ImmPort Contract

SYSTEM ARCHITECTURE AND SOFTWARE DESIGN SPECIFICATION

Version 7.0

Updated Date: October 30, 2023

Project Sponsor:

National Institutes of Health (NIH)
National Institute of Allergy and Infectious Diseases (NIAID)
Division of Allergy, Immunology, and Transplantation (DAIT)

Prepared by:

Peraton

Public Health Programs
2101 Gaither Road, Suite 500
Rockville, MD 20850
(301)527-6400 Main
(301)527-6401 Fax
Morgan.Crafts@peraton.com

Contents

1. INTRODUCTION **7**
1.1. SCOPE 7
1.2. PURPOSE 7
1.3. BACKGROUND 7
2. IMMPORT TOOLS ON AMAZON WEB SERVICES **8**
2.1. DATABASE ARCHITECTURE 9
2.1.1. OVERVIEW 9
2.1.2. Shared Data 9
2.1.2.1. Administrative 9
2.1.2.2. Study 10
2.1.2.3. Subject 11
2.1.2.4. Assessment 12
2.1.2.5. Lab Test 13
2.1.2.6. Experiment 14
2.1.2.7. Assay Results 14
2.1.2.8. Lookup Tables - Part 1 16
2.1.2.9. Lookup Tables - Part 2 18
2.1.3. Metric 19
2.1.3.1. Metric 19
2.1.4. Cell Ontology 19
2.2. HOSTED APPLICATIONS 19
2.2.1. ImmPort Shared Data 20
2.2.1.1. Feature Summary 20
2.2.1.1.1. Feature 1: Home Page 21
2.2.1.1.2. Feature 2: Search Bar and Results 22
2.2.1.1.3. Feature 3: Study Detail Page 23
2.2.1.2. Hardware and Software Architecture Components 24
2.2.1.3. Data Architecture 24
2.2.1.4. ImmPort Shared Data Server Architecture 25
2.2.1.5. ImmPort Shared Data Software Architecture 26
2.2.2. Study Search 27
2.2.2.1. Aggregated Fields 28
2.2.2.2. Boosting 28
2.2.2.3. Ngram search: 29
2.2.2.4. Phrase search 29
2.2.3. ImmPort Data Query API 32
2.2.3.1. Feature Summary 32
2.2.3.2. ImmPort Data Query API Server Architecture 33
2.2.3.3. ImmPort Data Query API Software Architecture 34

2.2.4. ImmPort Data Browser 35
2.2.4.1. Feature Summary 35
2.2.4.1.1. Feature 1: Home Page 36
2.2.4.1.2. Feature 2: Study Drill down 37
2.2.4.1.3. Feature 3: Aspera Connect Download 38
2.2.4.2. ImmPort Data Browser Architecture 38
2.2.5. ImmPort Data API 39
2.2.5.1. Feature Summary 39
2.2.5.2. ImmPort Data API Server Architecture 39
2.2.5.3. ImmPort Data Query API Software Architecture 40
2.2.6. ImmPort Resources 41
2.2.6.1. Feature Summary 41
2.2.6.1.1. Feature 1: Home Page 42
2.2.6.1.2. Feature 2: User agreement 43
2.2.6.1.3. Feature 2: Resources Page 43
2.2.6.2. SeroNet CDT 44
2.2.6.2.1. Introduction 44
2.2.6.2.2. Architecture 44
2.2.6.2.3. Feature Summary 44
2.2.7. ImmPort S3 API 46
2.2.7.1. Feature Summary 46
2.2.7.2. ImmPort S3 API Server Architecture 48
2.2.8. ImmPort CloudFront Distributions 48
2.2.9. Authentication of the Rest API - Resource Servers 48
2.2.10. Cell Ontology Browser 50
2.2.10.1. Cell Ontology Browser Architecture 51
2.2.11. ImmuneXpresso 52
2.2.11.1. ImmuneXpresso Architecture 53
2.2.12. Galaxy 53
2.2.13. ImmuneSpace 55
2.2.14. Metrics 55
2.2.14.1. A common data model for Metrics 55
2.2.14.2. Metrics Software Architecture 57
2.2.14.3. Visualizations in Kibana 57
2.2.14.3.1. Example of the Discovery tab 57
2.2.14.3.2. Example of the Visualization tab : 58
2.2.15. Utilization Report 59
2.2.16. ImmPort HAPI FHIR Server 59
2.3. Data Release 62
2.3.1. Overview of OCICB Components and Process 62
2.3.2. Overview of AWS Components and Process 65
2.4. Continuous Integration and Continuous Delivery (CICD) 66

3. ImmPort OCICB **67**
3.1. OVERVIEW 67
3.1.1. Feature Summary 68
3.2. OCICB ARCHITECTURE 69
3.3. DATABASE ARCHITECTURE 70
3.3.1. CORE_IMMPORT 71
3.3.1.1. Overview 71
3.3.1.2. Workspace 71
3.3.1.3. Administrative 72
3.3.1.4. Study 73
3.3.1.5. Subject 74
3.3.1.6. Assessment 75
3.3.1.7. Lab Test 76
3.3.1.8. Experiment 77
3.3.1.9. Assay Results 78
3.3.1.10. Lookup Tables - Part 1 79
3.3.1.11. Lookup Tables - Part 2 80
3.3.1.12. Lookup Tables - Part 3 81
3.3.1.13. Lookup Tables - Part 4 82
3.3.1.14. Upload Registration 83
3.3.1.15. Curation 83
3.3.1.16. BISC Security 84
3.4. Aspera Server 84
3.5. HOSTED APPLICATIONS 85
3.5.1. ImmPort Core Query API 85
3.5.1.1. Feature Summary 85
3.5.1.2. ImmPort Core Query API Software Architecture 86
3.5.2. Data Submission 87
3.5.2.1. Validation 88
3.5.2.2. Submission 88
3.5.3. Data Batch Update 89
3.5.3.1. Validation 89
3.5.3.2. Submission 89
3.5.4. File System Management Update 90
3.5.5. Data Manager 92
3.5.6. Sharing Tool 92
3.5.7. User Administration 94
3.5.7.1. Registering User 94
3.5.7.2. Creating an User by an Administrator 95
3.5.7.3. Search User 97
3.5.7.4. Applications/Groups/Roles 97
3.5.8. APIs 98

3.5.8.1. Batch Uploader API 98
3.5.8.2. Batch Updater API 100
3.5.9. ImmPort Study Registration Wizard 101

1. INTRODUCTION

1.1. SCOPE

The scope of the ImmPort contract is to provide advanced information technology support in the production, analysis, archiving, and exchange of scientific data for a diverse community of life science researchers.

1.2. PURPOSE

The Immunology Database and Analysis Portal (ImmPort) applications are being developed by a Peraton-led team. Previous development was performed by the same team under Northrop Grumman Information Systems (NGIS) with academic partners from the University of California San Francisco in the current contract (phase 2 and 3) and the University of Texas-Southwestern in the prior phase 1 contract. The ImmPort system is intended to serve as a long-term, sustainable archive of data generated by investigators funded through the Division of Allergy, Immunology, and Transplantation (DAIT) of the National Institute of Allergy and Infectious Disease (NIAID), National Institutes of Health (NIH). The ImmPort system consists of an extensive data warehouse containing an integration of experimental and clinical data supplied by NIAID/DAIT-funded investigators. The ImmPort system is freely accessible as a resource to all scientists in the research community.

This System Architecture and Software Design Specification (SASDS) defines the overall ImmPort architecture and software design specification identified by the Peraton ImmPort Team (hereinafter referred to as the ImmPort Team) for the ImmPort system developed for NIAID/DAIT. The architecture and design described in this document focus on the capabilities that are implemented in the ImmPort family of applications as of June 30, 2021.

1.3. BACKGROUND

The key objective of the SASDS version 6.0 is to provide an update to the hardware and software specifications of the system. The ImmPort project has evolved, such that systems are now hosted in a production or near-production mode at both the NIAID hosting facility as well as Amazon Web Services (AWS) cloud environment. In general, the long-term goal, which is in progress, is to provide hosting of data while it is private and being QC'ed and curated in ImmPort related applications housed at the NIAID hosting facility. When data is shared with the general scientific community, data would be transferred to AWS for easier re-use of these data for analysis, or in short, to bring the data to the analysis tool. This bifurcation of systems allows ImmPort and NIAID staff to maintain maximum control over data while it is sensitive and private, and more flexibility for re-use and distribution when the data is shared in AWS.

The ImmPort contract includes a shift in focus to lessen the level of effort spent on the development, maintenance, and outreach for analysis tools and reference data capabilities. The FLOCK flow analysis tool suite continued to have constant usage and increased interest and publications related to tool usage and results, so the decision was made to continue support of that application. As a result of these decisions, the retired tools and queries will not appear in this design document.

For the remaining features of ImmPort, it was recognized a general code refresh was necessary given the overall age of the software and supporting stack of frameworks. As a result, the ImmPort team progressed incrementally through the upgrade of features into a new software architecture detailed in this document. For the purposes of this document, newer code architecture is referred to as "ImmPort 3.0", while the prior architecture being gradually replaced is referred to as "ImmPort 2.0". This document will detail the ImmPort 3.0 architecture for features that have been upgraded or will soon be upgraded and will keep the existing documentation in place for ImmPort 2.0 features not yet upgraded in the production environment. As features are completed, this document will be accordingly updated. The functional requirements documents for features in progress for the upgrade will also be available and referenced in this document.

Compared to prior versions of this document that detailed ImmPort 2.0 architecture, the overall system architecture for ImmPort 3.0 is being simplified to have a less dense middle-tier. As a result, the need to document detailed design packages in this SASDS is reduced, since the same middle-tier approach is utilized across the features in a given application and the EJB tier has been removed.

The database documentation has been moved online, so is no longer described in detail in this document. References to the freely available online materials will be provided in this document. Far more detail about the database fields, tables, and ERD diagrams are available online than in prior versions of the SASDS, so the overall amount of information has increased markedly. Additionally, since the SASDS document was released the entire database in MySQL is available for anyone to download and re-use, making comprehension of the database architecture much simpler.

This document is to be considered a “work in progress” and will evolve during the life of the ImmPort effort as additional requirements are implemented, new requirements are identified, and others are modified or deleted.

2. IMMPORT TOOLS ON AMAZON WEB SERVICES

The ImmPort tools deployed on Amazon Web Services (AWS) are designed primarily to

1. Identify studies of interest for users to evaluate for future analysis. The application performing this feature is ImmPort Shared Data.
2. Download studies of interest. The application performing this feature is ImmPort Data Browser.
3. Provide a unified platform for several ImmPort resources such as documentation, tutorials, upload templates, example packages, blogs. The application performing this feature is ImmPort Portal.

In addition to the above applications, the AWS infrastructure is utilized to develop Alpha and Beta tools to obtain feedback on utility from the user community. Tools such as ImmuneXpresso and the Cell Ontology browser fit into this category. Finally, the AWS infrastructure is being used to host production applications developed by other research teams funded by DAIT without the funding to support a federal system. ImmuneSpace is the first example of this usage of ImmPort resources.

2.1. DATABASE ARCHITECTURE

2.1.1. OVERVIEW

Currently ImmPort has 3 databases instantiated in the production environment, 2 use Aurora MySQL databases, and the third (Ontology) uses a local MySQL installed on an Ubuntu server. The plan is to move the Ontology database to the Aurora MySQL in the next year.

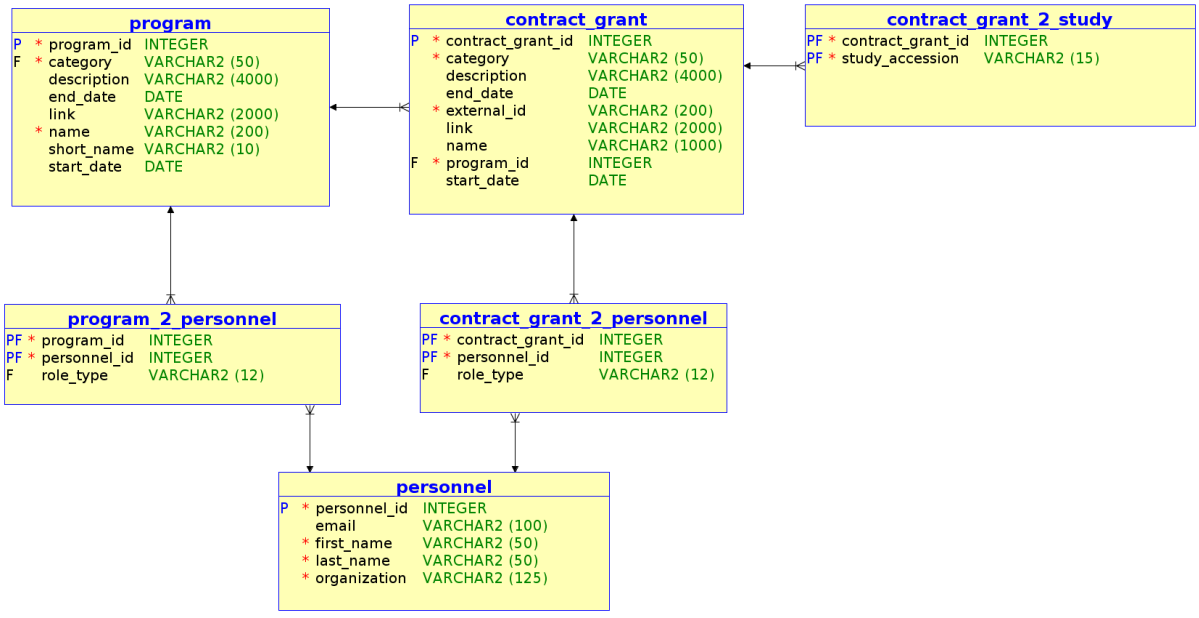
1. Shared_Data - read-only, data that has been shared to the public
2. Metric - read/write metric from the various applications is logged to this database
3. Ontology - read-only, contains data to support the Cell Ontology browser

2.1.2. Shared Data

The Shared_Data database contains all the data shared for public access as part of the Data Release process. The Data Release process occurs approximately 4 to 6 times per year. Between releases, data remains static and accessed primarily using read-only api. With each release, we create a new database, named DRXX_SHARED_DATA, and at the time of the Data Release, the connection string for the applications is updated to use the new version of the Shared_Data database.

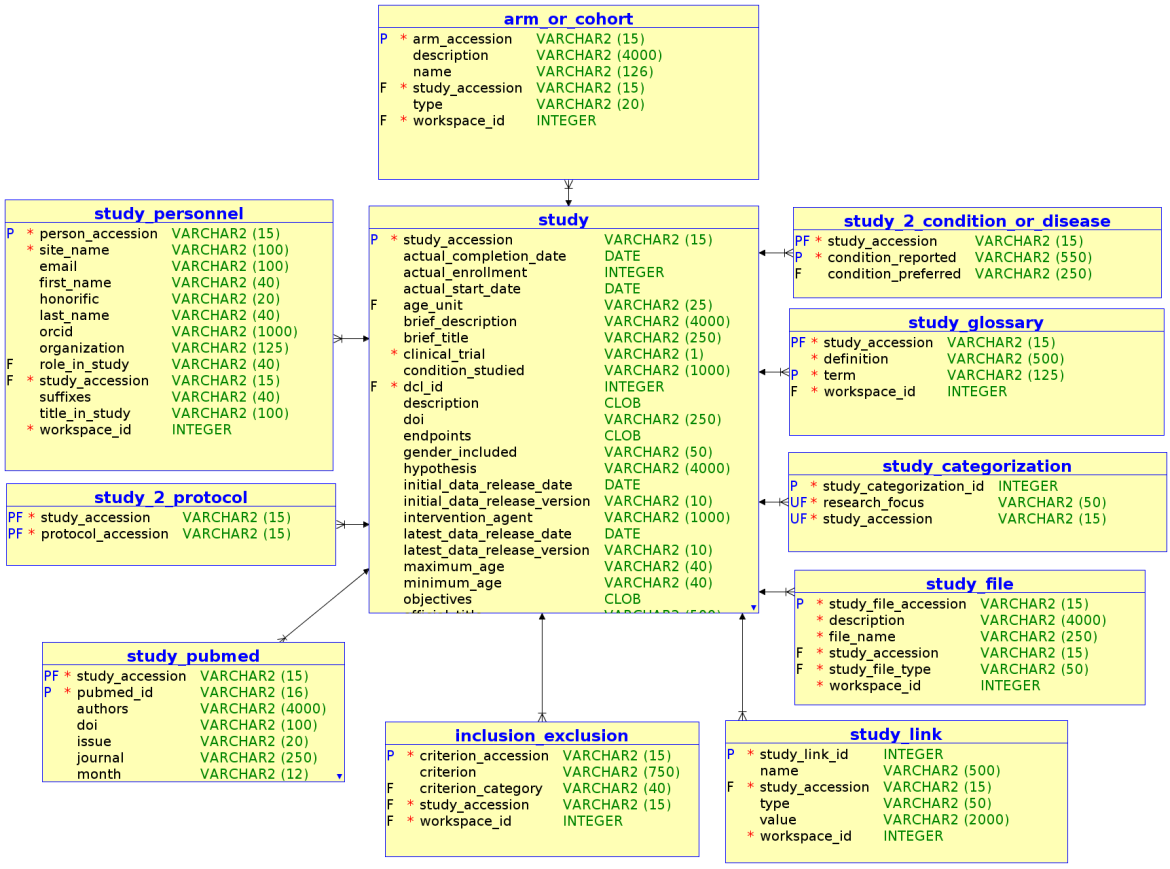
2.1.2.1. Administrative

This diagram represents the tables used to capture Program and Contract information. Several contracts can be linked to one Program and one or more studies can be linked to each contract.



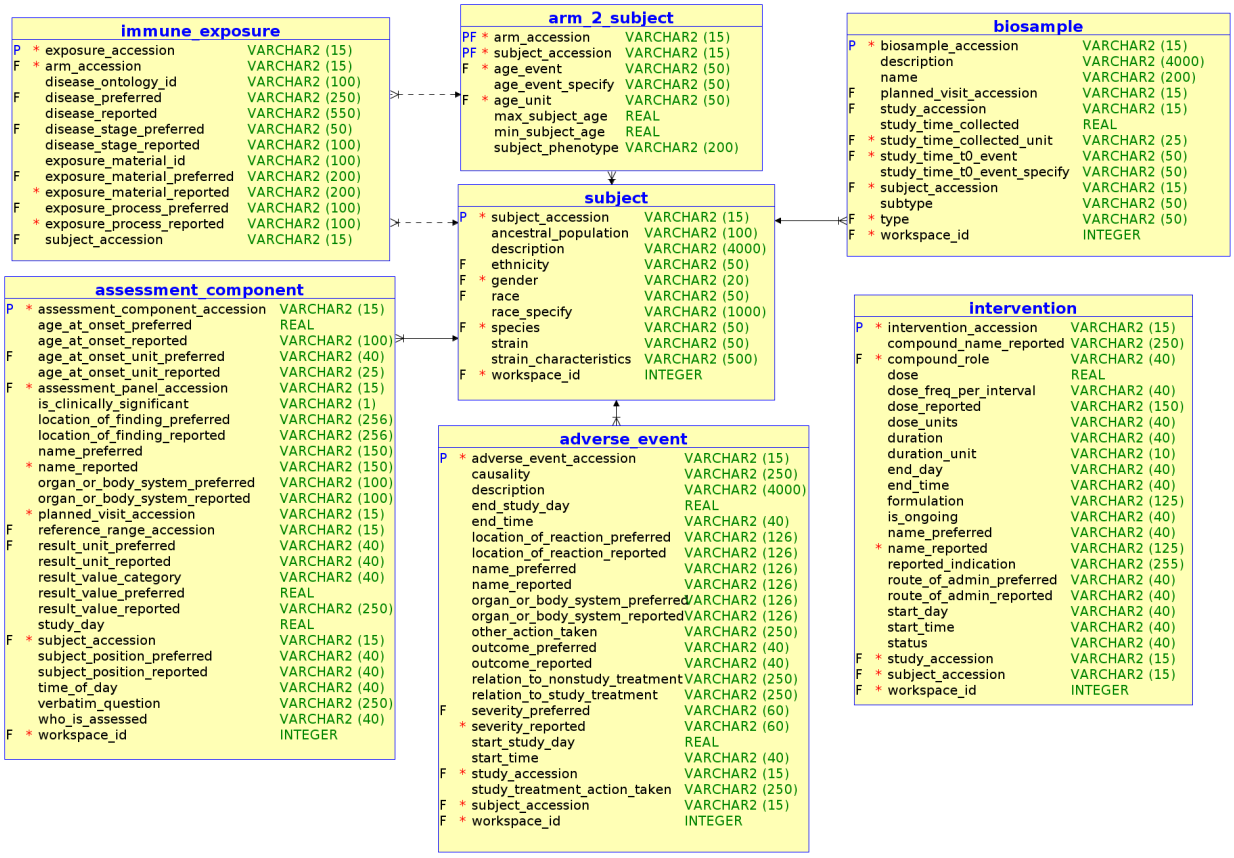
2.1.2.2. Study

This diagram represents information for the overall design of the **study**. The **arm_or_cohort** table is used to link studies to subjects using the **arm_2_subject table.** The **study_file** table is used to link various types of files, uploaded by data providers, where the file contents may or may not be structured. Examples of file types uploaded are Case Report forms, generic study data, lab results, assessments, etc. Some of these files may be parsed and loaded into tables like assessment/assessment_component and lab_test_panel/lab_test. Assay result files are not normally loaded into the **study_file** table but are stored in the **file_info** table and usually associated with an **expsample** record. The **study_2_condtion_or_disease** table supports associating one or more conditions/diseases to a study. The terms for the conditions/diseases are in the **lk_disease** table, which uses the Disease Ontology and Human Phenotype Ontology, as the source for the terms. The **study_categorization** table provides a method to link a **study** to broad research focus areas. Examples of research focus areas are Immune Response, Vaccine Response, Transplantation, etc.



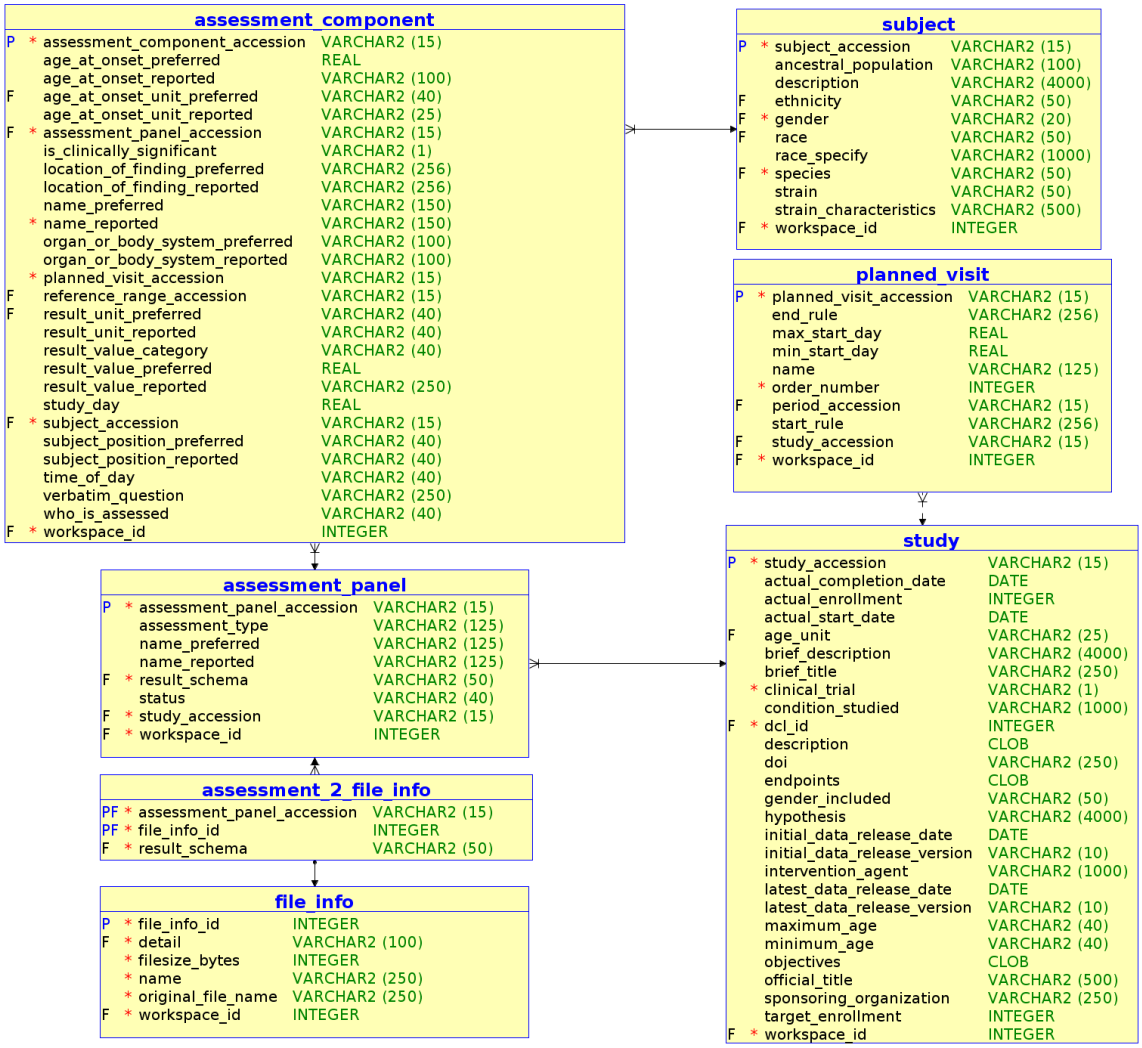
2.1.2.3. Subject

These tables contain subject-level information. A subject may be linked to one or more studies via the **arm_2_subject** table. The **arm_2_subject** allows the data model to capture the age of the subject at the time of each study. This is important when individual studies are part of a larger longitudinal study and the time frame can span several years. The **biosample** table represents the material obtained from the subject at a specific point in time. For example, if the protocol called for obtaining a blood sample on Day1, Day 7, and Day28, there would be 3 biosample records for each **subject.**



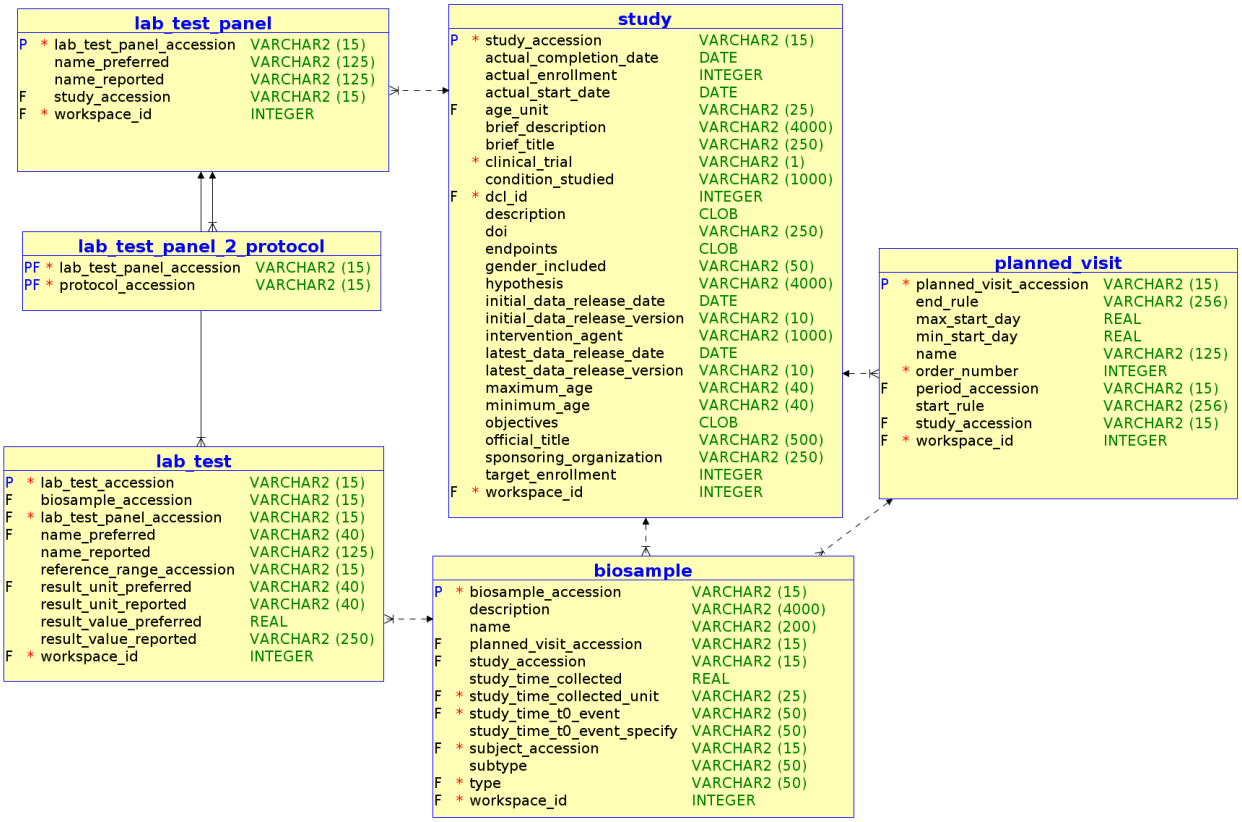
2.1.2.4. Assessment

Individual **assessment_component** records can be grouped together as a unit, using the **assessment_panel** record. For example, an **assessment_panel** record may represent a questionnaire filled out by a subject, and each of the 20 questions on the questionnaire is represented by 20 assessment_component records. **Subject** records are linked directly to **assessment_component** records, which is different from **biosample** records which are linked to **lab_test** records. The reason **subject** records are linked directly to an assessment is an assessment can occur without a biological specimen being collected, for example when filling out a questionnaire. **Assessment_component** records are linked to a **planned_visit** record, and the **planned_visit** record captures the temporal aspects of when the assessments were made.



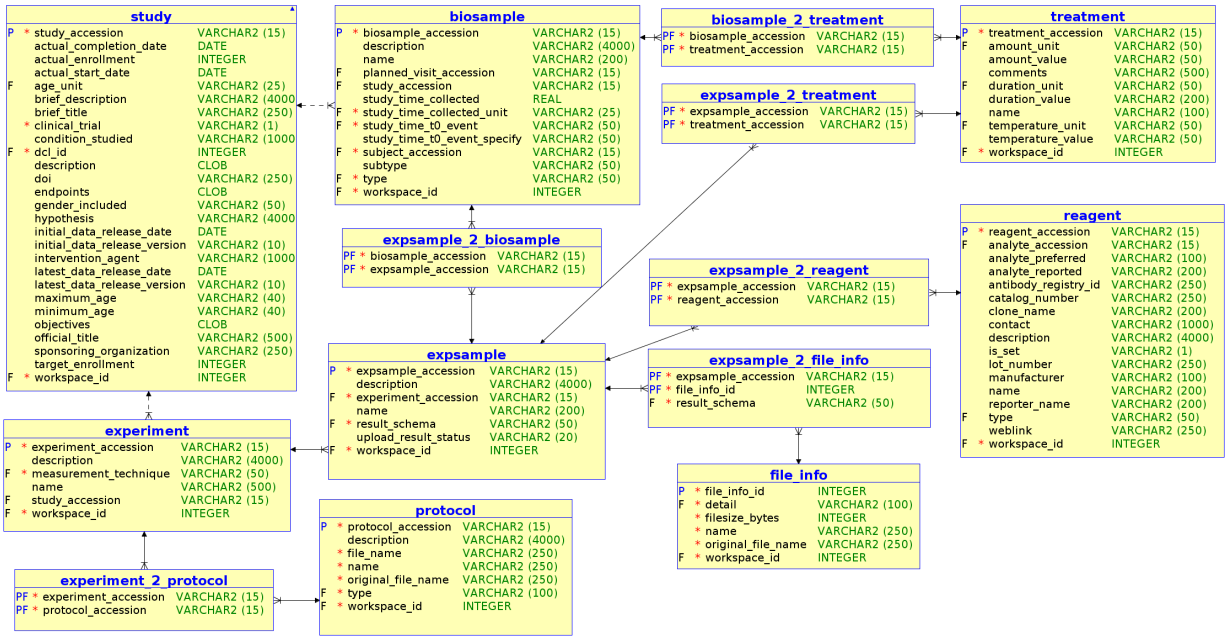
2.1.2.5. Lab Test

Individual **lab_test** records can be grouped as a unit using the **lab_test_panel** record. For example, a **lab_test_panel** record may represent a group of chemistry tests made on a single blood sample and each of the 10 tests that make up the chemistry test panel are represented by 10 lab_test records. **Biosample** records are linked directly to **lab_test** records. The **biosample** records are also linked to a **planned_visit** record, and the **planned_visit** record captures the temporal aspects of when the **biosample was** obtained and when lab tests were performed.



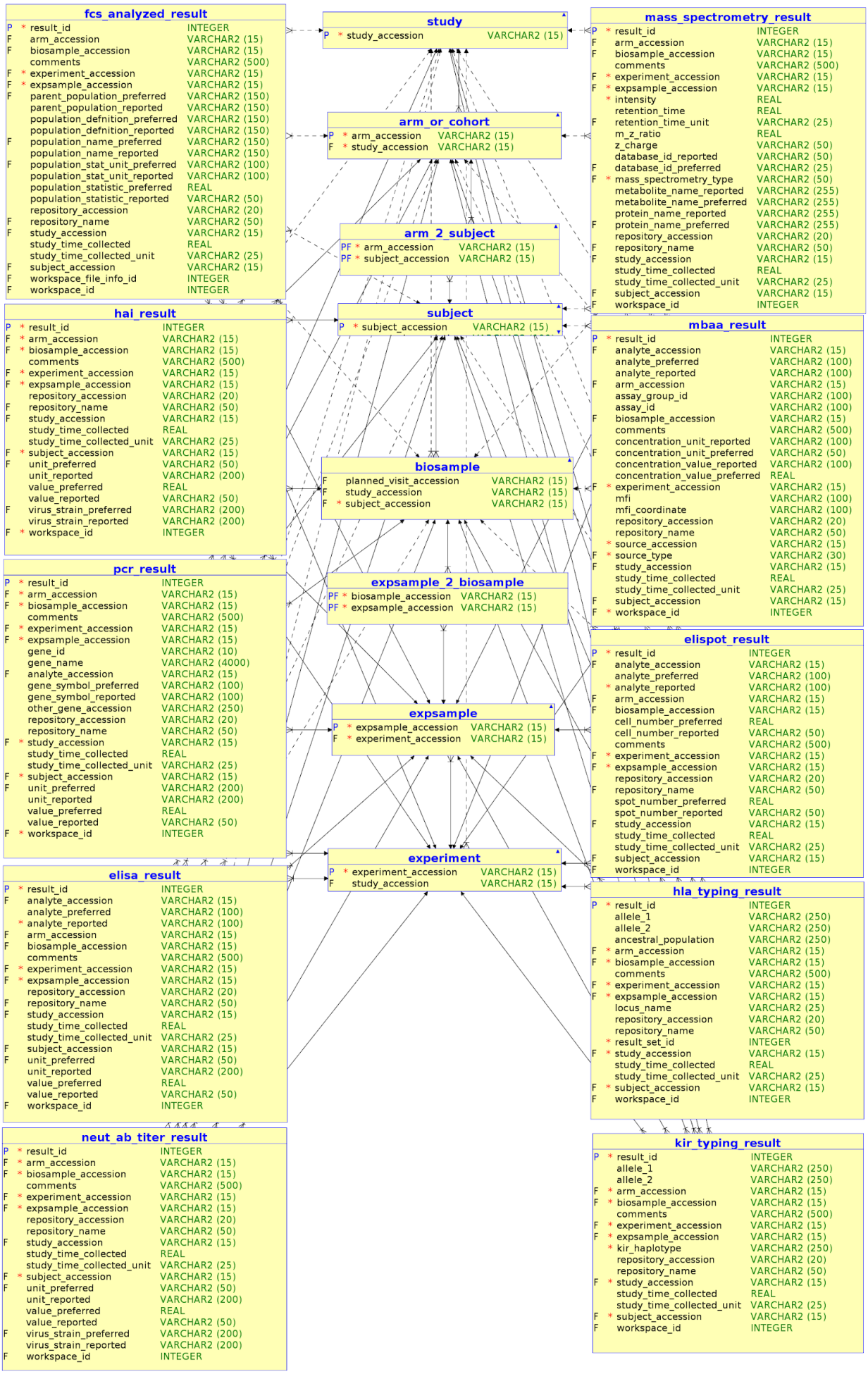
2.1.2.6. Experiment

The **experiment** table represents assays performed using multiple **expsample** records**.** Types of assay methods (measurement_techique) are ELISA, ELISPOT, PCR, Flow Cytometry, etc. The **expsample** record is obtained from the **biosample** record, in some experiments, the original **biosample** may have been divided into multiple **expsamples,** with each **expsample** used for a different assay method. If the original assay result file has been uploaded by the data provider, the **expsample** record is linked to the record in the **file_info** table, via the **expsample_2_file_info** table.



2.1.2.7. Assay Results

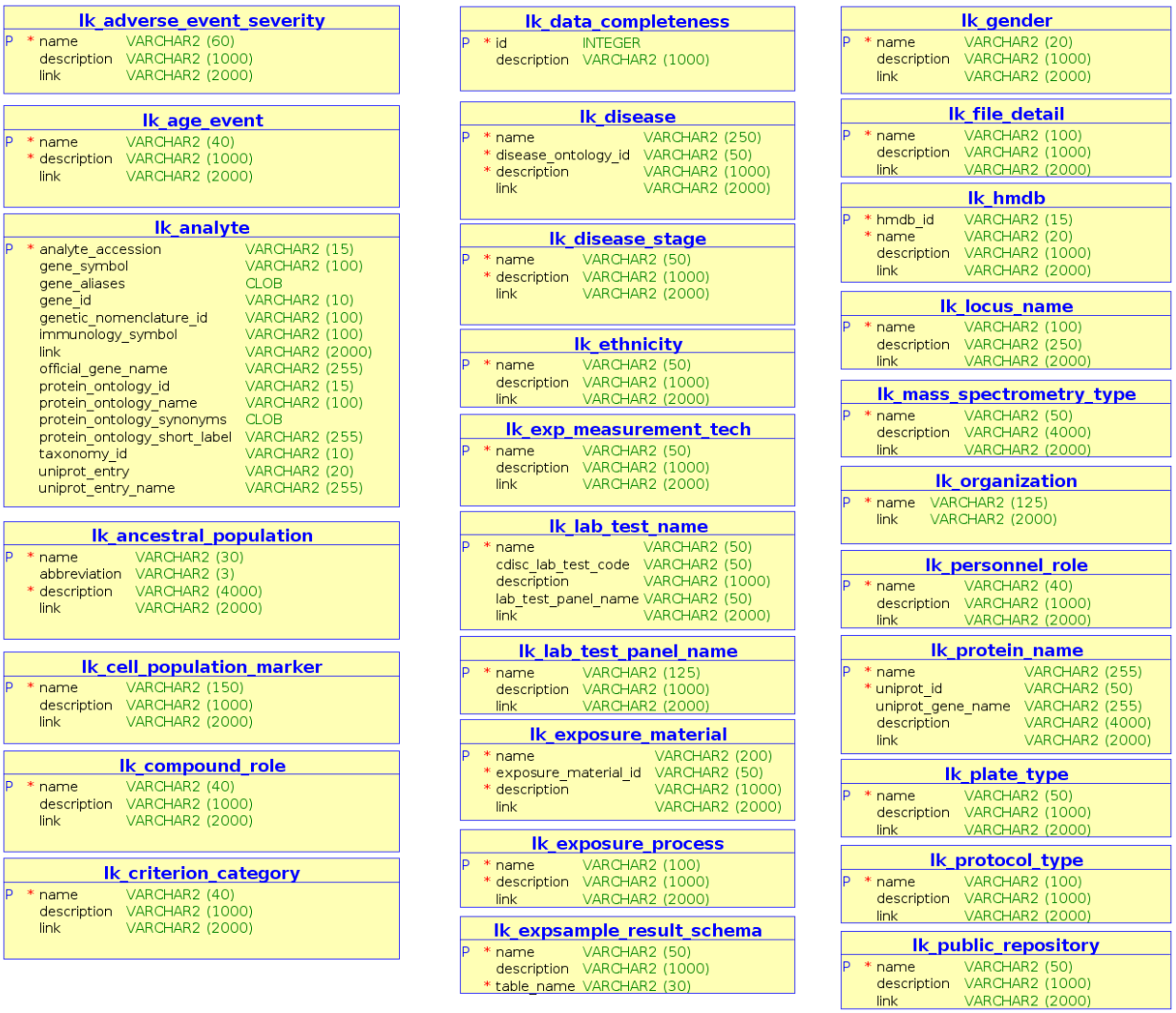
For common assay methods where result formats are fairly standardized the results are parsed into the result table for that assay method is supplied by the data provider. In the Shared_Data schema, when this information is extracted from the production operational database, several properties are denormalized into these tables to make them easier to use in downstream analysis. In the production database, the base table normally has only the **experiment_accession** and **expsample_accession**.


fcs_analyzed_result
P * result_id INTEGER
F arm_accession VARCHAR2 (15)
F biosample_accession VARCHAR2 (15)
comments VARCHAR2 (500)
F * experiment_accession VARCHAR2 (15)
F * expsample_accession VARCHAR2 (15)
F parent_population_preferred VARCHAR2 (150)
parent_population_reported VARCHAR2 (150)
population_defnition_preferred VARCHAR2 (150)
population_defnition_reported VARCHAR2 (150)
F population_name_preferred VARCHAR2 (150)
population_name_reported VARCHAR2 (150)
F population_stat_unit_preferred VARCHAR2 (100)
population_stat_unit_reported VARCHAR2 (100)
population_statistic_preferred REAL
population_statistic_reported VARCHAR2 (50)
repository_accession VARCHAR2 (20)
F repository_name VARCHAR2 (50)
F study_accession VARCHAR2 (15)
study_time_collected REAL
study_time_collected_unit VARCHAR2 (25)
F subject_accession VARCHAR2 (15)
F workspace_file_info_id INTEGER
F workspace_id INTEGER
study
P * study_accession VARCHAR2 (15)
arm_or_cohort
P * arm_accession VARCHAR2 (15)
F * study_accession VARCHAR2 (15)
arm_2_subject
PF * arm_accession VARCHAR2 (15)
PF * subject_accession VARCHAR2 (15)
subject
P * subject_accession VARCHAR2 (15)
mass_spectrometry_result
P * result_id INTEGER
F arm_accession VARCHAR2 (15)
F biosample_accession VARCHAR2 (15)
comments VARCHAR2 (500)
F * experiment_accession VARCHAR2 (15)
F * expsample_accession VARCHAR2 (15)
* intensity REAL
retention_time REAL
F retention_time_unit VARCHAR2 (25)
m_z_ratio REAL
z_charge VARCHAR2 (50)
database_id_reported VARCHAR2 (50)
F database_id_preferred VARCHAR2 (25)
F * mass_spectrometry_type VARCHAR2 (50)
metabolite_name_reported VARCHAR2 (255)
metabolite_name_preferred VARCHAR2 (255)
protein_name_reported VARCHAR2 (255)
F protein_name_preferred VARCHAR2 (255)
repository_accession VARCHAR2 (20)
F repository_name VARCHAR2 (50)
F study_accession VARCHAR2 (15)
study_time_collected REAL
study_time_collected_unit VARCHAR2 (25)
F subject_accession VARCHAR2 (15)
F workspace_id INTEGER
hai_result
P * result_id INTEGER
F * arm_accession VARCHAR2 (15)
F * biosample_accession VARCHAR2 (15)
comments VARCHAR2 (500)
F * experiment_accession VARCHAR2 (15)
F * expsample_accession VARCHAR2 (15)
repository_accession VARCHAR2 (20)
F repository_name VARCHAR2 (50)
F study_accession VARCHAR2 (15)
study_time_collected REAL
study_time_collected_unit VARCHAR2 (25)
F * subject_accession VARCHAR2 (15)
F unit_preferred VARCHAR2 (50)
unit_reported VARCHAR2 (200)
value_preferred REAL
value_reported VARCHAR2 (50)
F virus_strain_preferred VARCHAR2 (200)
virus_strain_reported VARCHAR2 (200)
F * workspace_id INTEGER
biosample
F planned_visit_accession VARCHAR2 (15)
F study_accession VARCHAR2 (15)
F * subject_accession VARCHAR2 (15)
mbaa_result
P * result_id INTEGER
F analyte_accession VARCHAR2 (15)
analyte_preferred VARCHAR2 (100)
analyte_reported VARCHAR2 (100)
F arm_accession VARCHAR2 (15)
assay_group_id VARCHAR2 (100)
assay_id VARCHAR2 (100)
F biosample_accession VARCHAR2 (15)
comments VARCHAR2 (500)
concentration_unit_reported VARCHAR2 (100)
F concentration_unit_preferred VARCHAR2 (50)
concentration_value_reported VARCHAR2 (100)
concentration_value_preferred REAL
F * experiment_accession VARCHAR2 (15)
mfi VARCHAR2 (100)
mfi_coordinate VARCHAR2 (100)
repository_accession VARCHAR2 (20)
repository_name VARCHAR2 (50)
* source_accession VARCHAR2 (15)
F * source_type VARCHAR2 (30)
F study_accession VARCHAR2 (15)
study_time_collected REAL
study_time_collected_unit VARCHAR2 (25)
F subject_accession VARCHAR2 (15)
F * workspace_id INTEGER
pcr_result
P * result_id INTEGER
F * arm_accession VARCHAR2 (15)
F * biosample_accession VARCHAR2 (15)
comments VARCHAR2 (500)
F * experiment_accession VARCHAR2 (15)
F * expsample_accession VARCHAR2 (15)
gene_id VARCHAR2 (10)
gene_name VARCHAR2 (4000)
F analyte_accession VARCHAR2 (15)
gene_symbol_preferred VARCHAR2 (100)
gene_symbol_reported VARCHAR2 (100)
other_gene_accession VARCHAR2 (250)
repository_accession VARCHAR2 (20)
repository_name VARCHAR2 (50)
F * study_accession VARCHAR2 (15)
study_time_collected REAL
study_time_collected_unit VARCHAR2 (25)
F * subject_accession VARCHAR2 (15)
F unit_preferred VARCHAR2 (200)
unit_reported VARCHAR2 (200)
value_preferred REAL
value_reported VARCHAR2 (50)
F * workspace_id INTEGER
expsample_2_biosample
PF * biosample_accession VARCHAR2 (15)
PF * expsample_accession VARCHAR2 (15)
elispot_result
P * result_id INTEGER
F analyte_accession VARCHAR2 (15)
analyte_preferred VARCHAR2 (100)
* analyte_reported VARCHAR2 (100)
F arm_accession VARCHAR2 (15)
F biosample_accession VARCHAR2 (15)
cell_number_preferred REAL
cell_number_reported VARCHAR2 (50)
comments VARCHAR2 (500)
F * experiment_accession VARCHAR2 (15)
F * expsample_accession VARCHAR2 (15)
repository_accession VARCHAR2 (20)
F repository_name VARCHAR2 (50)
spot_number_preferred REAL
spot_number_reported VARCHAR2 (50)
F study_accession VARCHAR2 (15)
study_time_collected REAL
study_time_collected_unit VARCHAR2 (25)
F subject_accession VARCHAR2 (15)
F workspace_id INTEGER
expsample
P * expsample_accession VARCHAR2 (15)
F * experiment_accession VARCHAR2 (15)
experiment
P * experiment_accession VARCHAR2 (15)
F study_accession VARCHAR2 (15)
elisa_result
P * result_id INTEGER
F analyte_accession VARCHAR2 (15)
analyte_preferred VARCHAR2 (100)
* analyte_reported VARCHAR2 (100)
F arm_accession VARCHAR2 (15)
F biosample_accession VARCHAR2 (15)
comments VARCHAR2 (500)
F * experiment_accession VARCHAR2 (15)
F * expsample_accession VARCHAR2 (15)
repository_accession VARCHAR2 (20)
F repository_name VARCHAR2 (50)
F study_accession VARCHAR2 (15)
study_time_collected REAL
study_time_collected_unit VARCHAR2 (25)
F subject_accession VARCHAR2 (15)
F unit_preferred VARCHAR2 (50)
unit_reported VARCHAR2 (200)
value_preferred REAL
value_reported VARCHAR2 (50)
F workspace_id INTEGER
hla_typing_result
P * result_id INTEGER
allele_1 VARCHAR2 (250)
allele_2 VARCHAR2 (250)
ancestral_population VARCHAR2 (250)
F * arm_accession VARCHAR2 (15)
F * biosample_accession VARCHAR2 (15)
comments VARCHAR2 (500)
F * experiment_accession VARCHAR2 (15)
F * expsample_accession VARCHAR2 (15)
locus_name VARCHAR2 (25)
repository_accession VARCHAR2 (20)
repository_name VARCHAR2 (50)
* result_set_id INTEGER
F * study_accession VARCHAR2 (15)
study_time_collected REAL
study_time_collected_unit VARCHAR2 (25)
F * subject_accession VARCHAR2 (15)
F workspace_id INTEGER
neut_ab_titer_result
P * result_id INTEGER
F * arm_accession VARCHAR2 (15)
F * biosample_accession VARCHAR2 (15)
comments VARCHAR2 (500)
F * experiment_accession VARCHAR2 (15)
F * expsample_accession VARCHAR2 (15)
repository_accession VARCHAR2 (20)
repository_name VARCHAR2 (50)
F study_accession VARCHAR2 (15)
study_time_collected REAL
study_time_collected_unit VARCHAR2 (25)
F * subject_accession VARCHAR2 (15)
F unit_preferred VARCHAR2 (50)
unit_reported VARCHAR2 (200)
value_preferred REAL
value_reported VARCHAR2 (50)
F virus_strain_preferred VARCHAR2 (200)
virus_strain_reported VARCHAR2 (200)
F workspace_id INTEGER
kir_typing_result
P * result_id INTEGER
allele_1 VARCHAR2 (250)
allele_2 VARCHAR2 (250)
F * arm_accession VARCHAR2 (15)
F * biosample_accession VARCHAR2 (15)
comments VARCHAR2 (500)
F * experiment_accession VARCHAR2 (15)
F * expsample_accession VARCHAR2 (15)
* kir_haplotype VARCHAR2 (250)
repository_accession VARCHAR2 (20)
repository_name VARCHAR2 (50)
F * study_accession VARCHAR2 (15)
study_time_collected REAL
study_time_collected_unit VARCHAR2 (25)
F * subject_accession VARCHAR2 (15)
F workspace_id INTEGER

