# OOR / Ontohub API



## Federation and general

General remark: Any ontology and any link can be optionally accompanied by a version id.

Ontology ids are instance specific

### Logic Services

* List all ontology languages () => (list (language id:name))
* List all supported logics of language (language id) => (list (logic id:name))
* List all serializations of language (langauge id) => (list (serial.id:name))
* List all logic translation () => (list (logic-translation id:name))
* List all logic translation with source (logic id) => (list (logic-translation id:name))
* List all logic translation with target (logic id) => (list (logic-translation id:name))
* List all ontology language translations
* List all ontology language translations with a given source
* List all ontology language translations with a given target

### Ontology Services

vid = version id

sid = serialization id

blue = formalized in OORService (see below)

pink = not formalized in OORService

 **Method Name**

* List all ontology latest versions find Latest OntologyVersions
* Get ontology latest version index Ontology derived
* Get ontology version metadata index OntologyVersion
* Get ontology symbols and sent. find Ontology SymbolsAndSentences derived
* Get o. version symbols and sent. find OntologyVersion SymbolsAndSentences
* Get ontology latest version metadata get Ontology Metadata derived
* Get an ontology version file get OntologyVersion File
* Get the ontology last version file get Ontology File derived
* Get metrics for an ontology version get OntologyVersion Metrics
* Get all ns prefixes of ontology get Ontology Prefix
* List all ontology categories list Categories
* List all ontologies for a category find Category Ontologies
* List all ontology-using groups list Groups
* List all ontologies, given a language find Language Ontologies
	+ (the language can be DOL, in the case, list all distributed ontologies)
* Given a distributed ontology, list all component ontologies and links
* Given an ontology or link, list all distributed ontologies it belongs to
* Get all comments/notes/proposals of an ontology
* Add a comments/notes/proposals to an ontology

## Not formalized in OORService

 index OntologyVersion (ontology[id,vid]) => (ontology)

 find Ontology SymbolsAndSentences

 (ontology[id]) => (list(symbol), list(sentence))

 find OntologyVersion SymbolsAndSentences

 (ontology[id,vid]) => (list(symbol), list(sentence))

 get Ontology Prefix (ontology[id]) => (prefix)

 list Categories () => (list (category[id,name])

 find Category Ontologies (category[id]) => (list (ontology[id,name])

 list Groups () => (list (group[id,name])

 find Language Ontologies (language[id]) => (list (ontology[id,name])

 upload OntologyVersion (ontology[id],file) => (vid)

download OntologyVersion (ontology[id,vid]) => (file)

**Formalized in OORService**

# not to be implemented in our system

**Ontology**

 find Ontology (name-fragment) => (list (ontology[id,name]))

 create Ontology (ontology) => (ontology[id])

 index Ontology (ontology[id]) => (ontology)

 update Ontology (ontology) => ()

 delete Ontology (ontology[id]) => ()

 get Ontology*Version* Metrics (ontology[id,vid]) => (metrics)

 update Ontology*Version* Metrics (ontology[id,vid],metrics) => ()

 extract Ontology*Version* Metrics (ontology[id,vid]) => (metrics)

 get Ontology*Version* File (ontology[id,vid,sid]) => (file)

 find Latest OntologyVersions

 find Latest ActiveOntologyVersions

**Note/Comment**

 get AllNotes ForOnto (ontology[id]) => (list (note))

 get AllNotes ForOnto ByAuthor (o[id], author[id]) => (list (note))

 get AllNotes ForConcept (o[id], concept[id]) => (list (note))

 get AllNotes ForIndividual (o[id], indiv.[id]) => (list (note))

 get AllNotes ForNote (o[id], note[id]) => (list (note))

 create Note

 update Note

 archive Note

 delete Note

unarchive Note

 get Note Bean

 get RootNote

 archive Thread

unarchive Thread

**Project**

 **c**reate Project

 **r**etrieve Project

 **u**pdate Project

 **d**elete Project

**Review**

 **c**reate Review

 **r**etrieve Review

 **u**pdate Review

 **d**elete Review

 get Reviews ForOnto

**Rating**

 **c**reate Rating

 **u**pdate Rating

 **d**elete Rating

 get AllRatingTypes

 retrieve RatingType

**Finding Commands**

 find OntologyOrView => find Ontology

 find LatestActiveOntologyVersions => find LatestActiveOntologyVersions

 find LatestOntologyVersions => find LatestOntologyVersions

cleanupOntologyCategory

getOntologyFile

### Mapping Services

* Get a single mapping by its id. Return type of mapping and list of mapping elements
* Get a list of mappings filtered by parameters
* Get a list of mappings for a symbol
* Get a list of mappings between two symbols
* Get a list of mappings for an ontology
* Get a list of mappings between two ontologies
* Create a new mapping
* Update a Mapping
* Delete a Mapping
* Mapping Statistics
* Get Recent Mappings
* Get Number of Mappings To/From Given Ontology
* Get Number of Mappings to Terms in Given Ontology
* Get Number of Mappings by Users for a Given Ontology

