Using CSW

Han Oostdijk

2018-11-19

Contents

Introduction: Catalog Services for the Web (CSW)

I ran into the Catalog Services for the Web (CSW) when I wanted to see which public geographical data was available for the Netherlands: the Nationaal GeoRegister (NGR) makes use of it. Apart from showing available datasets, the page also gives information about the underlying technology: CSW.

On the Catalogue Service page of the Open Geospatial Consortium (OGC) Standards page the full Specification for the latest version (2.0.2) can be found. See the References section for more links.

The Catalogue Service page describes:

Catalogue services support the ability to publish and search collections of descriptive information (metadata) for data, services, and related information objects. Metadata in catalogues represent resource characteristics that can be queried and presented for evaluation and further processing by both humans and software. Catalogue services are required to support the discovery and binding to registered information resources within an information community.

R package CSW

One of the ways to get information about and from a CSW catalog is the GET method of the HTTP protocol. By specifying a properly formed URL in a internet browser the requested information is shown in the browser window.

The `CSW` R package provides functions that use the same GET method to place the result in an R object as a data.frame, xml_document or integer variable.
`Operations` are recognized by the CSW interface in the URL by the phrase `request=`. The package has functions for the **read-only** `Operations`. Also included are some utility functions.

This package is until now only tested on the Nationaal GeoRegister(NGR) catalog with CSW 2.0.2. By using the `CSW_set_url` and `CSW_set_version` functions these values will be replaced by the ones provided by the user.

Let me know if you encounter problems with other catalogs: maybe these can be solved.

Utily functions

As stated in the Introduction the package is developped and tested for CSW version 2.0.2 for the the catalog of Nationaal GeoRegister(NGR) but other catalogs and versions can be used. To see which catalog or version is active, use a `get` utility function:

```r
library(CSW)
CSW_get_version()
#> [1] "2.0.2"
CSW_get_url()
#> [1] "http://nationaalgeoregister.nl/geonetwork/srv/dut/csw?"
```

To use another catalog or version use the corresponding `set` function. NB. this version or catalog will then be active; that is used during the remainder of the session until another use of the function. E.g.

```r
CSW_set_url("http://nationaalgeoregister.nl/geonetwork/srv/dut/inspire?")
CSW_get_url()
#> [1] "http://nationaalgeoregister.nl/geonetwork/srv/dut/inspire?"
```

The other utility function is `CSW_display_node`. Most CSW functions of the package have an option to produce an `xml` object as output. The standard `print` method can be also be used but truncates the output.

CSW functions

The functions in the `CSW` package that are related to interface operations are all prefixed with `CSW_`. Further they have in common the following three arguments:

- version : the default is given by `CSW_get_version()`. See previous section.

- baseurl : the default is given by `CSW_get_url()`. Also see previous section.
- verbose : FALSE (default), TRUE or F that indicates if the generated GET URL should be shown and if so then how. FALSE means do not show, TRUE means show only the (decoded) variable part of the request and F means show the full URL.

A question that could arise is: "what operations does the CSW interface have?" But I think that the first question a starting user of the CSW interface probably will have is: "which contents has this catalog?". Therefore I will describe briefly all CSW functions, but I will start with the `CSW_GetRecords` function.

The CSW_Get* functions

The easiest way to get the contents of the catalog in an R variable is to use the `CSW_GetRecords` function without specifying any argument. The formal arguments of the function will then be set to (the first of) the default value(s). This includes the `constraint` argument with as default value the empty string (i.e. no restriction on the records that are returned). So let us comment out the `CSW_GetRecords` statement and use the `CSW_GetHits` function to find out how many records we would have received.

```
#df = CSW_GetRecords()
( nh = CSW_GetHits(verbose='F') )
#> GetRecords request:
#> http://nationaalgeoregister.nl/geonetwork/srv/dut/inspire?service=CSW&ve
#> rsion=2.0.2&request=GetRecords&resultType=hits&maxRecords=1&startPositio
#> n=1&constraintLanguage=CQL_TEXT&constraint_language_version=1.1.0&typeNa
#> mes=csw%3ARecord
#> [1] 474
```

We specified `verbose='F'` to see which URL was generated and because `constraint` was not specified it was set to the empty string. So we see:

- the URL is still set to http://nationaalgeoregister.nl/geonetwork/srv/dut/inspire? as done in Utily functions
- this catalog contains in total 474 records.

