European Direction in GCI Enhancements

D3.6

The DAB APIs User Manual – v2.0

Workpackage:	WP3	
Task:	T3.3	
Author(s):	Roberto Roncella	CNR-IIA
	Mattia Santoro	CNR-IIA
	Paolo Mazzetti	CNR-IIA
Authorized by	Joost van Bemmelen	
Doc Id:	EDGE-WP3-DEL-D3.6	
Reviewer	UTB	
Dissemination Level	Public	

Abstract:

This document describes the high-level GEO Discovery and Access Broker Application Programming Interfaces (APIs). The GEO DAB is the middleware component which is in charge of interconnecting the capacities contributing to the Global Earth Observation System of Systems (GEOSS).

GEO DAB users are typically software agents, such as web-based or desktop client applications. In the context of the GEOSS Platform, these APIs are utilized by the GEOSS Portal (EDGE Deliverable 3.5) to interact with GEO DAB.

Document Log

Date	Author	Changes	Version	Status
25/05/2020	EDGE Team		2.0	Delivered

Executive Summary

This document, (*D3.6, The DAB APIs User Manual*), describes the high-level GEO Discovery and Access Broker Application Programming Interfaces. The GEO DAB is a middleware component which is in charge of interconnecting the heterogeneous and distributed capacities contributing to the Global Earth Observation System of Systems (GEOSS). In the context of the GEOSS Platform, these APIs are utilized by the GEOSS Portal (EDGE Deliverable 3.5) to interact with GEO DAB.

The main functionalities of the component are:

- **Data/Metadata Harmonization**: this layer provides harmonized discovery and access to heterogeneous data systems. The heterogeneity of data sources is hidden, resources appear as a single data source.
- **Data Access**: this layer provides data discovery and access functionalities to heterogeneous data systems.
- **Data Transformation**: this layer enriches access functionalities by allowing users to customize their downloads (e.g. change format and/or CRS).

Since it is a middleware component, GEO DAB users are typically software agents, such as web-based or desktop client applications. These can exploit the GEO DAB functionalities implementing the client-side of one (or more) of the protocols published by the GEO DAB for the above functionalities. The available protocols include:

- OGC Catalog Service for the Web (CSW)
- OpenSearch with geo, time and domain-specific extensions
- Open Archive Initiative (OAI) PMH
- OGC Web Processing Service

This document updates the previous version (D3.3) of GEO DAB APIs description.

TABLE OF CONTENTS

1. Introduction

1.1 Purpose and Scope

This document (*D3.6, The DAB APIs User Manual*) has been generated in the context of *WP3 - GEOSS Portal and GEO DAB Enhancements* within the *EDGE* (short for European Direction in GEOSS Common Infrastructure Enhancements) project, Grant Agreement no 776136 [1].

The primary objective of this document is to describe high level GEO Discovery and Access Broker (DAB) APIs. In the context of the GEOSS Platform, these APIs are utilized by the GEOSS Portal (EDGE Deliverable 3.5) to interact with GEO DAB. This document illustrates the APIs functionalities offered to the GEOSS Portal and of step-by-step instructions for using them. The new enhancements have been architecturally described in the EDGE-WP3-DEL-D3.4 [2], following the specification as identified and documented in the context of WP2 in the document EDGE-WP2-DEL-D2.4 [3], *Functional and Non-functional Enhancements Specification*, that underpin the user needs elicited and analysed in the context of the document EDGE-WP2-DEL-D2.3 [4], *Use Cases Description and User Requirements Document*.

1.2 Document Organisation

The document is organised as it follows:

- Section 1: Introduction: describe the purpose and scope of the document and its organization.
- Section 2: Describe the advanced GEO DAB APIs with particular focus on Client-Side API (Section 2.1), RESTful API (Section 2.2), Asynchronous Download (Section 2.3), Bulk Download facility (Section 2.4) and Statistics API (Section 2.5).
- Annex A: lists the references used in the document.
- Annex B: explain the meaning of the acronyms and definitions used in the document.

2. GEO DAB API

The GEO Discovery and Access Broker (DAB) is a component of the GEOSS Platform (since November 2011) that transparently connects GEOSS User's requests to the resources shared by the GEOSS Providers. It is a brokering framework that interconnects hundreds of heterogeneous and autonomous systems by providing mediation and harmonization capabilities. The GEO DAB provides the following main functionalities:

1. Discovery of geospatial data;
2. Harmonized access to geospatial data.

Since it is a middleware component, GEO DAB users are typically software agents, such as web-based or desktop client applications. These can exploit the GEO DAB functionalities implementing the client-side of one (or more) of the protocols published by the GEO DAB for the above functionalities. The available protocols include:

- OGC Catalog Service for the Web (CSW)[1];
- OpenSearch[2] with geo, time and domain-specific extensions;
- Open Archive Initiative (OAI) PMH[3];
- OGC Web Processing Service[4];
- Etc.

In order to simplify the development of applications and clients making use of the GEO DAB, a set of high-level client-side Open APIs (Application Program Interface) have been designed and developed along with documentation and usage examples. This documentation is intended to be used by developers. In particular we will describe how Web and mobile app developers can interact with the GEO DAB using the following available programming interfaces:

- **Client-side API**
- **RESTful API**
- **Asynchronous Download**
- **Bulk Download**
- **Statistics API**

The full version of this documentation with examples is available at the following address: http://api.eurogeoss-broker.eu.

[1] http://www.opengeospatial.org/standards/specifications/catalog

[2] http://www.opensearch.org/Home

[3] http://www.openarchives.org/OAI/openarchivesprotocol.html

[4] http://www.opengeospatial.org/standards/wps

2.1 GEO DAB Client-side API

The GEO DAB Client-Side API is a HTML5+Javascript+CSS library providing a set of predefined software objects which facilitates the development of web and mobile applications, hiding the interaction with the GEO DAB. The main modules of the GEO DAB Client-side API are:

- **Core**
- **AccessOptions**
- **Concept**
- **Refinement**
- **Report**
- **ResultSet**
- **Retrieval**

2.1.1 Core

This module contains the API entry point DAB and other core objects.

GINode

A GINode is a Geo Information node representing an abstract geospatial resource, a "dataset" or a "service", available as result of a query to the DAB.

Nodes properties such as title and abstract are described by a report. A particular Report property attribute is type, which defines which kind of interactions are available with the node. When a node represents a "service" such as WMS, WCS, etc. the report has an additional service property.

Node accessibility: Usually a node is linked to one or more data. If the linked data is accessible by means of a set of options, the node is accessible otherwise it is directly accessible; a node can be both accessible and directly accessible:

- accessible nodes provide one or more links that can be retrieved by means of the `accessLink` method. To test if this node is accessible use the `isAccessible` method
- directly accessible nodes provide one or more links that can retrieved by means of the `directAccessLinks` method. To test if this node is directly accessible use the `isDirectlyAccessible` method

Methods

accessLink (onResponse, [options]) async

If this node is accessible, retrieves the link to access the data linked to this node, otherwise if this node is not accessible, this method does nothing.

The data is accessed using a set of options which allow to specify several output parameters such as Coordinate Reference System. Not all the possible combinations of access options parameters are valid for a given data; the valid set of parameters to access the data of this node can be retrieved with the `accessOptions` method. If the options parameter is omitted, the data link is created using the available `reducedOption` that can be retrieved with the `accessOptions` method.

Parameters

- `onResponse : Function`

  Callback function for receiving asynchronous query response. The function has the following arguments:

  - `link : String`

    The access link; in case of error this argument is set to null.

  - `error : String (optional)`

    In case of error this argument contains a message which describes the problem occurred.

- `options : AccessOptions (optional)`

  The access options which allows to specify the data output parameters. If omitted, the data link is created using the available reducedOption (see `accessOptions` method).

accessOptions (onResponse) async

If this node is accessible, retrieves one or more objects containing a list of `validOptions`, the `defaultOption` and the `reducedOption` to use as options for the `accessLink` method.

If this node is not accessible, this method does nothing.

Each returned object contains a `validOptions` array, a `defaultOption` and a `reducedOption`.

Parameters

- `onResponse : Function`

  Callback function for receiving asynchronous query response. The function has the following arguments:

  - `options : Object`

    The access options; in case of error this argument is set to null

  - `error : String (optional)`

    In case of error this argument contains a message which describes the problem occurred.

directAccessLinks () : String[]

Returns an array containing all the available direct access links.

Returns

An array containing all the available direct access links or an empty array if none are available.

expand (onResponse, [pageSize=10]) async

If this node is composed, the result of the expansion is a result set split in pages (possibly one if the matched nodes are minor than the result set page size).

