OpenTox REST Application Programming Interface

OpenTox
EC FP7 funded project, aims to develop distributed framework for predictive toxicology. The building blocks considered are: data, chemical structures, algorithms and models. The framework allows to build models, apply models, validate models, access and query data in various ways. Technologies used are REST style web services and W3C Resource Description Framework for description of services.

Information about OpenTox is available at [http://opentoxt.org](http://opentoxt.org)

API = Application Programming Interface
- The way applications talk to each other
- The way developers talk to applications

Representational State Transfer (REST)

REST is a software architecture style, defined by Roy Fielding in his PhD thesis (2000). There is a multitude of information on RESTful design principles, development frameworks and examples. For a start, the following references could be recommended:

- Chapter 5 of Fielding's dissertation
- Leonard Richardson, Sam Ruby, RESTful Web Services, O'Reilly 2007
- From SOA to REST: Designing and Implementing RESTful Services, Tutorial at WWW2009
- How Do I Model State? Let Me Count the Ways, ACM Queue 2009
- rest-discuss Yahoo group
- Richardson Maturity Model: steps toward the glory of REST

Contrary to established WS-SOAP standards, there are no (currently) standards for RESTful applications, but merely design guides. Perhaps the first move towards standardization is the upcoming (First International Workshop on RESTful Design WS-REST 2010) to be held at WWW 2010 in Raleigh, North Carolina, USA next 26 April 2010.

Besides all discussions, setting REST against SOAP, this kind of comparison is not entirely correct, for SOAP is a protocol, and REST is an architectural style, not a protocol.

**SOAP vs REST**

REST style can be briefly summarized as:
Design principles

**Resource oriented**

Every object (resource) is named and addressable (e.g. HTTP URL)

Example: [http://perfsonarservice.net/MeasurementArchive/interface/interface_identifier](http://perfsonarservice.net/MeasurementArchive/interface/interface_identifier)

RESTful API design starts by identifying most important objects and groups of objects, supported by the software system and proceeds by defining URL patterns. Common patterns are:

- Returns list of objects in some format [http://myservice.net/myobject](http://myservice.net/myobject)
- Returns representation(s) of an object, identified by {myobjectid} [http://myservice.net/myobject/{myobjectid}](http://myservice.net/myobject/{myobjectid})
- Patterns may be nested, e.g. [http://myservice.net/myobject/{myobjectid}/details/{detailsid}](http://myservice.net/myobject/{myobjectid}/details/{detailsid})

**Transport protocol**

- HTTP is the most popular choice of transport protocol, but there are examples of systems using other protocols as well.

**Operations**

Resources (nouns) support limited number of operations (verbs). HTTP operations are the common choice, when the transport protocol is HTTP.

RESTful operations

- GET (retrieve the object under specified URL)
- PUT (update the content of an object at the specified URL)
- POST (create a new object and return the URL of the newly created resource)
- DELETE (delete the object)

All operations, except POST should be safe (no side effects) and idempotent (same effect if executed multiple times).

Non-RESTful operations

- Everything else, e.g. POST XML message to [http://myservice.net/doSmthHere](http://myservice.net/doSmthHere)

Deciding on the operation to be done, on the basis of interpreting POSTed message content (the way a typical SOAP service works!) is NOT recommended. This is referred to as "overloaded POST" and considered violation of RESTful principles!

**Resource representation**

Every resource is defined by an URI. If GET operation on a resource URI returns some content, it is regarded as "dereferencable" (effectively it is an URL). A resource may return content in
different formats (by specifying MediaType in the Accept: header of GET operation). The content is regarded as resource "representation". There could be multiple representations of the same resource (e.g. text/html or text/xml).

RESTfull API design includes specification of allowed media types for each resource/operation pair.

Hypermedia as the Engine of Application State (HATEOAS)

All resources should be reachable via a single (or minimum) number of entry points into a RESTful applications. Thus, a representation of a resource should return hypermedia links to related resources

- For example /dataset/{id} resource should return links to /compound/{id} and /property/{id}, used to represent the dataset content
- REST APIs must be hypertext driven

Error codes

HTTP Status codes are used to indicate an operation success or failure.

RESTfull API design includes specification of status codes for each resource/operation pair.
**Constraints**

Client-server - Clients are separated from servers by a uniform interface.

Cacheable - Clients are able to cache responses

Stateless design

No client context should be stored on the server between requests. Each request from any client contains all of the information necessary to service the request, and any state is held in the client.

Cookies are considered bad practice!

Layered

A client cannot ordinarily tell whether it is connected directly to the end server, or to an intermediary along the way. Intermediary servers may improve system scalability by enabling load balancing and by providing shared caches. They may also enforce security policies.

Uniform interface

The uniform interface between clients and servers, (HTTP GET/PUT/POST/DELETE verbs only) simplifies and decouples the architecture, which enables each part to evolve independently.

