Funded by the European Union

Ref. Ares(2024)3924670 - 31/05/2024

DELIVERABLE 5.4

Specification of the open API for accessing the data network

WP5 – Infrastructure for RWD reporting in regulatory / HTA decision making

Delivery date: *31st May 2024*

Due date: *M18*

Deliverable type: *Report (R)*

Dissemination level: *Public (PU)*

Authors

Name: Anni Ahonen-Bishopp (BCP), Fahad Syed (BCP), Tomi Laitinen (BCP)

Lead contributor	6. BCP
Other contributors	-

Document history

VERSION	DATE	AUTHOR	DESCRIPTION
0.1	15 Apr 2024	Anni Ahonen-Bishopp	Establishing the document
0.2	24th Apr 2024	Anni Ahonen-Bishopp	Document outline
0.3	13th May 2024	Anni Ahonen-Bishopp, Fahad Syed	BC internal review version
0.5	19th May 2024	Anni Ahonen-Bishopp, Fahad Syed, Jakub Rogalski	Partner review version
v.final	31st May 2024	Anni Ahonen-Bishopp	Finalized for submission to the EC

Internal review history

REVIEWED BY	DATE	DESCRIPTION
Nea Hellman	22nd May 2024	Reviewing draft version
Mikko Juvonen	27th May 2027	Reviewing draft version

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

Table of contents

Executive summary

Work Package (WP) 5 aims to build Infrastructure for real-world data (RWD) reporting in regulatory/HTA decision-making. The objective contributes to how international data harmonization standards are uniformly applied in European cancer centres to provide a critical RWD data source for federated analysis by health technology assessment (HTA) and regulatory bodies across Europe.

The cornerstone of any HTA process is ready access to relevant clinical data, harmonised and curated for this specific purpose. Once the data source has gone through sufficient curation, the typical point of use is the HTA body of the treatment centre, hospital, or network of health organisations.

The need is to observe HTA variations caused by slight differences in health policies, socioeconomical backgrounds, and standards of care, between hospitals and countries. For this purpose, the RWD collections should be available within a network for federated HTA protocols. This need poses not just problems in making different RWD collections comparable, but technical issues in data security, GDPR compliance, and analytical techniques applied in federated manner. To this end, OMOP CMD was chosen as the common data model to be used by the RWD data providers to fulfil the objective.

The deliverable 5.4 is part of WP5 that specifically addresses the above-mentioned issues. In this deliverable the focus is in resolving the data security and access problems, and ability to execute different analytical tools in a federated manner.

The main goal of this deliverable is the implementation of such application software services that allow multiple OMOP-harmonised RWD collections to reside within the respective IT administrations, whilst being available to respond to specifically described queries about their content, or to execute an analytical tool to generate a response. The implementation must be openly described (open Application Programming Interface, API), and be available for external authorised services, for local RWD partners to build their own applications and interfaces for internal purposes. At the same time the limitations and constraints in access, and availability of data in form of summaries and aggregations must be addressed, to ensure compliance with regulatory requirements.

The use of OMOP CDM (Observational Medical Outcomes Partnership Common Data Model) in this project is essential, for compatibility with other potential data sources beyond the ONCOVALUE project. The principle of relying on OMOP coded data makes it possible to standardise various query and aggregation protocols. Any ONCOVALUE clinical partner, who are able to provide OMOP-coded RWD to the project, is able to utilise these protocols and query remotely the content of their clinical database. These include for example the EMA DARWIN, and any EU-residing oncology hospitals, with the ability and motivation to part-take in an HTA analysis network.

1. List of abbreviations and definitions

Abbreviation	Definition
WP	Work Package
RWD	Real-world data
API	Application programming interface, often part of REST protocol
End-point	A method within an API
REST	Representational State Transfer; agreed communication protocol between computational services
HTA	Health technology assessment
QoL	Quality of life

2. Introduction

This deliverable summarises the API specification created for collaboration between ONCOVALUE clinical partners (listed below in subchapter 2.1). The main functionalities of this API are the ability to remotely request feasibility results from specified partner collections and provide aggregate data for visualisation in the HTA dashboard. HTA dashboard for medical products is a comprehensive tool designed to support decision-making by providing a clear and structured visualization of various data points related to the evaluation of medical product.

2.1 List of clinical partners

Abbreviation	Definition
NKI	NEDERLANDS KANKER INSTITUUT
RIJN	RIJNSTATE (Not participating in the federated network)
HUS	HELSINGIN JA UUDENMAAN SAIRAANHOITOPIIRIN KUNTAYHTYMÄ (HUS)
IRST	ISTITUTO ROMAGNOLO PER LO STUDIO DEI TUMORI DINO AMADORI
IPOP	INSTITUTO PORTUGUES DE ONCOLOGIA DO PORTO
RHP	REGION HOVEDSTADEN

The API is meant to be used either by service accounts, or by authorised individual user accounts. In the service use the API will take care of communication between the ONCOVALUE project portal (BC|RQUEST) and connected partner service instances (BC|LINK) containing the RWD collection data. In ONCOVALUE, the aim is to test the feasibility of implemented product in clinical partner sites. The collections on the partner BC|LINK instances are either line level anonymized patient datasets in OMOP format, or so-called virtual data set containing only data aggregations with filterable attributes.

