GitHub: https://github.com/biocompute-objects
biocomputeobject.org

OSF page: https://osf.io/r6s4u/

BioCompute Objects

Specification Document

BioCompute Object (BCO) specification document

v1.2.0

BioCompute Object Consortium members (BCOC):

FDA: Vahan Simonyan, Mark Walderhaug, Ruth Bandler, Eric Donaldson, Elaine Thompson, Alin Voskanian, Anton Golikov, Konstantinos Karagiannis, Elaine Johanson, Adrian Myers, Errol Strain, Khaled Bouri, Tong Weida, Wenming Xiao, Md Shamsuzzaman

GW: Raja Mazumder, Charles Hadley S. King IV, Amanda Bell, Jeet Vora, Krista M. Smith, Robel Kahsay

Documentation Community: Gil Alterovitz (Boston Children's Hospital/Harvard Medical School, SMART/FHIR/HL7, GA4GH), Michael R. Crusoe (CWL), Marco Schito (C-Path), Konstantinos Krampis (CUNY), Alexander (Sasha) Wait Zaranek (Curoverse), John Quackenbush (DFCI/Harvard), Geet Duggal (DNAnexus), Singer Ma (DNAnexus), Yuching Lai (DDL), Warren Kibbe (Duke), Tony Burdett (EBI), Helen Parkinson (EBI), Stuart Young (Engility Corp), Anupama Joshi (Epinomics), Vineeta Agarwala (Flatiron Health), James Hirmas (GenomeNext), David Steinberg (UCSC), Veronica Miller (HIV Forum), Dan Taylor (Internet 2), Paul Duncan (Merck), Jianchao Yao (Merck & Co., Inc., Boston, MA USA), Marilyn Matz (Paradigm4), Ben Busby (NCBI), Eugene Yaschenko (NCBI), Zhining Wang (NCI), Hsinyi (Steve) Tsang (NCI), Durga Addepalli (NCI/Attain), Heidi Sofia (NIH), Scott Jackson (NIST), Paul Walsh (NSilico Life Science), Toby Bloom (NYGC), Hiroki Morizono (CNMC), Jeremy Goecks (Oregon Health and Science University), Srikanth Gottipati (Otsuka-US), Alex Poliakov (Paradigm4), Keith Nangle (Pistoia Alliance), Jonas S Almeida (Stony Brook Univ, SUNY), Dennis A. Dean, II (Seven Bridges Genomics), Dustin Holloway (Seven Bridges Genomics), Nisha Agarwal (Solvuu), Stian Soiland-Reyes (UNIMAN), Carole Goble (UNIMAN), Susanna-Assunta Sansone (University of Oxford), Philippe Rocca-Serra (University of Oxford), Phil Bourne (Univ. of Virginia), Joseph Nooraga (Fred Hutchinson Cancer Research Center)

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

Contact: Raja Mazumder (mazumder@gwu.edu) and Vahan Simonyan (vahansim@gmail.com)

Table of Contents

1 Introduction

BioCompute is a paradigm and a BioCompute Object (BCO) is an instance of that paradigm. High-throughput sequencing (HTS), also referred to as next-generation sequencing (NGS) or massively parallel sequencing (MPS), has increased the pace at which we generate, compute and share genomic data in biomedical sciences. As a result, scientists, clinicians and regulators are now faced with a new data paradigm that is less portable, more complex and most of all poorly standardized. BCOs use a simple JSON format to encode important information on the execution of computational pipelines, or for the creation of knowledgebases. BioCompute can be process oriented (for software pipelines) and/or product oriented (for knowledge bases). So error domain can include information to do QA and/or QC. The goal of using a BCO is to streamline communication of these details between stakeholders in academia, industry and regulatory agencies.

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 HTS computations and data formats. Standardized HTS data processing and data formats will promote interoperability and simplify 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 versions 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 agents involved along with other important metadata, such as their specific contribution

Knowledge of input data is intended to be captured according to existing efforts, including MIRAGE, MIAPE, and STRENDA, and to be in accordance with Minimum Information Standards. In addition to all the information captured in the BCO, the BCO itself must be independent of the execution environment, whether it is a local high-performance or a cloud-based infrastructure.

Additional, non-normative, information on BCOs:
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5510742/

1.1 Mission of the BioCompute project

- Develop BioCompute Objects that will facilitate communication of HTS computational analysis details with the FDA.

- Develop a community of stakeholders to create a versatile data harmonization framework that allows the standardized definition of platform-independent bioinformatics pipelines for execution, and is easily read by humans AND machines.
- Facilitate the development of tools and facilities implementing data typing, instantiation, deposition, storage, and distribution of validated BioCompute Objects through a BioCompute database, in order to enable 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 tangible physical, chemical, and biological experiments due to the multitude of environmental and procedural factors is well documented. What is often systematically overlooked is that computational biology algorithms are affected by a multiplicity of parameters and are no less volatile. The complexities of computation protocols and interpretation of outcomes is only part of the challenge; there are also virtually no standardized and industry-accepted metadata schemas for reporting the computational pipelines and parameters 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. The BCO concept has been developed specifically to satisfy regulatory research needs for evaluation, validation, and verification of bioinformatics pipelines; however, there is potential utility of BCO within the larger scientific community. This utility can be increased through the creation of a BCO database comprising records relevant to the U.S. Food and Drug Administration.

A BioCompute Object database record will be similar to a GenBank record in form; however, 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 page.

1.2.1 Limitations of the 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, and at this stage, we are limiting our focus to HTS analysis and database creation.

1.3 Audience for this document

- Users performing HTS analysis with a regulatory science perspective
- HTS Platform Developers
- HTS related standard developers

1.4 Potential Stakeholders for the BioCompute project

- US Food and Drug Administration, as well as other Regulatory Agencies
- 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
- US National Institutes of Health (NIH) (particularly initiatives such as NCI/ITCR)
- Public cloud companies operating in the Life Sciences sector including electronic health record (EHR) systems

1.5 BCO User stories

- Reproducibility and Interpretation use case:

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 the location of the discrepancies. Attaching a BioCompute Object with the initial submission would prevent most of the ambiguity surrounding the discrepancies.

- Reusability use case:

A regulatory decision has been made where a computational analysis has been used as evidence. New data emerges after the product has been on the market over a year and the regulators cannot reproduce the original environment with the configuration of tools and parameters of pipelines to reanalyze the initial submission data or replicate the initial conclusion.

- Collaboration 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.

- Accountability 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.