The result set contains the underlying first level nodes (this node children). If this node is a DAB instance, than the first call of this method returns the correspondent sources.

If the result set is split in more than one page use the `expandNext` method to retrieve the next ones. If this is a simple node, the result set is empty.

Parameters

- `onResponse : Function`

  Callback function for receiving asynchronous query response. The function has the following argument:

  - `resultSet : ResultSet`

    The expansion result set. In case of error, this argument contains an empty result set with the ResultSet/error property describing the problem occurred.

- `pageSize=10 : Integer (optional)`

  The size of the result set pages.

expandNext (onResponse, [execute]) : Boolean async

If this node has already been expanded, this method tests if other pages in the result set are available. If the test is positive and the execute argument is true, retrieves the next result set page. If this node is a composed node but it has never been expanded, an Exception is thrown. If this is a simple node, this method returns false.

Parameters

- `onResponse : Function`

  Callback function for receiving asynchronous query response.

- `execute : Boolean (optional)`

  If omitted or set to false, this method tests if other pages in the result set are available and returns the test result. If set to true and the test is positive, this method retrieves the next result set page and returns true. If set to true and the test is negative this method returns false.

Returns

True if other pages in the result set are available, False otherwise.

googleImageMapType (map, [initOptions]) : google.maps.ImageMapType[]

Retrieves an array of google.maps.ImageMapType; if the current node does not provide any layer an empty array is returned. All the available ImageMapTypeOptions are allowed; this method implements the `getTileUrl` function in order to build the correct WMS GetMap request.

Parameters

- `map : google.maps.Map`

  Instance required for the implementation of the `getTileUrl` function.

- `initOptions : Boolean (optional)`

  See Google ImageMapTypeOptions for more info.

Returns

An array (possible empty) of google.maps.ImageMapType.

has_olVector_Layer () : String[]

This method checks if the node provides one or more layers of type `OpenLayers.Layer.Vector`. The method returns an array of Strings, each string is the name of an available vector layer.

Returns

The Array of names of available `OpenLayers.Layer.Vector` layers.

isAccessible ()

Tests if this node is accessible.

Returns

True if this node is linked to one or more data and it is asynchronously accessible, false otherwise.

isDirectlyAccessible ()

Tests if this node is directly accessible.

Returns

True if this node is linked to one or more data and it has one or more direct access links, false otherwise.

ol3WMS_Layer ([opt_options]) : ol.source.TileWMS []

Retrieves an array of ol.source.TileWMS ; if the current node does not provide any layer an empty array is returned. As explained in the ol.source.TileWMS constructor documentation, the returned layers can be optionally created with additional parameters and options for the GetMap request (see `opt_options` argument).

Parameters

- `opt_options : Object (optional)`

  See Tile WMS options in the ol.source.TileWMS constructor.

Returns

An Array of ol.source.TileWMS possible empty if this node does not provide any layer.

olTiles_Layer ([initOptions]) : OpenLayers.Layer.WMS[]

Retrieves an array of OpenLayers.Layer.WMS (TiledMapService type); if the current node does not provide any layer an empty array is returned. As explained in the OpenLayers.Layer.WMS constructor documentation, the returned layers can be optionally created with additional parameters and options for the GetMap request (see `initOptions` argument).

Parameters

- `initOptions : Object (optional)`

  Object literal of options passed the OpenLayers.Layer.WMS constructor. The function has the following argument:

  - `params : Object`

    This argument corresponds to the OpenLayers.Layer.WMS params argument. Note that the layers property is provided by the API.

- `options : String (optional)`

  This argument corresponds to the OpenLayers.Layer.WMS options argument.

Returns

An Array of OpenLayers.Layer.WMS possible empty if this node does not provide any layer.

olWMS_Layer ([initOptions]) : OpenLayers.Layer.WMS[]

Retrieves an array of OpenLayers.Layer.WMS; if the current node does not provide any layer an empty array is returned. As explained in the OpenLayers.Layer.WMS constructor documentation, the returned layers can be optionally created with additional parameters and options for the GetMap request (see `initOptions` argument).

Parameters

- `initOptions : Object (optional)`

  Object literal of options passed the OpenLayers.Layer.WMS constructor. The function has the following argument:

  - `params : Object`

    This argument corresponds to the OpenLayers.Layer.WMS params argument. Note that the layers property is provided by the API.

  - `options : String (optional)`

    This argument corresponds to the OpenLayers.Layer.WMS options argument.

Returns

An Array of OpenLayers.Layer.WMS possible empty if this node does not provide any layer.

olVector_Layer ([onResponse]) async

Retrieves an array of OpenLayers.Layer.Vector; if the current node does not provide any layer an empty array is returned. Besides, client applications are required (at the moment) to handle the cross-origin nature of the request which is executed (GML is an XML encoding).

Parameters

- `onResponse : Function`

  Callback function for receiving asynchronous result. The function has the following argument:

  - `result : Object/Array`

    The Array of available OpenLayers.Layer.Vector objects.

overview (id, onResponse, [options]) : Boolean async

If this node has an overview, creates an <img> having as source the overview link at the given `options.index` (0 if not specified) and appends <img> to the element having the given id. The overview property can be available also if this node is not accessible. The method throws an Exception if it does not correspond to an element or if `options.index` does not respect the following statement: `index >= 0 && index <= overview.length - 1`.

Parameters

- `id : String`

  The id of an element in which to append the <img> element of the selected overview.

- `onResponse : Function`

  Callback function for receiving asynchronous request status. The function has the following arguments:

  - `status : String`

    Possible status are 'success' and 'error'.

  - `message : String (optional)`

    In case of 'error' status this argument contains the error message.

  - `image : String (optional)`

    The loaded image element or null in case of error.

- `options : Object (optional)`

  - `index: Number (optional, default = 0)`

    Index of this node overview array.

  - `size : String (optional, default = medium)`

    The overview size (aspect ration is preserved). Possible values are:

    - "extra small": 48x48 px;
    - "small": 96x96 px;
    - "medium": 144x144 px;
    - "large": 288x288 px;
    - "original": the original size of the target overview image.

  - `force : Boolean (optional, default = false)`

    If set to True a "no overview" image is used in case of error or in case an overview is not available.

  - `backgroundURL : String (optional)`

    URL of an additional image to load as background of the overview image (use this only if the overview image is transparent).

Returns

True if this node has an overview, false otherwise.

report ([newReport]) : Report

Retrieves and optionally set a new report.

Parameters

- `newReport : Report (optional)`

  The new report.

Returns

The node report.

DAB (extends GINode)

The API entry point. This object provides the ability to create a node connected to the existing GEO DAB server instance. DAB is a composed node and it's the root of the hierarchical structure defined by the DAB brokered sources.

The main DAB operation allows to discover any nodes regardless of their level in the hierarchy.

The discover supports basic and advanced constraints. The basic constraints are:

- **what**
- **where**
- **when**
- **who**

The following parameters expressed as key-value pair allow to set advanced constraints:

- *Miscellaneous*
    - **loc**: name of a location where to constraint the discover. It can be used instead of the where constraint. E.g: "italy", "u.s.a", "africa".
    - **gdc**: includes only or excludes all the "GEOSS Data Core" datasets in the/from the discover. Possible values are 'true' or 'false'.
    - **sources**: comma separated list of sources identifiers; only the related sources are included in the discover. This is an alternative way to the `DABSource.include` method.
    - **sba**: a "GEO Societal Benefit Area". E.g.:"agriculture", "climate", "disasters".
    - **hl**: the type of node to discover. Possible values are: "dataset" for simple nodes, "series" for composed nodes.
    - **prot**: a protocol used to access the data linked to the discovered nodes. E.g: "HTTP", "urn:ogc:serviceType:WebMapService:1.1.1:HTTP", "OGC:WMS-1.1.1-http-get-map".
    - **frmt**: a format of the data linked to the discovered nodes. E.g.: "image/gif", "application/zip".
    - **kwd**: a keyword which describes the nodes to discover.
- *Legal constraints*
    - **uselim**: limitation applied on the use of the data linked to the discovered nodes (string).
    - **lac**: includes only or excludes all the nodes having some legal access constraints. Possible values are 'true' or 'false'.
    - **luc**: includes only or excludes all the nodes having some legal use constraints. Possible values are 'true' or 'false'.
- *IRIS Event constraints*
    - **minmag**: integer value of the minimum magnitude.
    - **maxmag**: integer value of the maximum magnitude.
    - **magt**: the type of magnitude (case insensitive). E.g.: "ML", "Ms", "mb", "Mw", "all", "preferred".
    - **mind**: limit to events with depths equal to or greater than the specified depth. Double value expressed in kilometres.