**Design goals**

- Scalability of component interactions;
- Generality of interfaces;
- Independent deployment of components;
- Intermediary components to reduce latency, enforce security and encapsulate legacy systems

RESTful design principles are advocated as being successful, for underlying the existing WWW architecture. REST application are becoming increasingly popular, the trend with major service vendors are to offer REST API along with an existing SOAP API. Some report REST API usage is increasing and SOAP API usage decreasing.

**RDF**

(description of the resources) 


Resources are described in terms of properties and property values using RDF statements.

- Statements are represented as triples, consisting of a

---

subject, predicate and object. [S, P, O]

Components
OpenTox considers the following set of entities as essential building blocks of a predictive toxicology framework:

- Chemical compound structure
- Chemical compound properties and identifiers
- Dataset of chemical compounds and various properties (measured or calculated)
- Algorithms
  - Data processing algorithms
    - Algorithms generating certain values, based on chemical structure (e.g. descriptor calculation)
    - Data preprocessing (e.g. Principal component analysis, feature selection)
    - Structure processing (e.g. structure optimization)
    - Algorithms, relating set of structures to another set of structures (e.g. similarity search or metabolite generation)
  - Machine learning algorithms
    - Supervised (e.g. Regression, Classification)
    - Unsupervised (e.g. Clustering )
  - Prediction algorithms, defined by experts (e.g. series of structural alerts, defined by human experts, not derived by learning algorithms)
- Models
  - Models are generated by applying machine learning algorithms to specific dataset, given specific parameters
  - In case of expert defined rules, models are generated by the corresponding algorithms, without requirement for a training dataset.
- Validation
  - Validation provides procedures independent of model building facilities (e.g. crossvalidation) and generates relevant statistics.
- Reports
  - Various types of reports might be generated, using building blocks listed (e.g. validation report can be generated using validation object, a model and a dataset).

In addition, the following components are introduced in order to address specific challenges in IT implementation, for example handling of time consuming calculations or access to protected resources

- Task
- Ontology service
- Authentication and authorization
Components implementation

- All components are implemented as REST web services. There could be multiple implementations of same type of components, as well as (subset of) services could be hosted by the same provider, or by multiple providers on separate locations.

Figure 1 Distributed OpenTox resources.

Components identification

All components are identified via unique web address, assigned according to the URL templates (Table 1). The host:port part depends on particular location of the resource.

Table 1 URI templates of OpenTox components

<table>
<thead>
<tr>
<th>OpenTox object</th>
<th>Description</th>
<th>URL Template (example)</th>
<th>Complete API documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compound</td>
<td>Representation of chemical compounds</td>
<td><a href="http://host:port/compound/%7Bcompoundid%7D">http://host:port/compound/{compoundid}</a></td>
<td><a href="http://opentox.org/dev/apis/api-1.1/structure">http://opentox.org/dev/apis/api-1.1/structure</a></td>
</tr>
<tr>
<td>Feature</td>
<td>Properties and identifiers</td>
<td><a href="http://host:port/feature/%7Bfeatureid%7D">http://host:port/feature/{featureid}</a></td>
<td><a href="http://opentox.org/dev/apis/api-1.1/Feature">http://opentox.org/dev/apis/api-1.1/Feature</a></td>
</tr>
<tr>
<td>Dataset</td>
<td>Encapsulates set of chemical compounds and their property values</td>
<td><a href="http://host:port/dataset/%7Bdatasetid%7D">http://host:port/dataset/{datasetid}</a></td>
<td><a href="http://opentox.org/dev/apis/api-1.1/dataset">http://opentox.org/dev/apis/api-1.1/dataset</a></td>
</tr>
<tr>
<td>Model</td>
<td>OpenTox model services</td>
<td><a href="http://host:port/model/%7Bmodelid%7D">http://host:port/model/{modelid}</a></td>
<td><a href="http://opentox.org/dev/apis/api-1.1/Model">http://opentox.org/dev/apis/api-1.1/Model</a></td>
</tr>
<tr>
<td>Algorithm</td>
<td>OpenTox algorithm services</td>
<td><a href="http://host:port/algorithm/%7Balgorithmid%7D">http://host:port/algorithm/{algorithmid}</a></td>
<td><a href="http://opentox.org/dev/apis/api-1.1/Algorithm">http://opentox.org/dev/apis/api-1.1/Algorithm</a></td>
</tr>
<tr>
<td>Validation, Report</td>
<td>A validation corresponds to the validation of a model on a test dataset.</td>
<td><a href="http://host:port/validation/%7Bvalidationid%7D">http://host:port/validation/{validationid}</a></td>
<td><a href="http://opentox.org/dev/apis/api-1.1/Validation">http://opentox.org/dev/apis/api-1.1/Validation</a></td>
</tr>
<tr>
<td>Task</td>
<td>Asynchronous jobs are handled via an intermediate Task resource. A resource, submitting an asynchronous job should return the URI of the task.</td>
<td><a href="http://opentox.org/dev/apis/api-1.1/AsyncTask">http://opentox.org/dev/apis/api-1.1/AsyncTask</a></td>
<td><a href="http://opentox.org/dev/apis/api-1.1/AsyncTask">http://opentox.org/dev/apis/api-1.1/AsyncTask</a></td>
</tr>
<tr>
<td>Ontology</td>
<td>Provides</td>
<td><a href="http://host:port/ontology">http://host:port/ontology</a></td>
<td><a href="http://opentox.org/dev/apis/api-1.1/Ontology">http://opentox.org/dev/apis/api-1.1/Ontology</a></td>
</tr>
</tbody>
</table>
service | storage and SPARQL search functionality for objects, defined in OpenTox services and relevant ontologies | -1.1/Ontology%20service
---|---|---