- Versioning use case:

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).

- Provenance use case:

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.

- Data integration use case:

A BCO can be used to provide clarity and transparency of the data integration process to both the new and existing collaborators. When new data is integrated into the existing data model, BCO can be used to describe data source information (eg- authors/contributors, data version etc), a QC workflow, data content, data modification if any. The BCO also allows reuse of the same workflow to integrate new data with same structure and source. BCO also provides a way to access and track data records which were eliminated in the integration/QC process due to rules or restrictions of the existing data model. Knowledgebases using BCOs in the form of ‘readme’ can provide provenance for every piece of data that is collected and presented to the user. Such granular tracking facilitates fair sharing of data and provides mechanisms for adherence to licensing requirements associated with specific datasets.

1.6 BCO community

The BioCompute Object working group facilitates a means for different stakeholders in the HTS communities to provide input on current practices on the BCO. This working group was formed during preparation for the 2017 HTS Computational Standards for Regulatory Sciences Workshop, and was initially made up of the workshop participants, both speakers and panelists. There has been a continual growth of the BCO working group as a direct result of the interaction between a variety of stakeholders from all interested communities in standardization of computational HTS data processing.

2 BioCompute Object explained

The fundamentals of data typing (type primitives, class inheritance, etc.) that are used to define BioCompute Objects are described in detail in Sections 3.1, 3.2, and 3.3. 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 Object 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.

BioCompute data types are defined as aggregates of the critical fields organized into a few domains: the descriptive domain, the identification and provenance domain, the input and output domains, the parametric domain, the environmental domain, the execution domain, the prerequisite domain, the usability domain, and the error domain. At the time of submission to the BioCompute Object 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 and a unique digital object identifier.

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. Required fields are indicated by the `"vital" : "True"` flag, which is shown in the data typing section (Sections 3.1 - 3.3).

2.0 Top Level Fields

These header fields uniquely define a BCO. These fields are required for every BCO.

2.0.1 BioCompute Object Identifier "BCO_id"

A unique identifier that should be applied to each BCO instance. These can be assigned by a BCO database engine. IDs should be URIs (expressed as a URN or URL). IDs should never be reused.

```
"BCO_id" : "https://github.com/biocompute-
objects/BCO_Spec_V1.2/blob/hadley_local/HCV1a.json"
```

2.0.2 Type “type”

As any object of type 'type,' it has its own fields: _type, _id, _inherits, name, title and description. Type of this JSON object is “antiviral_resistance_detection”

```
"type": "antiviral_resistance_detection"
```

2.0.3 Digital signature "digital_signature"

A string-type, read-only value generated and stored by a BCO database, protecting the object from internal or external alterations without proper validation. The string can be generated through the use of an SHA-256 or implementation specific hash function.

```
"digital_signature": "905d7fce3f3ac64c8ea86f058ca71658"
```

This value should not be submitted during deposition but can be read during downloading or transferring validated BCOs. The BCO server can provide an API validating the signature versus BCO content, allowing users to validate the signature "offline" on their own. The server will also must provide a reference to the signature creation algorithm, facilitating for greater interoperability.

2.0.4 BCO version “bco_spec_version”

The version of the BCO specification used to define this document.

```
"bco_spec_version": "v1.2"
```

2.1 Provenance Domain "provenance_domain"

2.1.1 Name "name"

Name for the BCO. This public field should take free text value using common biological research terminology supporting external reference linkage identifiers whenever possible for use in the structured name.

```
"name": "HCV1a ledipasvir resistance SNP detection"
```

2.1.2 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 [taxonomy:$taxonomy] mutation detection" will be visualized as "HCV1a [taxonomy:31646] mutation detection" assuming the BCO has a field called “taxonomy” with value “31646”.

```
"structured_name": "HCV1a [taxonomy:$taxonomy] mutation detection",
"taxonomy": "31646”,
...

=> HCV1a [taxonomy:31646] mutation detection
```

2.1.3 Version "version"

Records the versioning of this BCO instance object. In BCO versioning, a change in the BCO affecting 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": "2.1",
```

2.1.4 Review "review"

Describes the status of an object in the review process. The 'unreviewed' flag indicates that the object has been submitted, but no further evaluation or verification has occurred. The ‘in-review’ flag indicates that verification is underway. The 'approved' flag indicates that the BCO has been verified and reviewed. The 'suspended' flag indicates an object that was once valid is no longer considered valid. The 'rejected' flag indicates that an error or inconsistency was detected in the BCO, and it has been removed or rejected. The fields from the “`contributor`” object (described in section 2.1.10) is inherited to populate the reviewer section.

```
        "review": [
            {
                "status": "approved",
                "reviewer_comment": ["Approved by GW staff. Waiting for
approval from FDA Reviewer"],
                "reviewer": {
                    "name": "Charles Hadley King",
                    "affiliation": "George Washington University",
                    "email": "hadley_king@gwu.edu",
                    "contribution": ["curatedBy"],
                    "orcid": "https://orcid.org/0000-0003-1409-4549"
                }
            },
            {
                "status": "approved",
                "reviewer_comment": ["The revised BCO looks fine"],
                "reviewer": {
                    "name": "Eric Donaldson",
                    "affiliation": "FDA",
                    "email": "Eric.Donaldson@fda.hhs.gov",
                    "contribution": ["curatedBy"]
                }
            }
        ]
```

2.1.5 Inheritance/derivation “derived_from”

If the object is derived from another, this field will specify the parent object, in the form of the ‘objectid’. If the object inherits only from the base BioCompute Object or a type definition, then the value here is null.

```
          "derived_from" : null,
```

2.1.6 Obsolescence “obsolete”

If the object has an expiration date this field will specify that using the ‘datetime’ type which is in ISO-8601 format as clarified by W3C <https://www.w3.org/TR/NOTE-datetime>. This field is optional.

```
        "obsolete" : "2118-09-26T14:43:43-0400"
```

2.1.7 Embargo ‘embargo’

If the object has a period of time that it is not public, that range can be specified using these fields. Using the ‘datetime’ type a start and end time are specified for the embargo. These fields are optional.

```
"embargo" : {
    "start_time": "2000-09-26T14:43:43-0400",
    "end_time": "2000-09-26T14:43:45-0400"
},
```

2.1.8 Created ‘created’

Using the ‘datetime’ type the time of initial creation of the BCO is recorded in ISO-8601 format as clarified by W3C <https://www.w3.org/TR/NOTE-datetime>.

