Consent API v1

Ontology Autocomplete Service

Method	`GET`
Path	`/autocomplete`
Query Parameters	<table><tr><td>q</td><td>The query term (word fragment) which the service should try and complete</td></tr><tr><td>types</td><td>An optional list (comma-separated) of term types; if specified, only those types will be searched for autocomplete suggestions. If left unspecified, all available types will be searched. Available values for this service will initially be - disease - organization </td></tr><tr><td>count</td><td>An optional limit on the number of autosuggested results that are returned.</td></tr></table>
Success Response Code	200 (Ok)
Error Response Codes	None
Response Body	[AutocompleteResponse*] AutoComplete Response := { “id” : <string>, “label” : <string>,

	“definition” : <string>, “synonyms”: [<string>*] }

Consent Ingest

Method	`PUT`
Path	`/consent`
Request Headers	<table><tr><td>`Accept`</td><td>`application/json`</td></tr><tr><td>`Content-Type`</td><td>`application/json`</td></tr></table>
Request Body	A SampleConsent resource, see below for full grammar.
Success Response Code	<table><tr><td>201 (Created)</td><td>The consent has been created.</td></tr></table>
Error Response Codes	<table><tr><td>403 (Forbidden)</td><td>User is not allowed to create a consent</td></tr></table>
Response Headers	<table><tr><td>`Location`</td><td>URL of the created consent document.</td></tr><tr><td>`Content-Type`</td><td>`application/json`</td></tr></table>
Response Body	The SampleConsent resource which was created, and which (should be) accessible by a GET on the URL present in the Location header.

Creating a Consent requires a PUT, where the body of the request contains the SampleConsent of the consent to be created. The system will assign a unique identifier (in the form of a new URL) for the consent, and return that identifier within the Location header of the response upon successful Consent creation. Future updates and retrievals of the consent (see below) will require this URL.

SampleConsent and UseRestriction Grammars

A SampleConsent is a rendering of an OWL class, and a manual review flag, into JSON, using the following grammar:

```
SampleConsent := {
  "restriction": UseRestriction,
  "requiresManualReview": <boolean>
}
```

A SampleConsent expression represents the consent attached to a single sample or sample set. The "restriction" field contains the structured use restriction, and expression

```
UseRestriction := OrExpression
  | AndExpression
  | NotExpression
  | SomeRestrictionExpression
  | OnlyRestrictionExpression
  | NamedExpression
  | EverythingExpression
  | NothingExpression

OrExpression := {
  "type": "or",
  "operands": [ UseRestriction* ]
}

AndExpression := {
  "type": "and",
  "operands": [ UseRestriction* ]
}

NotExpression := {
  "type": "not",
  "operand": UseRestriction
}

SomeRestrictionExpression := {
  "type": "some",
  "property": <string>,
  "object": UseRestriction
}
```

```
OnlyRestrictionExpression := {
  "type": "only",
  "property": <string>,
  "object": UseRestriction
}

NamedExpression := {
  "type": "named",
  "name": <string>
}

EverythingExpression := {
  "type": "everything"
}

NothingExpression := {
  "type": "nothing"
}
```

Using the Grammar

Any valid expression in the UseRestriction grammar can be submitted to the Consent Ingest service as input – however, correct use of the core consent.owl ontology as well as the imported, external ontologies (DOID, SYMP, etc.) will be required in order for automatic consent matching and retrieval to work in the final system.

One of the simplest use expressions,
`{ "type": "everything" }`

represents a use restriction which *contains no restrictions at all* -- the use of samples annotated with this use restriction would always be allowed, no matter the intent of the use.

Conversely, the use expression,
`{ "type": "nothing" }`

represents a restriction which is completely closed -- no secondary use of the samples is permitted at all.

Intermediate forms of use restriction are usually represented using the NamedExpression, SomeRestrictionExpression, and the boolean operators (And, Or, Not).

For example, a sample which was only available for use in cancer research might be annotated with the expression,

```
{
  "type": "some",
  "property": "http://broadinstitute.org/ontology/consent/research_on",
  "object": {
    "type": "named",
    "name": "DOID:162"
  }
}
```

Here, the inner "named" expression names DOID:162, which is the term from the Disease Ontology for "cancer." The definition for DOID:162 begins "A disease of cellular proliferation..." and, since a *consent* is clearly not a disease of cellular proliferation, we don't use the term directly -- rather, we wrap it in a SomeRestrictionExpression, which tells us that we're talking about research which is *studying* such diseases.

The URL

```
http://broadinstitute.org/ontology/consent/research_on
```

is the name for this "research on" relation. We use the "some" restriction type (rather than the other available type, "only") to indicate that research which at least studies cancer is acceptable, even if it also studies other kinds or forms of disease. For example, research which studied BOTH cancer and diabetes would be an acceptable use of a sample annotated with this use restriction.

If we wanted to indicate that the research's *sole* purpose must be "cancer" in order for its use of a sample with this restriction to be acceptable, we could use the "only" type (instead of "some") in this use restriction expression instead.

Consent Update

Method	`POST`
Path	The URL of the consent (e.g. the value in the Location header of the response from the Consent Ingest service)
Request Headers	`Accept`: `application/json` `Content-Type`: `application/json`
Request Body	SampleConsent
Success Response Code	200 (Ok): The consent has been updated.

Error Response Codes	403 (Forbidden) — User is not allowed to update the consent
Response Headers	`Content-Type` — `application/json`
Response Body	SampleConsent

Updating a Consent should be straightforward: the user simply POSTs the updated form of the SampleConsent to the URL of the existing consent, and the expression (if valid) replaces the old expression. The updated value of the consent expression is returned in the body of the response, and future GETs to the consent's URL should also return the same expression.

Consent Retrieval

Method	`GET`
Path	The URL of the consent (e.g. the value of the Location header returned by the Consent Ingest service)
Request Headers	`Accept` — `application/json`
Success Response Code	200 (Ok) — The consent representation is present in the body of the response.
Error Response Codes	403 (Forbidden) — User is not allowed to view consent 404 (Not Found) — The URL doesn't name a valid consent.
Response Headers	`Content-Type` — `application/json`
Response Body	SampleConsent

Each Consent is identified by its URL (returned as the value of the Location header in the response to the Consent Ingest service call that originally created the consent). Executing a GET on the URL will return the latest representation of the consent.