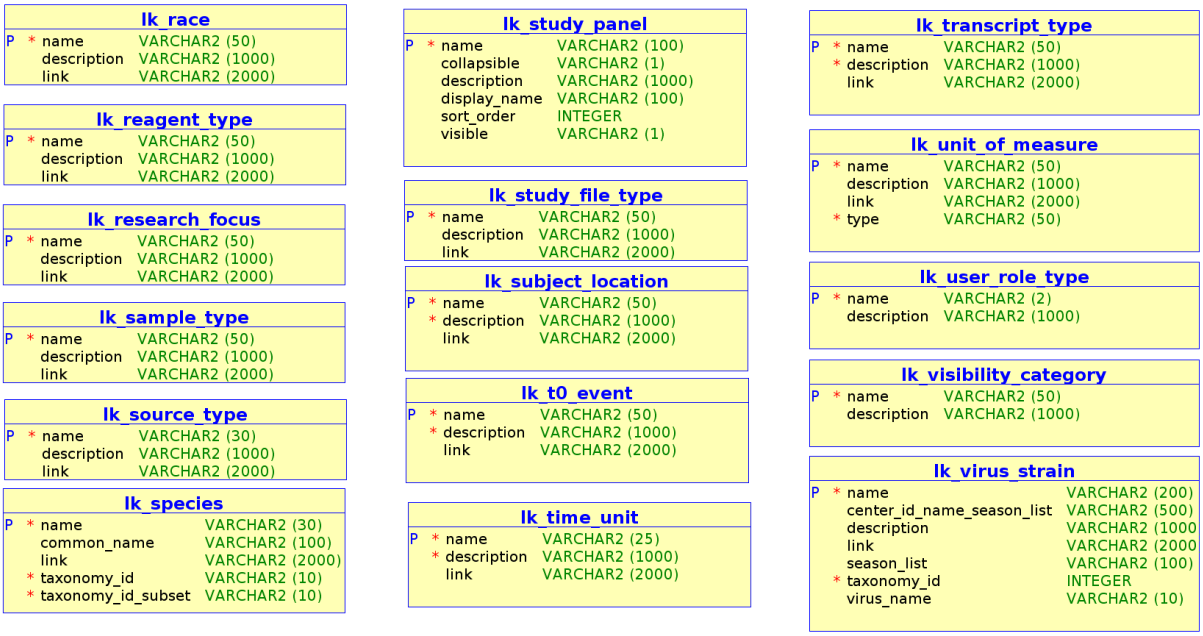
2.1.2.8. Lookup Tables - Part 1

There are approximately 45 tables in the Shared_Data schema that ImmPort identifies as lookup tables, but others may refer to them as controlled vocabulary tables. These tables are used to harmonize data from a study to study. For many of the base tables, ImmPort has both a reported_name and a preferred_name with the preferred name mapped to one of the lookup tables. In addition, several lookup tables are populated using terms from ontologies. Examples of the ontologies used are

1. Vaccine Ontology
2. Disease Ontology
3. Human Phenotype Ontology
4. Uberon Anatomy Ontology
5. Clinical Measurement Ontology
6. Cell Ontology
7. Protein Ontology
8. Gene Ontology
9. IPD-IMGT/HLA
10. NCBI Taxonomy
11. Ontology Biomedical Investigation

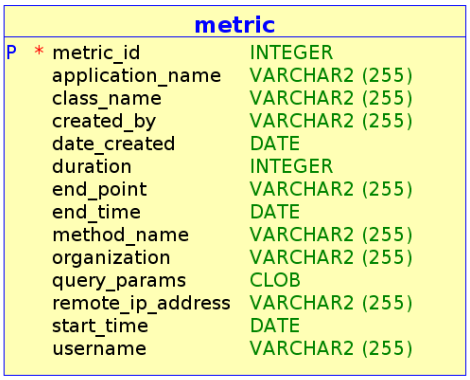

lk_adverse_event_severity
P * name VARCHAR2 (60)
description VARCHAR2 (1000)
link VARCHAR2 (2000)
lk_age_event
P * name VARCHAR2 (40)
* description VARCHAR2 (1000)
link VARCHAR2 (2000)
lk_analyte
P * analyte_accession VARCHAR2 (15)
gene_symbol VARCHAR2 (100)
gene_aliases CLOB
gene_id VARCHAR2 (10)
genetic_nomenclature_id VARCHAR2 (100)
immunology_symbol VARCHAR2 (100)
link VARCHAR2 (2000)
official_gene_name VARCHAR2 (255)
protein_ontology_id VARCHAR2 (15)
protein_ontology_name VARCHAR2 (100)
protein_ontology_synonyms CLOB
protein_ontology_short_label VARCHAR2 (255)
taxonomy_id VARCHAR2 (10)
uniprot_entry VARCHAR2 (20)
uniprot_entry_name VARCHAR2 (255)
lk_ancestral_population
P * name VARCHAR2 (30)
abbreviation VARCHAR2 (3)
* description VARCHAR2 (4000)
link VARCHAR2 (2000)
lk_cell_population_marker
P * name VARCHAR2 (150)
description VARCHAR2 (1000)
link VARCHAR2 (2000)
lk_compound_role
P * name VARCHAR2 (40)
description VARCHAR2 (1000)
link VARCHAR2 (2000)
lk_criterion_category
P * name VARCHAR2 (40)
description VARCHAR2 (1000)
link VARCHAR2 (2000)
lk_data_completeness
P * id INTEGER
description VARCHAR2 (1000)
lk_disease
P * name VARCHAR2 (250)
* disease_ontology_id VARCHAR2 (50)
* description VARCHAR2 (1000)
link VARCHAR2 (2000)
lk_disease_stage
P * name VARCHAR2 (50)
* description VARCHAR2 (1000)
link VARCHAR2 (2000)
lk_ethnicity
P * name VARCHAR2 (50)
description VARCHAR2 (1000)
link VARCHAR2 (2000)
lk_exp_measurement_tech
P * name VARCHAR2 (50)
description VARCHAR2 (1000)
link VARCHAR2 (2000)
lk_lab_test_name
P * name VARCHAR2 (50)
cdisc_lab_test_code VARCHAR2 (50)
description VARCHAR2 (1000)
lab_test_panel_name VARCHAR2 (50)
link VARCHAR2 (2000)
lk_lab_test_panel_name
P * name VARCHAR2 (125)
description VARCHAR2 (1000)
link VARCHAR2 (2000)
lk_exposure_material
P * name VARCHAR2 (200)
* exposure_material_id VARCHAR2 (50)
* description VARCHAR2 (1000)
link VARCHAR2 (2000)
lk_exposure_process
P * name VARCHAR2 (100)
* description VARCHAR2 (1000)
link VARCHAR2 (2000)
lk_expsample_result_schema
P * name VARCHAR2 (50)
description VARCHAR2 (1000)
* table_name VARCHAR2 (30)
lk_gender
P * name VARCHAR2 (20)
description VARCHAR2 (1000)
link VARCHAR2 (2000)
lk_file_detail
P * name VARCHAR2 (100)
description VARCHAR2 (1000)
link VARCHAR2 (2000)
lk_hmdb
P * hmdb_id VARCHAR2 (15)
* name VARCHAR2 (20)
description VARCHAR2 (1000)
link VARCHAR2 (2000)
lk_locus_name
P * name VARCHAR2 (100)
description VARCHAR2 (250)
link VARCHAR2 (2000)
lk_mass_spectrometry_type
P * name VARCHAR2 (50)
description VARCHAR2 (4000)
link VARCHAR2 (2000)
lk_organization
P * name VARCHAR2 (125)
link VARCHAR2 (2000)
lk_personnel_role
P * name VARCHAR2 (40)
description VARCHAR2 (1000)
link VARCHAR2 (2000)
lk_protein_name
P * name VARCHAR2 (255)
* uniprot_id VARCHAR2 (50)
uniprot_gene_name VARCHAR2 (255)
description VARCHAR2 (4000)
link VARCHAR2 (2000)
lk_plate_type
P * name VARCHAR2 (50)
description VARCHAR2 (1000)
link VARCHAR2 (2000)
lk_protocol_type
P * name VARCHAR2 (100)
description VARCHAR2 (1000)
link VARCHAR2 (2000)
lk_public_repository
P * name VARCHAR2 (50)
description VARCHAR2 (1000)
link VARCHAR2 (2000)

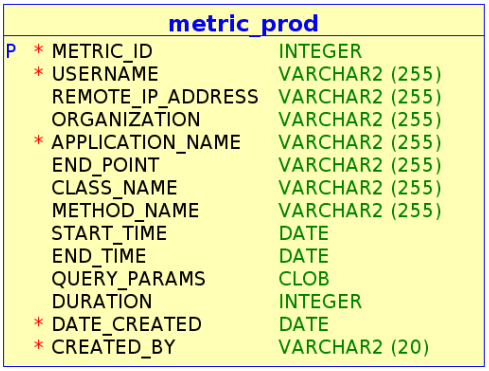
2.1.2.9. Lookup Tables - Part 2

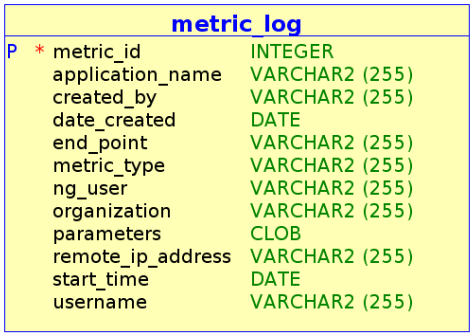


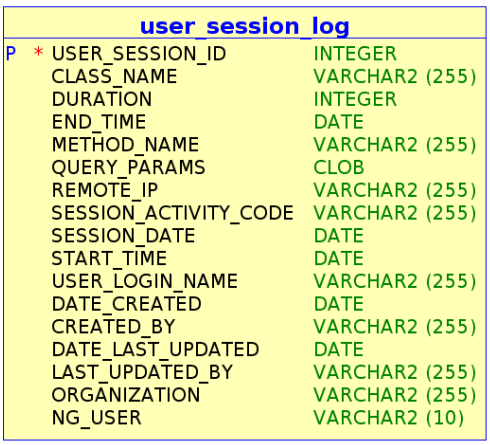
2.1.3. Metric

2.1.3.1. Metric



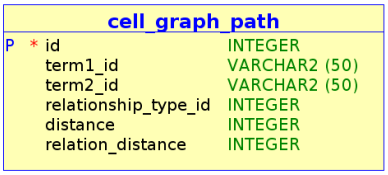


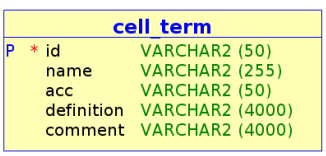


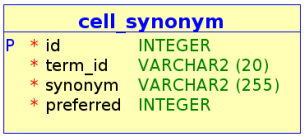


2.1.4. Cell Ontology

The three tables in this database represent information parsed from the cl.obo file. The **cell_term** table captures base information for all cell terms. The **cell_synonym** table contains names used for a cell term. The information in **cell_term** and **cell_synomym** are merged and the content is indexed using ElasticSearch to support the search capability in the application and is displayed in cell detail pop-ups on hover in the application. The **cell_graph** table supports the generation of the force-directed graph displayed in the application which shows the relationship between cell terms.







2.2. HOSTED APPLICATIONS

2.2.1. ImmPort Shared Data

ImmPort Shared Data is the application utilized by researchers to identify studies of interest for further exploration and analysis. As of December 31, 2020, 462 studies have been shared and cataloged in ImmPort Shared Data. Currently, no row-level results are viewable from within ImmPort Shared Data with the advantage that no user authorization is required. Viewing row-level data would require authorization/user login. ImmPort Shared Data utilizes a standard suite of Linux-based AWS servers and components described below and a Spring/Java and Angular web application architecture.

2.2.1.1. Feature Summary

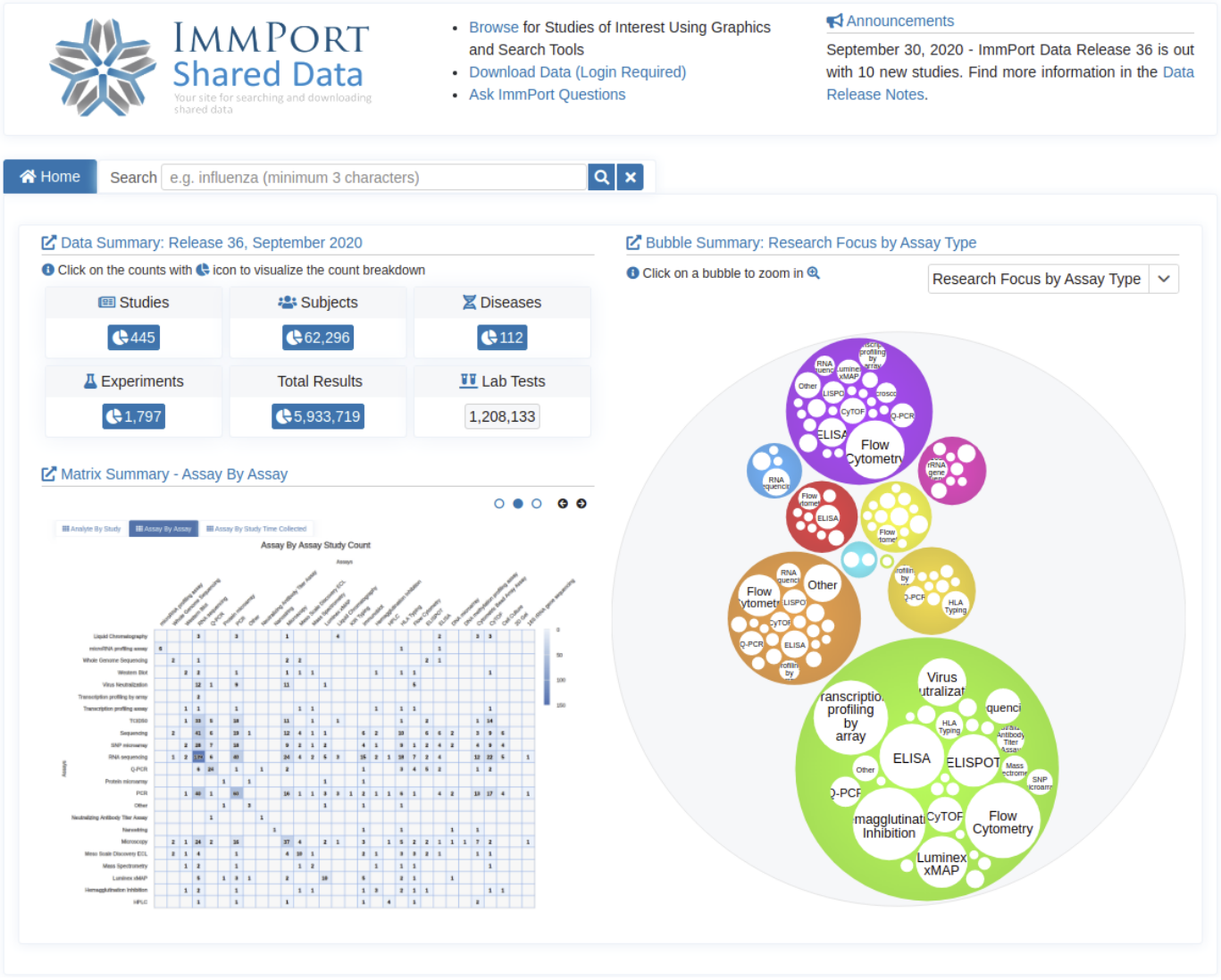
Table 2.4.1.1-1 below summarizes the major functionality of ImmPort Shared Data.

Table 2.4.1.1-1: Summary of ImmPort Capabilities and Features

#	Capabilities/Features	Capability/Feature Description
1	Search for Studies	Allows users to perform a “Google-like” search to identify shared studies of interest. Text entered by the user is searched against an index of the entire set of metadata stored in the ImmPort database.
2	View/Filter Query Results	After an initial search, users are presented with a list of studies, summary information about each study, and the search hits. From here users may use facets on the left-hand panel to further filter studies returned by categories of data such as assay method, species, sample type, etc.
3	View Study Details	Once a study of interest is identified, users are able to view extensive metadata and summary data about the study.
4	Visualize Summary Data	For selected aspects of the study data such as demographics, users are can filter and visualize data in standard plots such as bar charts by factors such as gender, ethnicity, and arm
5	Link to download data	For a given study, users can click to download the raw study data and are redirected to log in to the Data Browser application at NIAID.
6	View Reference and Static content	Static content such as tutorials, curated cytokine lists, system documentation, user documentation and ImmPort project information are hosted on the ImmPort Shared Data site.

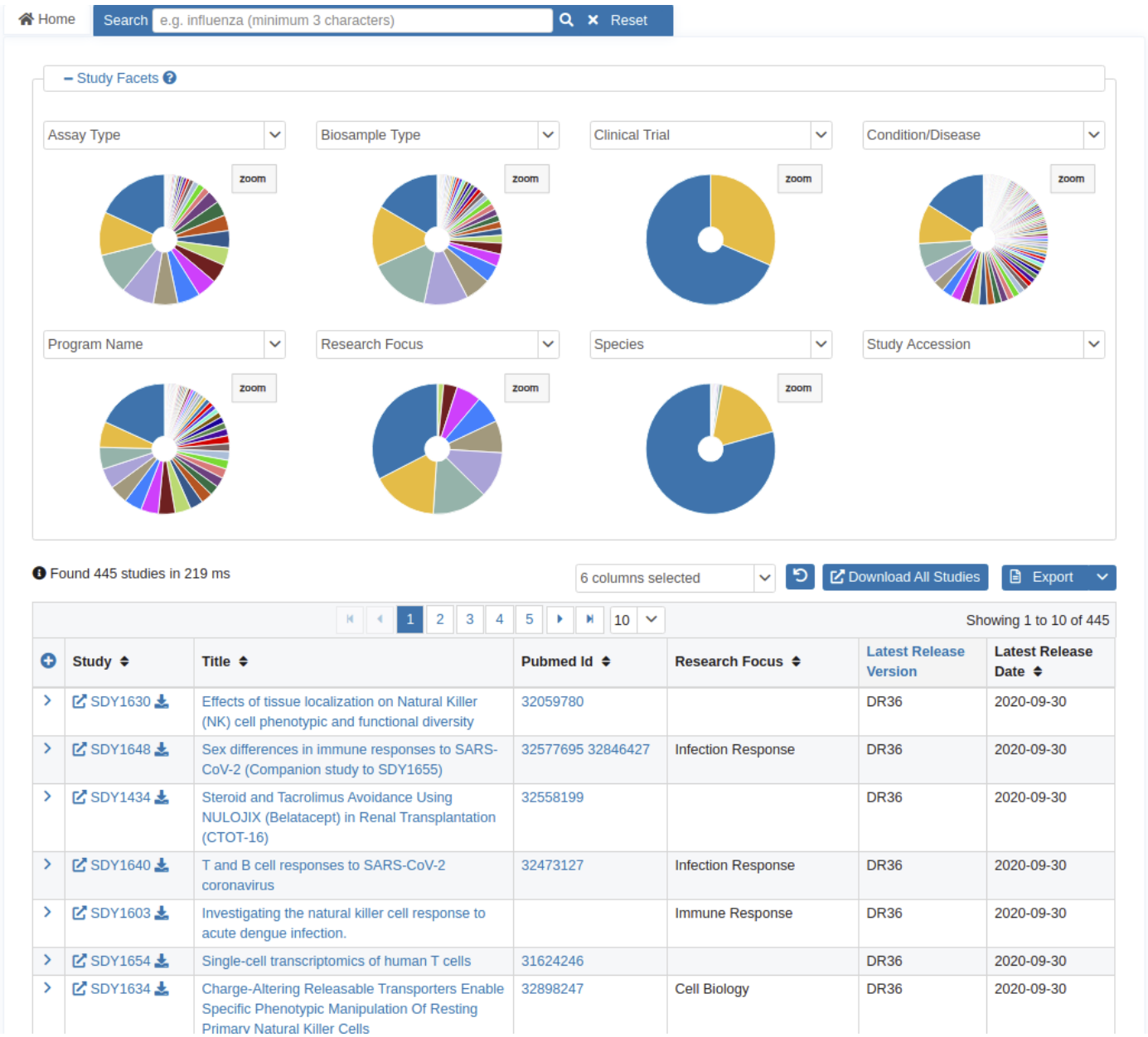
2.2.1.1.1. Feature 1: Home Page

The ImmPort Shared Data home page provides the entry-point into ImmPort Shared Data. Beyond being an information page containing announcements and ImmPort background information, users can right away begin searching and filtering for studies of interest. In addition, studies that may be of interest to the user community are highlighted in the slider bar based on factors such as recent publications, the popularity of the study based on views or downloads, and recent additions to the shared ImmPort content.



2.2.1.1.2. Feature 2: Search Bar and Results

Once the user has selected to view either all or a subset of studies a list of studies is presented as shown below. The page layout is a familiar design with faceted search capabilities illustrated in the left-hand panel, and a “Google-like” simple text search bar at the top. From this page, users may either select a study to view more details, filter the study list further based on facets, click to view a larger version of the study schematic graphic or click to download the study data. Clicking the Download button directs the user to the Data Browser application in the directory for that study after authentication.

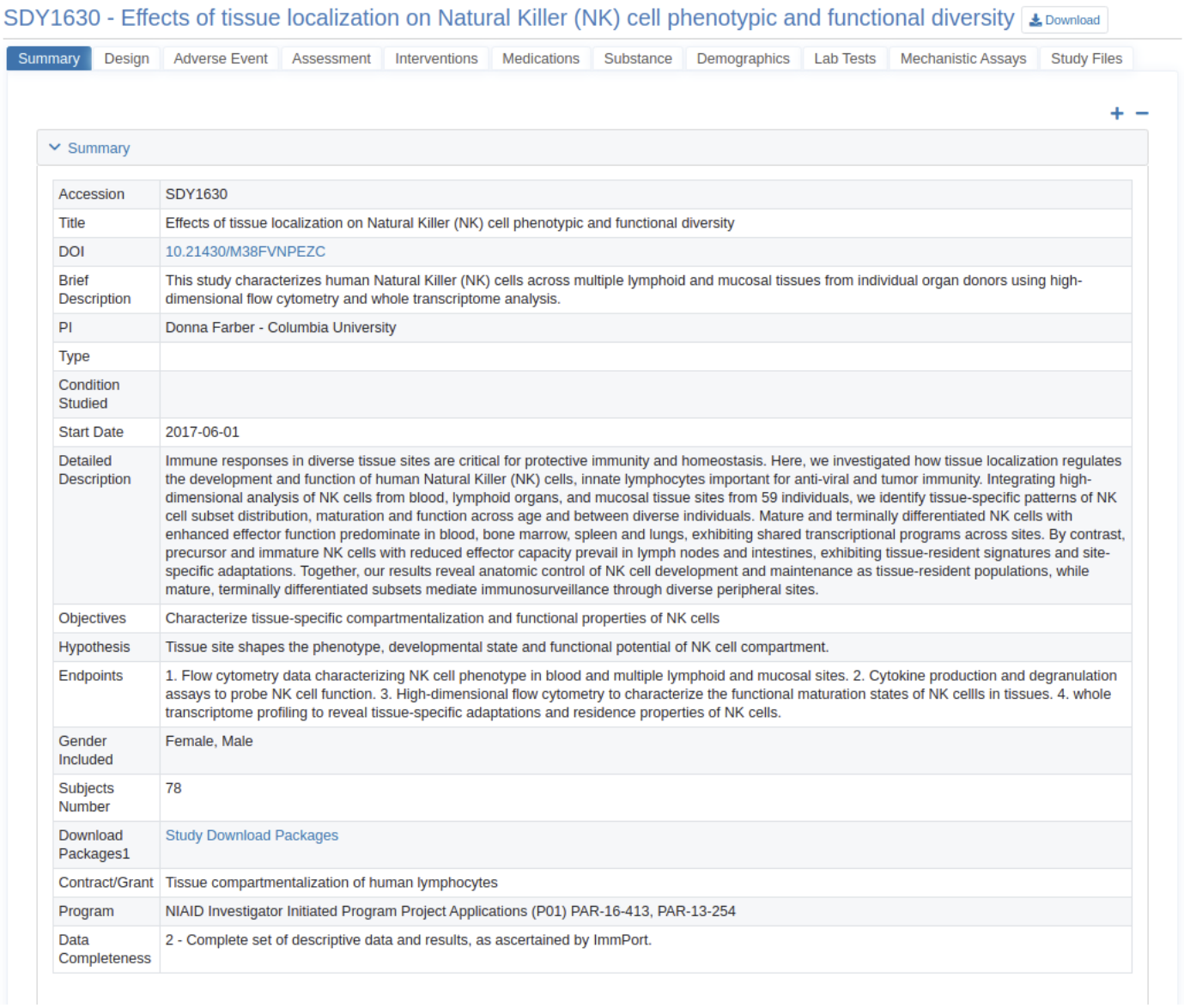


Study	Title	Pubmed Id	Research Focus	Latest Release Version	Latest Release Date
SDY1630	Effects of tissue localization on Natural Killer (NK) cell phenotypic and functional diversity	32059780		DR36	2020-09-30
SDY1648	Sex differences in immune responses to SARS-CoV-2 (Companion study to SDY1655)	32577695 32846427	Infection Response	DR36	2020-09-30
SDY1434	Steroid and Tacrolimus Avoidance Using NULOJIX (Belatacept) in Renal Transplantation (CTOT-16)	32558199		DR36	2020-09-30
SDY1640	T and B cell responses to SARS-CoV-2 coronavirus	32473127	Infection Response	DR36	2020-09-30
SDY1603	Investigating the natural killer cell response to acute dengue infection.		Immune Response	DR36	2020-09-30
SDY1654	Single-cell transcriptomics of human T cells	31624246		DR36	2020-09-30
SDY1634	Charge-Altering Releasable Transporters Enable Specific Phenotypic Manipulation Of Resting Primary Natural Killer Cells	32898247	Cell Biology	DR36	2020-09-30

2.2.1.1.3. Feature 3: Study Detail Page

When a study is selected, the user is presented with a study detail page. Within this page, there are multiple tabs presented with different aspects of study data based on what has been provided. The full set of tabs that may be displayed include

- Summary: title, description, PI, type, arms/cohorts, study schematic
- Study Design: study timeline, inclusion and exclusion criteria, schedule of events
- Adverse Event: summary of adverse event data by severity, name, arm
- Assessment: summary of assessments taken
- Interventions: summary of interventions performed
- Medications: concomitant medications taken
- Demographics: summary information about gender, age, ethnicity
- Lab Tests: summary of laboratory panels and tests
- Mechanistic Assays: summary of assays performed, protocols, platforms, reagents, treatments
- Study Files: catalog of study data files provided



Accession	SDY1630
Title	Effects of tissue localization on Natural Killer (NK) cell phenotypic and functional diversity
DOI	10.21430/M38FVNPEZC
Brief Description	This study characterizes human Natural Killer (NK) cells across multiple lymphoid and mucosal tissues from individual organ donors using high-dimensional flow cytometry and whole transcriptome analysis.
PI	Donna Farber - Columbia University
Type	
Condition Studied	
Start Date	2017-06-01
Detailed Description	Immune responses in diverse tissue sites are critical for protective immunity and homeostasis. Here, we investigated how tissue localization regulates the development and function of human Natural Killer (NK) cells, innate lymphocytes important for anti-viral and tumor immunity. Integrating high-dimensional analysis of NK cells from blood, lymphoid organs, and mucosal tissue sites from 59 individuals, we identify tissue-specific patterns of NK cell subset distribution, maturation and function across age and between diverse individuals. Mature and terminally differentiated NK cells with enhanced effector function predominate in blood, bone marrow, spleen and lungs, exhibiting shared transcriptional programs across sites. By contrast, precursor and immature NK cells with reduced effector capacity prevail in lymph nodes and intestines, exhibiting tissue-resident signatures and site-specific adaptations. Together, our results reveal anatomic control of NK cell development and maintenance as tissue-resident populations, while mature, terminally differentiated subsets mediate immunosurveillance through diverse peripheral sites.
Objectives	Characterize tissue-specific compartmentalization and functional properties of NK cells
Hypothesis	Tissue site shapes the phenotype, developmental state and functional potential of NK cell compartment.
Endpoints	1. Flow cytometry data characterizing NK cell phenotype in blood and multiple lymphoid and mucosal sites. 2. Cytokine production and degranulation assays to probe NK cell function. 3. High-dimensional flow cytometry to characterize the functional maturation states of NK cellls in tissues. 4. whole transcriptome profiling to reveal tissue-specific adaptations and residence properties of NK cells.
Gender Included	Female, Male
Subjects Number	78
Download Packages1	Study Download Packages
Contract/Grant	Tissue compartmentalization of human lymphocytes
Program	NIAID Investigator Initiated Program Project Applications (P01) PAR-16-413, PAR-13-254
Data Completeness	2 - Complete set of descriptive data and results, as ascertained by ImmPort.

2.2.1.2. Hardware and Software Architecture Components

Table 2.4.1.2-1 below describes the hardware components and software components that constitute the architecture of ImmPort Shared Data.

Table 2.4.1.2-1: Hardware and Software Architecture Components

Component Name	
JavaScript Object Notation (JSON)	A lightweight, human-readable data-interchange format widely used in web applications and web-based API's.
Spring Boot	Provides a ready to start Spring-based application deployment that utilizes best practices for the configuration (http://projects.spring.io/spring-boot/)
Spring Web MVC	Provides model-view-controller architecture (MVC) and components to develop flexible and loosely coupled web applications in the Spring family of components
AWS Aurora	A widely used open-source relational database system now owned by Oracle corporation. www.mysql.com
Angular 8+	A popular environment with interactive components for developing dynamic web-based applications using AJAX/JavaScript developed by Google but open source. https://angularjs.org
AWS ElasticSearch	Popular open source enterprise search platform built on Apache Lucene™ utilizing Apache Tomcat as the servlet container (lucene.apache.org)
Google Analytics	Web monitoring tool framework provided by Google.
Amazon Web Services (AWS)	Amazon Web Services (AWS) provides cloud infrastructure and a wide array of server capabilities for developers to build production applications. (aws.amazon.com)
Elastic Compute Cloud (EC2)	Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud, designed to make development easier through ease of server creation, duplication, and shutdown along with providing a suite of configuration options for hardware and software/OS specifications.
Virtual Private Cloud (VPC)	Allows administrators to provision a logically isolated section of the AWS cloud where AWS resources can be created and launched in a customized virtual network
Simple Storage Service (S3)	Provides secure, durable, highly-scalable cloud storage.
Relational Database Service (RDS)	Provides pre-configured database servers for Oracle, SQL Server, PostgreSQL, MySQL, and MariaDB where AWS performs the database administration allowing the development team to focus energies on application-specific details and development.

2.2.1.3. Data Architecture

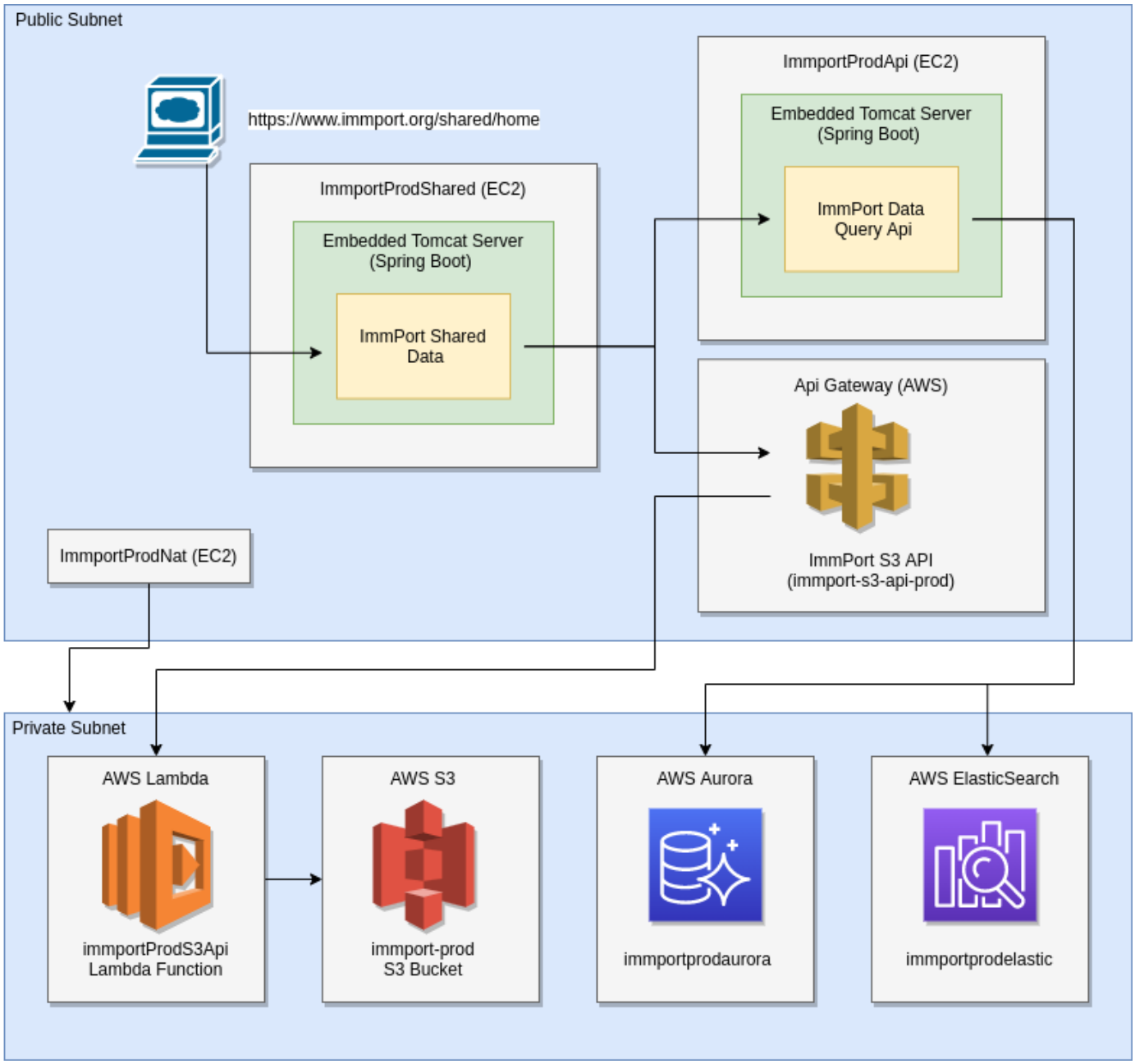
Data used by the ImmPort Shared Data application is stored and retrieved using multiple technologies. Primarily data is stored in an AWS Aurora MySQL database. The Shared Data schema maps closely to the Oracle production operational database but has been partly de-normalized to optimize query performance and to support the Data Query API. The database contains all information for all studies shared for public access.

An AWS ElasticSearch index is used to support free text and faceted searching of study information and to support site search. The content index is updated with each quarterly release. ImmPort also uses AWS S3 buckets to host content generated as part of the Data Release process and remains static for each release.

An overview of the ImmPort Shared Data model is available on the website on the Data Model page. The table and column documentation is available on the website on the Relational Database page.

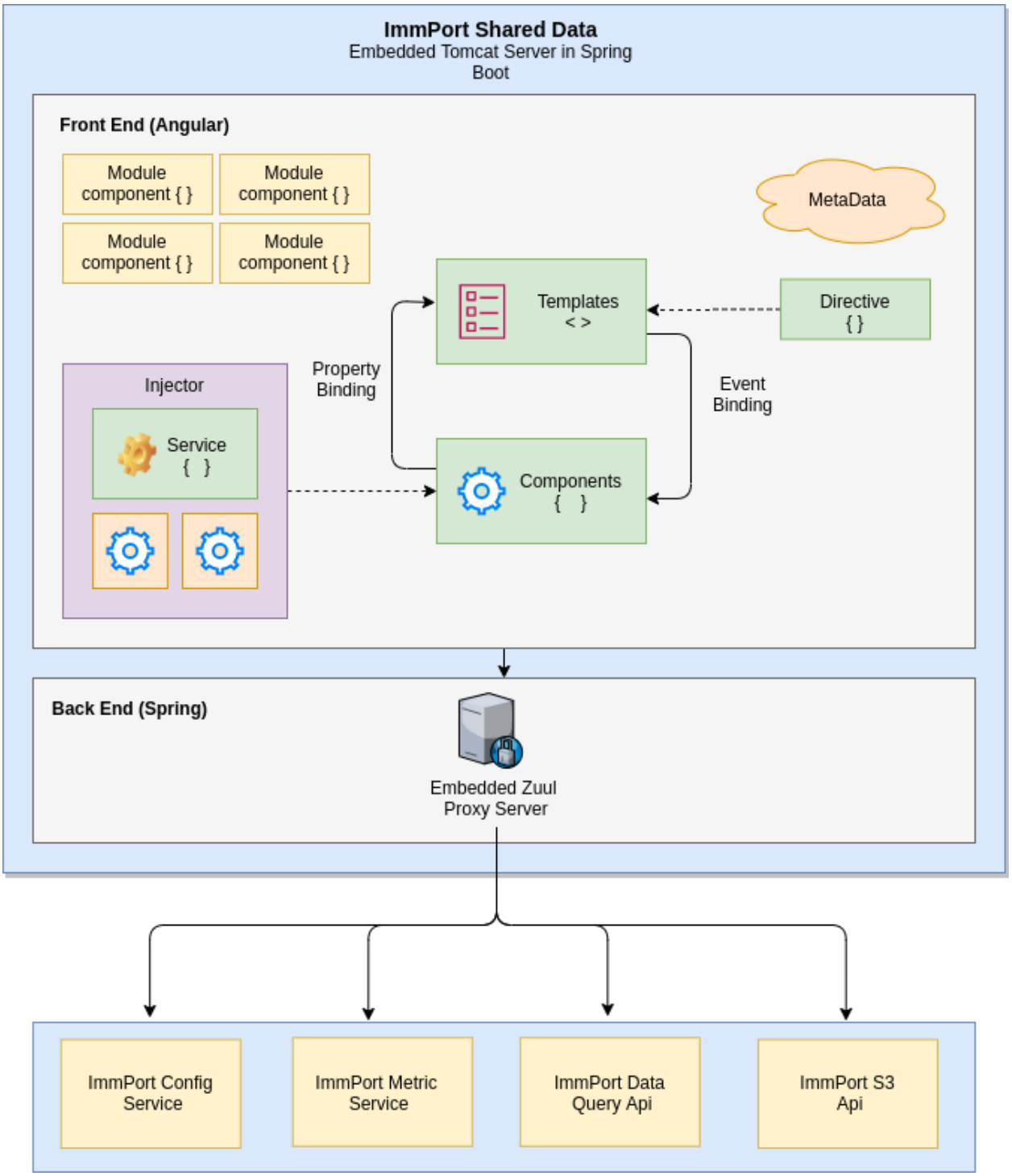
2.2.1.4. ImmPort Shared Data Server Architecture

ImmPort Shared Data, on AWS, is separated into its own private network or VPC. The outward-facing Tomcat web server is contained in a public subnet where the site www.immport.org is directed while data servers are contained in a private subnet only available through the NAT EC2 server within the public subnet. This separation limits exposure to security threats in the data tier of ImmPort Shared Data. The configuration follows the AWS recommendations described at http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Scenario2.html



2.2.1.5. ImmPort Shared Data Software Architecture

ImmPort Shared Data is a standard Java Spring-based web application. The client layer utilizes JavaScript AJAX frameworks such as JQuery and Angular to provide interactive graphical user interfaces. The Model, View, and Controller uses Spring Web MVC with Tiles/JSP to layout the web pages. Hibernate provides the data and persistence layer to the MySQL relational database via JDBC. Queries against SOLR are run through the web service connecting to the SOLR Tomcat server.