```
"created": "2017-01-20T09:40:17-0500"
```

2.1.9 Modification ‘modified’

Using the ‘datetime’ type the time of most recent modification of the BCO is recorded

```
"modified": "2018-03-21T18:31:48-0400"
```

2.1.10 Contributors "contributors"

This is a list to hold contributor identifiers and a description of their type of contribution, including a field for ORCIDs to record author information, as they allow for the author to curate their information after submission. ORCID identifiers must be valid and must have the prefix ‘https://orcid.org/’. The contribution type is a choice taken from PAV ontology: provenance, authoring and versioning, which also maps to the PROV-O.

```
"contributors": [
    {
        "name": "Charles Hadley King",
        "affiliation": "George Washington University",
        "email": "hadley_king@gwu.edu",
        "contribution": ["createdBy", "curatedBy"]
        "orcid": "https://orcid.org/0000-0003-1409-4549"
    },
    {
```

```
            "name": "Eric Donaldson",
            "affiliation": "FDA",
            "email": "Eric.Donaldson@fda.hhs.gov",
            "contribution": ["authoredBy"]
        }
    ]
```

2.1.11 License “license”

A space for Creative Commons licence or other licence information (text). The default or recommended licence can be Attribution 4.0 International: for example https://spdx.org/licenses/CC-BY-4.0.html

```
"license": "https://spdx.org/licenses/CC-BY-4.0.html"
```

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.

```
"usability_domain": [
    "Identify baseline single nucleotide polymorphisms SNPs [SO:0000694],
insertions [so:SO:0000667], and deletions [so: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:SO:0000048]
that correlate with antiviral drug treatment failure",
    "Determine whether the treatment emergent amino acid substitutions
[so: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"
],
```

2.3 Extension Domain "extension_domain"

The extension domain is for a user to add more structured information that is defined in the type definition. This section is not evaluated by checks for BCO validity or computational correctness. Two examples follow:

2.3.1 Extension to External References: SMART on FHIR Genomics

The FHIR Endpoint URL coupled with the specific resource type and a unique FHIR identifier leads to a resource that can contain everything from the date and time of the procedure, specimen details, sequence information, linked sequence repositories, associated pedigrees, or even a set of observations linked from diagnostic reports. The link to FHIR can also be added to the usability domain. More on FHIR Genomics in release 3 of FHIR can be found here: https://www.hl7.org/fhir/genomics.html

SMART on FHIR Genomics provides a framework for EHR-based apps built on FHIR that integrate clinical and genomic information. For more information on how to use the SMART on FHIR Genomics apps, please visit http://projects.iq.harvard.edu/smartgenomics/.


```
"extension_domain":{
    "FHIR_extension": [
        {
            "FHIRendpoint_Resource": "Sequence",
            "FHIRendpoint_URL": "http://fhirtest.uhn.ca/baseDstu3",
            "FHIRendpoint_Ids": ["21376"]
        },
        {
            "FHIRendpoint_Resource": "DiagnosticReport",
            "FHIRendpoint_URL": "http://fhirtest.uhn.ca/baseDstu3",
            "FHIRendpoint_Ids": ["6288583"]
        },
        {
            "FHIRendpoint_Resource": "ProcedureRequest",
            "FHIRendpoint_URL": "http://fhirtest.uhn.ca/baseDstu3",
            "FHIRendpoint_Ids": ["25544"]
        },
        {
            "FHIRendpoint_Resource": "Observation",
            "FHIRendpoint_URL": "http://fhirtest.uhn.ca/baseDstu3",
            "FHIRendpoint_Ids": ["92440"]
        },
        {
            "FHIRendpoint_ResourceType": "FamilyMemberHistory",
            "FHIRendpoint_URL": "http://fhirtest.uhn.ca/baseDstu3",
            "FHIRendpoint_Ids": ["4588936"]
        }
    ],
```

2.3.2 Extension to External References: GitHub

The external references also include an extension to GitHub repositories where HTS computational analysis pipelines, workflows, protocols, and tool or software source code can be stored/deposited/downloaded. 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_extension": {
            "github_repository": "https://github.com/common-workflow-
language/hive-cwl-examples",
            "github_URI":"https://github.com/common-workflow-language/hive-cwl-
examples/blob/c9ffea0b60fa3bcf8e138af7c99ca141a6b8fb21/workflow/hive-viral-
mutation-detection.cwl"
            }
    },
```

2.4 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.

2.4.1 Keywords "keywords"

This is a list of key map fields to hold a list of keywords to aid in search-ability and description of the object.

```
        "keywords": [
            {
                "key": "search terms",
                "value": [
                    "HCV1a",
                    "Ledipasvir",
                    "antiviral resistance",
                    "SNP",
                    "amino acid substitutions"
                ]
            }
        ]
```

2.4.2 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 Section 3.5 for a list of the CURIEs used in this example.


```
"xref": [
    {
        "namespace": "pubchem.compound",
        "name": "PubChem-compound",
        "ids": ["67505836"],
        "access_time": "2018-13-02T10:15-05:00"
    },
    {
        "namespace": "pubmed",
        "name": "PubMed",
        "ids": ["26508693"],
        "access_time": "2018-13-02T10:15-05:00"
    },
    {
        "namespace": "so",
        "name": "Sequence Ontology",
        "ids": ["0000048"],
        "access_time": "2018-13-02T10:15-05:00"
    },
    {
        "namespace": "taxonomy",
        "name": "Taxonomy",
        "ids": ["31646"],
        "access_time": "2018-13-02T10:15-05:00"
    }
],
```


2.4.3 Pipeline tools "pipeline_steps"

This is 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.