- **maxd**: limit to events with depths less than or equal to the specified depth. Double value expressed in kilometres.
- **evtOrd**: ordering of the results. Possible values are: "time" and "magnitude".

- *INPE constraints*
    - **inpe-sat-name**: satellite name (string). E.g.: "AQUA"
    - **inpe-instr-name**: instrument name (string). E.g.: "MODIS"
- *Earth Observation constraints*
    - **sta**: station description (string)
    - **sensor**: sensor description (string)
    - **bandwl**: histogram mean (double)
    - **proclev**: processing level (double)
    - **illazan**: illumination azimuth angle (double)
    - **illzean**: illumination zenith angle value (double)
    - **cloudcp**: cloud cover percentage (double)
    - **sarPolCh**: polarisation channels (string)
    - **sarPolMd**: polarisation mode (string)
    - **sensorResolution**: sensor resolution (double)
    - **sensorResolutionMax**: maximum sensor resolution (double)
    - **sensorResolutionMin**: minimum sensor resolution (double)

Constructor

DAB (endpoint, [viewId], [servicePath], [cswPath], [openSearchPath])

Parameters

- `endpoint : String`

  The endpoint of the DAB instance to connect.
- `viewId : String (Optional)`

  A "server-side" view identifier.
- `servicePath: String (Optional)`

  The path of services. Default value: *services*.
- `cswPath: String (Optional)`

  The path of OpenSearch service: Default value: *cswiso*.
- `openSearchPath: String (Optional)`

  The path of OpenSearch service: Default value: *opensearch*.

Methods

The DAB class inherits all methods from super-class GINode and implements the following.

discover (onResponse, [constraints], [options], [onStatus]) async

While the expand is limited to the node children, the result set of a discover query contains any GINode with a report matching the given constraints (if any), regardless of their level in the hierarchy.

Normally the result of this method is a single ResultSet. When the extension option is used, the discover might generate more than one ResultSet according to the following rules:

1. generation of concepts list
2. execution of a specific query for each concept in the list
3. creation of a ResultSet for each query with at least one result

The ResultSet can be extended by keyword or by concepts. In the first case the list of concepts is generated by retrieving concepts related to the provided keyword (see also `concept` method). In the second case the list of concepts is the provided one. Besides, if the provided relation is different from NONE, the generated list contains also concepts satisfying the provided relation.

The method throws an Exception if the extension option is not properly configured (see `extension` option for more info).

Parameters

- `onResponse : Function`

  Callback function for receiving asynchronous query response. The function has the following argument:

  - `response : Array`

    Array of ResultSet resulting from the query. in case of error or timeout (see `options.timeout`), the response array contains an empty ResultSet result and the Error String property describes the problem occurred.

- `constraints : Object (optional)`

  Only nodes matching the given constraints is returned as result of the discover. The constraints argument object supports the following properties:

  - **where**: A bounding box which includes the results. See also `spatial relation` option.
  - **when**: A time period which includes the results.
  - **what**: A single search keyword or an array of search keywords; in case of multiple keywords, unless specified with the options.searchOperator, the search terms are related by OR logic operator.
  - **who**: A single report id or an array report id; only the content of the correspondent nodes (matching the above constraints if any) is discovered. In case of simple nodes (which have no content), they are always added to the result set regardless of the above constraints (if any).
  - **kvp**: A single advanced constraint or array of advanced constraints submitted in the key-value pair format.

- `options : Object (optional)`

  This parameter object supports the following properties:

  - `start: Integer (optional)`

    The start index of the first result set node. Default value: 1.

  - `pageSize: Integer (optional)`

    The size of the result set pages. Default value: 10.

  - `extension : Object (optional)`

    This option allows to extend the discover and has the following properties:

    - `relation : Relation`

The relation to be used for the extension. If different from NONE and neither keyword nor concepts are set, an Exception is thrown.

    - `keyword : String (optional)`

      The keyword to used to generate the extension concepts. If the concepts is also set, an Exception is thrown.

    - `concepts : Array (optional)`

      The array of concepts to use for the extension. If keyword is also set, an Exception is thrown.

  - `timeout: Integer (optional)`

    set a request timeout (in milliseconds).

  - `searchFields : String (optional)`

    A single search field or a comma separated list of search fields in which to perform the keyword search (see `constraints.what`). Possible search fields are:

    - **title** (default): the search is performed in the report title
    - **keyword** (default): the search is performed in the report keyword
    - **anytext**: the search is performed in the whole textual content of the report
    - **description**: the search is performed in the report description

  - `searchOperator: String (optional)`

    The logic operator for multiple keywords in the constraints.what constraints. Possible values are: "OR" and "AND".

  - `spatialRelation: SpatialRelation (optional)`

    The spatial relation used to include the results in the bounding box selected with the where constraint.

  - `termFrequency : String (optional)`

    One or more comma separated list of term frequency target; by default all the term frequency targets are selected. Possible term frequency targets are:

    - **keyword**
    - **format**
    - **protocol**
    - **source**

- `onStatus : Function (optional)`

  Callback function called every second for receiving asynchronous source status info. It has the following argument:

  - `status : Array (optional)`

    Every second, this argument is updated with an array of SourceStatus, one for each DAB source plus another representing the DAB node with the status of the whole process and with title "DAB".

concept (keyword, onResponse, [options]) async

Retrieves concepts related to the given keyword.

Parameters

- `keyword : String`

  The keyword by which generates the related concepts.
- `onResponse : Function`

  Callback function for receiving asynchronous query response. The function has the following arguments:
  - `result : Array`

    Array of concepts In case of error, this argument contains an empty array.
  - `error : String (optional)`

    In case of error, this argument contains a message which describes the problem occurred.
- `options : Object (optional)`

  Object literal of available options. The options are:
  - **`start`** `: Integer (optional)`

    The start index of the first returned concept. Default value: 1.
  - `count : Integer (optional)`

    The maximum number of concepts to return. Default value: 10.
  - `topLevel : Boolean (optional)`

    If set to True only the top level concepts is retrieved.

cswPath () : String

Retrieves the CSW path.

Returns

The service CSW path.

openSearchPath () : String

Retrieves the OpenSearch path.

Returns

The service OpenSearch path.

servicePath () : String

Retrieves the service path.

Returns

The service path.

endpoint () : String

Retrieves the DAB endpoint.

Returns

The URL String of this DAB instance.

find (id, onResponse) async

Retrieves the node with the given report id.

Parameters

- `id : String`

  The node report id.
- `onResponse : Function`

  Callback function for receiving asynchronous query response. The function has the following arguments:
  - `result : GINode`

    The requested node object. If a node with the given report id does not exist or in case of error, this argument is null.
  - `error : String (optional)`

    In case of error, this argument contains a message which describes the problem occurred.

sources (onResponse, [viewId]) async

Retrieves the nodes representing the sources brokered by this DAB instance. DAB sources can also be retrieved as result of the first call of the expand method.

Parameters

- `onResponse : Function`

  Callback function for receiving asynchronous query response. The function has the following arguments:
  - `result : Array`

    Array of DAB sources In case of error, this argument contains an empty array.
  - `error : String (optional)`

    In case of error, this argument contains a message which describes the problem occurred.
- `viewId : String (optional)`

  The id of the View.

view ([action], [constraints])

Defining a "client-side" view on the result set. A "client-side" view is a way to limit the result set on one or more specific constraint like where constraint, thus providing a "view" of the entire result set. Additional constraints used in the discover are merged with the view constraints in order to further refine the initial result set.

For example, a web portal can provide earthquakes info only on the west coast of south America, by defining a view with the where constraint covering that area. When the users make a search they can retrieve only results from the area defined in the view; furthermore they can add, for example, when constraint to retrieve results in a particular temporal period.

Parameters

- `actions : String (optional)`

  Possible values are: "enable" to enable the view, "disable" to disable it. It can be omitted if this method is used only as "get" method.
- `constraints : Object (optional)`

  The view constraints (unnecessary if the action is "disable" or if this method is used only as "get" method).

Returns

The current view constraints.

DABSource (extends GINode)

This kind of node represents a source brokered by a DAB instance. DAB sources can be retrieved with the `sources` method. Since the DAB sources are first level nodes (the DAB direct "children"), they can also be retrieved as result of the first call of the `expand` method. By default all the sources brokered by the DAB are included in the discover. If a source is included, its content (depending on the given constraints) will be added to the result set, otherwise it is ignored and its content is not added to the result set. At least one source must be included in the discover so the exclusion of all the sources is equivalent to include them all.

