Memo

Aan
Leden DD-roosterdata-groep
cc: Leden DD-werkgroep

Datum	**Kenmerk**	**Aantal pagina's**
4 augustus 2020	11205913-DSC-18-v0.9	5

Van	**Doorkiesnummer**	**E-mail**
Stef Hummel	+31(0)6 1019 8112	Stef.Hummel@deltares.nl

Onderwerp
Bespreekverslag web conferences DD roosterdata 17 juni, 8 juli en 29 juli j.l.

Dit verslag bevat een kort overzicht van de laatste drie sessies over de DD-GRID-API:

- 17 juni
- 8 juli
- 29 juli

Deelnemers:

Bart Thonus	HKV
Casper van der Wel	Nelen & Schuurmans
Gerrit Hendriksen	Deltares
Jeroen Gerrits	VORtech
Koos Boersma	IHW
Sander Loos	Hydrologic
Stef Hummel	Deltares

17 juni

Ter bespreking liggen twee documenten:

- Door Casper opgestelde vergelijking *DD-GRID-API vs OGC API coverages*
- Door Stef n.a.v. de bespreking van 27 mei aangepaste versie (v2) van het voorstel voor de DD-GRID-API

Bij de bespreking van het vergelijkingsdocument blijkt dat de DD-GRID-API en de OGC API coverages sterk op elkaar lijken. Met een paar kleine wijzigingen valt DD-GRID-API qua structuur op de OGC API coverages aan te sluiten. De terminologie is volledig anders.

Besluit:
We gaan niet verder met het ontwikkelen van een eigen lijn, maar kijken of we met de OGC API Coverages (https://github.com/opengeospatial/ogc_api_coverages). Daar waar we er niet mee toe kunnen zouden we dan wijzigingen of uitbreidingen aan de ogc-versie moeten aanbrengen.

Koos vraagt bij GeoNovum na of men bij de ontwikkeling van de OGC API Coverages betrokken is.
Versie 2 van de voorgestelde DD-GRID-API is dus niet meer besproken.

8 juli

Op basis van de OGC API Coverages hebben Casper, Jeroen en Stef een simpel voorbeeld uitgewerkt, dat besproken wordt.

Op Koos' vraag aan Geonovum is als antwoord gekomen dat men wel sterk bij OGC ontwikkelingen is betrokken, maar niet direct bij deze API. Wel kent men de status, en die is als volgt:

De stand van zaken rondom de OGC API Coverages is enigszins controversieel te noemen. Deze standaard is nog in ontwikkeling en is onlangs door de OGC Architecture Board terug naar de tekentafel gestuurd omdat de RESTful principes niet goed werden toegepast en er teveel werd afgeweken van de werkwijze van de OGC API Features (die al goedgekeurd is als standaard). Trekker van dit werk is Peter Baumann, ik kan jullie in contact brengen als je dat wilt. Echter zou ik adviseren om nog even af te wachten wat er uit de volgende iteratie komt (planning hiervan weet ik niet).

Alternatieven:

1) *Je kan gaan voor de 'oude' generatie Coverage services, dan heb ik het over de Web Coverage Service (WCS). Naar ik begrijp niet gemakkelijk te implementeren en in gebruik, maar wel degelijk. Contactpersoon daarvoor is ook Peter Baumann.*
2) *Je zou een eigen REST API kunnen bouwen naar het voorbeeld van de OGC API Features (waarschijnlijk kan je die bijna helemaal as is toepassen, misschien heb je nog wat extra's nodig voor coverages dat daar niet in zit). Je kan dan de data serveren in bijvoorbeeld het formaat CoverageJSON. Dat is een moderner meer lichtgewicht formaat om coverages uit te wisselen. Met de maker van CoverageJSON kan ik je ook in contact brengen, Jon Blower is dat. Dit is echter nog geen standaard, binnen een paar jaar zou dit wel kunnen gebeuren. Clemens Portele, de editor van de OGC API Features, zou je goed kunnen helpen bij zulke experimenten.*

Besluit:
Naar aanleiding van dit antwoord gaan we niet alleen inzetten op de OGC API Coverages, maar gaan we ook kijken naar CoverageJson.

27 juli

Casper, Bart, Jeroen en Stef (Sander was verhinderd) bespreken het door Jeroen en Stef opgestelde voorbeeld van waterbeweging-gerelateerde data op een rechthoekig grid. Deze is nu in zowel *OGC API Coverages* als in *CoverageJson* uitgewerkt.

Voornaamste conclusies:

- Voor *OGC API Coverages* is er een json schema-definitie, wat heel prettig werkt. Voor *CoverageJson* is dat er niet, waardoor alle mogelijke opties uit de voorbeelden moeten worden gehaald. (Daar staat overigens tegenover dat er dientengevolge veel voorbeelden zijn; voor *OGC API Coverages* is dat minder het geval.)
- *CoverageJson* is compacter, en is daardoor voor de beschrijving van de assen overzichtelijker.
- Voor beschrijving van de parameters geldt dit echter niet. Dat komt in *CoverageJson* wat rommelig en onoverzichtelijk over. Het kunnen kiezen in welke taal men de parameter en de unit beschrijft lijkt leuk, maar geeft ook onnodige vrijheid en verwarring. Ook de terminologie bij de eenheid

("label"/"symbol.type/value/observedProperty" is minder eenduidig dan bij de *OGC API Coverages* (UnitOfMeasement).
- Belangrijke beperkingis zit hem in de data zelf, de "ranges". Men kent alleen het inline uitschrijven van de waarden, per parameter; er kan dus niet worden gerefereerd naar een file met daarin de waarden van een of meerdere parameters. (We hebben zelf daarom het attribute "fileReference" toegevoegd, vergelijkbaar met *OGC API Coverages*).
- *CoverageJson* is onaf, en er is weinig gaande m.b.t. het verder uitwerken ervan. Meest recente activiteit is die in https://github.com/opengeospatial/CoverageJSON (december 2018), maar binnen opengeospatials is men vooral bezig met de *OGC API Coverages* https://github.com/opengeospatial/ogc_api_coverages/commits/master.