## Parsing and Static analysis

* Get all kinds of symbols (for a given ontology language),
* Parse an ontology file and get all symbols and axioms (in a specific language)
* Parse a DOL file and get all ontologies and links of the distributed ontology (this implicitly includes computation of ontologies specified by the DOL structuring constructs, e.g. ontology combinations)
* Translate an ontology along a logic or language translation

## Search

There is only one method (search), having the following parameters:

* search string (with Boolean operators and wildcards, e.g. "foo bar -baz" will expand to "foo\* AND bar\* AND NOT baz\*")
* ontologyids=<ontologyid>,<ontologyid>… - limits the search to specific ontologies (default: all ontologies)
* searchontologynames=[1/0] – search in the ontology names (default: 1)
* searchsymbolnames=[1/0] – search in the symbol names (default: 1)
* isexactmatch=[1/0] – match the entire ontology resp. symbol name (default: 0)
* pagesize=<pagesize> - the number of results to display in a single request (default: all)
* pagenum=<pagenum> - the page number to display (pages are calculated using <total results>/<pagesize>) (default: 1)
* maxnumhits=<maxnumhits> - the maximum number of top matching results to return (default: 1000)
* symbolkinds=<kind,kind,..> - limits the results returned to these kinds, multitple kinds can be included in the parameter.
* includedefinitions={true} - if a search result is a hit for a symbol, adding this parameter will include the definition in the search result xml.

## Persistence

* Synchronize two repositories (also non-git ones, like triple stores)

### Difference

createDiff

createDiffForLatestActiveOntologyVersionPair

createDiffForAllActiveVersionsOfOntology

getAllDiffsForOntology

getDiffFileForOntologyVersions

## Local Inference

* get available inference tools by name, language/logic, type (prover, model finder, conservativity checker, module extractor) and input parameters (including options)
* prove open goals in an ontology. Output: list of used axioms, proof, proof status using SZS ontology http://tinyurl.com/szsontology
* check consistency / find model of an ontology. Output: model, represented by symbols + axioms
* disprove open goals in an ontology. Output: see find model
* check conservativity of a link. Output: conservativity status (NotCons, DontKnow, Cons, Mono, Def)
* module extraction for an ontology w.r.t. a subsignature=list of symbols and an extraction algorithm

## Distributed Inference

Open questions: should we use Hets development graph sessions, or send around (updates to) distributed ontologies (consisting of ontologies and links)? The latter needs a distributed ontology closed under the import/definition link relation - this can become quite large. (Should we therefore allow distributed ontologies to import other distributed ontologies?) Should a new proof state of a distributed ontology constitute a new version, possibly with new version of the

involved ontologies and links?

What happens with proof goals generated e.g. by interpretations? Can they spoil the target ontology?

How can we avoid that Hets needs to read in every new version of the distributed ontology from scratch? (Hets would need a caching mechanism.)

Here is the session based API:

* POST /libraries/<coded-iri>/sessions - create a new proof session for development graph
* GET /sessions/<id>?format=<f> - get proof state of session
* GET /menus - Get development graph menu structure
* GET /nodes/<coded-iri>?library=<coded-iri>&session=id - Get info for node
* GET /nodes/<coded-iri>/theory?library=<coded-iri>&session=id - Get theory of node
* GET /edges/<coded-iri>?library=<coded-iri>&session=id - Get info for edge
* PUT /libraries/<coded-iri>/proofs/<id>/<command> - execute command for session
* PUT /sessions/<id>/<command>?node=<iri>&edge=<iri>- execute command for node in session
* GET /sessions/<id>/provers?node=<iri>&translation=<iri> - Get provers for node
* GET /sessions/<id>/translations?node=<iri> - Get logic translations for node
* PUT /sessions/<id>/prove?node=<iri>?prover=<name>&translation=<iri> &timeout=<secs>&include=true - Call prover

List of available Hets commands (which ones do we need here?)