2.4.3.1 Step Number "step_number"

This is a non-negative integer value representing the position of the tool in a one-dimensional representation of the pipeline. The number is a suggestion for a *partial order* for presentation purposes, e.g. parallel computations assigned the same number based on their first possible execution. Actual execution order might differ from the step number. Gaps are allowed (e.g. step 20 follows step 10).

```
"step_number": "1"
```

2.4.3.2 Name "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.

```
"name": "HIVE-hexagon"
```

2.4.3.2 Tool Description "description"

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

```
"description": "Alignment of reads to a set of references",
```

2.4.3.3 Tool Version "version"

The version assigned to the instance of the tool used corresponding to the upstream release.

```
"version": "1.3",
```

2.4.3.4 Tool Prerequisites "prerequisite"

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

```
                    "prerequisite": [
                        {
                            "name": "Hepatitis C virus genotype 1",
                            "source": {
                                "address":
"http://www.ncbi.nlm.nih.gov/nuccore/22129792",
                                "access_time": "2017-01-24T09:40:17-0500"
                            }
```


```
                        },
                        {
                            "name": "Hepatitis C virus type 1b complete
genome",
                            "source": {
                                "address":
"http://www.ncbi.nlm.nih.gov/nuccore/5420376",
                                "access_time": "2017-01-24T09:40:17-0500"
                            }
                        },
                        {
                            "name": "Hepatitis C virus (isolate JFH-1) genomic
RNA",
                            "source": {
                                "address":
"http://www.ncbi.nlm.nih.gov/nuccore/13122261",
                                "access_time": "2017-01-24T09:40:17-0500"
                            }
                        },
                        {
                            "name": "Hepatitis C virus clone J8CF, complete
genome",
                            "source": {
                                "address":
"http://www.ncbi.nlm.nih.gov/nuccore/386646758",
                                "access_time": "2017-01-24T09:40:17-0500"
                            }
                        },
                        {
                            "name": "Hepatitis C virus S52 polyprotein gene",
                            "source": {
                                "address":
"http://www.ncbi.nlm.nih.gov/nuccore/295311559",
                                "access_time": "2017-01-24T09:40:17-0500"
                            }
                        }
                    ]
```

2.4.3.6 Input List "input_list"

Each tool lists the URIs (expressed as a URN or URL) of the input files. These are a catchall for read files, reference files or any other type of input. All of these fields are optional and for descriptive purposes, therefore the structure here is less rigid than in other fields.

```
                    "input_list": [
                        {
                            "address":
"https://hive.biochemistry.gwu.edu/dna.cgi?cmd=objFile&ids=514683",
                            "access_time": "2017-01-24T09:40:17-0500"
                        },
                        {
                            "address":
"https://hive.biochemistry.gwu.edu/dna.cgi?cmd=objFile&ids=514682",
                            "access_time": "2017-01-24T09:40:17-0500"
                        }
                    ],
```

2.4.3.7 Output List "output_list"

Each tool lists the URIs (expressed as a URN or URL) of the output files for that tool.

```
                    "output_list": [
                        {
                            "address":
"https://hive.biochemistry.gwu.edudata/514769/allCount-aligned.csv",
                            "access_time": "2017-01-24T09:40:17-0500"
                        }
                    ]
```

2.5 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."

2.5.1 Script Access Type "script_access_type"

This field indicates whether the code of the "script" to execute the BioCompute Object is accessed as an external file via HTTP (URI) or in-line text in the "script" field. Valid options are “URI” or “text”.

```
"script_access_type": "URI" OR "text"
```

2.5.2 Script "script"

The Script field points to an 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 Seven Bridges 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://example.com/workflows/antiviral_resistance_detection_hive.py"}
```

2.5.3 Pipeline Version "pipeline_version"

This field records the version of the pipeline implementation.

```
"pipeline_version": "2.0"
```

2.5.4 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"
```

2.5.5 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 specify “cwl-runner”. 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 publicly or privately accessible.

```
      "script_driver": "shell"
```

2.5.6 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. Recommended keys are “name” and “version”, but their interpretation is implementation-dependent for a given script_driver.

```
    "software_prerequisites": [
        {
            "name": "HIVE-hexagon",
            "version": "babajanian.1",
            "uri": {
                "address": "https://hive.biochemistry.gwu.edu/dna.cgi?cmd=dna-
hexagon&cmdMode=-",
                "access_time": "2017-01-24T09:40:17-0500",
                "sha1_chksum": null
            }
        },
        {
            "name": "HIVE-heptagon",
            "version": "albinoni.2",
            "uri": {
                "address": "https://hive.biochemistry.gwu.edu/dna.cgi?cmd=dna-
heptagon&cmdMode=-",
                "access_time": "2017-01-24T09:40:17-0500",
                "sha1_chksum": null
            }
        }
    ],
```

2.5.7 Domain Prerequisites "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. The values under this field present the requirements for network protocol endpoints used by a pipeline’s scripts, or other software. The key "url" defines an endpoint to be accessed. If the "path" is "/" then any resource at the given domain may be accessed, while if the path is more specific than only resources which path prefix matches may be accessed.

```
"domain_prerequisites": [
    {
        "name": "HIVE",
        "url": "https://hive.biochemistry.gwu.edu/dna.cgi?cmd=login"
    },
    {
        "name": "access to ftp",
        "url": "ftp://:22/"
    },
    {
        "name": "access to e-utils",
        "url": "http://eutils.ncbi.nlm.nih.gov/entrez/eutils/"
    },
      {
            "name": "generic name",
          "url": "protocol://domain:port/application/path",
            }
]
```

2.5.8 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. The possible keys are specific to each platform. The “value” should be a JSON string.

```
    "env_parameters": {
        "key": "HOSTTYPE",
        "value" : "x86_64-linux"
    }
}
```

2.6 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": {
```

```
            "HIVE-hexagon" : {
                "seed": 14,
                "minimum_match_len": 66,
                "divergence_threshold_percent": 0.30
            },
            "HIVE-heptagon": {
                "minimum_coverage": 15,
                "freq_cutoff": 0.10
            }
        }
```

2.7 Input and output Domain "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.

2.7.1 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. This allows the author to be very specific about a particular type of input file, if they so choose. For example: reference files have common names, and adding the common name here, in addition to the uri would make this more readable and understandable (eg, "HCV reference version..." or "human reference GRCH38"). For data integration workflows, the input files can be a table downloaded from a specific source which is then filtered for modified using rules described in the BCO.

```
            "input_subdomain": {
                "subject": [
                    {
                        "name": "Hepatitis C virus genotype 1",
                        "source": {
                            "address":
"http://www.ncbi.nlm.nih.gov/nuccore/22129792",
                            "access_time": "2017-01-24T09:40:17-0500"
```

```
                    }
                },
                {
                    "name": "Hepatitis C virus type 1b complete genome",
                    "source": {
                        "address":
"http://www.ncbi.nlm.nih.gov/nuccore/5420376",
                        "access_time": "2017-01-24T09:40:17-0500"
                    }
                },
            "query": [
                {
                    "name": "HCV1a_drug_resistant_sample0001-01",
                    "source": {
                        "address": "https://hive.biochemistry.gwu.edunuc-
read/514682",
                        "access_time": "2017-01-24T09:40:17-0500"
                    }
                },
                {
                    "name": "HCV1a_drug_resistant_sample0001-02",
                    "source": {
                        "address": "https://hive.biochemistry.gwu.edunuc-
read/514683",
                        "access_time": "2017-01-24T09:40:17-0500"
                    }
                }
            ]
        }
```

2.7.2 Output Subdomain "output_subdomain"

This field records the outputs for the entire pipeline. Each file should be an object with a key, and a title, URI, and media type (https://www.iana.org/assignments/media-types/) value.