2.1. Background

The preliminary implementation exists as part of federation technology developed by BC Platforms and has been in distribution as a commercial product. The existing solution has the capability to create machine readable queries using OMOP vocabulary in the research portal, and to execute them remotely in an access-restricted RWD dataset harmonised to OMOP data model. The results are aggregate summaries preventing disclosure of too small patient groups. In addition, the protocol can request a genomic statistical analysis from the remote RWD database, using pre-existing local tools.

2.2. Deliverable content

In this deliverable the expansion to existing protocol is divided in three parts, described in detail in section 3:

- The ability to use aggregated OMOP data files for queries, removing the need to always host line-level patient data in the BC|LINK collection

    - This allows for more complicated combination queries like comorbidity index and various diseases scores to be precalculated either per patient, or in aggregate, and used as covariates or for reporting
- The ability to use the query API of BC|RQUEST remotely, as an authorised end-user, or as a service account
    - Authorised end-user can operate within their allocated access scope, and use the aggregate and statistical results for more complicated downstream analyses
    - An authorised service account can be used to build in-house applications and interfaces using third-party tools, like business intelligence and reporting tools, for extra-project use-cases.
- The expansion of the query engine to handle timeline variables in the patient journey, and in the collection metadata
    - These increments enable more precise definition of primary and secondary events on patient timeline, and generation of ranged follow-up summaries like survival and progression reports
    - The observation period of a collection or cohort can be summarised over selected ranges in time, for example to produce annual reports for data accumulation

3. Application query interface description

As the work is partially built on an already existing communication framework, we have limited our description here only to the parts relevant for the new implementation, or where existing protocol has been directly modified or overwritten.

3.1. Protocol architecture

In Figure 1 flowchart we describe the protocol's order of events, starting with the end-user in the BC|RQUEST portal UI and propagating the query via the communication API to the query engine in BC|LINK. As has been previously implemented, the BC|LINK - hosting the RWD dataset – initiates all parts of the communication, hence leaving participation in the query to be controlled by the BC|LINK software, and – ultimately – the system administrator. This is a necessary safeguard against unsolicited 'push' requests and provides a firewall blockage that allows one-way traffic only.

In the new implementation the communication protocol can carry queries to RWD collections that do not present themselves as OMOP databases, but rather OMOP -derived aggregations. The protocol structure has been altered to accept a summary file as the source for data. This limits the complexity of the queries, as multidimensional questions cannot be answered by aggregations with a limited number of dimensions.

However, as a boon, this enables quick onboarding of data partners' collections, where provision of line-level OMOP harmonised database is not feasible for any reason. It enables partner participation in the HTA analysis network, albeit in limited manner. This protocol has proven useful in standardisation of derived variables as well, which may be very complicated to obtain via conventional database query methods and require almost always some level of curation and pre-calculation. Such variables are

typically used comorbidity indexes like CCI (Charlson et al, 1987), NCI (Klabunde at al, 2000) and various quality of life (QoL) scores.

