Biocomputeobject.org OSF page: https://osf.io/h59uh/

## 

BioCompute (BCO) specification document

Release 1 (Standard Trial Use)

[DRAFT]

EDITING IS FROZEN. YOU CAN STILL COMMENT

###############################################################################

Release 1 (STU)

For

September 2017 Release

High-throughput Sequencing Computational Standards for Regulatory Sciences (HTS-CSRS) Project

Contact: Raja Mazumder ([mazumder@gwu.edu](mailto:mazumder@gwu.edu)) and Vahan Simonyan ([vahan.simonyan@fda.hhs.gov](mailto:vahan.simonyan@fda.hhs.gov))

BioCompute Object Consortium members: Vahan Simonyan (FDA), Raja Mazumder (GW), Charles Hadley King (GW), Amanda Bell (GW), Jeet Vora (GW), Jianchao Yao (Merck & Co), Jeremy Goecks (Oregon Health and Science University), Paul Walsh (NSilico Life Science) Jonas S Almeida (Stony Brook Univ, SUNY), Mark Walderhaug (FDA), Elaine Thompson (FDA), Alin Voskanian (FDA), Anton Golikov (FDA), Konstantinos Karagiannis (FDA), Dennis A. Dean, II (Seven Bridges), Stian Soiland-Reyes (UNIMAN), Carole Goble (UNIMAN), Michael R. Crusoe (CWL) …

# Table Of Contents:

[**1 Introduction**](#_4d34og8) **6**

[1.1 Mission of the BioCompute project](#_2s8eyo1) 6

[1.2 Motivation](#_17dp8vu) 6

[1.2.1 Limitations of initial Effort](#_3rdcrjn) 7

[1.3 Audience for this document](#_26in1rg) 7

[1.4 Potential Stakeholders for the entire BioCompute project](#_lnxbz9) 7

[1.5 User stories](#_35nkun2) 7

[**2 Data type for BCOs**](#_44sinio) **8**

[2.1 Identification and Provenance fields](#_2jxsxqh) 9

[2.1.1 ID "id"](#_z337ya) 9

[2.1.2 Name "name"](#_3j2qqm3) 9

[2.1.3 Structured name "structured\_name"](#_1y810tw) 9

[2.1.4 Version "version"](#_4i7ojhp) 9

[2.1.5 Digital signature "digital\_signature"](#_2xcytpi) 10

[2.1.6 Verification status "verification\_status"](#_1ci93xb) 10

[2.1.7 Publication status "publication\_status"](#_3whwml4) 10

[2.1.8 Authors "authors"](#_2bn6wsx) 10

[2.1.8.1 Name "name"](#_qsh70q) 11

[2.1.8.2 ORCID "orcid"](#_3as4poj) 11

[2.2 Usability domain "usability\_domain"](#_1pxezwc) 11

[2.3 Description Domain "description\_domain"](#_49x2ik5) 11

[2.3.1 Keywords "keywords"](#_2p2csry) 11

[2.3.2 External References "xref"](#_147n2zr) 12

[2.3.3.1 Tool Name "tool\_name"](#_ihv636) 13

[2.3.3.2 Tool Description "tool\_desc"](#_32hioqz) 13

[2.3.3.3 Tool Version "tool\_version"](#_1hmsyys) 13

[2.3.3.4 Tool Requirements or Packages "tool\_package"](#_41mghml) 13

[2.3.3.5 Step Number "step\_number"](#_2grqrue) 13

[2.3.3.6 Input URI List "input\_uri\_list"](#_vx1227) 13

[2.3.3.7 Output URI List "output\_uri\_list"](#_3fwokq0) 13

[2.4 Execution domain "execution\_domain"](#_1v1yuxt) 14

[2.4.1 Script Type "script\_type"](#_4f1mdlm) 14

[2.4.2 Script "script"](#_19c6y18) 14

[2.4.3 Pipeline Version "pipeline\_version"](#_3tbugp1) 14

[2.4.4 Platform/Environment "platform"](#_28h4qwu) 14

[2.4.5 Script driver "script\_driver"](#_nmf14n) 15

[2.4.6 Algorithmic tools and Software Prerequisites "software\_prerequisites"](#_37m2jsg) 15

[2.4.7 Domain Permissions "domain\_prerequisites"](#_1mrcu09) 15

[2.4.7 Environmental parameters "env\_parameters"](#_46r0co2) 15

[2.5 Parametric domain "parametric\_domain"](#_2lwamvv) 16

[2.6 Input and output domains "io\_domain"](#_111kx3o) 16

[2.6.1 Input Subdomain "input\_subdomain"](#_206ipza) 16

[2.6.2 Output Subdomain "output\_uri\_list"](#_4k668n3) 16

[2.7 Error domain, acceptable range of variability "error\_domain"](#_2zbgiuw) 17

[**3 Appendices**](#_1egqt2p) **17**

[3.1 Appendix-I: BCO expanded view example](#_3ygebqi) 17

[3.2 Appendix-II: External reference database list](#_2dlolyb) 20

[3.3 Appendix III - FDA draft guidance documents and Title 21 CFR Part 11](#_3cqmetx) 22

[3.3.1 FDA Draft Guidance Documents.](#_1rvwp1q) 22

[3.3.2 Title 21 CFR Part 11](#_4bvk7pj) 23

[3.4 Appendix IV - Compatibility](#_2r0uhxc) 23

[3.4.1 ISA for the experimental metadata](#_1664s55) 23

[3.5 Appendix V - Security](https://docs.google.com/document/d/1kcCrsUGWqhMk2XobZRlgEN0OKASSrbaYHo8XZrQ6CV0/edit#heading=h.3q5sasy) [24](https://docs.google.com/document/d/1kcCrsUGWqhMk2XobZRlgEN0OKASSrbaYHo8XZrQ6CV0/edit#heading=h.3q5sasy)

[3.5.1 Blockchain](https://docs.google.com/document/d/1kcCrsUGWqhMk2XobZRlgEN0OKASSrbaYHo8XZrQ6CV0/edit#heading=h.25b2l0r) [24](https://docs.google.com/document/d/1kcCrsUGWqhMk2XobZRlgEN0OKASSrbaYHo8XZrQ6CV0/edit#heading=h.25b2l0r)

[3.6 Appendix VI Data typing](#_kgcv8k) 24

[3.6.1 Primitive data types](#_34g0dwd) 24

[3.6.2 Base BioCompute Type](#_1jlao46) 26

[3.6.3 Meaning associations](#_43ky6rz) 30

[3.6.4 Atomic and complex data](#_2iq8gzs) 30

[3.6.5 Fundamental and derivable types](#_xvir7l) 31

[3.6.5 Extensibility through inheritance and inclusion of data types](#_39kk8xu) 34

[3.6.6 Data lifecycle timeline](#_1opuj5n) 35

# 

# 1 Introduction

BioCompute Objects will facilitate HTS (NGS) computational analysis information communication between the FDA and industry and academic research stakeholders. BioCompute is a paradigm and BioCompute Object (BCO) is an instance.

The US Food and Drug Administration (FDA) and George Washington University (GW) have partnered to establish a framework for community-based standards development and harmonization of High-throughput Sequencing (HTS) computations and data formats. Standardized HTS computations and data formats will promote interoperability and ease the verification of bioinformatics protocols. To do this, a schema has been developed to represent instances of computational analysis as a BCO. A BCO includes: information about parameters and version of the executable programs in a pipeline, reference to input and output test data for verification of the pipeline, a usability domain, keywords, a list of authors along with other important metadata. As much as possible this description should be programmatically interoperable to public Cloud infrastructure.

Additional, non-normative, information on BCOs:

[https://hive.biochemistry.gwu.edu/htscsrs/BioCompute](https://hive.biochemistry.gwu.edu/htscsrs/biocompute)

<https://hive.biochemistry.gwu.edu/htscsrs/datatyping>

## 1.1 Mission of the BioCompute project

* BioCompute Objects will facilitate HTS (NGS) computational analysis information communication with FDA
* Develop a community of stakeholders to create a versatile data harmonization framework that allows the standardized definition of interoperable bioinformatics pipelines in a platform independent manner.
* Development of tools and facilities implementing data-typing, instantiation, deposition, storage, and distribution of validated BioCompute Objects through BioBompute database, in order to facilitate reproducible scientific research and regulatory submissions of data and computations.
* Facilitate portability of pipelines for execution on Public Cloud infrastructure.

## 1.2 Motivation

The unpredictability of actual physical, chemical, and biological experiments due to the multitude of environmental and procedural factors are well documented. Usually what is systematically overlooked is that computational biology algorithms are also affected by a multiplicity of parameters and have no lesser volatility. The complexities of computation protocols and interpretation of outcomes is only a part of the challenge: There are also virtually no standardized and industry-accepted metadata schemas for reporting the computational objects that record the parameters used for computations together with their results. Thus, it is often impossible to reproduce the results of a previously performed computation due to missing information on parameters, versions, arguments, conditions, and procedures of application launch. BioCompute Object concept has been developed specifically to satisfy regulatory research needs for evaluation, validation, and verification of bioinformatics pipelines. Potential usability of BioCompute Objects within the larger scientific community can be increased through the creation of a BioCompute Object database initially consisting of records relevant to the U.S. Food and Drug Administration. A BioCompute Object database record will be similar to a GenBank record in form; the difference being that instead of describing a sequence, the BioCompute record will include information related to parameters, dependencies, usage, and other information related to the specific computational instance. This mechanism will extend similar efforts and also serve as a collaborative ground to ensure interoperability between different platforms, industries, scientists, regulators, and other stakeholders interested in biocomputing.

For more information, see the project description on the [**FDA Extramural Research**](http://www.fda.gov/ScienceResearch/SpecialTopics/RegulatoryScience/ucm491893.htm) page.

### 1.2.1 Limitations of initial Effort

* At the initial stages of BioCompute development, we address the challenges of HTS (NGS) bioinformatics.
* BCOs could very easily be extended to other types of computational analysis, but at this stage, we are limiting our focus to HTS/NGS only.

## 1.3 Audience for this document

* Users performing HTS (NGS) analysis with a regulatory science perspective
* HTS (NGS) bioinformatics platform developers
* HTS (NGS) related standard developers

## 1.4 Potential Stakeholders for the entire BioCompute project

* US Food and Drug Administration
* Medical product manufacturers and their suppliers
* Laboratories developing clinical testing protocols.
* Bioinformatics tool and platform developers who wish to operate in a regulatory environment, including cloud service (PaaS, IaaS, SaaS, FaaS) providers.
* Journals / Scientific Publishing / peer reviewing process
* NIH (particularly initiatives such as NCI/ITCR)
* Public cloud companies operating in the Life Sciences sector including electronic health record systems (EHR)

## 1.5 User stories

* A pharmaceutical company is submitting NGS data and the FDA conducts a reanalysis of the data. The reanalysis does not concur with the original results. It can be very lengthy and costly to figure out where the discrepancies are. [reproducibility and interpretation use cases]  
    
  Attaching a BioCompute Object with the initial submission would prevent most of the ambiguity surrounding the discrepancies.
* A regulatory decision has been made where computational analysis has been used as evidence. New data emerges after the product has been on the market over a year and the regulators can not reproduce the original environment with all unspecified configuration of tools and parameters of pipelines to reanalyze the initial submission data or replicate the initial conclusion. The product is on the market already. [recomputability use case]
* Authors and pharmaceutical scientists are unaware of how the regulatory industry is using workflows to analyze data. Openness and transparency are hindered by the lack of ability to communicate, not a lack of willingness. Scientific merit is compromised as a result of not having a common "language" for communicating computations. [collaboration use case]
* A bioinformatics platform provider can use BCO as part of its verification and validation process. A customer submits NGS data provided by a third party sequencing provider. The sequencing data is poor quality. Reproducible pipelines, validated and verified as a “BCO”, were used to demonstrate the fault lies in the sequencing step and not the bioinformatics pipeline.
* One potential use case related to this is one of 'differential impact' of how different choices in the workflow affect the outcome of the computational analysis/experiment (e.g. changing expression estimation procedure). "different choices" of workflows are different instances and hence will be different BCOs and you are absolutely right that it will change the parametric domain and output domain.
* BCOs can serve as a history of what was computed. An example pertaining to provenance, from experience: data are generated and QC'ed as far as possible, and then passed on for analysis. The analysis diagnoses a problem with one or more samples (e.g., cryptic relatedness), which are then locally excluded from the analysis. But that exclusion is not reflected back to the original data, and the same bad samples are included in the next analysis. In this way a record exists of which samples can be excluded in future analysis.

## 1.6 How the BCO community will grow

The BCO working groups facilitates a means for different stakeholders in the HTS (NGS) communities to provide input on current practices on the BCO. There has been a continual growth of the BioCompute Object working group due to interaction between variety of participants from all perspectives in standardization of computational HTS(NGS) data. The Public-Private partnerships formed between universities, private genomic data companies, software platforms, government and regulatory institutions has been an easy point of entry of new individuals or institutions into the BCO project to engage in the discussion of best practices for the objects.

# 2 Data type for BCOs

The fundamentals of data typing (type primitives, class inheritance, etc) that are used to define BioCompute Objects are described in detail in section Appendix VI. Developers of BCO enabled platforms should reference this section for details on how to support the creation of BCO programmatically or manually. BCOs are represented in JSON (JavaScript Notation) formatted text. The JSON format was chosen because it is both human and machine readable/writable. For a detailed description of JSON see [www.json.org](http://www.json.org).

BioCompute datatypes are defined as aggregates of critical fields organized into a few domains: descriptive domain, identification and provenance domain, input and output domains, parametric domain, environmental domain, execution domain, prerequisite domain, usability domain, and error domain. At the moment of submission to the BioCompute database an instance of BCO type is created, populated with actual values compliant with the data type definitions and assigned a unique identifier. The object could then be assigned a unique digital signature. (see security section, Appendix V.)

Three of the domains in a BioCompute Object become immutable upon assignment of the digital signature: 1) the Parametric Domain, 2) the Execution Domain and 3) the I/O Domain. Changing anything within these domains invalidates the verification and will break the digital signature. Optional fields are indicated by the "vital": "True" flag, which is shown in the data typing section below (Appendix VI).