```
        "output_subdomain": [
            {
                "mediatype": "text/csv",
                "source": {
                    "address":
"https://hive.biochemistry.gwu.edudata/514769/dnaAccessionBased.csv",
                    "access_time": "2017-01-24T09:40:17-0500"
                }
            },
```

```
            {
                "mediatype": "text/csv",
                "uri": {
                    "address":
"https://hive.biochemistry.gwu.edudata/514801/SNPProfile.csv",
                    "access_time": "2017-01-24T09:40:17-0500"
                }
            }
        ]
    },
```

2.8 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.

For data integration BCOs used to develop knowledgebases the error domain can, for example, contain a records of annotations that did not pass a set of rules.

The possible keys within each subdomains are workflow-specific free text which should be readable for a human.

```
    "error_domain": {
        "empirical_error": {
            "false negative alignment hits": "<0.0010",
            "false discovery": "<0.05"
        },
```

```
            "algorithmic_error": {
                "false_positive_mutation_calls_discovery": "<0.00005",
                "false_discovery": "0.005"
            }
        }
```

3 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 a federation of other objects.

3.1 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" : {
        "_comment": "primitive type hash with short descriptions",
        "_type" : {
            "string": "alphanumeric string",
            "integer": "integer number",
            "float": "floating point number",
            "boolean": "can only be assigned true, false, or null",
            "date": "specified ISO 8601",
            "time": "ISO 8601",
            "dateTime": "ISO 8601",
            "blob": "binary data stored as a single object",
            "json" : "any json formatted subobject, the structure to json is
not imposed by base biocompute type, but should follow guidelines of using the
predefined primitives"
        }
```

```
    },
    "objectid": {
        "_type": "string",
        "_comment": ["a string of characters designed for unambiguous
identification of resources and extensibility via the URI scheme
(https://www.w3.org/wiki/UriSchemes)"],
        "_constraint": {
            "regex" : "url_regex"
        }
    },
    "version": {
        "_type": "string",
        "_comment": "suggestion major[.minor[.build_nunber]]"
    },
    "email":{
        "_type": "string",
        "_constraint": {
            "regex" : "email_regex"
        }
    },
    "keyval" : {
        "key" : {
            "_type" : "string"
        },
        "value" : {
            "_type" : "string"
        }
    },
    "keymap" : {
        "key" : {
            "_type" : "string"
        },
        "value" : {
            "_type" : "string",
            "_multi_value" : true
        }
    },
    "uri": {
        "address": {
            "_type": "objectid"
        },
        "access_time": {
            "_type" : "dateTime",
            "_optional" : true
        },
```

```
            "sha1_chksum": {
                "_type": "string",
                "_optional" : true,
                "_constraint" : {
                    "regex" : "[A-Za-z0-9]+"
                }
            }
        },
        "contribution": {
            "_type" : "string",
            "_multi_value" : true,
            "_comment": "taken from https://doi.org/10.1186/2041-1480-4-37",
            "_constraint" :{
               "_choice" : ["authoredBy", "contributedBy", "createdAt",
"createdBy", "createdWith", "curatedBy", "derivedFrom", "importedBy",
"importedFrom", "providedBy", "retrievedBy", "retrievedFrom",
"sourceAccessedBy"]
            }
        },
        "contributor": {
            "_comment": [""],
            "name": {
                "_type": "string",
                "_optional" : true
            },
            "affiliation": {
                "_type": "string",
                "_optional" : true
            },
            "email" : {
                "_type" : "email",
                "_optional" : true
            },
            "contribution": {
                "_type": "contribution"
            },
            "orcid": {
                "_type": "objectid"
            }
        },
        "file": {
            "uri" : {
                "_type" : "uri"
            },
           "mediatype": {
```

```
            "_type" : "string",
            "_optional" : true
        }
    },
    "xref" : {
        "namespace" : {
            "_type" : "string",
            "_comment" : "can be a prefix in identifiers.org or a db named in a
source"
        },
        "ids" : {
            "_type" : "string",
            "_multi_value" : true
        },
        "name" : {
            "_comment": "This can be the common name for the db",
            "_type" : "string",
            "_optional" :true
        },
        "access_time":{
            "_type": "dateTime",
            "_optional" : true
        }
    }
}
```

3.2 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 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.