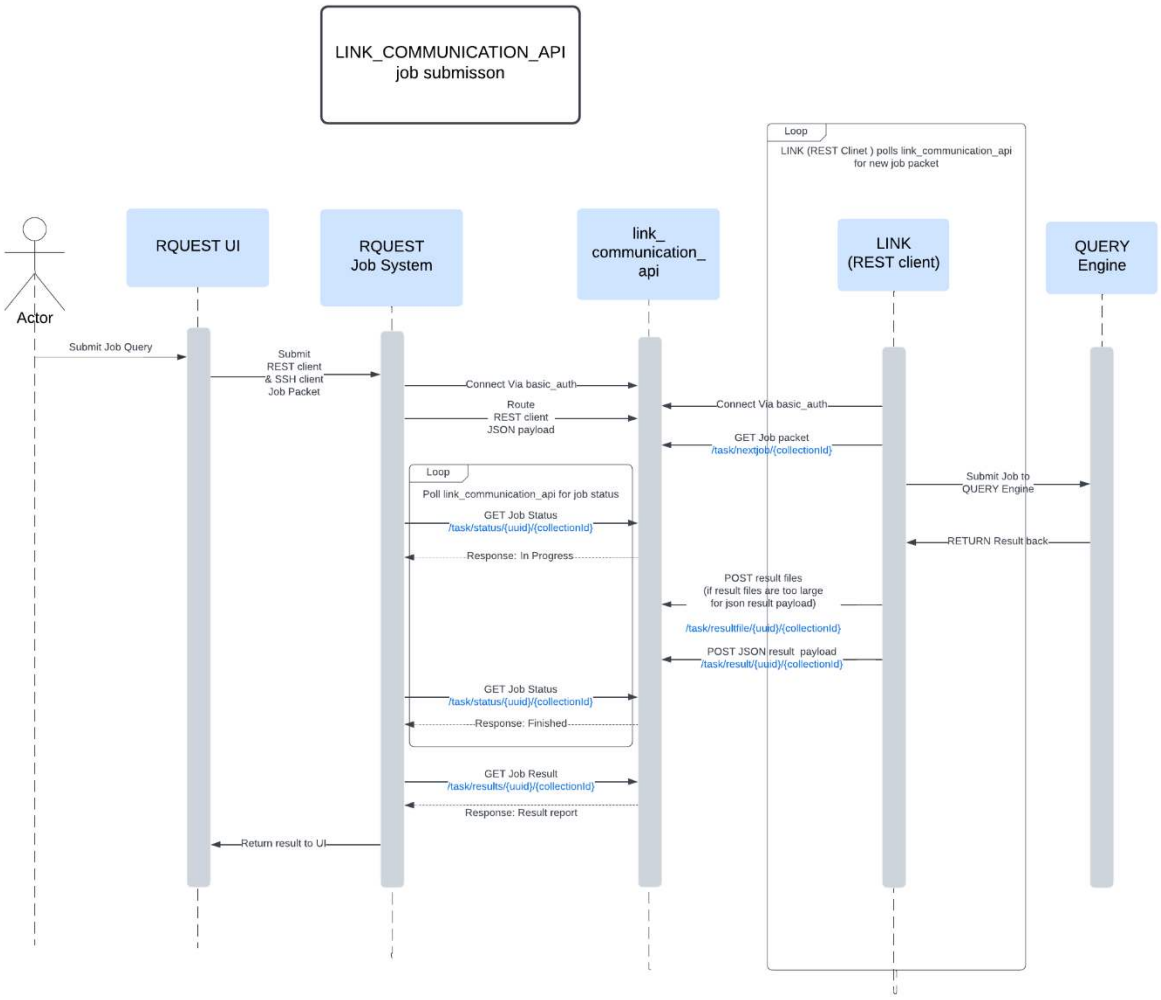

Figure 1. A flowchart of the communication protocol between the ONCOVALUE portal query engine and an HTA collection instance.

Figure 2 showcases the remote query ability enabled by this protocol architecture. In the connection API between BC|RQUEST portal /generating queries by user), and the multiple BC|LINK RWD databases (responding to queries independently), the functionality is transparent to the end-user. The application generates all necessary views to results, and responsiveness of different collections. With the remote querying, however, the end-user is empowered with more precise control over generating the queries and processing the results.

The metadata and aggregations provided as the results can be directly fed into a downstream meta-analysis or reporting pipeline, created by the advanced end-user. This requires some understanding and knowledge of JSON structures and reading files programmatically, but it is a powerful tool for researchers who want to follow data accumulation, or meta-analyse cohorts from different sites. When the remote requests are evoked by an authorised service account (not a human), the system can be used to feed data for internal reporting using business intelligence tools like Power BI, Tableau, etc.

Figure 2. A flow chart and an example of existing API calls for remote query generation within ONCOVALUE portal and HTA collections.

3.2. Request and response protocols

The ONCOVALUE project has a dedicated environment for testing and project demonstration purposes. Current full Swagger documentation is accessible in this environment. Access requires credentials, which can be obtained by project members by sending a request for BC Platforms ONCOVALUE project manager.

- https://rquest-oncovalue.bcplatforms.cloud/link_connector_api/swagger-ui/index.html

The Swagger API document application in the above URL provides also examples of the request and response -formats and allows testing directly against the REST API.

In addition, the endpoints are provided as Appendices A and B for this report as PDFs. For convenience some screenshots are attached here, to describe the main structure of the API request and response

protocols. Figure 3 shows the generic structure that describes an endpoint, the request object structure, and the expected response type from the portal. Figure 4 is a closeup of the request body structure.