## 2.1 Identification and Provenance fields

### 2.1.1 ID "id"

Namespaced (source/db/id) unique identifier of this BCO instance assigned by a BCO database engine. IDs should never be deprecated or reused.

"id": "https://hive.biochemistry.gwu.edu/bco\_db/obj.1270"

### 2.1.2 Name "name"

Name for the BCO. This public field should take free text value using common biological research terminology along with external reference linkage using identifiers whenever possible.

"name": "HCV1a [taxonomy:31646] ledipasvir [pubchem.compound:67505836] resistance SNP [so:0000694] detection"

### 2.1.3 Structured name "structured\_name"

Structured name is an optional templated computable text field designed to represent a BCO instance name in visible interfaces. This field can refer to other fields within the same or other objects. For example, a string like "HCV1a [taxid:$taxid] mutation detection" will be visualized as "HCV1a [taxonomy:31646] mutation detection" assuming the BCO has a field called taxonomy and value 31646.

HCV1a [taxonomy:$taxonomy] mutation detection = HCV1a [taxonomy:31646] mutation detection

### 2.1.4 Version "version"

Records the versioning of this BCO instance object. In BCO versioning, a change in the BCO which will affect the outcome of the computation should be deposited as a new BCO, not as a new version. If a parameter in a tool is changed within a BCO, which in turn changes the outcome of the pipeline and the original BCO, a new BCO, not a new version, will be created.

In such cases the connection between the new object and the older one may or may not be (on author’s discretion) retained in the form of references. Changes that cannot affect the results of the computation can be incorporated into a new version of the existing BCO. Such changes might include name and title, comments, authors, validity dates, etc.

"version": "1.1"

### 2.1.5 Digital signature "digital\_signature"

A string, read-only value generated by a BCO database, protecting the object from internal or external alterations without proper validation. This value should not be submitted during deposition but can be read during downloading or transferring validated BCOs. The BCO server can provide API validating the signature vs BCO content, allowing users to validate the signature "offline" on their own. The server will also be able to provide a reference to the signature creation algorithm, allowing for greater interoperability.

"digital\_signature": "905d7fce3f3ac64c8ea86f058ca71658"

### 2.1.6 Verification status "verification\_status"

Describes the status of an object in the verification process. The 'unreviewed' flag indicates that the object has been submitted, but no further evaluation or verification has occurred. The 'in\_progress' flag indicates that verification is underway. The 'reviewed' flag indicates that the BCO has been verified and reviewed. The 'manual' flag indicates that the object has been manually verified. The 'suspended' flag indicates an object that was once valid is no longer considered valid. The 'error' flag indicates that an error was detected with the BCO.

"verification\_status": "in\_progress" OR "unreviewed" OR "reviewed" OR "published" OR "rejected"

### 2.1.7 Publication status "publication\_status"

This is a choice field with three options. The 'draft' status indicates that an object is in draft form and is still being edited. The 'open\_access' status indicates that an object has been published and is freely available to anyone. The objects with the 'private' status have restrictions on who can view and access them. This is a way for researchers using restricted data or metadata to ensure the confidentiality is maintained. The permissions to the object will be defined by the database access control rules.

"publication\_status": "draft" OR "in\_progress" OR "private" OR "open\_access"

### 2.1.8 Authors "authors"

A list to hold author identifiers. We encourage the use of ORCIDs to record author information, as they allow for the author to curate their information after submission. If an author does not have an ORCID then they can provide a name using free unicode text.

"author":["Charles Darwin", "https://orcid.org/0000-0000-0000-0000"]

## 2.2 Usability domain "usability\_domain"

This field provides a space for the author to define the usability domain of the BCO. It is an array of free text values. This field is to aid in search-ability and provide a specific description of the object. The usability domain along with keywords can help determine when and how the BCO can be used. Novel use of the BCO could result in the creation of a new entry with a new usability domain.

"Identify baseline single nucleotide polymorphisms SNPs [SO:0000694], insertions [SO:0000667], and deletions [SO:0000045] that correlate with reduced ledipasvir [PubChem:67505836] antiviral drug efficacy in Hepatitis C virus subtype 1 [taxID:31646]"

"GitHub CWL example: https://github.com/mr-c/hive-cwl-examples/blob/master/workflow/hive-viral-mutation-detection.cwl#L20"

## 2.3 Description Domain "description\_domain"

Structured field for description of external references, the pipeline steps, and the relationship of I/O objects. Information in this domain is not used for computation. This domain is meant to capture information that is currently being provided in FDA submission in journal format. It is possible that in the future this field can be semi-automatically generated from the execution\_domain information.

### 2.3.1 Keywords "keywords"

This is an array field to hold a list of keywords to aid in search-ability and description of the object.

"keywords": ["antiviral resistance", "SNP", "Ledipasvir", "HCV1a", "amino acid substitution"]

### 2.3.2 External References "xref"

This field contains a list of the databases and/or ontology IDs that are cross-referenced in the BCO. The external references are used to provide more specificity in the information related to BCO entries. Cross-referenced resources need to be available in the public domain. The external references are stored in the form of prefixed identifiers (CURIEs). These CURIEs map directly to the URIs maintained by identifiers.org. See Appendix-II for more a list of the URIs.

"xref": ["so:0000694", "so:0000667", "so:0000045", "pubchem.compound:67505836", "so:0000048", "taxonomy:31646", "pubmed:25123381", "pubmed:26508693"]

#### 2.3.2.1 Extension to External References: FHIR

The external references includes an optional extension to FHIR resource where specified data elements can be extracted from EHR systems without compromising patient and providers’ information. This is because the portions being transferred contain no identifiable information about the patient. Instead there is a reference to the actual resource instance (via FHIR URI).

The FHIR URI leads to the report that will contain the date and time of the procedure, specimen details, and a set of observations in the diagnostic results as a narrative. The diagnostic report will also contain the information that pertains to the sequence of interest in the BCO such as the biological information of the sequence, the FHIR defined ID, the sequence code, base number of the coordinate system, reference of patient (not who it is but what it is about), reference used for specimen sequencing, method used for sequencing, organization responsible for the test result, the number of copies of the specified sequence, the reference sequence, chromosome which contains the genetic finding, start and end positions of the chromosomes, quality scores of the sequence, etc. The link to FHIR can also be added to the usability domain.

"FHIR\_extension":[

{

"FHIR\_resourceURL": "https://www.hl7.org/fhir/resourcelist.html",

"diagnostics": "DiagnosticsReport",

"identifier": "business identifier for the report",

"reference": "the procedural request"

},

{

"diagnostics": "sequence",

"sequence\_identifier": "unique ID for the specific sequence",

"datasetID": "ID for the dataset",

"type": "sequence type such as aa, dna or rna",

"reference\_specimen": "specimen used for sequencing"

}

]

#### 2.3.2.2 Extension to External References: GitHub

The external references also includes the extension to GitHub repository where the NGS computational analysis pipeline/workflow protocols can be stored, source code of the software or tools can be deposited and various versions of the software or tools can be maintained. The BCO would contain link to the GitHub repository where the information is stored and easily retrieved. The links to GitHub can be added to the usability domain.

"GitHub CWL example: https://github.com/mr-c/hive-cwl-examples/blob/master/workflow/hive-viral-mutation-detection.cwl#L20"

### 2.3.3 Pipeline tools "pipeline\_steps"

An optional structured domain for recording the specifics of a pipeline. Each individual tool (or a well defined and reusable script) is represented as step, at the discretion of the author. Parallel processes are given the same step number. URIs are listed for the inputs and outputs for each tool.

#### 2.3.3.1 Tool Name "tool\_name"

Name for the specific tool. This field is a string (A-z, 0-1) and should be a single uniquely identifying word for the tool.

"tool\_name": "HIVE-hexagon"

#### 2.3.3.2 Tool Description "tool\_desc"

A free text field for describing the specific use/purpose of the tool

"tool\_desc": "Alignment of reads to a set of references",

#### 2.3.3.3 Tool Version "tool\_version"

The version assigned to the the instance of the tool used

"tool\_version": "1.3",

#### 2.3.3.4 Tool Requirements or Packages "tool\_package"

The is a list of text values to indicate any packages or prerequisites for running the tool used.

"tool\_package": ["Python version 2.6", "megablast version 1.2"]

#### 2.3.3.5 Step Number "step\_number"

This is an integer value representing the position of the tool in the one dimensional representation of the pipeline. Parallel computations are assigned the same number.

"step\_number": "1"

#### 2.3.3.6 Input URI List "input\_uri\_list"

Each tool lists the URI of the input files for that tool.

"input\_uri\_list": [

"http://www.ncbi.nlm.nih.gov/nuccore/22129792",  
 "http://www.ncbi.nlm.nih.gov/nuccore/5420376",  
 "http://www.ncbi.nlm.nih.gov/nuccore/13122261",  
 "http://www.ncbi.nlm.nih.gov/nuccore/386646758",  
 "http://www.ncbi.nlm.nih.gov/nuccore/295311559",  
 "hive://nuc-read/514683",

"hive://nuc-read/514682"

"hive://data/514769/dnaAccessionBased.csv"

]

#### 2.3.3.7 Output URI List "output\_uri\_list"

Each tool lists the URI of the output files for that tool.

"output\_uri\_list": [

"hive://data/514769/allCount-aligned.csv"

"hive://data/514801/SNPProfile.csv",

"hive://data/14769/allCount-aligned.csv"

]

## 2.4 Execution domain "execution\_domain"

The fields required for execution of the BCO have been encapsulated together in order to clearly separate information needed for deployment, software configuration and running applications in a dependent environment. One byproduct of an accurate BCO definition is facilitation of reproducibility as defined by the *Oxford English Dictionary* as "the extent to which consistent results are obtained when produced repeatedly."

### 2.4.1 Script Type "script\_type"

This is a choice field to indicate whether the "script" to execute the BioCompute Object is a reference to an external file (URI) or text in the "script" field.

"script\_type": "URI" OR "text"

### 2.4.2 Script "script"

The internal or external reference to a script object that was used to perform computations for this BCO instance. This may be a reference to Galaxy Project or SB-genomics pipeline, a Common Workflow Language (CWL) object in GitHub, a High-performance Integrated Virtual Environment (HIVE) computational service or any other type of script.

"script": "https://hive/workflows/antiviral\_resistance\_detection\_hive.py"

### 2.4.3 Pipeline Version "pipeline\_version"

This field records the version of the pipeline implementation.

"pipeline\_version": "2.0"

### 2.4.4 Platform/Environment "platform"

The multi-value reference to a particular deployment of an existing platform where this BCO can be reproduced. A platform can be a bioinformatic platform such as Galaxy or HIVE or it can be a software package such as CASAVA or apps that includes multiple algorithms and software.

"platform": "HIVE"

### 2.4.5 Script driver "script\_driver"

The reference to an executable that can be launched in order to perform a sequence of commands described in the script (see above) in order to run the pipeline. For example, if the pipeline is driven by a HIVE script, the script driver is the "hive" execution engine. For CWL based scripts, the cwl-runner tool is the driver which is capable of interpreting the commands in the script. Another very general script driver commonly used in Linux based operating systems is "shell" and the type of scripts it can run are operating system shell scripts. The combination of script driver and script is a capability to run a particular sequence of computational steps in order to produce BCO outputs given the inputs and parameters.

It is noteworthy to mention that scripts and script drivers by themselves can be objects. These objects can exist in internal (BCO) or external databases and be publically or privately accessible.

"driver": "shell"

### 2.4.6 Algorithmic tools and Software Prerequisites "software\_prerequisites"

An optional multi-value field listing the minimal necessary prerequisites, library, tool versions needed to successfully run the script to produce BCO.

"software\_prerequisites": [

{"name":"HIVE\_hexagon","version":"1.3"},

{"name":"HIVE\_heptagon","version":"1.3"}

]

### 2.4.7 Domain Permissions "domain\_prerequisites"

An optional multi-value field listing the minimal necessary domain specific external data source access in order to successfully run the script to produce BCO.

"domain\_prerequisites": [

{"url":"protocol://domain:port/application/path",  
 "Name":"generic name"},

{"url":"ftp://:22/",  
 "Name": "access to ftp"},

{"url":http://eutils.ncbi.nlm.nih.gov/entrez/eutils",  
 "Name":"access to e-utils"}

]

### 2.4.7 Environmental parameters "env\_parameters"

Multi-value additional key value pairs useful to configure the execution environment on the target platform. For example, one might specify the number of compute cores, or available memory use of the script.

"env\_parameters": [{"OSTYPE":"linux"},{"QPRIDE\_BIN":"~qpride/bin"}]

## 2.5 Parametric domain "parametric\_domain"

This represents the list of parameters customizing the computational flow which can affect the output of the calculations. These fields are custom to each type of analysis and are tied to a particular pipeline implementation. All BCOs should inherit from the fundamental BioCompute data type and as such inherit all of the core fields described in document. Specific BioCompute types introduce specific fields designed to customize the use of pipelines for a particular use pattern. Please refer to documentation of individual scripts and specific BCO descriptions for details.

"parametric\_domain": {

"heptagon\_divergence\_threshold\_percent": "30",

"hexagon\_minimum\_coverage": "0.15",

"hexagon\_seed": "14",

"heptagon\_freq\_cutoff": "0.10",

"hexagon\_minimum\_match\_len": "66"

},

## 2.6 Input and output domains "io\_domain"

This represents the list of global input and output files created by the computational workflow, excluding the intermediate files. These fields are pointers to objects that can reside in the system performing the computation or any other accessible system. Just like the fields of parametric domain, these fields are custom to every specific BCO implementation and can refer to named input output arguments of underlying pipelines. Please refer to documentation of individual scripts and specific BCO descriptions for further details.

### 2.6.1 Input Subdomain "input\_subdomain"

This field records the references and input files for the entire pipeline. Each type of input file is listed under a key for that type. The file types are specified when the BCO type is created.