2.2.2. Study Search

2.2.2.1. Fields

When a word is entered in the study search box. The following fields are searched:

study_accession^200,
study_accession.ngram^100,
brief_title^150,
brief_title.ngram^100,
brief_description^100,
brief_description.ngram^30,
description^10,
description.ngram^5,
doi^100,
doi.ngram^30,
endpoints^50,
endpoints.ngram^20,
hypothesis^50,
hypothesis.ngram^20,
objectives^50,
objectives.ngram^20,
official_title^75,
official_title.ngram^20,
sponsoring_organization^50,
sponsoring_organization.ngram^20,
research_focus^5,
arm_or_cohort_all^5,
arm_or_cohort_all.ngram^2,
biosample.type^5,
contract_grant.name^5,
contract_grant.name.ngram^2,
contract_grant.external_id^5,
program.program_name^5,
program.program_name.ngram^2,
condition_preferred^50,
condition_preferred.ngram^20,
condition_reported^50,
condition_reported.ngram^20,
experiment_all.ngram^2,
pubmed_all.ngram^2,
personnel_all.ngram^2,
elisa_result_all.ngram^2,
elispot_result_all.ngram^2,
fcs_analyzed_result_all.ngram^2,
hai_result_all.ngram^2,
hla_typing_result_all.ngram^2,
kir_typing_result_all.ngram^2,

mbaa_result_all.ngram^2,
neut_ab_titer_result_all.ngram^2,
pcr_result_all.ngram^2,
reagent_all.ngram^2,
adverse_event_all.ngram^2

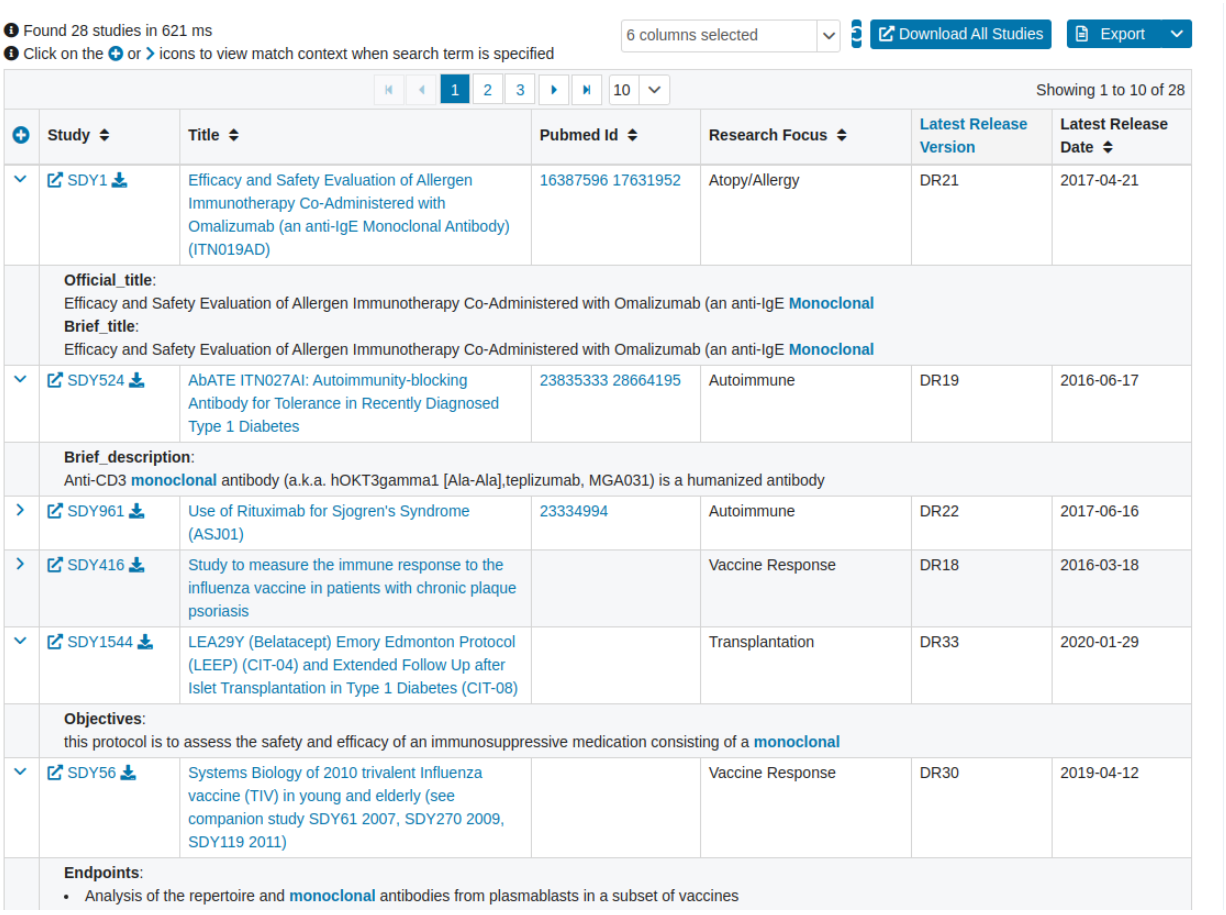
2.2.2.1. Aggregated Fields

The fields ending with "_all" is a collection of the data of all the fields in that category. For example pcr_result_all is the data from the following fields:

pcr_result.gene_id
pcr_result.gene_name
pcr_result.gene_symbol_preferred
Pcr_result.gene_symbol_reported

2.2.2.2. Boosting

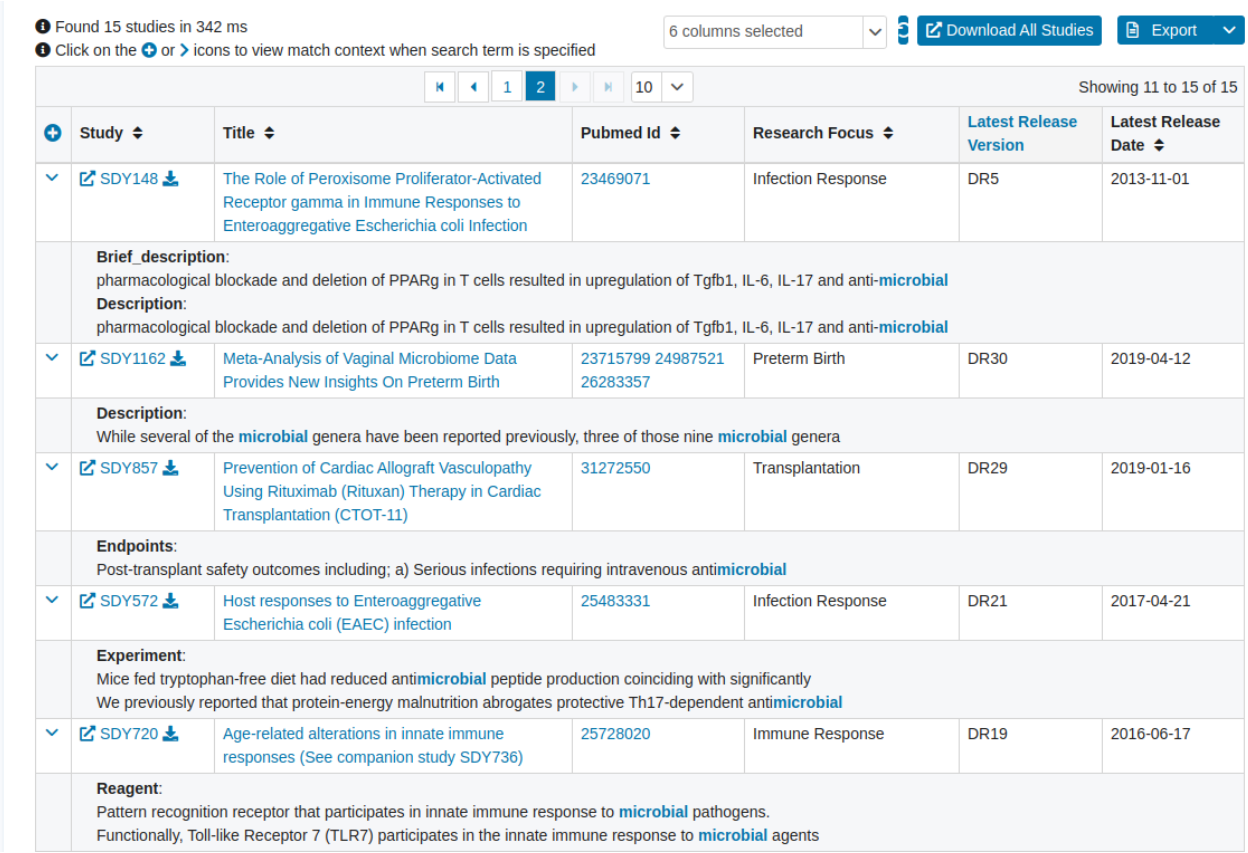
Individual fields can be boosted with the caret (^) notation. Matches on the study_accession (boost : 200), brief_title (boost : 150), brief_description (boost : 100),official_title (boost : 75) have more relevance than the other fields since their boost factor is more. Example: If you search for the word "Monoclonal",



SDY1 gets the highest score because the word “Monoclonal” was found in brief_title (150) and official_titile (75) both have a high boost value.
SDY524 - has the second-highest score since the word “Monoclonal” was found in brief_description (100)
SDY1544 and SDY56 come later since Objectives (50) and EndPoints (50) have lower boost values

2.2.2.3. Ngram search:

Ngrams helps to search whether a term belongs to a word fully or partially. For example, if you search for “microbial”


Found 15 studies in 342 ms
Click on the or icons to view match context when search term is specified
6 columns selected
Download All Studies
Export
1 2 10
Showing 11 to 15 of 15

Study	Title	Pubmed Id	Research Focus	Latest Release Version	Latest Release Date
SDY148	The Role of Peroxisome Proliferator-Activated Receptor gamma in Immune Responses to Enteroaggregative Escherichia coli Infection	23469071	Infection Response	DR5	2013-11-01
Brief_description: pharmacological blockade and deletion of PPARg in T cells resulted in upregulation of Tgfb1, IL-6, IL-17 and anti-**microbial** **Description**: pharmacological blockade and deletion of PPARg in T cells resulted in upregulation of Tgfb1, IL-6, IL-17 and anti-**microbial**					
SDY1162	Meta-Analysis of Vaginal Microbiome Data Provides New Insights On Preterm Birth	23715799 24987521 26283357	Preterm Birth	DR30	2019-04-12
Description: While several of the **microbial** genera have been reported previously, three of those nine **microbial** genera					
SDY857	Prevention of Cardiac Allograft Vasculopathy Using Rituximab (Rituxan) Therapy in Cardiac Transplantation (CTOT-11)	31272550	Transplantation	DR29	2019-01-16
Endpoints: Post-transplant safety outcomes including; a) Serious infections requiring intravenous anti**microbial**					
SDY572	Host responses to Enteroaggregative Escherichia coli (EAEC) infection	25483331	Infection Response	DR21	2017-04-21
Experiment: Mice fed tryptophan-free diet had reduced anti**microbial** peptide production coinciding with significantly We previously reported that protein-energy malnutrition abrogates protective Th17-dependent anti**microbial**					
SDY720	Age-related alterations in innate immune responses (See companion study SDY736)	25728020	Immune Response	DR19	2016-06-17
Reagent: Pattern recognition receptor that participates in innate immune response to **microbial** pathogens. Functionally, Toll-like Receptor 7 (TLR7) participates in the innate immune response to **microbial** agents					

SDY148 - anti-microbial (partial)
SDY1162 - microbial (full)
SDY857 - antimicrobial (partial)

2.2.2.4. Phrase search

If a phrase is searched without the double quotes then the results will use the best_fields type search. The best_fields type is most useful when you are searching for multiple words best found in the same field. For instance “brown fox” in a single field as a phrase is more meaningful than “brown” in one field and “fox” in the other and “brown” and “fox” separated by other words in a single field. For example, if the phrase is ‘clinical islet transplantation’ , 170 studies were found since first the studies with the phrase is found and then individual words are searched

Study	Title	Pubmed Id	Research Focus	Latest Release Version	Latest Release Date
SDY1432	Clinical Islet Transplantation Consortium (CITC)	23630300 24085506 24691031 25524910 25629445 27208344 27325286 27465220 27571180 28319051 29563196 30211831	Transplantation	DR30	2019-04-12

Condition_reported:
Pancreatic **Islet Transplantation**
Brief_description:
All studies treated participants with up to 3 separate infusions of **islet**s.
Description:
All studies treated participants with up to 3 separate infusions of **islet**s.
Objectives:
The studies investigated methods to improve the safety and efficacy of **islet transplantation** in treating
Condition_preferred:
Pancreatic **Islet Transplantation**
Hypothesis:
Islet transplantation is safe and effective for the treatment of patients with type 1 diabetes.
Official_title:
Clinical Islet Transplantation Consortium (CITC)
Brief_title:
Clinical Islet Transplantation Consortium (CITC)
Pubmed:
Improvement in b-cell secretory capacity after human **islet transplantation** according to the CIT07 protocol
Improvement in insulin sensitivity after human **islet transplantation** for type 1 diabetes.
Restoration of Glucose Counterregulation by **Islet Transplantation** in Long-standing Type 1 Diabetes.
Positron Emission Tomography to Assess the Outcome of Intraportal **Islet Transplantation**.
National Institutes of Health-Sponsored **Clinical Islet Transplantation** Consortium Phase 3 Trial: Manufacture
Contract_grant.name:
CLINICAL ISLET TRANSPLANTATION: DATA COORDINATING CENTER
INNATE IMMUNITY IN **CLINICAL ISLET TRANSPLANTATION**
CLINICAL REFINEMENT OF **ISLET TRANSPLANTATION**
CLINICAL ISLET TRANSPLANTATION AT NORTHWESTERN
CLINICAL ISLET TRANSPLANTATION: **CLINICAL** CENTERS
Program.program_name:
Clinical Islet Transplantation Consortium RFA-DK-09-501

SDY1432 came up first since clinical islet transplantation was found in brief title(boost value 150)

⌄	SDY1544	LEA29Y (Belatacept) Emory Edmonton Protocol (LEEP) (CIT-04) and Extended Follow Up after Islet Transplantation in Type 1 Diabetes (CIT-08)		Transplantation	DR33	2020-01-29

Condition_reported:
Pancreatic **Islet Transplantation**
Endpoints:
study is the proportion of insulin-independent subjects at day 75 (+/- 5 days) following the first **islet**
HbA1c <7.0% AND free of severe hypoglycemic events from Day 28 to Day 365, inclusive, after the first **islet**
HbA1c <7.0% AND free of severe hypoglycemic events from Day 28 to Day 365, inclusive, after the final **islet**
CIT-08: The primary endpoint is duration of sustained **islet** allograft function as determined by evidence
c-peptide level greater than or equal to 0.3 ng/mL at 0, 60, or 90 minutes will be considered evidence of **islet**
Brief_description:
CIT-04: This trial is a prospective, two-center, open-label, pilot study of **islet transplantation** assessing
purpose of this protocol is to collect long-term follow-up information on the safety and efficacy of **islet transplantation** in CIT subjects after their completion in their CIT parent study.
Description:
All studies treated participants with up to 3 separate infusions of **islet**s.
Objectives:
a monoclonal antibody IL-2 receptor blocker (basiliximab), belatacept and mycophenolate mofetil in **islet transplantation**.
The secondary objective is to assess **islet** graft function in the absence of calcineurin inhibitor drugs
proportion of patients attaining and maintaining insulin independence after receiving a maximum of 3 **islet**
primary objective is to provide extended follow-up for safety and efficacy and to support continued **islet**
Condition_preferred:
Pancreatic **Islet Transplantation**
Hypothesis:
CIT-04 and CIT-08: **Transplantation** of pancreatic **islet**s is a safe and effective treatment, when combined
complete avoidance of the anti-angiogenic drug, sirolimus will create a more favorable environment for **islet**
Official_title:
LEA29Y (Belatacept) Emory Edmonton Protocol (LEEP) (CIT-04) and Extended Follow Up after **Islet Transplantation**
Brief_title:
LEA29Y (Belatacept) Emory Edmonton Protocol (LEEP) (CIT-04) and Extended Follow Up after **Islet Transplantation**
Contract_grant.name:
ADVANCING **ISLET** TRANSPLANTS FOR TYPE 1 DIABETES CARE
ISLET TRANSPLANT - COSTIMULATORY BLOCKADE WITH LEA29Y
CLINICAL REFINEMENT OF **ISLET TRANSPLANTATION**
CLINICAL ISLET TRANSPLANTATION AT NORTHWESTERN
Program.program_name:
Clinical Islet Transplantation Consortium RFA-DK-09-501

SDY1544 came up ninth since clinical islet transplantation was found in contract grant name(boost value 5)

⌄	SDY960	Viral Triggers in Pediatric Lung Transplantation (CTOTC-03)	27941431 28639398 29082660 31216376 32686323	Transplantation	DR29	2019-01-16

Arm_or_cohort:
Pediatric lung **transplantation**
Condition_reported:
Lung **Transplantation**
Pediatric Lung **Transplantation**
Endpoints:
the earliest time to bronchiolitis obliterans syndrome (BOS) or obliterative bronchiolitis (OB), re**transplantation**
The secondary endpoints are times to each of the following events: BOS or OB, re**transplantation** or death
Brief_description:
infections increase the risk of bronchiolitis obliterans syndrome, obliterative bronchiolitis, death or re**transplantation**
Description:
infections increase the risk of bronchiolitis obliterans syndrome, obliterative bronchiolitis, death or re**transplantation**
Condition_preferred:
Lung **Transplantation**
Hypothesis:
RVI) increase the risk of bronchiolitis obliterans syndrome, obliterative bronchiolitis, death or re**transplantation**
Official_title:
Viral Triggers of Alloimmunity and Autoimmunity in Pediatric Lung **Transplantation** (CTOTC-03)
Brief_title:
Viral Triggers in Pediatric Lung **Transplantation** (CTOTC-03)
Pubmed:
Role of Circulating MicroRNAs in the Immunopathogenesis of Rejection After Pediatric Lung **Transplantation**
Anellovirus loads are associated with outcomes in pediatric lung **transplantation**.
Absence of evidence that respiratory viral infections influence pediatric lung **transplantation** outcomes
Epidemiology and Persistence of Rhinovirus in Pediatric Lung **Transplantation**.
Contract_grant.name:
VIRAL TRIGGERS OF ALLOIMMUNITY AND AUTOIMMUNITY IN PEDIATRIC LUNG **TRANSPLANTATION** (CTOTC-03)
Program.program_name:
Clinical Trials in Organ **Transplantation** in Children (CTOT-C) RFA-AI-12-005

SDY960 came up 10th since the phrase was tokenized by space. And the tokenized words were found in fields.
For example, if the phrase is 'Ad35.CS.01' is tokenized and then searched.
If a phrase is searched with the double quotes then the results will use the phrase_prefix type search. The whole phrase will be found in the fields For example if the phrase is "clinical islet transplantation", 8 studies were found. If "Ad35.CS.01' is searched with double quotes only one study shows else 18

2.2.3. ImmPort Data Query API

The Data Query API provides programmatic access to ImmPort Shared Data. This API works as a query tool to access ImmPort descriptive data (metadata) and interpret results for assays such as ELISA, ELISPOT, MBAA (Luminex), HAI, Neutralizing Antibody Titers, HLA and KIR typing, QPCR, flow and mass cytometry-based on various input filter parameters. The API returns a JSON output by default. A tab-separated output can also be returned if format=tsv is passed as a parameter to the specified endpoint. The HTTP method supported by this API is GET for this version of this API.

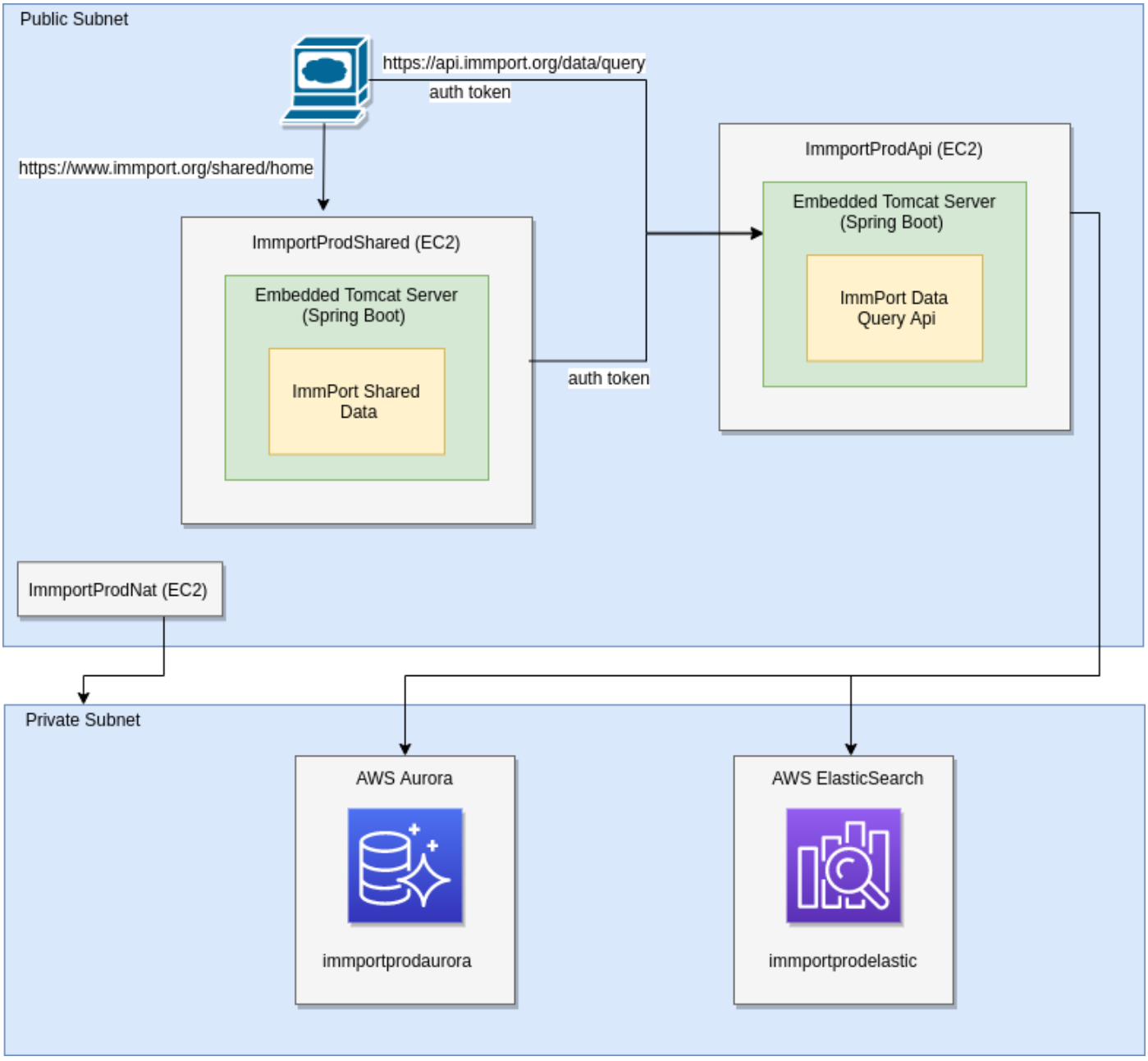
Data Query API endpoints can be accessed directly by a user or by an application. The ImmPort Shared Data application uses some of the endpoints to get data for the search and study detail pages. All requests to the Data Query API require authentication and the Data Query API uses tokens for authentication. Users can obtain tokens by posting to the ImmPort Authentication URL- https://auth.immport.org/auth/token with their username and password. They must include the authentication token as an Authorization: bearer in the custom HTTP header.

2.2.3.1. Feature Summary

This link documents the endpoints of the API https://docs.immport.org/#API/DataQueryAPI/dataqueryapi/

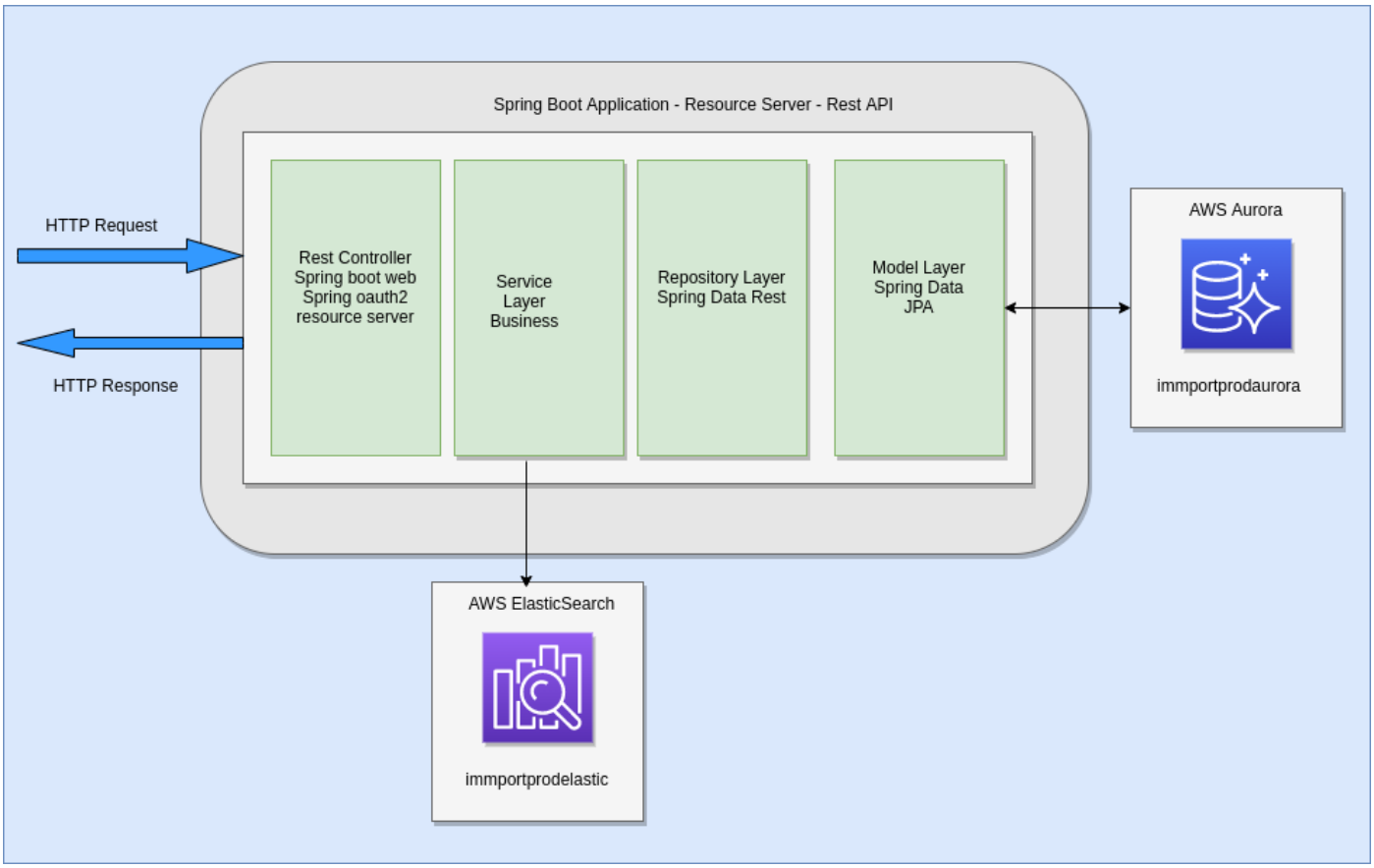
2.2.3.2. ImmPort Data Query API Server Architecture

The ImmPort Data Query API is hosted in AWS on an EC2 instance in the public subnet. The data servers are contained in a private subnet only available through the NAT EC2 server within the public subnet. This separation limits exposure to security threats in the data tier of ImmPort Data Query API. The configuration follows the AWS recommendations described at http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Scenario2.html



2.2.3.3. ImmPort Data Query API Software Architecture

ImmPort Data Query API is a Spring Boot application using Spring JPA which makes it easier to build Spring-powered applications that use data access technologies. It also exposes a study search endpoint that calls a method on the service layer. The service layer builds the elastic search queries based on the parameters sent to the endpoint and then queries the AWS ElasticSearch service that has a study index that contains study JSON files. The Model, Repository, and Service Layers are different java applications. The API layer depends on these applications and packages them during the build process into a single jar file.



2.2.4. ImmPort Data Browser

The Data Browser is a web-based application with the use of similar web frameworks as ImmPort Shared Data described previously. The interactive UI is handled by Angular components using AJAX and JavaScript, while the MVC architecture is implemented in Spring. Aspera Security is utilized when calls are made to download content on the data files and directories identified in the Aspera Node Server. The Aspera Connect Server governs interactions with the data files and directories via the Aspera Connect Client.

2.2.4.1. Feature Summary

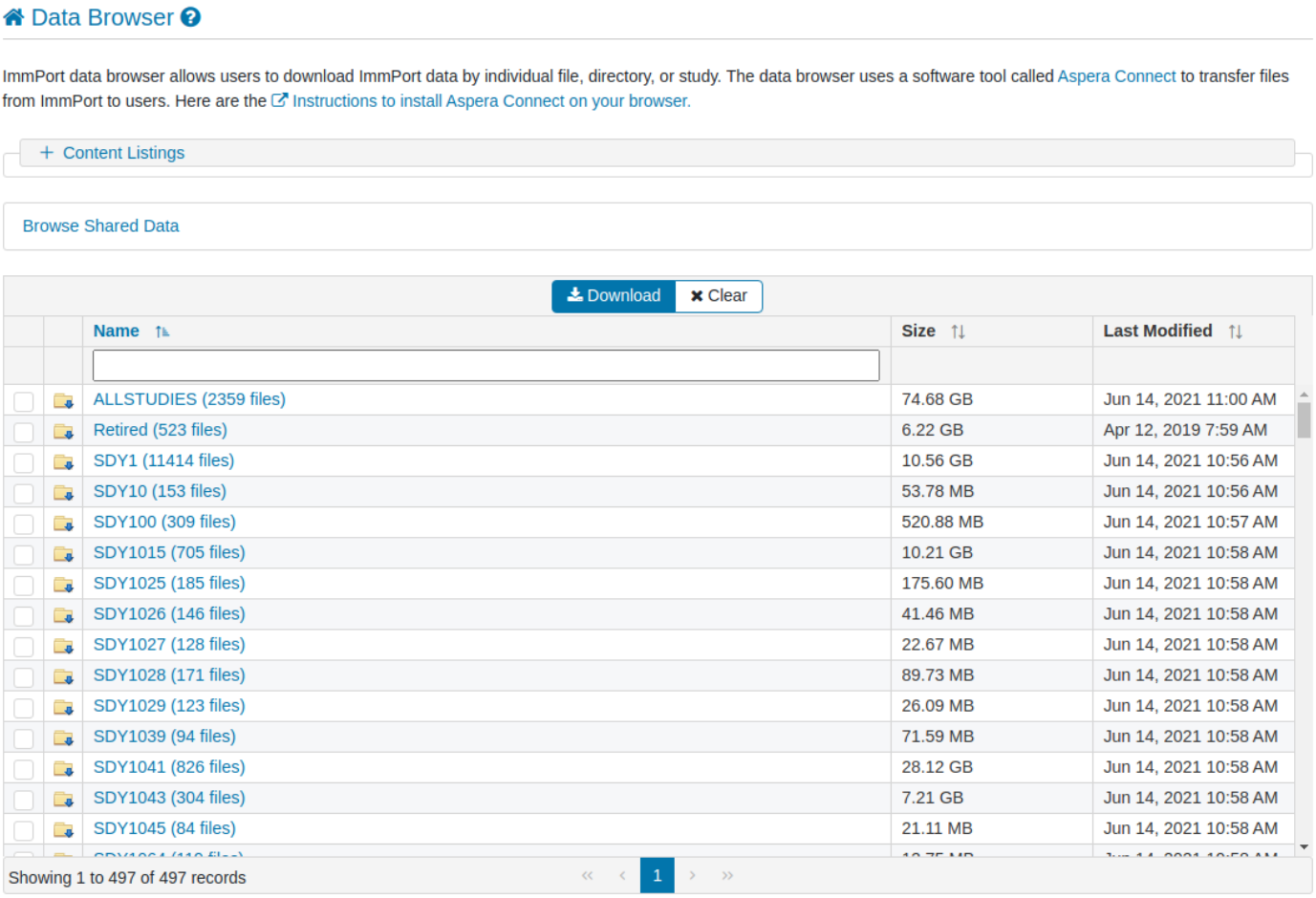
Table 2.4.4.1-1 below summarizes the major functionality of ImmPort Data Browser.

Table 2.4.4.1-1: Summary of ImmPort Capabilities and Features

#	Capabilities/Features	Capability/Feature Description
1	Browse Study Files	Allows users to browse the shared data study files and packages that are available to download in the current data release
2	Download Study Files/Packages	Allows users to select and download study files packages from the latest data release.
3	Download archive	Allows users to download study files from the older versions that are moved into the archives directory.

2.2.4.1.1. Feature 1: Home Page

The Data Browser home page provides a list of studies and files available to download in the latest data release. Users can browse through the list of studies and can select a study directory or pick files across studies to download.



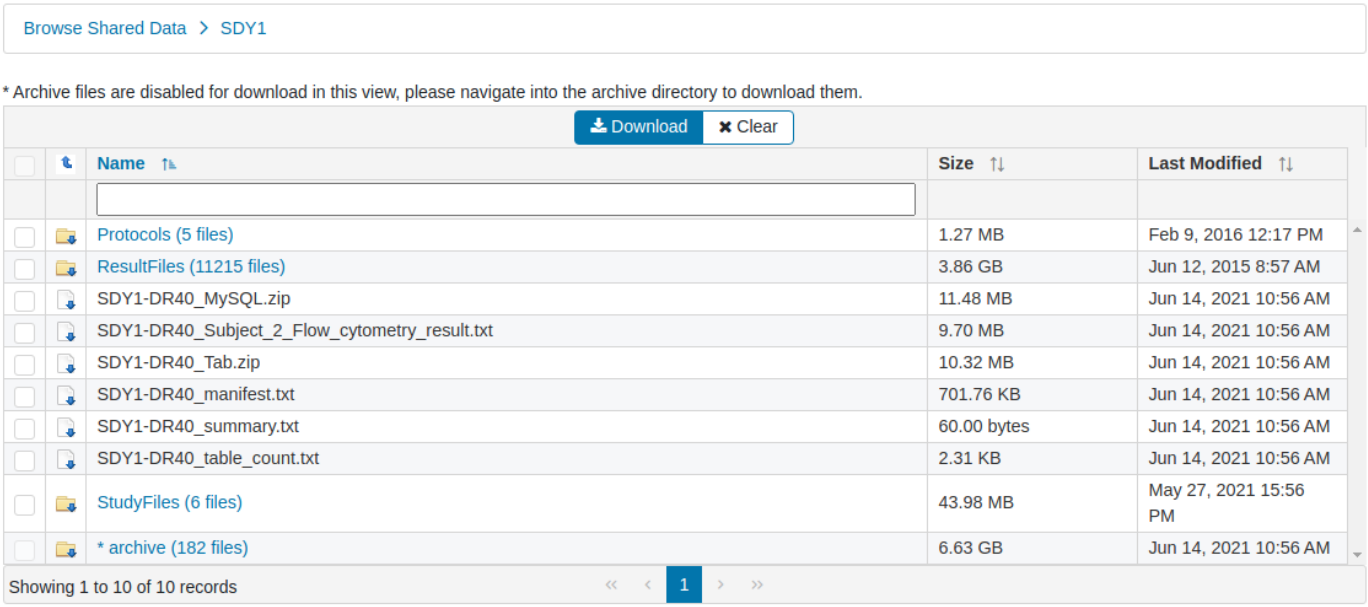
2.2.4.1.2. Feature 2: Study Drill down

Users can drill down into each study to select individual files and packages to download. An example snapshot of data available in each study is below.

Example: SDYxx-DRxx_MySQL.zip is a MySQL dump of the study SDYxx in data release DRxx

Similarly, tab-separated files packaged as a zip file are available to download for each study.

Each study has an archive directory where previous versions of the study can be found. If a user is looking for a study in a particular data release it can be found in the archives directory.

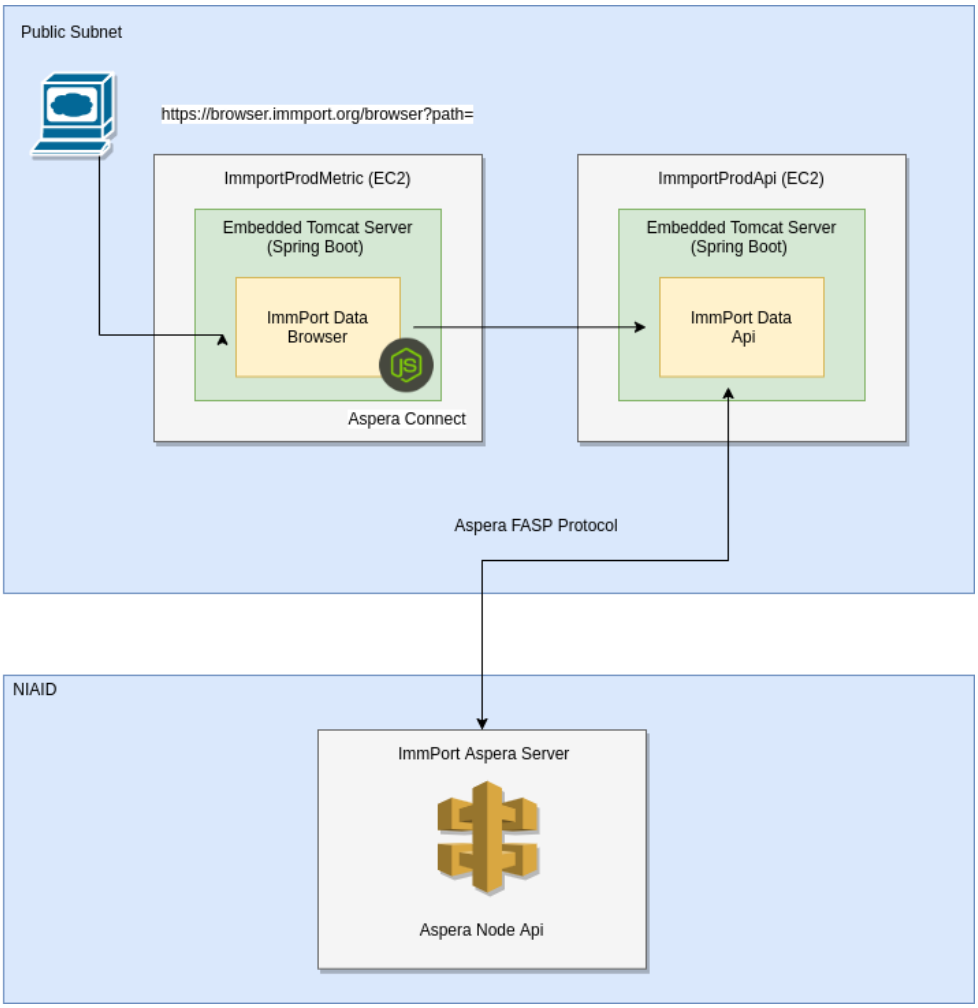


2.2.4.1.3. Feature 3: Aspera Connect Download

The ImmPort data browser allows users to download ImmPort data by individual file, directory, or study. The data browser uses a software tool called Aspera Connect to transfer files from ImmPort to users. This software works with the user's internet browser to quickly and securely transfer files. Aspera Connect requires data downloaders to install the Aspera Connect Client Plugin onto their computer where the files will be downloaded. The Aspera Connect plugin can be downloaded directly through the ImmPort data browser. Here are the instructions for the Aspera Connect Installation https://www.immport.org/installAsperaHelp

Alternatively, the installation files and documentation for the plugin can also be found here http://downloads.asperasoft.com/connect2/

2.2.4.2. ImmPort Data Browser Architecture



2.2.5. ImmPort Data API

The ImmPort Data API is used by the ImmPort Data Browser to browse and download ImmPort shared data on the Aspera Production Server at NIAID. It is also used by ImmPort users who wish to programmatically browse and download ImmPort shared data after obtaining an ImmPort token.

ImmPort Data API allows users to browse and download files and directories on an Aspera Server and to create content listing files for these files and directories (all via POST requests only). ImmPort Data API endpoints are protected and require an ImmPort token to access. ImmPort Data Api fetches an Aspera token internally to talk to the Aspera Server to download the files.

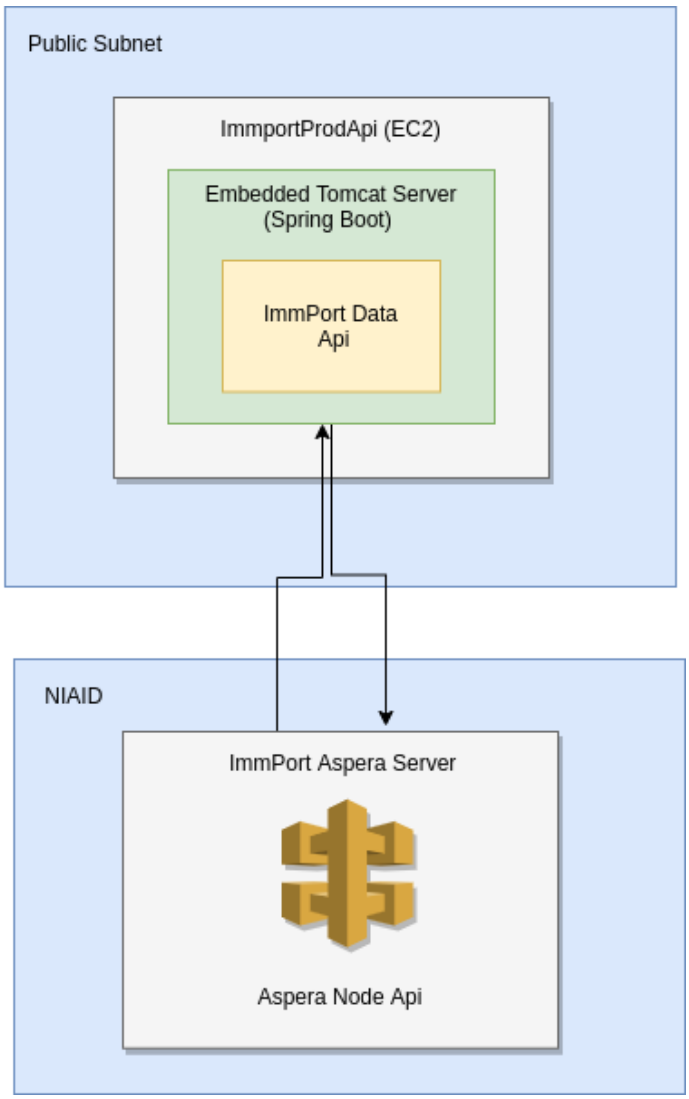
2.2.5.1. Feature Summary

Following are the endpoints available in ImmPort Data API

```
https://api.immport.org/data/content/listing/information
gets the name of the content listing directory and the start
and end times of the content listing file generation.

https://api.immport.org/data/content/listing/creation
creates files containing the content listings sorted
alphabetically, by size, and by last modification date in both ascending
and descending order for each directory.

https://api.immport.org/data/content/listing/report
Returns a a JSON report of the content listing creation

https://api.immport.org/data/list
takes a list of files and directories and downloads them from the aspera server as
a zip package.

https://api.immport.org/data/download/token
Returns an aspera download token

https://api.immport.org/data/download/specification
Returns an aspera download specification
```

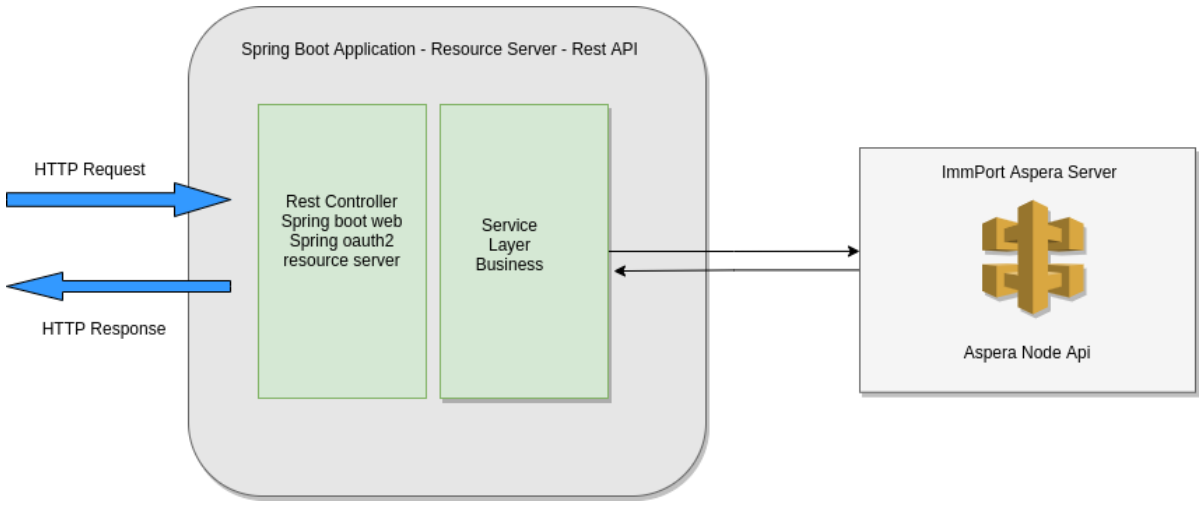
2.2.5.2. ImmPort Data API Server Architecture

The ImmPort Data API is hosted on an EC2 instance in the public subnet on AWS. The Aspera server is hosted on the NIAID On-premises infrastructure.



2.2.5.3. ImmPort Data Query API Software Architecture

ImmPort Data API is a Spring Boot application with a service layer containing a Content Service for listing and downloading ImmPort shared data using the Aspera Node API.



2.2.6. ImmPort Resources

ImmPort Portal is a static web application that hosts all static web pages required across ImmPort web applications.

2.2.6.1. Feature Summary

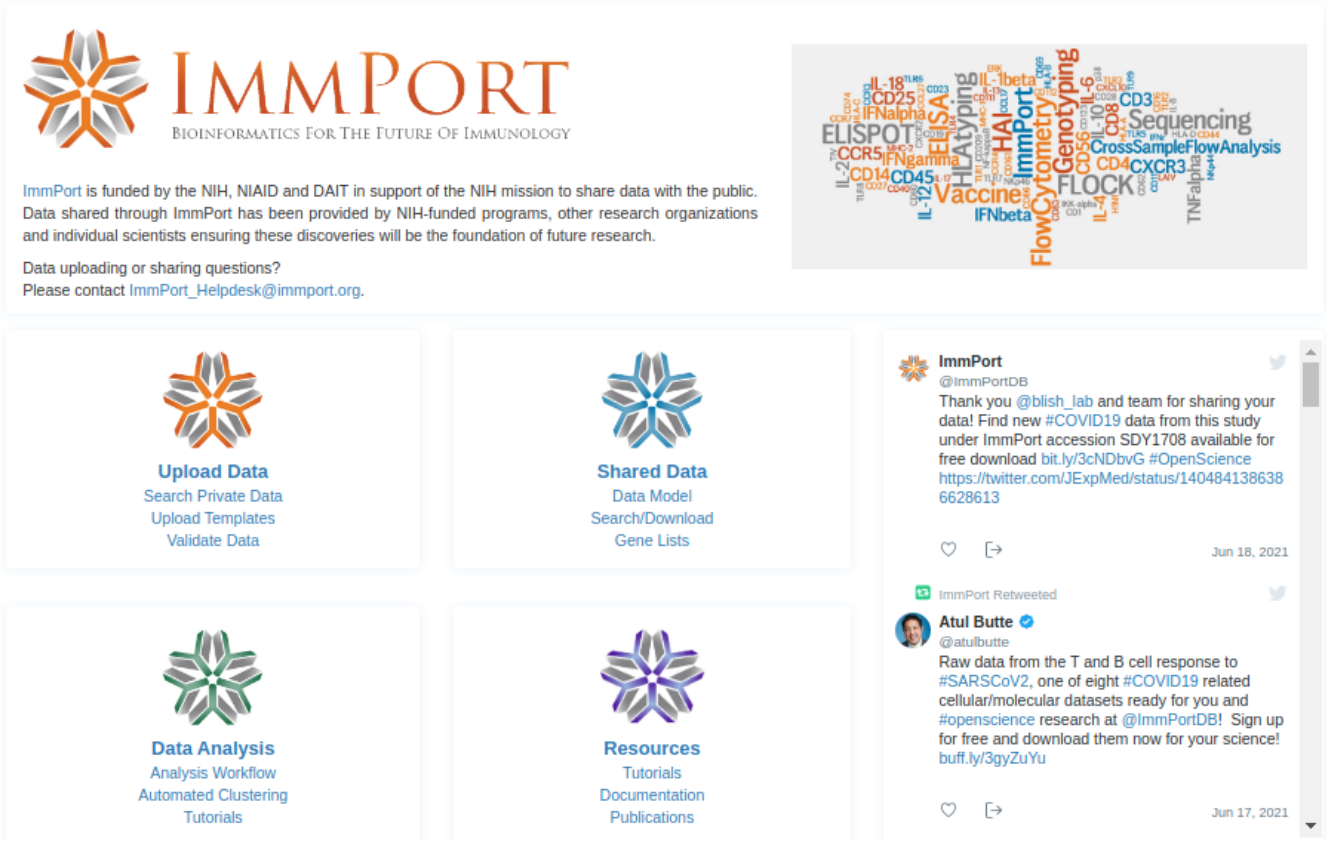
Table 2.4.6.1-1 below summarizes the major functionality of ImmPort Data Browser.

Table 2.4.6.1-1: Summary of ImmPort Capabilities and Features

#	Capabilities/Features	Capability/Feature Description
1	Static page	All static web pages required across all ImmPort applications Ex: Home page, user agreement page, Aspera installation instructions, etc.,
2	ImmPort Tutorials	Hosts several ImmPort tutorials and the instructions to use them
3	ImmPort Documentation	Data Upload and templates documentation

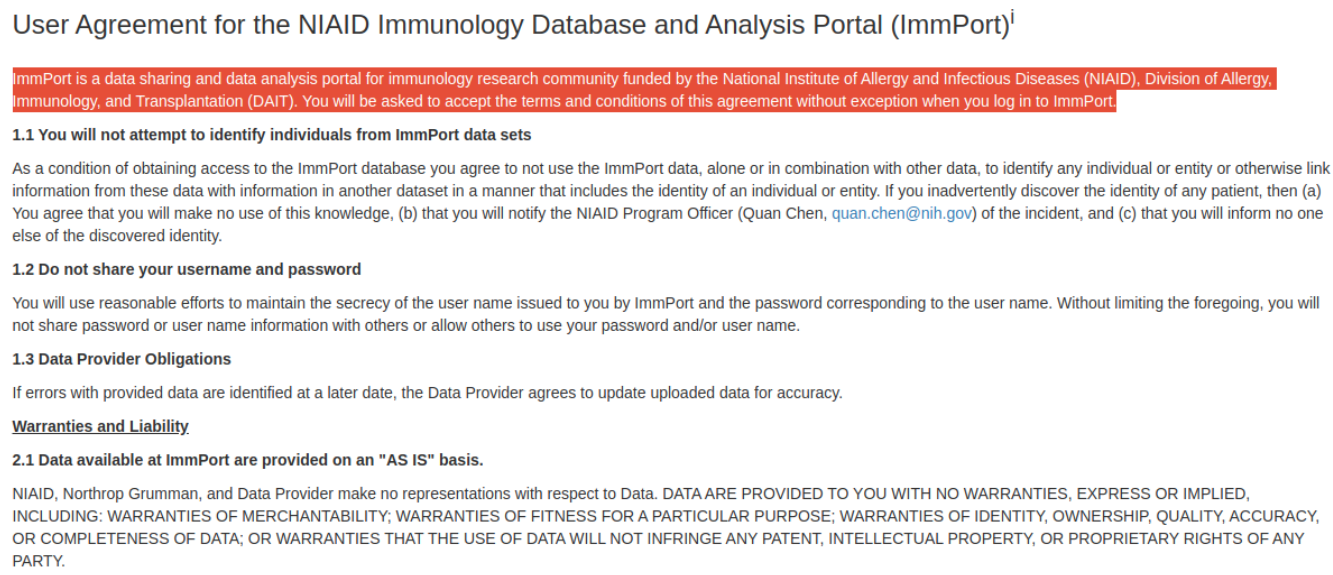
2.2.6.1.1. Feature 1: Home Page

The ImmPort home page is the landing page of the ImmPort Ecosystem and hosts several links to various ImmPort applications



2.2.6.1.2. Feature 2: User agreement

ImmPort is a data sharing and data analysis portal for the immunology research community funded by the National Institute of Allergy and Infectious Diseases (NIAID), Division of Allergy, Immunology, and Transplantation (DAIT). Users will be asked to accept the terms and conditions of this agreement without exception when they log in to ImmPort.



User Agreement for the NIAID Immunology Database and Analysis Portal (ImmPort)[i]

ImmPort is a data sharing and data analysis portal for immunology research community funded by the National Institute of Allergy and Infectious Diseases (NIAID), Division of Allergy, Immunology, and Transplantation (DAIT). You will be asked to accept the terms and conditions of this agreement without exception when you log in to ImmPort.

1.1 You will not attempt to identify individuals from ImmPort data sets

As a condition of obtaining access to the ImmPort database you agree to not use the ImmPort data, alone or in combination with other data, to identify any individual or entity or otherwise link information from these data with information in another dataset in a manner that includes the identity of an individual or entity. If you inadvertently discover the identity of any patient, then (a) You agree that you will make no use of this knowledge, (b) that you will notify the NIAID Program Officer (Quan Chen, quan.chen@nih.gov) of the incident, and (c) that you will inform no one else of the discovered identity.