**Actions on components**

REST style restricts actions on resources to four HTTP verbs (GET, PUT, POST, DELETE). The common interpretation of HTTP actions in OpenTox framework is:

**Individual resources (e.g. a dataset or a model)**

- GET – retrieve representation of the resource
- PUT – update representation of the resource
- POST
  - replace representation of the resource with a new one (e.g. replace the dataset with new content)
  - Initiate calculations, based on this resource (e.g. submit dataset URI to an algorithm resource and obtain a model URI as a result)
- DELETE – delete the resource

**Collections of resources (e.g. list of all available models, or list of available datasets)**

- GET – retrieve representation of multiple resources
  - E.g. retrieve all available algorithms
- PUT - N/A
- POST – create new resource and return its URI
  - e.g. create a new dataset by submitting new dataset content to the dataset service
- DELETE – N/A
1) GET the algorithm URL to retrieve information about the algorithm and 2) POST to the algorithm URL, with dataset URL as parameter, in order to create a model. An example with command line cURL tool is shown below:

**Components representation**

All OpenTox components are defined by OWL (Web Ontology Language) ontology, available at [http://opentox.org/api/1.1/opentox.owl](http://opentox.org/api/1.1/opentox.owl). This ontology is the ultimate source, used to describe OpenTox components, relationships between them and to generate object's RDF representations. An overview of components and their relationships is presented at Figure 2. RDF representation is mandatory for all OpenTox components.

![Figure 2 Relationships between OpenTox resources, modeled in OpenTox ontology. The image is generated by Protege ontology editing software.](image)

The namespace of the current version of the OpenTox ontology is `xmlns:ot=http://www.opentox.org/api/1.1#`. All components are subclasses of `ot:OpenToxResource`. Description of every component, its usage and examples follows.

---

Compound

Provides different representations for chemical compounds with a unique and defined chemical structure.

Documentation
http://opentox.org/dev/apis/api-1.1/structure

Representation
A subclass of ot:OpenToxResource.

Example 1. Retrieve compound as smiles

```bash
$ curl -H "Accept:chemical/x-daylight-smiles" http://apps.ideaconsult.net:8080/ambit2/compound/1
O=C
```

Example 2. Retrieve compound as smiles, show HTTP headers

```bash
* About to connect() to apps.ideaconsult.net port 8080 (#0)
* Trying 93.123.36.100... connected
* Connected to apps.ideaconsult.net (93.123.36.100) port 8080 (#0)
> GET /ambit2/compound/1 HTTP/1.1
> User-Agent: curl/7.18.2 (x86_64-pc-linux-gnu) libcurl/7.18.2 OpenSSL/0.9.8g zlib/1.2.3.3 libidn/1.8 libssh2/0.18
> Host: apps.ideaconsult.net:8080
> Accept:chemical/x-daylight-smiles
>
< HTTP/1.1 200 OK
HTTP/1.1 200 OK
< Vary: Accept-Charset, Accept-Encoding, Accept-Language, Accept
Vary: Accept-Charset, Accept-Encoding, Accept-Language, Accept
< Accept-Ranges: bytes
Accept-Ranges: bytes
< Server: Restlet-Framework/2.0.06
Server: Restlet-Framework/2.0.06
< Content-Type: chemical/x-daylight-smiles;charset=UTF-8
Content-Type: chemical/x-daylight-smiles;charset=UTF-8
< Transfer-Encoding: chunked
Transfer-Encoding: chunked

<
O=C
* Connection #0 to host apps.ideaconsult.net left intact
* Closing connection #0
```

Example 3. Retrieve compound as InChI

```bash
$ curl -H "Accept:chemical/x-inchi" http://apps.ideaconsult.net:8080/ambit2/compound/1
```
InChI=1S/CH2O/c

Example 4. Retrieve compound in MOL format

```bash
$ curl -H "Accept:chemical/x-mdl-molfile" http://apps.ideaconsult.net:8080/ambit2/compound/1
CH2O
APtc1cactv09040902283D 0 0.00000 0.00000
4 3 0 0 0 0 0 0 0 0 0 9999 V2000
-0.6004 0.0000 0.0001 O 0 0 0 0 0 0 0 0 0 0 0 0
0.6072 0.0000 -0.0004 C 0 0 0 0 0 0 0 0 0 0 0 0
1.1472 0.9353 0.0016 H 0 0 0 0 0 0 0 0 0 0 0 0
1.1472 -0.9353 0.0016 H 0 0 0 0 0 0 0 0 0 0 0 0
1 2 2 0 0 0 0
2 3 1 0 0 0 0
2 4 1 0 0 0 0
```