"input\_subdomain": {

"HCV\_Genome\_to\_detect\_variations":[

"http://www.ncbi.nlm.nih.gov/nuccore/22129792",

"http://www.ncbi.nlm.nih.gov/nuccore/5420376"

],

"read\_files": [

"hive://nuc-read/514683",

"hive://nuc-read/514682"

]

}

### 2.6.2 Output Subdomain "output\_uri\_list"

This field records the outputs for the entire pipeline. Each file should be an object with a key, and a title, URI, and mime-type (https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics\_of\_HTTP/MIME\_types) value.

"output\_subdomain": {

"hit\_list$Obj\_ID": {

"title": "hit list",

"uri": "hive://data/514769/dnaAccessionBased.csv",

"mime-type" : "text/csv"

},

"mutation\_profile": {

"title": "mutation profile",

"uri": "hive://data/514801/SNPProfile\*.csv",

"mime-type": "text/csv"

}

}

## 2.7 Error domain, acceptable range of variability "error\_domain"

The error domain consists of two subdomains: empirical and algorithmic.

The empirical error subdomain contains the limits of detectability, false positives, false negatives, statistical confidence of outcomes, etc. This can be measured by running the algorithm on multiple data samples of the usability domain or in carefully designed in-silico spiked data. For example a set of spiked, well characterized samples can be run through the algorithm to determine the false positives, negatives and limits of detection.

The algorithmic subdomain is descriptive of errors that originated by fuzziness of the algorithms, driven by stochastic processes, in dynamically parallelized multi-threaded executions, or in machine learning methodologies where the state of the machine can affect the outcome. This can be measured in repeatability experiments of multiple runs or using some rigorous mathematical modeling of the accumulated errors. For example: Bootstrapping is frequently used with stochastic simulation based algorithms to accumulate sets of outcomes and estimate statistically significant variability for the results.

"error\_domain": {

"empirical\_error": {

"false negative alignment hits": "<0.0010"

},

"algorithmic\_error": {

"false positive mutation calls discovery": "<0.0005"

}

}

# 3 Appendices

## 3.1 Appendix-I: BCO expanded view example

{

"id": "obj.1270",

"name": "HCV1a [taxonomy:31646] ledipasvir [pubchem.compound:67505836] resistance SNP [so:0000694] detection",

"version": "1.1",

"createdby": "hadley\_king@gwmail.gwu.edu",

"created": "Jan 24, 2017 09:40:17",

"modified": "Mar 27, 2017 13:27:02",

"digital\_signature": "905d7fce3f3ac64c8ea86f058ca71658",

"verification\_status": "unreviewed",

"publication\_status": "draft",

"usability\_domain": [

"Identify baseline single nucleotide polymorphisms SNPs [so:0000694], insertions [so:0000667], and deletions [so:0000045] that correlate with reduced ledipasvir [pubchem.compound:67505836] antiviral drug efficacy in Hepatitis C virus subtype 1 [taxonomy:31646]",

"Identify treatment emergent amino acid substitutions [SO:0000048] that correlate with antiviral drug treatment failure",

"Determine whether the treatment emergent amino acid substitutions [SO:0000048] identified correlate with treatment failure involving other drugs against the same virus",

"GitHub CWL example: https://github.com/mr-c/hive-cwl-examples/blob/master/workflow/hive-viral-mutation-detection.cwl#L20"

],

"authors": ["Charles Darwin", "https://orcid.org/0000-0000-0000-0000"],

"description\_domain": {

"keywords": [

"Antiviral resistance",

"SNP",

"Ledipasvir",

"HCV1a",

"Amino acid substitution"

],

"xref": [

"so:0000694",

"so:0000667",

"so:0000045",

"pubchem.compound:67505836",

"so:0000048",

"taxonomy:31646",

"pubmed:25123381",

"Pubmed:26508693"

],

"pipeline\_steps": [

{

"tool\_name": "HIVE-hexagon",

"tool\_desc": "Alignment of reads to a set of references",

"tool\_version": "1.3",

"tool\_package": "",

"step\_number": "1",

"input\_uri\_list": [

"http://www.ncbi.nlm.nih.gov/nuccore/22129792",

"http://www.ncbi.nlm.nih.gov/nuccore/5420376",

"http://www.ncbi.nlm.nih.gov/nuccore/13122261",

"http://www.ncbi.nlm.nih.gov/nuccore/386646758",

"http://www.ncbi.nlm.nih.gov/nuccore/295311559",

"hive://nuc-read/514683",

"hive://nuc-read/514682"

],

"output\_uri\_list": [

"hive://data/514769/allCount-aligned.csv"

]

},

{

"tool\_name": "HIVE-heptagon",

"tool\_desc": "variant calling",

"tool\_version": "1.3",

"tool\_package": "",

"step\_number": "2",

"input\_uri\_list": [

"hive://data/514769/dnaAccessionBased.csv"

],

"output\_uri\_list": [

"hive://data/514801/SNPProfile.csv",

"hive://data/14769/allCount-aligned.csv"

]

}

]

},

"execution\_domain": {

"script\_type": "URI",

"script": "https://hive.biochemistry.gwu.edu/workflows/antiviral\_resistance\_detection\_hive.sh",

"pipeline\_version": "2.0",

"platform": "hive",

"driver": "shell",

"software\_prerequisites": [

{"name":"HIVE-hexagon","version":"1.3"},

{"name":"HIVE-heptagon","version":"1.3"}

],

"access\_prerequisites": [

{

"url":"protocol://domain:port/application/path",

"name":"generic name"

},

{

"url":"ftp://:22/",

"name": "access to ftp"

},

{

"url":"http://eutils.ncbi.nlm.nih.gov/entrez/eutils/",

"name":"access to e-utils"

}

],

"env\_parameters": [

{"OSTYPE":"linux"},

{"QPRIDE\_BIN":"~qpride/bin"}

]

},

"parametric\_domain": {

"heptagon\_divergence\_threshold\_percent": "30",

"hexagon\_minimum\_coverage": "0.15",

"hexagon\_seed": "14",

"heptagon\_freq\_cutoff": "0.10",

"hexagon\_minimum\_match\_len": "66"

},

"io\_domain": {

"input\_subdomain": {

"HCV\_Genome\_to\_detect\_variations":[

"http://www.ncbi.nlm.nih.gov/nuccore/22129792",

"http://www.ncbi.nlm.nih.gov/nuccore/5420376"

],

"read\_files": [

"hive://nuc-read/514683",

"hive://nuc-read/514682"

]

},

"output\_subdomain": {

"hit\_list": {

"title": "hit list",

"uri": "hive://data/514769/dnaAccessionBased.csv",

"mime-type" : "text/csv"

},

"mutation\_profile": {

"title": "mutation profile",

"uri": "hive://data/514801/SNPProfile\*.csv",

"mime-type": "text/csv"

}

}

},

"error\_domain": {

"empirical\_error": {

"false negative alignment hits": "<0.0010"

},

"algorithmic\_error": {

"false positive mutation calls discovery": "<0.0005"

}

}

}

## 3.2 Appendix-II: External reference database list

This list contains the databases that are currently being used in our BCOs. We use the CURIEs that map to URIs maintained by *identifiers.org*.

“*Identifiers.org* is an established resolving system the enables the referencing of data for the scientific community, with a current focus on the Life Sciences domain. *Identifiers.org* provides direct access to the identified data using one selected physical location (or resource). Where multiple physical locations are recorded in the [registry](http://identifiers.org/registry) the most stable one is selected for resolution. This allows the location independent referencing (and resolution if required) of data records.”

In the entries below the “namespace” and identifier combine to become the CURIEs.

##### Recommended name: Taxonomy

##### Namespace: taxonomy Identifier pattern: ^\d+$ Registry identifier: MIR:00000006

##### URI: http://identifiers.org/taxonomy/

##### Recommended name: Sequence Ontology

##### Namespace: so Identifier pattern: ^SO:\d{7}$ Registry identifier: MIR:00000081

##### URI: http://identifiers.org/so/

##### Recommended name: PubMed

##### Namespace: pubmed Identifier pattern: ^\d+$ Registry identifier: MIR:00000015

##### URI: http://identifiers.org/pubmed/

##### Recommended name: PubChem-compound

##### Namespace: pubchem.compound Identifier pattern: ^\d+$ Registry identifier: MIR:00000034

##### URI: http://identifiers.org/pubchem.compound/

## 3.3 Appendix III - FDA draft guidance documents and Title 21 CFR Part 11

### 3.3.1 FDA Draft Guidance Documents.

The FDA has proposed draft guidance documents proposing methods to streamline oversight of HTS (NGS)-based tests. The documents are open for public input and we strongly encourage BioCompute stakeholders to contribute their critical considerations. Significant expertise of BioCompute stakeholders can serve as a forum to efficiently develop a coherent vision, evaluate validity and utility of these guidance documents.   
  
This guidance document is an early attempt to signify the importance of computation and data provenance and is congruent to larger BioCompute effort and contains information from BCO domains. Some of the suggested computational aspects described in the draft guidance proposal are summarized:

* All the software used in the workflow requires detailed documentation on versioning, source and modification.
* Documentation is required for the software versions and attribution, reference sequence assembly and components needed to compile, install and run (remotely or locally) the bioinformatics pipeline.
* Documentation and specificity of databases which were used internally or externally should be included.

Relevant draft guidance document links:

[DRAFT Guidance: Use of Standards in FDA Regulatory Oversight of Next Generation Sequencing (NGS)-Based In Vitro Diagnostics (IVDs) Used for Diagnosing Germline Diseases](http://www.fda.gov/downloads/MedicalDevices/DeviceRegulationandGuidance/GuidanceDocuments/UCM509838.pdf)

[Use of Public Human Genetic Variant Databases to Support Clinical Validity for Next Generation Sequencing (NGS)-Based In Vitro Diagnostics](https://www.fda.gov/downloads/MedicalDevices/DeviceRegulationandGuidance/GuidanceDocuments/UCM509837.pdf)

### 3.3.2 Title 21 CFR Part 11

*Code of Federal Regulations Title 21 Part 11: Electronic Records - Electronic Signatures*

BioCompute project is being developed with Title 21 CFR Part 11 compliance in mind. The digital signatures incorporated into the format will provide the basis for provenance of BioCompute Object integrity using NIST proposed encryption algorithms. Execution domain and parametric domain (that have a potential impact on a result of computation) and identity domain will be used to create hash values and digital signature encryption keys which later can be used for computer or human validation of transmitted objects.   
  
Discussions are now taking place to consider relevance of BioCompute Objects with relation to Title 21 CFR part 11. We encourage continuous input from BioCompute stakeholders on this subject now and while the concept is becoming more mature and more widely accepted by scientific and regulatory communities.

Relevant document link:

[Part 11: Electronic Records](http://www.fda.gov/RegulatoryInformation/Guidances/ucm125067.htm)

## 3.4 Appendix IV - Compatibility

### 3.4.1 ISA for the experimental metadata

ISA is a metadata framework to manage an increasingly diverse set of life science, environmental and biomedical experiments that employ one or a combination of technologies. Built around the **Investigation** (the project context), **Study** (a unit of research) and **Assay** (analytical measurements) concepts, ISA helps to provide rich descriptions of experimental metadata (i.e. sample characteristics, technology and measurement types, sample-to-data relationships) so that the resulting data and discoveries are reproducible and reusable. The ISA Model and Serialization Specifications define an Abstract Model of the metadata framework that has been implemented in two format specifications, ISA-Tab and ISA-JSON (<http://isa-tools.org/format/specification>), both of which have supporting tools and services associated with them, including by a programmable Python AP ([http://isa-tools.org](http://isa-tools.org/format/specification)) and a varied user community and contributors (<http://www.isacommons.org>). ISA focuses on structuring experimental metadata; raw and derived data files, codes, workflows etc are considered as external file that are referenced. An example, along its complementarity with other models and a computational workflow is illustrated in this paper, which shows how to explicitly declare elements of experimental design, variables, and findings: <http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0127612>

## 3.5 Appendix VI Data typing

The conceptual schema for BCO creation is built on top of two layers: the data definition layer and the BCO layer. The first layer is where all fundamental data types are defined. Complex types are composed of multiple atomic or complex types, like a character string. Using these principles one can construct a datum that has the ability to represent any level of complexity that is needed. A BCO is essentially a federation of other objects.

### 3.5.1 Primitive data types

When defining a field in a data type, one can place any number of constraints on the data and the field will be accepted as valid. So, if a data type field was being constructed for holding DNA sequencing information, one could restrain the type of characters that field would accept. This further refinement would ensure that only the characters used for representing nucleic acids would be accepted as input in this field (e.g. A, T, C, G). A list of the primitive types used in BCO data typing is below.