1.2 Do not share your username and password

You will use reasonable efforts to maintain the secrecy of the user name issued to you by ImmPort and the password corresponding to the user name. Without limiting the foregoing, you will not share password or user name information with others or allow others to use your password and/or user name.

1.3 Data Provider Obligations

If errors with provided data are identified at a later date, the Data Provider agrees to update uploaded data for accuracy.

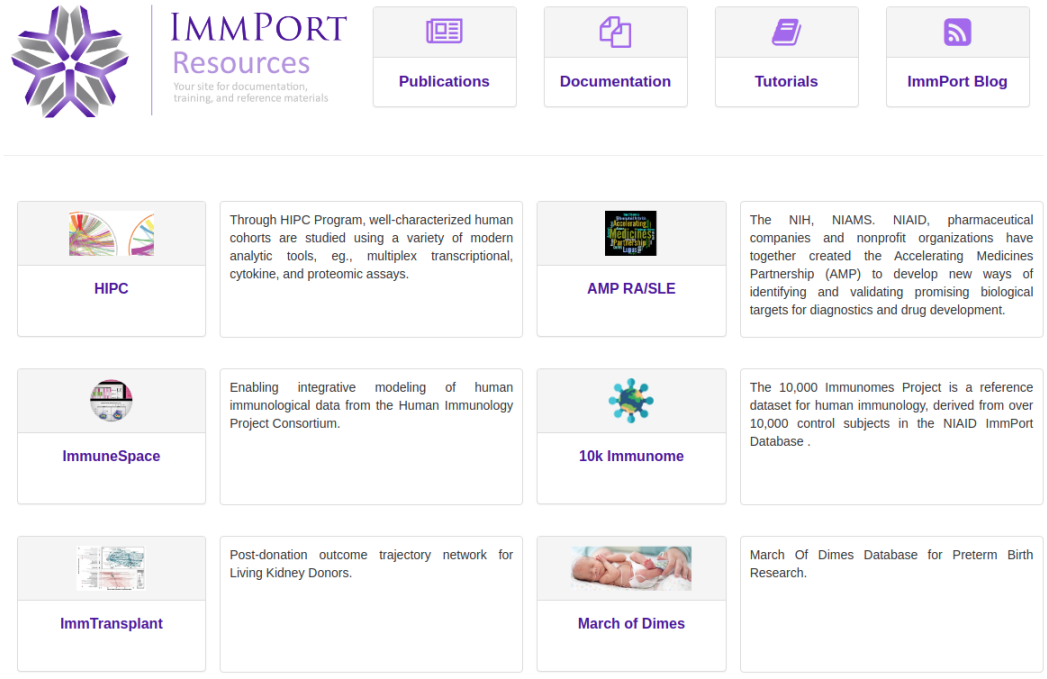
Warranties and Liability

2.1 Data available at ImmPort are provided on an "AS IS" basis.

NIAID, Northrop Grumman, and Data Provider make no representations with respect to Data. DATA ARE PROVIDED TO YOU WITH NO WARRANTIES, EXPRESS OR IMPLIED, INCLUDING: WARRANTIES OF MERCHANTABILITY; WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE; WARRANTIES OF IDENTITY, OWNERSHIP, QUALITY, ACCURACY, OR COMPLETENESS OF DATA; OR WARRANTIES THAT THE USE OF DATA WILL NOT INFRINGE ANY PATENT, INTELLECTUAL PROPERTY, OR PROPRIETARY RIGHTS OF ANY PARTY.

2.2.6.1.3. Feature 2: Resources Page

ImmPort Portal also hosts links to several ImmPort resources like ImmuneXpresso, Cell Ontology, ImmuneSpace, 10K Immunome etc.,



2.2.6.2. SeroNet CDT

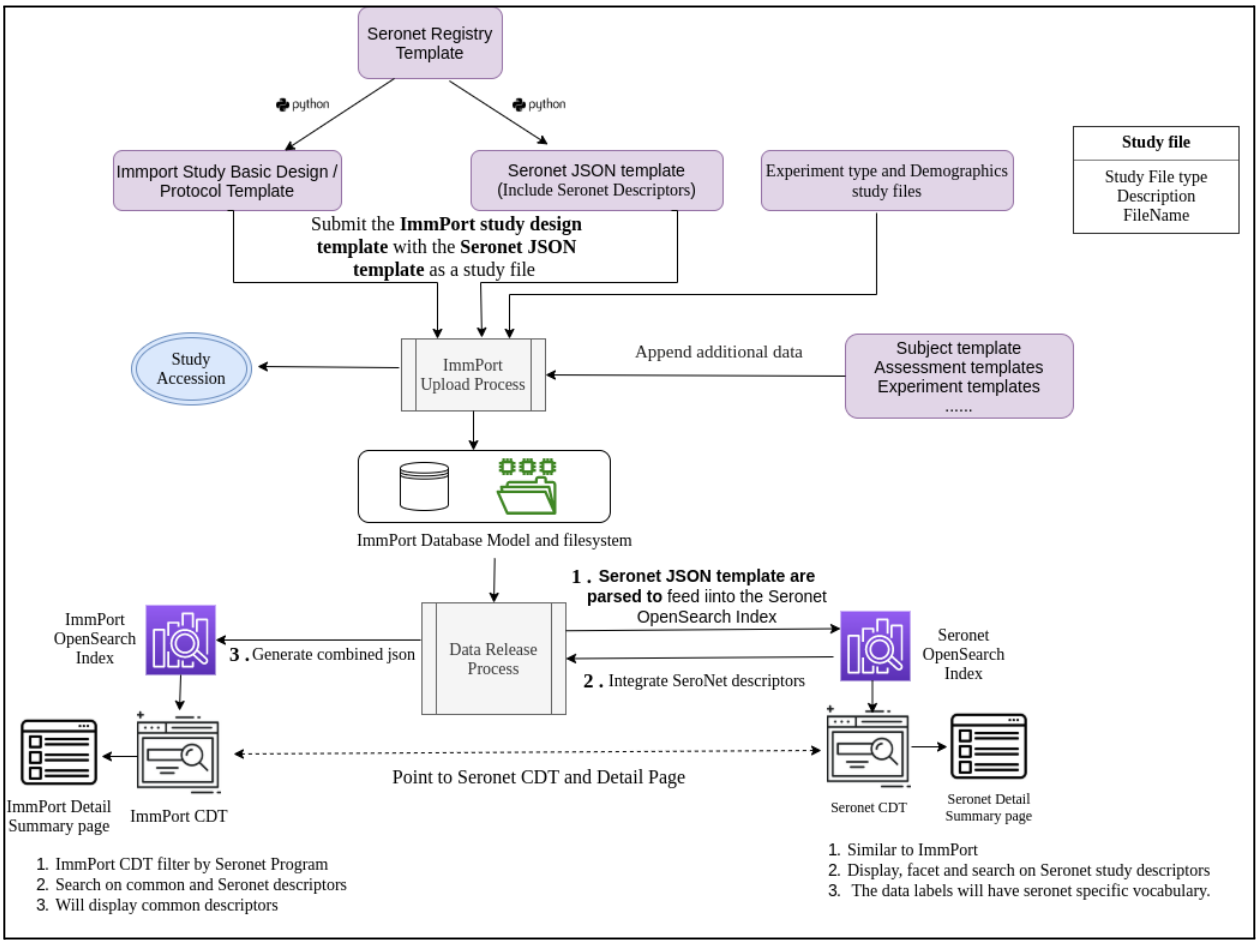
2.2.6.2.1. Introduction

Purpose: A Resource Study catalog of SeroNet studies with faceted search capability

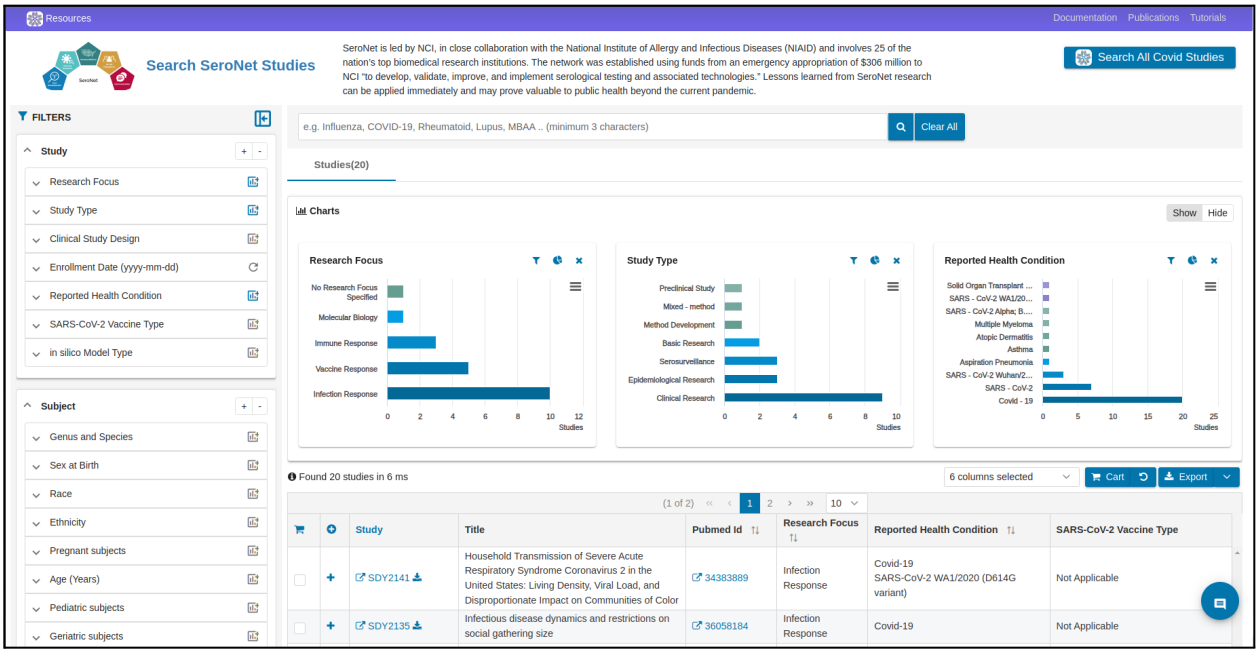
Worked with the SeroNet team on defining the JSON structure needed for building the OpenSearch index

- Developed software program to extract data from the SeroNet registry template into the JSON structure
- Developed SeroNet specific Search Interface that mimics the ImmPort CDT UI and facet on their specific study descriptors
- Developed SeroNet specific Study Detail pages to display their study descriptors and vocabulary
- Extended the ImmPort search to include SeroNet specific descriptors in the free text search
- Extended the data release process to build OpenSearch indexes for SeroNet search
- Added navigation links between ImmPort CDT and SeroNet CDT

2.2.6.2.2. Architecture

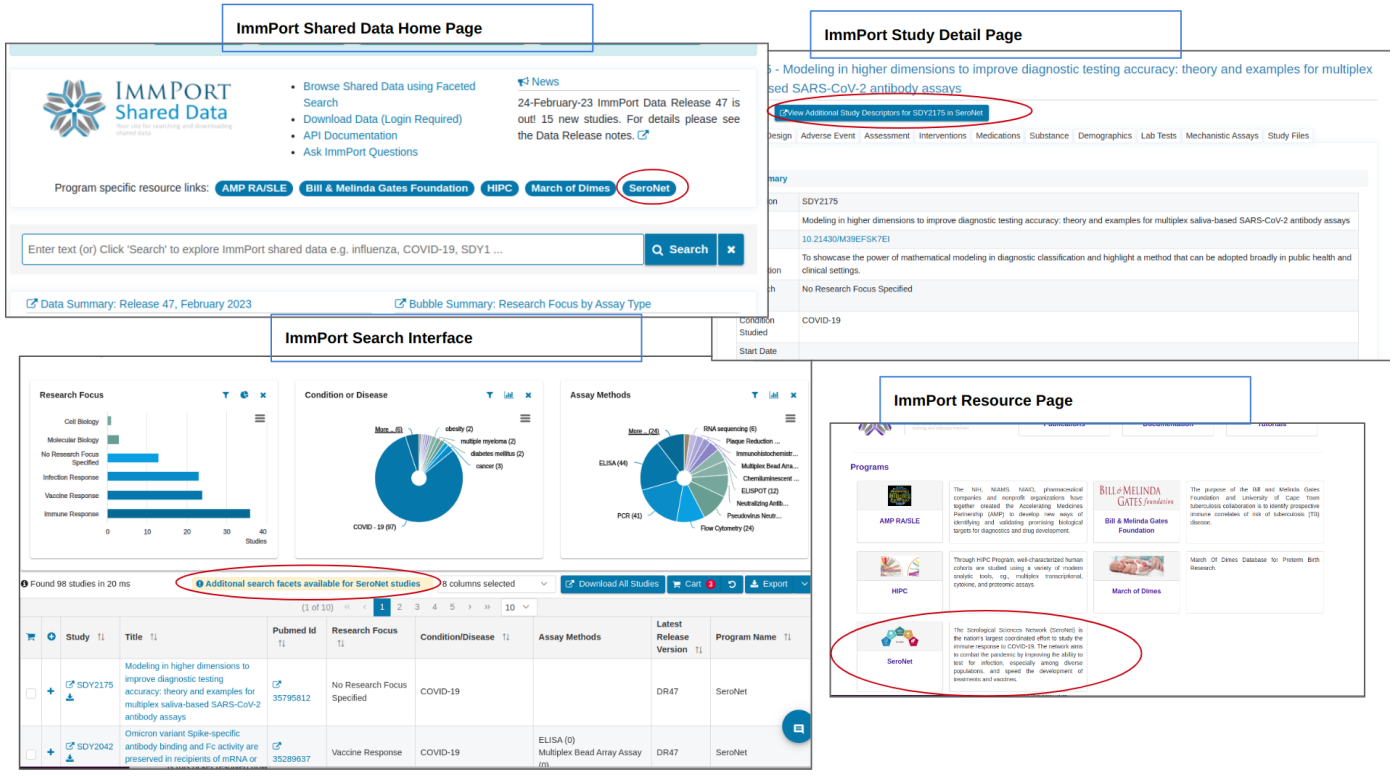


2.2.6.2.3. Feature Summary



Developing the SeroNet Search Interface was a flexible Approach

- New field can be easily added
- SeroNet JSON file - contains all SeroNet Descriptors in a particular format
- Develop SeroNet Search Interface/ SeroNet Detail Page
    - Facet/Search on SeroNet descriptors
    - Fields will have SeroNet vocabulary
- Extend ImmPort Search Interface
    - Augment the ImmPort descriptors with SeroNet descriptors (do not fit in our data model) to better search SeroNet studies (Under Exploration)
    - Point to SeroNet Search Interface when the user is querying for SeroNet studies



2.2.7. ImmPort S3 API

ImmPort S3 API is an Amazon API Gateway fully managed by AWS. It is a RESTful API created to read and write to the JSON files stored on the AWS S3 buckets. The purpose of this API is to push updates to ImmPort web pages outside their build cycles. For example, ImmPort announcements on the Shared Data home page are stored in a JSON file on the S3 bucket, a new announcement will require only a JSON file update on the S3 bucket instead of a complete build cycle of the software. AWS Lambda is employed for some of the endpoints to do any data processing that is needed prior to returning the data to the user.

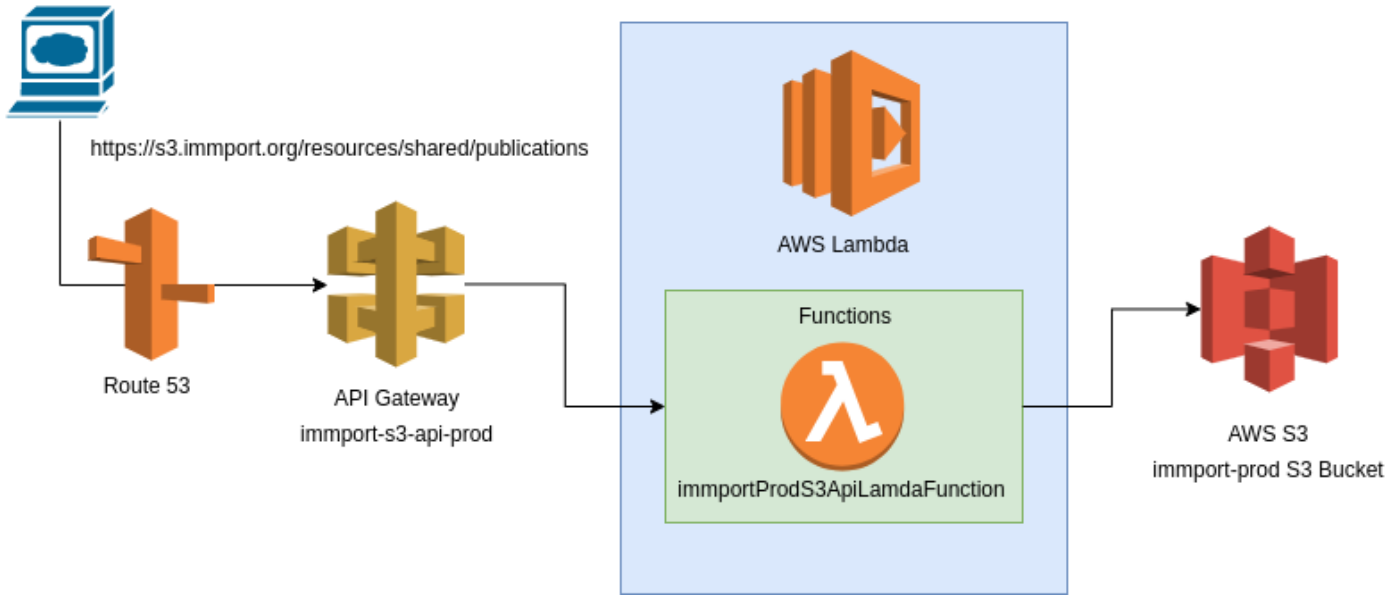
2.2.7.1. Feature Summary

Following are the endpoints available in ImmPort S3 API accessible at the URL https://s3.immport.org/

- prod
  - /
    - /banner
      - /banner/{applicationName}
        - GET
        - OPTIONS
    - /release
      - /release/data
        - /release/data/{fileName}
          - GET
          - OPTIONS
      - /release/genelists
        - /release/genelists/{fileName}
          - GET
          - OPTIONS
      - /release/graphics
        - /release/graphics/{visualizationType}
          - /release/graphics/{visualizationType}/{dataType}
            - GET
            - OPTIONS
      - /release/metadata
        - /release/metadata/{fileName}
          - GET
          - OPTIONS
      - /release/notes
        - /release/notes/{releaseVersion}
          - GET
          - OPTIONS
      - /release/tableDefinition
        - /release/tableDefinition/{fileName}
          - GET
          - OPTIONS
      - /release/templateDefinition
        - /release/templateDefinition/{fileName}
          - GET
          - OPTIONS
    - /resources
      - /resources/{resourceType}
        - /resources/{resourceType}/{fileName}
          - GET
          - OPTIONS

2.2.7.2. ImmPort S3 API Server Architecture

The ImmPort Data API is hosted on an EC2 instance in the public subnet on AWS. The Aspera server is hosted on the NIAID On-premises infrastructure.



2.2.8. ImmPort CloudFront Distributions

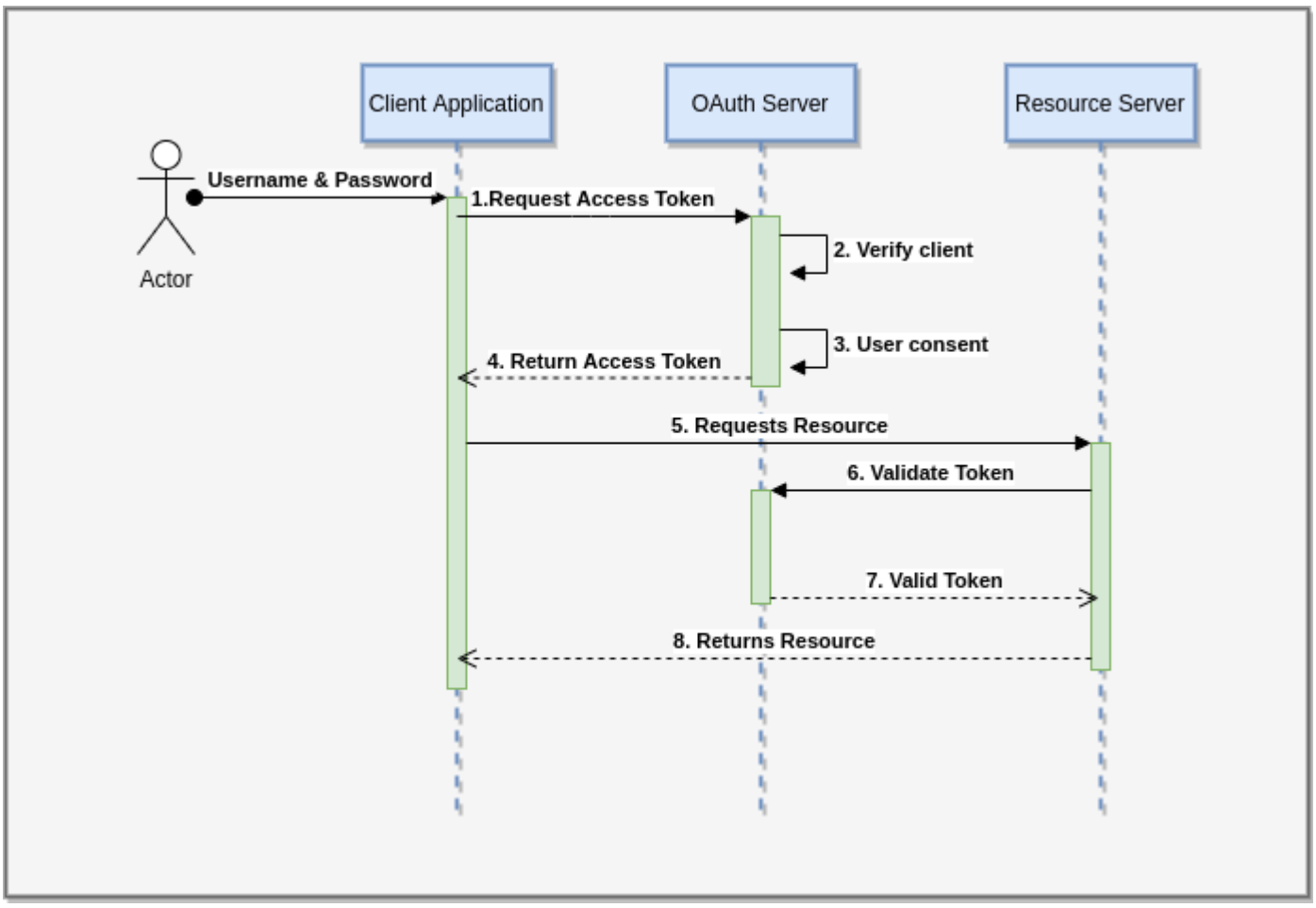
ImmPort configures Amazon CloudFront distributions to server HTTPS requests for the following two ImmPort S3 buckets.

1. downloads.dev.immport.org: Used for allowing users to download data upload templates and example packages.
2. docs.immport.org: Used for Static website hosting of API documentation website.

Content is for public use on these S3 buckets.

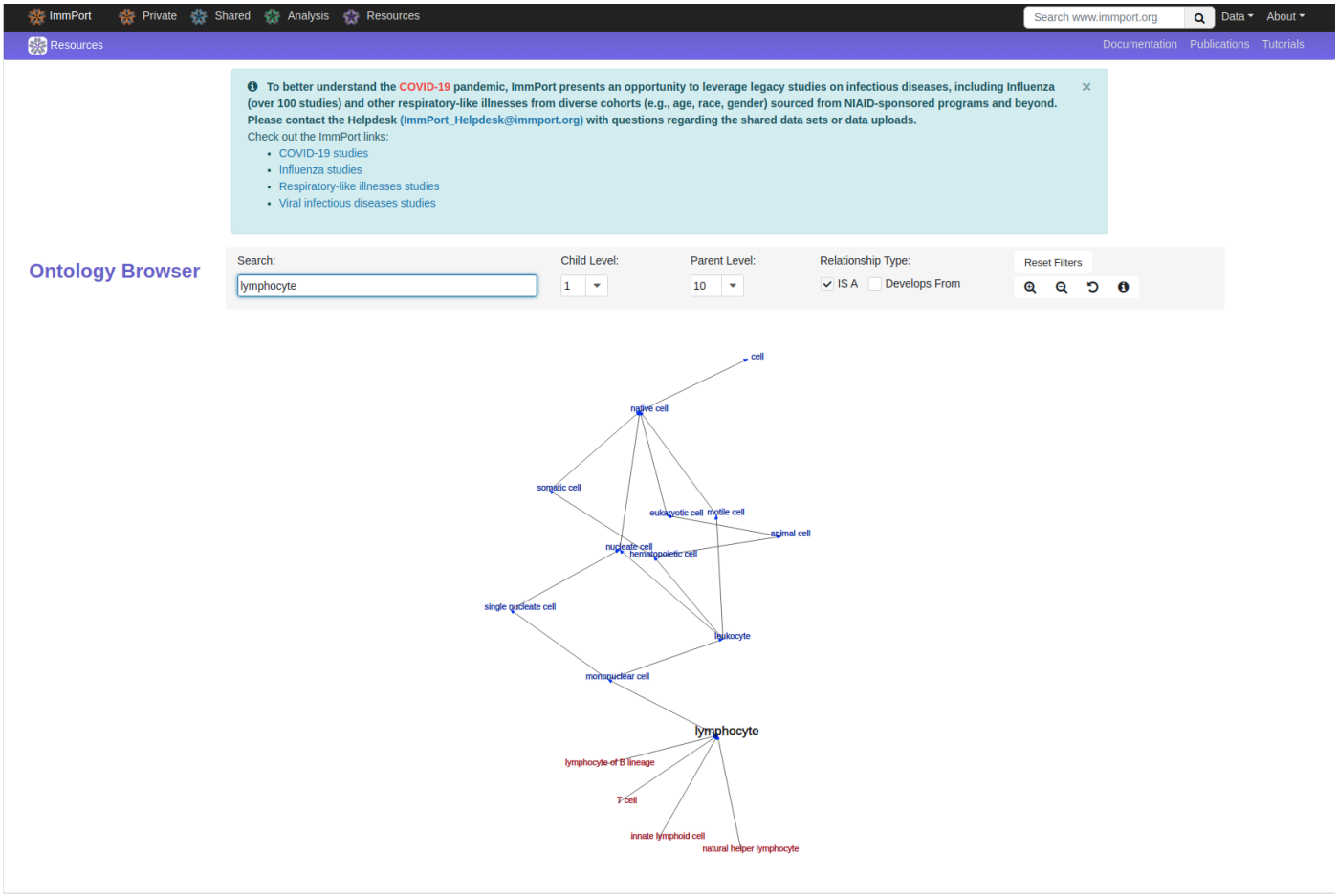
2.2.9. Authentication of the Rest API - Resource Servers

The Rest APIs are OAuth 2.0 Resource Servers built using Spring Security 5. In the context of OAuth 2.0, a resource server is an application that protects resources via OAuth tokens. These tokens are issued by an authorization server, typically to a client application. The job of the resource server is to validate the token before serving a resource to the client. JWT, or JSON Web Token is a way to transfer sensitive information securely in the widely-accepted JSON format. The contained information could be about the user, or about the token itself, such as its expiry and issuer. To visualize, let's look at a sequence diagram for the authorization code flow and see all the actors in action:

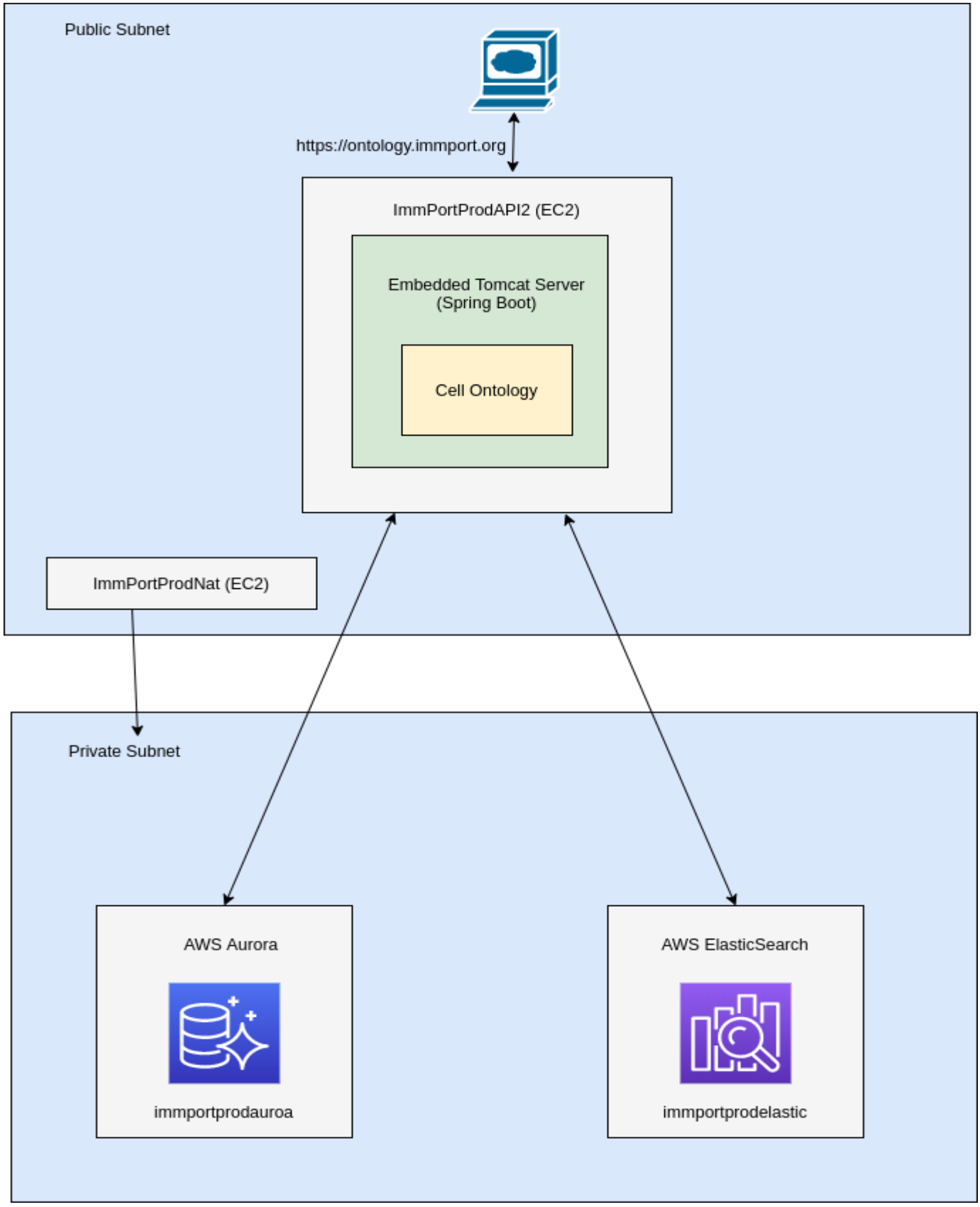

Client Application
OAuth Server
Resource Server
Actor
Username & Password
1.Request Access Token
2. Verify client
3. User consent
4. Return Access Token
5. Requests Resource
6. Validate Token
7. Valid Token
8. Returns Resource

2.2.10. Cell Ontology Browser

The Cell ontology browser was developed to support the visualization of the Cell Ontology (http://obofoundry.org/ontology/cl.html), which is of great utility in the curation of ImmPort data for the standardization of cell populations. The browser provides a force-directed graph visualization of the ontology, and utilizes the same software stack (Angular, ElasticSearch, MySQL) as other ImmPort applications, to allow for searching for Cell Ontology terms with the addition of D3 for the visualization component. The content is updated as part of the ImmPort data release process.

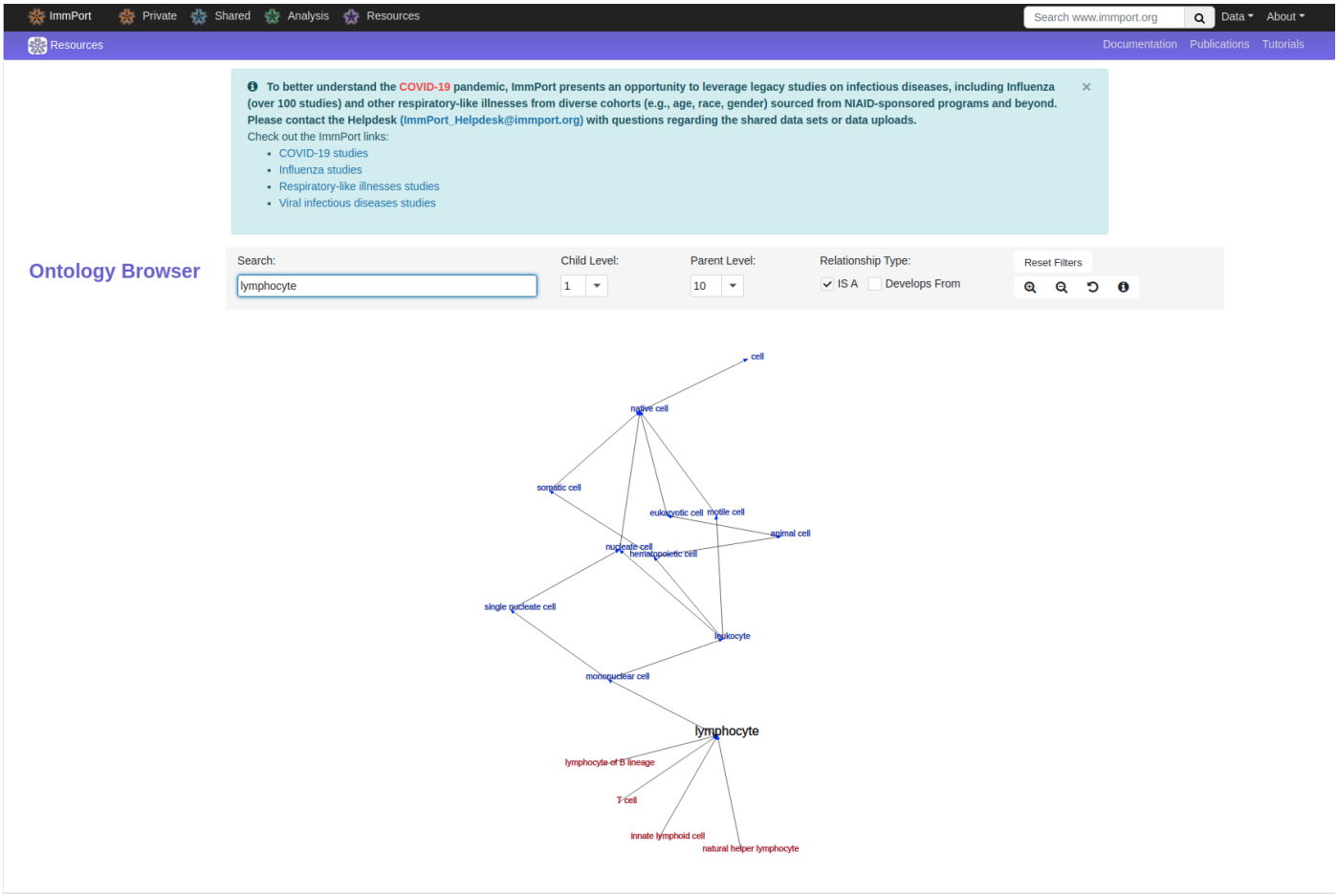


2.2.10.1. Cell Ontology Browser Architecture

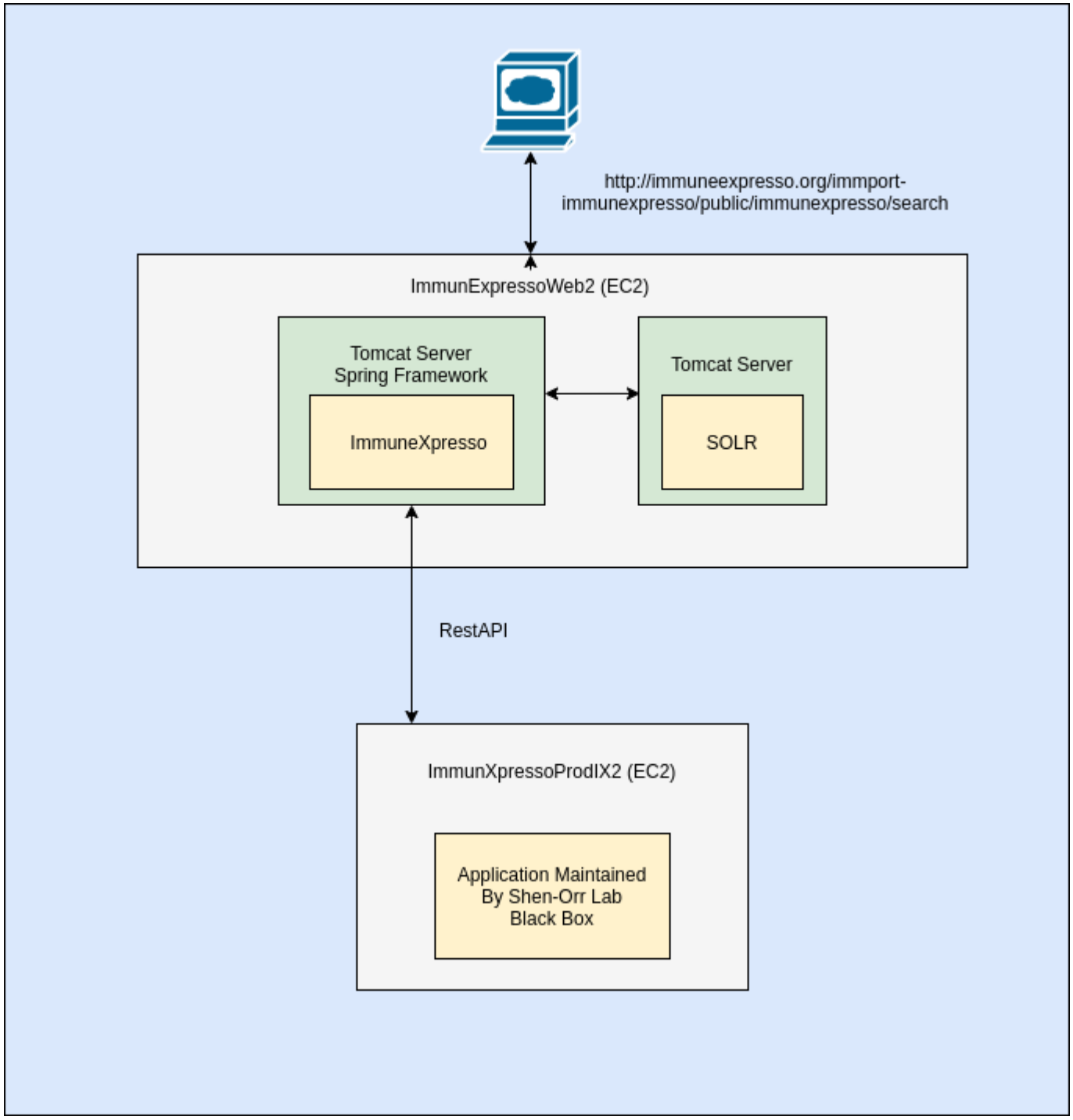


2.2.11. ImmuneXpresso

The ImmuneXpresso application was built under the BISC Contract Option 7 in years 2014-2015 and developed in collaboration with the team at the Shen-Orr lab at Technion (http://shenorrlab.technion.ac.il/). ImmuneXpresso continues the work of the Shen-Orr lab in mining PubMed abstracts to determine relationships between cells and cytokines. The primary content is stored on a standalone EC2 hosting a MySQL database and accessible via a RESTFul API. This EC2 instance is maintained by the Shen-Orr lab, and a black box to the ImmPort team. The front-end technology follows the same design pattern as most single-page applications but was written before frameworks like React and Angular were available. The code base is 5 years old and not updated to use the Angular framework like other ImmPort applications. ImmuneXpresso term queries are supported by SOLR indexing of cell and cytokine terms.



2.2.11.1. ImmuneXpresso Architecture



2.2.12. Galaxy

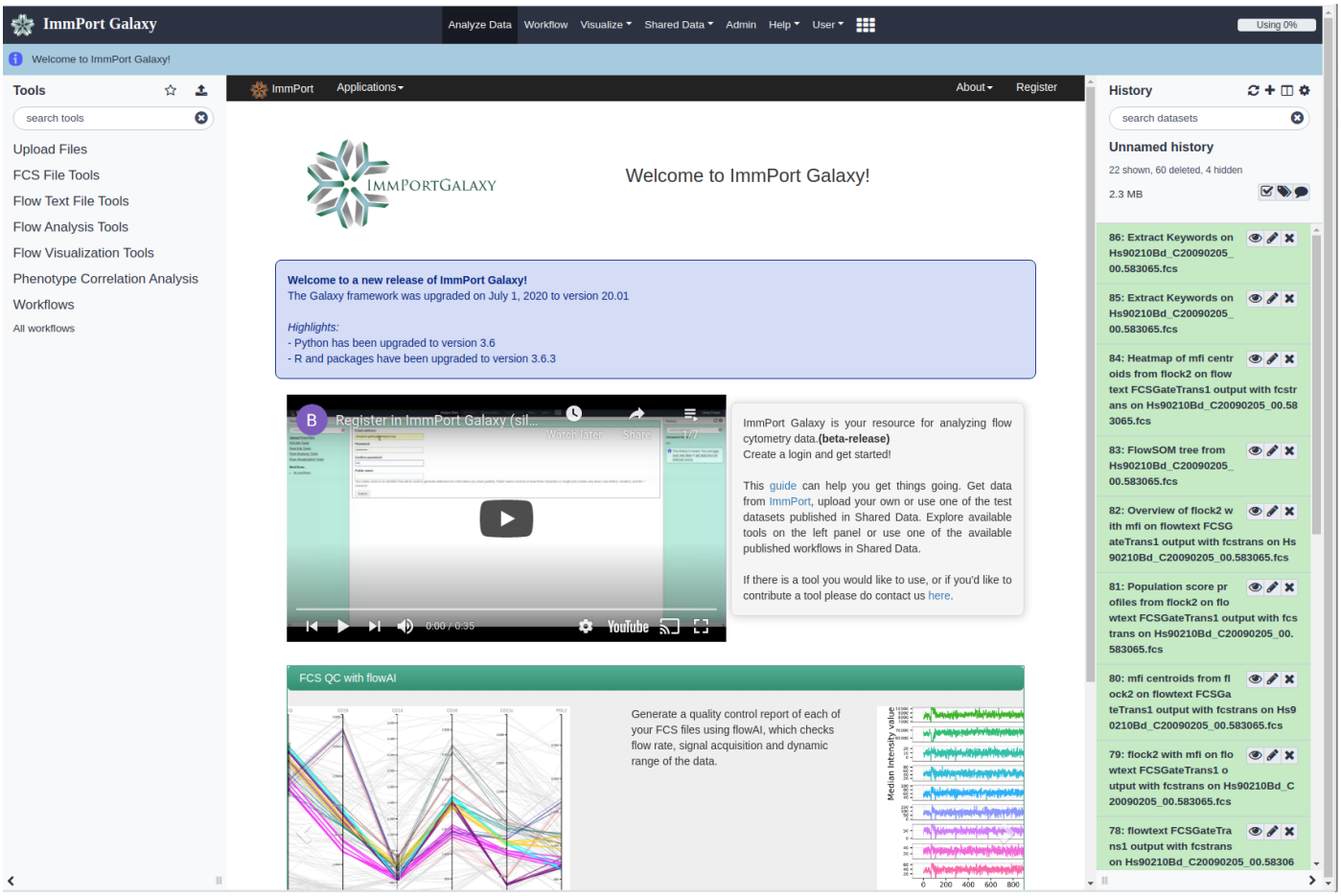
The original ImmPort Open application provided support for flow cytometry analysis primarily using the FLOCK (FLOw Clustering without K) algorithm. During the current contract, we have chosen to utilize the popular analysis workflow engine Galaxy (https://galaxyproject.org/) to optimize the modular method and component development and eventual sharing of data and workflows. The use of a publicly available web analysis framework was chosen over

the direct replacement of existing code in ImmPort because of the existence of open-source tools that largely perform the workflow capabilities of queuing, bursting, and chaining methods in a generic way and to reduce the cost to the ImmPort development team when developing and maintaining code to perform those workflow capabilities.

Galaxy is an open, web-based platform for accessible, reproducible, and transparent computational research. The ImmPort Galaxy platform is focused on providing tools for flow cytometry analysis. In addition to implementing many R/BioConductor packages for flow cytometry analysis, the ImmPort team has written several modules to aid in the visualization of the results. Below is a list of some of the tools available in the ImmPort Galaxy application:

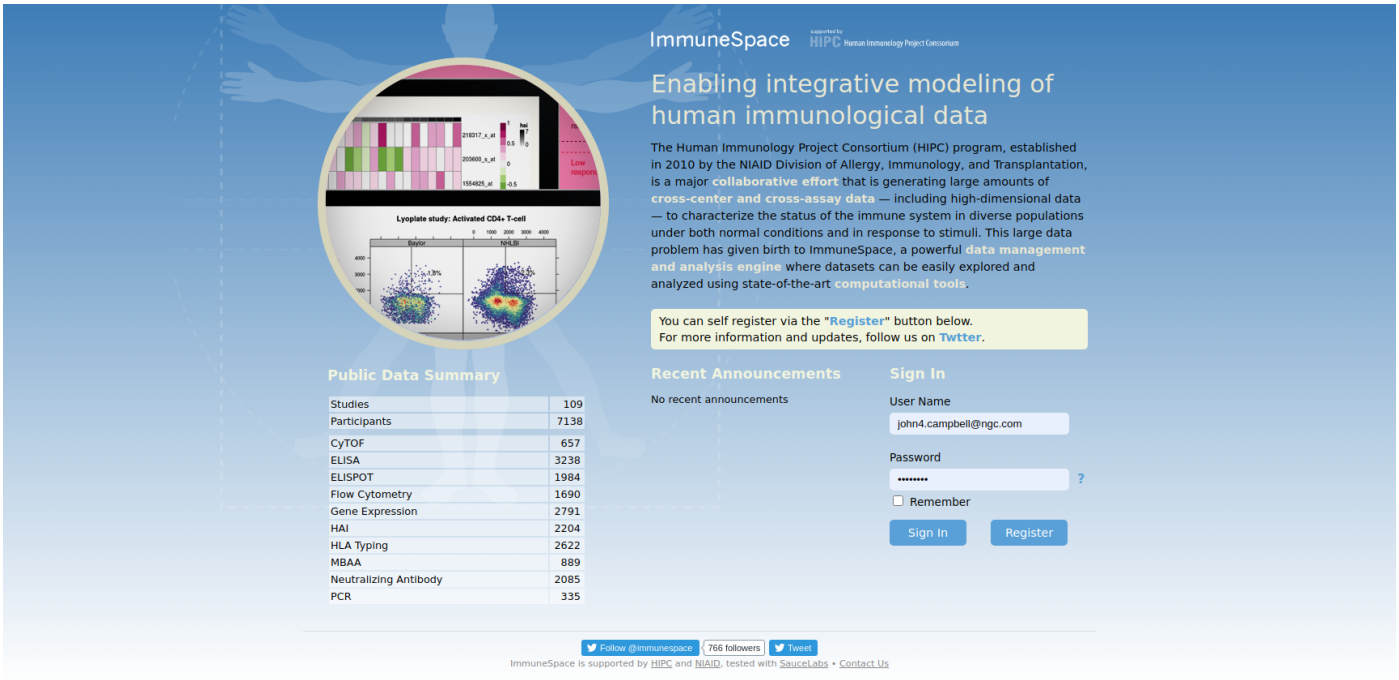
- Clustergrammer
- Flock version 2 and 3
- flowAI
- flowCL
- flowDensity
- flowStats
- flowViz
- FlowSOM
- MetaCyto

The ImmPort Galaxy instance is hosted on an AWS EC instance and uses additional volumes to host the Galaxy file system and an Aurora PostgreSQL instance to support the Galaxy database.