Example 5. Retrieve compound in SDF format

```bash
curl -H "Accept:chemical/x-mdl-sdfile" http://apps.ideaconsult.net:8080/ambit2/compound/1
CH2O
APtc1cactv09040902283D 0 0.00000 0.00000
4 3 0 0 0 0 0 0 0 0 0 9999 V2000
-0.6004 0.0000 0.0001 O 0 0 0 0 0 0 0 0 0 0 0 0
0.6072 0.0000 -0.0004 C 0 0 0 0 0 0 0 0 0 0 0 0
1.1472 0.9353 0.0016 H 0 0 0 0 0 0 0 0 0 0 0 0
1.1472 -0.9353 0.0016 H 0 0 0 0 0 0 0 0 0 0 0 0
1 2 2 0 0 0 0
2 3 1 0 0 0 0
2 4 1 0 0 0 0
M END
$$ $$
```

Example 6. Retrieve compound in CML format

```bash
$ curl -H "Accept:chemical/x-cml" http://apps.ideaconsult.net:8080/ambit2/compound/1
<?xml version="1.0" encoding="ISO-8859-1"?><list dictRef="cdk:moleculeSet" xmlns="http://www.xml-cml.org/schema">
<molecule id="m1" title="CH2O" xmlns="http://www.xml-cml.org/schema">
<atomArray>
<atom id="a1" elementType="O" x3="-0.6004" y3="0.0" z3="1.0E-4" formalCharge="0" isotopeNumber="16"/>
<atom id="a2" elementType="C" x3="0.6072" y3="0.0" z3="-4.0E-4" formalCharge="0" isotopeNumber="12"/>
<atom id="a3" elementType="H" x3="1.1472" y3="0.9353" z3="0.0016" formalCharge="0" isotopeNumber="1"/>
<atom id="a4" elementType="H" x3="1.1472" y3="-0.9353" z3="0.0016" formalCharge="0" isotopeNumber="1"/>
</atomArray>
<bondArray>
<bond id="b1" atomRefs2="a1 a2" order="D"/>
```
Feature

A Feature is a resource, representing any kind of a property or identifier, assigned to a Compound. The feature types are determined via their links to ontologies (Feature ontologies, Descriptor ontologies, Endpoints ontologies).

Representation

Represented by ot:Feature, a subclass of ot:OpentoxResource. The following properties are supported:

- Name, defined by dc:title (Dublin Core namespace);

  <dc:title rdf:datatype="&xsd;string">XLogP</dc:title>

- Units, defined by ot:units annotation property (OpenTox namespace);
  - Creator, defined by dc:creator annotation property (Dublin Core namespace);
  - The origin of the Feature is defined by ot:hasSource object property (OpenTox namespace) element and can be ot:Algorithm, ot:Model or ot:Dataset;
  - Relations to other resources, which represent the same entity, could be established via owl:sameAs property. This approach can be used for example to link the ot:Feature resource to a resource form another ontology, as in the example.

At Figure 3, a Protégé screenshot of clases, defined in OpenTox ontology is shown, with properties of ot:Feature class displayed at the right side.

Note there are subclasses of ot:Feature (namely ot:NumericFeature, ot:StringFeature, ot:NominalFeature), which are used to denote if a feature holds numeric, nominal or string values.

```bash
<rdf:RDF
  xmlns:ot="http://www.opentox.org/api/1.1#"
  xmlns:bx="http://purl.org/net/nknouf/ns/bibtex#"
  xmlns:otee="http://www.opentox.org/echaeEndpoints.owl#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:ota="http://www.opentox.org/algorithmTypes.owl#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:rdfs="http://www.w3.org/2002/07/owl#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl"
  xml:base="http://apps.ideaconsult.net:8080/ambit2/"
  <owl:Class rdf:about="http://www.opentox.org/api/1.1#Algorithm"/>
  <owl:Class rdf:about="http://www.opentox.org/api/1.1#Feature"/>
  <owl:Class rdf:about="http://www.opentox.org/api/1.1#NumericFeature">
    <rdfs:subClassOf rdf:resource="http://www.opentox.org/api/1.1#Feature"/>
  </owl:Class>
```
Example 7 displays an OpenTox feature with title “XLogP” and identified by the URI http://apps.ideaconsult.net:8080/ambit2/feature/22114.

The algorithm used to generate values for this feature is identified by the URI http://apps.ideaconsult.net:8080/ambit2/algorith/m/org.openscience.cdk.qsar.descriptors.molecular.XLogPDescriptor. Note the URI identifies OpenTox Algorithm resource, and the algorithm URI itself cannot only be read⁴, as will be shown later in this text, and also can be used to initiate calculations of XLogP descriptor.