{

"primitives" : {

"description": "primitives branch",

"bool" : {

"\_type": "core",

"\_limit" : {

"true" : 1,

"false" : 0

}

},

"int" : {

"\_type" : "core"

},

"uint" : {

"\_type" : "int",

"\_limit" : {

"eval-c/c++" : "$val>=0"

}

},

"percent" : {

"\_type" : "uint",

"\_limit" : {

"eval-c/c++" : "$val<=100 && $val>=0"

}

},

"string" : {

"\_type" : "core"

},

"text" : {

"\_type" : "core"

},

"memory" : {

"\_type" : "uint"

},

"cmdline" : {

"\_type" : "text"

},

"url" : {

"\_type" : "text"

},

"file" : {

"\_type" : "string"

},

"date" : {

"\_type" : "core",

"\_default" : "$datenow()"

},

"time" : {

"\_type" : "core",

"\_default" : "$timenow()"

},

"datetime" : {

"\_type" : "core",

"date": {

"\_type": "date",

"\_default" : "$datenow()"

},

"time": {

"\_type": "time",

"\_default" : "$timenow()"

}

},

"timespan" : {

"\_type" : "core",

"start" : {

"\_type" : "datetime"

},

"end" : {

"\_type" : "datetime"

}

},

"ref" : {

"\_type" : "core",

"source" : {

"\_type" : "string"

},

"db" : {

"\_type" : "string"

},

"id" : {

"\_type" : "string",

"\_vital": true

}

},

"password" : {

"\_type" : "string"

},

"email" : {

"\_type" : "string"

},

"keyval" : {

"\_type" : "core",

"key" : {

"\_type" : "string"

},

"value" : {

"\_type" : "string",

"\_plural" : true

}

}

}

}

### 3.5.2 Base BioCompute Type

The second layer is constructed with objects from first layer, producing a derived data type called the "base BioCompute type". Extending the same principles that allowed us to construct a string representing a DNA sequence from the primitive character type, one can construct a type definition that is the absolute minimum fields necessary to create a BCO. By taking the primitive BCO type and adding parametric and metadata fields unique to a particular instance, a BCO can be created. Below it the type definition for “BioCompute\_base\_type”:

{

"BioCompute\_base\_type": {

"\_type" : "type",

"\_id" : "$newid()",

"\_field": {

"id": {

"\_type" :"string",

"title" : "Identifier of the object",

"\_public" : true,

"\_write" : false

},

"name": {

"\_type" :"string",

"title" : "public searchable name for BioCompute Objects",

"\_public" : true

},

"structured\_name": {

"\_type" :"string",

"title" : "public searchable name for BioCompute Objects",

"\_public" : true,

"\_vital" : false

},

"version": {

"\_type" :"string",

"title" : "version for BioCompute Object",

"\_public" : true

},

"digital\_signature": {

"\_type" : "string",

"title" : "digital signature of the BioCompute Object",

"\_write":false,

"\_vital" : false

},

"verification\_status": {

"\_type" : "string",

"\_limit": {

"choice ": ["in\_progress", "unreviewed", "reviewed" , "published", "rejected" ]

},

"title" : "the current verification status of the BioCompute Object",

"\_default" : "in\_progress"

},

"publications\_status": {

"\_type" : "string",

"\_limit": {

"choice ": ["draft", "in\_progress", "published", "embargoed" ]

},

"title" : "the current publication status of the BioCompute Object",

"\_default" : "draft"

},

"authors": {

"\_type" : "core",

"name": {

"\_type" : "string",

"\_hidden" : "eval-js: orcid===undefined"

},

"orcid": {

"\_type" : "string",

"\_hidden" : "eval-js: name===undefined"

},

"\_plural": true

},

"usability\_domain": {

"\_type" : " string",

"\_plural": true,

"descr" : "text from biospec"

},

"description domain": {

"keywords": {

"\_type": "keyval",

"\_plural": true,

"vital": false

},

"xref": {

"\_type": "xref",

"\_plural": true,

"vital": false

},

"pipeline\_steps": {

"tool\_name" : {

"\_type": "string",

"descr" : "this is a recognized name of the software tool"

},

"tool\_desc" : {

"\_type": "text"

},

"tool\_version": {

"\_type": "string"

},

"tool\_package" : {

"\_type": "string",

"\_plural": true

},

"step\_number": {

"\_type": "uint",

"\_limit": {

"eval-js": "$step\_number >= 1"

}

},

"input\_uri\_list" : {

"\_type" : "url",

" \_vital": false

},

"output\_uri\_list" : {

"\_type" : "url",

" \_vital": false

},

"\_plural": true

},

"\_vital": false

},

"execution\_domain": {

"script\_type": {

"\_type":"string",

"\_limit": {

"choice": ["uri","text" ]

}

},

"script": {

"\_type": "$script\_type"

},

"script\_driver": {

"\_type": "string"

},

"pipeline\_version": {

"\_type": "string"

},

"platform": {

"\_type": "string"

},

"software\_prerequisites": {

"name": {

"\_type": "string"

},

"version": {

"\_type": "string"

},

"url": {

"\_type": "url",

"\_vital": false

},

"\_plural": true

},

"access\_prerequisites": {

"name": {

"\_type": "string"

},

"url": {

"\_type": "url",

"\_plural" : true

}

},

"environmental\_parameters": {

"\_type": "keyval",

"\_plural" : true

},

"\_vital" : true

},

"parametric\_domain": {

"descr" : "all fields in this domain should be defined in inheriting BioCompute subtypes"

},

"io\_domain": {

"input\_subdomain": {

"descr" : "all fields in this subdomain are specific BioCompute specific and should be defined in inheriting BioCompute subtypes"

},

"output\_subdomain":{

"descr" : "all fields in this subdomain are specific BioCompute specific and should be defined in inheriting BioCompute subtypes"

}

}

},

"\_inherit" : "base\_database\_object",

"name" : "BioCompute\_base\_type",

"title" : "Base type for all BioCompute Objects",

"descr" : "all BioComputes must inherit from this type in order to be compliant with BioCompute framework"

}

}

### 3.5.3 Meaning associations

All data field values stored in the data format have an associated file name and/or absolute or relative field value location. The association and assignment of meaning to a particular data value can be ontological – derived from a name-value pair, or topological – derived from a location-value pair.

*As an example, in a name-value pair, the name of the field is explicitly specified. Examples of name-value pairing are: a JSON object; the mapping from ids to nodes in an XML document; or the association between the column names and columns of a CSV table. In a location-value pair, the name of the field is not specified but assumed, according to the order and positioning of the value assumes its meaning. An example of location-value pairing is a FASTQ file, where sequence and quality lines do not have explicit names, but their relative locations are used to identify the data line type.*

### 3.5.4 Atomic and complex data

A data value can be atomic, not decomposable into smaller sets without the loss of meaning, or complex, containing other simpler values or topologies. Data structures of arbitrary complexity can be created using substructures and hybrids of name-value and location-value aggregations. Some fields may be declared parent fields of others and the field name value pairs can be allocated along the hierarchy of field types. It is important to note that atomicity of the field values here does not directly imply independent scientific interpretability of the atomic values: it only implies that such information cannot be fragmented into smaller more fundamental types.

*For example, the value for a person's age is an atomic value and cannot be decomposed. However, a person's identity is a complex data which contains first name, last name, age, social security number, passport number and perhaps other simpler values which have a meaning in of their own.*

### 3.5.5 Fundamental and derivable types

Primitive field types, such as numbers, strings, bit-fields, and uncharacterized blobs, are generally the most frequently used: this fact is reflected in computer science where these four types are chosen as fundamental units of data representation. A data type by itself does not carry physical interpretation; this interpretation comes from the value associations mentioned above. Any other field types (sequences, alignments, etc.), atomic or complex, can be constructed by using the fundamental types and adjusting their interpretive significance. Computable data types, such as references and formulas, are also useful for creating operational information infrastructures; such fields do not carry a value themselves but link to another value, in the same or another object, directly or through the usage of a transformation formula.

In the person’s identity example mentioned in the previous section the atomic subfields first name, last name, age, etc. are primitive data types such as strings and numbers.

Data typing is the process of creating a derived data type as a collection of primitive field types and previously defined complex types. During this process, field attributes such as name, type, constraints, default value are specified for each of the fields.

*To construct the type "Identity" we define a collection of the fundamental fields that comprise this derived type.* ***Identity:***

* *First name is a free text string.*
* *Last name is a free text string.*
* *Age is a single whole number.*

*Identity: {Charles, Darwin, 208}*

Derived/computed fields

Derived/computed data types are produced during the output of the object instance and cannot be entered during instantiation of the object into the database. This fields are just the reproducible outcome of other field values through some kind of derivation mechanisms. For example: a taxonomy identifier is non-redundantly unique enough to recover different names, synonyms and relations for a taxonomic unit. However a derived field "name" can be a useful informative construct derived and outputted every time where taxonomic identification is necessary.

*TaxID: 9606*

Virtual fields

Field values can be explicitly specified in an object instance or computable by other means. For example, the metadata object describing a file can have a field named "file-size" reflective of the actual file size in storage. The value of that field cannot be explicitly specified or computed from other values in the same object. Such virtual fields are descriptive and can be used for validation of the object itself and for definition of further constraints.

Constraints

Field values can have explicit and implicit constraints determining the universe of possible values for that particular field. Implicit constraints may be derived from the actual data types where, for example, numeric fields can have only numeric values and strings can have only characters of a particular alphabet. Explicit constraints are those specifically targeting a particular field in a data format, further limiting the value to a smaller set of possible values. Implicit and explicit constraints can be validated either by ensuring their syntax conformance or by evaluation of validation expression criteria. Correlation constraints can be defined on a field as well, demanding a condition when constraints on one field depend on the value(s) of other fields. There can be a scale of constraint rigorousness depending on the impact due to violation of such constraints. Soft constraint nonconformity can signify potential devaluation of a particular field value pair while hard constraint violation in a single field value pair would invalidate the entirety of the data as a whole.

Existence of static and dynamic constraints is necessitated by the changing nature of interdependencies of local versus global information. Static constraints are those whose validation expression depends only on the data itself while dynamic constraints depend on external or dynamically changeable information from local or global sources. An example of an explicit, dynamic, and correlated constraint-carrying field type is XREF (cross-reference) for which both the ID and the cross-referenced source must be included and the validity of the field can be evaluated only by inclusion of potentially changeable dynamic content of the external source.

It is important to note that constraints may limit the universe of possible values to discrete or continuous subsets of a finite or infinite enumerable list of values. Actual implementations of constraints may vary from a form of numerical open (]min,max[, closed ([min,max] or mixed ([min,max[, ]min,max]) ranges, lists of ranges, lists of strings and possible values to mathematical expressions producing true or false Boolean values based on the computation of a formula.

Single and multi-value

An additional level of empowering complexity can be achieved if fields are allowed to have multiple values. Unlike single-value fields, a multi-value field carries multiple values associated with a single name-value or location-value assignation. Using this paradigm, one can introduce notions of a list and an array into data format containers. The difference between these types is that an array’s children elements are topologically bound to their location in a row and therefore, are related to each other. In the list however, children elements of the same or of a different kind are simply a collection of non-coupled values.

Public and private fields

Security and access control rights play an important role when considering data access patterns in modern collaborative information systems. Data can be shared in such environments for viewing and modifications. One may want to share only a subset of fields containing certain descriptive values broadcasting information about the existence of a particular metadata instance through a search and browsing engine. Thus, the researcher or collaborator who is interested in having access to your data may then ask for more complete (read or write) access to the dataset that was found through browsing its public properties. To satisfy this need, we can have an attribute called public for a field that declares it to be available for searching and browsing by other users.

Uniqueness and key values

The importance of any particular field’s role in data format definition can vary within a local or global context. Some fields are unique within the given instance (object) of the data file and some are required to be unique within the global scope of such objects in some database. Such fields can be used as unique key identifiers of a whole data file within a known context.

Incompleteness

Data formats can support both mandatory and optional fields. A valid data file must include all its mandatory fields without exception. Such fields are typically essential in representation of the underlying information and their absence can devalue or introduce ambiguities into interpretation of other fields. Optional fields are those carrying non-critical values; absence of those values does not devalue the interpretation of other fields. There are two possible approaches to optional fields: default value and undefined value. An empty optional field can be treated as having an agreed upon predefined default value or can simply call the value undefined or NULL which must be interpreted as non-equal to any other value, even to another undefined. Any operation performed with undefined values must also be assigned the value of undefined.

Arbitrary key-value pairs

Certain file formats allow incorporation of arbitrary key value pairs without specific designation with the objective of adopting unstructured, descriptive, or information that may be useful in the future. Although the ultimate goal of such arbitrary fields is to maintain some level of extensibility in formats, those arbitrary fields usually end up containing a significant amount of ill structured and unverifiable, non-canonical information. By providing other means to extensibility, we strongly discourage using arbitrary key-value pairs for anything other than purely descriptive, non-critical information.

### 3.6.5 Extensibility through inheritance and inclusion of data types

It is of the utmost importance to generate extensible metadata formats capable of providing the basis for more complex new types. There are two proposed ways to extend a data format: inheritance and inclusion.

The concept of inheritance assumes that a more complex data type inherits all the field value pairs from another, simpler data type and extends the content only with additional field value pairs or customizes (redefines some characteristics) of existing fields. The concept of inclusion assumes that a particular field of an object is of a previously declared complex type and that it contains all the fields of a simpler data type. A single data type can inherit from multiple data types and can include multiple data types multiple times.

Using these two paradigms, one can design a number of layered standard objects based on predefined objects and extend their functionality with specific fields. For example: imagine a metadata object of type bio-sample which has a predefined fundamental description applied to a generalized sample. This object can have its properties repeatedly inherited to create a human-sample object with increasingly specific information about particularities of that sample description.

The two proposed extensibility models allow avoiding the overuse of optional field attributes that are present in conglomerated flat data type designs. Instead of designing wide and flat data types with all possible fields for different use-cases, one may choose to design more targeted types with specifically mandated fields inside.

*For example, having an optional tissue-location field in all biological-sample objects might lead to sparse population of the field as it will be unpopulated for all environmental, metagenomic, bacterial, and viral samples where the notion of a tissue is irrelevant. However, designing an inherited animal-sample data type can have a mandatory tissue location field for instances when it is important to know from which part of the animal a particular sample was collected.*

The power of the inheritance and inclusion methods to extend and implement new data types is evident when one considers the need to create new subtypes or a branch of existing types after the initial data -type structure is established. This step can be accomplished without modification of existing database objects by defining the new intermediates within the framework of the pre-defined metadata type hierarchy.

The relationship implemented by inheritance subtyping is a is-a relationship. For example, the type "fish" can have three subtypes "eel", "shark" and "salmon". Each subtype is a variety of the "fish" supertype and inherits all "fish" characteristics but has some specific differences.

### 3.6.6 Data lifecycle timeline

Data objects are typically records stored in an information system: a file system or a database. The life of such a record starts at the moment of metadata file submission. Typical preprocessing steps include: files being parsed and validated regarding their conformity with data standards, application of quality control processes and designation of appropriate permissions for later use. Depending on the size and complexity of the data, as well as the load on the data processing subsystems, this period may take seconds to days for NGS data. After this initial preprocessing stage, objects become visible to the owner/submitter of the information.

The user can then specify the validity start time before which the object is not to be accessed by anyone other than the user. This feature is useful for providing pre-publication delays or time fixed processing procedures. The user can also specify the validity end period after which the object is not to be used by anyone other than the owner of the record.

Optional soft and hard expiration periods can be set. These properties signify when the object should become "expired" from the database and should be treated as "deleted," and when the record is actually deleted from the database, respectively. The timespan between these two time periods is the potential recovery period; during this period the deletion can be reverted by manual or electronic inquiry from the user to the DB administrator/manager.