2.2.13. ImmuneSpace

ImmuneSpace, available at www.immunespace.org, was developed by the team at the Gottardo lab (http://www.rglab.org) at Fred Hutchinson Cancer Center with the team at Labkey Software (www.labkey.com) underfunding of the Human Immunology Project Consortium (www.immuneprofiling.org). Details about the project are available at the ImmuneSpace site and the architecture in the Labkey product pages. The ImmPort team provides hosting and basic IT services on AWS for ImmuneSpace in the AWS instances funded by NIAID/DAIT.



2.2.14. Metrics

Elasticsearch and Kibana (EK) Stack on AWS is used for storing, searching, and visualizing log and metric data. This allows for better searches and creates more analytical graphs for usage metrics. An Elasticsearch, Fluentd, and Kibana (EFK) Stack was initially considered but we decided on the Elasticsearch, Metric Rest API, Kafka, Aurora Mysql stack.

2.2.14.1. A common data model for Metrics

- A common JSON format was devised after looking at the various data elements of each of the different applications so that it can be easily searched and visualized. The common format that was formulated was as follows:

```
{
  // common parameters
   "metricId": “”,
```

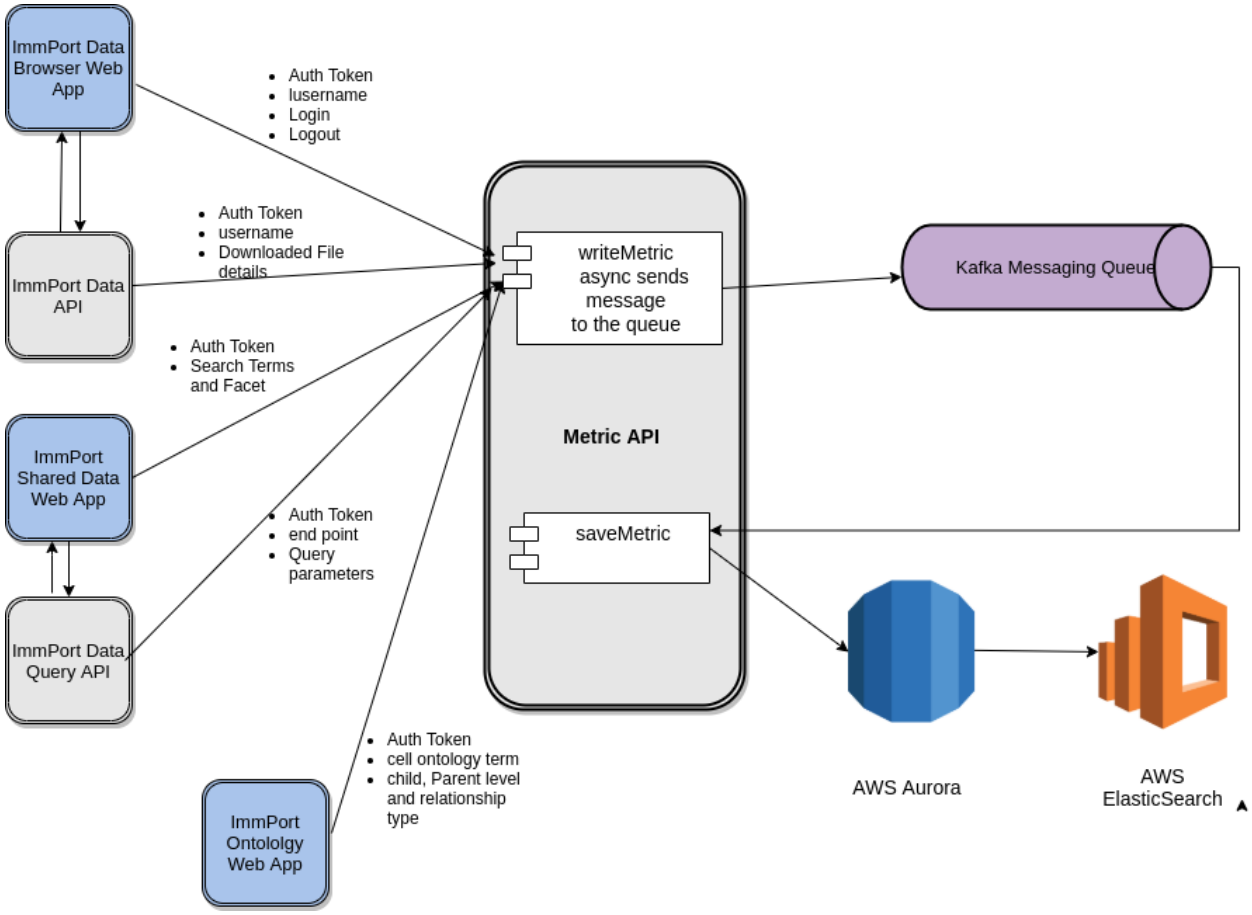
```
  "metricType": "",
  "username": "",
  "remoteIpAddress": "",
  "organization": "",
  "applicationName": "",
  "endPoint": "",
  "startTime": "",
  "ngUser": "",
  "dateCreated": "",
  "createdBy": "",
```

###The parameters field is an object of data elements for the different applications. The different applications will fill in the appropriate fields.

```
  "parameters": {
    "searchTerm": "",  parameter for ImmPort shared data
    "clinical": "Y",
    …..
    “fileName”: “” parameters for ImmPort Data Browser
    “fileSize”: “”,
    …..
   “parentLevel” :””, parameters for ImmPort Ontology
   “relationshipType” :””,
    ……
  “ageEvent”:””, parameters for ImmPort Data Query API
 “expsampleAccession”:””,
….


  }
 }
```

- Historic metric data currently stored in CORE_IMMPORT was copied to the Metric Database on AWS Aurora
- The current and historic data in the Metric Database on AWS Aurora was cleaned, validated and transformed to the new format and stored in a new table in AWS Aurora MySQL database called metric_log which conforms to the new format.
- Data from this table was extracted as JSON files and then sent to ElasticSearch.

2.2.14.2. Metrics Software Architecture



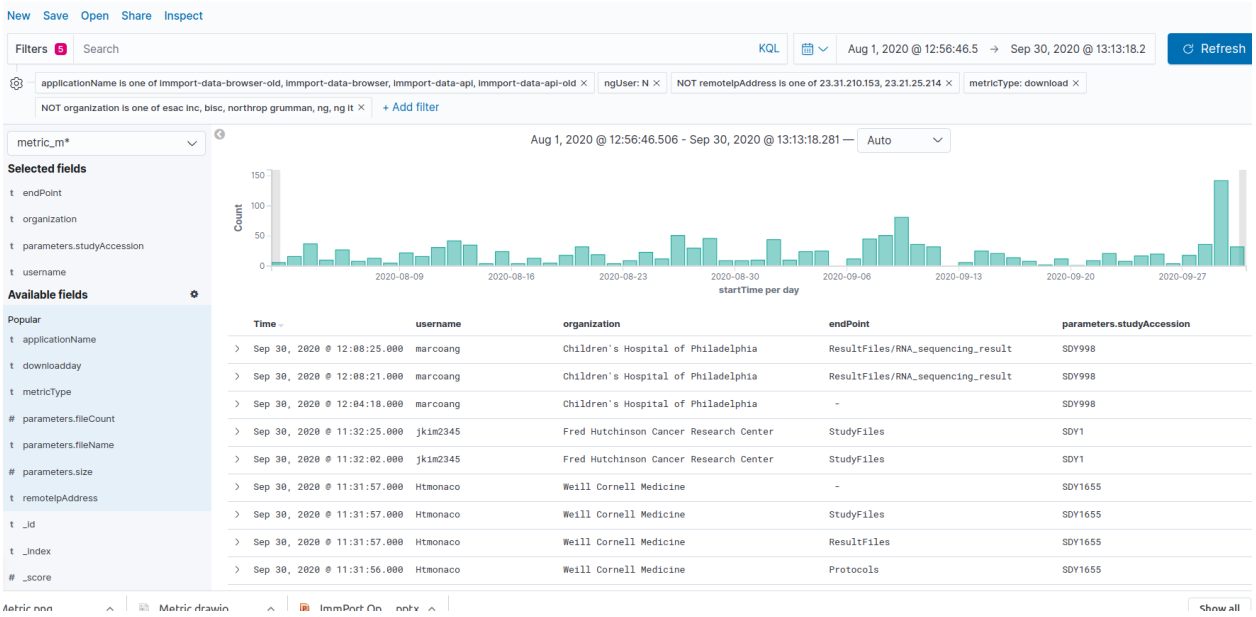
- REST has quickly become the de-facto standard for building web services because they're easy to build and consume. We built a Rest API to collect metrics from all ImmPort Applications so we have a central application to collect metrics.
- The Metric Rest API endpoints require an authentication token for access. ImmPort Applications call the writeMetric endpoint on the Metric Rest API which takes a metric object.
- The Metric API asynchronously sends the metric object to the Kafka messaging queue. This enables ImmPort Applications to log the metric and immediately return. The method in the ImmPort Applications to call the metric endpoint is also asynchronous. At no point will the ImmPort Applications be blocked due to logging.
- The Metric API is also a listener to the Kafka messaging queue. As soon as the queue receives a metric object the listener calls the saveMetric endpoint to save the metric to the database.
- A cron job runs every 1 minute to check whether a new metric is saved to the database. If there are metrics these rows are converted to a json object and sent to Elastic search

2.2.14.3. Visualizations in Kibana

Searching the metric logs and creating usage visualizations becomes easy since the data is stored in Elasticsearch

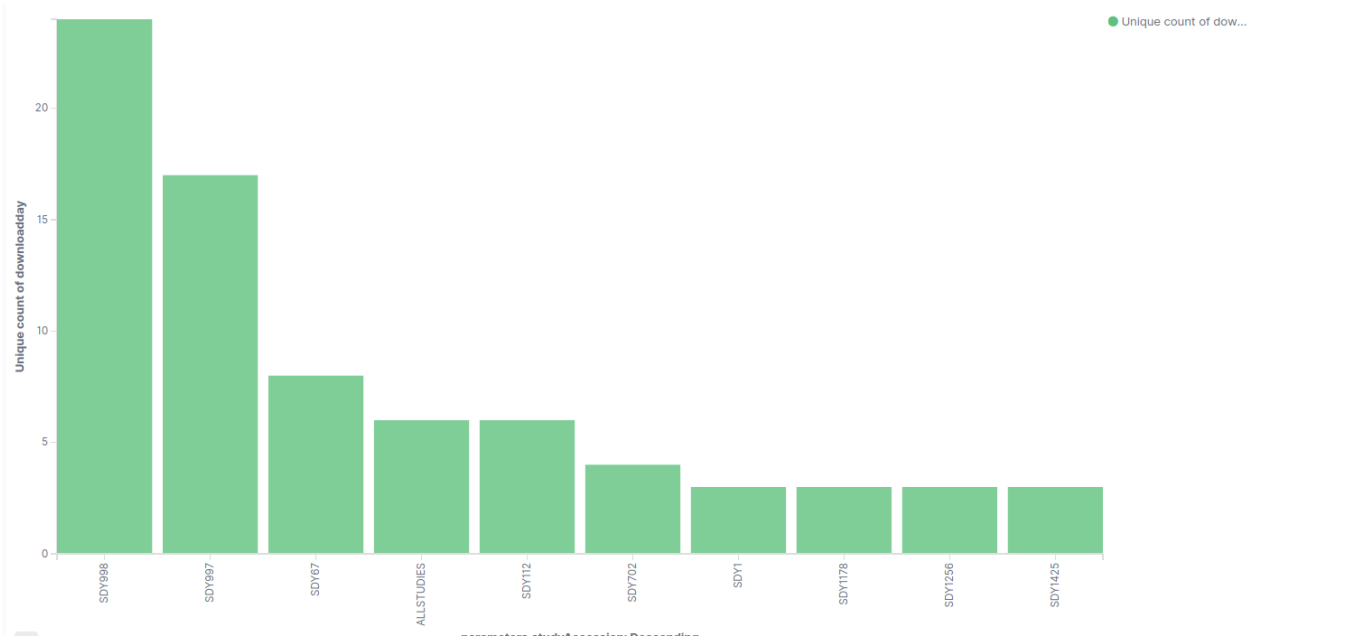
2.2.14.3.1. Example of the Discovery tab

ImmPort Data Browser and ImmPort Data Api usage



The figure above shows some of the fields of metrics collected for the data browser and data api, e.g., username, organization,endPoint, study accession. Other fields that can be shown are filename,file count, file size, application name, remote ip address of the user etc.

2.2.14.3.2. Example of the Visualization tab :



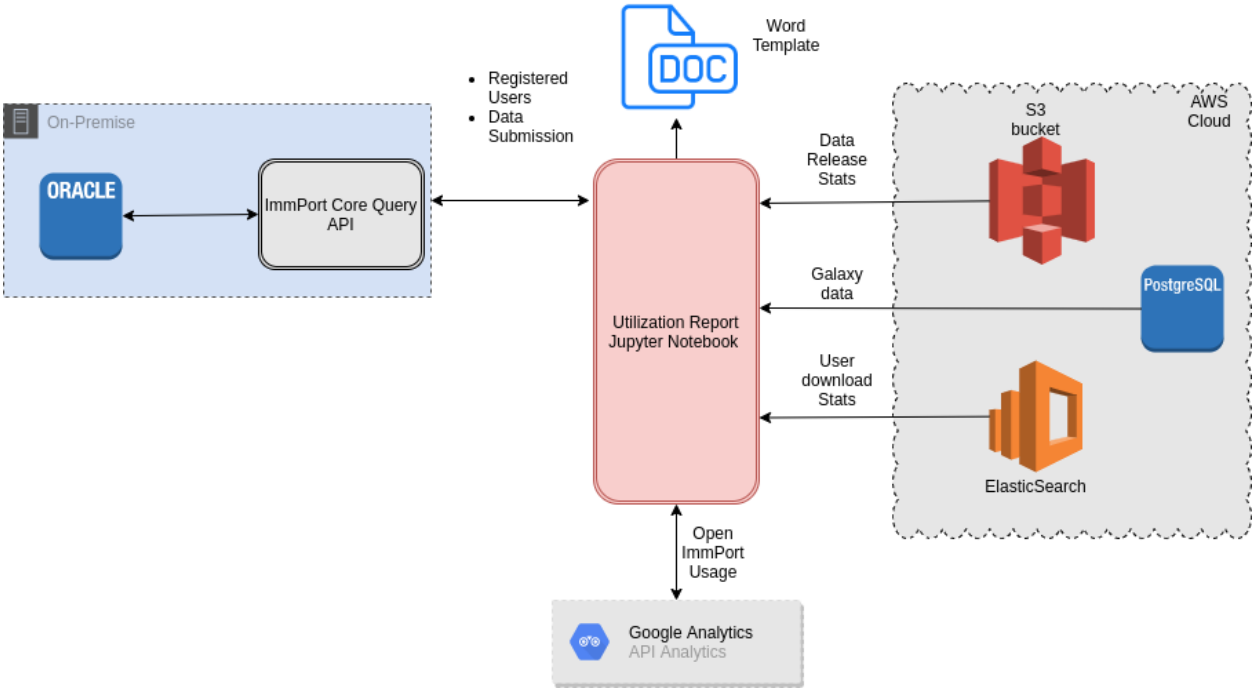
The above figure shows the top ten studies downloaded for the month of September.
Various types of visualizations and data can be aggregated in the ElasticSearch and Kibana stack.

2.2.15. Utilization Report

We have automated the Monthly ImmPort Utilization Report using Jupyter Notebook. A predefined Jinga template has been created and the notebook populates the template with the specifics for that report.

The monthly information is currently pulled from either:

- The ImmPort Core Query API
- AWS Elasticsearch
- S3 bucket
- ImmPort Galaxy Postgres
- Google Analytics API



2.2.16. ImmPort HAPI FHIR Server

The FHIR project began with a mapping exercise between the fields of the ImmPort basic study design worksheet, used to load new studies into ImmPort, to numerous FHIR resources. We started with FHIR 4.0.1 but found many mappings missing or difficult to map to ResearchStudy. A subsequent review of FHIR 5.0.0 revealed the updates to the ResearchStudy resource made mapping more straightforward. We followed with a comparison of FHIR mappings from other systems (listed below) in an effort to identify variations and preferred approaches.

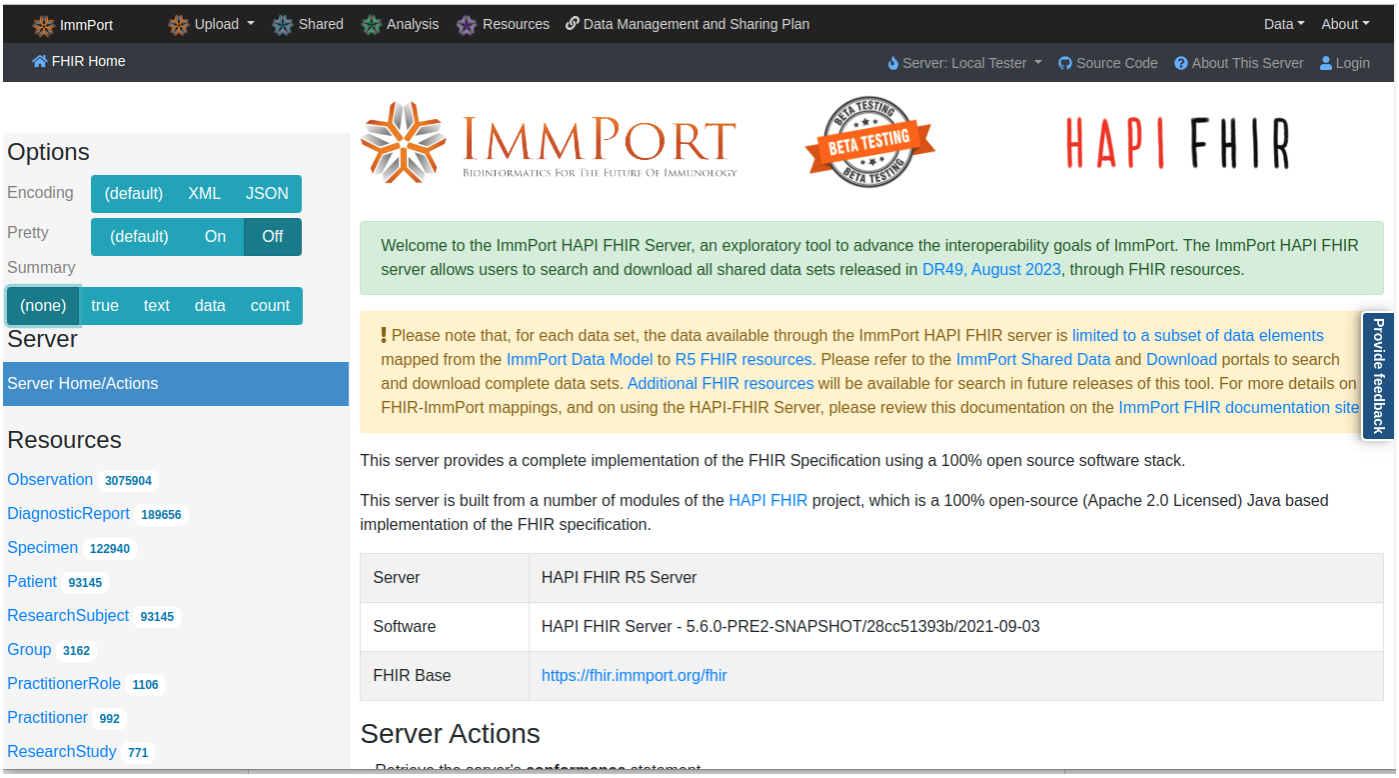
- ClinicalTrials.gov
- FHIR4FAIR
- KidsFirst (which was mapped to FHIR 4.0.1)
- dbGap

The comparison resulted in the identification of key FHIR resources for which we mapped each with as much detail as possible. Please see the attached spreadsheet as well the worksheets named for each FHIR Resource. With this

detailed mapping in hand we developed Python code and Jinja2 templates for generating FHIR JSON resources from the ImmPort Database. The Python code queries the appropriate tables and columns and adds them to a model object used by the Jinja2 templates to populate a skeleton JSON file to generate FHIR resources. A sample of how we mapped a resource is shown below.

Mapping for ResearchStudy Resource				
Using FHIR R5				
Resource Attribute	**Complex attribute elements**	**ImmPort Table**	**ImmPort Column**	**Hardcoded Value**
identifier	use			official
	type.coding.system			http://terminology.hl7.org/CodeSystem/v2-0203
	type.coding.code			ACSN
	type.coding.display			Accession ID
	value	**study**	**study_accession**	
version		**study**	**latest_data_release_version**	
title		**study**	**official_title**	
label	type.coding.code			short
	value	**study**	**brief_title**	
relatedArtifact	type			citation
	label	**study_link**	**name**	
	url	**study_link**	**value**	
status				active
category	text			Clinical Trial
focus	coding.system	**lk_research_focus**	**link**	
	coding.code	**lk_research_focus**	**link**	
	coding.display	**lk_research_focus**	**name**	
	text	**lk_research_focus**	**name**	
condition	coding.system	**lk_disease**	**link**	
	coding.code	**lk_disease**	**link**	
	coding.display	**study_2_condition_or_disease**	**condition_preferred**	
	text	**study_2_condition_or_disease**	**condition_preferred**	
descriptionSummary		**study**	**brief_description**	
description		**study**	**description**	
objective	name	**study**	**objectives**	
sponsor	reference	**study**	**sponsoring_organization**	
outcomeMeasure	description	**study**	**endpoints**	

A FHIR server was set up, using the open source HAPI FHIR reference implementation, to validate the FHIR resources generated and to make the mapped data easily available for review by others. The Python code was modified to PUT the resources to the FHIR server using ImmPort accession IDs as the ID.

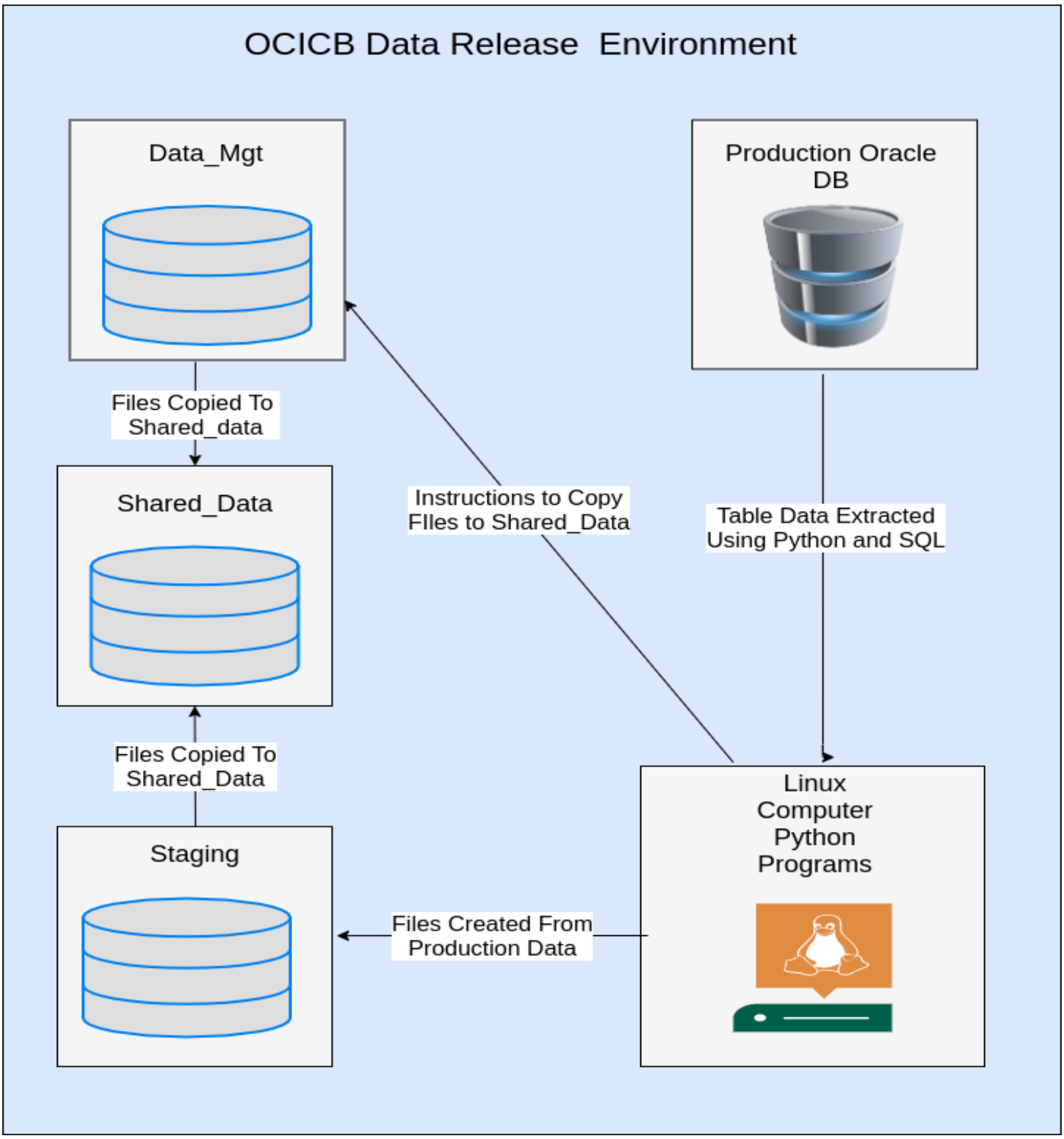


This approach allows users to query the FHIR versions of the ImmPort shared studies and display the completed mappings. Accession IDs allow users to query any given study using the FHIR API and the study accession ID like https://fhir.immport.org/fhir/ResearchStudy/SDY1

2.3. Data Release

The ImmPort team currently produces 4 to 6 data releases per year. With each release additional studies are made publicly available as well as updates made to previously existing shared studies. The process involves using computers, data in file systems and an Oracle database in the NIAID OCICB environment, as well as an AWS Aurora MySQL instance, S3 buckets, and software in the ImmPort AWS environment. The current contract mandates the production database and data uploaded to ImmPort be hosted in the NIAID OCICB facility.

2.3.1. Overview of OCICB Components and Process

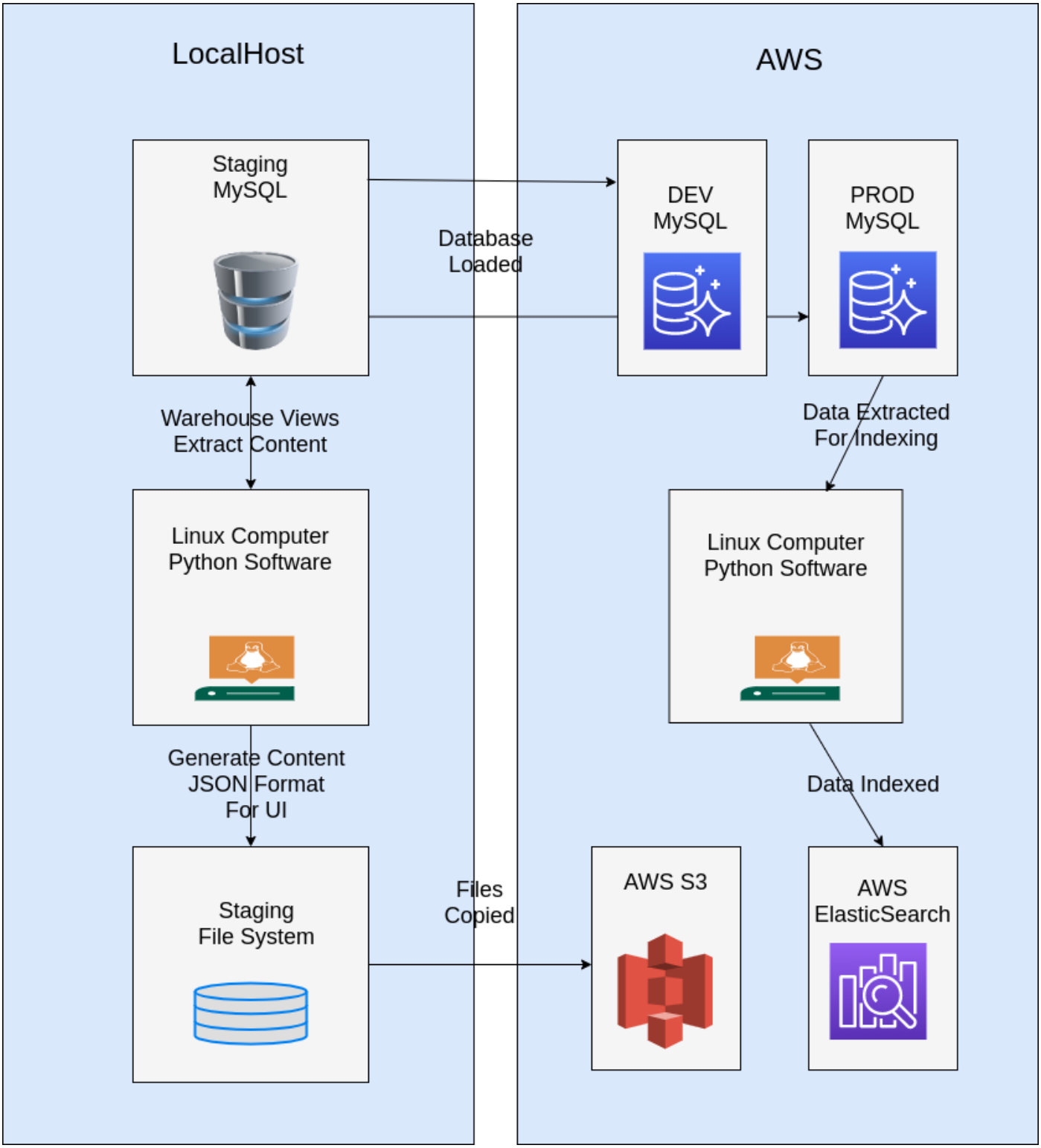


Steps to perform a data release are depicted above and outlined below. In this section we focus on how Study packages, including the ALLSTUDIES package, are created as part of the data release process. We will not discuss the initial step, executed by the Data Curation team, that identifies which studies are ready for public data sharing.

1. Create a DOI using the DataCite platform for new shared studies. New DOIs are uploaded to the production database.

2. A Python process is run to construct the ALLSTUDIES package that includes all shared studies bundled into a package suitable for loading into a MySQL database. The process runs on the Linux server and executes code to extract information from the Oracle database which is used to create the ALLSTUDIES package. The files created by this process are placed in the Staging file system.
3. The ALLSTUDIES package is used to populate a local MySQL database. This process ensures data extracted matches the data we expect to be extracted for the data release. QC steps run scripts to measure whether table row counts have increased from the previous release to the current release. Another script checks whether the table structure of the previous data release matches the table structure of the current release, etc. The primary QC check occurs when foreign keys are applied for each table as the final step in building the MySQL database. If data has not been extracted properly foreign key violations arise requiring review before the data release process can continue. The ALLSTUDIES package is placed in the Staging file system.
4. Once ALLSTUDIES data have been extracted properly another Python process is executed to extract information from the Oracle database and construct an individual package for each study. During this step result files for the new studies are copied from the Data_Mgt file system to the Shared_Data file system. The study packages are placed in the Staging file system.
5. On the day of the release the current contents for the previous release are moved into their respective **archive directory** on the Shared_Data file system**.** New content generated by Step 3 and Step 4 are moved into the base directory on the Shared_Data file system for each study including the ALLSTUDIES package.
6. The Final step is to run the Python process to update the DataBrowser content making the new release files available for public download

2.3.2. Overview of AWS Components and Process



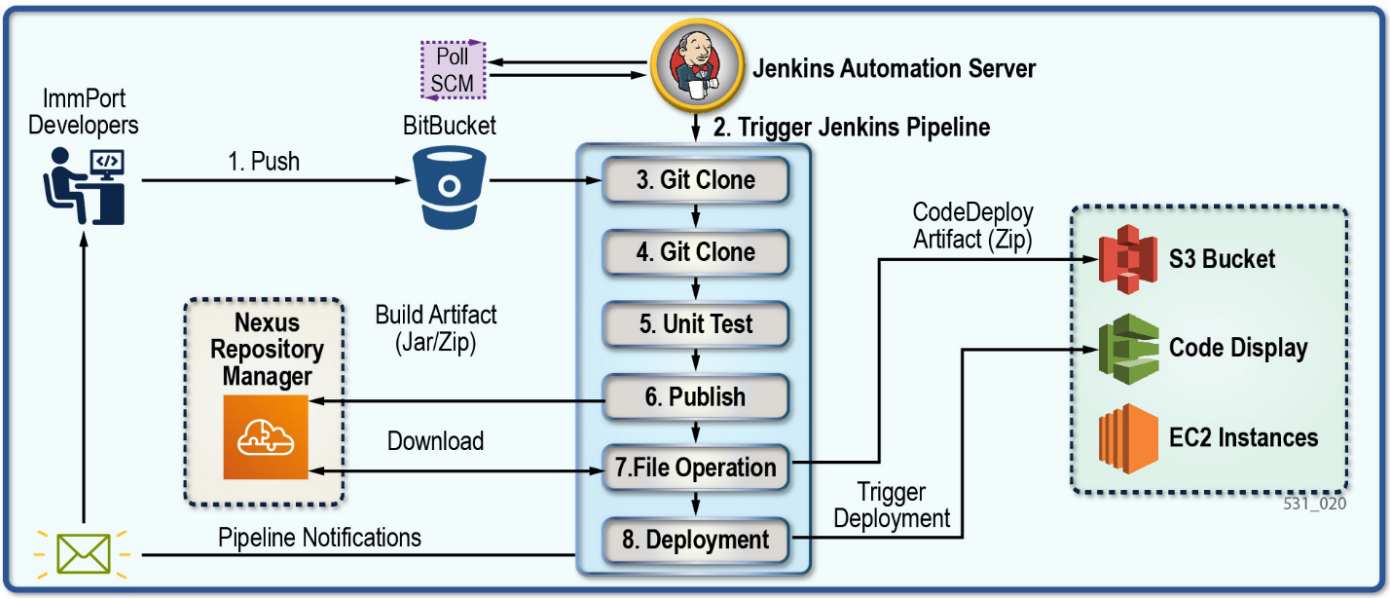
Steps to perform a data release are depicted above and outlined below. This section focuses on the production database and how files to support UI and ElasticSearch content are generated and deployed for use by API and UI.

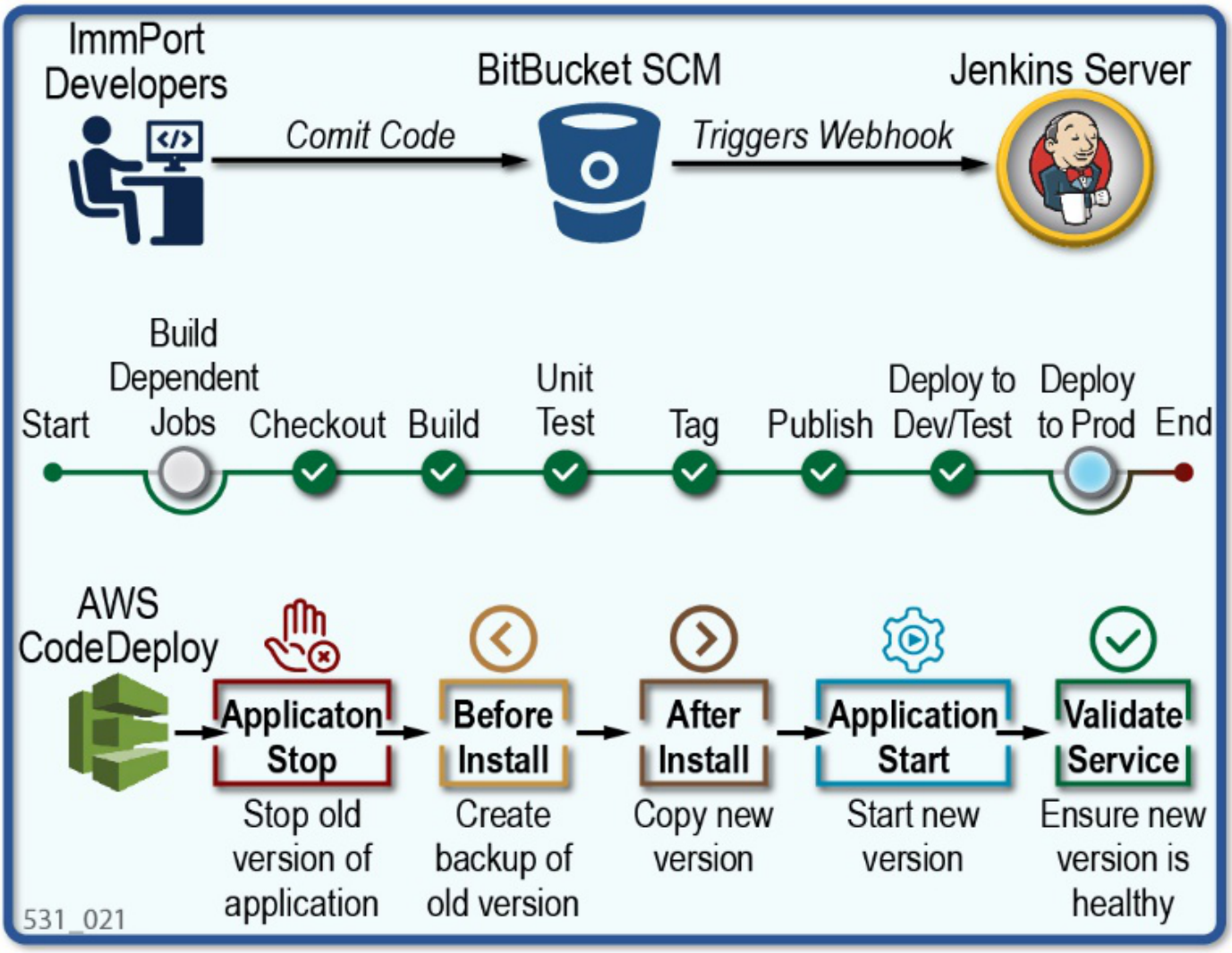
1. Several views and materialized tables are created on top of the MySQL Shared_Data database to support the UI and API.
2. When step 1 has been completed, the warehouse version of the database is copied and loaded into the DEV and Production MySQL Aurora databases on AWS.
3. The next step is to generate JSON files containing static content for use by the UI, to display graphics, data release notes, and to support indexing by Google for improved search engine optimization.
4. Files generated in step 3 are copied to the development AWS S3 buckets for QC testing and then, on the day of the data release, copied to the production AWS S3 buckets.
5. A similar process is used to extract information from the MySQL database to prepare and load this information into ElasticSearch. The development environment is loaded for QC testing and, on the day of the data release, the production environment is loaded and indexed.
6. Another process is run to generate content to support site search which is also indexed using ElasticSearch.

2.4. ImmPort Continuous Integration and Continuous Delivery (CICD)

CI/CD pipelines are the first prerequisites of cloud-native microservices architecture development. Continuous Integration (CI) enables continuous integration of source code into a single shared and easy to access repository. Continuous Delivery (CD) enables continuous delivery of the code stored in the repository to production. CI/CD creates a fast and effective process for getting a product to market and for releasing new features and bug fixes. These pipelines also enable organizations to bridge the gap between developers and customers efficiently and to create reliable, robust, and scalable applications.

The figure below depicts the CICD implementation architecture for the ImmPort applications hosted in the AWS environments. This architecture applies to the AWS and NIAID development applications hosted in AWS and AWS production applications hosted in AWS. Currently, the production and quality assurance applications at NIAID are deployed manually, and ImmPort is actively working on setting up another similar CICD architecture at the NIAID facility. We have installed standalone Jenkins servers in the development and production environments to run the CICD pipelines to build, test, tag, publish, and deploy to EC2 instance using AWS managed CodeDeploy service.





The figure above depicts a typical CICD pipeline for an ImmPort application. The standalone Jenkins server continuously polls for any code changes in the source code git repositories and triggers the underlying CICD pipeline for that repository. The pipeline then clones the source code and builds and runs the unit tests. On successful execution of the unit tests, the repository is tagged and a release artifact pushed to the Nexus artifact repository. Also, a Docker image is built and pushed to the Amazon Elastic Container Registry (ECR) for applications deployed as docker containers. The pipeline then deploys the artifact or the Docker image to the appropriate EC2 instance where the AWS CodeDeploy agent listens for instruction from the CodeDeploy service configured in the Jenkins plugin. Finally, the ImmPort application deploys onto the EC2 instance and the application restarts.

3. ImmPort OCICB

3.1. OVERVIEW

Most ImmPort public-facing applications are hosted using AWS infrastructure whereas operational applications and databases are hosted by the Office of Cyber Infrastructure and Computational Biology(OCICB), which manages

technologies supporting NIAID biomedical research programs. Since 2011 the OCICB and ImmPort teams have collaborated in the deployment and maintenance of applications in support of ImmPort operational activities.

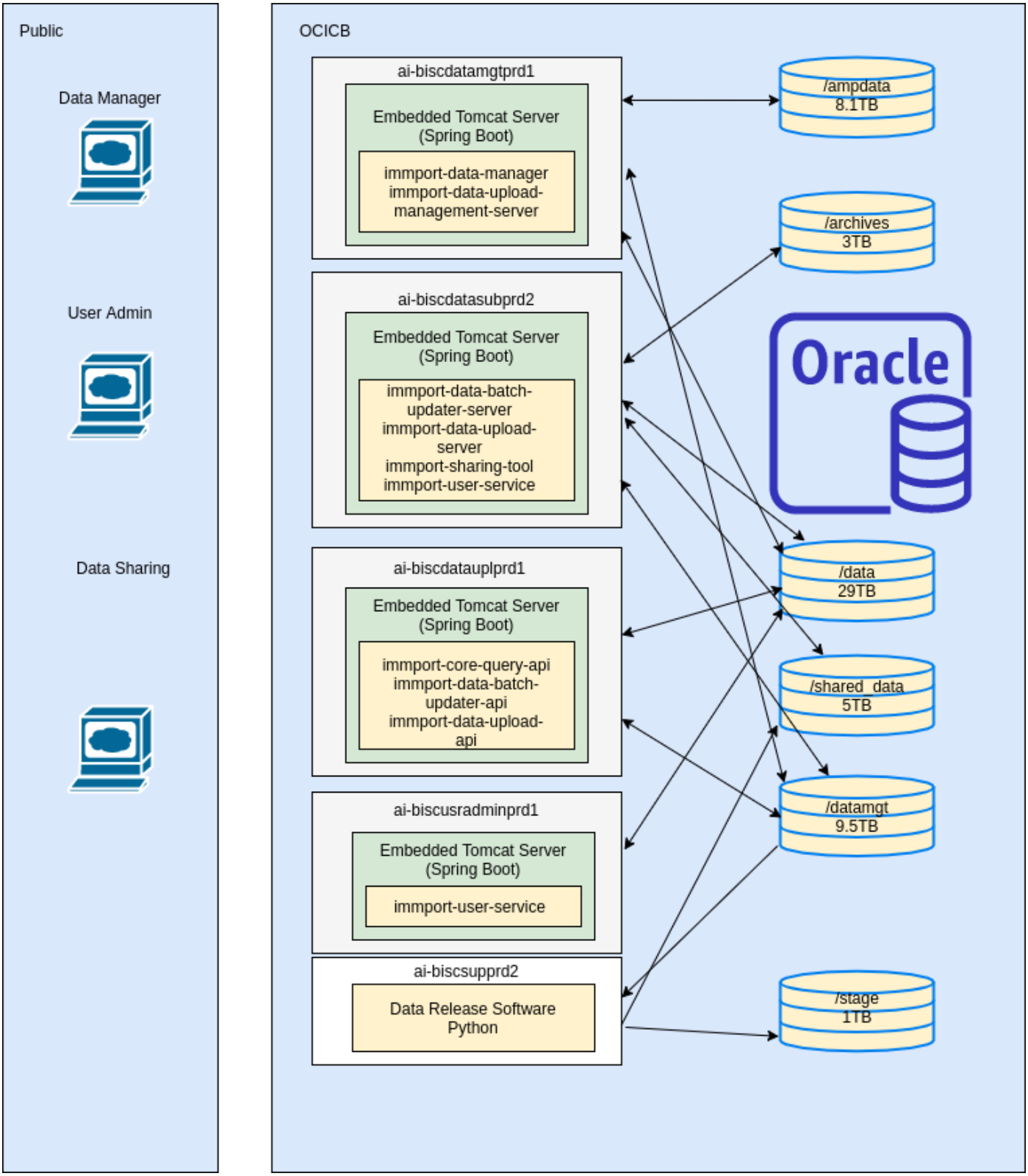
The OCICB infrastructure supports Production and QA environments. Applications hosted at NIAID are primarily focused on upload, management, QC, and curation of private data sets under embargo. This ensures private data sets have the highest level of security and access controls, provided by the NIAID facility during the embargo period when access to the data needs to be restricted to a limited set of users. When research and clinical data have been curated and released from the embargo and shared with the larger user community, these data continue to be made available to registered users using the Amazon Web Services (AWS) infrastructure.