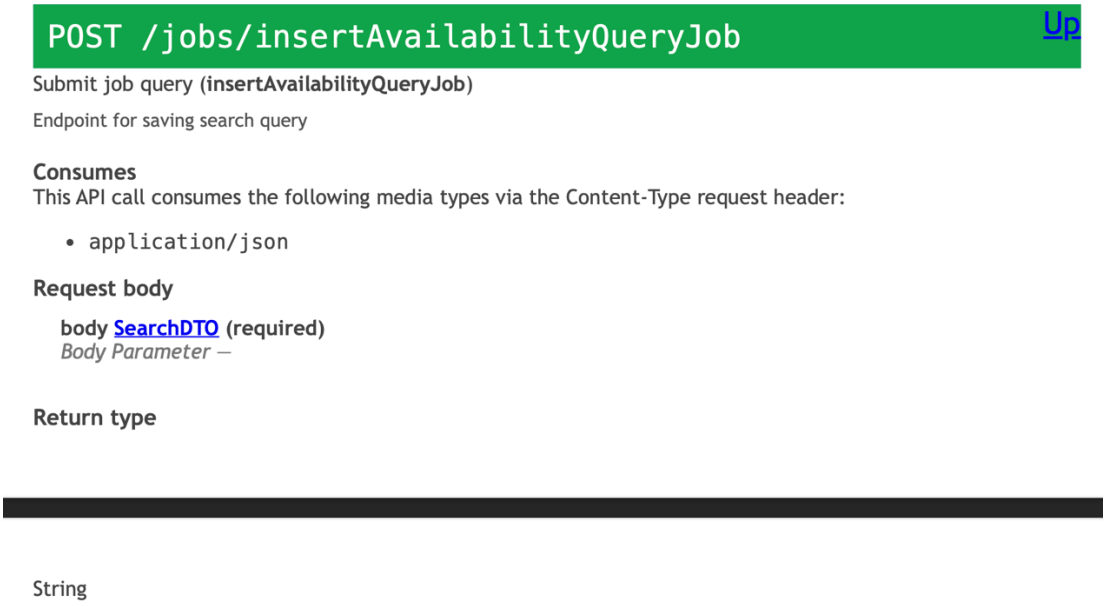

Figure 3. An endpoint for requesting a availability query from the portal. The body of the request has specific structure, SearchDTO, which is accessible for the document link. Return type has been defined as a String. The REST API will not provide immediate results, but the the user of the endpoint must refer to a job number provided in the response string, to request updates of the job progress, and finally the results as files.

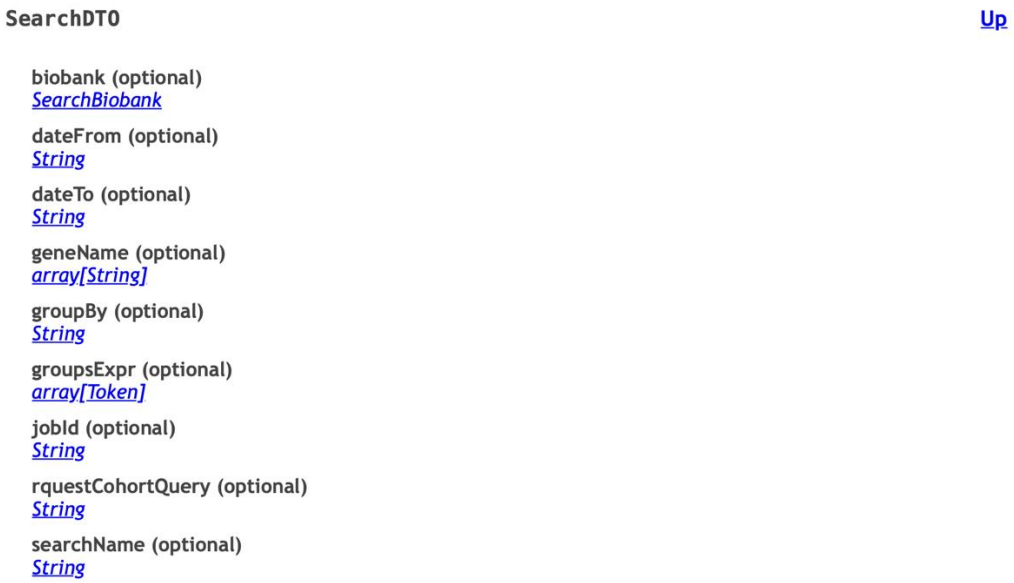

Figure 4. The structure of the request body is described as DTO object in the document.

The protocol for remote access was extended to also include other job administration tasks, otherwise available for the end-users in the portal UI, but deemed convenient to be executed remotely as well. Table 1 describes the key protocol end points that are required for the basic workflow described in

Figure 2. These are the minimum necessary methods for any end user to launch a query and fetch results remotely.

Table 1. The key new endpoints for remote querying and response capture. Full documentation in Appendix B.

Endpoint	Definition
GET /collections/availablityCollections	List of collections with availability query access for the requesting user
GET /variables/classifications	List of classifications of all available variables in the portal
GET /variables/questions	Parameterised request for question OMOP terms per classification, used to select variables for query building
POST /jobs/insertAvailabilityQueryJob	Create and submit a query as request object, returns job identifier
GET /jobs/results/<jobid>	Fetch the results for the job, returns the job status and result files.

3.3. Accessing the protocols

The internal communication protocol operating for portal application end-users is always system controlled, and users are authenticated and then authorised by the OpenID provider system configured for the portal. Within the portal the users are divided into workgroups, where each workgroup has specific data access settings to listed, active collections. Data access is further divided into availability query access and analytical access in the portal data access settings, controlled by portal administrators. In addition, each BC|LINK administrator within each hospital can configure their collections to be usable for availability queries and/or analytical use.