For demonstration purposes we will better apply a constraint (see the section Queries for more about queries in CSW) and being Dutch I want to see all entries with a phrase starting with 'water' :

```
( nw = CSW_GetHits(constraint="Title LIKE 'water%' ") )
#> [1] 37
```

This constraint would lead to 37 records.

summary report in `data.frame`

To actually retrieve these records we use the `CSW_GetRecords` function:

```
df1 = CSW_GetRecords(constraint="Title LIKE 'water%'")
str(df1,strict.width='cut')
#> 'data.frame':	10 obs. of  8 variables:
#>  $ abstract   : chr  "De Europese Kaderrichtlijn Water (KRW) is in 2000 v"..
#>  $ format     : chr  NA "" NA NA ...
#>  $ identifier: chr  "6b8c0c45-09a5-4e8b-b30a-b771f7262a61" "f8d840bc-18d"..
#>  $ modified  : chr  "2018-09-10" NA "2016-07-12" "2017-07-19" ...
#>  $ relation  : chr  NA "" NA NA ...
#>  $ subject   : chr  "Hydrografie, gebiedsbeheer, gebieden waar beperking"..
#>  $ title     : chr  "Kaderrichtlijn Water RWS WMS" "Emissies naar lucht "..
#>  $ type      : chr  "service" "dataset" "service" "service" ...
```

We notice here:

- the result `df` is a data.frame because `table` is the default value for the `output` argument. Other possible values are `list` and `xml`.
- the dimensions of `df` are 10 and 8. The number of rows is 10 and not 37 because the argument `maxRecords` has the default value `10`. The number of columns is 8 because the argument `ElementSetName` has the default value `summary`. Other values are `full` (that gives more) and `brief` (that gives less) columns as output.

`full` report in `data.frame` within `list`

The following code shows the `list` output with the maximum number of columns: `ElementSetName = 'full'`. Notice that because of `output='list'` the result now includes the number of total and retrieved records. The latter now being 15 because we set `maxRecords=15`. In the previous code block we retrieved the first 10 records because the default value for `startPosition` is 1. By setting `startPosition = 11` in combination with `maxRecords = 15` we now retrieve the records 11 up to 25.

```
lst1 = CSW_GetRecords(constraint="Title LIKE 'water%'",
    ElementSetName = 'full', maxRecords=15, startPosition=11,output='list')
```

```
str(lst1,strict.width='cut')
#> List of 3
#>  $ nrm: num 37
#>  $ nrr: num 15
#>  $ df :'data.frame': 15 obs. of  14 variables:
#>   ..$ abstract   : chr [1:15] "Deze view service is gebaseerd op data va"..
#>   ..$ BoundingBox: chr [1:15] NA NA NA NA ...
#>   ..$ date       : chr [1:15] "2016-07-12" "2018-11-15" "2018-07-30" "20"..
#>   ..$ description: chr [1:15] "Deze view service is gebaseerd op data va"..
#>   ..$ format     : chr [1:15] NA NA NA NA ...
#>   ..$ identifier : chr [1:15] "6d86c14f-cccf-41e9-a58e-ead9d52ae35d" "58"..
#>   ..$ language   : chr [1:15] NA NA NA NA ...
#>   ..$ modified   : chr [1:15] "2016-07-12" "2018-11-15" NA NA ...
#>   ..$ rights     : chr [1:15] "otherRestrictions" "otherRestrictions" "o"..
#>   ..$ source     : chr [1:15] NA NA NA NA ...
#>   ..$ subject    : chr [1:15] "infoMapAccessService, Rapportageeenheden,"..
#>   ..$ title      : chr [1:15] "KRW Waterlichamen en stroomgebieden - WMS"..
#>   ..$ type       : chr [1:15] "service" "service" "service" "service" ...
#>   ..$ URI        : chr [1:15] "http://data.waterkwaliteitsportaal.nl/ins"..
```

