar de geassocieerde resource:

```
"_embedded": {
    "GridOperator": {
        "MRID": "7110001345494",
        "AccountID": "341",
        "Name": "Enexis",
        "Location": {
            "PostalAddress": {
                "CityName": "Den Bosch",
                "PostalCode": "1234AA",
                "PostOfficeBox": "23"
            }
        },
        "_links": {
            "self": {"href": "https://api...luitingen/v1/meetpunten/871687110001345494/netbeheerders/7110001345494"},
            "Location.Facility": {"href": "https://api...0001345494/netbeheerders/7110001345494/vestigingen/1200"}
        }
    }
}
```

# RESTful API's – Eager- & Lazy Loading

Binnen een link ***kan*** gebruik worden gemaakt van extra properties om de betekenis van de link te verduidelijken:

- ✓ '*name*' = De naam van het **type** van het door de link geadresseerde **object.** Dit is met name van belang als de link verschillende gespecialiseerde typen van een algemeen base-type kan bevatten. Voorbeeld: base-type = "*Meter*", gespecialiseerde typen = "*SmartMeter*" of "*ConventionalMeter*". Als het type eenduidig is gebruiken we deze property i.h.a. niet;
- ✓ '*title*' = Een beschrijvende naam die typisch is bedoeld om de geassocieerde resource aan eindgebruikers te tonen op een gebruikersvriendelijke manier;
- ✓ '*id*' = De MRID van het object dat door de link wordt gerepresenteerd. In het algemeen is dit gelijk aan het laatste veld in de 'href';

```
"_embedded": {
    "GridOperator": {
         "MRID": "7110001345494",
         "AccountID": "341"
         },
         "_links": {
             "self": {"href": "https://api...luitingen/v1/meetpunten/871687110001345494/netbeheerders/7110001345494"},
             "Location.Facility": {"href": "https://api...1345494/netbeheerders/7110001345494/vestigingen/1200",
                                    "name": "FacilityType",
                                    "title": "Vestiging Zuid",
                                    "id": "1200"}
         }
    }
}
```



# RESTful API's – Eager- & Lazy Loading

## Collecties opvragen:

Gegeven een **Meter resource** met daaronder de **Register sub-resource**. Een meter kent meerdere registers maar het aantal kan variëren. Omdat registers een **collectie** is, kunnen we hem bevragen en krijgen we een lijst van de inhoud in de vorm van "*items*". Omdat '*registers*' een ***impliciete collectie*** is krijgen we per definitie deze lijst als een geëxpandeerde lijst van meter resources terug (impliciete collecties worden immers per definitie al geëxpandeerd). We vragen hier om maximaal 2 resultaten:

```
GET https://api.enexis.nl/meters/v1/slimme-meters/E000900000149/registers&_limit=2
```

Het resultaat van deze operatie is getoond op de volgende slide.



# RESTful API's – Eager- & Lazy Loading


```
{
    “TotalCount”: 4,
    “RemainingCount”: 2,
    “ReturnedCount”: 2,
    “_embedded”: {
        ”Items”: [{
            "MRID": "1.8.1",
            "MeasurementUnitCode": "KWH",
            "MeasurementDirectionCode": "LVR",
            "Multiplier": "1.0",
            "_links": {
               "self": {"href": "https:/api.enexi.../slimme-meters/E000900000149/registers/1.8.1”}
            }
        },
        {
            "MRID": "1.8.2",
            "MeasurementUnitCode": "KWH",
            "MeasurementDirectionCode": "LVR",
            "Multiplier": "1.0",
            "_links": {
               "self": {"href": "https:/api.enexi.../slimme-meters/E000900000149/registers/1.8.2”}
            }
        }]
    },
    “_links”: {
        “self”: {“href”: “https://api.enexi.../slimme-meters/E000900000149/registers?_page=1&_limit=2”},
        “next”: {“href”: “https://api.enexi.../slimme-meters/E000900000149/registers?_page=2&_limit=2”},
        “last”: {“href”: “https://api.enexi.../slimme-meters/E000900000149/registers?_page=2&_limit=2”}
    }
}
```


SAMEN WERKEN WE AAN EEN
BETROUWBARE EN DUURZAME
ENERGIEVOORZIENING
VOOR VANDAAG EN VOOR DE
TOEKOMST.
For more information, please contact:
CST-Integratie@enexis.nl
ENEXIS
GROEP