As a new feature, the remote query API has been developed to be accessible to end-users, who have obtained a security token from the portal administrators. In the case of ONCOVALUE BC|RQUEST portal, the security tokens can be requested from the BC Platforms ONCOVALUE project manager. Figure 5 describes the process of obtaining a token as part of the end-user's activities, but currently the portal administrators will request per user account this token upon request. This gives one more level of control to the administration of advanced user access to the query system.

There are no notable changes to the security constraints in the previous implementation, when it comes to query protocol or query engine. Briefly: the query engine obeys a disclosure threshold for each collection, set within the BC|LINK instance (not accessible from the portal). This threshold dictates for queries and analysis results the minimum group size that can be used to report upon. In statistical distributions, the commonly used variables are averages of minimum group sizes. For example, the values of min and max in continuous variables are not exact values, but averages of the disclosure group size N in either minimum end of range or maximum end of range, respectively.

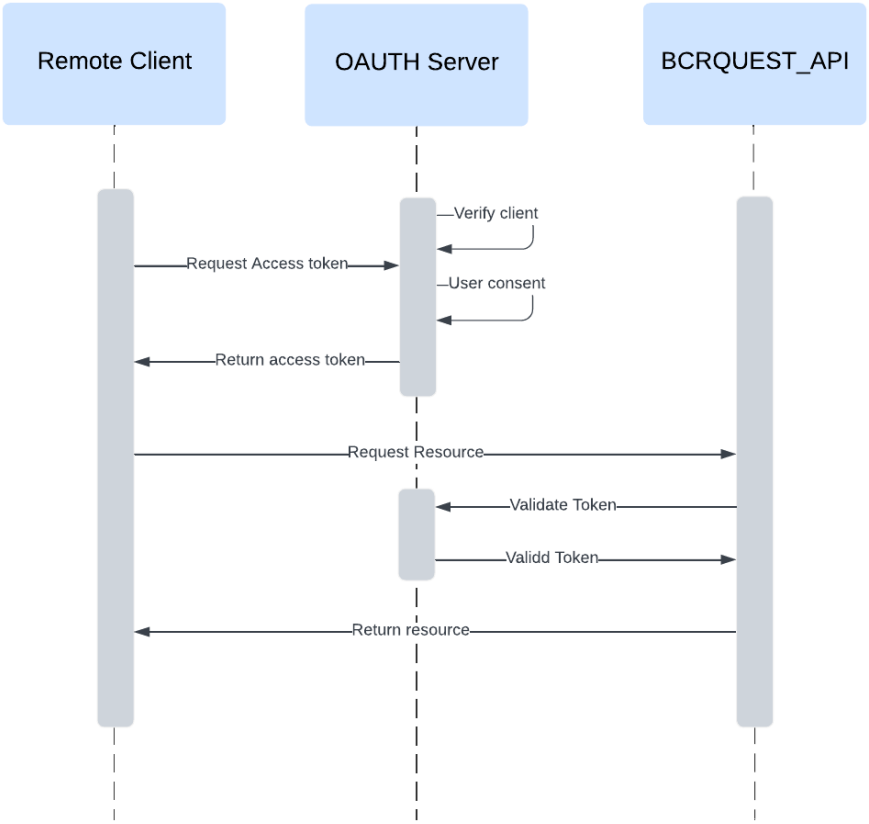

Figure 5. An overview of the authentication and authorisation in place when the remote API is used.

3.4. Using derived non-OMOP variables, and timeline questions

The derived variables, like comorbidity and QoL scores, are typically the product of the data harmonisation process. Often, they do not have any meaningful OMOP term that could be used directly. Hence the query system must support non-OMOP terminology. The query structure allows free coding for custom terms, but these terms must be present in both BC|RQUEST portal data model, and in the responding BC|LINK collection data model.

When OMOP harmonisation work is done according to standard process, dates (or timestamps) of any events are recorded. For performing timeline queries, we have defined 'anchor' event, e.g. first diagnosis of breast cancer and can define time-windows around that event (before and after) and look for other events.

Anchors are typically the diagnosis events, start of treatments, and any decision points to changes in treatments. For observational reports like survival and progression, the curated data needs to reveal if the patient has died, with date of death, or if the patient has been lost to follow-up. The distinction between patients that are actively followed, and patients that have been lost to observation, is important, but can be tricky to achieve in case patients has been moved between hospital districts. The custom terminology support in the query format supports different ways to codify these findings in the curated data.

It is often so that the data made available through such heavy curation process is provided in aggregate, as the individual patient profiles are too easily individualised in any query system. To take advantage of these data, the query system now supports availability queries and filtering directly to aggregate datasets, which are typically either files, or hosted in a light-weight SQL engine with standard ANSI SQL interface.

3.5. Federated analysis

For the purposes of directing federated analysis from the BC|RQUEST portal, each participating collection administrator can evaluate the used analysis tool, and this tool will be specifically installed in the BC|LINK. The BC|LINK collections must opt-in to allow the analysis tools to be used. For the portal to generate a query that involves an analysis using these tools, the analysis job needs to be configured to the portal Analysis page, or as part of normal Availability query, providing an extra downloadable report. These decisions can vary and can be freely defined by the tool developers. However, the protocol must be the same, and the tool is executed with the same parameters, and using the same cohort definitions in all collections.