`brief` report in `xml` document

The last option for `output` is `xml`. Because this output is rather voluminous we demonstrate this with the `CSW_GetRecordById` function. This function uses the argument `id` instead of `constraint`. Here we indicate that we want the `brief` output (with only the fields `BoundingBox`, `identifier`, `title` and `type`) and we use the utility function `CSW_display_node` to show the resulting xml document.

```
xml1 = CSW_GetRecordById(
    id="baa1ea45-1cdc-4589-9793-b9f245b7776d",
    ElementSetName = 'brief', output='xml')
CSW_display_node(xml1)
#> <?xml version="1.0" encoding="UTF-8"?>
#> <csw:GetRecordByIdResponse xmlns:csw="http://www.opengis.net/cat/csw/2.0.2">
#>   <csw:BriefRecord xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:ows="http://www.opengis.net/ows
#>     <dc:identifier>baa1ea45-1cdc-4589-9793-b9f245b7776d</dc:identifier>
#>     <dc:title>Totaal stikstof, Waterbeheerplan</dc:title>
#>     <dc:type>dataset</dc:type>    <ows:BoundingBox crs="EPSG::28992">
#>       <ows:LowerCorner>6.05 51.31</ows:LowerCorner>
#>       <ows:UpperCorner>5.06 51.86</ows:UpperCorner>    </ows:BoundingBox>
#>   </csw:BriefRecord></csw:GetRecordByIdResponse>
```

I think it is most convenient that the data is returned in the form of table. But the entries contain more data than is included in the tables (even with the `full` output). When one needs that information one can use the `xml` output as shown above. In that case a second data model can be requested by specifying a different `outputSchema` like done here:

```
xml2 = CSW_GetRecordById(
    id="baa1ea45-1cdc-4589-9793-b9f245b7776d",
    namespace = 'gmd:http://www.isotc211.org/2005/gmd',
    outputSchema='http://www.isotc211.org/2005/gmd',
    ElementSetName = 'brief', output='xml')
CSW_display_node(xml2)
#> <?xml version="1.0" encoding="UTF-8"?>
#> <csw:GetRecordByIdResponse xmlns:csw="http://www.opengis.net/cat/csw/2.0.2">
#>   <gmd:MD_Metadata xmlns:gmd="http://www.isotc211.org/2005/gmd" xmlns:gco="http://www.isotc211.org/2
#>     <gmd:fileIdentifier>
#>       <gco:CharacterString>baa1ea45-1cdc-4589-9793-b9f245b7776d</gco:CharacterString>
#>     </gmd:fileIdentifier>    <gmd:hierarchyLevel>
#>       <gmd:MD_ScopeCode codeList="http://standards.iso.org/ittf/PubliclyAvailableStandards/ISO_19139
#>     </gmd:hierarchyLevel>    <gmd:identificationInfo>
#>       <gmd:MD_DataIdentification>        <gmd:citation>
#>           <gmd:CI_Citation>            <gmd:title>
#>               <gco:CharacterString>Totaal stikstof, Waterbeheerplan</gco:CharacterString>
#>             </gmd:title>          </gmd:CI_Citation>        </gmd:citation>
#>         <gmd:graphicOverview>          <gmd:MD_BrowseGraphic>
#>             <gmd:fileName>
#>               <gco:CharacterString>http://www.nationaalgeoregister.nl:80/geonetwork/srv/eng/resource
#>             </gmd:fileName>          </gmd:MD_BrowseGraphic>
#>         </gmd:graphicOverview>        <gmd:graphicOverview>
#>           <gmd:MD_BrowseGraphic>            <gmd:fileName>
#>               <gco:CharacterString>http://www.nationaalgeoregister.nl:80/geonetwork/srv/eng/resource
#>             </gmd:fileName>          </gmd:MD_BrowseGraphic>
#>         </gmd:graphicOverview>        <gmd:extent>          <gmd:EX_Extent>
#>             <gmd:geographicElement>
#>               <gmd:EX_GeographicBoundingBox>
#>                 <gmd:westBoundLongitude>
#>                   <gco:Decimal>5.06</gco:Decimal>
#>                 </gmd:westBoundLongitude>
#>                   <gmd:eastBoundLongitude>
#>                     <gco:Decimal>6.05</gco:Decimal>
#>                   </gmd:eastBoundLongitude>
#>                   <gmd:southBoundLatitude>
#>                     <gco:Decimal>51.31</gco:Decimal>
#>                   </gmd:southBoundLatitude>
#>                   <gmd:northBoundLatitude>
#>                     <gco:Decimal>51.86</gco:Decimal>
#>                   </gmd:northBoundLatitude>
#>                 </gmd:EX_GeographicBoundingBox>
#>               </gmd:geographicElement>          </gmd:EX_Extent>
#>           </gmd:extent>      </gmd:MD_DataIdentification>
#>       </gmd:identificationInfo>  </gmd:MD_Metadata>
#> </csw:GetRecordByIdResponse>
```