3.1.1. Feature Summary

#	Capabilities/Features	Capability/Feature Description
1	User Administration	System administration capabilities include allowing users to request system access (register), approve registration requests, create user accounts, update user information, query user information, and deactivate users.
2	Log In/Off	Authentication and authorization capabilities including allowing users to login, logoff, and retrieve login information when the account or password is forgotten.
3	Manage NIAID/DAIT Programs	Contract/grant management capabilities including creating, searching, deleting, modifying, and viewing contracts and grants and assigning a PI for a contract or grant
4	Manage Research Project (RP) /Private Project Workspace (PPW)	Manage Research Projects (RP) including allowing a user designated as a PI or PM on a grant to create a project and it's associated RP and update the information associated with the project, manage user access to the RP.
5	Manage Collaborative Project (CP)	Manage the CP capabilities including allowing a user designated as a PI or PM on a contract or grant to create a CP, update the information associated with the CP, control user access to the CP, and share datasets in a CP.
6	Data Submission	Batch loading of experimental data into a RP for multiple types of relevant metadata to provide the minimum information for multiple experimental assay types.
	Data Update	Bulk editing of data
	Data Management	UI application for managing data
	Data Sharing	UI application to control the release of study information for public distribution
	Database Audit History and Archiving	Maintain a complete audit history of research data (including analysis toolset data created) that is both updated and deleted. The audit history is defined as the ability to capture "who", "what", and "when" of the data involved in a change or deletion to Research data contained in the ImmPort System. Additionally, audit and collect limited summary information with respect to auditing/tracking of user session activity on a limited number of database areas. The focus is to obtain summary information on system activity such as logins, information created and updated in the several areas of the Administration Module, and usage of baseline Analysis Tools. Additionally, provide the capability to audit and track user session log information.
	Core Query API	Provides programmatic access to the Core ImmPort Data

3.2. OCICB ARCHITECTURE

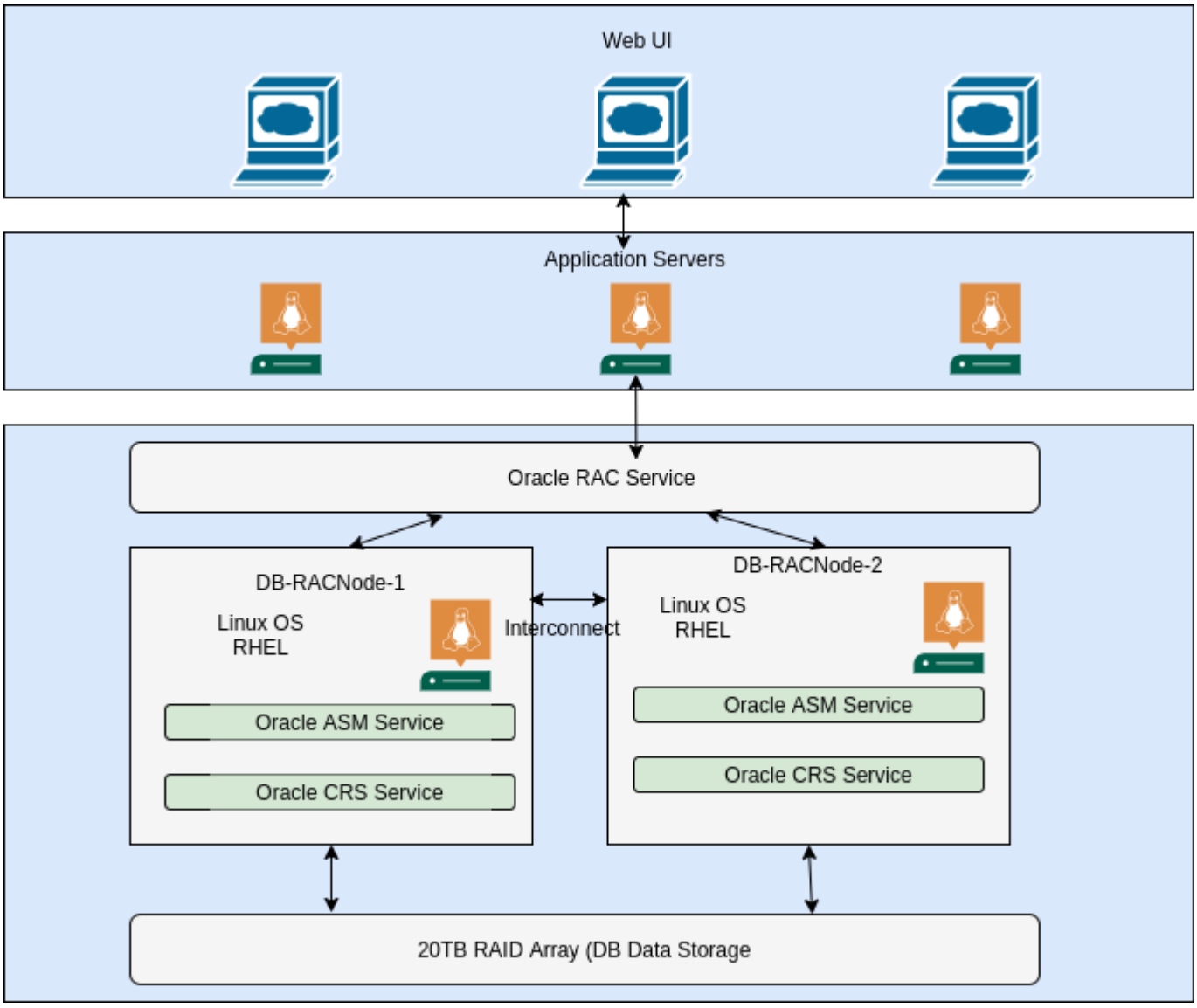
Drawing still in progress



3.3. DATABASE ARCHITECTURE

The ImmPort system database architecture is stored and maintained in an Oracle 17g Enterprise Edition database utilizing Real Application Clusters (RAC), installed on a Linux EL7 operating system. Installed database options include Oracle Partitioning, On-Line Analytical Processing (OLAP), and Oracle Data Mining (ODM) options. The RAC environment provides necessary system load distribution and load balancing, while at the same time providing system redundancy and failover capabilities.

In RAC, multiple active instances of the database server on different servers or processors can concurrently execute transactions against a shared database. It allows large tasks to be divided into subtasks and distributed among multiple nodes, which provides great efficiency. RAC automatically handles load balancing by distributing the load on multiple nodes and also supports parallel processing of data on multiple nodes. This becomes critical when handling the heavy processing loads required for many of the ImmPort analysis tools, or the large batch upload data submissions. RAC inherently provides high availability by guaranteeing that the database system is operational as long as one node in the cluster is up, and reliability by providing user-transparent transaction fail-over.



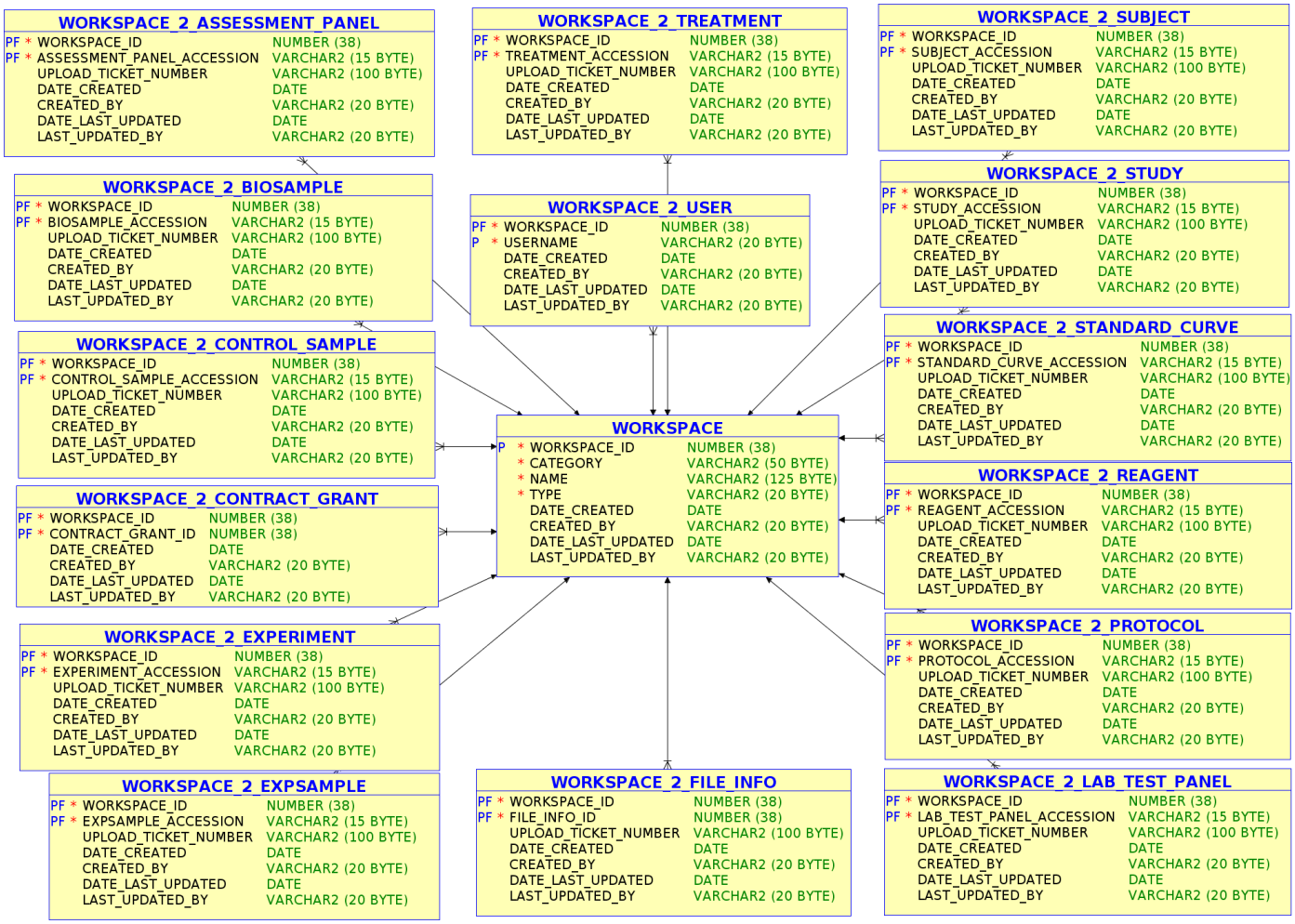
3.3.1. CORE_IMMPORT

3.3.1.1. Overview

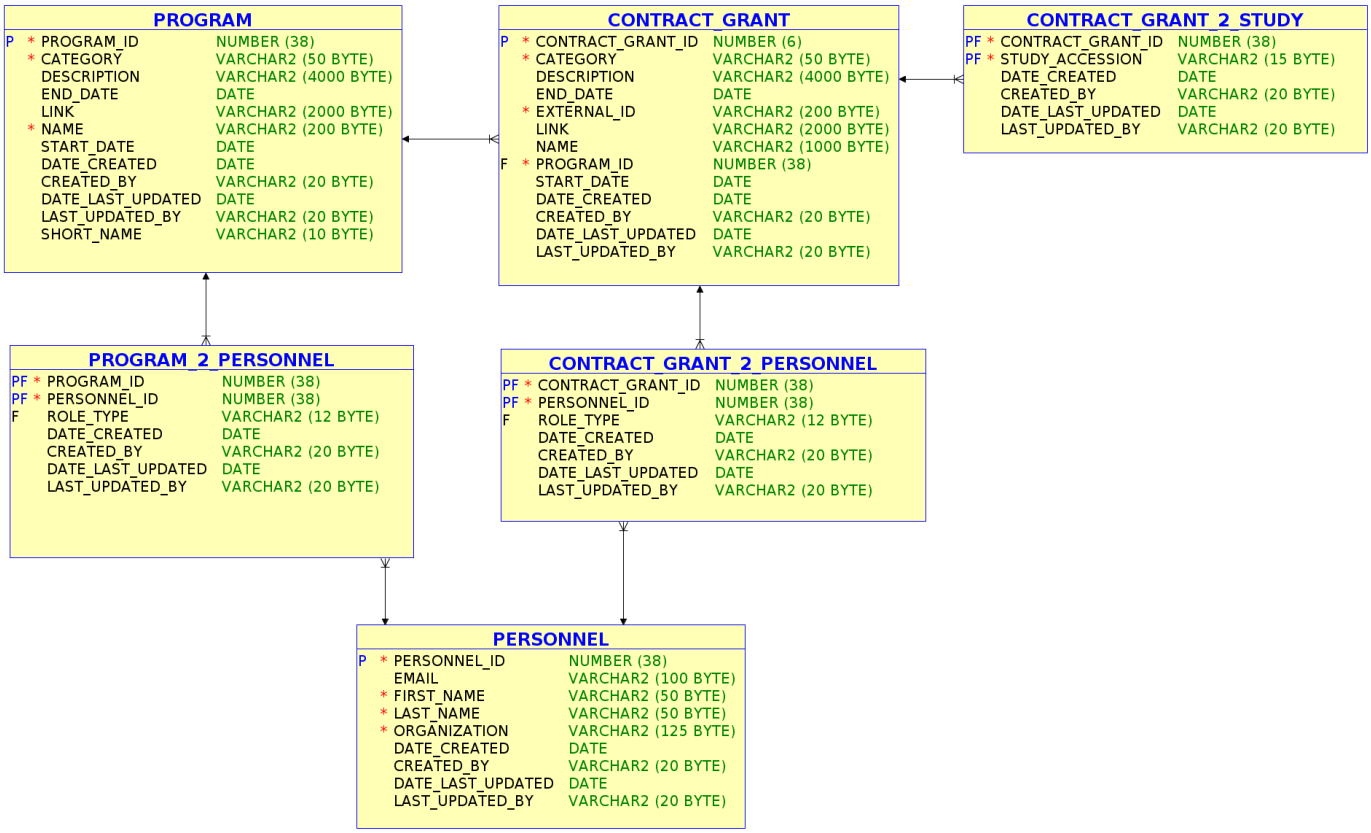
The CORE_IMMPORT database is the primary operational database where all read/write activity occurs. Almost all tables in this database include audit/editorial columns that track: date_created, created_by, data_last_updated and last_updated_by. In addition, the CORE_IMMPORT database is mirrored by the CORE_IMMPORT_HIST database, which tracks edits made to the tables in the CORE_IMMPORT database.

3.3.1.2. Workspace

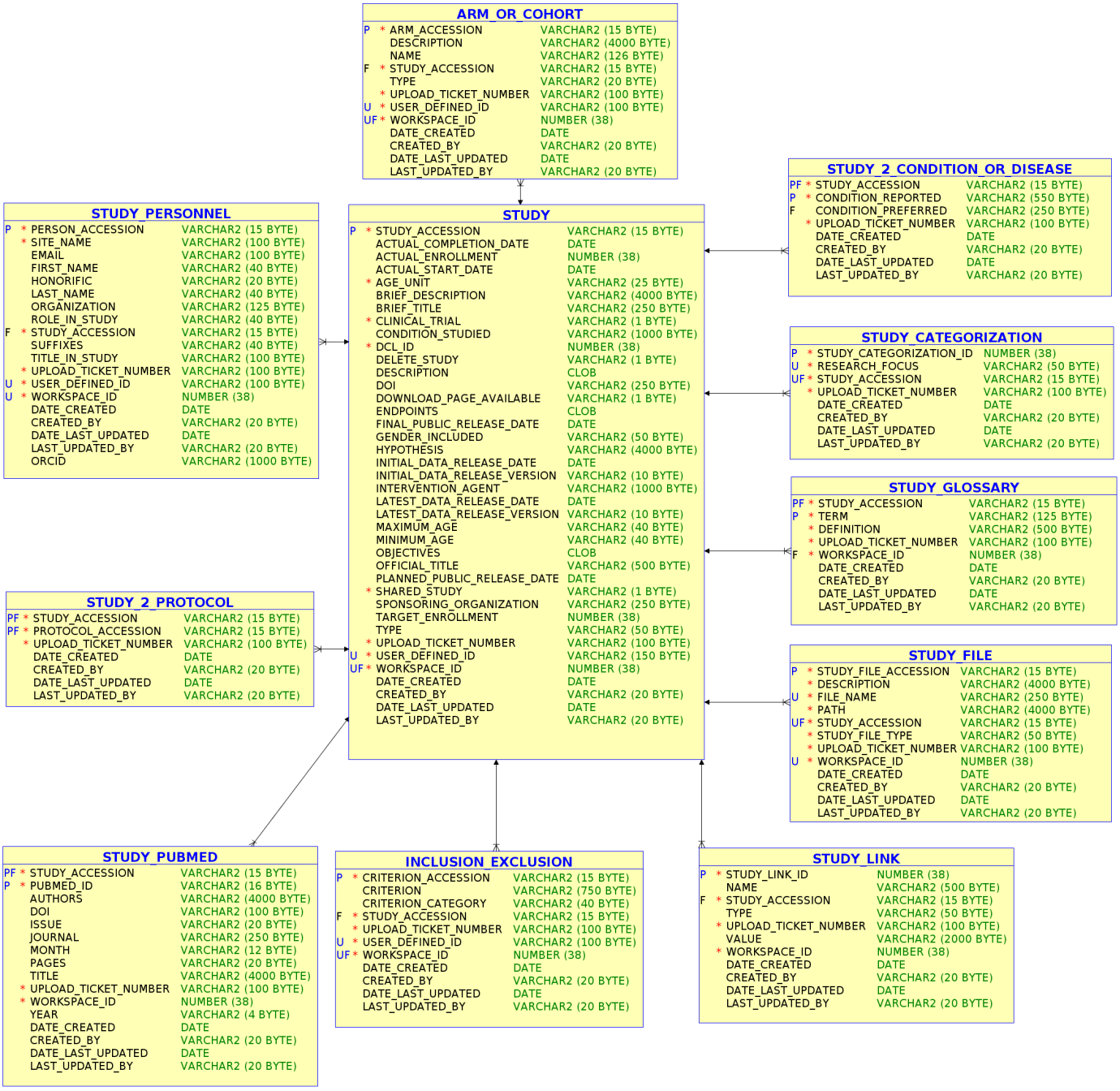
The workspace tables are important components for the process of releasing private studies to the public. In addition, they are used to control which users have access to individual rows in the database and the **workspace_2_user** supports access control to a workspace and all the objects linked to that workspace. In the workspace table there is a special workspace with workspace_id equal to zero. When rows from the base tables are released to the public as part of the Data Release sharing process, rows for the objects being released are inserted into the workspace_2_XXX table, with the workspace_id set to zero. This allows the ImmPort Data Release process to use views that make sure only rows in the base tables, linked to the workspace_2_XXX table, are included in the output.



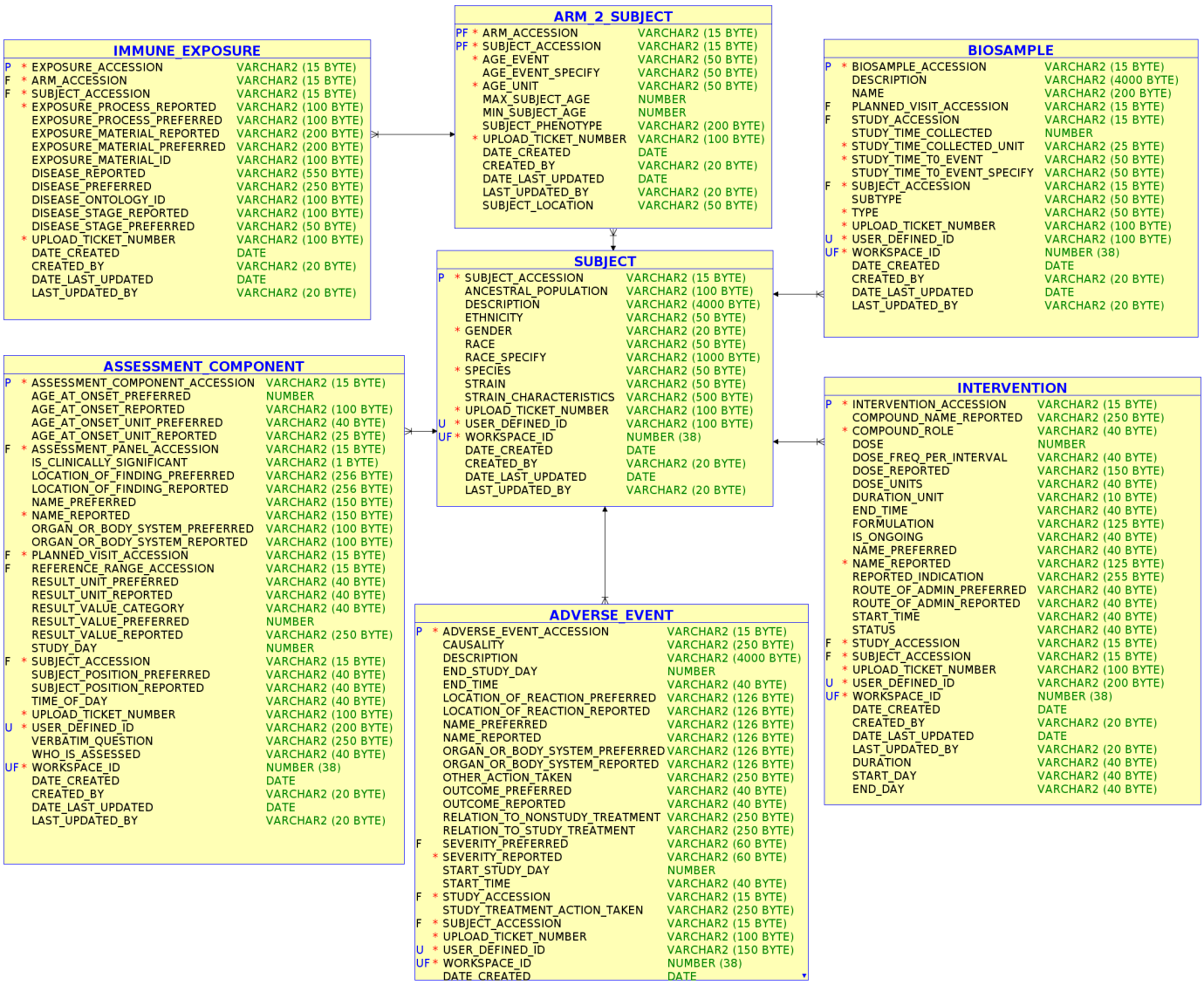
3.3.1.3. Administrative



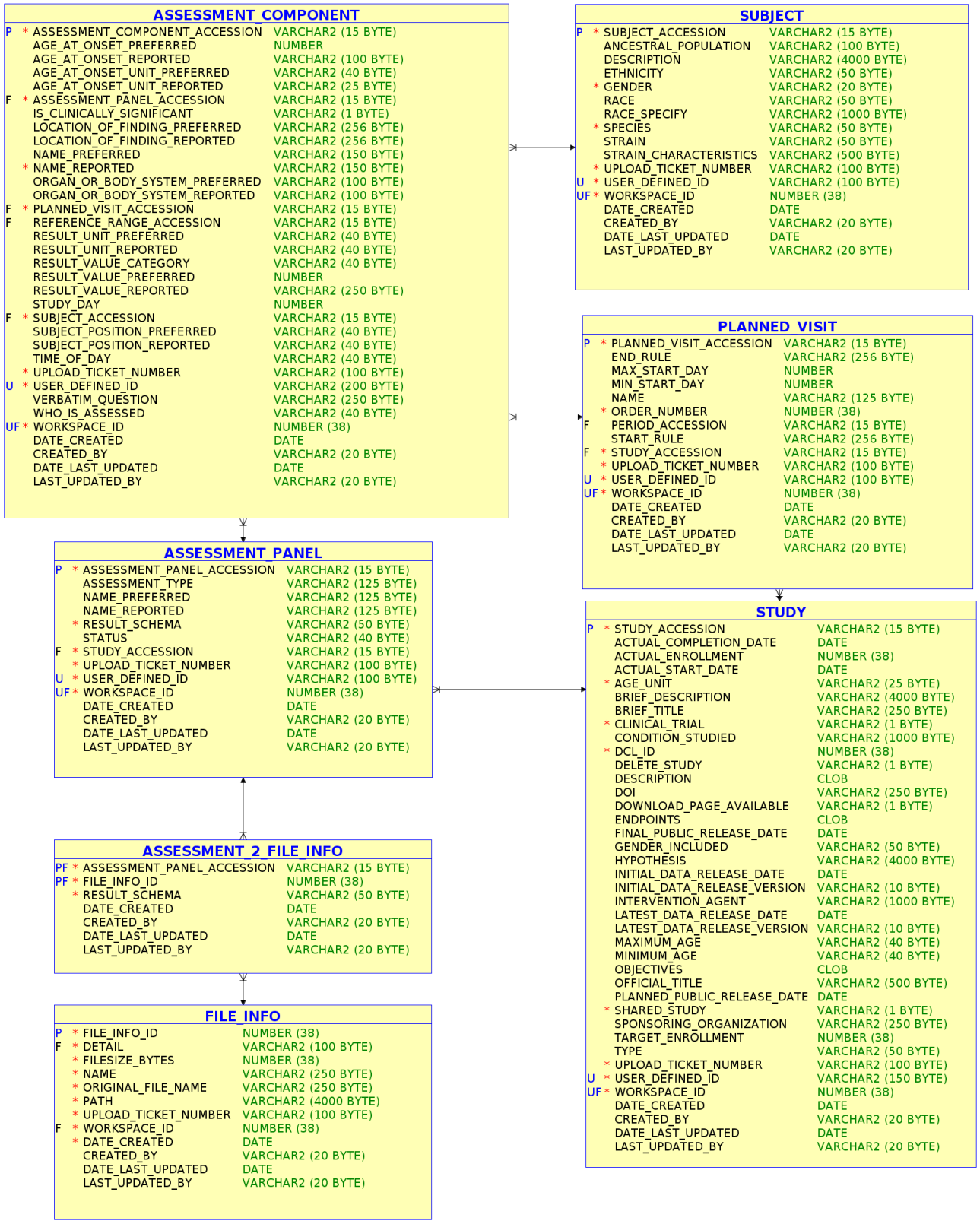
3.3.1.4. Study


ARM_OR_COHORT
P * ARM_ACCESSION VARCHAR2 (15 BYTE)
DESCRIPTION VARCHAR2 (4000 BYTE)
NAME VARCHAR2 (126 BYTE)
F * STUDY_ACCESSION VARCHAR2 (15 BYTE)
TYPE VARCHAR2 (20 BYTE)
* UPLOAD_TICKET_NUMBER VARCHAR2 (100 BYTE)
U * USER_DEFINED_ID VARCHAR2 (100 BYTE)
UF * WORKSPACE_ID NUMBER (38)
DATE_CREATED DATE
CREATED_BY VARCHAR2 (20 BYTE)
DATE_LAST_UPDATED DATE
LAST_UPDATED_BY VARCHAR2 (20 BYTE)
STUDY_PERSONNEL
P * PERSON_ACCESSION VARCHAR2 (15 BYTE)
* SITE_NAME VARCHAR2 (100 BYTE)
EMAIL VARCHAR2 (100 BYTE)
FIRST_NAME VARCHAR2 (40 BYTE)
HONORIFIC VARCHAR2 (20 BYTE)
LAST_NAME VARCHAR2 (40 BYTE)
ORGANIZATION VARCHAR2 (125 BYTE)
ROLE_IN_STUDY VARCHAR2 (40 BYTE)
F * STUDY_ACCESSION VARCHAR2 (15 BYTE)
SUFFIXES VARCHAR2 (40 BYTE)
TITLE_IN_STUDY VARCHAR2 (100 BYTE)
* UPLOAD_TICKET_NUMBER VARCHAR2 (100 BYTE)
U * USER_DEFINED_ID VARCHAR2 (100 BYTE)
U * WORKSPACE_ID NUMBER (38)
DATE_CREATED DATE
CREATED_BY VARCHAR2 (20 BYTE)
DATE_LAST_UPDATED DATE
LAST_UPDATED_BY VARCHAR2 (20 BYTE)
ORCID VARCHAR2 (1000 BYTE)
STUDY
P * STUDY_ACCESSION VARCHAR2 (15 BYTE)
ACTUAL_COMPLETION_DATE DATE
ACTUAL_ENROLLMENT NUMBER (38)
ACTUAL_START_DATE DATE
* AGE_UNIT VARCHAR2 (25 BYTE)
BRIEF_DESCRIPTION VARCHAR2 (4000 BYTE)
BRIEF_TITLE VARCHAR2 (250 BYTE)
* CLINICAL_TRIAL VARCHAR2 (1 BYTE)
CONDITION_STUDIED VARCHAR2 (1000 BYTE)
* DCL_ID NUMBER (38)
DELETE_STUDY VARCHAR2 (1 BYTE)
DESCRIPTION CLOB
DOI VARCHAR2 (250 BYTE)
DOWNLOAD_PAGE_AVAILABLE VARCHAR2 (1 BYTE)
ENDPOINTS CLOB
FINAL_PUBLIC_RELEASE_DATE DATE
GENDER_INCLUDED VARCHAR2 (50 BYTE)
HYPOTHESIS VARCHAR2 (4000 BYTE)
INITIAL_DATA_RELEASE_DATE DATE
INITIAL_DATA_RELEASE_VERSION VARCHAR2 (10 BYTE)
INTERVENTION_AGENT VARCHAR2 (1000 BYTE)
LATEST_DATA_RELEASE_DATE DATE
LATEST_DATA_RELEASE_VERSION VARCHAR2 (10 BYTE)
MAXIMUM_AGE VARCHAR2 (40 BYTE)
MINIMUM_AGE VARCHAR2 (40 BYTE)
OBJECTIVES CLOB
OFFICIAL_TITLE VARCHAR2 (500 BYTE)
PLANNED_PUBLIC_RELEASE_DATE DATE
* SHARED_STUDY VARCHAR2 (1 BYTE)
SPONSORING_ORGANIZATION VARCHAR2 (250 BYTE)
TARGET_ENROLLMENT NUMBER (38)
TYPE VARCHAR2 (50 BYTE)
* UPLOAD_TICKET_NUMBER VARCHAR2 (100 BYTE)
U * USER_DEFINED_ID VARCHAR2 (150 BYTE)
UF * WORKSPACE_ID NUMBER (38)
DATE_CREATED DATE
CREATED_BY VARCHAR2 (20 BYTE)
DATE_LAST_UPDATED DATE
LAST_UPDATED_BY VARCHAR2 (20 BYTE)
STUDY_2_CONDITION_OR_DISEASE
PF * STUDY_ACCESSION VARCHAR2 (15 BYTE)
P * CONDITION_REPORTED VARCHAR2 (550 BYTE)
F CONDITION_PREFERRED VARCHAR2 (250 BYTE)
* UPLOAD_TICKET_NUMBER VARCHAR2 (100 BYTE)
DATE_CREATED DATE
CREATED_BY VARCHAR2 (20 BYTE)
DATE_LAST_UPDATED DATE
LAST_UPDATED_BY VARCHAR2 (20 BYTE)
STUDY_CATEGORIZATION
P * STUDY_CATEGORIZATION_ID NUMBER (38)
U * RESEARCH_FOCUS VARCHAR2 (50 BYTE)
UF * STUDY_ACCESSION VARCHAR2 (15 BYTE)
* UPLOAD_TICKET_NUMBER VARCHAR2 (100 BYTE)
DATE_CREATED DATE
CREATED_BY VARCHAR2 (20 BYTE)
DATE_LAST_UPDATED DATE
LAST_UPDATED_BY VARCHAR2 (20 BYTE)
STUDY_GLOSSARY
PF * STUDY_ACCESSION VARCHAR2 (15 BYTE)
P * TERM VARCHAR2 (125 BYTE)
* DEFINITION VARCHAR2 (500 BYTE)
* UPLOAD_TICKET_NUMBER VARCHAR2 (100 BYTE)
F * WORKSPACE_ID NUMBER (38)
DATE_CREATED DATE
CREATED_BY VARCHAR2 (20 BYTE)
DATE_LAST_UPDATED DATE
LAST_UPDATED_BY VARCHAR2 (20 BYTE)
STUDY_2_PROTOCOL
PF * STUDY_ACCESSION VARCHAR2 (15 BYTE)
PF * PROTOCOL_ACCESSION VARCHAR2 (15 BYTE)
* UPLOAD_TICKET_NUMBER VARCHAR2 (100 BYTE)
DATE_CREATED DATE
CREATED_BY VARCHAR2 (20 BYTE)
DATE_LAST_UPDATED DATE
LAST_UPDATED_BY VARCHAR2 (20 BYTE)
STUDY_FILE
P * STUDY_FILE_ACCESSION VARCHAR2 (15 BYTE)
* DESCRIPTION VARCHAR2 (4000 BYTE)
U * FILE_NAME VARCHAR2 (250 BYTE)
* PATH VARCHAR2 (4000 BYTE)
UF * STUDY_ACCESSION VARCHAR2 (15 BYTE)
* STUDY_FILE_TYPE VARCHAR2 (50 BYTE)
* UPLOAD_TICKET_NUMBER VARCHAR2 (100 BYTE)
U * WORKSPACE_ID NUMBER (38)
DATE_CREATED DATE
CREATED_BY VARCHAR2 (20 BYTE)
DATE_LAST_UPDATED DATE
LAST_UPDATED_BY VARCHAR2 (20 BYTE)
STUDY_PUBMED
PF * STUDY_ACCESSION VARCHAR2 (15 BYTE)
P * PUBMED_ID VARCHAR2 (16 BYTE)
AUTHORS VARCHAR2 (4000 BYTE)
DOI VARCHAR2 (100 BYTE)
ISSUE VARCHAR2 (20 BYTE)
JOURNAL VARCHAR2 (250 BYTE)
MONTH VARCHAR2 (12 BYTE)
PAGES VARCHAR2 (20 BYTE)
TITLE VARCHAR2 (4000 BYTE)
* UPLOAD_TICKET_NUMBER VARCHAR2 (100 BYTE)
* WORKSPACE_ID NUMBER (38)
YEAR VARCHAR2 (4 BYTE)
DATE_CREATED DATE
CREATED_BY VARCHAR2 (20 BYTE)
DATE_LAST_UPDATED DATE
LAST_UPDATED_BY VARCHAR2 (20 BYTE)
INCLUSION_EXCLUSION
P * CRITERION_ACCESSION VARCHAR2 (15 BYTE)
CRITERION VARCHAR2 (750 BYTE)
CRITERION_CATEGORY VARCHAR2 (40 BYTE)
F * STUDY_ACCESSION VARCHAR2 (15 BYTE)
* UPLOAD_TICKET_NUMBER VARCHAR2 (100 BYTE)
U * USER_DEFINED_ID VARCHAR2 (100 BYTE)
UF * WORKSPACE_ID NUMBER (38)
DATE_CREATED DATE
CREATED_BY VARCHAR2 (20 BYTE)
DATE_LAST_UPDATED DATE
LAST_UPDATED_BY VARCHAR2 (20 BYTE)
STUDY_LINK
P * STUDY_LINK_ID NUMBER (38)
NAME VARCHAR2 (500 BYTE)
F * STUDY_ACCESSION VARCHAR2 (15 BYTE)
TYPE VARCHAR2 (50 BYTE)
* UPLOAD_TICKET_NUMBER VARCHAR2 (100 BYTE)
VALUE VARCHAR2 (2000 BYTE)
* WORKSPACE_ID NUMBER (38)
DATE_CREATED DATE
CREATED_BY VARCHAR2 (20 BYTE)
DATE_LAST_UPDATED DATE
LAST_UPDATED_BY VARCHAR2 (20 BYTE)

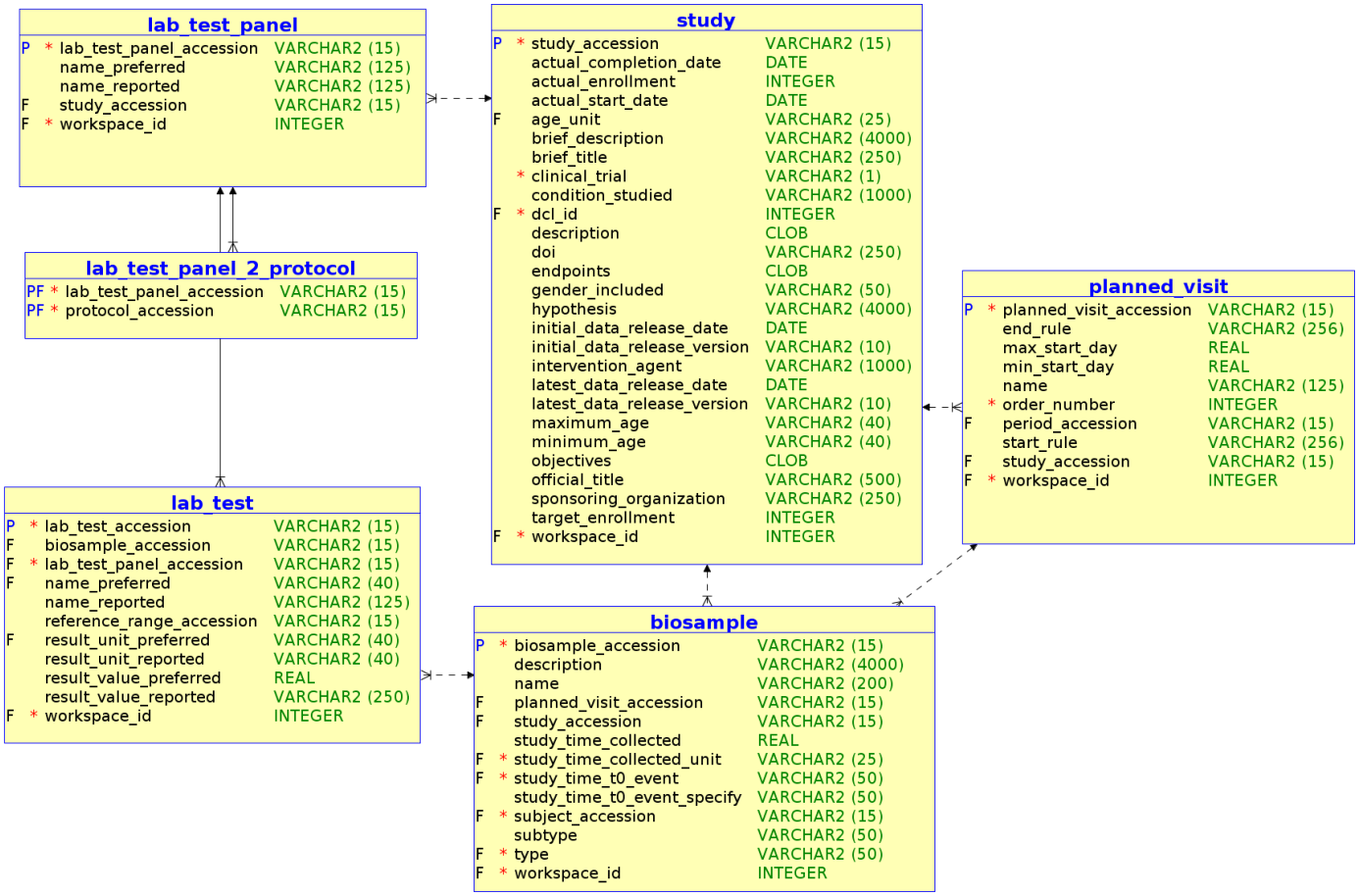
3.3.1.5. Subject



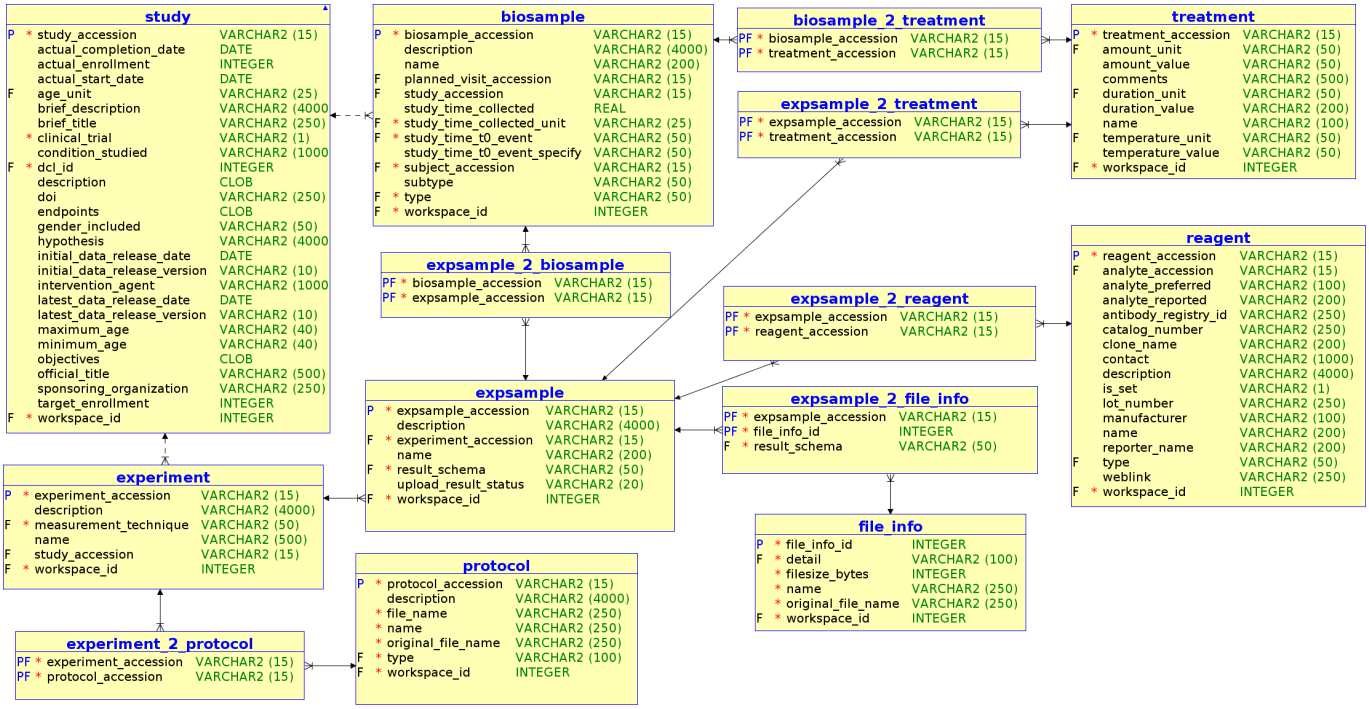
3.3.1.6. Assessment



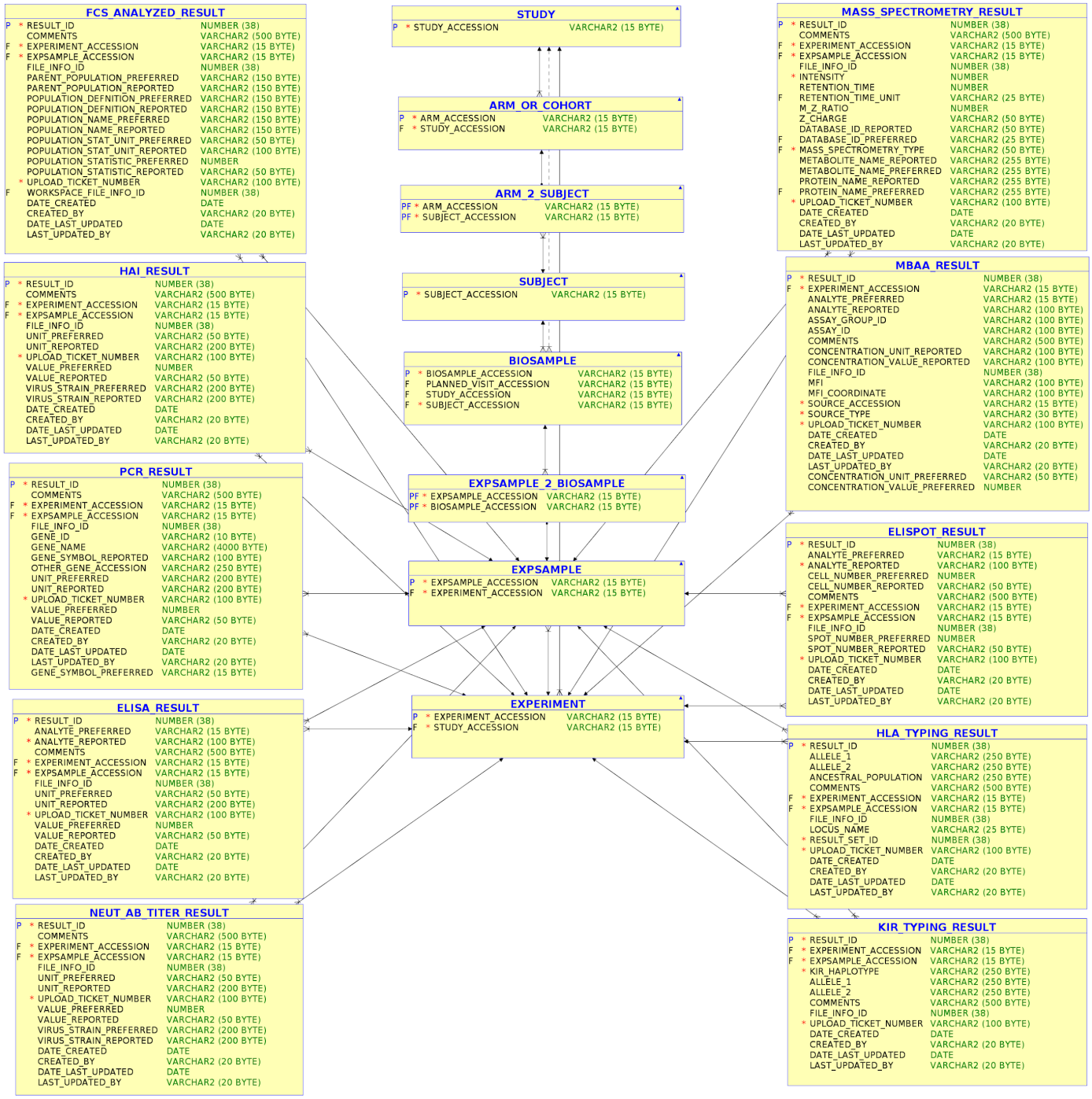
3.3.1.7. Lab Test



3.3.1.8. Experiment

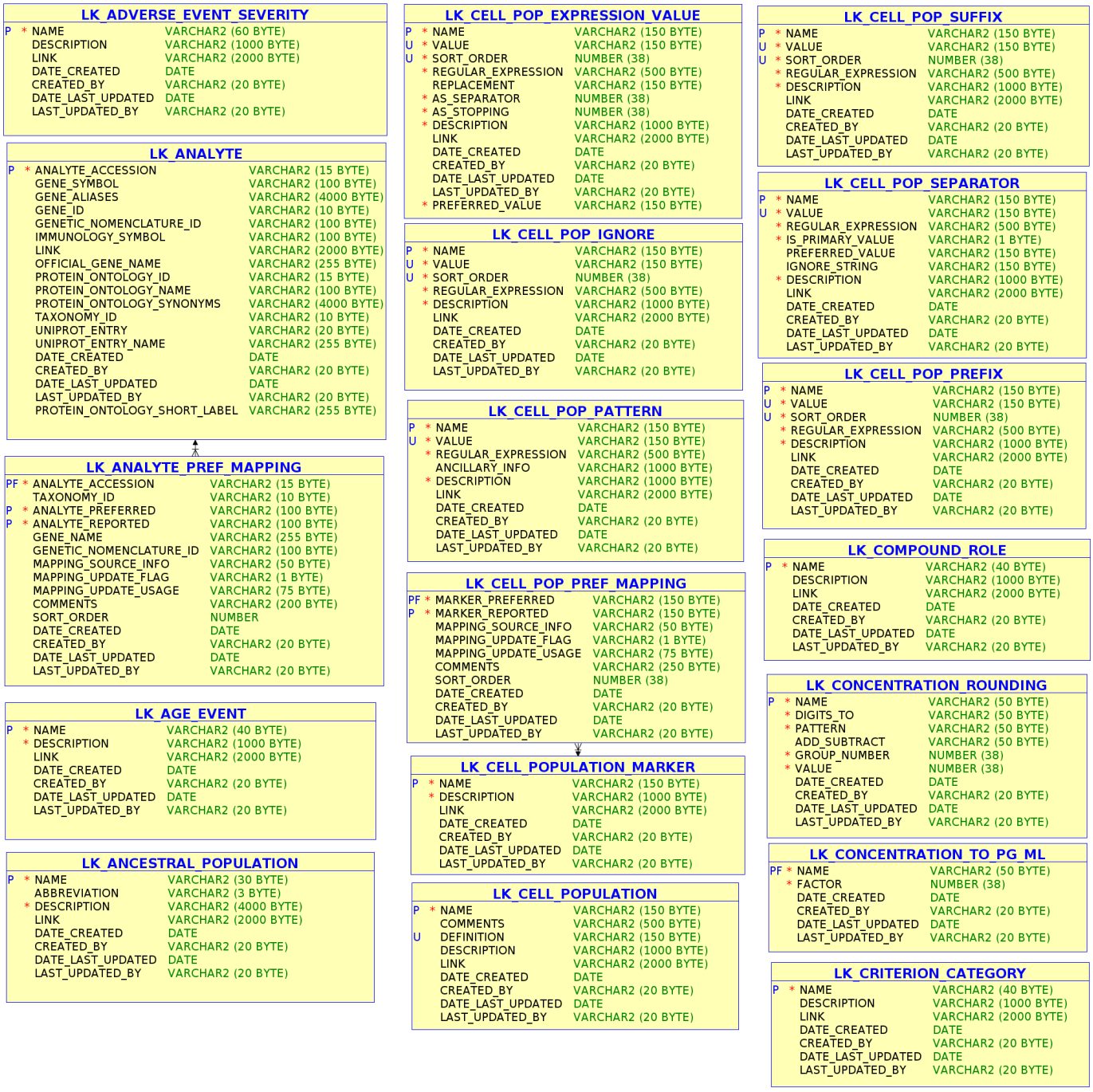


3.3.1.9. Assay Results



3.3.1.10. Lookup Tables - Part 1

There are approximately 65 tables in the IMMPORT_CORE schema, that ImmPort calls lookup tables, but other groups may refer to as controlled vocabulary tables. These tables are used to help harmonize the data from study to study. For many of the base tables ImmPort has both a reported_name and preferred_name and the preferred name is mapped to one of the lookup tables. In addition, several lookup tables are populated using terms from ontologies.