Table 2 describes the settings each collection administrator must set for their collection to be able to participate in the federated analysis. Note that if the BC|RQUEST portal is configured to run multiple different custom analyses, each one of these needs to be configured to each collection in the BC|LINK. This allows the collection administrators a full control over the decision to allow certain tasks but not the others. If a collection has no configuration for any of the analysis tools, the collection is simply ignored during analysis.

Future development, not related to this API definition deliverable, includes ability to centralise the vetting of the analysis algorithms, and opt-in options in such way that there is no need to specifically install the tools in each BC|LINK, but the analysis system will pull the necessary tools for each task from centralised repository.

Table 2. The extra parameters required in the BC|LINK collection registry settings, to enable specific analysis tools to be used..

Parameter	Definition
rquest.custom_avail_scripts_allowed	Options are ‘yes’ or ‘no’, default is ‘no’. This allows the collection to part-take in the custom federated analysis tasks. Each task must be separately defined in addition to this enabling flag.
rquest.custom_availability_reports start/stop	Specifies a block between ‘start’ and ‘stop’ entries, where the different analysis tools and their parameterisation are specified.
<tool name>;<tool path>;extra_data1=…;extra_dataN=	The ‘tool_name’ is shared with all other collections and the portal, and identifies the specific analysis tools required for a federated job. The ‘tool path’ gives the running location of the tool in the BC|LINK. Parameters ‘extra_data’ are defined as numbered data source identifiers, which are read as SQL data. There can be unlimited number of ‘extra_data’ parameters at the moment.

4. Conclusions

The work done in this deliverable 5.4 expand on an earlier existing implementation of remote - or federated - query execution. The principle of relying on OMOP-coded data makes it possible to standardise various query and aggregation protocols. Any ONCOVALUE clinical partner, which is able to provide OMOP-coded RWD to the project, can utilise these protocols and query remotely the content of their clinical database.

The most important aspect of this expansion is the ability to create - either as a private user or as a service account - queries remotely to the participating collections. This enables building of in-house processes to support local hospitals HTA work. It also allows for comparison of aggregate summaries between different hospitals and even EU countries. In addition, with the rest of the work streams and deliverables in this specific work package, this deliverable acts as a solid foundation for building a community enabled HTA platform, and technology network, which will improve dramatically hospitals, payers and pharmaceutical companies' ability to assess treatment efficacy and safety faster and in more holistic way.

We understand that the end goals of the project, to build the infrastructure for RWD reporting in regulatory/HTA decision-making, where OMOP harmonization standards are uniformly applied in European cancer centres to provide a critical RWD data source for federated analysis by health technology assessment (HTA) and regulatory bodies across Europe are somewhat futuristic and the number of partners who are truly able to participate with RWD collections is not high. However, as this implementation now allows some locations to fully exploit their existing clinical insights to specific cancer treatments, we believe that it acts as a steppingstone and further motivation for other oncology clinics to take advantage of the tooling created in this project.

5. References

The following sources have been referred to in this document:

Articles:

Charlson ME, Pompei P, Ales KL, MacKenzie CR. A new method of classifying prognostic comorbidity in longitudinal studies: development and validation. *J Chronic Dis* 1987;40(5):373-83.

Klabunde CN, Potosky AL, Legler JM, Warren JL. Development of a comorbidity index using physician claims data. *J Clin Epidemiol* 2000 Dec;53(12):1258-67.

Appendices:

Appendix A: BCRQUEST Job generation API for Collections.pdf

Appendix B: BCRQUEST Job generation API for queries.pdf

Appendix A: BCRQUEST Job generation API for Collections

BcRquest API

API documentation for BcRquest
Version: v1
BasePath:/bcrquest_api/v1

Access

Methods

[Jump to Models]

Table of Contents

Collections

- GET /collections/availabilityCollections

Collections

GET /collections/availabilityCollections

Up

Get availability collections (**getAvailabilityCollections**)

Endpoint for getting list of user collections with availability access

Return type
CollectionDTO

Example data
Content-Type: application/json

```
{
  "workgroupId" : 1,
  "collectionAcronym" : "collectionAcronym",
  "permissions" : [ "permissions", "permissions" ],
  "biobankId" : 0,
  "workgroupName" : "workgroupName",
  "biobankCollectionId" : 6,
  "collectionName" : "collectionName"
}
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses
200
Found collections CollectionDTO

Models

[Jump to Methods]

Table of Contents

1. CollectionDTO

CollectionDTO Up

biobankId (optional)
Integer format: int32

biobankCollectionId (optional)
Integer format: int32

collectionAcronym (optional)
String

collectionName (optional)
String

workgroupName (optional)
String

workgroupId (optional)
Integer format: int32

permissions (optional)
array[String]

Appendix B; BCRQUEST Job generation API for Collections

BC|RQUEST API

API documentation for BC|RQUEST
Version: 2.0.0
BasePath:/bcrquest_api/v1

Access

Methods

[Jump to Models]

Table of Contents

Jobs

DELETE /jobs/{id}

Up

Delete query result (**deleteQueryResult**)

Endpoint for delete selected (by ID) query result

Path parameters

id (required)
Path Parameter — Query job id

Return type
String

Example data
Content-Type: application/json