Queries

In the section CSW_Get* functions we showed how to constraint the records that are retrieved (`CSW_GetRecords`) or counted (`CSW_GetHits`) with the `constraint` argument. The query language and its version are set with the arguments `constraintLanguage` (default `CQL_TEXT`) and `constraint_language_version` (default `1.1.0`).

The GeoServer Tutorial and the ECQL Reference that is mentioned in this tutorial provide examples of possible queries. NB: GeoServer has extended the CQL language and these references don't make clear which language constructs are genuine CQL and which belong to the extended set. And apparently there are more CSW servers than GeoServer.

Wildcard queries

It possible to use wildcards in constraint. The `_` (underscore) stands for one arbitrary character and the `%` (percentage) stands for a character vector of arbitrary characters of length zero or more. The following examples count the records that have the phrase `water` somewhere in the title with and without the wildcards :

- `n1` counts the records that have the word `water` (irrespective to case) in the title. Records without this word but with e.g. `Rijkswaterstaat` are not counted.
- `n2` counts the records that have a word starting with `water` (irrespective to case) in the title. Records with `waterkwaliteit` are counted but also the words counted under `n1`.
- `n3` counts the records that have a word ending with `water` (irrespective to case) in the title. Records with `drinkwater` are counted but also the words counted under `n1`.
- `n4` counts the records that have a word containing the phrase `water` (irrespective to case) in the title. Records with e.g. `Rijkswaterstaat` are counted but also those under `n1`, `n2` and `n3`.
- `n5` counts the records that have a word containing the phrase `water` (irrespective to case) with one additional character in the title. Records with `Physical Waters` will match but not records with (only) `Rijkswaterstaat`.

```
(n1 = CSW_GetHits(constraint = "Title LIKE 'water'"))
#> [1] 13
(n2 = CSW_GetHits(constraint = "Title LIKE 'water%'"))
#> [1] 37
(n3 = CSW_GetHits(constraint = "Title LIKE '%water'"))
#> [1] 21
(n4 = CSW_GetHits(constraint = "Title LIKE '%water%'"))
#> [1] 57
(n5 = CSW_GetHits(constraint = "Title LIKE 'water_'"))
#> [1] 3
```

Of course, to actually retrieve these records replace `CSW_GetHits` by `CSW_GetRecords`.

Spatial queries

It is also possible to apply a constraint on the 474 records in this catalog based on the coordinates of the entries. These coordinates have to specified in the 'standard' coordinates in degrees in latitude/longitude order (EPSG:4326,WGS84). In the code below we firstly request (again) the number of records in the catalog. Then we request the number of records for which the BoundingBox intersects a region around Rotterdam. One of the data elements of a catalog entry is the `BoundingBox` and the Rotterdam area is also specified by its `BoundingBox`. In the last query we request the number of entries that do not intersect with the Rotterdam area: fully lie outside this area. In this way all entries with maps for the whole of the Netherlands will be excluded because they include the Rotterdam area. We are happy to see that the numbers `n7` and `n8` add up to `n6`.

```
# all records
(n6 = CSW_GetHits(constraint = ""))
#> [1] 474
# number of records for a bounding box intersecting with an area around Rotterdam
xmin = 3.91 ; xmax = 4.67 ; ymin = 51.78 ; ymax = 53.06 ;
( bbox_Rdam = glue::glue("{xmin}, {ymin}, {xmax}, {ymax}") )
#> 3.91, 51.78, 4.67, 53.06
( n7 = CSW_GetHits(constraint = glue::glue("BBOX(the_geom, {bbox_Rdam})")) )
#> [1] 454
# number of records for a bounding box not intersecting (this) Rotterdam area
( pol_Rdam = glue::glue("{xmin} {ymin}, {xmin} {ymax}, {xmax}  {ymax},  {xmax}  {ymin}, {xmin} {ymin}")
#> 3.91 51.78, 3.91 53.06, 4.67  53.06,  4.67  51.78, 3.91 51.78
( n8 = CSW_GetHits(constraint = glue::glue("DISJOINT(the_geom, POLYGON(({pol_Rdam})))")) )
#> [1] 20
```