The link to ECHA endpoints ontology (otee: http://www.opentox.org/echaEndpoints.owl namespace) is also provided, specifying that this feature represents the physicochemical property “Octanol water partitioning coefficient”. The ECHA endpoints ontology is derived from ECHA classification of endpoints, published in REACH guidance documents, and consists of hierarchical arrangement of physicochemical properties and various toxicological endpoints. The hierarchy doesn’t represent the complexity of toxicological assays, but can be used as a first approximation to assign meaning to the data entries. As the RDF representation doesn’t restrict the usage of this particular ontology, more specific description of toxicological assays can be used as well.

The BlueObelisk ontology (http://www.blueobelisk.org/ontologies/chemoinformatics-algorithms/#xlogP) links to the description of XLogP algorithm, providing details of the algorithm itself, e.g. publication reference.

RDF supports multiple serialization formats. Example 8 shows the same information as in Example 7, but in N3 format.


```
```

⁴ Try the URL in a web browser.
Example 9 displays RDF/XML of another ot:Feature resource, again representing XLogP descriptor, but generated by different implementation. In this case, its name is “TUM_CDK_XLogP”, and the algorithm resource used to generate resides at Technical University of Munich (TUM) premises (http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/CDKPhysChem/XLogPDescriptor). This algorithm URL could also be used to initiate descriptor calculations.


```json
@prefix ot: <http://www.opentox.org/api/1.1#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix : <http://apps.ideaconsult.net:8080/ambit2/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

ot:Algorithm
   a      owl:Class .

ot:hasSource
   a      owl:ObjectProperty .

ot:units
   a      owl:DatatypeProperty .

af:22114
   a      ot:Feature , ot:NumericFeature ;
   dc:creator "http://www.blueobelisk.org/ontologies/chemoinformatics-algorithms/#xlogP" ;
   dc:title "XLogP" ;
   ot:hasSource
   ot:units "" ;
   =      otee:Octanol-water_partition_coefficient_Kow .

ot:Feature
   a      owl:Class .

ot:NumericFeature
   a      owl:Class ;
   rdfs:subClassOf ot:Feature .

   a      ot:Algorithm .
```
As an illustration of an ot:Feature, imported from a file and not calculated, Example 10 shows a feature, representing EINECS number, imported from the ECHA preregistration list (denoted by the link to OpenTox ontology entry http://www.opentox.org/api/1.1#EC).


<rdf:RDF
  xmlns:ot="http://www.opentox.org/api/1.1#"
  xmlns:bx="http://purl.org/net/nknouf/ns/bibtex#"
  xmlns:otee="http://www.opentox.org/echaEndpoints.owl#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns="http://apps.ideaconsult.net:8080/ambit2/"
  xmlns:rdf="http://www.w3.org/1999/02/rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:ota="http://www.opentox.org/algorithms.owl#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xml:base="http://apps.ideaconsult.net:8080/ambit2/">
  <owl:Class rdf:about="http://www.opentox.org/api/1.1#Feature"/>
  <owl:ObjectProperty rdf:about="http://www.opentox.org/api/1.1#hasSource"/>
  <owl:DatatypeProperty rdf:about="http://www.opentox.org/api/1.1#units"/>
  <ot:Feature rdf:about="feature/3">
    <ot:hasSource>ECHA</ot:hasSource>
    <owl:sameAs rdf:resource="http://www.opentox.org/api/1.1#EINECS"/>
    <ot:units></ot:units>
  </ot:Feature>
</rdf:RDF>
And finally, all features, available at a feature resource can be listed in either of RDF formats, or just as a list of URIs, via text/uri-list mime type (Example 11).


http://apps.ideaconsult.net:8080/ambit2/feature/1
http://apps.ideaconsult.net:8080/ambit2/feature/2
http://apps.ideaconsult.net:8080/ambit2/feature/3
http://apps.ideaconsult.net:8080/ambit2/feature/4
http://apps.ideaconsult.net:8080/ambit2/feature/5
http://apps.ideaconsult.net:8080/ambit2/feature/6
http://apps.ideaconsult.net:8080/ambit2/feature/7
http://apps.ideaconsult.net:8080/ambit2/feature/8
http://apps.ideaconsult.net:8080/ambit2/feature/10

TODO : write example (annotation)

Summary
OpenTox Feature resource provides a way to uniquely identify properties and identifiers, assigned to a compound, via feature URIs. These URIs are dereferencable (the content behind them can be inspected, either by web browser or by HTTP client tools). There are means to assign different levels of meaning, by linking to entries to ontologies (e.g. algorithms or toxicological endpoints), as well as linking to the algorithms, which can be used to generated property values. The same approach can be used to denote assays, provided that the assay is defined by an ontology.