The content of the sources brokered by the DAB can be "distributed" or "harvested":

- **distributed**: the content of the source is retrieved "on demand" during the discover process. The response time is affected by the source latency and several features (such as term frequency) are not available for distributed content. The retrieved content is always up to date.
- **harvested**: the content of the source is permanently stored in database accessed by the DAB. The response time of a discover is much faster than for harvested sources and several features (such as term frequency) are available only for harvested content. Not all the sources content can be harvested, and (depending on the harvesting frequency) then content can be out of date.

Methods

The DABSource class inherits all methods from super-class GINode and implements the following.

contentType ()

Returns the content type of this source.

Returns

Returns the content type of this source; possible values are "distributed" and "harvested".

include (include)

Includes or excludes this source from/in the discover.

Parameters

- `include : Boolean`

True if include the source, false otherwise.

GIAPI

This object provides some utility methods and a single global variable for all the objects of this API.

Methods

clone (object) static

Creates a clone of the given object.

Parameters

- `object : Object`

  The object to clone.

Returns

The cloned object.

emptyResultSet ([error]) static

Creates an empty ResultSet.

Parameters

- `error : Object (optional)`

  An optional error message.

Returns

An empty ResultSet.

formatWords (string) static

Formats all the words in the given string by capitalizing the first character of each word and setting the lower case to the remaining characters.

Parameters

- `string : String`

  The words to format.

Returns

The formatted string.

isoDateTime () static

Returns the current time in the ISO8601 format YYYY-MM-DDThh:mm:ss.mls.

Returns

The current time in the ISO8601 format YYYY-MM-DDThh:mm:ss.mls.

mergeConstraints (constraints1 , constraints2) static

Merges the given basic constraints and returns the merged ones.

Parameters

- `constraints1 : Object`

  The first constraints object to merge.
- `constraints2 : Object`

  The second constraints object to merge.

Returns

The merged constraints.

random () static

Returns a string of 16 random alphanumeric characters.

Returns

A string of 16 random alphanumeric characters.

spaces () static

Returns a string of count characters.

Returns

The formatted string.

thousandsSeparator (integer) static

Formats the given integer number inserting a '.' to separate the thousands.

Parameters

- `integer : String/Integer`

  Integer number.

Returns

The formatted number as a string.

SourceStatus

This class represents the status of a DAB source during a discover.

Methods

id ( ) : String

Retrieves the report id of this source.

Returns

A String with the report id of this source.

message () : String

Retrieves a message describing the current activity of this source.

Returns

A message describing the current activity of this source.

phase ([dataURL=false]) : String

Retrieves the name of phase of completion (default) or a data URL of a correspondent predefined icon.

Parameters

- `dataURL : Boolean (optional)`

  If omitted or set to false returns the name of the phase of completion, otherwise returns the data URL of a correspondent predefined icons.

Returns

A String with the name of the phase of completion or a data URL of a correspondent predefined icon.

progress () : Integer

Retrieves the percentage of completion of this source.

Returns

The percentage of completion of this source.

title () : String

Retrieves the report title of this source.

Returns

A String with the report title of this source.

SpatialRelation

Enumeration of available Spatial Relations. Possible values are:

- **CONTAINS**

- **DISJOINT**
- **OVERLAPS**

2.1.2 AccessOptions

This module provides a set of objects related to the options of the GINode `accessLink` method.

AccessOptions

This object provides a set of parameters to use as options with the GINode `accessLink` method.

AxisSize

This object represent the Axis size.

2.1.3 Concept

This module contains the Concept object and other related objects.

Concept

This object represents a concept from a Controlled Vocabulary.

Methods

description ( ) : Array

Retrieves the descriptions of this concept.

Returns

An array of strings, each string is a description of this concept in different languages.

extend (onResponse, [relation], [options]) async

Retrieves concepts linked to this concept according to one or more relations from the remote Controlled Vocabulary. The size of the extension, that is the number of concepts resulting from the extension, can be retrieved using the method `extensionSize`.

Parameters

- `onResponse : Function`

  Callback function for receiving asynchronous query response. The function has the following arguments:

  - `result : Array`

    The array of linked concepts.
  - `error : String (optional)`

In case of error, this argument contains a message which describes the problem occurred.

- `relation : Relation/String[] (optional)`

  An array of relations or relations values used to retrieve the linked concepts. If omitted, all available relations are used.
- `options : Object (optional)`

  Object literal of available options. The options are:
  - `start : Integer (optional)`

    The start index of the first returned concept. Default value: 1.
  - `count : Integer (optional)`

    The maximum number of concepts to return. Default value: 10.

extensionSize ([relation]) : Integer

Retrieves the number of concepts resulting from the extension of this concept with the specified relation.

Parameters

- `relation : Relation (optional)`

  The relation used to extend the concept. If no relation is provided, all relations are used to calculate the extension size.

Returns

The number of concepts resulting from the extension of this concept with the specified relation.

label () : String[]

Retrieves the labels of this concept.

Returns

An array of strings. Each string is a label of this concept in different languages.

property (onResponse, name) async

Retrieves the Property of this Concept with the specified name.

Parameters

- `onResponse  : Function`

  Callback function for receiving asynchronous query response. The function has the following arguments:
  - `property : Property`

    The Property with the specified name, or null in case of error or if no Property exists with the specified name.
  - `error : String (optional)`

In case of error, this argument contains a message which describes the problem occurred.

- `name : String`

  The name of the property. See also `PropertyName`.

rootRelation () : Relation

Retrieves the relation used to find this concept from the origin concept.

Returns

The relation which links this concept to the origin concept.

sourceThesaurus () : String

Retrieves the labels of this concept.

Returns

The name of the Controlled Vocabulary.

stringify () : String

Retrieves the name of the Controlled Vocabulary that originated this concept.

Returns

The string representation of this concept.

uri () : String

Retrieves the URI of this concept.

Returns

The URI string of this concept.

Property

This objects represents concept properties. The properties are name and values.

PropertyName

Enumeration of well known concept property names. This names can be used to retrieve concept properties with the Concept property method.

Relation

This class provides possible relations which can be used to extend a discover or to extend a concept. The properties are:

- BROAD_MATCH: more general relation between two concepts belonging to different Controlled Vocabularies.

- BROADER: more general relation between two concepts belonging to the same Controlled Vocabulary.
- CLOSE_MATCH: this relation indicates that two concepts belonging to different Controlled Vocabularies are similar and, in some applications,they can be used interchangeably.
- EXACT_MATCH: this relation is used is used to link two concepts, indicating a high degree of confidence that these can be used interchangeably across a wide range of information retrieval applications.
- NARROW_MATCH: more specific relation between two concepts belonging to different Controlled Vocabularies.
- NARROWER: more specific relation between two concepts belonging to the same Controlled Vocabulary.
- RELATED: this relation is used to assert an associative link between two concepts belonging to the same Controlled Vocabulary.
- RELATED_MATCH: This relation is used to assert an associative link between two concepts belonging to different Controlled Vocabularies.
- NONE: no relation.

Methods

decode (value)

Returns the Relation instance having the provided value. An Exception is thrown if value is unknown.

Parameters

- `value : String`

  The Relation value:

Returns

The Relation instance having the provided value.

2.1.4 Refinement

This module allows to to refine the content of a Result Set by means of the Refiner and TermFrequency objects.

Refiner

This object is optionally provided as property of a Result Set. A new instance is created every time a discover is performed, and it is maintained with its chronology, until a new discover is performed. The Refiner is initialized as follows:

1. The cursor value is 0.
2. The chronology.length is 1 and contains the constraints and the options of the origin discover.

The Refiner provides the following main features:

- allows to refine the related result set by merging the given constraints with the constraints of the origin discover (see point 2 above);

- keeps track of the refinements (both constraints and options) in a chronology that can be scrolled back and ahead.

Methods

browse (onResponse, who, [options]) async

Browses the content of the composed node having the given who identifier. The correspondent result set is the same that could be retrieved by calling the `expand` method on the composed node having the given who identifier. However the Refiner allows to take advantage of the chronology.

If who corresponds to a non existent node or to a Report/simple:property node, the correspondent result set is empty.

Parameters

- `onResponse : Function`

  Callback function for receiving asynchronous query response.
- `who : String`

  Identifier of the composed node.
- `options : Object (optional)`

  All the options are allowed except start, termFrequency and extension.

chronology () : Object[]

Returns the chronology of the constraints used to refine the related result set.