See the spatial section of ECQL Reference for more possibilities for spatial queries

CSW_GetCapabilities

The `CSW_GetCapabilities` function gives information about the capabilities that are available to query the catalog. By calling this function without arguments we receive an `xml` document with this information

```
gc = CSW_GetCapabilities()
```

We can view the contents of this document (e.g. with the utility function `CSW_display_node`) and study its structure. We see among other things the operations that are available and the parameters with which these operations can be called. Apart of viewing the document we can also select parts of it with `xml2` package functions by using `XPATH` expressions. Here we will not display the whole document but only the part that is concerned with the `GetRecords` operation. So first we will use the `XPATH` language to find select (and display) the section about `GetRecords` in the `GetCapabilities` output

```
CSW_display_node(
    xml2::xml_find_first(gc, '//ows:Operation[@name="GetRecords"]'))
#> <ows:Operation name="GetRecords">      <ows:DCP>        <ows:HTTP>
#>           <ows:Get xlink:href="https://www.nationaalgeoregister.nl:443/geonetwork/srv/dut/csw-inspir
#>           <ows:Post xlink:href="https://www.nationaalgeoregister.nl:443/geonetwork/srv/dut/csw-inspi
#>         </ows:HTTP>      </ows:DCP>
#>       <!-- FIXME : Gets it from enum or conf -->
#>       <ows:Parameter name="resultType">        <ows:Value>hits</ows:Value>
#>         <ows:Value>results</ows:Value>
#>         <ows:Value>validate</ows:Value>      </ows:Parameter>
#>       <ows:Parameter name="outputFormat">
#>         <ows:Value>application/xml</ows:Value>      </ows:Parameter>
#>       <ows:Parameter xmlns:gfc="http://www.isotc211.org/2005/gfc" xmlns:dcat="http://www.w3.org/ns/d
#>         <ows:Value>http://www.opengis.net/cat/csw/2.0.2</ows:Value>
#>         <ows:Value>http://www.isotc211.org/2005/gfc</ows:Value>
#>         <ows:Value>http://www.w3.org/ns/dcat#</ows:Value>
#>         <ows:Value>http://www.isotc211.org/2005/gmd</ows:Value>
#>       </ows:Parameter>
#>       <ows:Parameter xmlns:gfc="http://www.isotc211.org/2005/gfc" xmlns:dcat="http://www.w3.org/ns/d
#>         <ows:Value>csw:Record</ows:Value>
#>         <ows:Value>gfc:FC_FeatureCatalogue</ows:Value>
#>         <ows:Value>dcat</ows:Value>
#>         <ows:Value>gmd:MD_Metadata</ows:Value>      </ows:Parameter>
#>       <ows:Parameter name="CONSTRAINTLANGUAGE">
#>         <ows:Value>FILTER</ows:Value>
#>         <ows:Value>CQL_TEXT</ows:Value>      </ows:Parameter>
#>       <ows:Constraint name="PostEncoding">