```
""
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- */*

Responses
201
Query result deleted successfully String

GET /jobs/downloadFileForChart

Up

Downloads annotated freqs (**downloadFileForChart**)

Query parameters

jobId (required)
Query Parameter —

title (required)
Query Parameter —

Return type
array[byte[]]

Example data
Content-Type: application/json

```
[ "", "" ]
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/octet-stream`

Responses
200
Found job / search results

PUT /jobs/editName/{id}

Up

Edit job name (**editJobName**)

Endpoint for editing name for selected (by ID) query

Path parameters

id (required)
Path Parameter — Job id

Query parameters

name (required)
Query Parameter — Job name

Return type
SimpleResultDTO

Example data
Content-Type: application/json

```
{
  "jobName" : "jobName",
  "owner" : "owner",
  "lastName" : "lastName",
  "count" : "count",
  "rerunAvailable" : true,
  "finished" : "2000-01-23T04:56:07.000+00:00",
  "firstName" : "firstName",
  "jobId" : "jobId",
  "sharingDetails" : {
    "workgroupId" : 0,
    "sharedAt" : "2000-01-23T04:56:07.000+00:00",
```

```
    "sharedBy" : "sharedBy",
    "users" : [ {
      "fullName" : "fullName",
      "username" : "username"
    }, {
      "fullName" : "fullName",
      "username" : "username"
    } ]
  },
  "submitted" : "2000-01-23T04:56:07.000+00:00",
  "jobType" : "jobType",
  "selected" : true,
  "status" : "status"
}
```


Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
201
Query name edited successfully SimpleResultDTO

GET /jobs/data/genes

Up

Get all genes (**findAllGenesPaginate**)

Endpoint for getting list of all genes for query (Lazy loading enable - default value of result element = 100).

Query parameters

userSearch (optional)
Query Parameter — User search string

limit (optional)
Query Parameter — Limit value for result list format: int32

offset (optional)
Query Parameter — Offset value for result list format: int32

Return type
array[String]

Example data
Content-Type: application/json


```
[ "", "" ]
```


Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200
Found list of genes

GET /jobs/resultsByIds

Up

Get all job's results (**findAllResultsByIds**)

Endpoint for getting list of search results for selected user.

Query parameters

ids (required)
Query Parameter —

Return type
array[JobStatusDTO]

Example data
Content-Type: application/json

```
[ {
  "owner" : "owner",
  "jobId" : "jobId",
  "progress" : {
    "key" : "progress"
  },
  "status" : "status"
}, {
  "owner" : "owner",
  "jobId" : "jobId",
  "progress" : {
    "key" : "progress"
  },
  "status" : "status"
} ]
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses
200
Found job / search results

GET /jobs/simpleAvailabilityJobs

Up

Get all availability jobs (**findAllSimpleAvailabilityJobs**)

Endpoint for getting list of simplified availability jobs for analysis forms.

Return type
array[SimpleAvailabilityJobDTO]

Example data
Content-Type: application/json

```
[ {
  "jobName" : "jobName",
  "owner" : "owner",
  "jobId" : "jobId",
  "rquestCohortQuery" : "rquestCohortQuery"
}, {
  "jobName" : "jobName",
  "owner" : "owner",
  "jobId" : "jobId",
  "rquestCohortQuery" : "rquestCohortQuery"
} ]
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses
200
Found jobs

GET /jobs/result/{jobId}

Up

Get search / job result (**findJobResult**)

Endpoint for getting search result for selected user and search / job ID.

Path parameters

jobId (required)
Path Parameter — Search / job ID

Return type
AvailabilityJobResultDTO

Example data
Content-Type: application/json


```
{
  "jobName" : "jobName",
  "userCollectionId" : "userCollectionId",
  "jsonJobPacket" : "jsonJobPacket",
  "countTableVis" : {
    "visType" : "BAR",
    "data" : {
      "key" : {
        "externalUrl" : "externalUrl",
        "collectionAcronym" : "collectionAcronym",
        "count" : 7,
        "collectionId" : 6,
        "linkName" : "linkName",
        "status" : "status"
      }
    },
    "visId" : "visId",
    "title" : "title"
  },
  "sexChartVis" : {
    "visType" : "BAR",
    "data" : {
      "key" : [ {
        "percentage" : "percentage",
        "count" : 4,
        "label" : "label"
      }, {
        "percentage" : "percentage",
        "count" : 4,
        "label" : "label"
      } ]
    },
    "visId" : "visId",
    "title" : "title"
  },
  "ageChartVis" : {
    "visType" : "BAR",
    "data" : {
      "key" : {
        "q1" : 2.3021358869347655,
        "q3" : 7.061401241503109,
```