Documentation
Further documentation available at http://opentox.org/dev/apis/api-1.1/Feature

Dataset
Provides access to chemical compounds and their features (e.g. structural, physical-chemical, biological, toxicological properties)

Representation
The concept of a Dataset of chemical compounds is central to OpenTox web services functionality. Algorithm services accept dataset URI in order to build a model, or to generate descriptor values. Model services accept dataset URI in order to apply a model and obtain predictions. Predictions are again returned as dataset URI, which could be subsequently retrieved. Search results (similarity or substructure) are also available as datasets.
The OpenTox Dataset can be thought as a file of chemical compounds, along with their properties, which however, doesn’t have a filename, but unique web address and can be read and written remotely. The dataset representation in RDF format is defined in OpenTox ontology as ot:Dataset class, and can be briefly summarized as follows:

- The dataset consists of data entries (or data rows);
- Each row is associated with exactly one chemical compound, identified by its URL and available via OpenTox Compound service API;
- One and the same compound URL can be associated with multiple dataset rows;
- Every column is associated with a Feature URL, representation should be available via OpenTox Feature API (described above). A feature is identified by its URL and has name and source, along with other properties. Any OpenTox Dataset, Algorithm or Model can serve as feature source. If the source is an algorithm or model, this allows to exactly identify how the values in the column were generated, and run the same calculations, for new chemical compounds.

This simplified view is illustrated by Table 2.

Table 2: Simplified representation of OpenTox Dataset

<table>
<thead>
<tr>
<th>URL</th>
<th>Solvent</th>
<th>compound</th>
<th>Feature</th>
<th>Property</th>
<th>Source</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://apps.ideaconsult.net:8080/ambit2/conformer/413/409421">http://apps.ideaconsult.net:8080/ambit2/conformer/413/409421</a></td>
<td>60-11-7</td>
<td>CN(C1=CC=CC(C(C=C1)N=N/C2=CC=CC=C2)C</td>
<td>3</td>
<td>3.31</td>
<td>225.3</td>
<td>YES</td>
</tr>
<tr>
<td><a href="http://apps.ideaconsult.net:8080/ambit2/conformer/44497/409422">http://apps.ideaconsult.net:8080/ambit2/conformer/44497/409422</a></td>
<td>28322-02-3</td>
<td>4-AAAF; 4-acetamidofluorene</td>
<td>1</td>
<td>NP</td>
<td>223.28</td>
<td>YES</td>
</tr>
<tr>
<td><a href="http://apps.ideaconsult.net:8080/ambit2/conformer/602/409424">http://apps.ideaconsult.net:8080/ambit2/conformer/602/409424</a></td>
<td>67-66-3</td>
<td>Formyl trichloride; methane trichloride</td>
<td>3</td>
<td>262</td>
<td>119.38</td>
<td>YES</td>
</tr>
</tbody>
</table>

In practice, the RDF representation looks slightly more complex, because each cell in the table is represented by a separate instance of FeatureValue class, which links to the ot:Feature (column header) and holds the value itself.
Besides RDF, one can retrieve various information about the dataset by using text/uri-list mime type and following templates:

<table>
<thead>
<tr>
<th>Description</th>
<th>URI Template</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retrieve entire dataset content. If uri-list, retrieve only compound URIs</td>
<td><a href="http://host:port/dataset/%5Bid">http://host:port/dataset/[id</a>]</td>
</tr>
<tr>
<td>Retrieve representation of features (columns) of the dataset</td>
<td><a href="http://host:port/dataset/%5Bid%5D/feature">http://host:port/dataset/[id]/feature</a></td>
</tr>
<tr>
<td>Retrieves dataset metadata (name, etc.)</td>
<td><a href="http://host:port/dataset/%5Bid%5D/metadata">http://host:port/dataset/[id]/metadata</a></td>
</tr>
</tbody>
</table>

Example 12. Retrieve dataset metadata in RDF/XML

```
<rdf:RDF
 xmlns:ac="http://apps.ideaconsult.net:8080/ambit2/compound/
 xmlns:ot="http://www.opentox.org/api/1.1#"
 xmlns:bx="http://purl.org/net/nknouf/ns/bibtex#"
 xmlns:otee="http://www.opentox.org/echaEndpoints.owl#"
 xmlns:dc="http://purl.org/dc-elements/1.1/
 xmlns="http://apps.ideaconsult.net:8080/ambit2/
 xmlns:am="http://apps.ideaconsult.net:8080/ambit2/model/
 xmlns:rdf="http://www.w3.org/1999/02/rdf-syntax-ns#"
 xmlns:ad="http://apps.ideaconsult.net:8080/ambit2/dataset/
 xmlns:ag="http://apps.ideaconsult.net:8080/ambit2/algorithm/
 xmlns:owl="http://www.w3.org/2002/07/owl#"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
 xml:base="http://apps.ideaconsult.net:8080/ambit2/">
 <owl:Class rdf:about="http://www.opentox.org/api/1.1#Dataset"/>
 <owl:Class rdf:about="http://purl.org/net/nknouf/ns/bibtex#Entry"/>
 <ot:Dataset rdf:about="dataset/9">
 <dc:source>ISSCAN_v3a_1153_19Sept08.1222179139.sdf</dc:source>
 <dc:publisher>nina</dc:publisher>
 <rdfs:seeAlso>
 <bx:Entry rdf:about="reference/20117">
 <rdfs:seeAlso>http://www.epa.gov/NCCT/dsstoxx/sdf_isscan_external.html</rdfs:seeAlso>
 <dc:title>ISSCAN_v3a_1153_19Sept08.1222179139.sdf</dc:title>
 </bx:Entry>
 </rdfs:seeAlso>
 <dc:title>ISSCAN: Istituto Superiore di Sanita, CHEMICAL CARCINOGENS: STRUCTURES AND EXPERIMENTAL DATA</dc:title>
 </ot:Dataset>
</rdf:RDF>
```