#>         <ows:Value>XML</ows:Value>        <ows:Value>SOAP</ows:Value>
#>       </ows:Constraint>      <ows:Constraint name="SupportedISOQueryables">
#>         <ows:Value>CreationDate</ows:Value>
#>         <ows:Value>GeographicDescriptionCode</ows:Value>
#>         <ows:Value>OperatesOn</ows:Value>
#>         <ows:Value>Modified</ows:Value>
#>         <ows:Value>DistanceUOM</ows:Value>
#>         <ows:Value>Operation</ows:Value>
#>         <ows:Value>ResourceIdentifier</ows:Value>
#>         <ows:Value>Format</ows:Value>
#>         <ows:Value>Identifier</ows:Value>
#>         <ows:Value>Language</ows:Value>
#>         <ows:Value>ServiceType</ows:Value>
#>         <ows:Value>OrganisationName</ows:Value>
#>         <ows:Value>KeywordType</ows:Value>
#>         <ows:Value>AnyText</ows:Value>
#>         <ows:Value>PublicationDate</ows:Value>
#>         <ows:Value>AlternateTitle</ows:Value>
#>         <ows:Value>Abstract</ows:Value>
#>         <ows:Value>HasSecurityConstraints</ows:Value>
#>         <ows:Value>Title</ows:Value>
#>         <ows:Value>CouplingType</ows:Value>
#>           <ows:Value>TopicCategory</ows:Value>
#>           <ows:Value>ParentIdentifier</ows:Value>
#>           <ows:Value>Subject</ows:Value>
#>           <ows:Value>ResourceLanguage</ows:Value>
#>           <ows:Value>TempExtent_end</ows:Value>
#>           <ows:Value>ServiceTypeVersion</ows:Value>
#>           <ows:Value>Type</ows:Value>
#>           <ows:Value>RevisionDate</ows:Value>
#>           <ows:Value>OperatesOnName</ows:Value>
#>           <ows:Value>Denominator</ows:Value>
#>           <ows:Value>DistanceValue</ows:Value>
#>           <ows:Value>TempExtent_begin</ows:Value>
#>           <ows:Value>OperatesOnIdentifier</ows:Value>      </ows:Constraint>
#>         <ows:Constraint name="AdditionalQueryables">
#>           <ows:Value>SpecificationDate</ows:Value>
#>           <ows:Value>AccessConstraints</ows:Value>
#>           <ows:Value>ResponsiblePartyRole</ows:Value>
#>           <ows:Value>Degree</ows:Value>        <ows:Value>Lineage</ows:Value>
#>           <ows:Value>OnlineResourceMimeType</ows:Value>
#>           <ows:Value>ConditionApplyingToAccessAndUse</ows:Value>
#>           <ows:Value>Date</ows:Value>
#>           <ows:Value>MetadataPointOfContact</ows:Value>
#>           <ows:Value>OnlineResourceType</ows:Value>
#>           <ows:Value>Relation</ows:Value>
#>           <ows:Value>SpecificationDateType</ows:Value>
#>           <ows:Value>Classification</ows:Value>
#>           <ows:Value>OtherConstraints</ows:Value>
#>           <ows:Value>SpecificationTitle</ows:Value>      </ows:Constraint>
#>       </ows:Operation>
```