Another important milestone of the data existence is set by FDA’s mandate to maintain an archival copy of any review data used to make regulatory decisions. This copy does not necessarily reside in any easily accessible database or file system and is managed by a different set of regulations, the description of which lies outside the scope of this document.

OLD MATERIAL DO NOT EDIT BELOW THIS SECTION

###############################################################################

Version 0.01 January 2017

High-throughput Sequencing Computational Standards for Regulatory Sciences (HTS-CSRS) Project

Contact: Raja Mazumder ([mazumder@gwu.edu](mailto:mazumder@gwu.edu)) and Vahan Simonyan ([vahan.simonyan@fda.hhs.gov](mailto:vahan.simonyan@fda.hhs.gov))

BioCompute Object Consortium members: Vahan Simonyan (CBER/FDA), Raja Mazumder (GW), Charles Hadley King (GW), Amanda Bell (GW), Jianchao Yao (Merck & Co), Jeremy Goecks (Oregon Health and Science University), Paul Walsh (NSilico Life Science), Jonas S Almeida (Stony Brook Univ, SUNY), Mark Walderhaug (FDA), Elaine Thompson (FDA), Alin Voskanian (CBER/FDA), Anton Golikov (CBER/FDA), Konstantinos Karagiannis (CBER/FDA), Dennis A. Dean, II (Seven Bridges), Stian Soiland-Reyes (UNIMAN), Carole Goble (UNIMAN), Michael R. Crusoe (CWL) …

# This content BELOW is **NO LONGER** being edited

Table Of Contents:

[**1 Introduction**](#_5j5cnj64ngh8) **6**

[1.1 Mission of the BioCompute project](#_1s38ctl7zmyy) 6

[1.2 Motivation](#_9vj10835pu7y) 6

[1.2.1 Limitations of initial Effort](#_ulpodjwvtuta) 7

[1.3 Audience for this document](#_p68hrlv1fo77) 7

[1.4 Potential Stakeholders for the entire BioCompute project](#_fp8neyi63yib) 7

[1.5 User stories](#_m8uh7ci8z2qo) 7

[**2 Data type for BCOs**](#_iv7zr6j11l0g) **8**

[2.1 Identification and Provenance fields](#_iwekywr5swyz) 9

[2.1.1 ID "id"](#_c2j45w5z1d8l) 9

[2.1.2 Name "name"](#_912sem3j7ihg) 9

[2.1.3 Structured name "structured\_name"](#_m9cqcvvqoxvo) 9

[2.1.4 Version "version"](#_ngw8udrf9hc9) 9

[2.1.5 Digital signature "digital\_signature"](#_wpjkkpl9wyd2) 10

[2.1.6 Verification status "verification\_status"](#_dbbe7b10vp3p) 10

[2.1.7 Publication status "publication\_status"](#_6ragmh27bq7m) 10

[2.1.8 Authors "authors"](#_woztxowyn3zh) 10

[2.1.8.1 Name "name"](#_j331n64xfji8) 11

[2.1.8.2 ORCID "orcid"](#_iuhg3g8vfydi) 11

[2.2 Usability domain "usability\_domain"](#_gny2kavoti6) 11

[2.3 Description Domain "description\_domain"](#_hci4bg7dwf1o) 11

[2.3.1 Keywords "keywords"](#_r0ydphil0j85) 11

[2.3.2 External References "xref"](#_6um148e41254) 12

[2.3.3.1 Tool Name "tool\_name"](#_4atfij2d99w0) 13

[2.3.3.2 Tool Description "tool\_desc"](#_50g75pl503y8) 13

[2.3.3.3 Tool Version "tool\_version"](#_w192tq7cvdme) 13

[2.3.3.4 Tool Requirements or Packages "tool\_package"](#_58ul2q69qu6d) 13

[2.3.3.5 Step Number "step\_number"](#_wv7rqly0ryt7) 13

[2.3.3.6 Input URI List "input\_uri\_list"](#_w3q2wtbp1ovc) 13

[2.3.3.7 Output URI List "output\_uri\_list"](#_vpsqk1v3k93x) 13

[2.4 Execution domain "execution\_domain"](#_2n1a58l1flh9) 14

[2.4.1 Script Type "script\_type"](#_ct8s52e1j6fi) 14

[2.4.2 Script "script"](#_8c4k8ulodq4b) 14

[2.4.3 Pipeline Version "pipeline\_version"](#_xl0lacaciv64) 14

[2.4.4 Platform/Environment "platform"](#_l0ww2sabj8mm) 14

[2.4.5 Script driver "script\_driver"](#_156qvxm089ib) 15

[2.4.6 Algorithmic tools and Software Prerequisites "software\_prerequisites"](#_1dwmov5yj08h) 15

[2.4.7 Domain Permissions "domain\_prerequisites"](#_vld9liea4sw2) 15

[2.4.7 Environmental parameters "env\_parameters"](#_de0d2tnrnh3b) 15

[2.5 Parametric domain "parametric\_domain"](#_5lzleogej36w) 16

[2.6 Input and output domains "io\_domain"](#_nuqiik9jcd55) 16

[2.6.1 Input Subdomain "input\_subdomain"](#_yf8t4njbtaux) 16

[2.6.2 Output Subdomain "output\_uri\_list"](#_8568kshxobnz) 16

[2.7 Error domain, acceptable range of variability "error\_domain"](#_7l3rerkvshsq) 17

[**3 Appendices**](#_gc4bbl10bg0z) **17**

[3.1 Appendix-I: BCO expanded view example](#_o91by4ieq6vd) 17

[3.2 Appendix-II: External reference database list](#_772h9twb860e) 20

[3.3 Appendix III - FDA draft guidance documents and Title 21 CFR Part 11](#_qo324mougkhk) 22

[3.3.1 FDA Draft Guidance Documents.](#_selnf3efr7wt) 22

[3.3.2 Title 21 CFR Part 11](#_54u419fe3438) 23

[3.4 Appendix IV - Compatibility](#_rbrqjba4ouoa) 23

[3.4.1 ISA for the experimental metadata](#_ju0avo5qnv24) 23

[3.5 Appendix V - Security](#_esgruchayox4) 24

[3.5.1 Blockchain](#_uo9s6kpni3t4) 24

[3.6 Appendix VI Data typing](#_9vyu6ez2pnkc) 24

[3.6.1 Primitive data types](#_o2w64d992k) 24

[3.6.2 Base BioCompute Type](#_eb6h39mree4u) 26

[3.6.3 Meaning associations](#_72cp8vuw5evw) 30

[3.6.4 Atomic and complex data](#_d2r9b41tidb0) 30

[3.6.5 Fundamental and derivable types](#_sdksonpj5841) 31

[3.6.5 Extensibility through inheritance and inclusion of data types](#_lkx0x0llfazi) 34

[3.6.6 Data lifecycle timeline](#_fryv5bmzp5d4) 35

# 

# 1 Introduction

BioCompute Objects will facilitate HTS (NGS) computational analysis information communication between the FDA and industry and academic research stakeholders. BioCompute is a paradigm and BioCompute Object (BCO) is an instance.

US Food and Drug Administration (FDA) and George Washington University (GW) have partnered to establish a framework for community-based standards development and harmonization of High-throughput Sequencing (HTS) computations and data formats. Standardized HTS computations and data formats will promote interoperability and ease the verification of bioinformatics protocols. To do this, a schema has been developed to represent instances of computational analysis as a BCO. A BCO includes: information about parameters and version of the executable programs in a pipeline, reference to input and output test data for verification of the pipeline, a usability domain, keywords, a list of authors along with other important metadata. As much as possible this description should be programmatically interoperable to public Cloud infrastructure.

Additional, non-normative, information on BCOs:

[https://hive.biochemistry.gwu.edu/htscsrs/BioCompute](https://hive.biochemistry.gwu.edu/htscsrs/biocompute)

<https://hive.biochemistry.gwu.edu/htscsrs/datatyping>

## 1.1 Mission of the BioCompute project

* BioCompute Objects will facilitate HTS (NGS) computational analysis information communication with FDA.
* Develop a community of stakeholders to create a versatile data harmonization framework that allows standardized definition of interoperable bioinformatics pipelines in a platform independent manner.
* Development of tools and facilities implementing data-typing, instantiation, deposition, storage, and distribution of validated BioCompute Objects through BioCompute database, in order to facilitate reproducible scientific research and regulatory submissions of data and computations.
* Facilitate portability of pipelines for execution on Public Cloud infrastructure.

## 1.2 Motivation

The unpredictability of actual physical, chemical, and biological experiments due to the multitude of environmental and procedural factors is well documented. What is systematically overlooked, however, is that computational biology algorithms are also affected by a multiplicity of parameters and have no lesser volatility. The complexities of computation protocols and interpretation of outcomes is only a part of the challenge: There are also virtually no standardized and industry-accepted metadata schemas for reporting the computational objects that record the parameters used for computations together with the results of computations. Thus, it is often impossible to reproduce the results of a previously performed computation due to missing information on parameters, versions, arguments, conditions, and procedures of application launch. BioCompute Object concept has been developed specifically to satisfy regulatory research needs for evaluation, validation, and verification of bioinformatics pipelines. Potential usability of BioCompute Objects within the larger scientific community can be increased through the creation of a BioCompute Object database initially consisting of records relevant to the U.S. Food and Drug Administration. A BioCompute Object database record will be similar to a GenBank record in form; the difference being that instead of describing a sequence, the BioCompute record will include information related to parameters, dependencies, usage, and other information related to specific computational instance. This mechanism will extend similar efforts and also serve as a collaborative ground to ensure interoperability between different platforms, industries, scientists, regulators, and other stakeholders interested in biocomputing.

For more information, see the project description on the [**FDA Extramural Research**](http://www.fda.gov/ScienceResearch/SpecialTopics/RegulatoryScience/ucm491893.htm) page.

### 1.2.1 Limitations of initial Effort

* At the initial stages of BioCompute development, we address the challenges of HTS (NGS) bioinformatics.

## 1.3 Audience for this document

* Users performing HTS (NGS) analysis with a regulatory science perspective
* HTS (NGS) bioinformatics platform developers
* HTS (NGS) related standard developers

## 1.4 Potential Stakeholders for the entire BioCompute project

* US Food and Drug Administration
* Medical product manufacturers and their suppliers
* Laboratories developing clinical testing protocols.
* Bioinformatics tool and platform developers who wish to operate in a regulatory environment, including cloud service (PaaS, IaaS, SaaS, FaaS) providers.
* Journals / Scientific Publishing / peer reviewing process
* NIH (particularly initiatives such as NCI/ITCR)
* Public cloud companies operating in the Life Sciences sector.

## 1.5 User stories

* A pharmaceutical company is submitting NGS data and the FDA conducts a reanalysis of the data. The reanalysis does not concur with the original results. It can be very lengthy and costly to figure out where the discrepancies are. [reproducibility and interpretation use cases]  
    
  Attaching a BioCompute Object with the initial submission would prevent most of the ambiguity surrounding the discrepancies.
* A research scientist performs computations and submits to journal for peer review. The reviewers determine the results are ambiguous and not easily interpretable, perhaps, due to the fact that not all the details are present. Exchanges like this greatly hinder the advance of the field of science. Submission of a BioCompute Object with the publication would allow the reviewers to determine the reproducibility of the computations much sooner.
* A regulatory decision has been made where computational analysis has been used as evidence. New data emerges after the product has been on the market over a year and the regulators can not reproduce the original environment with all unspecified configuration of tools and parameters of pipelines to reanalyze the initial submission data or replicate the initial conclusion. The product is on the market already. [recomputability use case]
* Authors and pharmaceutical scientists are unaware of how the regulatory industry is using workflows to analyze data. Openness and transparency are hindered by the lack of ability to communicate, not a lack of willingness. Scientific merit is compromised as a result of not having a common "language" for communicating computations. [collaboration use case]
* A bioinformatics platform provider can use BCO as part of its verification and validation process. A customer submits NGS data provided by a third party sequencing provider. The sequencing data is poor quality. Reproducible pipelines, validated and verified as a “BCO”, were used to demonstrate the the fault lies in the sequencing step and not the bioinformatics pipeline.
* Insert use case about introspection; reviewing the choices made in a data analysis workflow
* Insert use case about provenance, BCO as a history of what was computed
* Use case about "preserving value from large scale data analytics over time through selective re-computation"

## 1.6 How the BCO community will grow

This is community driven project and

Reader would benefit from a vision on how the BCO community will grow. Raja's discussion on post meeting working groups and interaction opportunities could invite the reader in. Public-Private partnerships to evolve best practices

# 2 Data type for BCOs

The fundamentals of data typing (type primitives, class inheritance, etc) that are used to define BioCompute Objects are described in detail in section Appendix VI. Developers of BCO enabled platforms should reference this section for details on how to support creation of BCO programmatically or manually.

JSON syntax paragraph

BioCompute datatypes are defined as aggregates of critical fields organized into a few domains: descriptive domain, identification and provenance domain, input and output domains, parametric domain, environmental domain, execution domain, prerequisite domain, usability domain, and error domain. At the moment of submission to the BioCompute database an instance of BCO type is created, populated with actual values compliant with the data type definitions and assigned a unique identifier. The object is then assigned a unique digital signature.

Three of the domains in a BioCompute Object become immutable upon assignment of the digital signature: 1) the Parametric Domain, 2) the Execution Domain and 3) the I/O Domain. Changing anything within these domains invalidates the verification and will break the digital signature. Optional fields are indicated by the “vital”: “Tue” flag, which is shown in the daya typing section below (Appendix VI).

## 2.1 Identification and Provenance fields

### 2.1.1 ID "id"

Namespaced (source/db/id) unique identifier of this BCO instance assigned by a BCO database engine. IDs should never be deprecated or reused.

"id": "https://hive.biochemistry.gwu.edu/bco\_db/obj.1270"

### 2.1.2 Name "name"

Name for the BCO. This public field should take free text value using common biological research terminology along with external reference linkage using identifiers whenever possible.

"name": "HCV1a [taxID:31646] ledipasvir [PubChem:67505836] resistance SNP [SO:0000694] detection"

### 2.1.3 Structured name "structured\_name"

Structured name is a templated computable text field designed to represent a BCO instance name in visible interfaces. This field can refer to other fields within the same or other objects. For example, a string like "HCV1a [taxid:$taxid] mutation detection" will be visualized as "HCV1a [taxid:31646] mutation detection" assuming the BCO has a field called taxID and value 31646.