Example 13. Retrieve list of features URI, used in this dataset

```
```
Or just the compound URIs (restricted to first 3) via text/uri-list mime type:

Example 14. Dataset representation in N3 format

```
```

 RDF representation

OpenTox Dataset encapsulates compounds and their property (feature) values. The RDF triples naturally allow to model binary relationships via Subject-Predicate-Object construct (e.g. molecular_weight has_value 200 ). In order to model higher order relationships (e.g. CompoundX hasProperty molecular_weight with value 200 and more complex statements), two more classes have been introduced in OpenTox resource ontology - namely **ot:FeatureValue** and **ot:DataEntry**.

**ot:FeatureValue** class encapsulates the relationship **Feature** - **hasValue** - **Value**. This is formally defined via object property **ot:feature** (links to the **ot:Feature** class), and data property **ot:value** (holds the value itself). This class can be interpreted as a cell in a table, where each cell contains not only the value, but a reference to the column header as well. This results in a flexible representation, not limited to tabular values.

**ot:DataEntry** class encapsulates the relationship **Compound** - **has values for** - **specific Features**. This can be thought as a row in a table, where the object property **ot:compound** specifies the **ot:Compound** resource, where **ot:values** object property specifies all the cells (**ot:FeatureValue** instances) , available in the data entry.
A dataset consists of multiple `ot:DataEntry`-ies. These classes can be represented as anonymous classes in RDF notations, or have unique URIs.

![Protégé screenshot](image)

Figure 4 `ot:Dataset` representation, Protégé screenshot.
As an illustration, the content of the dataset http://apps.ideaconsult.net:8080/ambit2/dataset/9 can be retrieved in RDF/XML and N3:


By default, querying http://port//dataset/{id} will retrieve the entire dataset content, with all compounds and features used. To restrict these, the query parameters compound_uris[] and/or feature_uris[] can be used:


To minimize the output, Example 15 and Example 16 use the following URL to retrieve first 2 rows and two features, instead of the entire dataset:
Example 15. Dataset representation in RDF/XML format

```
<rdf:RDF
    xmlns:ac="http://apps.ideaconsult.net:8080/ambit2/compound/
    xmlns:ot="http://www.opentox.org/api/1.1#"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns="http://apps.ideaconsult.net:8080/ambit2/
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:ad="http://apps.ideaconsult.net:8080/dataset/
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:ota="http://www.opentox.org/algorithmTypes.owl#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xml:base="http://apps.ideaconsult.net:8080/ambit2/"/>
<owl:Class rdf:about="http://www.opentox.org/api/1.1#Dataset"/>
<owl:Class rdf:about="http://www.opentox.org/api/1.1#Compound"/>
<owl:Class rdf:about="http://www.opentox.org/api/1.1#Feature"/>
<owl:Class rdf:about="http://www.opentox.org/api/1.1#FeatureValue"/>
<owl:Class rdf:about="http://www.opentox.org/api/1.1#NumericFeature"/>
<owl:Class rdf:about="http://www.opentox.org/api/1.1#NominalFeature"/>
<owl:Class rdf:about="http://www.opentox.org/api/1.1#DataEntry"/>
<owl:ObjectProperty rdf:about="http://www.opentox.org/api/1.1#acceptValue"/>
<owl:ObjectProperty rdf:about="http://www.opentox.org/api/1.1#dataEntry"/>
<owl:ObjectProperty rdf:about="http://www.opentox.org/api/1.1#hasSource"/>
<owl:ObjectProperty rdf:about="http://www.opentox.org/api/1.1#values"/>
<owl:ObjectProperty rdf:about="http://www.opentox.org/api/1.1#compound"/>
<owl:ObjectProperty rdf:about="http://www.opentox.org/api/1.1#feature"/>
<owl:DatatypeProperty rdf:about="http://www.opentox.org/api/1.1#units"/>
<owl:DatatypeProperty rdf:about="http://www.opentox.org/api/1.1#value"/>
<owl:AnnotationProperty rdf:about="http://purl.org/dc/elements/1.1/creator"/>
<ot:Dataset rdf:about="dataset/9">
  <ot:DataEntry>
    <ot:DataEntry>
      <ot:values>
        <ot:FeatureValue>
          <ot:value rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
            NP
          </ot:value>
        </ot:FeatureValue>
        <ot:NumericFeature rdf:about="feature/21576">
          <ot:hasSource>ISSCAN_v3a_1153_19Sept08.1222179139.sdf</ot:hasSource>
        </ot:NumericFeature>
      </ot:values>
    </ot:DataEntry>
  </ot:DataEntry>
</ot:Dataset>
```
Example 16. Dataset representation in N3 format

```n3
@prefix ot: <http://www.opentox.org/api/1.1#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix : <http://apps.ideaconsult.net:8080/ambit2/> .
@prefix ota: <http://www.opentox.org/algorithmTypes.owl#> .
@prefix otee: <http://www.opentox.org/echaEndpoints.owl#> .
@prefix ar: <http://apps.ideaconsult.net:8080/ambit2/reference/> .
@prefix bx: <http://purl.org/net/nknouf/ns/bibtex#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix am: <http://apps.ideaconsult.net:8080/ambit2/model/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix ac: <http://apps.ideaconsult.net:8080/ambit2/compound/> .
@prefix ad: <http://apps.ideaconsult.net:8080/ambit2/dataset/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ag: <http://apps.ideaconsult.net:8080/ambit2/algorithm/> .
@prefix af: <http://apps.ideaconsult.net:8080/ambit2/feature/> .

af:21576
    a        ot:Feature , ot:NumericFeature ;
    dc:creator "http://www.epa.gov/NCCT/dsstox/sdf_isscan_external.html" ;
    dc:title "TD50_Rat" ;
    ot:hasSource "ISSCAN_v3a_1153_19Sept08.1222179139.sdf" ;
    ot:units "" ;
    =       otee:Carcinogenicity .

ot:Dataset
    a        owl:Class .

ot:acceptValue
    a        owl:ObjectProperty .

ot:FeatureValue
    a        owl:Class .

ot:NominalFeature
    a        owl:Class ;
    rdfs:subClassOf ot:Feature .

<http://apps.ideaconsult.net:8080/ambit2/compound/413/conformer/409421>
    a        ot:Compound .

dc:type
    a        owl:AnnotationProperty .

ot:hasSource
    a        owl:ObjectProperty .
```
ot:NumericFeature
  a owl:Class;
  rdfs:subClassOf ot:Feature.