In the previous output we see the parameters and the fields that can be used in the `GetRecords` operation (i.e. the `CSW_GetRecords` function discussed above): e.g. parameter `outputFormat` is restricted to the value `application/xml` and parameter `typeNames` accepts four different values. In the same way we find the names of the operation sections.

```
ops=purrr::map_chr(xml2::xml_find_all(gc, '//ows:Operation'),~xml2::xml_attr(.,'name'))
print(ops)
#> [1] "GetCapabilities" "DescribeRecord"  "GetDomain"       "GetRecords"
#> [5] "GetRecordById"   "Transaction"     "Harvest"
```

So we see that `GetCapabilities` declares 7 operations: `GetCapabilities`, `DescribeRecord`, `GetDomain`, `GetRecords`, `GetRecordById`, `Transaction` and `Harvest`. We will discuss now the remaining (read-only) operations.

CSW_DescribeRecords

One of the CSW requests is for retrieving a description of the data that can be retrieved. I could not get it to work until I saw the gist by FrieseWoudloper: the request uses the `typeName` and not the `typeNames` parameter. Both parameter names are now accepted by the package. The gist also mentions and demonstrates two data models. The example below does three request to retrieve a data model as show in the gist: first for the `Dublin Core metadatamodel`, then for the `ISO 19119 metadatamodel` and lastly for both models. The outputs are not shown here because they are very voluminous.

```
## Dublin Core metadatamodel :
dr= CSW_DescribeRecord(
    namespace = 'csw:http://www.opengis.net/cat/csw/2.0.2',
    typeNames = 'csw:Record',verbose='F')
CSW_display_node(dr)
## ISO 19119 metadatamodel :
dr=  CSW_DescribeRecord(
    namespace = 'gmd:http://www.isotc211.org/2005/gmd',
    typeNames = 'gmd:MD_Metadata')
CSW_display_node(dr)
## both Dublin Core and ISO 19119 metadatamodel :
dr =  CSW_DescribeRecord(
    namespace = c('csw:http://www.opengis.net/cat/csw/2.0.2',
        'gmd:http://www.isotc211.org/2005/gmd'),
    typeNames = c('csw:Record','gmd:MD_Metadata') )
CSW_display_node(dr)
```

CSW_GetDomain

With the `GetDomain` operation the CSW server can return information about `ParameterNames` or `PropertyNames`.

CSW_GetDomain ParameterName

We can inquire after the parameters of an operation by specifying the `ParameterName` argument. This argument consists of one or more `operation.parameter` pairs as in the following example. The output is a list (or alternatively an xml_document) with the values that can be used for the parameters. In each pair the part before the point should be an operation; the part after the point a parameter. CSW_GetDomainParameterNames gives you the same information for all possible combinations (at the cost of some extra run-time). An example of `CSW_GetDomain` for two parameters:

```
x = CSW_GetDomain(
    ParameterName='DescribeRecord.outputFormat,GetRecords.outputSchema',
    output='list')
print(x)
#> $DescribeRecord.outputFormat
#> [1] "application/xml"
#>
#> $GetRecords.outputSchema
#> [1] "http://www.isotc211.org/2005/gmd"
#> [2] "http://www.opengis.net/cat/csw/2.0.2"
#> [3] "http://www.isotc211.org/2005/gfc"
#> [4] "http://www.w3.org/ns/dcat#"
```

So we see that the `outputFormat` parameter of the `DescribeRecord` operation can take one value ('application/xml') and the `outputSchema` parameter of the `GetRecords` operation can take four.

CSW_GetDomain PropertyName

We can also use the `CSW_GetDomain` function to inquire which values a certain field can take in a catalog. We do this by specifying the fieldname in the `PropertyName` argument. This argument consists of one or more fieldnames as in the following example. The output is (just as in the ParameterName case) a list (or alternatively an xml_document) with the values are taken (at that moment) by the field. CSW_GetQueryables will show you which fieldnames (`PropertyNames`) are recognized.

```
gd = CSW_GetDomain(
    PropertyName='Language,GeographicDescriptionCode',
    output='list')
print(gd)
#> $Language
#> [1] "dut"
#>
#> $GeographicDescriptionCode
#> [1] "Nederland"
#> [2] "Nederland / Noordzee"
#> [3] "Nederland (land)"
#> [4] "Nederland (land en zee)"
#> [5] "Netherlands"
#> [6] "Provincie Fryslân"
#> [7] "Provincie Gelderland, Overijssel, Limburg en Utrecht"
```

CSW_GetDomainParameterNames

The function `CSW_GetDomainParameterNames` calls CSW_GetDomain for each `operator*parameter` combination and places the result in a data.frame.

```
gdp = CSW_GetDomainParameterNames()
knitr::kable(gdp)
```

n	v
DescribeRecord.namespace	http://www.isotc211.org/2005/gfc
DescribeRecord.namespace	http://www.isotc211.org/2005/gmd
DescribeRecord.namespace	http://www.opengis.net/cat/csw/2.0.2
DescribeRecord.namespace	http://www.w3.org/ns/dcat#
DescribeRecord.outputFormat	application/xml
DescribeRecord.typeName	csw:Record
DescribeRecord.typeName	dcat
DescribeRecord.typeName	gfc:FC_FeatureCatalogue
DescribeRecord.typeName	gmd:MD_Metadata
GetRecords.ElementSetName	brief
GetRecords.ElementSetName	full
GetRecords.ElementSetName	summary
GetRecords.outputFormat	application/xml
GetRecords.outputSchema	http://www.isotc211.org/2005/gfc
GetRecords.outputSchema	http://www.isotc211.org/2005/gmd
GetRecords.outputSchema	http://www.opengis.net/cat/csw/2.0.2
GetRecords.outputSchema	http://www.w3.org/ns/dcat#
GetRecords.resultType	hits
GetRecords.resultType	results
GetRecords.resultType	results_with_summary
GetRecords.resultType	validate
GetRecords.typeNames	csw:csw:Record
GetRecords.typeNames	dcat:dcat