Returns

An array of basic constraints.

cursor () : Integer

Returns the chronology cursor, an integer value indicating the current constraints of the chronology. The value satisfies the following statement: cursor >= 0 && cursor <= history.length.

Returns

An integer value indicating the current constraint of the chronology.

forward ([onResponse]) : Boolean async

Scrolls ahead the chronology. Increases the cursor by one and restores the chronology to the current cursor value. The method is executed only if cursor is less than `chronology.length` and onResponse is provided.

Parameters

- `onResponse : Function (optional)`

Callback function for receiving asynchronous query response; if omitted the method is only tested but not executed.

Returns

True if the cursor is less than chronology.length, False otherwise.

rewind ([onResponse]) : Boolean async

Scrolls back the chronology. Decreases the cursor by one and restores the chronology to the current cursor value. The method is executed only if cursor is greater than 0 and onResponse is provided.

Parameters

- `onResponse : Function (optional)`

  Callback function for receiving asynchronous query response; if omitted the method is only tested but not executed.

Returns

True if the cursor is greater than 0, False otherwise.

refine (onResponse, constraints, [options]) async

Refines the related result set:

1. merges the given constraints with the original discover constraints
   a. if cursor is at the end of the chronology (its value is chronology.length - 1), it appends the merged constraints to the chronology
   b. if cursor is less than chronology.length - 1 (due to one or more calls to the `rewind/forward` methods), it overwrites the chronology at index cursor + 1 with the merged constraints and clears the remaining part of the chronology
2. increases the cursor by 1
3. refines the related result set with the merged constraints and the given options

Only basic constraints are supported.

Parameters

- `onResponse : function`

  Callback function for receiving asynchronous query response.
- `constraints : String`

  The constraints to refine the discover. Only basic constraints are supported.
- `options : Object (optional)`

  All the options are allowed except start, termFrequency and extension.

restore ([onResponse], cursor) : Boolean async

Refines the the related result set by restoring the constraints and the options in the chronology at the specified cursor. The method is executed only if `cursor >= 0 && cursor <= chronology.length` and onResponse is provided.

Parameters

- `onResponse : Function (optional)`

  Callback function for receiving asynchronous query response; if omitted the method is only tested but not executed.
- `cursor : Integer`

  An integer value indicating the constraints in the chronology to restore.

Returns

True if cursor >= 0 && cursor <= chronology.length , False otherwise.

reset ()

Resets the Refiner to its initial state. The cursor value is set to 0,the chronology.length is set to 1 and contains the constraints and the options of the origin discover.

TermFrequency

This object is optionally provided as property of a result set. It allows to refine the related result set using as constraints one or more term frequency items related to a term frequency target. The available term frequency targets (valid for all methods parameters) are:

- **keyword**: frequency at which the keyword field items appear in a node report
- **format**: frequency at which the format field items appear in a node report
- **protocol**: frequency at which the protocol field items appear in a node report
- **source**: frequency at which the nodes of the result set own to a source

Methods

checkedItems (target) : TermFrequencyItem[]

Return an array of all the checked term frequency items related to the given term frequency target. See also `checkItems` method.

Parameters

- `target : String`

  A term frequency target.

Returns

An array of checked term frequency items.

checkItems (target, targetItems)

Checks the term frequency items of the given term frequency target. Once the items are checked, the `refine` method is ready to be called. See also `refine` method.

Parameters

- `target : String`

  A term frequency target.
- `targetItems : TermFrequencyItem[]`

clearCheckedItems (target)

Clear all the checked term frequency items related to the given term frequency target. See also `checkItems` method.

Parameters

- `target : String`

  A term frequency target.

isCheckedItem (target, item)

Return true if the given term frequency items related to the given term frequency target is checked. See also `checkItems` method.

Parameters

- `target : String`

  A term frequency target.
- `item : TermFrequencyItem`

  The term frequency item to check.

items (target) : TermFrequencyItem[]

Returns an array of all the available term frequency items related to the given target.

Parameters

- `target : String`

  A term frequency target.

Returns

An array of all the available term frequency items related to the given target.

itemsCount (target) : Integer

Returns the number of all the available term frequency items related to the given target.

Parameters

- `target : String`

  A term frequency target.

Returns

The number of all the available term frequency items related to the given target.

.

refine ()

Refines the related result set by adding constraints basing on the currently checked term frequency items. See also `checkItems` and `items` method.

targets () : String[]

Return an array of all the available term frequency targets. See also `termFrequency` option

Returns

An array of all the available term frequency targets.

targetsCount () : Integer

Returns the number of all the available term frequency targets. See also `targets` method.

Returns

The number of all the available term frequency targets.

TermFrequencyItem

This object provides the frequency of a term derived from a term frequency target. It has the following properties:

- **decodedTerm**: the term (decoded) of the term of a term frequency target
- **freq**: the frequency at which term appears in the nodes report
- **term**: the term of the term of a term frequency target

2.1.5 Report

This module contains the Report object and other objects that compose it.

Bbox

This object provides EPSG:4326 lat\lon coordinates of a rectangular region. It has the following properties:

- **east**: Double value in between -180 and 180
- **north**: Double value in between -90 and 90
- **west**: Double value in between -180 and 180
- **south**: Double value in between -90 and 90

ContactInfo

This object provides info about a contact. It has the following properties:

- **city**
- **email**

- **individualName**
- **link**
- **linkDescription**
- **linkName**
- **organizationName**
- **positionName**
- **url**

OnlineInfo

This object provides info about an online resource. It is composed by the following properties:

- **accessType**
- **description**
- **function**
- **name**
- **protocol**
- **url**

Report

This object provides several info about the associated node. The properties listed below are not described in detail, they are mainly derived from ISO 19115 specification:

- **alternateTitle**
- **author**
- **contributor**
- **coverageAttribute**
- **created**
- **dataAuthority**
- **dataIdentifiers**
- **description**
- **format**
- **geossCategory**
- **id**
- **keyword**
- **online**
- **overview**
- **parentId**
- **rights**
- **service**
- **spatialRepresentationType**
- **title**

- **topic**
- **type**
- **updated**
- **when**
- **where**

ServiceInfo

This object provides info about a service. The properties are:

- **description**
- **operation**
- **source**
- **title**
- **type**
- **version**

ServiceOperation

This object provides info about services operation. The properties are:

- **binding**
- **name**
- **operations**

TimePeriod

This object provides info about a time period expressed in ISO8601 (the time part is optional). The main properties are:

- **from**
- **to**

2.1.6 ResultSet

This module contains the ResultSet and ResultSetExtension objects, and also contains objects to retrieve and refine its content. Furthermore the module is also a rollup of the following modules: Refinement (Section 2.1.4) and Retrieval (Section 2.1.7)**.**

Page

See Section 2.1.7.

Paginator

See Section 2.1.7.

Refiner

See Section 2.1.4.

ResultSet

This object is provided as result of a discover or expand/expandNext operation and has several information such as the number of the returned nodes (the size of the result set) and the number of pages with which the result set is split.

The result set nodes can be retrieved depending on the executed operation by means of one of the following properties:

- **discover**: by means of the Paginator object in the paginator property
- **expand/expandNext**: by means of the Page object in the page property

In case of error or timeout, none of the above properties is provided.

The refinement allows to decrease the size of the result set adding constraints to the last performed discover in a simplified way. A result set of a discover can be refined by means of one of the following properties:

- **termFrequency**
- **refiner**

The complete list of ResultSet follows:

- **extension**: This property is set only if the discover extension has produced more than one result set.
- **pageCount**: Number of pages of this result set.
- **pageIndex**: Index of the current page between 1 and pageCount.
- **pageSize**: Size of this result set pages. This value depends on the discover pageSize option
- **paginator**: The Paginator resulting from a discover operation. In case of error or timeout this property is omitted.
- **refiner**: The Refiner object.
- **size**: Number of nodes of this result set.
- **start**:The start index of the current node between 1 and size. After a DAB discover query, this index depends from the discover start option. After a call of Paginator methods `next` or `prev` this index increments or decrements of a value equals to pageSize.
- **termFrequency**: The TermFrequency object.

ResultSetExtension

This object provides several info about the result of a DAB discover extension. It has the following properties:

- **label**: The label of the concept related to this result set (see discover `extension` for more info).
- **language**: The label language.
- **resultSetIndex**: The index of this result set between 0 and width – 1.
- **scheme**: URI of the Controlled Vocabulary which contains the concept related to this result set.
- **size**: The sum of the size of all the produced result set.
- **uri**: The URI of the concept related to this result set (see discover `extension` for more info).

- **width**: Number of paginators produced by the discover extension.