```
        "min" : 5.637376656633329,
        "median" : 5.962133916683182,
        "max" : 6.027456183070403,
        "mean" : 1.4658129805029452,
        "count" : 0,
        "label" : "label"
      }
    },
    "visId" : "visId",
    "title" : "title"
  },
  "rquestCohortQuery" : "rquestCohortQuery",
  "resultFileExists" : true,
  "collections" : [ {
    "acronym" : "acronym",
    "name" : "name",
    "id" : 9
  }, {
    "acronym" : "acronym",
    "name" : "name",
    "id" : 9
  } ],
  "failure" : {
    "key" : { }
  },
  "geneTableVis" : {
    "visType" : "BAR",
    "data" : {
      "key" : {
        "externalUrl" : "externalUrl",
        "genes" : {
          "key" : "genes"
        },
        "collectionAcronym" : "collectionAcronym",
        "collectionId" : 1,
        "linkName" : "linkName",
        "status" : "status"
      }
    },
    "visId" : "visId",
    "title" : "title"
  },
  "startTime" : "startTime",
  "endTime" : "endTime",
  "id" : "id",
  "comorbiditiesTableVis" : {
    "visType" : "BAR",
    "data" : {
      "key" : {
        "totalOfCollection" : 1,
        "categories" : [ {
          "codes" : [ {
            "parent" : "parent",
            "children" : [ null, null ],
            "level" : 2,
            "percentageOfCohort" : 4,
            "count" : 3,
            "name" : "name",
            "description" : "description"
          }, {
            "parent" : "parent",
            "children" : [ null, null ],
            "level" : 2,
            "percentageOfCohort" : 4,
            "count" : 3,
```

```
            "name" : "name",
            "description" : "description"
          } ],
          "totalOfCollection" : 1,
          "percentageOfCohort" : 1,
          "count" : 7,
          "name" : "name"
        }, {
          "codes" : [ {
            "parent" : "parent",
            "children" : [ null, null ],
            "level" : 2,
            "percentageOfCohort" : 4,
            "count" : 3,
            "name" : "name",
            "description" : "description"
          }, {
            "parent" : "parent",
            "children" : [ null, null ],
            "level" : 2,
            "percentageOfCohort" : 4,
            "count" : 3,
            "name" : "name",
            "description" : "description"
          } ],
          "totalOfCollection" : 1,
          "percentageOfCohort" : 1,
          "count" : 7,
          "name" : "name"
        } ]
      }
    },
    "visId" : "visId",
    "title" : "title"
  },
  "status" : "COMPLETED_WITH_ERROR"
}
```


Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200
Found full job result AvailabilityJobResultDTO

POST /jobs/insertAvailabilityQueryJob

Up

Submit job query (**insertAvailabilityQueryJob**)

Endpoint for saving search query

Consumes
This API call consumes the following media types via the Content-Type request header:

- `application/json`

Request body

body SearchDTO (required)
Body Parameter —

Return type

String

Example data
Content-Type: application/json

```
""
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
201
Search query created String

POST /jobs/rerun/{id}

Up

Rerun job query (**rerunQuery**)

Endpoint for rerunning selected (by ID) query

Path parameters

id (required)
Path Parameter — Query job id

Return type
String

Example data
Content-Type: application/json

```
""
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
201
Query run successfully String

Variables

GET /variables/classifications

Up

Get all classifications (**findAllClassifications**)

Endpoint for getting all classifications.

Query parameters

collections (optional)
Query Parameter —

Return type
array[String]

Example data
Content-Type: application/json

```
[ "", "" ]
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200
Found classifications

GET /variables/questions

Up

Get search tokens for tree view (**getQuestions**)

Endpoint for getting hierarchical search tokens to tree view.

Query parameters

parentId (optional)
Query Parameter —

categoryId (optional)
Query Parameter —

collections (optional)
Query Parameter —

page (required)
Query Parameter — format: int32

size (required)
Query Parameter — format: int32

Return type
array[VariableDTO]

Example data
Content-Type: application/json


```
[ {
  "code" : "code",
  "synonyms" : {
    "key" : [ "synonyms", "synonyms" ]
  },
  "type" : "CATEGORY",
  "parentId" : "parentId",
  "codeValue" : "codeValue",
  "variableType" : "TEXT",
  "children" : [ null, null ],
  "collections" : [ "collections", "collections" ],
  "name" : "name",
  "variableCount" : 1,
  "id" : "id",
  "secondaryModifiers" : {
    "stats" : {
      "timeAvailable" : 6,
      "options" : [ {
        "choice" : "choice",
        "value" : "value"
      }, {
        "choice" : "choice",
        "value" : "value"
      } ],
      "ageAvailable" : 0
```