Memo

Aan
Bart Thonus, Casper van der Wel, Erik Plaggenmars, Jeroen Gerrits, Sander Loos, Tom Bogaard.

Datum	**Kenmerk**	**Aantal pagina's**
20 december 2019	11203680-DSC-28-v0.5	6
Van	**Doorkiesnummer**	**E-mail**
Stef Hummel	+31(0)6 1019 8112	Stef.Hummel@deltares.nl

Onderwerp
Uitwerking Digitale Delta API voor roosterdata (DD-GRID)

Inleiding

Dit memo bevat een beschrijving van de conceptversie van DD-GRID, de Digitale Delta API voor roosterdata (oftewel grid data). Deze conceptversie is het resultaat van een achttal vergaderingen met de roosterdata-groep, een subgroep van de DD-werkgroep. Het eerste deel van de sessies richtten zich op het in kaar brengen van de gewenste functionaliteit, het tweede deel op het uitwerken van het DD-GRID-voorstel.

De opbouw van het memo is als volgt:

- Het eerste hoofdstuk beschrijft de tijdens het uitwerken gemaakte afspraken;
- Het tweede hoofdstuk geeft enkele onderwerpen aan waarover nog een keuze over gemaakt moet worden;
- Het derde hoofdstuk bevat de conceptversie van de end points van de DD-GRID-API.
- Het vierde hoofdstuk bevat en nog enigszins summiere conceptversie van de response van de end points van de DD-GRID-API.
  De inhoud van de resource objects in de response is afgeleid uit:
  - Vergelijkbare resource objects in de DD-API
  - De overzichtstabel met functionele eisen (FunctioneleEisenRoosterdataDdAPI-v0.95.xlsx)

  Zodra deze conceptversies redelijk definitief zijn zal e.e.a. worden omgezet in een OAS3-specificatie, die we op github gaan zetten.

Gemaakte afspraken

Tijdens de bespreking van de diverse conceptversies zijn de volgende algemene afspraken gemaakt:

- Bij de DD-API gebruiken we paging. Bij de DD-GRID api stellen we dat niet verplicht. Het mag echter wel worden geïmplementeerd. Nelen & Schuurmans b.v. zal dat wel doen, omdat ze orde 10.000 rastersets hebben.
- I.v.m. de consistentie voegen we, net als bij de DD-API, aan de metadata response de ‘node’ informatie toe (en we voegen nodeId toe als query parameter)
- De boundingBox query parameter kan bij vrijwel alle end points worden meegeven, t.b.v. ruimtelijke selectie
  In de response zitten de grids die geheel *of gedeeltelijk* binnen de boundingBox vallen.
- M.b.t. de naming convention van de endpoints en de query-parameters baseren we ons op de voorbeelden van API strategie voor de Nederlandse overheid: parameters camelCase, en end points zonder hoofdletters. Bij complexe end point namen (maar

die hebben we momenteel nog niet) scheiden we de delen d.m.v. een '-', zulks conform de naming conventions van https://restfulapi.net/.

- Overwogen is om /griddatasets te mengen met de /timeseries van de DD-API. Besloten is echter om de roosterdata er helemaal naast te zetten, want:
  - Voor een gebruiker gaat het om andere functionaliteit dan tijdseries
  - Niet alle systemen leveren de mix van zowel tijdseries als roosterdata.

  Wel is er overlap met de DD-API:
  - Provider info en error handling wordt conform de DD-API gedaan
  - Als een databron paging implementeert wordt dat gedaan conform de DD-API
  - Als van uit grid-data een tijdserie op één roosterpunt wordt opgevraagd is de respons gelijk aan die van het DD-API */timeseries* end point.
- Tijdens het opstellen van de conceptversie van de end points rees de vraag of we aparte */sources* en */grids* end points nodig hebben, of dat combineren handiger is, omdat het grotendeels overlap (en bij b.v. Matroos het effectief hetzelfde is).
  Na enig nadenken is besloten om – in elk geval voorlopig – alleen */grids* te kennen. Onderdeel van de respons is dan de source.
  Meerdere grids voor een zelfde model kan dan betekenen:
  - dat er een grof en een fijn rooster is
  - dat het totale grid is opgedeeld in subdomeinen.

  We bekijken in de loop van de tijd want nodig is, mede aan de hand van de info in de */grid*-response en de vraag waar je op wilt kunnen filteren.
- Er is overwogen om, naast het kunnen specificeren van de gewenste projectie bij het opvragen van data, ook te kunnen specificeren in welk crs de bounding-box bij de query is opgegeven. Dit leidde voor een eerste versie tot teveel complexiteit. Besloten is dus om dat te laten vallen en het volgende af te spreken:
  - Bij discovery functies wordt de bounding box altijd in wgs84 uitgedrukt
  - Bij het opvragen van data wordt aangegeven in welke projectie de data moet worden geleverd, en wordt de bounding box waarbinnen data geleverd moet worden uitgedrukt in diezelfde projectie.