TermFrequency

See Section 2.1.4.

TermFrequencyItem

See Section 2.1.4.

2.1.7 Retrieval

This module allows to to retrieve the content of result set by means of the Paginator and Page objects.

Page

A result set page of nodes.

Methods

count () : Integer

Retrieves the number of nodes of this page.

Returns

An integer with the number of nodes contained in this page.

hasNext () : Boolean

Tests if one or more node is available.

Returns

True if the next call of the next method will return a GINode, false otherwise (see also `reset` method).

next () : GINode

Retrieves the next node from this page.

Returns

The next node from this page or null if the last has already been returned (see also methods `hasNext` and `reset`).

nodes () : GINode[]

Retrieves the nodes of this page.

Returns

Array of GINode of this page (an empty array for an empty page).

reset ()

Reset this page to its original state, as if the `next` method had never been called.

size () : Integer

Retrieves the maximum number of nodes that this page can contain.

Returns

An integer with the maximum number of nodes that this page can contain.

Paginator

Objects of this class can be used to retrieve the nodes of the related result set.

Methods

first (onResponse, [execute]) : Boolean async

Tests if the current page is not the first of the Paginator/resultSet:method. If execute is True and the test is positive, retrieves the first page from the Paginator/resultSet:method. An Exception is thrown if the last call to one of the methods `next`, `prev`, `skip` or `last` is still running.

Parameters

- `onResponse : Function`

  Callback function for receiving asynchronous query response. It has the following parameter:

  - `result : Array`

    This array contains a reference to this paginator.
- `execute : Boolean (optional)`

  If omitted or false, the method only executes the test. If true and the test is positive, retrieves the first page from the Paginator/resultSet:method.

Returns

True if the current page is not the first of the Paginator/resultSet:method, False otherwise.

last (onResponse, [execute]) : Boolean async

Tests if the current page is not the last of the Paginator/resultSet:method. If execute is true and the test is positive, retrieves the last page from the Paginator/resultSet:method. An Exception is thrown if the last call to one of the methods `next`, `prev`, `skip` or `first` is still running.

Parameters

- `onResponse : Function`

  Callback function for receiving asynchronous query response. It has the following parameter:

  - `result : Array`

    This array contains a reference to this paginator.
- `execute : Boolean (optional)`

  If omitted or false, the method only executes the test. If true and the test is positive, retrieves the last page from the Paginator/resultSet:method.

Returns

True if the current page is not the last of the Paginator/resultSet:method, False otherwise.

next (onResponse, [execute]) : Boolean async

Tests if subsequent pages in the Paginator/resultSet:method are available. If execute is true and the test is positive, retrieves the next page from the Paginator/resultSet:method. An Exception is thrown if the last call to one of the methods `next`, `prev`, `skip` or `last` is still running.

Parameters

- `onResponse : Function`

  Callback function for receiving asynchronous query response. It has the following parameter:

  - `result : Array`

    This array contains a reference to this paginator.
- `execute : Boolean (optional)`

  If omitted or false, the method only executes the test. If true and the test is positive, retrieves the next page from the Paginator/resultSet:method.

Returns

True if subsequent pages are available, False otherwise.

page () : Page

Retrieves the current Paginator/ResultSet.

Returns

The current Paginator/resultSet.

prev (onResponse, [execute]) : Boolean async

Tests if previous pages in the Paginator/resultSet:method are available. If execute is true and the test is positive, retrieves the previous page from the Paginator/resultSet:method. An Exception is thrown if the last call to one of the methods `next`, `prev`, `skip` or `last` is still running.

Parameters

- `onResponse : Function`

  Callback function for receiving asynchronous query response. It has the following parameter:

  - `result : Array`

    This array contains a reference to this paginator.

- `execute : Boolean (optional)`

  If omitted or false, the method only executes the test. If true and the test is positive, retrieves the previous page from the Paginator/resultSet:method.

Returns

True if previous pages are available, False otherwise.

skip (onResponse, newPageIndex, [execute]) : Boolean async

Tests if newPageIndex satisfies the following statement:

```
newPageIndex != currentPageIndex && newPageIndex >= 1 && newPageIndex
                                  <= pageCount
```

If execute is true and the test is positive, retrieves the *newPageIndex-th* page from the Paginator/resultSet:method. An Exception is thrown if the last call to one of the methods, `next`, `prev`, `last` or `first` is still running.

Parameters

- `onResponse : Function`

  Callback function for receiving asynchronous query response. It has the following parameter:

  - `result : Array`

    This array contains a reference to this paginator.

- `newPageIndex : Integer`

  The index in the Paginator/resultSet:method of the page to retrieve.

- `execute : Boolean (optional)`

  If omitted or false, the method only executes the test. If true and the test is positive, retrieves the `newPageIndex-th` page from the Paginator/resultSet:method.

Returns

True if `newPageIndex` satisfies the above statement, False otherwise.

2.2 GEO DAB RESTful API

This RESTful APIs (Application Program Interface) provides simple search capabilities and resources encoded in JSON, which simplify the development of applications and clients making use of the GEO

DAB. RESTful API gives direct access to resources supporting the Create-Retrieve-Update-Delete (CRUD) pattern.

The main resources exposed by the GEO DAB RESTful API are:

- **Sources**
- **Progress**
- **View**
- **Datasets**

RESTful API requires an *API key* for some configuration, such as for example views management.

2.2.1 Sources

Retrieves the brokered GEO DAB sources. Each source is connected to a data service and provides datasets available through the `/datasets` and `/{apiKey}/{viewId}/datasets` operations. By default the datasets search includes all the sources. You can use the *`sources`* parameter in the `/datasets` and `/{apiKey}/{viewId}/datasets` operations body to include only the sources with the given identifiers.

Parameters: `/sources`

Path: `/sources`

HTTP METHOD: `GET`

2.2.2 Progress

Retrieves the current progress of the datasets search with the given identifier (see `searchId` parameter of the `/datasets` operation body). The returned array contains a `SearchProgress` object for each DAB source, which provides info about the progress of the related sources, and a `SearchProgress` object for the DAB, which shows the overall progress.

Parameters

- `searchId : String`

  The search identifier.

Path: `/progress/{searchId}`

HTTP METHOD: `GET`

2.2.3 View

Manage the GEO DAB views. This object has the following methods.

Create View

Creates a GEO DAB view. A view defines a `ResultSet` by means of a fixed set of constraints. The identifier of the created view can be used along with the `apiKey` in the `/{apiKey}/{viewId}/datasets` operation.

Parameters

- `apiKey : String`

  The API key.
- `body : Object`

  A set of the following parameters.
  - `label : String`

    A descriptive label for the view.
  - `sources : String[]`

    Array of sources identifiers. Only the content of the correspondent sources (matching the other view constraints if any) is included in the ResultSet.
  - `keywords : String[]`

    Array of Keywords.
  - `protocols : String[]`

    Array of transfer protocols names. Use the {apiKey}/views/protocols operation for a list of availble protocols.
  - `where : Object`

    A bounding box (expressed in EPSG:4326) which includes the search results.
  - `when : Object`

    A time period which includes the discover results. Date or datetime expressed in ISO8601 format.
  - `accessParameters : String[]`

    Comma separated list of format (expressed by its mime type), transfer protocol name and CRS (e,g.: image/png,OGC Web Coverage Service 1.0.0 Protocol,EPSG:4326). Only the datasets that can be downloaded with the specified parameters are included in the ResultSet.

Path: `/{apiKey}/views`

HTTP METHOD: `POST`

Retrieve View

Retrieves the GEO DAB views associated to the given `apiKey`.

Parameters

- `apiKey : String`

  The API key.

Path: `/{apiKey}/views`

HTTP METHOD: `GET`

Delete View

Deletes the DAB view with the the given `viewId`.

Parameters

- `apiKey : String`

  The API key.
- `viewId : String`

  The view Identifier.

Path: `/{apiKey}/views/{viewId}`

HTTP METHOD: `DELETE`

Retrieve protocols

Retrieves the available transfer protocols. The names can be used as view constraints in the `protocols` field and as parameter in the `accessParameters` field.

Path: `/views/protocols`

HTTP METHOD: `GET`

2.2.4 Datasets report

Retrieves datasets reports. It has the following methods.

Retrieve dataset report by ID

Retrieves the dataset report with the given dentifier.

Parameters

- `id : String`

  The dataset identifier.

Path: `/datasets/{id}/report`

HTTP METHOD: `GET`

Retrieve datasets reports with optional constraints and options

Set the given optional constraints and options and starts the search of datasets reports from the GEO DAB sources. The body parameter can be left empty or set as a plain empty object; in such case the default parameters and options are submitted.