3.3 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 is the type definition for “BioCompute_base_type”:

```
{
    "name" : "BioCompute_base_type",
    "title" : "Base type for all BioCompute Object types",
    "description" : "All BioCompute object types must inherit from this type in
order to be compliant with BioCompute framework",
    "_comment" : [
```

```
        "Since JSON format do not allow comments '_comment' key is used for
comments and should be ignored by parsers",
        "As any object of type 'type' it has its own fields: _type, _id,
_inherits, name, title and description",
        "As base type this type inherits from none",
        "Identifier of the object since type itself is an object",
        "Type of this JSON object is 'type'",
        "For each field default value for '_multi_value', '_optional', and
'_read_only' key is 'false' "
    ],
    "_fields" : {
        "_comment" : [
            "List of fields of the BioCompute Object type"
        ],

        "BCO_id":{
            "_type" : "string",
            "_read_only" : true,
            "_comment" : [
                "A unique identifier that should be applied to each BCO
instance, assigned by a BCO database engine.",
                "IDs should never be reused"
            ]
        },
        "type": {
            "_type": "_type",
            "_read_only" : true
        },
        "digital_signature" : {
            "_type" : "string",
            "_read_only" : true,
            "_comment" : ["Digital signature of BioCompute Object by
Authority"],
            "_constraint" : {
                "regex" : "[A-Za-z0-9]+"
            }
        },
        "bco_spec_version" : {
            "_type" : "string",
            "_read_only" : true,
            "_comment" : ["The version of the BCO specification used to define
this document."]
        },

        "provenance_domain": {
```

```
            "name" : {
                "_type" :"string",
                "_comment" : "Public searchable name for BioCompute Object"
            },
            "structured_name" :{
                "_type" :"string",
                "_optional" : true,
                "_comment" :[
                    "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."
                ]

            },
            "version" : {
                "_type" : "version",
                "_comment" : "version of this BioCompute Object"
            },
            "review" : {
                "_multi_value" : true,
                "_optional" : true,
                "status" : {
                    "_type" : "string",
                    "_constraint" : {
                        "_choice" : [ "unreviewed", "in-review", "approved",
"rejected", "suspended" ]
                    },
                    "_comment" : "Current verification status of the BioCompute
Object",
                    "_default_value" : "unreviewed"
                },
                "reviewer" : {
                    "_type": "contributor",
                    "_multi_value" : true
                },
                "date" : {
                    "_type" : "dateTime"
                },
                "reviewer_comment" : {
                    "_type" : "string",
                    "_optional" : true,
                    "_multi_value" : true
                }
            },
```

```
            "derived_from": {
                "_type": "objectid",
                "_optional" : true,
                "_comment" : ["value of _id field of another bio compute
object"]
            },
            "obsolete_after" : {
                "_type": "datetime" ,
                "_optional" : true
            },
            "embargo" : {
                "start_time" : {
                    "_type" : "datetime",
                    "_optional" : true
                },
                "end_time" : {
                    "_type" : "datetime",
                    "_optional" : true
                }
            },
            "created" : {
                "_type" :"datetime",
                "_comment" : ["Date and time of the BioCompute Object
creation"],
                "_read_only" : true
            },
            "modified" : {
                "_type" :"datetime",
                "_comment" : ["Date and time of the BioCompute Object was last
modified"],
                "_read_only" : true
            },
            "contributors" : {
                "_type" : "contributor",
                "_multi_value" : true
            },
            "license" : {
                "_type" : "string",
                "_optional" : true,
                "_comment" : ["Creative Commons licence or other licence
information (text) space. The default or recommended licence can be Attribution
4.0 International: for example https://spdx.org/licenses/CC-BY-4.0.html"]
            }
        },
```

```
        "usability_domain" : {
            "_type" : "string",
            "_comment" : ["Text from biospec"],
            "_multi_value" : true
        },

        "extension_domain":{
            "_type" : "json",
            "_optional" : true,
            "_comment" : [
                "all fields in this domain are BioCompute specific and should
be defined in inherited BioCompute type",
                "This domain allows for the addition of "
            ]
        },

        "description_domain" : {
            "keywords" : {
                "_type" : "keymap",
                "_multi_value" : true
            },
            "xref" : {
                "_type" : "xref",
                "_optional" : true,
                "_multi_value" : true
            },
            "pipeline_steps" : {
                "tool" : {
                    "_multi_value" : true,
                    "name" : {
                        "_type" : "string",
                        "_comment" : ["this is a recognized name of the
software tool"]
                    },
                    "description" : {
                       "_type" : "string"
                    },
                    "step_number": {
                        "_type": "integer"
                    },
                    "version" : {
                       "_type" : "version"
                    },
                    "prerequisite" : {
                        "_multi_value" : true,
```

```
                        "_optional" : true,
                        "_comment" : "reference or required prereqs",
                        "name" : {
                            "_type" :"string",
                            "_comment" : ["Public searchable name for reference
or prereq"]
                        },
                        "source":{
                            "_type" : "uri"
                        }
                    },
                    "input_list" : {
                        "_type" : "uri",
                        "_multi_value" : true
                    },
                    "output_list" : {
                        "_type" : "uri",
                        "_multi_value" : true
                    }
                }
            }
        },

        "execution_domain" : {
            "script_access_type" : {
               "_type" :"string",
                "_constraint" : {
                    "_choice" : [ "URI", "text" ]
                }
            },
            "script" : {
               "_type" : "string"
            },
            "script_driver" : {
               "_type" : "string"
            },
            "pipeline_version" : {
                "_type" : "version"
            },
            "platform" : {
                "_type" : "string"
            },
            "software_prerequisites" : {
                "_multi_value" : true,
                "name" : {
```

```
                    "_type" : "string"
                },
                "version" : {
                    "_type" : "version"
                },
                "uri" : {
                    "_type" : "uri",
                    "_optional" : true
                }
            },
            "domain_prerequisites" : {
                "_multi_value" : true,
                "name" : {
                    "_type" : "string"
                },
                "url" : {
                    "_type" : "string",
                    "_multi_value" : true
                }
            },
            "environment_variables" : {
                "_type" : "keyval",
                "_multi_value" : true
            }
        },

        "parametric_domain" : {
            "_type" : "json",
            "_comment" : [
                "all fields in this domain should be defined in inheriting
BioCompute subtypes",
                "see example in bco_type_examples.json object
parametric_domain_type_definition_examples"
            ]
        },

        "io_domain" : {
            "input_subdomain" : {
                "_type" : "json",
                "_comment" : "all fields in this domain are BioCompute specific
and should be defined in inherited BioCompute type"
            },

            "output_subdomain" : {
                "_type" : "file",
```

```
                "_multi_value" : true,
                "_comment" : "output is a file object"
            }
        },
        "error_domain": {
            "empirical_error": {
                "_type" : "json",
                "_comment" : "all fields in this domain are BioCompute specific
and should be defined in inherited BioCompute type"
            },
            "algorithmic_error": {
                "_type" : "json",
                "_comment" : "all fields in this domain are BioCompute specific
and should be defined in inherited BioCompute type"
            }
        }
    }
}
```