3.3.1.11. Lookup Tables - Part 2

LK_DATA_COMPLETENESS

P	* ID	NUMBER (38)
	DESCRIPTION	VARCHAR2 (1000 BYTE)
	DATE_CREATED	DATE
	CREATED_BY	VARCHAR2 (20 BYTE)
	DATE_LAST_UPDATED	DATE
	LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

LK_DISEASE

P	* NAME	VARCHAR2 (250 BYTE)
	* DISEASE_ONTOLOGY_ID	VARCHAR2 (50 BYTE)
	* DESCRIPTION	VARCHAR2 (1000 BYTE)
	LINK	VARCHAR2 (2000 BYTE)
	DATE_CREATED	DATE
	CREATED_BY	VARCHAR2 (20 BYTE)
	DATE_LAST_UPDATED	DATE
	LAST_UPDATED_BY	VARCHAR2 (20 BYTE)
	HUMAN_PHENOTYPE_ID	VARCHAR2 (50 BYTE)

LK_DISEASE_STAGE

P	* NAME	VARCHAR2 (50 BYTE)
	* DESCRIPTION	VARCHAR2 (1000 BYTE)
	LINK	VARCHAR2 (2000 BYTE)
	DATE_CREATED	DATE
	CREATED_BY	VARCHAR2 (20 BYTE)
	DATE_LAST_UPDATED	DATE
	LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

LK_ETHNICITY

P	* NAME	VARCHAR2 (50 BYTE)
	DESCRIPTION	VARCHAR2 (1000 BYTE)
	LINK	VARCHAR2 (2000 BYTE)
	DATE_CREATED	DATE
	CREATED_BY	VARCHAR2 (20 BYTE)
	DATE_LAST_UPDATED	DATE
	LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

LK_EXP_MEASUREMENT_TECH

P	* NAME	VARCHAR2 (50 BYTE)
	DESCRIPTION	VARCHAR2 (1000 BYTE)
	LINK	VARCHAR2 (2000 BYTE)
	DATE_CREATED	DATE
	CREATED_BY	VARCHAR2 (20 BYTE)
	DATE_LAST_UPDATED	DATE
	LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

LK_EXPOSURE_MATERIAL

P	* NAME	VARCHAR2 (200 BYTE)
	* EXPOSURE_MATERIAL_ID	VARCHAR2 (50 BYTE)
	* DESCRIPTION	VARCHAR2 (1000 BYTE)
	LINK	VARCHAR2 (2000 BYTE)
	DATE_CREATED	DATE
	CREATED_BY	VARCHAR2 (20 BYTE)
	DATE_LAST_UPDATED	DATE
	LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

LK_EXPOSURE_MATERIAL_PREF_MAP

P	* EXPOSURE_MATERIAL_REPORTED	VARCHAR2 (200 BYTE)
F	* EXPOSURE_MATERIAL_PREFERRED	VARCHAR2 (200 BYTE)
	COMMENTS	VARCHAR2 (200 BYTE)
	DATE_CREATED	DATE
	CREATED_BY	VARCHAR2 (20 BYTE)
	DATE_LAST_UPDATED	DATE
	LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

LK_EXPOSURE_PROCESS

P	* NAME	VARCHAR2 (100 BYTE)
	* DESCRIPTION	VARCHAR2 (1000 BYTE)
	LINK	VARCHAR2 (2000 BYTE)
	DATE_CREATED	DATE
	CREATED_BY	VARCHAR2 (20 BYTE)
	DATE_LAST_UPDATED	DATE
	LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

LK_EXPSAMPLE_RESULT_SCHEMA

P	* NAME	VARCHAR2 (50 BYTE)
	DESCRIPTION	VARCHAR2 (1000 BYTE)
	* TABLE_NAME	VARCHAR2 (30 BYTE)
	DATE_CREATED	DATE
	CREATED_BY	VARCHAR2 (20 BYTE)
	DATE_LAST_UPDATED	DATE
	LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

LK_FCS_HEADER_PATTERN

P	* NAME	VARCHAR2 (150 BYTE)
	* VALUE	VARCHAR2 (150 BYTE)
	* REGULAR_EXPRESSION	VARCHAR2 (500 BYTE)
	* PATTERN_TYPE	VARCHAR2 (50 BYTE)
	* DESCRIPTION	VARCHAR2 (1000 BYTE)
	LINK	VARCHAR2 (2000 BYTE)
	DATE_CREATED	DATE
	CREATED_BY	VARCHAR2 (20 BYTE)
	DATE_LAST_UPDATED	DATE
	LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

LK_FILE_DETAIL

P	* NAME	VARCHAR2 (100 BYTE)
	DESCRIPTION	VARCHAR2 (1000 BYTE)
	LINK	VARCHAR2 (2000 BYTE)
	DATE_CREATED	DATE
	CREATED_BY	VARCHAR2 (20 BYTE)
	DATE_LAST_UPDATED	DATE
	LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

LK_FILE_SYSTEM_OPERATION

P	* NAME	VARCHAR2 (50 BYTE)
	DESCRIPTION	VARCHAR2 (1000 BYTE)
	DATE_CREATED	DATE
	CREATED_BY	VARCHAR2 (20 BYTE)
	DATE_LAST_UPDATED	DATE
	LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

LK_FILE_SYSTEM_STATUS

P	* NAME	VARCHAR2 (50 BYTE)
	DESCRIPTION	VARCHAR2 (1000 BYTE)
	DATE_CREATED	DATE
	CREATED_BY	VARCHAR2 (20 BYTE)
	DATE_LAST_UPDATED	DATE
	LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

LK_GENDER

P	* NAME	VARCHAR2 (20 BYTE)
	DESCRIPTION	VARCHAR2 (1000 BYTE)
	LINK	VARCHAR2 (2000 BYTE)
	DATE_CREATED	DATE
	CREATED_BY	VARCHAR2 (20 BYTE)
	DATE_LAST_UPDATED	DATE
	LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

LK_HMDB

P	* HMDB_ID	VARCHAR2 (15 BYTE)
	* NAME	VARCHAR2 (255 BYTE)
	DESCRIPTION	VARCHAR2 (4000 BYTE)
	LINK	VARCHAR2 (2000 BYTE)
	DATE_CREATED	DATE
	CREATED_BY	VARCHAR2 (20 BYTE)
	DATE_LAST_UPDATED	DATE
	LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

LK_LAB_TEST_NAME

P	* NAME	VARCHAR2 (50 BYTE)
	CDISC_LAB_TEST_CODE	VARCHAR2 (50 BYTE)
	DESCRIPTION	VARCHAR2 (1000 BYTE)
	LAB_TEST_PANEL_NAME	VARCHAR2 (50 BYTE)
	LINK	VARCHAR2 (2000 BYTE)
	DATE_CREATED	DATE
	CREATED_BY	VARCHAR2 (20 BYTE)
	DATE_LAST_UPDATED	DATE
	LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

LK_LAB_TEST_PANEL_NAME

P	* NAME	VARCHAR2 (125 BYTE)
	DESCRIPTION	VARCHAR2 (1000 BYTE)
	LINK	VARCHAR2 (2000 BYTE)
	DATE_CREATED	DATE
	CREATED_BY	VARCHAR2 (20 BYTE)
	DATE_LAST_UPDATED	DATE
	LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

LK_LOCUS_NAME

P	* NAME	VARCHAR2 (100 BYTE)
	DESCRIPTION	VARCHAR2 (250 BYTE)
	LINK	VARCHAR2 (2000 BYTE)
	DATE_CREATED	DATE
	CREATED_BY	VARCHAR2 (20 BYTE)
	DATE_LAST_UPDATED	DATE
	LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

LK_MASS_SPECTROMETRY_TYPE

P	* NAME	VARCHAR2 (50 BYTE)
	DESCRIPTION	VARCHAR2 (4000 BYTE)
	LINK	VARCHAR2 (2000 BYTE)
	DATE_CREATED	DATE
	CREATED_BY	VARCHAR2 (20 BYTE)
	DATE_LAST_UPDATED	DATE
	LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

LK_ONTOLOGY

P	* NAME	VARCHAR2 (50 BYTE)
	DESCRIPTION	VARCHAR2 (1000 BYTE)
	LINK	VARCHAR2 (2000 BYTE)
	DATE_CREATED	DATE
	CREATED_BY	VARCHAR2 (20 BYTE)
	DATE_LAST_UPDATED	DATE
	LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

3.3.1.12. Lookup Tables - Part 3

LK_ORGANIZATION

P	*	NAME	VARCHAR2 (125 BYTE)
		LINK	VARCHAR2 (2000 BYTE)
		DATE_CREATED	DATE
		CREATED_BY	VARCHAR2 (20 BYTE)
		DATE_LAST_UPDATED	DATE
		LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

LK_PERSONNEL_ROLE

P	*	NAME	VARCHAR2 (40 BYTE)
		DESCRIPTION	VARCHAR2 (1000 BYTE)
		LINK	VARCHAR2 (2000 BYTE)
		DATE_CREATED	DATE
		CREATED_BY	VARCHAR2 (20 BYTE)
		DATE_LAST_UPDATED	DATE
		LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

LK_PLATE_TYPE

P	*	NAME	VARCHAR2 (50 BYTE)
		DESCRIPTION	VARCHAR2 (1000 BYTE)
		LINK	VARCHAR2 (2000 BYTE)
		DATE_CREATED	DATE
		CREATED_BY	VARCHAR2 (20 BYTE)
		DATE_LAST_UPDATED	DATE
		LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

LK_PROTOCOL_TYPE

P	*	NAME	VARCHAR2 (100 BYTE)
		DESCRIPTION	VARCHAR2 (1000 BYTE)
		LINK	VARCHAR2 (2000 BYTE)
		DATE_CREATED	DATE
		CREATED_BY	VARCHAR2 (20 BYTE)
		DATE_LAST_UPDATED	DATE
		LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

LK_PROTEIN_NAME

P	*	NAME	VARCHAR2 (255 BYTE)
	*	UNIPROT_ID	VARCHAR2 (50 BYTE)
		UNIPROT_GENE_NAME	VARCHAR2 (255 BYTE)
		DESCRIPTION	VARCHAR2 (4000 BYTE)
		LINK	VARCHAR2 (2000 BYTE)
		DATE_CREATED	DATE
		CREATED_BY	VARCHAR2 (20 BYTE)
		DATE_LAST_UPDATED	DATE
		LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

LK_PUBLIC_REPOSITORY

P	*	NAME	VARCHAR2 (50 BYTE)
		DESCRIPTION	VARCHAR2 (1000 BYTE)
		LINK	VARCHAR2 (2000 BYTE)
		DATE_CREATED	DATE
		CREATED_BY	VARCHAR2 (20 BYTE)
		DATE_LAST_UPDATED	DATE
		LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

LK_RACE

P	*	NAME	VARCHAR2 (50 BYTE)
		DESCRIPTION	VARCHAR2 (1000 BYTE)
		LINK	VARCHAR2 (2000 BYTE)
		DATE_CREATED	DATE
		CREATED_BY	VARCHAR2 (20 BYTE)
		DATE_LAST_UPDATED	DATE
		LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

LK_REAGENT_TYPE

P	*	NAME	VARCHAR2 (50 BYTE)
		DESCRIPTION	VARCHAR2 (1000 BYTE)
		LINK	VARCHAR2 (2000 BYTE)
		DATE_CREATED	DATE
		CREATED_BY	VARCHAR2 (20 BYTE)
		DATE_LAST_UPDATED	DATE
		LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

LK_RELEASE_STATUS

P	*	NAME	VARCHAR2 (50 BYTE)
		DESCRIPTION	VARCHAR2 (1000 BYTE)
		DATE_CREATED	DATE
		CREATED_BY	VARCHAR2 (20 BYTE)
		DATE_LAST_UPDATED	DATE
		LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

LK_RESEARCH_FOCUS

P	*	NAME	VARCHAR2 (50 BYTE)
		DESCRIPTION	VARCHAR2 (1000 BYTE)
		LINK	VARCHAR2 (2000 BYTE)
		DATE_CREATED	DATE
		CREATED_BY	VARCHAR2 (20 BYTE)
		DATE_LAST_UPDATED	DATE
		LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

LK_SAMPLE_TYPE

P	*	NAME	VARCHAR2 (50 BYTE)
		DESCRIPTION	VARCHAR2 (1000 BYTE)
		LINK	VARCHAR2 (2000 BYTE)
		DATE_CREATED	DATE
		CREATED_BY	VARCHAR2 (20 BYTE)
		DATE_LAST_UPDATED	DATE
		LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

LK_SOURCE_TYPE

P	*	NAME	VARCHAR2 (30 BYTE)
		DESCRIPTION	VARCHAR2 (1000 BYTE)
		LINK	VARCHAR2 (2000 BYTE)
		DATE_CREATED	DATE
		CREATED_BY	VARCHAR2 (20 BYTE)
		DATE_LAST_UPDATED	DATE
		LAST_UPDATED_BY	VARCHAR2 (20 BYTE)
	*	TABLE_NAME	VARCHAR2 (30 BYTE)

LK_SPECIES

P	*	NAME	VARCHAR2 (30 BYTE)
		COMMON_NAME	VARCHAR2 (100 BYTE)
		LINK	VARCHAR2 (2000 BYTE)
U	*	TAXONOMY_ID	VARCHAR2 (10 BYTE)
		DATE_CREATED	DATE
		CREATED_BY	VARCHAR2 (20 BYTE)
		DATE_LAST_UPDATED	DATE
		LAST_UPDATED_BY	VARCHAR2 (20 BYTE)
	*	TAXONOMY_ID_SUBSET	VARCHAR2 (10 BYTE)

LK_STUDY_CONDITION_PREF_MAPPNG

P	*	CONDITION_REPORTED	VARCHAR2 (550 BYTE)
F	*	CONDITION_PREFERRED	VARCHAR2 (250 BYTE)
		COMMENTS	VARCHAR2 (200 BYTE)
		DATE_CREATED	DATE
		CREATED_BY	VARCHAR2 (20 BYTE)
		DATE_LAST_UPDATED	DATE
		LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

LK_STUDY_FILE_TYPE

P	*	NAME	VARCHAR2 (50 BYTE)
		DESCRIPTION	VARCHAR2 (1000 BYTE)
		LINK	VARCHAR2 (2000 BYTE)
		DATE_CREATED	DATE
		CREATED_BY	VARCHAR2 (20 BYTE)
		DATE_LAST_UPDATED	DATE
		LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

LK_STUDY_PANEL

P	*	NAME	VARCHAR2 (100 BYTE)
		COLLAPSIBLE	VARCHAR2 (1 BYTE)
		DESCRIPTION	VARCHAR2 (1000 BYTE)
		DISPLAY_NAME	VARCHAR2 (100 BYTE)
		SORT_ORDER	NUMBER (38)
		VISIBLE	VARCHAR2 (1 BYTE)
		DATE_CREATED	DATE
		CREATED_BY	VARCHAR2 (20 BYTE)
		DATE_LAST_UPDATED	DATE
		LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

LK_SUBJECT_LOCATION

P	*	NAME	VARCHAR2 (50 BYTE)
	*	DESCRIPTION	VARCHAR2 (4000 BYTE)
		LINK	VARCHAR2 (2000 BYTE)
		DATE_CREATED	DATE
		CREATED_BY	VARCHAR2 (20 BYTE)
		DATE_LAST_UPDATED	DATE
		LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

LK_T0_EVENT

P	*	NAME	VARCHAR2 (50 BYTE)
	*	DESCRIPTION	VARCHAR2 (1000 BYTE)
		LINK	VARCHAR2 (2000 BYTE)
		DATE_CREATED	DATE
		CREATED_BY	VARCHAR2 (20 BYTE)
		DATE_LAST_UPDATED	DATE
		LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

3.3.1.13. Lookup Tables - Part 4

LK_TEMPLATE_MAPPING

P	*	TEMPLATE_NAME	VARCHAR2 (100 BYTE)
P	*	FILE_TYPE	VARCHAR2 (30 BYTE)
P	*	FILE_NAME	VARCHAR2 (250 BYTE)
		COMMENTS	VARCHAR2 (250 BYTE)
		DESCRIPTION	VARCHAR2 (1000 BYTE)
F		FILE_DETAIL	VARCHAR2 (100 BYTE)
		LINK	VARCHAR2 (2000 BYTE)
F		RESULT_SCHEMA	VARCHAR2 (50 BYTE)
		TEMPLATE_TYPE	VARCHAR2 (30 BYTE)
		DATE_CREATED	DATE
		CREATED_BY	VARCHAR2 (20 BYTE)
		DATE_LAST_UPDATED	DATE
		LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

LK_TIME_UNIT

P	*	NAME	VARCHAR2 (25 BYTE)
	*	DESCRIPTION	VARCHAR2 (1000 BYTE)
		LINK	VARCHAR2 (2000 BYTE)
		DATE_CREATED	DATE
		CREATED_BY	VARCHAR2 (20 BYTE)
		DATE_LAST_UPDATED	DATE
		LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

LK_UNIT_OF_MEASURE

P	*	NAME	VARCHAR2 (50 BYTE)
		DESCRIPTION	VARCHAR2 (1000 BYTE)
		LINK	VARCHAR2 (2000 BYTE)
	*	TYPE	VARCHAR2 (50 BYTE)
		DATE_CREATED	DATE
		CREATED_BY	VARCHAR2 (20 BYTE)
		DATE_LAST_UPDATED	DATE
		LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

LK_TRANSCRIPT_TYPE

P	*	NAME	VARCHAR2 (50 BYTE)
		DESCRIPTION	VARCHAR2 (1000 BYTE)
		LINK	VARCHAR2 (2000 BYTE)
		DATE_CREATED	DATE
		CREATED_BY	VARCHAR2 (20 BYTE)
		DATE_LAST_UPDATED	DATE
		LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

LK_UPLOAD_METHOD

P	*	NAME	VARCHAR2 (50 BYTE)
		DESCRIPTION	VARCHAR2 (1000 BYTE)
		DATE_CREATED	DATE
		CREATED_BY	VARCHAR2 (20 BYTE)
		DATE_LAST_UPDATED	DATE
		LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

LK_UPLOAD_STATUS

P	*	NAME	VARCHAR2 (50 BYTE)
		DESCRIPTION	VARCHAR2 (1000 BYTE)
		DATE_CREATED	DATE
		CREATED_BY	VARCHAR2 (20 BYTE)
		DATE_LAST_UPDATED	DATE
		LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

LK_USER_ROLE_TYPE

P	*	NAME	VARCHAR2 (2 BYTE)
		DESCRIPTION	VARCHAR2 (1000 BYTE)
		DATE_CREATED	DATE
		CREATED_BY	VARCHAR2 (20 BYTE)
		DATE_LAST_UPDATED	DATE
		LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

LK_VIRUS_STRAIN

P	*	NAME	VARCHAR2 (200 BYTE)
		CENTER_ID_NAME_SEASON_LIST	VARCHAR2 (500 BYTE)
		DESCRIPTION	VARCHAR2 (1000 BYTE)
		LINK	VARCHAR2 (2000 BYTE)
		SEASON_LIST	VARCHAR2 (100 BYTE)
	*	TAXONOMY_ID	NUMBER (38)
		VIRUS_NAME	VARCHAR2 (10 BYTE)
		DATE_CREATED	DATE
		CREATED_BY	VARCHAR2 (20 BYTE)
		DATE_LAST_UPDATED	DATE
		LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

LK_VISIBILITY_CATEGORY

P	*	NAME	VARCHAR2 (50 BYTE)
		DESCRIPTION	VARCHAR2 (1000 BYTE)
		DATE_CREATED	DATE
		CREATED_BY	VARCHAR2 (20 BYTE)
		DATE_LAST_UPDATED	DATE
		LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

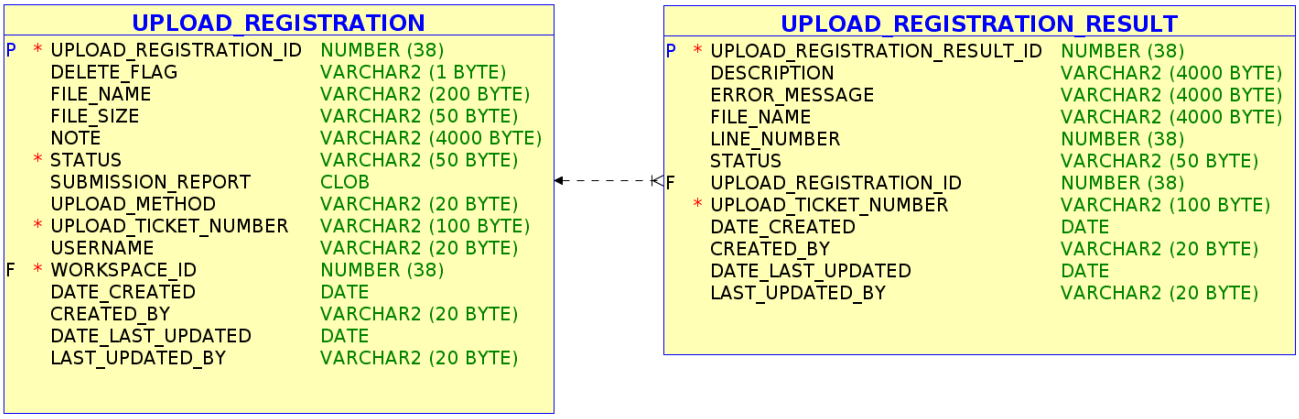
LK_WORKSPACE_APPLICATION

P	*	NAME	VARCHAR2 (50 BYTE)
		DESCRIPTION	VARCHAR2 (1000 BYTE)
		DATE_CREATED	DATE
		CREATED_BY	VARCHAR2 (20 BYTE)
		DATE_LAST_UPDATED	DATE
		LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

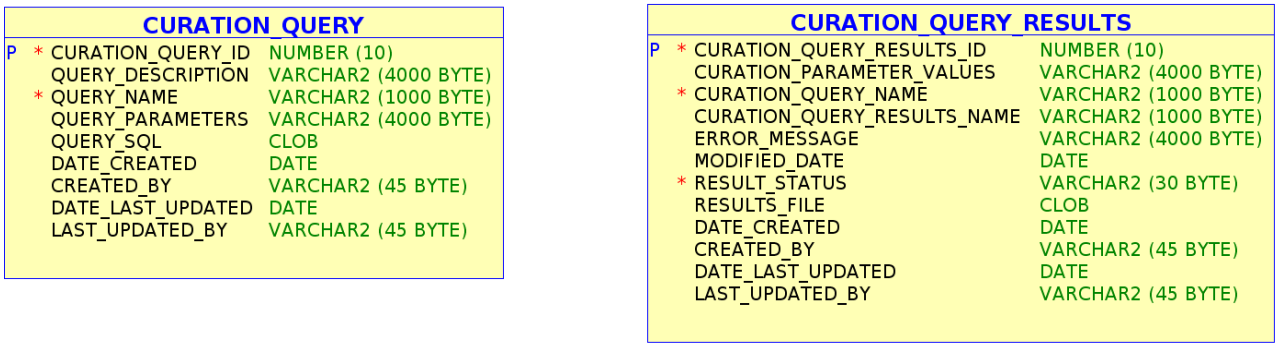
LK_WORKSPACE_TYPE

P	*	NAME	VARCHAR2 (20 BYTE)
	*	DESCRIPTION	VARCHAR2 (1000 BYTE)
		DATE_CREATED	DATE
		CREATED_BY	VARCHAR2 (20 BYTE)
		DATE_LAST_UPDATED	DATE
		LAST_UPDATED_BY	VARCHAR2 (20 BYTE)

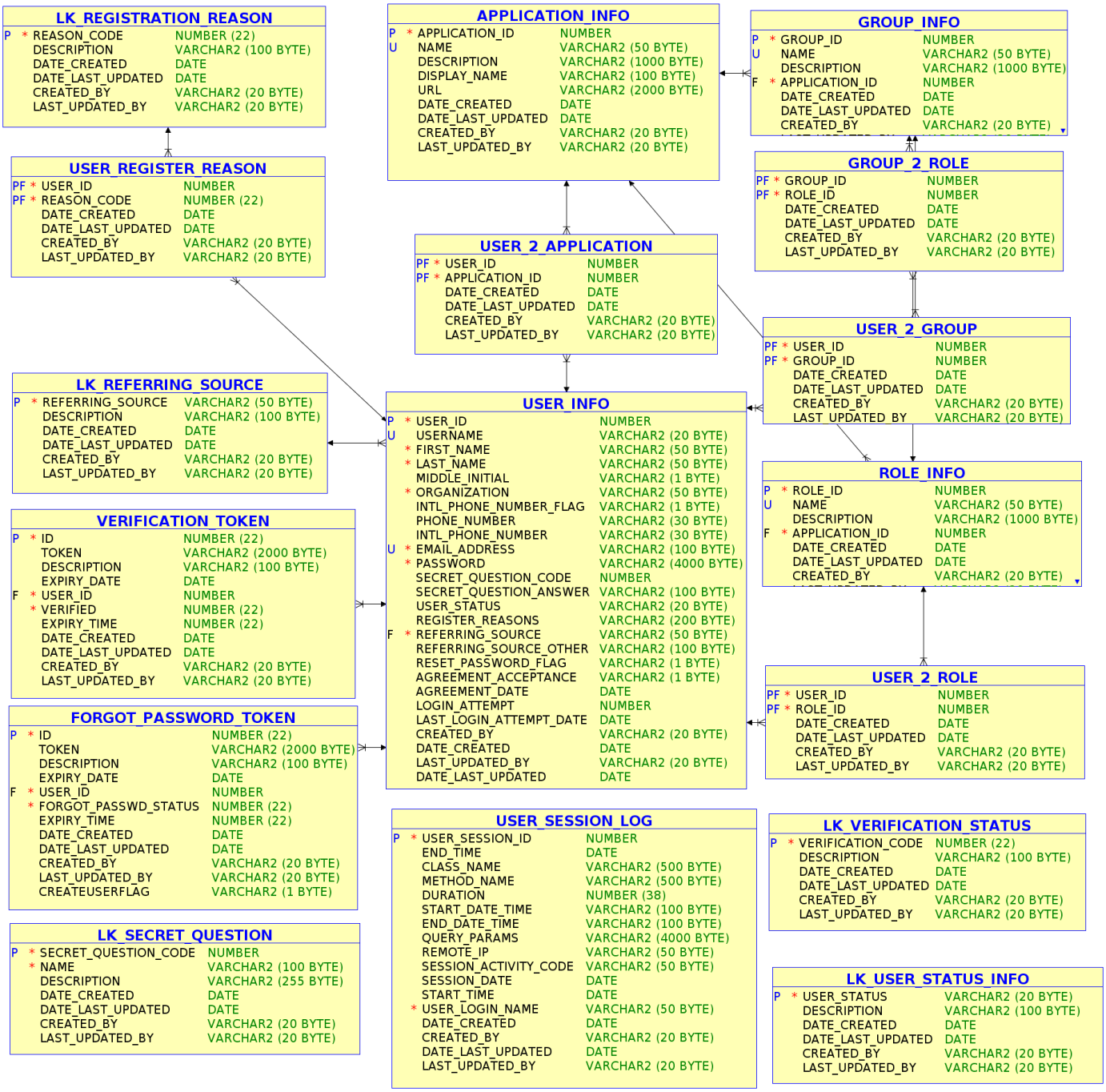
3.3.1.14. Upload Registration



3.3.1.15. Curation



3.3.1.16. BISC Security



3.4. Aspera Server

We integrated technology from Aspera, an IBM company, into the ImmPort system architecture. The Aspera Connect Server using the patented FASP® technology allows for optimized data transfer speeds across the Internet; we have utilized this technology for data submission and data downloads for large files.

The Aspera security infrastructure provides user authentication and permissions on file systems. A free Aspera Connect Client provides a plug-in for users to install locally to take advantage of the FASP® UDP

based transfer optimization. The Aspera SDK is utilized currently for accessing the Aspera Connect Server from the ImmPort application and data submission servers to queue up transfer tasks and return results to users.

More detailed documentation regarding the Aspera Connect Server, the Aspera SDK, FASP® technology, and the Aspera Connect Client can be found online at www.ibm.com/products/aspera.

3.5. HOSTED APPLICATIONS

3.5.1. ImmPort Core Query API

The Core Query API provides programmatic access to Core ImmPort Data. This API works as a SQL query tool to access data in the relational database (Oracle). The API returns a tab delimited output by default. The Core Query API endpoints can be accessed directly by a user or by an application. All requests to the Core Query API require authentication and the Core Query API uses tokens for authentication. Users obtain tokens by posting to the ImmPort Authentication URL https://auth.immport.org/auth/token with a username and password. They must include the authentication token as an Authorization: bearer in the custom HTTP header.

3.5.1.1. Feature Summary

The primary function of the core query api as currently developed is to assist curators in executing and saving database SQL queries that help with daily operational tasks in getting data collected, QC'ed, shared and curated. Based on usefulness to the internal curation team, it can be evaluated over time if the application would also be useful for external users.

The endpoints of the API are listed below

HTTP URL	Parameters (pass in the body of the request)	Description	GET/POST
/query/delete/{queryID}		Deletes the query with the specified query id stored in the database	GET
/query/id/{queryId}		Get backs the query object in json format	GET
/query/all		Gets all the query objects	GET
/query/username/{username		Gets all the query objects for the specified username	GET
/query/add	JSON Query Object Example : { "queryName": "Study query", "queryDescription": "study query", "querySql": "select * from study where study_accession = '{study_accession}'" }	Adding a query object to be stored in the database	POST
query/update	JSON Query Object	Updating query object in the database. The json object passed in	POST

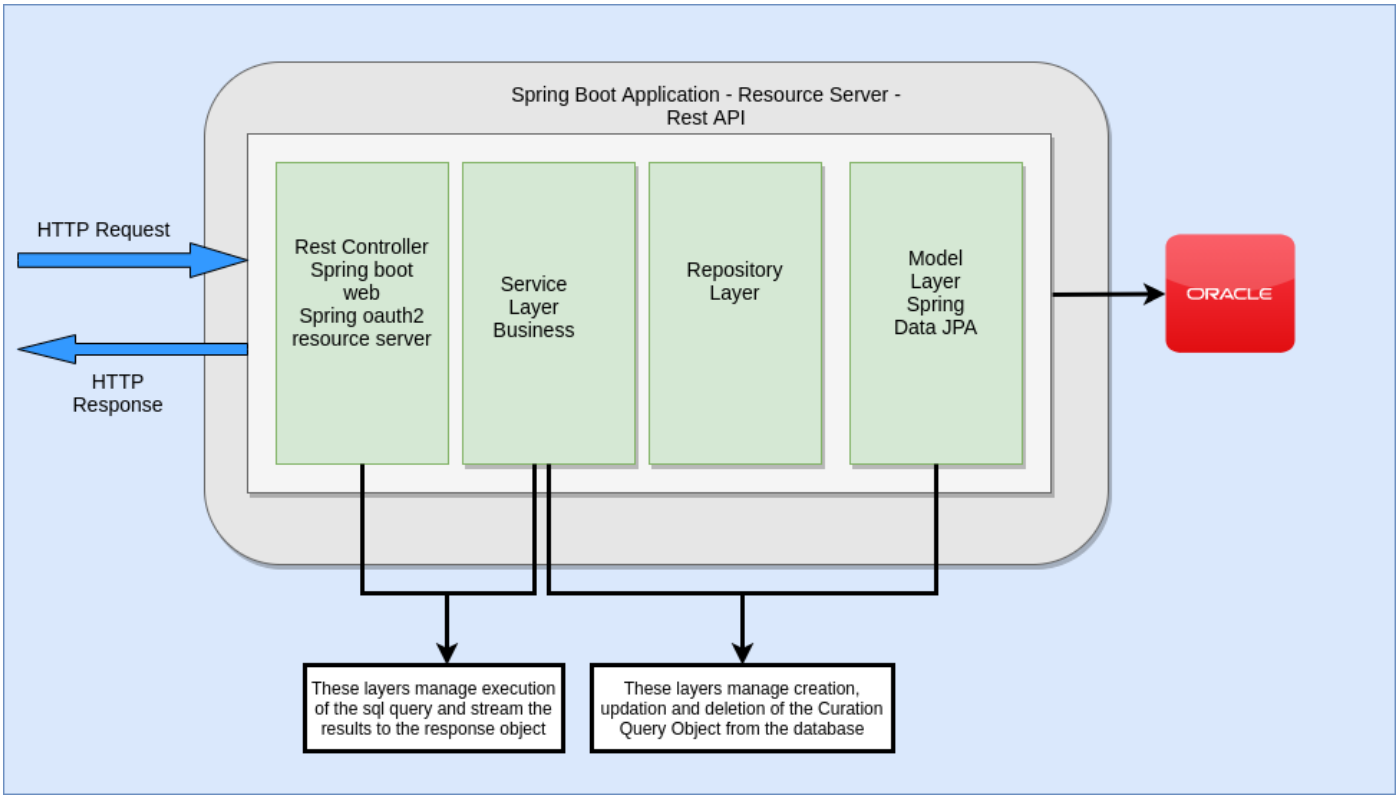
	Example : {
	"curationQueryId": 1150,
	"querySql": "select * from study"
	}
		the object should have a curationQueryId	

/query/execute/{queryId}	Example :
	Url :
	/query/execute//1198
	Parameters: {
	"study_accession":"SDY1"
	}
		Executes the sql query stored in the database for the specified query id with the parameters passed	POST

query/executeSql	Example:
	{
	"querySql": "select * from study where study_accession = 'SDY2'"
	}
		Executes the sql query passed as a parameter	POST

3.5.1.2. ImmPort Core Query API Software Architecture

The purpose of this API is to execute SQL queries dynamically and return tab delimited results back. The execute endpoint is called with a query id, the parameterized query is retrieved from the database and the parameters values passed in the POST body are replaced in the query string. The executeSql endpoint has the SQL query that needs to be executed in the body of the POST. The response to both endpoints are tab delimited results. The return columns specified in the query can be of different types and column type is determined by the ResultSetMetaData from the resultset when the query is executed. The fetch size is set to one so the results can be streamed to the response without an overhead on the memory on the server where the application is running.



3.5.2. Data Submission

The ImmPort data collection and sharing process is the product of extensive interaction, discussion, prototyping, and refinement with the DAIT POs, data providers and researchers who use the shared data. In order to encourage standardization of terms and vocabulary the ImmPort team developed a set of templates to capture, name, and define key elements of biomedical research data. The templates are informed by community standards where available. The ImmPort team engages with data standards communities such as the HIPC Standards Working Group, ISA Tools, CDISC, the minimal information standard groups, CEDAR, the Antibody Registry, and ontology developers to explore how to enhance the description of data captured in the templates. Templates are provided in a Microsoft Excel version to provide inline comments, validation, and ease-of-use features such as dropdown lists and color-coding of related data fields. The operational version of the templates is a simple tab separated value format that is widely used in bioinformatics.

Each template is fully documented consistent with the requirements of NIAID and the research community. Explanations include the purpose of the template, structure of the template (e.g. section, column, and row names), elements of the template required, whether numeric, preferred vocabulary or free text should be entered into a data field, how data elements are linked to each other across templates, and a glossary of terms and their reference sources. All templates, reference guides and example completed templates are version controlled and published when a software release is deployed.

Detailed information on data submission and templates are available at https://www.immport.org/resources/dataTemplates. Online interactive information for the templates is available at https://www.immport.org/shared/templateDocumentation.

3.5.2.1. Validation

A Batch Upload consists of a single Template file or group of Template file(s) and zero or more data file(s) provided in a zip-file or folder. Data files are either required by the template file(s) or designated as archive file(s). A Template file is a self formatted file consisting of a header segment, column specification and data column rows. Batch upload validation consists of validating all data in the set of templates and associated file before any data is uploaded into Oracle database and ImmPort file system. If any data in a Batch Upload fails validation no data is uploaded into Oracle database or file loaded into ImmPort file system by the the batch uploader. The set of all possible ImmPort templates have a specific validation and upload order. That is, data in one template must be loaded into the Oracle database prior to another template being uploaded since data in a latter template can depend upon data in a former template. All data within a Batch Upload must be valid to be uploaded into the Oracle database and ImmPort file system. Data in each template is uploaded into one or more Oracle database tables and files associated with the template are uploaded into the ImmPort file system. The data in a Batch Upload is valid if all the foreign keys for the set of Oracle tables to be loaded are satisfied and no duplicate rows are to be loaded into the set of Oracle database tables. Data columns within a template are mapped to one or more Oracle database table(s) columns. Several of these table columns are required to satisfy a controlled vocabulary. That is, the table column has a foreign key constraint to a Oracle database (lookup) controlled vocabulary. The Oracle tables have other foreign key constraints defined between the Oracle tables that provide parent-child relationships among the tables (for example, experiment_accession in EXPSAMPLE is related to experiment_accession in experiment).

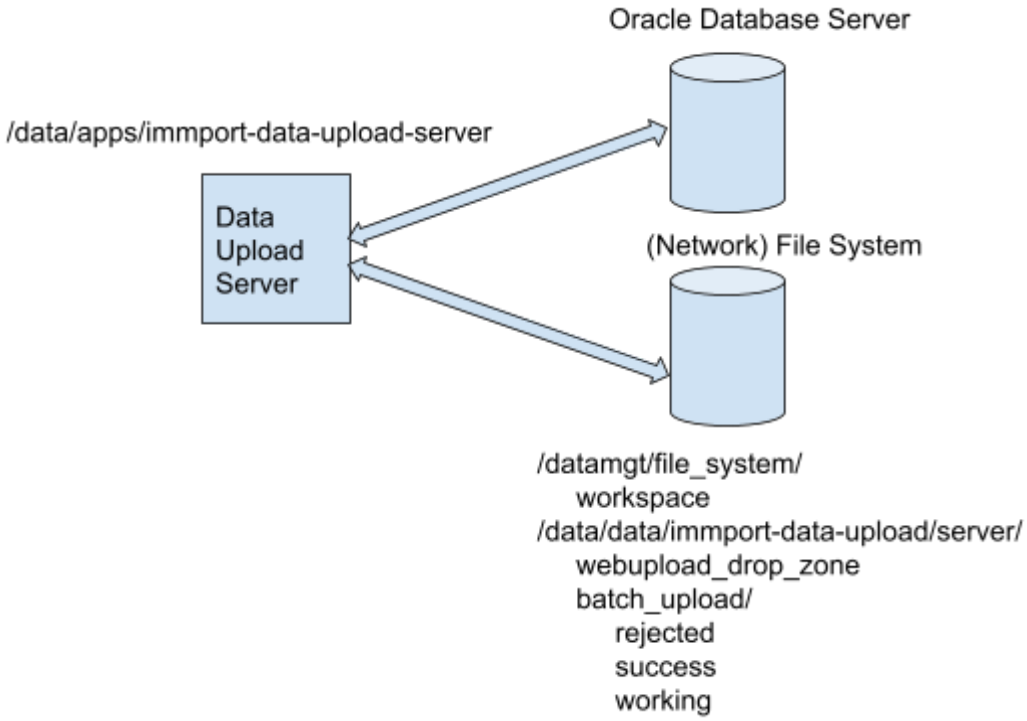
Validation requirements for each template are specified declaratively in XML. The XML specification provides the following: declaration of row uniqueness, the specification of generation of unique ID's for table rows, parent-child foreign key requirements, rules that must be satisfied on the template columns to be valid for upload, processes to process template columns into database columns and make further checks, and controlled vocabulary checks. The XML also specifies the mapping of template columns to database table(s) columns, and the specific validation queries that support parent-child foreign key requirements. The XML specification is currently implemented in the Batch Uploader Java software system.

A validation service, where no data is uploaded into the Oracle database or files are uploaded into ImmPort file system, is provided through the Data Manager and Batch Upload API. The Data Manager uses Batch Upload API to perform the validation.

3.5.2.2. Submission

Data Upload Submission is provided through the Data Manager and the Batch Uploader API. A Batch Upload Submission submits the file or folder to the upload zone and registers the upload job as pending in the Oracle database. The Batch Upload back-end server performs the validation as specified above and, if the upload is valid, it loads the data into the Oracle database and associated file(s) into ImmPort file system. The back-end Batch Upload server processes uploads through a cyclic basic (cron job). The submission process wakes every five (5) minutes and processes all pending Batch Uploads in submission timestamp order. This allows upload submissions to depend on one another.

The software architecture of the Batch Uploader server provided below.



3.5.3. Data Batch Update

The Data Batch Updater provides the mechanism for updating Oracle database tables (update, delete, and insert special linkages) after they have been uploaded by the Batch Uploader. The Data Batch Updater also allows for management of controlled vocabulary (lookup) tables (insert, update, and delete). A Data Batch Updater upload consists of a single formatted text-file that specifies an operation on a single Oracle database table that will perform either updates, insertions, or deletions. As with the Batch Uploader the operations on the Oracle database tables are specified declaratively in XML and implemented in the Data Batch Updater Java software system. The Batch Updater does not affect columns in the Oracle database that specify path information for associated files residing in the ImmPort file system. Also, the batch updater does not delete any row in an Oracle database table that contains ImmPort file system path information (See File System Management Update).

3.5.3.1. Validation

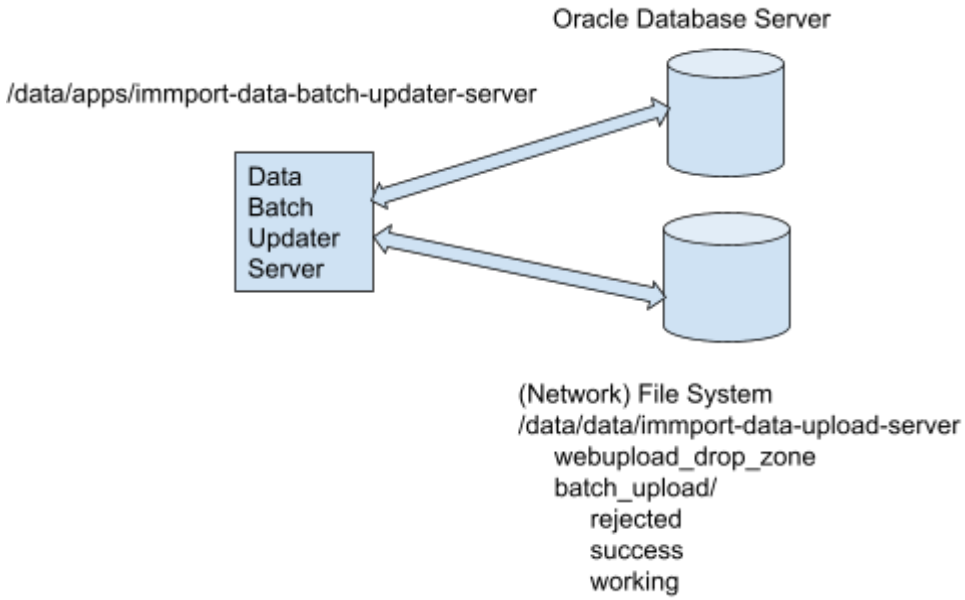
As with the Batch Uploader, the Batch Updater performs a validation phase on the batch updater file. These validations include: check duplicate rows by primary key or unique indices, check required columns, check foreign keys on columns of rows being inserted or updated, and in the case of deletion that there are no foreign key linkages involving the row being deleted with other Oracle database tables. If all validations are successful the updater file is uploadable. A validation service, where no data is uploaded or modified in the Oracle database, is provided through the Data Manager and Batch Updater API. The Data Manager uses Batch Updater API to perform the validation.

3.5.3.2. Submission

Batch Updater Submission is provided through the Data Manager or the Batch Updater API. The submission includes putting the updater file into the upload zone and registering a pending Batch Updater job in the Oracle database. A Data Batch Updater upload back-end server processes the pending Batch Updater jobs. The server validates the updater job and, if validated, processes it into an Oracle database. The back-end Batch Updater server manages batch

update jobs through a cyclic (cron) process. The server wakes every five (5) minutes and processes all the pending updater jobs in submission timestamp order. This allows updater submissions to depend on one another.

The software architecture of the Batch Updater server provided below.



3.5.4. File System Management Update

The File System Management Update Application manages updates to the Oracle database and ImmPort file system, keeping the two consistent between each other. The File System Management Update Application performs the following operations each identified as a single File System Management Update submission:

Remove Workspace

Remove Workspace operation removes the current content of Oracle database and the associated files in the ImmPort file system specific to the workspace and leaves the workspace empty to be operated on again. An option allows the workspace to be removed completely from the Oracle database and ImmPort file system. In the latter case the workspace no longer exists in the ImmPort file system.

Remove Upload Ticket Number

Remove Upload Ticket Number removes all Oracle database table content and associated files in the ImmPort file system related to the upload ticket. Only the recent completed upload ticket for a given workspace can be removed, after which is it marked as deleted.

Remove Study

Remove Study removes all the Oracle database table content and associated files in ImmPort file system related to a given study.

Remove File

For a given data file stored in the ImmPort file system (file info file (FILE_INFO.FILE_INFO_ID), protocol file (PROTOCOL.PROTOCOL_ACCESSION), study file (STUDY_FILE.STUDY_FILE_ACCESSION), or study image file (STUDY_IMAGE.SCHEMATIC_ACCESSION)), the file is removed from the ImmPort file system, and all linkages to the file in Oracle database for file info and protocol files. Finally, the row in the associated table is removed.

Remove Multiple Files

Remove multiple files requires a file comprising one line per ImmPort file system file to remove. Each line is comprised of file ID: file_info_id (FILE_INFO), protocol_accesion (PROTOCOL), study_file_accession (STUDY_FILE) , or schematic_accession (STUDY_IMAGE). All files must reside in the same workspace. The operation performs Remove File for each file ID.

Move Archive Files

For files that are designated as archived and stored in FILE_INFO (file detail ::= Archived) can be moved to a result file designation within a workspace. That is, the file can be associated with ASSESSMENT_PANEL (ASSESSMENT_2_FILE_INFO), EXPSAMPLE (EXPSAMPLE_2_FILE_INFO), CONTROL_SAMPLE (CONTROL_SAMPLE_2_FILE_INFO), or STANDARD_CURVE (STANDARD_CURVE_2_FILE_INFO). This operation requires a tab-separated file as input that contains one or more line with the format: (FILE_INFO_ID, ACCESSION, FILE_DETAIL). The FILE_DETAIL comes from the LK_FILE_DETAIL controlled vocabulary table. If a given FILE_INFO_ID repeats in the file, then the set of ACCESSIONS associated with it must belong to the same study, and the associated FILE_DETAIL must be the same. The ACCESSION must satisfy the accession format for the following accessions respectively: ASSESSMENT_PANEL_ACCESSION, EXPSAMPLE_ACCESSION, CONTROL_SAMPLE_ACCESSION, or STANDARD_CURVE_ACCESSION.

Transfer Files

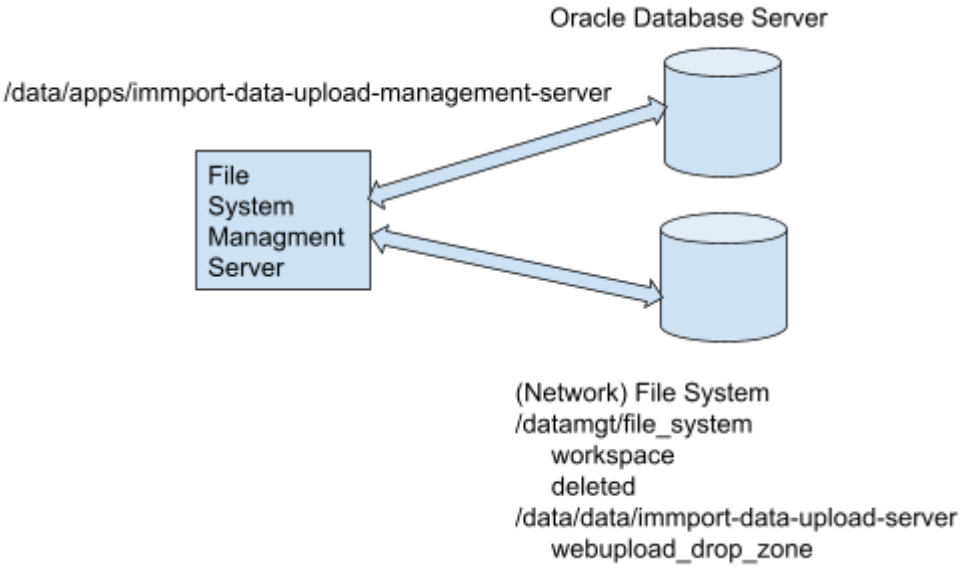
The (result) files can be transferred from one study to another within a workspace. The operation takes a tab-separated file where each line has the format: (FILE_ID, STUDY_ACCESSION). The FILE_ID can be one of the following FILE_INFO_ID (FiLE_INFO), STUDY_FILE_ACCESSION (STUDY_FILE), or STUDY_IMAGE (SCHEMATIC_ACCESSION). The STUDY_ACCESSION must differ from the study to which the FILE_ID is currently associated.

