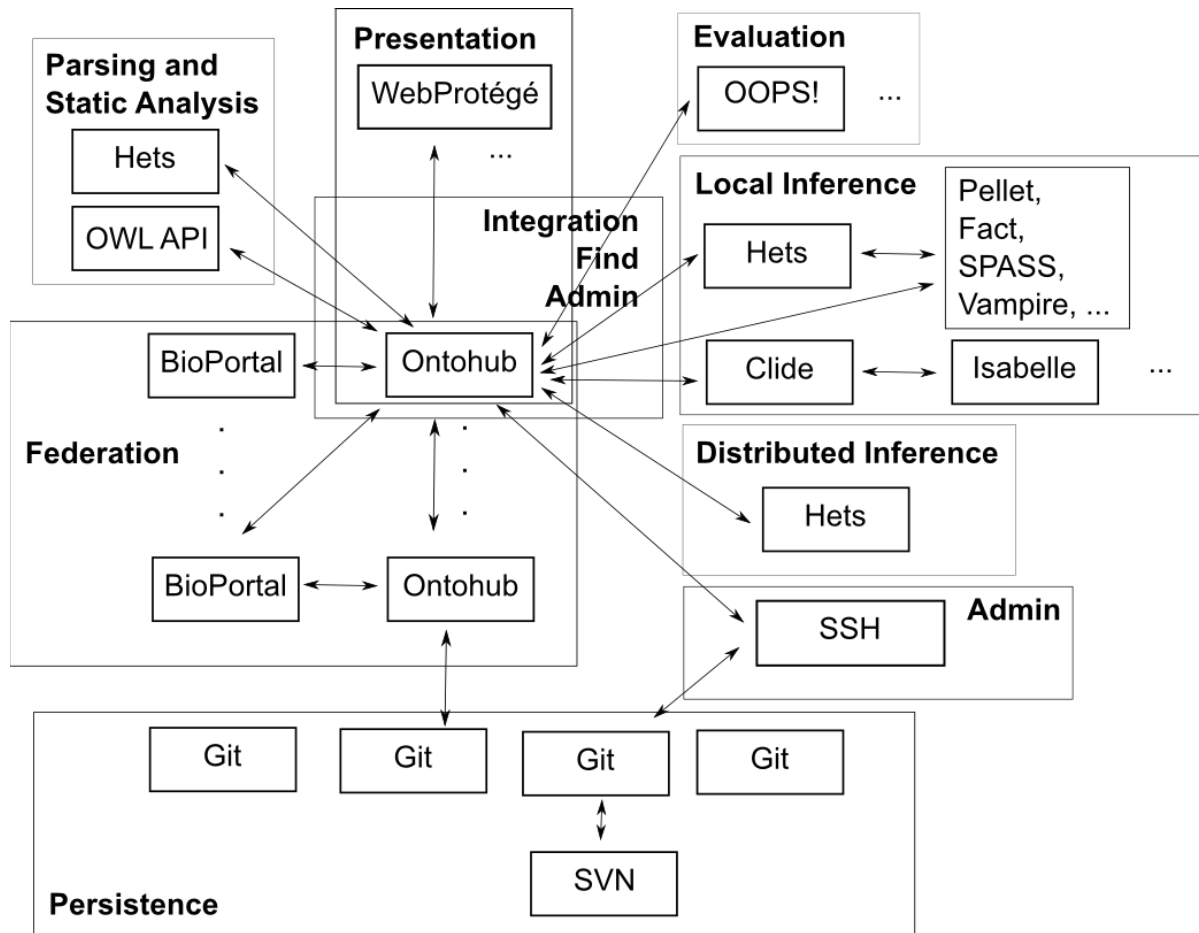


## OOR / Ontohub API



### 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 (language 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

### Parameters

- List all ontology latest versions () => (list (ontology id:name, version id))
- Get ontology latest version (ontology id) => (version id)
- Get ontology version metadata (ontology id, version id) => (list (metadata), list (categories)), also language and logic
- Get ontology version symbols and sentences (including status of sentences)
- Get ontology latest version metadata(ontology id) => (list (metadata)) **derived**
- Get an ontology version file (ontology id, version id, serial. id) => (file)
- Get the ontology last version file (ontology id, serial. id) => (file) **derived**
- Get metrics for an ontology (ontology id) => (list (metrics))
- Get all ns prefixes of ontology(ontology id) => (list (prefix))
- List all ontology categories () => (list (category id:name))
- List all ontology for a category (category id) => (list (ontology id:name))
- List all ontology-using groups () => (list (group id:name))
- List all ontologies, given a language (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

### Taken from OORService

\* not specified in the 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 Metrics	(ontology[id])	=> ()
update Ontology Metrics	(ontology[id])	=> ()
extract Ontology Metrics	(ontology[id])	=> ()
upload Ontology Version*	(ontology[id], file)	=> (version[id])
download Ontology Version*	(o[id], v[id])	=> (file)
get Ontology File	(o[id], v[id])	=> (file)
find LatestOntologyVersions		
find LatestActiveOntologyVersions		

#### 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**

```
create Project
retrieve Project
update Project
delete Project
```