3.4 BCO expanded view example

```
{
    "BCO_id": "https://github.com/biocompute-
objects/BCO_Spec_V1.2/blob/master/HCV1a.json",
    "type": "antiviral_resistance_detection",
    "digital_signature": "905d7fce3f3ac64c8ea86f058ca71658",
    "bco_spec_version" : "v1.2",
    "provenance_domain": {
        "name": "HCV1a ledipasvir resistance SNP detection",
        "structured_name": "HCV1a [taxonomy:31646] ledipasvir
[pubchem.compound:67505836] resistance SNP [so:0000694] detection",
        "version": "2.9",
        "review": [
            {
                "status": "approved",
                "reviewer_comment": ["Approved by GW staff. Waiting for
approval from FDA Reviewer"],
                "reviewer": {
                    "name": "Charles Hadley King",
                    "affiliation": "George Washington University",
                    "email": "hadley_king@gwu.edu",
                    "contribution": ["curatedBy"],
                    "orcid": "https://orcid.org/0000-0003-1409-4549"
```


```
                }
            },
            {
                "status": "approved",
                "reviewer_comment": ["The revised BCO looks fine"],
                "reviewer": {
                    "name": "Eric Donaldson",
                    "affiliation": "FDA",
                    "email": "Eric.Donaldson@fda.hhs.gov",
                    "contribution": ["curatedBy"]
                }
            }
        ],
        "derived_from" : null,
        "obsolete" : "2118-09-26T14:43:43-0400",
        "embargo" : {
            "start_time": "2000-09-26T14:43:43-0400",
            "end_time": "2000-09-26T14:43:45-0400"
            },
        "created": "2017-01-24T09:40:17-0500",
        "modified": "2018-09-21T14:06:14-0400",
        "contributors": [
            {
                "name": "Charles Hadley King",
                "affiliation": "George Washington University",
                "email": "hadley_king@gwu.edu",
                "contribution": ["createdBy", "curatedBy"],
                "orcid": "https://orcid.org/0000-0003-1409-4549"
            },
            {
                "name": "Eric Donaldson",
                "affiliation": "FDA",
                "email": "Eric.Donaldson@fda.hhs.gov",
                "contribution": ["authoredBy"]
            }
        ],
        "license": "https://spdx.org/licenses/CC-BY-4.0.html"
    },
    "usability_domain": [
        "Identify baseline single nucleotide polymorphisms SNPs [SO:0000694],
insertions [so:SO:0000667], and deletions [so: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:SO:0000048]
that correlate with antiviral drug treatment failure",
```


```
        "Determine whether the treatment emergent amino acid substitutions
[so: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"
    ],
    "extension_domain":{
        "FHIR_extension": [
            {
                "FHIRendpoint_Resource": "Sequence",
                "FHIRendpoint_URL": "http://fhirtest.uhn.ca/baseDstu3",
                "FHIRendpoint_Ids": ["21376"]
            },
            {
                "FHIRendpoint_Resource": "DiagnosticReport",
                "FHIRendpoint_URL": "http://fhirtest.uhn.ca/baseDstu3",
                "FHIRendpoint_Ids": ["6288583"]
            },
            {
                "FHIRendpoint_Resource": "ProcedureRequest",
                "FHIRendpoint_URL": "http://fhirtest.uhn.ca/baseDstu3",
                "FHIRendpoint_Ids": ["25544"]
            },
            {
                "FHIRendpoint_Resource": "Observation",
                "FHIRendpoint_URL": "http://fhirtest.uhn.ca/baseDstu3",
                "FHIRendpoint_Ids": ["92440"]
            },
            {
                "FHIRendpoint_ResourceType": "FamilyMemberHistory",
                "FHIRendpoint_URL": "http://fhirtest.uhn.ca/baseDstu3",
                "FHIRendpoint_Ids": ["4588936"]
            }
        ],
        "github_extension": {
              "github_repository": "https://github.com/common-workflow-
language/hive-cwl-examples",
              "github_URI":"https://github.com/common-workflow-language/hive-
cwl-examples/blob/c9ffea0b60fa3bcf8e138af7c99ca141a6b8fb21/workflow/hive-viral-
mutation-detection.cwl"
        }
    },
    "description_domain": {
        "keywords": [
            {
```


```
            "key": "search terms",
            "value": [
                "HCV1a",
                "Ledipasvir",
                "antiviral resistance",
                "SNP",
                "amino acid substitutions"
            ]
        }
    ],
    "xref": [
        {
            "namespace": "pubchem.compound",
            "name": "PubChem-compound",
            "ids": ["67505836"],
            "access_time": "2018-13-02T10:15-05:00"
        },
        {
            "namespace": "pubmed",
            "name": "PubMed",
            "ids": ["26508693"],
            "access_time": "2018-13-02T10:15-05:00"
        },
        {
            "namespace": "so",
            "name": "Sequence Ontology",
            "ids": ["0000048"],
            "access_time": "2018-13-02T10:15-05:00"
        },
        {
            "namespace": "taxonomy",
            "name": "Taxonomy",
            "ids": ["31646"],
            "access_time": "2018-13-02T10:15-05:00"
        }
    ],
    "pipeline_steps": {
        "tool": [
            {
                "step_number": "1",
                "name": "HIVE-hexagon",
                "description": "Alignment of reads to a set of references",
                "version": "1.3",
                "prerequisite": [
                    {
```

```
                                "name": "Hepatitis C virus genotype 1",
                                "source": {
                                    "address":
"http://www.ncbi.nlm.nih.gov/nuccore/22129792",
                                    "access_time": "2017-01-24T09:40:17-0500"
                                }
                            },
                            {
                                "name": "Hepatitis C virus type 1b complete
genome",
                                "source": {
                                    "address":
"http://www.ncbi.nlm.nih.gov/nuccore/5420376",
                                    "access_time": "2017-01-24T09:40:17-0500"
                                }
                            },
                            {
                                "name": "Hepatitis C virus (isolate JFH-1) genomic
RNA",
                                "source": {
                                    "address":
"http://www.ncbi.nlm.nih.gov/nuccore/13122261",
                                    "access_time": "2017-01-24T09:40:17-0500"
                                }
                            },
                            {
                                "name": "Hepatitis C virus clone J8CF, complete
genome",
                                "source": {
                                    "address":
"http://www.ncbi.nlm.nih.gov/nuccore/386646758",
                                    "access_time": "2017-01-24T09:40:17-0500"
                                }
                            },
                            {
                                "name": "Hepatitis C virus S52 polyprotein gene",
                                "source": {
                                    "address":
"http://www.ncbi.nlm.nih.gov/nuccore/295311559",
                                    "access_time": "2017-01-24T09:40:17-0500"
                                }
                            }
                        ],
                        "input_list": [
                            {
```

```
                            "address":
"https://hive.biochemistry.gwu.edu/dna.cgi?cmd=objFile&ids=514683",
                            "access_time": "2017-01-24T09:40:17-0500"
                        },
                        {
                            "address":
"https://hive.biochemistry.gwu.edu/dna.cgi?cmd=objFile&ids=514682",
                            "access_time": "2017-01-24T09:40:17-0500"
                        }
                    ],
                    "output_list": [
                        {