Parameters

- `body : Object`

  A set of the following parameters.
  - `start : Integer`

    The start index of the first ResultSet report. Default value: 1.
  - `what : String`

One or more search terms; in case of multiple search terms they must be separated by 'AND' or 'OR'. E.g.: 'water AND fire', 'sea OR sun'.

- `who : String[]`

  Array of datasets identifiers. Only the content of the correspondent collection datasets (matching the given constraints if any) is searched. In case of simple datasets (which have no content), they are always added to the ResultSet regardless of the above constraints (if any).

- `sources : String[]`

  Array of sources identifiers. Only the content of the correspondent sources (matching the given constraints if any) is searched.

- `where : Object`

  A bounding box (expressed in EPSG:4326) which includes the search results.

- `when : Object`

  A time period which includes the discover results. Date or datetime expressed in ISO8601 format.

- `searchId : String`

  Search identifier; use it to get the search status.

- `pageSize : Integer`

  The size of the ResultSet pages. Determines the maximum number of Report objects in the ResultSet. Default value: 10.

- `timeout : Milliseconds`

  Request timeout.

- `searchFields : String[]`

  A single search field or a comma separated list of search fields in which to perform the terms search. Possible search fields are: title, keyword, description, anytext.

- `spatialRelation : String`

  The spatial relation applied to the bounding box of the 'where' constraint.

- `termFrequency : String[]`

  One or more comma separated list of term frequency target; by default all are selected.

- `extensionRelation : String`

  This option allows to retrieve multiple ResultSet and requires a single search term in the 'what' parameter.

Path: `/datasets/report`

HTTP METHOD: `POST`

Retrieve datasets reports with View

Like the `/datasets/report` operation but in addition to the optional constraints. The search always applies the constraints of the view with the given identifier.

Parameters

- `viewId : String`

  The View identifier.

- `body : Object`

  A set of the following parameters.

  - `start : Integer`

    The start index of the first ResultSet report. Default value: 1.

  - `what : String`

    One or more search terms; in case of multiple search terms they must be separated by 'AND' or 'OR'. E.g.: 'water AND fire', 'sea OR sun'.

  - `who : String[]`

    Array of datasets identifiers. Only the content of the correspondent collection datasets (matching the given constraints if any) is searched. In case of simple datasets (which have no content), they are always added to the ResultSet regardless of the above constraints (if any).

  - `sources : String[]`

    Array of sources identifiers. Only the content of the correspondent sources (matching the given constraints if any) is searched.

  - `where : Object`

    A bounding box (expressed in EPSG:4326) which includes the search results.

  - `when : Object`

    A time period which includes the discover results. Date or datetime expressed in ISO8601 format.

  - `searchId : String`

    Search identifier; use it to get the search status.

  - `pageSize : Integer`

    The size of the ResultSet pages. Determines the maximum number of Report objects in the ResultSet. Default value: 10.

  - `timeout : Milliseconds`

    Request timeout.

  - `searchFields : String[]`

    A single search field or a comma separated list of search fields in which to perform the terms search. Possible search fields are: title, keyword, description, anytext.

  - `spatialRelation : String`

    The spatial relation applied to the bounding box of the 'where' constraint.

  - `termFrequency : String[]`

    One or more comma separated list of term frequency target; by default all are selected.

  - `extensionRelation : String`

    This option allows to retrieve multiple ResultSet and requires a single search term in the 'what' parameter.

Path: `/{viewId}/datasets/report`

HTTP METHOD: `POST`

2.2.5 Datasets content

Retrieves the given dataset content. It has the following methods.

Retrieve available content options

Retrieves available options for the content of the dataset with the given identifier. Each option provides a valid combination of parameters to use for the `/dataset/{id}/content` operation.

Parameters

- `id : String`

  The Dataset identifier.

Path: `/datasets/{id}/content/options`

HTTP METHOD: `GET`

Retrieve the given dataset content

Retrieves the dataset content with the given identifier using the options available by means of the `/dataset/{id}/content/options` operation.

Parameters

- `id : String`

  The Dataset identifier.
- `format : String`

  The data format of the output data.
- `subset : String`

  Spatial extent (lower-corner, upper-corner) of the output data expressed with the subsetCRS. The lower-corner and upper-corner order is according to the subsetCRS
- `subsetCRS : String`

  The coordinate reference system of the subset bounding box.
- `temporalBegin : String`

  Start time of the temporal extent expressed in ISO8601.
- `temporalEnd : String`

  End time of the temporal extent expressed in ISO8601
- `size : String`

  Number of samples along the first and second spatial axis. The order is according to the CRS.
- `CRS : String`

  The coordinate reference system of the output data. Values of the size parameter are according to this CRS.

Path: `/datasets/{id}/content`

HTTP METHOD: `GET`

2.3 GEO DAB Custom Download

The GEO DAB Asynchronous Download provides a mechanism to process and transform data in order to allow DAB users and clients to customize their own downloads. Currently this feature is available for a subset of all DAB datasets. The datasets that can be downloaded with asynchronous download contains an augmented metadata element named *advancedAccessLink*.

The *advancedAccessLink* is used to perform advanced data download and represents the base endpoint link to PATH `/{viewId}/gwps/dataset/{id}` where *viewId* is the DAB View identifier and *id* is the univocal identifier related to the dataset.

To use the *advancedAccessLink* and support the Asynchronous Download, GEO DAB extends the Web Processing Service (WPS) interface creating two new processes: “gi-axe-capabilities” and “gi-axe-transform”.

gi-axe-capabilities

Retrieves all available common grids capabilities (advanced options) for downloading the related data. The capabilities request is built using the *adavancedAccessLink* plus the following default KVP values:

service=WPS&request=execute&identifier=gi-axe-capabilities

DAB clients send Gi-axe-capabilities requests and submit them through HTTP GET method to the WPS service endpoint. GEO DAB replies with a WPS execute response XML document containing the most relevant information inside the *capabilityCubes* complex element which describes the available download options. The main options are described below.

Options

- `format : String[]`

  A list of supported formats for downloading the dataset. To select the chosen format, the “outputFormat” download option is used in *gi-axe-transform* operation.
- `spatialGrid : Object`

  Determines the available dataset grid. There can be more than one grid for each resource.
  - `spatialAxis : Object`

    Usually the spatial grid is defined using two spatial axis, ordered according to the CRS. SpatialAxis object contains the following fields:
    - `abbreviation : String`

      Short axis name.
    - `label: String`

      Descriptive axis name.
    - `units: String`

      The axis units (e.g. DEGREES)
    - `direction: String`

      The axis direction (e.g. NORTH)
    - `numberOfPoints: Integer , Integer`

The native grid axis points (to be used as the "outputSize" download option in *gi-axe-transform* operation)

- `defaultExtent : Object`

  The coverage extent to be used by default. It has two main elements: *lowerCorner* and *upperCorner*. These elements are characterized by the following fields:

  - `crs : Object`

    The Coordinate Reference System related to the upper or lower corner are identified by an authority (e.g. EPSG) and a code (e.g. 4326). To select the chosen CRS the "outputSubsetCRS" download option is used in *gi-axe-transform* operation.
  - `coordinates: Double`

    Coordinates representing the upper or lower corner are ordered according to the specified crs. To select the chosen CRS the "outputSubset" download option is used in *gi-axe-transform* operation

- `totalExtent : Object`

  The complete coverage source extent. It has the same elements of defaultExtent described above.

Path:
`/{viewId}/gwps/dataset/{id}?service=WPS&request=execute&identifier=gi-axe-capabilities`

HTTP METHOD: *GET*

gi-axe-transform

Retrieves the desired data by specifying a set of access parameters used in the transform request.

The transform request is built using the *adavancedAccessLink* plus the following default KVP values:

service=WPS&request=execute&identifier=gi-axe-transform& storeexecuteresponse=true

Then the transform request is finalized postfixing the DataInputs parameter composed by the following set of main access parameters (extracted through the *gi-axe-capabilities* operation):

Parameters

- `outputFormat : String`

  The desired format of data (e.g. outputFormat=IMAGE_PNG).
- `outputSize : Integer , Integer`

  The desired size in terms of width and height of data (e.g. outputSize=100, 200).
- `outputSubsetCRS : String`

  The desired CRS of data (e.g. outputSubsetCRS=EPSG:4326).
- `outputSubset : Double , Double , Double , Double`

  The desired coordinates of data expressed in south-west-north-east order (e.g. outputSubset=-90,-180,90,180).