HCV1a [taxID:$taxID] mutation detection = HCV1a [taxID:31646] mutation detection

### 2.1.4 Version "version"

Records the versioning of this BCO instance object. In BCO versioning, a change in the BCO which will affect the outcome of the computation should be deposited as a new BCO, not as a new version. In such cases the connection between the new object and the older one may or may not be (on author’s discretion) retained in the form of references. Changes that cannot affect the results of the computation can be incorporated into a new version of the existing BCO. Such changes might include name and title, comments, authors, validity dates, etc.

"version": "1.1"

### 2.1.5 Digital signature "digital\_signature"

A string, read-only value generated by a BCO database, protecting the object from internal or external alterations without proper validation. This value should not be submitted during deposition but can be read during downloading or transferring validated BCOs. The BCO server can provide API validating the signature vs BCO content.

"digital\_signature": "905d7fce3f3ac64c8ea86f058ca71658"

### 2.1.6 Verification status "verification\_status"

Describes the status of an object in the verification process. The 'unreviewed' flag indicates that the object has been submitted, but no further evaluation or verification has occurred. The 'in\_progress' flag indicates that verification is underway. The 'reviewed' flag indicates that the BCO has been verified and reviewed. The 'manual' flag indicates that the object has been manually verified. The 'suspended' flag indicates an object that was once valid is no longer considered valid. The 'error' flag indicates that an error was detected with the BCO.

"verification\_status": "in\_progress" OR "unreviewed" OR "reviewed" OR "published" OR "rejected"

vc

### 2.1.7 Publication status "publication\_status"

This is a choice field with three options. The 'draft' status indicates that an object is in draft form and is still being edited. The 'open\_access' status indicates that an object has been published and is freely available to anyone. The objects with the 'private' status have restrictions on who can view and access them. This is a way for researchers using restricted data or metadata to ensure the confidentiality is maintained. The permissions to the object will be defined by the database access control rules.

"publication\_status": "draft" OR "in\_progress" OR "private" OR "open\_access"

### 2.1.8 Authors "authors"

A list to hold author identifiers. We encourage the use of ORCIDs to record author information, as they allow for the author to curate their information after submission. For BCOs curated from publications, only the corresponding author name is provided, and in the format used by the publication. If an author does not have an ORCID then they can provide a name using free unicode text.

#### 2.1.8.1 Name "name"

Name of author. This is a free Unicode text field for authors to use who do not have an ORCID.

"name": "Charles Darwin"

#### 2.1.8.2 ORCID "orcid"

A unique identifier for researchers and scholars (<https://orcid.org/>). This is the method for recording author information. The identifier MUST start with “http://orcid.org/” (not https) and comply with the official [ORCID identifier structure](https://support.orcid.org/knowledgebase/articles/116780-structure-of-the-orcid-identifier).

"orcid": "http://orcid.org/0000-0002-1825-0097"

## 2.2 Usability domain "usability\_domain"

This field provides a space for the author to define the usability domain of the BCO. It is a multi-value field that holds a list of free text values. This field is to aid in search-ability and provide a specific description of the object. The usability domain along with keywords can help determine when and how the BCO can be used. Novel use of the BCO should (could?) result in the creation of a new entry with a new usability domain.

"Identify baseline single nucleotide polymorphisms SNPs [SO:0000694], insertions [SO:0000667], and deletions [SO:0000045] that correlate with reduced ledipasvir [PubChem:67505836] antiviral drug efficacy in Hepatitis C virus subtype 1 [taxID:31646]"

## 2.3 Description Domain "description\_domain"

Structured field for description of external references, the pipeline steps, and the relationship of I/O objects. Information in this domain is not used for computation. This domain is meant to capture information that is currently being provided in FDA submission in journal format. It is possible that in the future this field can be semi-automatically generated from the execution\_domain information.

### 2.3.1 Keywords "keywords"

This is a multi-value field to hold a list of keywords to aid in search-ability and description of the object.

"keywords": ["antiviral resistance", "SNP"]

### 2.3.2 External References "xref"

This field contains a list of the databases and/or ontology IDs that are cross-referenced in the BCO. The external references are used to provide more specificity in the information related to BCO entries. Cross-referenced resources need to be available in the public domain. The external references are stored in the form of prefixed identifiers (CURIEs). These CURIEs map directly to the URIs maintained by identifiers.org. See Appendix-II for a list of the CURIEs to URIs.

"xref": ["SO:0000694", "SO:0000667", "SO:0000045", "PubChem:67505836", "SO:0000048", "taxID:31646", "PMID:25123381", "PMID:26508693"]

#### 2.3.2.1 Extensions to External References:

FHIR

The external references includes an optional extension to FHIR resource where specified data elements can be extracted from EHR systems without compromising patient and providers’ information. This is because the portions being transferred contain no identifiable information about the patient. Instead there is a reference to the actual resource instance (via FHIR URI).

The FHIR URI leads to the report that will contain the date and time of the procedure, specimen details, and a set of observations in the diagnostic results as a narrative. The diagnostic report will also contain the information that pertains to the sequence of interest in the BCO such as the biological information of the sequence, the FHIR defined ID, the sequence code, base number of the coordinate system, reference of patient (not who it is but what it is about), reference used for specimen sequencing, method used for sequencing, organization responsible for the test result, the number of copies of the specified sequence, the reference sequence, chromosome which contains the genetic finding, start and end positions of the chromosomes, quality scores of the sequence, etc.

"FHIR\_extension":[

{

"FHIR\_resourceURL": "https://www.hl7.org/fhir/resourcelist.html",

"diagnostics": "DiagnosticsReport",

"identifier": "business identifier for the report",

"reference": "the procedural request"

},

{

"diagnostics": "sequence",

"sequence\_identifier": "unique ID for the specific sequence",

"datasetID": "ID for the dataset",

"type": "sequence type such as aa, dna or rna",

"reference\_specimen": "specimen used for sequencing"

}

]

### 2.3.3 Pipeline tools "pipeline\_steps"

An optional structured domain for recording the specifics of a pipeline. Each individual tool (or a well defined and reusable script) is represented as step, at the discretion of the author. Parallel processes are given the same step number. URIs are listed for the inputs and outputs for each tool.

#### 2.3.3.1 Tool Name "tool\_name"

Name for the specific tool. This field is a string (A-z, 0-1) and should be a single uniquely identifying word for the tool.

"tool\_name": "HIVE-hexagon"

#### 2.3.3.2 Tool Description "tool\_desc"

A free text field for describing the specific use/purpose of the tool

#### 2.3.3.3 Tool Version "tool\_version"

The version assigned to the the instance of the tool used

#### 2.3.3.4 Tool Requirements or Packages "tool\_package"

The is a list of text values to indicate any packages or prerequisites for running the tool used.

"tool\_package": ["Python version 2.6", "megablast version 1.2"]

#### 2.3.3.5 Step Number "step\_number"

This is an integer value representing the position of the tool in the one dimensional representation of the pipeline. Parallel computations are assigned the same number.

"step\_number": "1"

#### 2.3.3.6 Input URI List "input\_uri\_list"

Each tool lists the URI of the input files for that tool.

"input\_uri\_list": ["http://www.ncbi.nlm.nih.gov/nuccore/22129792", "http://www.ncbi.nlm.nih.gov/nuccore/5420376"]

#### 2.3.3.7 Output URI List "output\_uri\_list"

Each tool lists the URI of the output files for that tool.

"output\_uri\_list": ["hive://data/514769/allCount-aligned.csv"]

## 2.4 Execution domain "execution\_domain"

The fields required for execution of the BCO have been encapsulated together in order to clearly separate information needed for deployment, software configuration and running applications in a dependent environment. One byproduct of an accurate BCO definition is facilitation of reproducibility as defined by the *Oxford English Dictionary* as "the extent to which consistent results are obtained when produced repeatedly."

### 2.4.1 Script Type "script\_type"

This is a choice field to indicate whether the "script" to execute the BioCompute Object is a reference to an external file (URI) or text in the "script" field.

"script\_type": "URI" OR "text"

### 

### 2.4.2 Script "script"

The internal or external reference to a script object that was used to perform computations for this BCO instance. This may be a reference to Galaxy Project or SB-genomics pipeline, a Common Workflow Language (CWL) object in GitHub, a High-performance Integrated Virtual Environment (HIVE) computational service or any other type of script.

"script": "https://hive/workflows/antiviral\_resistance\_detection\_hive.py"

### 2.4.3 Pipeline Version "pipeline\_version"

This field records the version of the pipeline implementation.

"pipeline\_version": "2.0"

### 2.4.4 Platform/Environment "platform"

The multi-value reference to a particular deployment of an existing platform where this BCO can be reproduced. A platform can be a bioinformatic platform such as Galaxy or HIVE or it can be a software package such as CASAVA or apps that includes multiple algorithms and software.

"platform": "HIVE"

### 2.4.5 Script driver "script\_driver"

The reference to an executable that can be launched in order to perform a sequence of commands described in the script (see above) in order to run the pipeline. For example, if the pipeline is driven by a HIVE script, the script driver is the "hive" execution engine. For CWL based scripts, the cwl-runner tool is the driver which is capable of interpreting the commands in the script. Another very general script driver commonly used in Linux based operating systems is "shell" and the type of scripts it can run are operating system shell scripts. The combination of script driver and script is a capability to run a particular sequence of computational steps in order to produce BCO outputs given the inputs and parameters.

It is noteworthy to mention that scripts and script drivers by themselves can be objects. These objects can exist in internal (BCO) or external databases and be publically or privately accessible.

"driver": "shell"

### 2.4.6 Algorithmic tools and Software Prerequisites "software\_prerequisites"

An optional multi-value field listing the minimal necessary prerequisites, library, tool versions needed to successfully run the script to produce BCO.

"software\_prerequisites": [

{"name":"HIVE\_hexagon","version":"1.3"},

{"name":"HIVE\_heptagon","version":"1.3"}

]

### 2.4.7 Domain Permissions "domain\_prerequisites"

An optional multi-value field listing the minimal necessary domain specific external data source access in order to successfully run the script to produce BCO.

"domain\_prerequisites": [

{"url":"protocol://domain:port/application/path",

"name":"generic name"},

{"url":"ftp://:22/",

"name": "access to ftp"},

{"url":http://eutils.ncbi.nlm.nih.gov/entrez/eutils",

"name":"access to e-utils"}

]

### 2.4.7 Environmental parameters "env\_parameters"

Multi-value additional key value pairs useful to configure the execution environment on the target platform. For example, one might specify the number of compute cores, or available memory use of the script.

"env\_parameters": [{"OSTYPE":"linux"},{"QPRIDE\_BIN":"~qpride/bin"}]

## 2.5 Parametric domain "parametric\_domain"

This represents the list of parameters customizing the computational flow which can affect the output of the calculations. These fields are custom to each type of analysis and are tied to a particular pipeline implementation. All BCOs should inherit from the fundamental BioCompute data-type and as such inherit all of the core fields described in document. Specific BioCompute types introduce specific fields designed to customize the use of pipelines for a particular use pattern. Please refer to documentation of individual scripts and specific BCO descriptions for details.

"parametric\_domain": {

"heptagon\_freq\_cutoff": "0.10",

"hexagon\_minimum\_match\_len": "66"

}

## 2.6 Input and output domains "io\_domain"

This represents the list of global input and output files created by the computational workflow. These fields are pointers to objects that can reside in the system performing the computation or any other accessible system. Just like the fields of parametric domain, these fields are custom to every specific BCO implementation and can refer to named input output arguments of underlying pipelines. Please refer to documentation of individual scripts and specific BCO descriptions for further details.

### 2.6.1 Input Subdomain "input\_subdomain"

This field records the references and input files for the entire pipeline. Each type of input file is listed under a key for that type. The file types are specified when the BCO type is created.

"input\_subdomain": {

"HCV\_Genome\_to\_detect\_variations":[

"http://www.ncbi.nlm.nih.gov/nuccore/22129792",

"http://www.ncbi.nlm.nih.gov/nuccore/5420376"

],

"read\_files": [

"hive://nuc-read/514683",

"hive://nuc-read/514682"

]

}

### 2.6.2 Output Subdomain "output\_uri\_list"

This field records the outputs for the entire pipeline. Each file should be an object with a key, and a title, URI, and [mime-type](https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types) value.

"output\_subdomain": {

"hit\_list$Obj\_ID": {

"title": "hit list",

"uri": "hive://data/514769/dnaAccessionBased.csv",

"mime-type" : "text/csv"

},

"mutation\_profile": {

"title": "mutation profile",

"uri": "hive://data/514801/SNPProfile\*.csv",

"mime-type": "text/csv"

}

}

## 2.7 Error domain, acceptable range of variability "error\_domain"

The error domain consists of two subdomains: empirical and algorithmic.

The empirical error subdomain contains the limits of detectability, false positives, false negatives, statistical confidence of outcomes, etc. This can be measured by running the algorithm on multiple data samples of the usability domain or in carefully designed in-silico spiked data. For example a set of spiked, well characterized samples can be run through the algorithm to determine the false positives, negatives and limits of detection.

The algorithmic subdomain is descriptive of errors that originated by fuzziness of the algorithms, driven by stochastic processes, in dynamically parallelized multi-threaded executions, or in machine learning methodologies where the state of the machine can affect the outcome. This can be measured in repeatability experiments of multiple runs or using some rigorous mathematical modeling of the accumulated errors. For example: Bootstrapping is frequently used with stochastic simulation based algorithms to accumulate sets of outcomes and estimate statistically significant variability for the results.