Assign CRF Files

This operation transfers study-based file system files in STUDY_FILE and assigns them as a CRF-file to an assessment panel in ASSESSMENT_2_FILE_INFO within the same study. That is, both the study file and assessment panel are in the same study. The operation requires a tab-separated file with the following format: (STUDY_FILE_ACCESSION, ASSESSMENT_PANEL_ACCESSION). All studies must reside within the same workspace.

File System Management Update validates a given operation. Validation includes determining the existence of the object(s) to be operated upon and the specific requirements of the operation. The File System Management Update submission is accessed through the Data Manager that defines the above operations as a single submission. The pending operation is stored in a scheduling table in the Oracle database and any associated file for the operation in the upload zone. The back-end File System Management Update server is a cron process that wakes every five (5) minutes and processes all the pending File System Management Update jobs in submission timestamp order. This allows operation submissions to depend on one another.

The software architecture of the File System Management Server is provided below

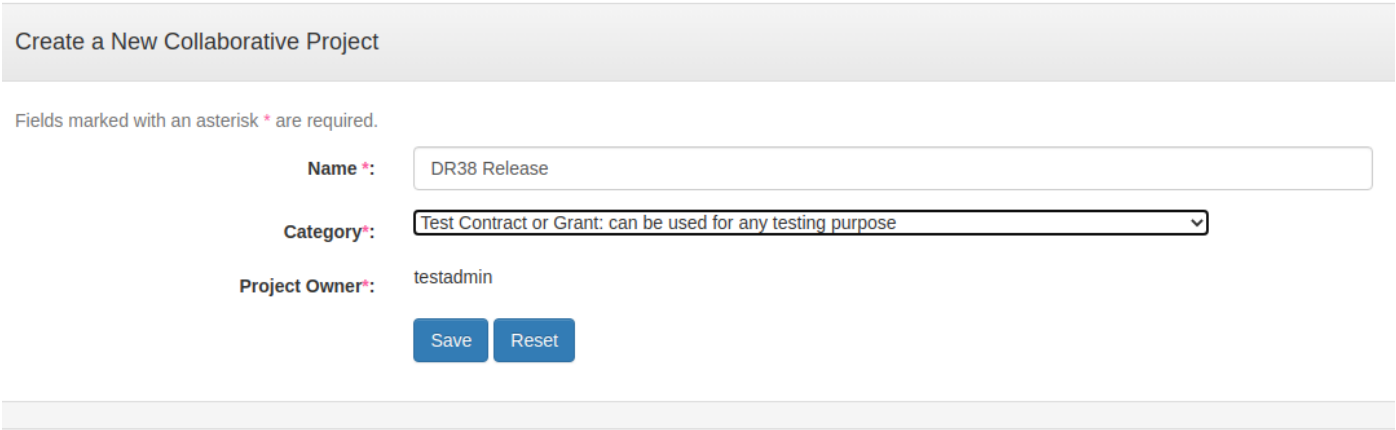


3.5.5. Data Manager

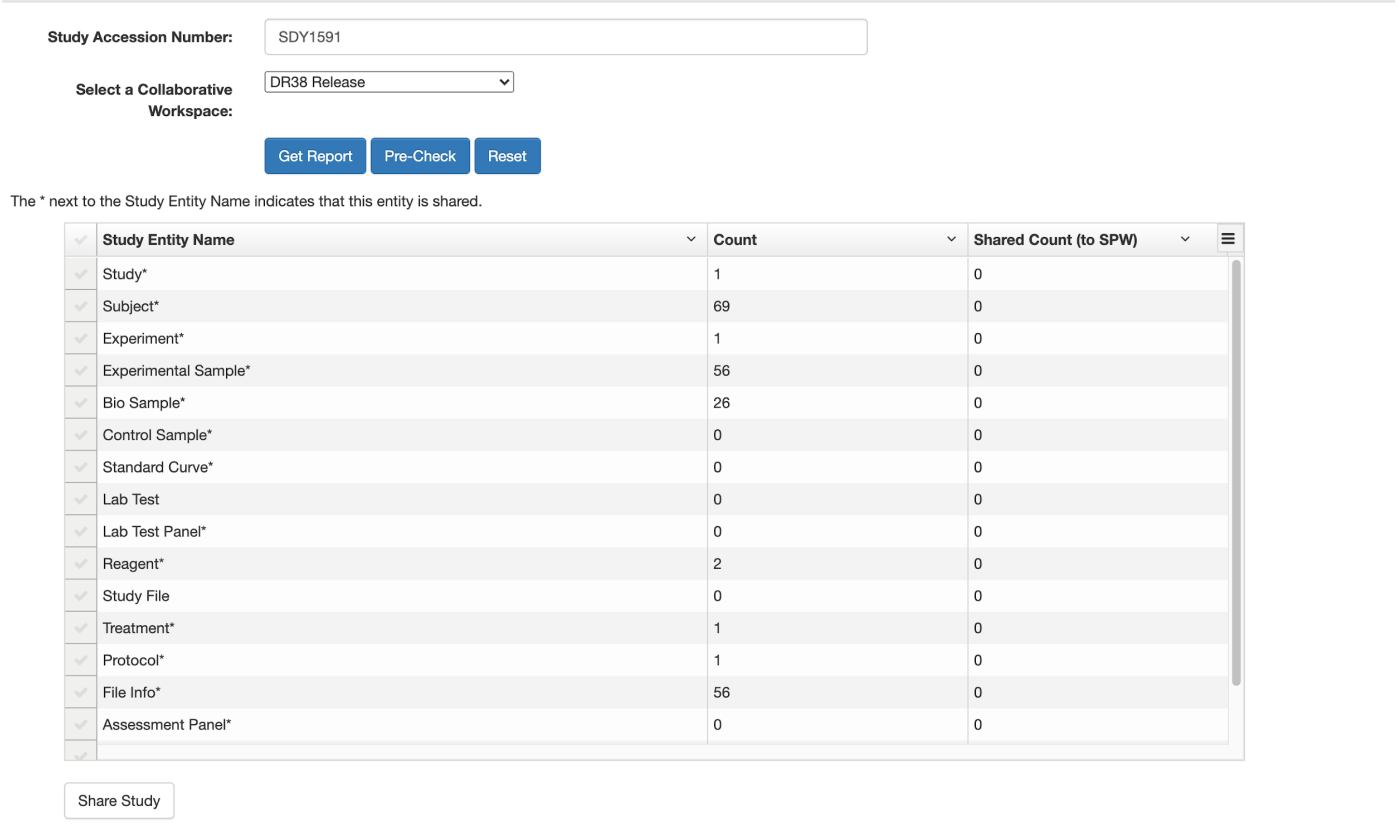
The Data Management Server hosts the web application that provides users with the interfaces to submit, query, and edit private research and clinical data to which they have access via a private workspace.

3.5.6. Sharing Tool

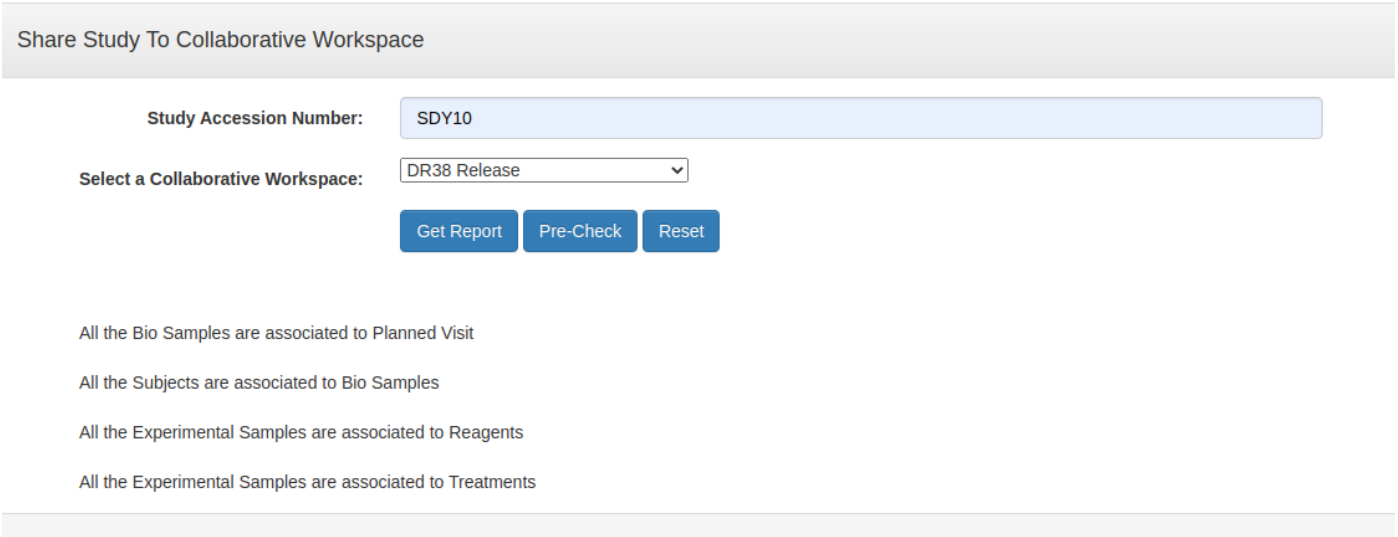
The Sharing tool hosts a web application designed to share study data from private workspaces to a collaborative workspace and then to a public workspace. This tool is used during the study data release process. When a set of studies is ready to be publicly shared or re-shared the following process is followed in the sharing tool.
A collaborative workspace is created by clicking on the Create Collaborative workspace menu.



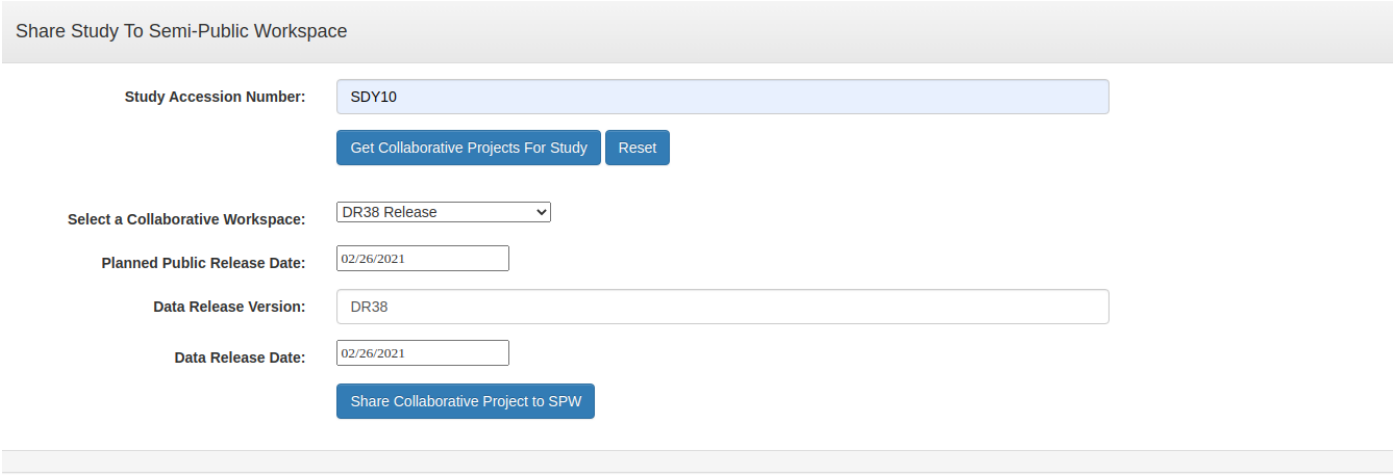
The study to be shared is first shared to the newly created Collaborative workspace by clicking on the Share Study button. Before you click the button you will get a report on the details of the study data that are to be shared. If the study is being re-shared the Shared Count column will have non-zero counts indicating how many experiments or lab tests or other study data have already been shared.



The Pre-Check button can be clicked to do some validations on the study data being shared.



Once the study data is shared to the Collaborative workspace, it can be shared to the Public workspace



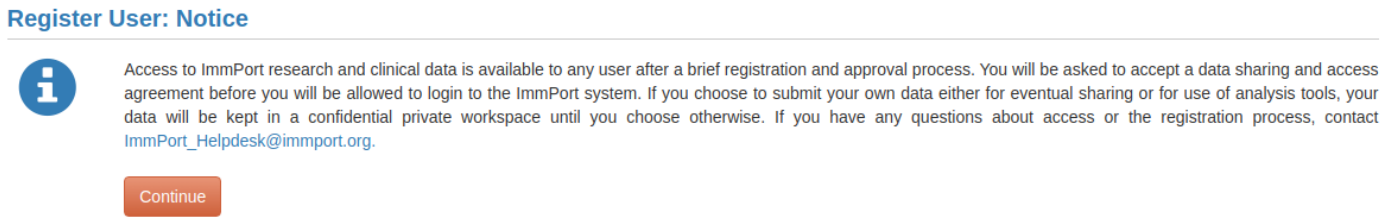
All study data shared to the public workspace is exported to the Aurora MySQL database on AWS.

3.5.7. User Administration

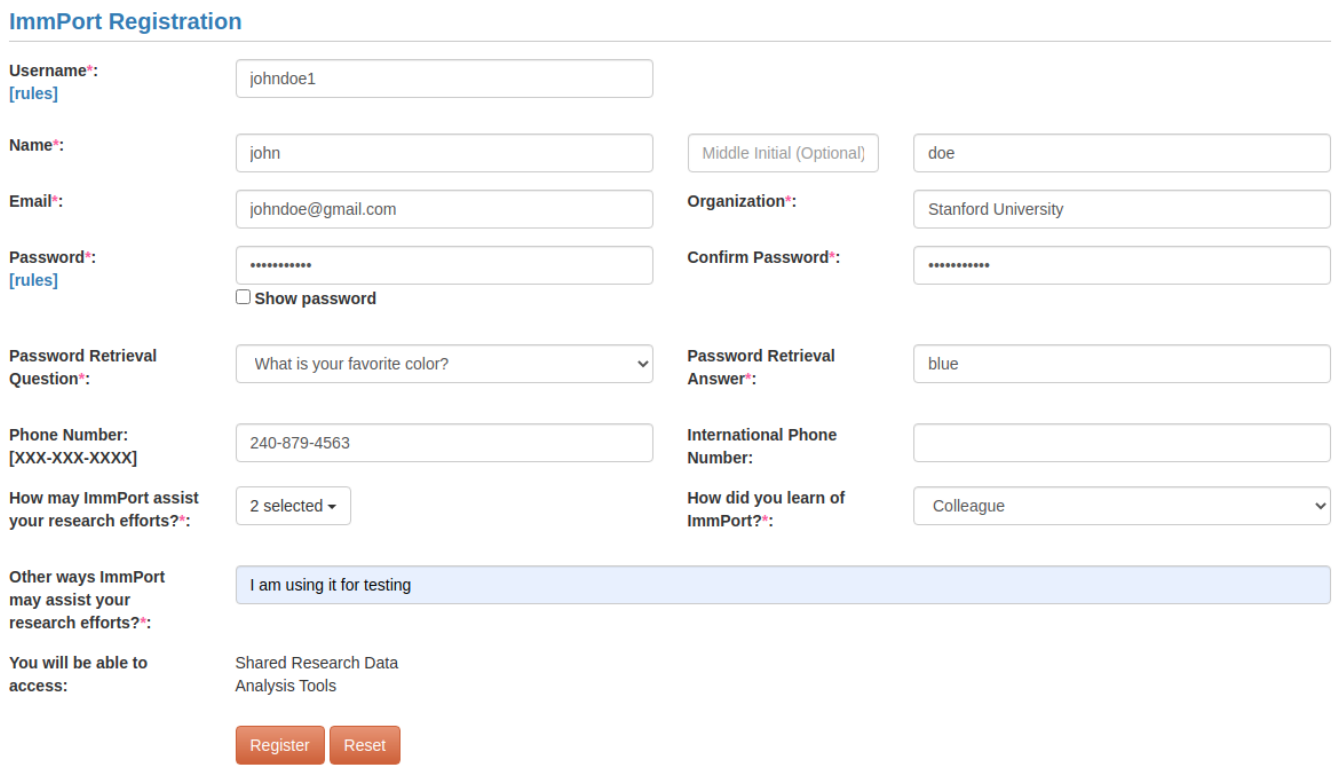
The User Administration Server hosts the web application designed to manage user registrations, accounts, and project access. The Manage User design artifacts model system administration capabilities which include allowing users to request system access (register), creating user accounts, updating user information, querying users, assigning groups and roles to users and deactivating users from the system. The User Administration Application is a Spring boot application with its frontend coded in AngularJS.

3.5.7.1. Registering User

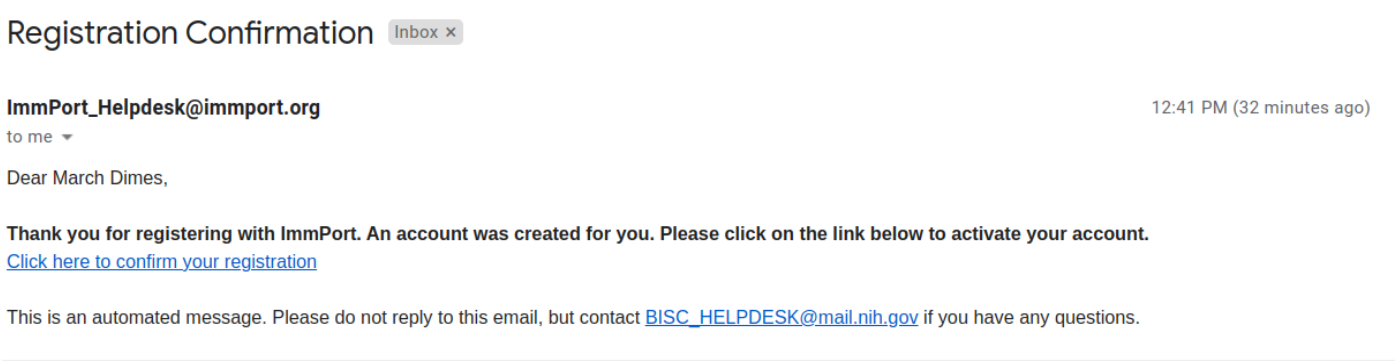
A user can register to gain access to ImmPort applications. By default the user is assigned the role “ROLE_USER”. One of the access rights this role gives the user is access to the data browser application to download studies. The individual initiates a registration request upon which the user sees the Notice



The system displays a page to gather information about the user. The username and email address are unique to the system so the user cannot add a username or email address that already exists in the system. The system validates the user information and saves the registration request, provided the data submitted are valid. Appropriate error messages are displayed if errors are encountered.

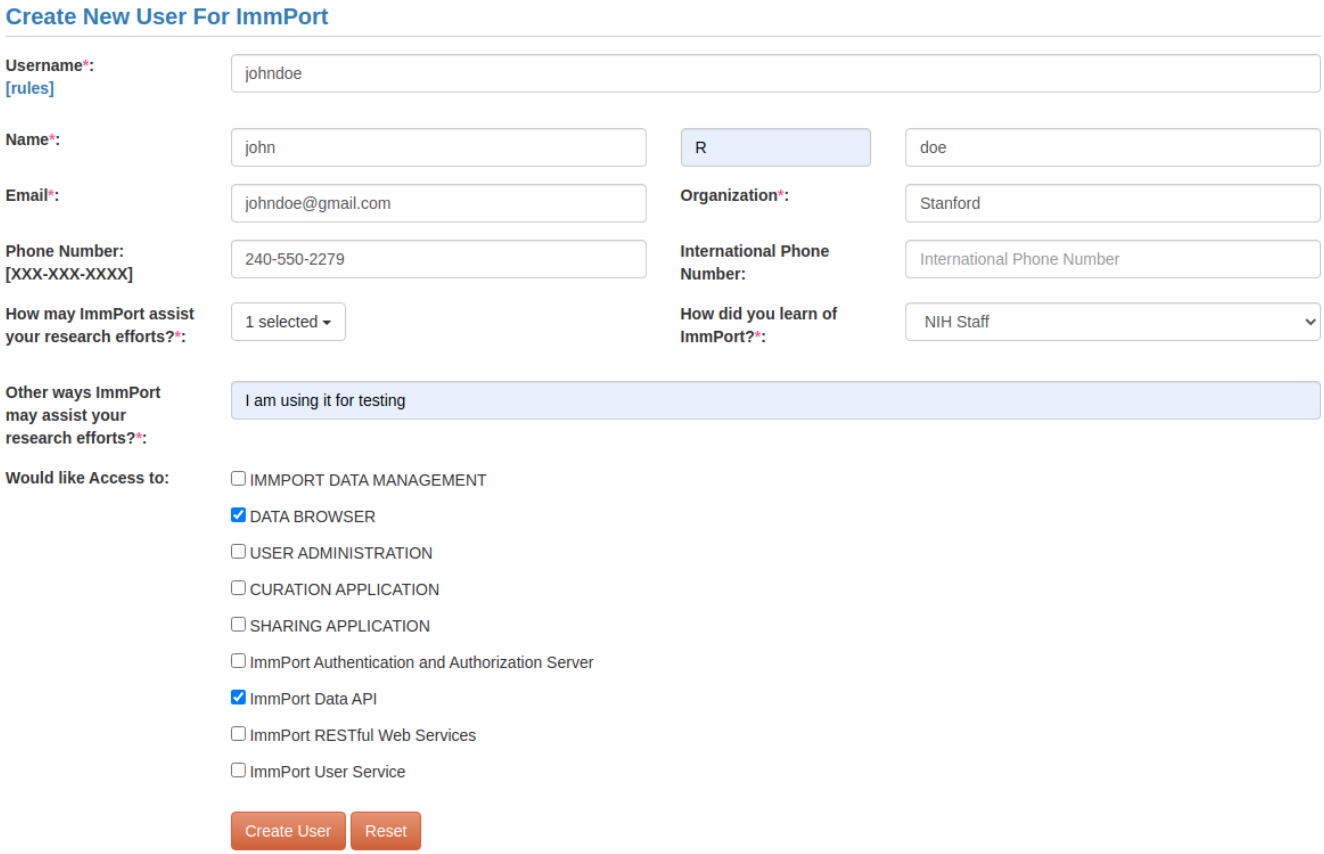


An email is sent to the user to confirm the email address provided. This email contains a link the user needs to click to activate the account.

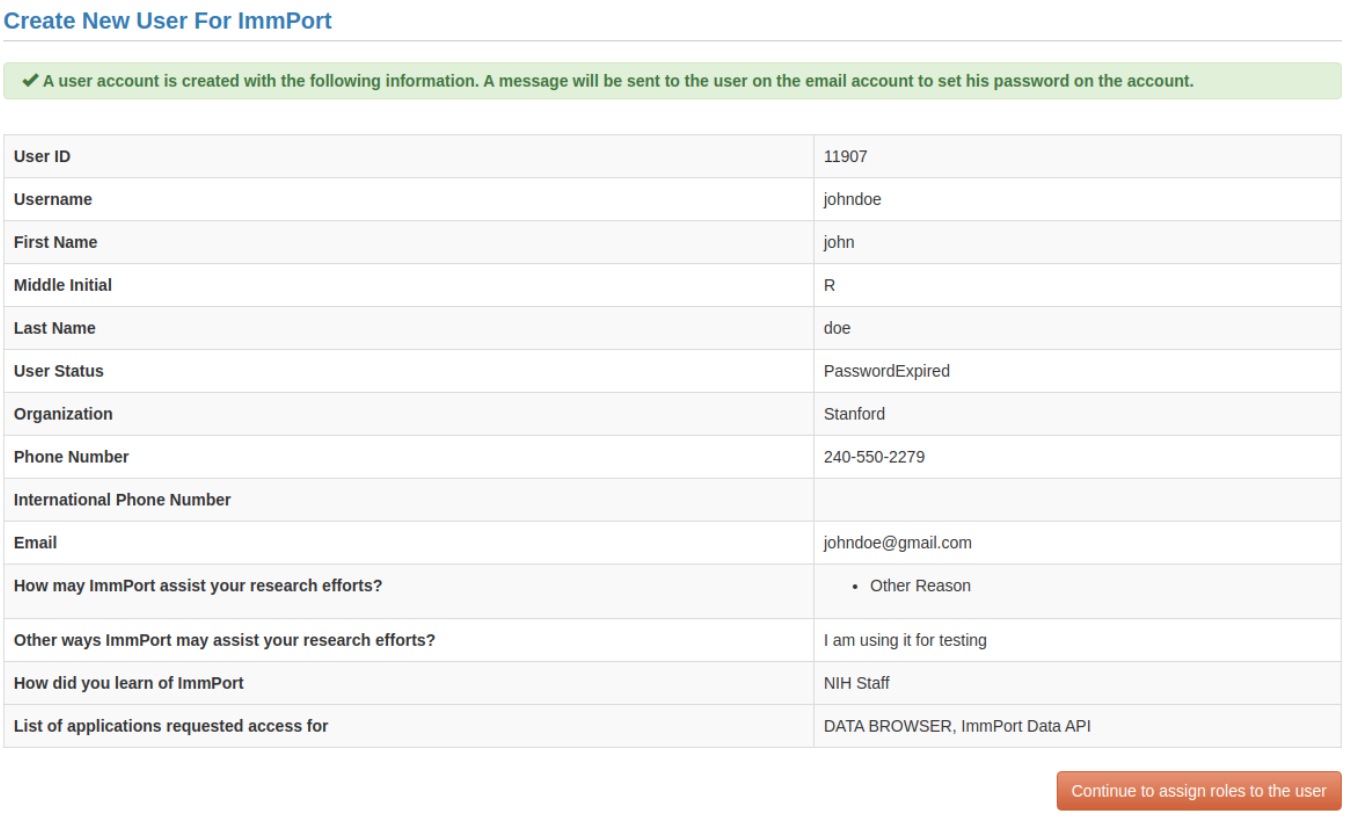


3.5.7.2. Creating an User by an Administrator

The ImmPort Administrator can create a user and assign appropriate access roles. The admin fills in the appropriate user information and clicks Create User



A user account is created and an email is sent to the user's email address to set his password. When the user sets his password , his account is activated.



Create New User For ImmPort

✔ A user account is created with the following information. A message will be sent to the user on the email account to set his password on the account.

User ID	11907
Username	johndoe
First Name	john
Middle Initial	R
Last Name	doe
User Status	PasswordExpired
Organization	Stanford
Phone Number	240-550-2279
International Phone Number	
Email	johndoe@gmail.com
How may ImmPort assist your research efforts?	• Other Reason
Other ways ImmPort may assist your research efforts?	I am using it for testing
How did you learn of ImmPort	NIH Staff
List of applications requested access for	DATA BROWSER, ImmPort Data API

Continue to assign roles to the user

The admin can continue to assign roles to the user. Since the admin selected Data Browser and Data API while creating the user a normal user role is assigned for these applications.

3.5.7.3. Search User

A user can be searched by the username,first name, last name and email address. A user once found, can be deactivated and his password can be reset. The admin can click on the Reset Password button and email will be sent to the user's email to reset his password.

3.5.7.4. Applications/Groups/Roles

A group is a means of organizing users whereas a role is usually a means of organising rights. Each role assigns some access rights for an application. Roles are assigned to groups and groups are assigned to applications. For example the User Service Rest API application has two roles REST_ADMIN and REST_USER. REST_ADMIN is assigned to the REST_ADMINS group and REST_USER is assigned to REST_USERS group. If a user mod2021 needs access to the User Service Rest API application as an admin the REST_ADMIN role needs to be assigned to the user. The user will need to be associated with the Application REST and to the group REST_ADMINS. This will give the user mod2021 access to the REST_ADMIN role.

Applications/Groups the user has access to:

Application Name	Group Names
DATA_BROWSER	
DATA-API	
REST	REST_USERS, REST_ADMINS

Edit Application Access | Edit Group Access

3.5.8. APIs

The Batch Uploader and Batch Updater APIs provide programmatic access to the operational capabilities of the Batch Uploader and Batch Updater. The API endpoints can be accessed directly by a user or by an application (for example, Data Manager). All requests to these APIs require authentication. The Core APIs use tokens for authentication. Users can obtain tokens by posting to the ImmPort Authentication URL- https://auth.immport.org/auth/token with a username and password. They must include the authentication token as an Authorization: bearer in the custom HTTP header.

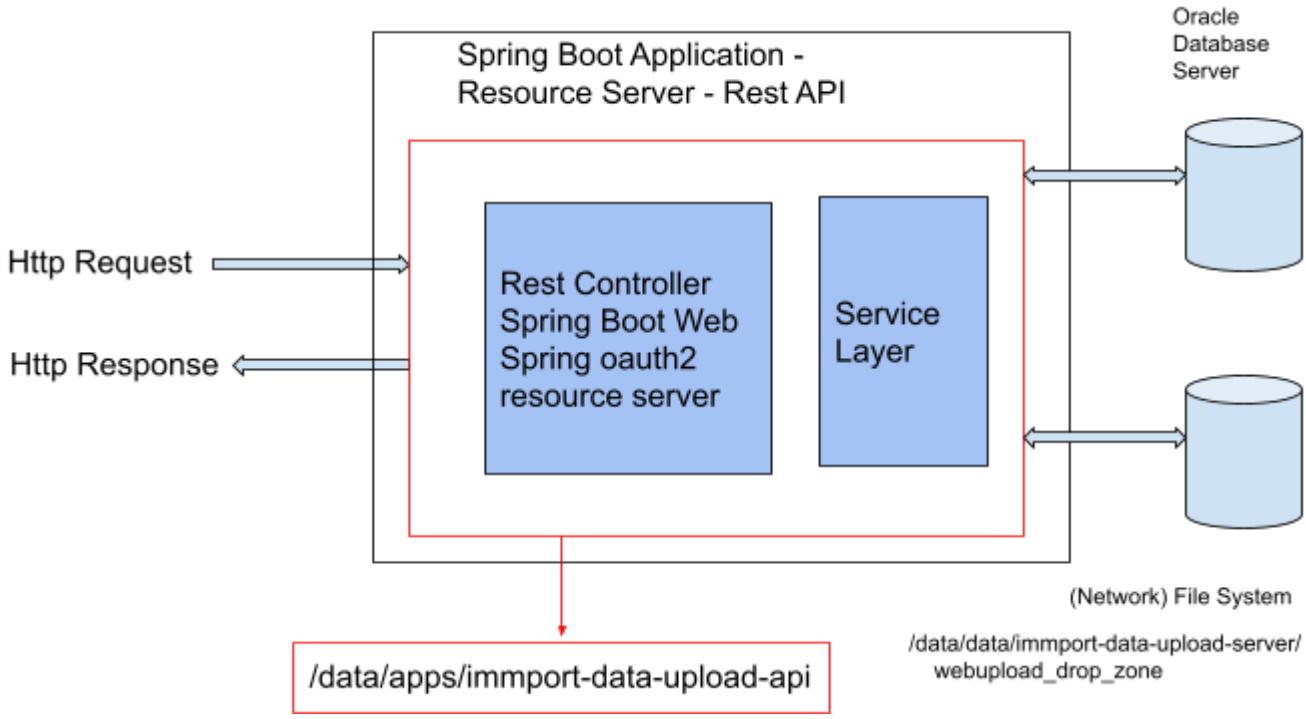
3.5.8.1. Batch Uploader API

The endpoints of the Batch Uploader API are listed below.

HTTP URL	Parameters (pass in the body of the request)	Description	GET/ POST
/data/upload/documentation/templates/WORKSPACE_ID		Documentation Generation: Generate documentation templates for a specific workspace (WORKSPACE_ID)	GET
/data/upload/type/offline	-F "workspaceId=WORKSPACE_ID" -F "packageName=PACKAGE_NAME" -F "uploadNotes=UPLOAD_NOTES" -F "uploadPurpose=uploadData" -F "serverName=SERVER_NAME"	OffLine File(s) Upload: Request for an off-line upload; creates upload registration in preparation for receipt of the file	POST
/data/upload/type/online	-F "workspaceId=WORKSPACE_ID" -F "packageName= -F "uploadNotes=UPLOAD_NOTES" -F "uploadPurpose=uploadData" -F "serverName=SERVER_NAME -F "file=@UPLOAD_ZIP_FILE_PATH_ON_CLIENT"	Zip-File Upload: Request upload of a zip-file; transfers file and creates upload registration and performs upload	POST
/data/upload/type/online	-F "workspaceId=WORKSPACE_ID" -F "packageName=" -F "uploadNotes=UPLOAD_NOTES" -F "uploadPurpose=uploadData" -F "serverName=SERVER_NAME" -F "file=@UPLOAD_ZIP_FILE_PATH_ON_CLIENT"	Zip-File Upload: Request upload of a zip-file; transfers file and creates upload registration and performs upload	POST
/data/upload/type/online	-F "workspaceId=WORKSPACE_ID" -F "packageName=PACKAGE_NAME" -F "uploadNotes=UPLOAD_NOTES" -F "uploadPurpose=uploadData" -F "serverName=SERVER_NAME -F "file=@UPLOAD_FILE_PATH_ON_CLIENT"	Multiple Files Upload (Single File): Request upload of a single file; transfers file and creates upload registration and performs upload; Note that single file is specified with the following -F parameter, -F "file=@UPLOAD_FILE_PATH_ON_CLIENT"	POST
/data/upload/type/online	-F "workspaceId=WORKSPACE_ID" -F "packageName=PACKAGE_NAME” -F "uploadNotes=UPLOAD_NOTES"	Multiple Files Upload (Multiple Files): Request upload of a several files; transfers files and creates upload registration and performs upload;	POST

	-F "uploadPurpose=uploadData" -F "serverName=SERVER_NAME" -F "file=@UPLOAD_FILE1_PATH_ON_CLIENT" -F "file=@UPLOAD_FILE_PATH_ON_CLIENT ...	Note that each file is specified with the following -F parameter, -F "file=@UPLOAD_FILE_PATH_ON_CLIENT"	
/data/upload/type/online	-F "workspaceId=WORKSPACE_ID" -F "packageName=" -F "uploadNotes=UPLOAD_NOTES" -F "uploadPurpose=uploadData" -F "serverName=SERVER_NAME -F "file=@UPLOAD_ZIP_FILE_PATH_ON_CLIENT"	Zip-file Upload for Validation: Zip-file validation is a two step process where the zip-file is uploaded to the server and the upload registration generated and then the validation is requested (see Validation of a File)	POST
/data/upload/type/online	-F "workspaceId=WORKSPACE_ID" -F "packageName=PACKAGE_NAME" -F "uploadNotes=UPLOAD_NOTES" -F "uploadPurpose=validateData -F "serverName=SERVER_NAME" -F "file=@UPLOAD_FILE1_PATH_ON_CLIENT -F "file=@UPLOAD_FILE_PATH_ON_CLIENT" ...	Multiple File Upload for Validation (Multiple Files): Request upload of several files not as a zip-file package; Note that each file is specified with the following -F parameter, -F "file=@UPLOAD_FILE_PATH_ON_CLIENT"	POST
/data/upload/validation	-F "uploadTicketNumber=UPLOAD_TICKET_NUMBER”	Validation of Upload Ticket: Validation of job that is identified by the upload ticket number; Note this endpoint uses the -F parameter, -F "uploadTicketNumber=UPLOAD_TICKET_NUMBER"	POST
/data/upload/registration/UPLOAD_TICKET_NUMBER/status		Status of Upload Ticket: Return the current status of an upload ticket (UPLOAD_TICKET_NUMBER)	GET
/data/upload/registration/UPLOAD_TICKET_NUMBER/reports/summary		Summary Information on Upload Ticket: On completed jobs (either Completed or Rejected), provide the information on the upload ticket (UPLOAD_TICKET_NUMBER)	GET
/data/upload/registration/UPLOAD_TICKET_NUMBER/reports/database		Database Information on Upload Ticket: On completed jobs (Completed only) provide database information (UPLOAD_TICKET_NUMBER)	GET
/data/upload/registration/UPLOAD_TICKET_NUMBER/reports/database		Database Information on Upload Ticket: On completed jobs (Completed only) provide database information (UPLOAD_TICKET_NUMBER)	GET

The software architecture for the Batch Upload API is similar to the ImmPort Core Query API



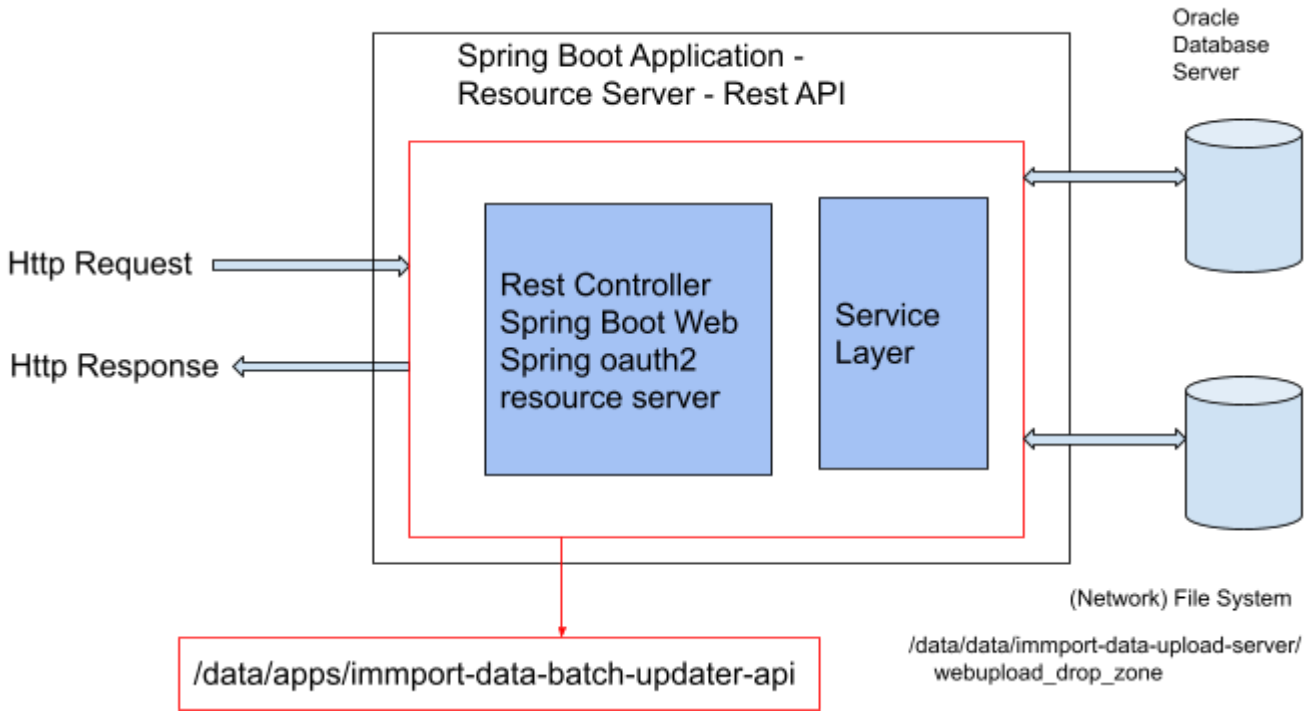
3.5.8.2. Batch Updater API

The endpoints of the Batch Updater API are listed below.

HTTP URL	Parameters (pass in the body of the request)	Description	GET/ POST
/data/batch/updater/documentation/templates		Documentation Generation: Generate documentation templates	GET
/data/batch/updater	-F "workspaceId=WORKSPACE_ID" -F "packageName=" -F "uploadNotes=UPLOAD_NOTES" -F "uploadPurpose=batchUpdateUpload -F "serverName=SERVER_NAME -F "file=@UPLOAD_BATCH_UPDATER_FILE_PATH_ON_CLIENT"	Batch Update Upload: Request upload of a zip-file; transfers file and creates upload registration and performs batch update requested	POST
/data/batch/updater	-F "workspaceId=WORKSPACE_ID" -F "packageName=" -F "uploadNotes=UPLOAD_NOTES" -F "uploadPurpose=batchUpdateValidate" -F "serverName=SERVER_NAME" -F "file=@UPLOAD_BATCH_UPDATER_FILE_PATH_ON_CLIENT"	Batch Update Upload for Validation: Batch update validation is a two step process where the batch update file is uploaded to the server and the upload registration generated (this endpoint), and then the validation is requested (see Validation of Upload Ticket endpoint)	POST
/data/batch/updater/validation	-F "uploadTicketNumber=UPLOAD_TICKET_NUMBER"	Validation of Upload Ticket: Validation a batch updater file that is identified by the upload ticket number; Note this endpoint uses the -F parameter, -F "uploadTicketNumber=UPLOAD_TICKET_NUMBER"	POST
/data/batch/updater/registration/UPLOAD_TICKET_NUMBER/status		Status of Upload Ticket: Return the current status of an upload	GET

		ticket (UPLOAD_TICKET_NUMBER)	
/data/batch/updater/registration/UPLOAD_TICKET_NUMBER/summary		Summary Information on Upload Ticket: On completed jobs (either Completed or Rejected), provide the information on the upload ticket (UPLOAD_TICKET_NUMBER)	GET
/data/batch/updater/registration/UPLOAD_TICKET_NUMBER/database		Database Information on Upload Ticket: On completed jobs (Completed only) provide database information (UPLOAD_TICKET_NUMBER)	GET
/workspaces		Set of Workspaces: Return the set of workspace(s) on which a user can perform and upload or validation	GET

The software architecture for the Batch Upload API is similar to the ImmPort Core Query API



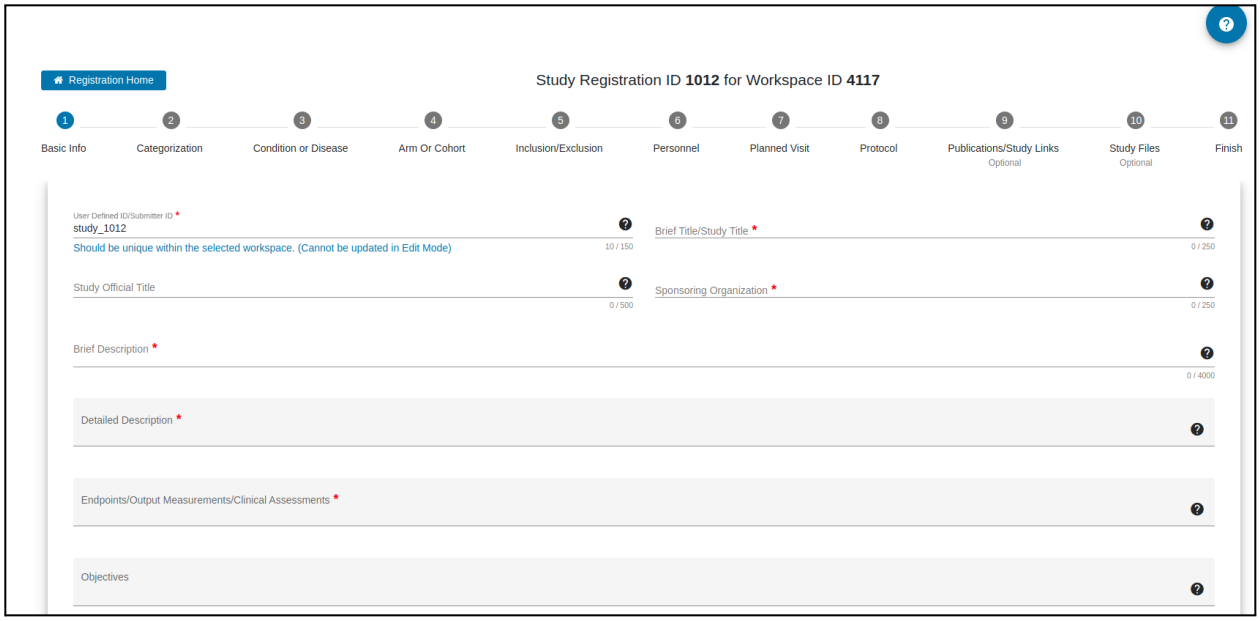
3.5.9. ImmPort Study Registration Wizard

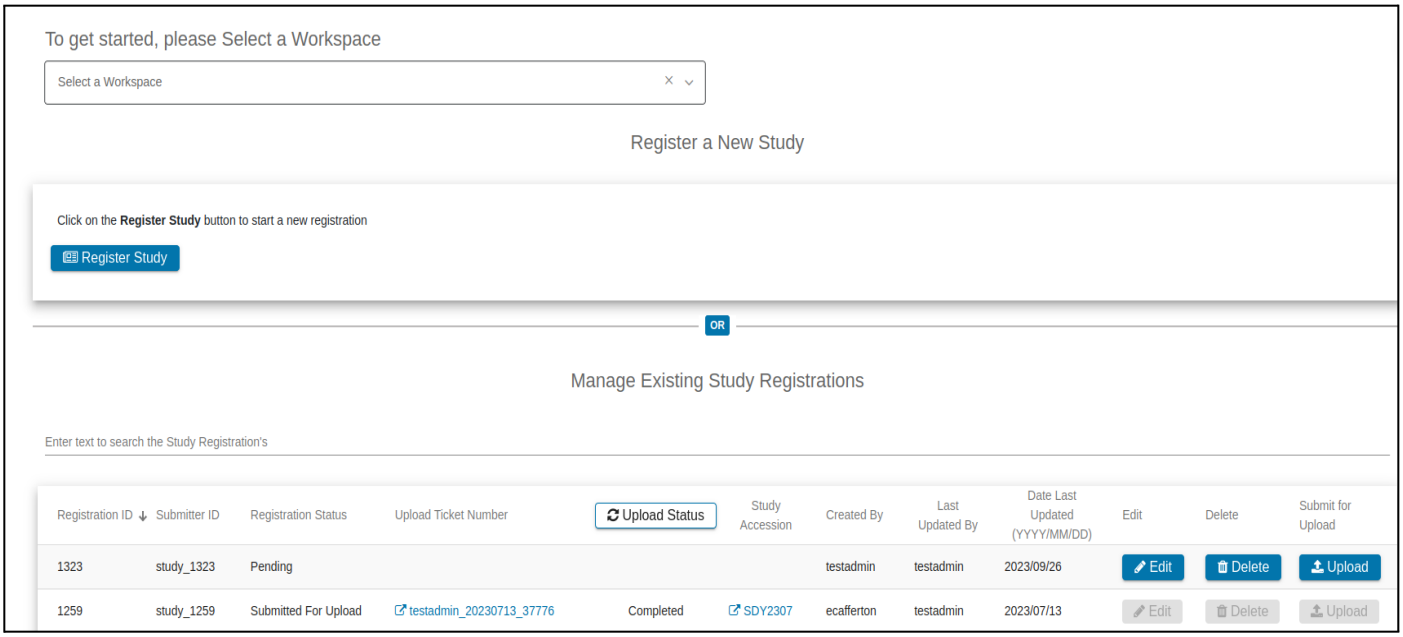
To further streamline the data submission process, and address requests from our data providers to simplify data uploads, an easy-to-use UI Wizard is under development.. The Study Registration Wizard (SRW) is an interactive and intuitive tool for entering study data, validating data in real time and mapping terms reported to the standard ontology terms. The SRW accelerates the process of data submission and motivates data providers to submit their studies to the ImmPort repository in a timely manner. It also aids the process of storing their study data in the ImmPort repository and the generation of an ImmPort Study Accession that can be reported in publications. Additionally, this tool

empowers end users to load their research data with little to no assistance from the ImmPort staff, enhancing the scalability of uploads.

The figure below shows the data elements of a basic study design template as steps. The wizard takes the user through the different steps required to register a study with ImmPort. Steps 1 to 8 are required data elements and step 9 are optional data elements. Step 10 leads you to the page where you save the data or validate and submit for upload.

For additional assistance with uploads, SRW tutorials and documentation were integrated into the new documentation site. A current view of the Study Registration Interface can be seen in the below figure





In the June 2022 software release the following functionalities were developed and deployed:

- Text box where users can enter their protocol information
- Support XLS and TSV options to upload row-level data for Planned visits, Inclusion/Exclusion, Arm/Cohort.
- Updated definitions for fields
- Add interactive feedback buttons to collect user feedback