Openstaande punten

Ongetwijfeld komen er bij het in detail in OAS3 uitwerken van de specificaties nog diverse te bespreken punten naar voren. Bij dit eerste voorstel echter staan de volgende punten open:

- Is de naamgeving van de end points inhoudelijk goed? (griddatasets, griddata, etc.?)
- Hoe noemen we de API? (Hangt waarschijnlijk samen met voorgaande vraag.)
  Voorlopig noemen we hem DD-GRID.

Conceptversie DD-GRID end points

End point	Parameters	Beschrijving
/projections		Welke projecties ondersteunt het systeem? Response: lijst van projecties waar bij het opvragen van data naar toe kan worden getransformeerd
/grids		Welke grids kent de provider? Response: lijst met metadata-objecten van grids (kan mengeling van rasters en curvilineair zijn)
	boundingBox	Gebied van interesse
	observationTypeId	observationType identifier
	quantity	quantity (attribuut van observation type)
	parameterCode	parameter (attribuut van observation type)
/grids/{gridId}		metadata van een grid
/observationtypes		Welke observation types kent de provider? Response: lijst met metadata-objecten van observationTypes
Opm.: filters bedoeld om apart te gebruiken (combinatie kan niet-bestaand zijn)	***boundingBox***	Gebied van interesse
	gridId	gridId
/observationtypes/{observationtypeId}		metadata van een observationTypes
/griddatasets		Welke datasets kent de provider? Response: lijst met daadwerkelijk beschikbare data (telkens een combinatie grid en observationType, of grid en quantity)
Opm.: filter-combinatie kan niet-bestaand zijn	***gridId***	gridId
	observationTypeId	observationTypeId
	quantity	quantity (attribuut van observation type)
	parameterCode	parameter (attribuut van observation type)
	startTime	'data vanaf (inclusief)'
	endTime	'data tot en met'
	analysisTime	Productietijd van de data set

/griddatasets/{gridDataSetId}	metadata van een dataset (grid en observationType)

/dataformats	Ondersteunde data formats. Minimaal “netcdf-cf”

/griddata		Opvragen van data van een of meer datasets Leiden tot één of meer variabelen in de netcdf-cf file
Opm.: filter-combinatie van gridId en observationTypeId of quantityId kan niet-bestaand zijn	***dataSetId[]****)	dataSetId(s) óf gridId + observationTypeId(s) óf gridId + quantitie(s)
	gridId observationTypeId[]*)	
	gridId quantity[]*)	
	parameterCode	parameter (attribuut van observation type)
	startTime	‘data vanaf (inclusief)’
	endTime	‘data tot en met’
	analysisTime	Productietijd van de data set
	realization	Realization index
	point[]*)	X,Y-punt(en) Response is dan een json file met een lijst van tijdseries conform de response van de DD-API. (De lijst is 1 lang als er om 1 grootheid op 1 punt is gevraagd.)
	projection	Gewenste projectie. (Tevens projectie van de areaOfInterest.)
	areaOfInterest	WKT-string die de gewenste uitsnede beschrijft

*) Voorstel voor syntax bij een ‘array’: variabelen scheiden door ‘;’.B.v.:
/griddata?dataSetId=dscm_waterlevel;dscm_bedlevel&startTime=…
(is inhoudelijk hetzelfde als:
/griddata?gridId=dscm&dataSetId=waterlevel;bedlevel&startTime=…
)

Conceptversie DD-GRID response

Response */projections*

string[] array (lijst met EPSG projectie codes - of moet dit uitgebreider?)

Response */grids*

Grid[] array

Grid:

	id:	*string* (uniek id binnen databron)
	type:	*enum: raster/curvilinear/unstructured*
	extent:	**Extent** (conform DD-API, echter zonder realizationCount, zie
	source:	**Source** **GridDataSet**)
	raster:	**RasterGrid**
	curvilinear:	**CurvilinearGrid**
	unstructured:	**UnstructuredGrid**

Extent: […,…,…,…] (lowerLeftX, lowerLeftY, upperRightX, upperRightY)

RasterGrid:

xLLCorner
yLLCorner
nCols
nRows
colSize
rowsize
rotation

CurvilinearGrid:

mMax	(aantal roostercellen in M richting)
nMax	(aantal roostercellen in N richting)
kMax	(aantal lagen)

UnstructuredGrid:

numNodes	(aantal roosterpunten)
numEdges	(aantal links tussen roosterpunten)
numFaces	(aantal roostercellen)

Response */observationtypes*

ObservationType[]
array
ObservationType: (conform DD-API)

Response */griddatasets*

GridDataSet[] array
GridDataSet:

id:	*string*	(uniek id binnen databron)	
observationType:	**ObservationType**	(conform DD-API)	
startTime:	*DateTime*		
endTime:	*DateTime*		
analysisTime:	*DateTime*	(optional)	
realizationCount:		(optional; >1 betekent ensemble run)	

Response */dataformats*

string[] array (lijst met ondersteunde formaten - moet in elk geval "netcdf-cf" bevatten)

Response */griddata*

netcdf-cf file met één grid (raster, curvilear, unstructured)
formele specs en voorbeelden volgen