dg-all auto Apply automatic tactic - needed

dg-all glob-decomp Apply rule global-decomposition - to start with, auto should suffice

dg-all global-subsume Apply rule global-subsumption - to start with, auto should suffice

dg-all loc-decomp Apply rule local-decomposition - to start with, auto should suffice

dg-all local-infer Apply rule local-inference - to start with, auto should suffice

dg-all comp prove composed edges - to start with, auto should suffice

dg-all comp-new create composed proven edges - to start with, auto should suffice

dg-all cons Apply rule conservativity - to start with, auto should suffice

dg-all hide-thm Apply rule hide-theorem-shift - to start with, auto should suffice

dg-all thm-hide Apply rule theorem-hide-shift - to start with, auto should suffice

compute-colimit compute colimit - not needed, since this is called by static analysis of “combine”

compute-normal-form Compute normal forms for nodes with incoming hiding links - needed for proving in presence of hiding

triangle-cons triangle-cons - needed

freeness freeness - not needed in DOL

flattening importing Flatten all theories and delete all importing links - needed for interfacing to standard theorem provers

flattening disjoint-union Create intersection nodes and ensure only disjoint unions - needed for interfacing to some (but not many) theorem provers

flattening renaming Flatten out renaming - needed for interfacing to some (but not many) theorem provers

flattening hiding Delete all hiding links - needed for interfacing to some (but not many) theorem provers

flattening heterogeneity Flatten out heterogeneity - needed for interfacing to some (but not many) theorem provers

qualify-all-names Qualify and disambiguate all signature names

undo Undo last change - not needed

redo Redo last change - not needed

use <File> Read HetCASL file - not needed

dg basic <Nodes> Select node - needed

translate <Comorphism> Choose translation - needed

prover <Prover> Choose prover - needed

set goals <Goal> Set goal - needed

prove Applies selected prover to selected goals - needed

check-consistency check consistency - needed

drop-translations Drops any selected comorphism - needed

cons-checker <ConsChecker> Choose consistency checker - needed

conservativity-check <Edges> Choose conservativity checker - needed

set time-limit <Number> Set the time-limit for the next proof - needed

set axioms <Axiom> Set the axioms used for the next proof - needed

set include-theorems true Include proven theorems - needed

set include-theorems false Do not include proven theorems - needed

nodes Show Nodes - not needed

edges Show Edges - not needed

show-undo-history Show Undo-History - not needed

show-redo-history Show Redo-History - not needed

show-proven-goals-current Show Proven Goals of selected node - needed

show-unproven-goals-current Show Unproven Goals of selected node - needed

show-all-axioms-current Show All Axioms of selected node - needed

show-all-goals-current Show All Goals of selected node - needed

show-computed-theory-current Show Computed Theory of selected node - needed

show-taxonomy-current Show Taxonomy of selected node - not needed

show-concept-current Show Concept of selected node - not needed

show-node-info-current Show Node-Info of selected node - needed

show-node-info <Nodes> Show Node-Info - needed

show-computed-theory <Nodes> Show Computed Theory - needed

show-all-goals <Nodes> Show All Goals - needed

show-proven-goals <Nodes> Show Proven Goals - needed

show-unproven-goals <Nodes> Show Unproven Goals - needed

show-all-axioms <Nodes> Show All Axioms - needed

show-taxonomy <Nodes> Show Taxonomy - not needed

show-concept <Nodes> Show Concept - not needed

show-edge-info <Edges> Show Edge-Info - needed

expand Extend current node - ???

addview Add a view - ???

help Show all available commands - see DG menus?

quit Quit - not needed

Here is an API for sending around updates:

* prove link. Input: IRI of link. Output: list of new links and/or proof goals for simple ontologies that will prove the link

## Evaluation and other services

### OOPS! and similar services

we propose the following abstraction from the OOPS! API:

input: ontology[[1]](#footnote-0)

output: list of response elements of the following form:

type (for OOPS: pitfall, warning, suggestion)

code (an integer)

name

description

list of involved symbols[[2]](#footnote-1)

### Annotator Service

This service it specific to bio ontologies. How to generalise it to other domains? It seems that some (more static) list of service types and (more dynamically growing) list of actual services (conforming to these service types) would be useful. This of course also includes services like OOPS!

### Ontology Recommender

Interesting challenge to generalise this to ontologies written in arbitrary languages...

### Resource Index Service

could be adapted for Ontohub, if "concept" is replaced by "symbol"

### Notes Service (Term Proposals and Comments)

### Logic-specific services

OWL specific services involving the class hierarchy

These services could also be used for other languages if there is a suitable projection to OWL.

## Remaining stuff from OOR

 find AllOntologyOrViewVersionsByVirtualId#

 find LatestAutoPulledOntologyVersions#

 find LatestActiveOntologyOrViewVersion#

 find LatestActiveOntologyViewVersions#

 find LatestOntologyOrViewVersion#

 find LatestOntologyViewVersions#

1. OOPS! has more inputs, but we let the list of pitfalls empty, and the output format be XML. [↑](#footnote-ref-0)
2. OOPS! outputs structured XML elements that may contain multiple n-ary relations between symbols (e.g. oops:MightBeEquivalentProperty and oops:MightBeEquivalentAttribute). We prefer to have only one such relation per response element. [↑](#footnote-ref-1)