Besluit:
Al uitwerkend was al, en al besprekend wordt ook, geconcludeerd dat we niet verdergaan met *coverageJSON*.
De *OGC API Coverages* variant staat ondertussen op github:
https://github.com/DigitaleDeltaOrg/dd-grid-api/blob/master/voorbeelden/dd-grid-regular-ogc-api-coverage.json
Om verwarring te voorkomen staat de afgevallen *coverageJSON*-versie er niet. Voor de volledigheid is die opgenomen in de bijlage van het voorliggende verslag.

Als gevolg van deze keuze is het curvilineaire grid alleen uitgewerkt in *OGC API Coverages*.
Het resultaat hiervan staat op github:
https://github.com/DigitaleDeltaOrg/dd-grid-api/blob/master/voorbeelden/dd-grid-displacement-ogc-api-coverage.json

Enkele opmerkingen hierover:
- De envelope is hier qua structuur (vanzelfsprekend) exact hetzelfde als bij het raster.
- Bij de assen wordt een splitsing gemaakt tussen de reguliere assen (tijd en realisatie) en de 'vervormde' assen (x en y).
- Bij x en y verwacht je vervolgens voor beide een twee-dimensionale array van getallen, maar dat blijkt volgens het OGC API Coverages json-schema ééndimensionaal te moeten zijn (en aanwezig te moeten zijn).
  Dat zou nog kunnen werken als ergens te achterhalen is dat deze eendimensionale array een platgeslagen nX*nY array is, maar die dimensies staan nergens opgegeven.

Het laatste punt konden we dus niet goed oplossen, temeer daar we nergens een uitgewerkt voorbeeld konden vinden. We hebben vervolgens besloten om de coordinates array te verwijderen. We hebben dus het OGC-schema aangepast, zei het zeer gering: het niet langer verplicht stellen van "coordinates".
Het aangepaste schema staat op github:
https://github.com/DigitaleDeltaOrg/dd-grid-api/blob/master/voorbeelden/ogc-schema-dd.json
(Het originele OGC schema staat in:
https://github.com/opengeospatial/ogc_api_coverages/blob/master/standard/openapi/coverage-schema.json)

Vervolgacties:
- Casper maakt de voorbeelden wat compacter door json dictionaries toe te passen
- Jeroen en Stef stellen een concept op voor de end points
- Volgende bespreking op donderdag 3 september.

Bijlage: CoverageJSON-uitwerking

(Deze aanpak is vervallen.)

```
{
  "type" : "Coverage",
  "domain" : {
    "type": "Domain",
    "domainType": "Grid",
    "axes": {
      "x": { "start": 269024, "stop": 244024, "num": 100 },
      "y": { "start": 617934, "stop": 625134, "num": 72 },
      "t": { "start": "2020-02-26T00:00:00Z", "stop": "2020-03-04T00:00:00Z",
"num": 169},
      "realization": {"start": 0, "stop": 50, "num": 51}
    },
    "referencing": [{
      "coordinates": ["x","y"],
      "system": {
        "type": "RD",
        "id": "https://epsg.org/28992"
      }
    }, {
      "coordinates": ["t"],
      "system": {
        "type": "TemporalRS",
        "calendar": "Gregorian"
      }
    }]
  },
  "parameters" : [
    {
      "WATHTE": {
        "type" : "Parameter",
        "description" : {
          "en": "Waterlevel"
        },
        "unit" : {
          "label": {
            "en": "m"
          },
          "symbol": {
            "value": "m",
            "type": "meters above NAP"
          }
        },
        "observedProperty" : {
          "id" : "http://<path>/parameters/WATHTE",
          "label" : {
            "en": "Waterlevel"
          }
        }
      }
    },
```

```
    {
      "T": {
        "type" : "Parameter",
        "description" : {
          "en": "Temperature"
        },
        "unit" : {
          "label": {
            "en": "Celsius"
          },
          "symbol": {
            "value": "Celsius",
            "type": ""
          }
        },
        "observedProperty" : {
          "id" : "http://<path>/parameters/TEMP",
          "label" : {
            "en": "Temperature"
          }
        }
      }
    },
    {
      "SALNTT": {
        "type" : "Parameter",
        "description" : {
          "en": "Salinity"
        },
        "unit" : {
          "label": {
            "en": "-"
          },
          "symbol": {
            "value": "-",
            "type": ""
          }
        },
        "observedProperty" : {
          "id" : "http://<path>/parameters/SALT",
          "label" : {
            "en": "Salinity"
          }
        }
      }
    }
  ],
  "ranges" : {
    "fileReference" : "https://<path>/EemsDollard/20200226000000.nc"
  }
}
```