The path request is reported below. The download options for DataInputs are KVP separated by “;” and they are URL encoded. The *FORMAT*, *SIZE*, *CRS* and *SUBSET* are placeholders that must be replaced with the real options of data.

The GEO DAB HTTP GET response contains a link to the status message for the download; in fact the *gi-axe-transform* operation is executed asynchronously in order to allow clients to continue with other operations. When the transformation is completed, the desired data is ready to be downloaded and the status message will contain the direct link to the requested data.

Path:
```
/{viewId}/gwps/dataset/{id}?service=WPS&request=execute&identifier=gi
-axe-transform&storeexecuteresponse=true&
DataInputs=outputFormat%3DFORMAT%3BoutputSize%3DSIZE%3BoutputSubsetCR
S%CRS%3BoutputSubset%3DSUBSET
```

HTTP METHOD: `GET`

2.4 GEO DAB Bulk Download

The GEO DAB Bulk Download is a facility tool that allow DAB users and DAB clients to download multiple datasets through one single action. GEO DAB users is able to pick-up multiple data during the discovery phase (from several different catalogues and searches) and to download all the chosen data executing one operation only. The result is a ZIP archive containing all the requested data.

GEO DAB extends the Web Processing Service (WPS) interface to support the Bulk Download creating a new process called “bulk-download”.

DAB clients send requests (encoded in XML) and submit them through HTTP POST method to the WPS service endpoint. GEO DAB replies with a WPS response containing a link to the status message for the download. The Bulk Download operation is performed in asynchronously way in order to avoid deadlock (clients are blocked waiting for the response). At the end of the execution (STATUS: Completed) the status message will contain the direct link to the ZIP archive with the requested data.

The HTTP POST request template for Bulk Download with three datasets to download is reported below. The *DATA_URL_1*, *DATA_URL_2* and *DATA_URL_3* are placeholders that must be replaced with the real link to data.

HTTP POST REQUEST TEMPLATE:

```
<?xml version="1.0" encoding="UTF-8"?>
<wps:Execute version="1.0.0" service="WPS"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:wfs="http://www.opengis.net/wfs"
   xmlns:wps="http://www.opengis.net/wps/1.0.0"
xmlns:ows="http://www.opengis.net/ows/1.1"
   xmlns:gml="http://www.opengis.net/gml" xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:wcs="http://www.opengis.net/wcs/1.1.1"
xmlns:xlink="http://www.w3.org/1999/xlink"
   xsi:schemaLocation="http://www.opengis.net/wps/1.0.0
http://schemas.opengis.net/wps/1.0.0/wpsAll.xsd">
   <ows:Identifier>bulk-download</ows:Identifier>
   <wps:DataInputs>
      <wps:Input>
      <ows:Identifier>download</ows:Identifier>
      <ows:Title>Direct download1</ows:Title>
      <wps:Reference xlink:href="DATA_URL_1"></wps:Reference>
    </wps:Input>
    <wps:Input>
      <ows:Identifier>download</ows:Identifier>
      <ows:Title>Direct download2</ows:Title>
      <wps:Reference xlink:href=" DATA_URL_2"></wps:Reference>
    </wps:Input>
    <wps:Input>
      <ows:Identifier>download</ows:Identifier>
      <ows:Title>Custom download3</ows:Title>
      <wps:Reference xlink:href=" DATA_URL_3"></wps:Reference>
    </wps:Input>
  </wps:DataInputs>
  <wps:ResponseForm>
    <wps:ResponseDocument storeExecuteResponse="true">
      <wps:Output>
        <ows:Identifier>Bulk download report</ows:Identifier>
      </wps:Output>
    </wps:ResponseDocument>
  </wps:ResponseForm>
</wps:Execute>
```

The HTTP POST response template is displayed below. The *STATUS_URL* placeholder represents the link to PATH `/{viewId}/gwps/status/{status-identifier}` that contains the status message for the download.

HTTP POST RESPONSE TEMPLATE:

```
<ns2:ExecuteResponse xmlns:ows="http://www.opengis.net/ows/1.1"
xmlns:ns2="http://www.opengis.net/wps/1.0.0" xmlns:ns3="http://www.w3.org/1999/xlink"
statusLocation="STATUS_URL" service="WPS" version="1.0.0"/>
```

The request and response parameters are described below.

Request Parameters

- `ows:Identifier : String`

  The identifier of the operation. The value must be *bulk-download*

- `wps:DataInputs: Object`

  A list of wps:Input element for each dataset to be downloaded. Each wps:Input contains the following parameters

  - `ows:Identifier : String`

    The identifier of the input. The value must be *download* for each input.

  - `ows:Title : String`

    The title of the dataset. The value is used to name the correspondent file in the created ZIP archive.

    - `wps:Reference : String`

      The xlink:href attribute contains the download URL to retrieve the dataset. The URL should be XML escaped.

- `wps:ResponseForm : Object`

  This element contains a wps:ResponseDocument Object with one wps:Output element that has the following parameter

    - `wps:Identifier : String`

      The identifier of the output.

Response Parameter

- `statusLocation : String`

  This attribute element contains the URL to the status message for the download.

Path: `/{viewId}/gwps`

HTTP METHOD: `POST`

2.5 GEO DAB Statistics

The GEO DAB exposes RESTful Statistics API interface. The documentation with examples is available at http://gs-service-production.geodab.eu/gs-service/api/gwpstat/#/.

The main statistics exposed by the Statistics API are:

- **mostPopularResources**: most frequently returned datasets.
- **mostPopularCatalogs**: most frequently returned Data Sources.
- **mostPopularOrg**: most frequently returned Organizations.
- **mostPopularAreas**: most frequently returned Geographical Area.
- **mostPopularKeywords**: most frequently returned keywords.
- **numberOfSearches**: number of queries submitted.

These numbers can be queried by a given time period and with a given resoultion (number of minutes, hours, day, week, month).

Parameters

- `statistic : String`

  The required statistic (e.g. mostPopularCatalogs, mostPopularResources, etc.).

- `period : String`

  Predefined analyzed period (e.g. last year, lastMonth, etc.). If the period is set, the parameters *periodRangeStart* and *periodRangeEnd* must be left blank.

- `periodRangeStart : String`

  Start of analyzed period compliant with ISO8601 (e.g. 2000-01-01T00:00Z). If the period is set, the parameter *period* must be left blank.

- `periodRangeEnd : String`

  End of analyzed period compliant with ISO8601 (e.g. 2020-01-01T00:00Z). This optional parameter can be set only if also the *periodRangeStart* is set
- `interval : String (optional)`

  Interval of the statistics results. Useful only if the selected statistic is *numberOfSearches*, for all other statistics this parameter can be left blank (otherwise is ignored).
- `intervalSize : Integer (optional)`

  Size of the interval expressed by the the *interval* parameter. Useful only if the selected statistic is *numberOfSearches*, for all other statistics this parameter can be left blank (otherwise is ignored).
- `catalog : String (optional)`

  A catalog target of a selected statistic. May be omitted to target all catalogs.
- `maxResult : Integer (optional)`

  Maximum number of presented results, ignored for the *numberOfSearches* statistic.

Path: `/rest/stats/gp`

HTTP METHOD: `GET`

ANNEX A. References

[1] EDGE: European Direction in GEOSS Common Infrastructure Enhancements – Grant.

[2] EDGE-WP3-DEL-D3.4: GEOSS Portal and GEO DAB Enhancements.

[3] EDGE-WP2-DEL-D2.4: Functional and non-functional enhancements specification.

[4] EDGE-WP2-DEL-D2.3: Use Cases Description and User Requirements Document.

ANNEX B. Acronym

API	Application Programming Interface
CNR-IIA	Consiglio Nazionale delle Ricerche – Istituto per l’Inquinamento Atmosferico
CRS	Coordinate Reference System
CSS	Cascading Style Sheets
CSW	Catalog Service for the Web
DEL	Deliverable
EDGE	European Direction in GEOSS Common Infrastructure Enhancements
EO	Earth Observation
EOP	Earth Observation Programme
ESA	European Space Agency
EU	European Union
GCI GEOSS	Common Infrastructure
GEO	Group on Earth Observation
GEO DAB	GEO Discovery and Access Broker
GEOSS	Global Earth Observation System of Systems
GML	Geography Markup Language
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
OGC	Open Geospatial Consortium
PU	Public Usage
REST	Representational state transfer
SBA	Societal Benefit Areas
UTB	User and Technical Board
WCS	Web Coverage Service
WMS	Web Map Service
WP	Work Package
XML	eXtensible Markup Language