### **Review**

```
create Review
retrieve Review
update Review
delete Review
    get Reviews ForOnto
```

### **Rating**

```
create Rating
update Rating
delete 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, multiple 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, 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 question: should we use Hets development graph sessions, or only send around updates to ontologies and links?

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 - <b>needed</b>
dg-all glob-decomp	Apply rule global-decomposition - <b>to start with, auto should suffice</b>
dg-all global-subsume	Apply rule global-subsumption - <b>to start with, auto should suffice</b>
dg-all loc-decomp	Apply rule local-decomposition - <b>to start with, auto should suffice</b>
dg-all local-infer	Apply rule local-inference - <b>to start with, auto should suffice</b>
dg-all comp	prove composed edges - <b>to start with, auto should suffice</b>

dg-all comp-new		create composed proven edges - <b>to start with, auto should suffice</b>
dg-all cons		Apply rule conservativity - <b>to start with, auto should suffice</b>
dg-all hide-thm		Apply rule hide-theorem-shift - <b>to start with, auto should suffice</b>
dg-all thm-hide		Apply rule theorem-hide-shift - <b>to start with, auto should suffice</b>
compute-colimit		compute colimit - <b>not needed, since this is called by static analysis</b>
		<b>of "combine"</b>
compute-normal-form		Compute normal forms for nodes with incoming hiding links - <b>needed for proving in presence of hiding</b>
triangle-cons		triangle-cons - <b>needed</b>
freeness		freeness - <b>not needed in DOL</b>
flattening importing		Flatten all theories and delete all importing links - <b>needed for interfacing to standard theorem provers</b>
flattening disjoint-union		Create intersection nodes and ensure only disjoint unions - <b>needed for interfacing to some (but not many) theorem provers</b>
flattening renaming		Flatten out renaming - <b>needed for interfacing to some (but not many) theorem provers</b>
flattening hiding		Delete all hiding links - <b>needed for interfacing to some (but not many) theorem provers</b>
flattening heterogeneity		Flatten out heterogeneity - <b>needed for interfacing to some (but not many) theorem provers</b>
qualify-all-names		Qualify and disambiguate all signature names
undo		Undo last change - <b>not needed</b>
redo		Redo last change - <b>not needed</b>
use	<File>	Read HetCASL file - <b>not needed</b>
dg basic	<Nodes>	Select node - <b>needed</b>
translate	<Comorphism>	Choose translation - <b>needed</b>
prover	<Prover>	Choose prover - <b>needed</b>
set goals	<Goal>	Set goal - <b>needed</b>
prove		Applies selected prover to selected goals - <b>needed</b>
check-consistency		check consistency - <b>needed</b>
drop-translations		Drops any selected comorphism - <b>needed</b>
cons-checker	<ConsChecker>	Choose consistency checker - <b>needed</b>
conservativity-check	<Edges>	Choose conservativity checker - <b>needed</b>
set time-limit	<Number>	Set the time-limit for the next proof - <b>needed</b>
set axioms	<Axiom>	Set the axioms used for the next proof - <b>needed</b>
set include-theorems true		Include proven theorems - <b>needed</b>
set include-theorems false		Do not include proven theorems - <b>needed</b>
nodes		Show Nodes - <b>not needed</b>
edges		Show Edges - <b>not needed</b>
show-undo-history		Show Undo-History - <b>not needed</b>
show-redo-history		Show Redo-History - <b>not needed</b>
show-proven-goals-current		Show Proven Goals of selected node - <b>needed</b>
show-unproven-goals-current		Show Unproven Goals of selected node - <b>needed</b>

show-all-axioms-current		Show All Axioms of selected node - <b>needed</b>
show-all-goals-current		Show All Goals of selected node - <b>needed</b>
show-computed-theory-current		Show Computed Theory of selected node - <b>needed</b>
show-taxonomy-current		Show Taxonomy of selected node - <b>not needed</b>
show-concept-current		Show Concept of selected node - <b>not needed</b>
show-node-info-current		Show Node-Info of selected node - <b>needed</b>
show-node-info	<Nodes>	Show Node-Info - <b>needed</b>
show-computed-theory	<Nodes>	Show Computed Theory - <b>needed</b>
show-all-goals	<Nodes>	Show All Goals - <b>needed</b>
show-proven-goals	<Nodes>	Show Proven Goals - <b>needed</b>
show-unproven-goals	<Nodes>	Show Unproven Goals - <b>needed</b>
show-all-axioms	<Nodes>	Show All Axioms - <b>needed</b>
show-taxonomy	<Nodes>	Show Taxonomy - <b>not needed</b>
show-concept	<Nodes>	Show Concept - <b>not needed</b>
show-edge-info	<Edges>	Show Edge-Info - <b>needed</b>
expand		Extend current node - <b>???</b>
addview		Add a view - <b>???</b>
help		Show all available commands - <b>see DG menus?</b>
quit		Quit - <b>not needed</b>

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<sup>1</sup>

output: list of response elements of the following form:

type (for OOPS: pitfall, warning, suggestion)

code (an integer)

name

description

list of involved symbols<sup>2</sup>

<sup>1</sup> OOPS! has more inputs, but we let the list of pitfalls empty, and the output format be XML.

<sup>2</sup> 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.

## Annotator Service

This service is 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#
```