n	v
GetRecords.typeNames	gfc:gfc:FC_FeatureCatalogue
GetRecords.typeNames	gmd:gmd:MD_Metadata

CSW_GetQueryables

The function `CSW_GetQueryables` retrieves the names of properties (fields) that can be used in the CSW_GetDomain (`PropertyName` case) or in a query. The output is a list with two sublists: one with the 'SupportedISOQueryables' and one with the 'AdditionalQueryables'.

```
gq = CSW_GetQueryables()
gq
#> $SupportedISOQueryables
#>  [1] "CreationDate"              "GeographicDescriptionCode"
#>  [3] "OperatesOn"                "Modified"
#>  [5] "DistanceUOM"               "Operation"
#>  [7] "ResourceIdentifier"        "Format"
#>  [9] "Identifier"                "Language"
#> [11] "ServiceType"               "OrganisationName"
#> [13] "KeywordType"               "AnyText"
#> [15] "PublicationDate"           "AlternateTitle"
#> [17] "Abstract"                  "HasSecurityConstraints"
#> [19] "Title"                     "CouplingType"
#> [21] "TopicCategory"             "ParentIdentifier"
#> [23] "Subject"                   "ResourceLanguage"
#> [25] "TempExtent_end"            "ServiceTypeVersion"
#> [27] "Type"                      "RevisionDate"
#> [29] "OperatesOnName"            "Denominator"
#> [31] "DistanceValue"             "TempExtent_begin"
#> [33] "OperatesOnIdentifier"
#>
#> $AdditionalQueryables
#>  [1] "SpecificationDate"                 "AccessConstraints"
#>  [3] "ResponsiblePartyRole"              "Degree"
#>  [5] "Lineage"                           "OnlineResourceMimeType"
#>  [7] "ConditionApplyingToAccessAndUse" "Date"
#>  [9] "MetadataPointOfContact"            "OnlineResourceType"
#> [11] "Relation"                          "SpecificationDateType"
#> [13] "Classification"                    "OtherConstraints"
#> [15] "SpecificationTitle"
```

In the first sublist we see the fields `Language` and `GeographicDescriptionCode` that were used in CSW_GetDomain PropertyName.

References

Supported filter languages OpenGIS Catalog Services Specification

- Nationaal GeoRegister (NGR) : hub for location spatial information for the Netherlands (in Dutch) http://nationaalgeoregister.nl/geonetwork/srv/dut/search

- Publieke Dienstverlening op de Kaart (PDOK) : platform for open spatial data (in Dutch)
  https://www.pdok.nl/
- Description APIs (including CSW) for PDOK environment (in Dutch)
  https://pdok-ngr.readthedocs.io/.
- Description of GeoServer services (including CSW)
  https://docs.geoserver.org/latest/en/user/services/index.html
- Catalogue Service page of the Open Geospatial Consortium (OGC)
  http://www.opengeospatial.org/standards/cat
- Specification of Catalog Services for the Web (CSW) version 2.0.2
  http://portal.opengeospatial.org/files/?artifact_id=20555
- GeoServer Tutorial section about queries
  https://docs.geoserver.org/stable/en/user/tutorials/cql/cql_tutorial.html
- GeoServer reference for ECQL queries
  https://docs.geoserver.org/stable/en/user/filter/ecql_reference.html#filter-ecql-reference
- Examples for DescribeRecords by FrieseWoudloper
  https://gist.github.com/FrieseWoudloper/b7cad022cb75ba531ebbececf5fc85db

Back to top