                            "address":
"https://hive.biochemistry.gwu.edudata/514769/allCount-aligned.csv",
                            "access_time": "2017-01-24T09:40:17-0500"
                        }
                    ]
                },
                {
                    "step_number": "2",
                    "name": "HIVE-heptagon",
                    "description": "variant calling",
                    "version": "1.3",
                    "prerequisites": null,
                    "input_list": [
                        {
                            "address":
"https://hive.biochemistry.gwu.edudata/514769/dnaAccessionBased.csv",
                            "access_time": "2017-01-24T09:40:17-0500"
                        }
                    ],
                    "output_list": [
                        {
                            "address":
"https://hive.biochemistry.gwu.edudata/514801/SNPProfile.csv",
                            "access_time": "2017-01-24T09:40:17-0500"
                        },
                        {
                            "address":
"https://hive.biochemistry.gwu.edudata/14769/allCount-aligned.csv",
                            "access_time": "2017-01-24T09:40:17-0500"
                        }
                    ]
                }
            ]
```


```
        }
    },
    "execution_domain": {
        "script_access_type": "text",
        "script":
["https://example.com/workflows/antiviral_resistance_detection_hive.py"],
        "script_driver": "manual",
        "pipeline_version": "2.0",
        "platform": "hive",
        "software_prerequisites": [
            {
                "name": "HIVE-hexagon",
                "version": "babajanian.1",
                "uri": {
                    "address":
"https://hive.biochemistry.gwu.edu/dna.cgi?cmd=dna-hexagon&cmdMode=-",
                    "access_time": "2017-01-24T09:40:17-0500",
                    "sha1_chksum": null
                }
            },
            {
                "name": "HIVE-heptagon",
                "version": "albinoni.2",
                "uri": {
                    "address":
"https://hive.biochemistry.gwu.edu/dna.cgi?cmd=dna-heptagon&cmdMode=-",
                    "access_time": "2017-01-24T09:40:17-0500",
                    "sha1_chksum": null
                }
            }
        ],
        "domain_prerequisites": [
            {
                "name": "HIVE",
                "url": "https://hive.biochemistry.gwu.edu/dna.cgi?cmd=login"
            },
            {
                "name": "access to ftp",
                "url": "ftp://:22/"
            },
            {
                "name": "access to e-utils",
                "url": "http://eutils.ncbi.nlm.nih.gov/entrez/eutils/"
            },
                {
```


```
                        "name": "generic name",
                       "url": "protocol://domain:port/application/path"
                        }
        ],
        "env_parameters": {
            "key": "HOSTTYPE",
            "value" : "x86_64-linux"
        }
    },
    "parametric_domain": {
        "HIVE-hexagon" : {
            "seed": 14,
            "minimum_match_len": 66,
            "divergence_threshold_percent": 0.30
        },
        "HIVE-heptagon": {
            "minimum_coverage": 15,
            "freq_cutoff": 0.10
        }
    },
    "io_domain": {
        "input_subdomain": {
            "subject": [
                {
                    "name": "Hepatitis C virus genotype 1",
                    "source": {
                        "address":
"http://www.ncbi.nlm.nih.gov/nuccore/22129792",
                        "access_time": "2017-01-24T09:40:17-0500"
                    }
                },
                {
                    "name": "Hepatitis C virus type 1b complete genome",
                    "source": {
                        "address":
"http://www.ncbi.nlm.nih.gov/nuccore/5420376",
                        "access_time": "2017-01-24T09:40:17-0500"
                    }
                },
                {
                    "name": "Hepatitis C virus (isolate JFH-1) genomic RNA",
                    "source": {
                        "address":
"http://www.ncbi.nlm.nih.gov/nuccore/13122261",
                        "access_time": "2017-01-24T09:40:17-0500"
```

```
                    }
                },
                {
                    "name": "Hepatitis C virus clone J8CF, complete genome",
                    "source": {
                        "address":
"http://www.ncbi.nlm.nih.gov/nuccore/386646758",
                        "access_time": "2017-01-24T09:40:17-0500"
                    }
                },
                {
                    "name": "Hepatitis C virus S52 polyprotein gene",
                    "source": {
                        "address":
"http://www.ncbi.nlm.nih.gov/nuccore/295311559",
                        "access_time": "2017-01-24T09:40:17-0500"
                    }
                }
            ],
            "query": [
                {
                    "name": "HCV1a_drug_resistant_sample0001-01",
                    "source": {
                        "address": "https://hive.biochemistry.gwu.edunuc-
read/514682",
                        "access_time": "2017-01-24T09:40:17-0500"
                    }
                },
                {
                    "name": "HCV1a_drug_resistant_sample0001-02",
                    "source": {
                        "address": "https://hive.biochemistry.gwu.edunuc-
read/514683",
                        "access_time": "2017-01-24T09:40:17-0500"
                    }
                }
            ]
        },
        "output_subdomain": [
            {
               "mediatype": "text/csv",
                "source": {
                    "address":
"https://hive.biochemistry.gwu.edudata/514769/dnaAccessionBased.csv",
                    "access_time": "2017-01-24T09:40:17-0500"
```

```
                }
            },
            {
               "mediatype": "text/csv",
                "uri": {
                    "address":
"https://hive.biochemistry.gwu.edudata/514801/SNPProfile*.csv",
                    "access_time": "2017-01-24T09:40:17-0500"
                }
            }
        ]
    },
    "error_domain": {
        "empirical_error": {
            "false negative alignment hits": "<0.0010",
            "false discovery": "<0.05"
        },
        "algorithmic_error": {
            "false positive mutation calls discovery": "<0.0005",
            "false_positive_mutation_calls_discovery": "<0.00005",
            "false_discovery": "0.005"
        }
    }
}
```

3.5 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 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/
```

For instance, the inline CURIE [taxonomy:31646] expands to http://identifiers.org/taxonomy/31646 as the namespace "taxonomy" corresponds to the prefix `http://identifiers.org/taxonomy/` to be augmented with the identifier "31646". Resolving the resulting URI will redirect (currently to https://www.ncbi.nlm.nih.gov/Taxonomy/Browser/wwwtax.cgi?mode=Info&id=31646) showing that term [taxonomy:31646] means "Hepatitis C virus subtype 1a" in the NCBI Taxonomy browser. Note that some identifier patterns result in a repetition when combined with the prefix, e.g. `[so:0000667]` expands to http://identifiers.org/so/SO:0000667

3.6 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.

4 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

5 Compatibility

5.1 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) 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

6 Acknowledgements

This document began development during the 2017 HTS-CSRS workshop. The discussion during the workshop facilitated the refinement and completion of this document. The workshop participants were a major part of the initial BCO community, and the comments and suggestions collected during the sessions were incorporated into Version 1.0. This work was continued through 2018 and culminated in this document, Version 1.2. The people who participated in the creation of this