"[error\_domain](#_c5nzws6tovyl)": {

"empirical\_error": {"false negative alignment hits": "<0.0010"},

"algorithmic\_error":{"false positive mutation calls discovery": "<0.0005"}

}

# 3 Appendices

## 3.1 Appendix-I: BCO expanded view example

{

"id": "obj.1270",

"name": "HCV1a [taxID:31646] ledipasvir [PubChem:67505836] resistance SNP [SO:0000694] detection",

"version": "1.1",

"createdby": "hadley\_king@gwmail.gwu.edu",

"created": "Jan 24, 2017 09:40:17",

"modified": "Mar 27, 2017 13:27:02",

"digital\_signature": "905d7fce3f3ac64c8ea86f058ca71658",

"verification\_status": "unreviewed",

"publication\_status": "draft",

"usability\_domain": [

"Identify baseline single nucleotide polymorphisms SNPs [SO:0000694], insertions [SO:0000667], and deletions [SO:0000045] that correlate with reduced ledipasvir [PubChem:67505836] antiviral drug efficacy in Hepatitis C virus subtype 1", "Identify treatment emergent amino acid substitutions [SO:0000048] that correlate with antiviral drug treatment failure",

"Determine whether the treatment emergent amino acid substitutions [SO:0000048] identified correlate with treatment failure involving other drugs against the same virus",

"GitHub CWL example: https://github.com/mr-c/hive-cwl-examples/blob/master/workflow/hive-viral-mutation-detection.cwl#L20"

],

"authors": [

{

"name": "Josiah Carberry"

},

{

"orcid": "http://orcid.org/0000-0002-1825-0097"

}

],

"description\_domain": {

"keywords": [

"Antiviral resistance",

"SNP",

"Ledipasvir"

"HCV1a",

"Amino acid substitution"

],

"xref": [

"SO:0000694",

"SO:0000667",

"SO:0000045",

"PubChem:67505836",

"SO:0000048",

"taxID:31646",

"PMID:25123381",

"PMID:26508693"

],

"pipeline\_steps": [

{

"tool\_name": "HIVE-hexagon",

"tool\_desc": "Alignment of reads to a set of references",

"tool\_version": "1.3",

"tool\_package": "",

"step\_number": "1",

"input\_uri\_list": [

"http://www.ncbi.nlm.nih.gov/nuccore/22129792",

"http://www.ncbi.nlm.nih.gov/nuccore/5420376",

"http://www.ncbi.nlm.nih.gov/nuccore/13122261",

"http://www.ncbi.nlm.nih.gov/nuccore/386646758",

"http://www.ncbi.nlm.nih.gov/nuccore/295311559",

"hive://nuc-read/514683",

"hive://nuc-read/514682"

],

"output\_uri\_list": [

"hive://data/514769/allCount-aligned.csv"

]

},

{

"tool\_name": "HIVE-heptagon",

"tool\_desc": "variant calling",

"tool\_version": "1.3",

"tool\_package": "",

"step\_number": "2",

"input\_uri\_list": [

"hive://data/514769/dnaAccessionBased.csv"

],

"output\_uri\_list": [

"hive://data/514801/SNPProfile.csv",

"hive://data/14769/allCount-aligned.csv"

]

}

]

},

"execution\_domain": {

"script\_type": "URI",

"script": "https://hive/workflows/antiviral\_resistance\_detection\_hive.sh",

"pipeline\_version": "2.0",

"platform": "hive",

"driver": "shell",

"software\_prerequisites": [

{"name":"HIVE-hexagon","version":"1.3"},

{"name":"HIVE-heptagon","version":"1.3"}

],

"access\_prerequisites": [

{

"url":"protocol://domain:port/application/path",

"name":"generic name"

},

{

"url":"ftp://:22/",

"name": "access to ftp"

},

{

"url":"http://eutils.ncbi.nlm.nih.gov/entrez/eutils/",

"name":"access to e-utils"

}

],

"env\_parameters": [

{"OSTYPE":"linux"},

{"QPRIDE\_BIN":"~qpride/bin"}

]

},

"parametric\_domain": {

"heptagon\_divergence\_threshold\_percent": "30",

"hexagon\_minimum\_coverage": "0.15",

"hexagon\_seed": "14",

"heptagon\_freq\_cutoff": "0.10",

"hexagon\_minimum\_match\_len": "66"

},

"io\_domain": {

"input\_subdomain": {

"HCV\_Genome\_to\_detect\_variations":[

"http://www.ncbi.nlm.nih.gov/nuccore/22129792",

"http://www.ncbi.nlm.nih.gov/nuccore/5420376"

],

"read\_files": [

"hive://nuc-read/514683",

"hive://nuc-read/514682"

]

},

"output\_subdomain": {

"hit\_list": {

"title": "hit list",

"uri": "hive://data/514769/dnaAccessionBased.csv",

"mime-type" : "text/csv"

},

"mutation\_profile": {

"title": "mutation profile",

"uri": "hive://data/514801/SNPProfile\*.csv",

"mime-type": "text/csv"

}

}

},

"error\_domain": {

"empirical\_error": {

"false negative alignment hits": "<0.0010"

},

"algorithmic\_error": {

"false positive mutation calls discovery": "<0.0005"

}

}

}

## 3.2 Appendix-II: External reference database list

This list contains the databases that are currently being used in our BCOs. We use the CURIEs that map to URIs maintained by *identifiers.org*.

“*Identifiers.org* is an established resolving system the enables the referencing of data for the scientific community, with a current focus on the Life Sciences domain… *Identifiers.org* provides direct access to the identified data using one selected physical location (or resource). Where multiple physical locations are recorded in the [registry](http://identifiers.org/registry) the most stable one is selected for resolution. This allows the location independent referencing (and resolution if required) of data records.”  
  
Gene Ontology Consortium : GO  
http://amigo.geneontology.org/amigo/medial\_search?q=%s  
  
Nucleotide : NucCore  
http://www.ncbi.nlm.nih.gov/nuccore/?term=%s   
  
Disease-Ontology, standardized ontology for human disease : DO  
http://disease-ontology.org/  
  
International Classification of Disease (Tenth Edition) : ICD  
http://www.icd10data.com/Search.aspx?search=%s  
  
Sequence Read Archive : SRA  
http://www.ncbi.nlm.nih.gov/sra/?term=%s  
  
Online Mendelian Inheritance in Man : OMIM  
http://www.ncbi.nlm.nih.gov/omim/?term=%s  
  
Open Researcher and Contributor ID : ORCID  
http://orcid.org/%s  
  
Sequence Ontology : SO  
http://www.sequenceontology.org/browser/current\_svn/term/%s  
  
Taxonomy : taxID  
http://www.ncbi.nlm.nih.gov/taxonomy/?term=%s  
  
PubChem Compound : PubChem  
https://pubchem.ncbi.nlm.nih.gov/compound/%s  
  
Uberon : UBERON  
http://www.ebi.ac.uk/ols/search?q=%S&submit=Search+Uberon&ontology=uberon

ClinVar : ClinVar  
https://www.ncbi.nlm.nih.gov/clinvar/?term=

MedGen : MedGen  
https://www.ncbi.nlm.nih.gov/medgen/?term=

dbSNP : dbSNP  
https://www.ncbi.nlm.nih.gov/snp/?term=

HUGO Gene Nomenclature Committee : HGNC  
http://www.genenames.org/cgi-bin/search?search\_type=all&search=

Database of Genotypes and Phenotypes : dbGaP

https://www.ncbi.nlm.nih.gov/gap/?term=

UniProt/SwissProt : UniProt

http://www.uniprot.org/uniprot/?query=

Variation Ontology : VariO

http://variationontology.org/VariOtator.php

PubMed : PMID

https://www.ncbi.nlm.nih.gov/pubmed/?term=

NCBI-GEO:GSE

https://www.ncbi.nlm.nih.gov/gds/?term=

FHIR : Diagnostic Report-Genetics Profile - reference value

https://www.hl7.org/fhir/diagnosticreport.html

## 3.3 Appendix III - FDA draft guidance documents and Title 21 CFR Part 11

### 3.3.1 FDA Draft Guidance Documents.

The FDA has proposed draft guidance documents proposing methods to streamline oversight of HTS (NGS)-based tests. The documents are open for public input and we strongly encourage BioCompute stakeholders to contribute their critical considerations. Significant expertise of BioCompute stakeholders can serve as a forum to efficiently develop a coherent vision, evaluate validity and utility of these guidance documents.   
  
This guidance document is an early attempt to signify the importance of computation and data provenance and is congruent to larger BioCompute effort and contains information from BCO domains. Some of the suggested computational aspects described in the draft guidance proposal are summarized:

* All the software used in the workflow requires detailed documentation on versioning, source and modification.
* Documentation is required for the software versions and attribution, reference sequence assembly and components needed to compile, install and run (remotely or locally) the bioinformatics pipeline.
* Documentation and specificity of databases which were used internally or externally should be included.

Relevant draft guidance document links:

[DRAFT Guidance: Use of Standards in FDA Regulatory Oversight of Next Generation Sequencing (NGS)-Based In Vitro Diagnostics (IVDs) Used for Diagnosing Germline Diseases](http://www.fda.gov/downloads/MedicalDevices/DeviceRegulationandGuidance/GuidanceDocuments/UCM509838.pdf)

[Use of Public Human Genetic Variant Databases to Support Clinical Validity for Next Generation Sequencing (NGS)-Based In Vitro Diagnostics](https://www.fda.gov/downloads/MedicalDevices/DeviceRegulationandGuidance/GuidanceDocuments/UCM509837.pdf)

### 3.3.2 Title 21 CFR Part 11

*Code of Federal Regulations Title 21 Part 11: Electronic Records - Electronic Signatures*

BioCompute project is being developed with Title 21 CFR Part 11 compliance in mind. The digital signatures incorporated into the format will provide the basis for provenance of BioCompute Object integrity using NIST proposed encryption algorithms. Execution domain and parametric domain (that have a potential impact on a result of computation) and identity domain will be used to create hash values and digital signature encryption keys which later can be used for computer or human validation of transmitted objects.   
  
Discussions are now taking place to consider relevance of BioCompute Objects with relation to Title 21 CFR part 11. We encourage continuous input from BioCompute stakeholders on this subject now and while the concept is becoming more mature and more widely accepted by scientific and regulatory communities.

Relevant document link:

[Part 11: Electronic Records](http://www.fda.gov/RegulatoryInformation/Guidances/ucm125067.htm)

## 3.4 Appendix IV - Compatibility

### 3.4.1 ISA for the experimental metadata

Compatibility with ISA-Tab file format. MORE TEXT NEEDED.

ISA is a metadata framework to manage an increasingly diverse set of life science, environmental and biomedical experiments that employ one or a combination of technologies. Built around the **Investigation** (the project context), **Study** (a unit of research) and **Assay** (analytical measurements) concepts, ISA helps to provide rich descriptions of experimental metadata (i.e. sample characteristics, technology and measurement types, sample-to-data relationships) so that the resulting data and discoveries are reproducible and reusable. The ISA Model and Serialization Specifications define an Abstract Model of the metadata framework that has been implemented in two format specifications, ISA-Tab and ISA-JSON (<http://isa-tools.org/format/specification>), both of which have supporting tools and services associated with them, including by a programmable Python AP ([http://isa-tools.org](http://isa-tools.org/format/specification)) and a varied user community and contributors (<http://www.isacommons.org>). ISA focuses on structuring experimental metadata; raw and derived data files, codes, workflows etc are considered as external file that are referenced. An example, along its complementarity with other models and a computational workflow is illustrated in this paper, which shows how to explicitly declare elements of experimental design, variables, and findings: <http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0127612>

## 3.5 Appendix V - Security

### 3.5.1 Blockchain

Use of Blockchain technology to record events….ADD TEXT HERE

## 3.6 Appendix VI Data typing

The conceptual schema for BCO creation is built on top of two layers: the data definition layer and the BCO layer. The first layer is where all fundamental data types are defined. Complex types are composed of multiple atomic or complex types, like a character string. Using these principles one can construct a datum that has the ability to represent any level of complexity that is needed. A BCO is essentially a federation of other objects.

### 3.6.1 Primitive data types

When defining a field in a data type, one can place any number of constraints on the data and the field will be accepted as valid. So, if a data type field was being constructed for holding DNA sequencing information, one could restrain the type of characters that field would accept. This further refinement would ensure that only the characters used for representing nucleic acids would be accepted as input in this field (e.g. A, T, C, G). A list of the primitive types used in BCO data typing is below.

"primitives" : {

"description": "primitives branch",

"bool" : {

"\_type": "core",

"\_limit" : {

"true" : 1,

"false" : 0

}

},

"int" : {

"\_type" : "core"

},

"uint" : {

"\_type" : "int",

"\_limit" : {

"eval-c/c++" : "$val>=0"

}

},

"percent" : {

"\_type" : "uint",

"\_limit" : {

"eval-c/c++" : "$val<=100 && $val>=0"

}

},

"string" : {

"\_type" : "core"

},

"text" : {

"\_type" : "core"

},

"memory" : {

"\_type" : "uint"

},

"cmdline" : {

"\_type" : "text"

},

"url" : {

"\_type" : "text"

},

"file" : {

"\_type" : "string"

},

"date" : {

"\_type" : "core",

"\_default" : "$datenow()"

},

"time" : {

"\_type" : "core",

"\_default" : "$timenow()"

},

"datetime" : {

"\_type" : "core",

"date": {

"\_type": "date",

"\_default" : "$datenow()"

},

"time": {

"\_type": "time",

"\_default" : "$timenow()"

}

},

"timespan" : {

"\_type" : "core",

"start" : {

"\_type" : "datetime"

},

"end" : {

"\_type" : "datetime"

}

},

"ref" : {

"\_type" : "core",

"source" : {

"\_type" : "string"

},

"db" : {

"\_type" : "string"

},

"id" : {

"\_type" : "string",

"\_vital": true

}

},

"password" : {

"\_type" : "string"

},

"email" : {

"\_type" : "string"

},

"keyval" : {

"\_type" : "core",

"key" : {

"\_type" : "string"

},

"value" : {

"\_type" : "string",

"\_plural" : true

}

}

}

### 3.6.2 Base BioCompute Type

The second layer is constructed with objects from first layer, producing a derived data type called the "base BioCompute type". Extending the same principles that allowed us to construct a string representing a DNA sequence from the primitive character type, one can construct a type definition that is the absolute minimum fields necessary to create a BCO. By taking the primitive BCO type and adding parametric and metadata fields unique to a particular instance, a BCO can be created. Below it the type definition for “BioCompute\_base\_type”:

{

"BioCompute\_base\_type": {

"\_type" : "type",

"\_id" : "$newid()",

"\_field": {

"id": {

"\_type" :"string",

"title" : "Identifier of the object",

"\_public" : true,

"\_write" : false

},

"name": {

"\_type" :"string",

"title" : "public searchable name for BioCompute Objects",

"\_public" : true

},

"structured\_name": {

"\_type" :"string",

"title" : "public searchable name for BioCompute Objects",

"\_public" : true,

"\_vital" : false

},

"version": {

"\_type" :"string",

"title" : "version for BioCompute Object",

"\_public" : true

},

"digital\_signature": {

"\_type" : "string",

"title" : "digital signature of the BioCompute Object",

"\_write":false,

"\_vital" : false

},

"verification\_status": {

"\_type" : "string",

"\_limit": {

"choice ": ["in\_progress", "unreviewed", "reviewed" , "published", "rejected" ]

},

"title" : "the current verification status of the BioCompute Object",

"\_default" : "in\_progress"

},

"publications\_status": {

"\_type" : "string",

"\_limit": {

"choice ": ["draft", "in\_progress", "published", "embargoed" ]

},

"title" : "the current publication status of the BioCompute Object",

"\_default" : "draft"

},

"authors": {

"\_type" : "core",

"name": {

"\_type" : "string",

"\_hidden" : "eval-js: orcid===undefined"

},

"orcid": {

"\_type" : "string",

"\_hidden" : "eval-js: name===undefined"

},

"\_plural": true

},

"usability\_domain": {

"\_type" : " string",

"\_plural": true,

"descr" : "text from biospec"

},

"description domain": {

"keywords": {

"\_type": "keyval",

"\_plural": true,

"vital": false

},

"xref": {

"\_type": "xref",

"\_plural": true,

"vital": false

},

"pipeline\_steps": {

"tool\_name" : {

"\_type": "string",

"descr" : "this is a recognized name of the software tool"

},

"tool\_desc" : {

"\_type": "text"

},

"tool\_version": {

"\_type": "string"

},

"tool\_package" : {

"\_type": "string",

"\_plural": true

},

"step\_number": {

"\_type": "uint",

"\_limit": {

"eval-js": "$step\_number >= 1"

}

},

"input\_uri\_list" : {

"\_type" : "url",

" \_vital": false

},

"output\_uri\_list" : {

"\_type" : "url",

" \_vital": false

},

"\_plural": true

},

"\_vital": false

},

"execution\_domain": {

"script\_type": {

"\_type":"string",

"\_limit": {

"choice": ["uri","text" ]

}

},

"script": {

"\_type": "$script\_type"

},

"script\_driver": {

"\_type": "string"

},

"pipeline\_version": {

"\_type": "string"

},

"platform": {

"\_type": "string"

},

"software\_prerequisites": {

"name": {

"\_type": "string"

},

"version": {

"\_type": "string"

},

"url": {

"\_type": "url",

"\_vital": false

},

"\_plural": true

},

"access\_prerequisites": {

"name": {

"\_type": "string"

},

"url": {

"\_type": "url",

"\_plural" : true

}

},

"environmental\_parameters": {

"\_type": "keyval",

"\_plural" : true

},

"\_vital" : true

},

"parametric\_domain": {

"descr" : "all fields in this domain should be defined in inheriting BioCompute subtypes"

},

"io\_domain": {

"input\_subdomain": {

"descr" : "all fields in this subdomain are specific BioCompute specific and should be defined in inheriting BioCompute subtypes"

},

"output\_subdomain":{

"descr" : "all fields in this subdomain are specific BioCompute specific and should be defined in inheriting BioCompute subtypes"

}

}

},

"\_inherit" : "base\_database\_object",

"name" : "BioCompute\_base\_type",

"title" : "Base type for all BioCompute Objects",

"descr" : "all BioComputes must inherit from this type in order to be compliant with BioCompute framework"

}

}

### 3.6.3 Meaning associations

All data field values stored in the data format have an associated file name and/or absolute or relative field value location. The association and assignment of meaning to a particular data value can be ontological – derived from a name-value pair, or topological – derived from a location-value pair.

*As an example, in a name-value pair, the name of the field is explicitly specified. Examples of name-value pairing are: a JSON object; the mapping from ids to nodes in an XML document; or the association between the column names and columns of a CSV table. In a location-value pair, the name of the field is not specified but assumed, according to the order and positioning of the value assumes its meaning. An example of location-value pairing is a FASTQ file, where sequence and quality lines do not have explicit names, but their relative locations are used to identify the data line type.*

### 3.6.4 Atomic and complex data

A data value can be atomic, not decomposable into smaller sets without the loss of meaning, or complex, containing other simpler values or topologies. Data structures of arbitrary complexity can be created using substructures and hybrids of name-value and location-value aggregations. Some fields may be declared parent fields of others and the field name value pairs can be allocated along the hierarchy of field types. It is important to note that atomicity of the field values here does not directly imply independent scientific interpretability of the atomic values: it only implies that such information cannot be fragmented into smaller more fundamental types.

*For example, the value for a person's age is an atomic value and cannot be decomposed. However, a person's identity is a complex data which contains first name, last name, age, social security number, passport number and perhaps other simpler values which have a meaning in of their own.*

EXAMPLES SUCH AS THE ONE ABOVE ARE NEEDED FOR ALL OF THE ITEMS BELOW. WE REQUEST OTHERS TO INCLUDE SUCH EXAMPLES IF THEY CAN.

### 3.6.5 Fundamental and derivable types

Primitive field types, such as numbers, strings, bit-fields, and uncharacterized blobs, are generally the most frequently used: this fact is reflected in computer science where these four types are chosen as fundamental units of data representation. A data type by itself does not carry physical interpretation; this interpretation comes from the value associations mentioned above. Any other field types (sequences, alignments, etc.), atomic or complex, can be constructed by using the fundamental types and adjusting their interpretive significance. Computable data types, such as references and formulas, are also useful for creating operational information infrastructures; such fields do not carry a value themselves but link to another value, in the same or another object, directly or through the usage of a transformation formula.

In the person’s identity example mentioned in the previous section the atomic subfields first name, last name, age, etc. are primitive data types such as strings and numbers.

Data typing is the process of creating a derived data type as a collection of primitive field types and previously defined complex types. During this process, field attributes such as name, type, constraints, default value are specified for each of the fields.

*To construct the type "Identity" we define a collection of the fundamental fields that comprise this derived type.* ***Identity:***

* *First name is a free text string.*
* *Last name is a free text string.*
* *Age is a single whole number.*

*Identity: {Charles, Darwin, 208}*

Derived/computed fields

Derived/computed data types are produced during the output of the object instance and cannot be entered during instantiation of the object into the database. This fields are just the reproducible outcome of other field values through some kind of derivation mechanisms. For example: a taxonomy identifier is non-redundantly unique enough to recover different names, synonyms and relations for a taxonomic unit. However a derived field "name" can be a useful informative construct derived and outputted every time where taxonomic identification is necessary.

*TaxID: 9606*

Virtual fields

Field values can be explicitly specified in an object instance or computable by other means. For example, the metadata object describing a file can have a field named "file-size" reflective of the actual file size in storage. The value of that field cannot be explicitly specified or computed from other values in the same object. Such virtual fields are descriptive and can be used for validation of the object itself and for definition of further constraints.

Constraints

Field values can have explicit and implicit constraints determining the universe of possible values for that particular field. Implicit constraints may be derived from the actual data types where, for example, numeric fields can have only numeric values and strings can have only characters of a particular alphabet. Explicit constraints are those specifically targeting a particular field in a data format, further limiting the value to a smaller set of possible values. Implicit and explicit constraints can be validated either by ensuring their syntax conformance or by evaluation of validation expression criteria. Correlation constraints can be defined on a field as well, demanding a condition when constraints on one field depend on the value(s) of other fields. There can be a scale of constraint rigorousness depending on the impact due to violation of such constraints. Soft constraint nonconformity can signify potential devaluation of a particular field value pair while hard constraint violation in a single field value pair would invalidate the entirety of the data as a whole.

Existence of static and dynamic constraints is necessitated by the changing nature of interdependencies of local versus global information. Static constraints are those whose validation expression depends only on the data itself while dynamic constraints depend on external or dynamically changeable information from local or global sources. An example of an explicit, dynamic, and correlated constraint-carrying field type is XREF (cross-reference) for which both the ID and the cross-referenced source must be included and the validity of the field can be evaluated only by inclusion of potentially changeable dynamic content of the external source.

It is important to note that constraints may limit the universe of possible values to discrete or continuous subsets of a finite or infinite enumerable list of values. Actual implementations of constraints may vary from a form of numerical open (]min,max[, closed ([min,max] or mixed ([min,max[, ]min,max]) ranges, lists of ranges, lists of strings and possible values to mathematical expressions producing true or false Boolean values based on the computation of a formula.

Single and multi-value

An additional level of empowering complexity can be achieved if fields are allowed to have multiple values. Unlike single-value fields, a multi-value field carries multiple values associated with a single name-value or location-value assignation. Using this paradigm, one can introduce notions of a list and an array into data format containers. The difference between these types is that an array’s children elements are topologically bound to their location in a row and therefore, are related to each other. In the list however, children elements of the same or of a different kind are simply a collection of non-coupled values.

Public and private fields

Security and access control rights play an important role when considering data access patterns in modern collaborative information systems. Data can be shared in such environments for viewing and modifications. One may want to share only a subset of fields containing certain descriptive values broadcasting information about the existence of a particular metadata instance through a search and browsing engine. Thus, the researcher or collaborator who is interested in having access to your data may then ask for more complete (read or write) access to the dataset that was found through browsing its public properties. To satisfy this need, we can have an attribute called public for a field that declares it to be available for searching and browsing by other users.

Uniqueness and key values

The importance of any particular field’s role in data format definition can vary within a local or global context. Some fields are unique within the given instance (object) of the data file and some are required to be unique within the global scope of such objects in some database. Such fields can be used as unique key identifiers of a whole data file within a known context.

Incompleteness

Data formats can support both mandatory and optional fields. A valid data file must include all its mandatory fields without exception. Such fields are typically essential in representation of the underlying information and their absence can devalue or introduce ambiguities into interpretation of other fields. Optional fields are those carrying non-critical values; absence of those values does not devalue the interpretation of other fields. There are two possible approaches to optional fields: default value and undefined value. An empty optional field can be treated as having an agreed upon predefined default value or can simply call the value undefined or NULL which must be interpreted as non-equal to any other value, even to another undefined. Any operation performed with undefined values must also be assigned the value of undefined.

Arbitrary key-value pairs

Certain file formats allow incorporation of arbitrary key value pairs without specific designation with the objective of adopting unstructured, descriptive, or information that may be useful in the future. Although the ultimate goal of such arbitrary fields is to maintain some level of extensibility in formats, those arbitrary fields usually end up containing a significant amount of ill structured and unverifiable, non-canonical information. By providing other means to extensibility, we strongly discourage using arbitrary key-value pairs for anything other than purely descriptive, non-critical information.

### 3.6.5 Extensibility through inheritance and inclusion of data types

It is of the utmost importance to generate extensible metadata formats capable of providing the basis for more complex new types. There are two proposed ways to extend a data format: inheritance and inclusion.

The concept of inheritance assumes that a more complex data type inherits all the field value pairs from another, simpler data type and extends the content only with additional field value pairs or customizes (redefines some characteristics) of existing fields. The concept of inclusion assumes that a particular field of an object is of a previously declared complex type and that it contains all the fields of a simpler data type. A single data type can inherit from multiple data types and can include multiple data types multiple times.

Using these two paradigms, one can design a number of layered standard objects based on predefined objects and extend their functionality with specific fields. For example: imagine a metadata object of type bio-sample which has a predefined fundamental description applied to a generalized sample. This object can have its properties repeatedly inherited to create a human-sample object with increasingly specific information about particularities of that sample description.

The two proposed extensibility models allow avoiding the overuse of optional field attributes that are present in conglomerated flat data type designs. Instead of designing wide and flat data types with all possible fields for different use-cases, one may choose to design more targeted types with specifically mandated fields inside.

*For example, having an optional tissue-location field in all biological-sample objects might lead to sparse population of the field as it will be unpopulated for all environmental, metagenomic, bacterial, and viral samples where the notion of a tissue is irrelevant. However, designing an inherited animal-sample data type can have a mandatory tissue location field for instances when it is important to know from which part of the animal a particular sample was collected.*

The power of the inheritance and inclusion methods to extend and implement new data types is evident when one considers the need to create new subtypes or a branch of existing types after the initial data -type structure is established. This step can be accomplished without modification of existing database objects by defining the new intermediates within the framework of the pre-defined metadata type hierarchy.

The relationship implemented by inheritance subtyping is a is-a relationship. For example, the type "fish" can have three subtypes "eel", "shark" and "salmon". Each subtype is a variety of the "fish" supertype and inherits all "fish" characteristics but has some specific differences.

### 3.6.6 Data lifecycle timeline

Data objects are typically records stored in an information system: a file system or a database. The life of such a record starts at the moment of metadata file submission. Typical preprocessing steps include: files being parsed and validated regarding their conformity with data standards, application of quality control processes and designation of appropriate permissions for later use. Depending on the size and complexity of the data, as well as the load on the data processing subsystems, this period may take seconds to days for NGS data. After this initial preprocessing stage, objects become visible to the owner/submitter of the information.

The user can then specify the validity start time before which the object is not to be accessed by anyone other than the user. This feature is useful for providing pre-publication delays or time fixed processing procedures. The user can also specify the validity end period after which the object is not to be used by anyone other than the owner of the record.

Optional soft and hard expiration periods can be set. These properties signify when the object should become "expired" from the database and should be treated as "deleted," and when the record is actually deleted from the database, respectively. The timespan between these two time periods is the potential recovery period; during this period the deletion can be reverted by manual or electronic inquiry from the user to the DB administrator/manager.

Another important milestone of the data existence is set by FDA’s mandate to maintain an archival copy of any review data used to make regulatory decisions. This copy does not necessarily reside in any easily accessible database or file system and is managed by a different set of regulations, the description of which lies outside the scope of this document.