ot:compound
  a owl:ObjectProperty.

dc:title
  a owl:AnnotationProperty.

dc:description
  a owl:AnnotationProperty.

ot:Compound
  a owl:Class.

ot:dataEntry
  a owl:ObjectProperty.

<http://apps.ideaconsult.net:8080/ambit2/compound/44497/conformer/409422>
  a ot:Compound.

ot:Feature
  a owl:Class.

ot:value
  a owl:DatatypeProperty.

ot:DataEntry
  a owl:Class.

ot:units
  a owl:DatatypeProperty.

ad:9 a ot:Dataset;
  ot:dataEntry
    [ a ot:DataEntry;
      ot:compound <http://apps.ideaconsult.net:8080/ambit2/compound/413/conformer/409421>;
      ot:values
        [ a ot:FeatureValue;
          ot:feature af:21576;
          ot:value "3.309999942779541"^^xsd:double
          ];
        ot:values
        [ a ot:FeatureValue;
          ot:feature af:21573;
          ot:value "3.0"^^xsd:double
          ]
    ];
  ot:dataEntry
    [ a ot:DataEntry;
      ot:values
The dataset services optionally support formats other than RDF, for example text/csv (Comma delimited), text/x-arff (Weka ARFF), application/pdf, chemical/x-mdl-sdfile, other Chemical MIME formats. This allows retrieving the same data in convenient format, but the URL links to compound and feature resources are being lost. Thus, the

Example 17. Retrieve Dataset representation in N3 format

```
```

Modify (write and update)

POST

TODO Code examples

Documentation

http://opentox.org/dev/apis/api-1.1/Dataset

Summary
Algorithm

Provides access to OpenTox algorithms. There are several algorithm services, developed by different OpenTox partners. List of algorithms can be retrieved by HTTP GET operation at http://host:port/algorithm (copy URLs from Example 18, Example 19, Example 20 into your browser, or use cURL command).

Example 18. Retrieve list of all algorithm URIs at IDEA algorithm service

```
http://apps.ideaconsult.net:8080/ambit2/algorithm/SimpleKMeans
http://apps.ideaconsult.net:8080/ambit2/algorithm/LR
http://apps.ideaconsult.net:8080/ambit2/algorithm/pka
http://apps.ideaconsult.net:8080/ambit2/algorithm/toxtreecramer
http://apps.ideaconsult.net:8080/ambit2/algorithm/toxtreecramer2
http://apps.ideaconsult.net:8080/ambit2/algorithm/toxtreeeverhaar
http://apps.ideaconsult.net:8080/ambit2/algorithm/toxtreeeye
http://apps.ideaconsult.net:8080/ambit2/algorithm/toxtreeskinirritation
http://apps.ideaconsult.net:8080/ambit2/algorithm/toxtreemic
http://apps.ideaconsult.net:8080/ambit2/algorithm/toxtreeskinsens
http://apps.ideaconsult.net:8080/ambit2/algorithm/toxtreemichaelacceptors
http://apps.ideaconsult.net:8080/ambit2/algorithm/toxtreecarc
http://apps.ideaconsult.net:8080/ambit2/algorithm/toxtreebiodeg
http://apps.ideaconsult.net:8080/ambit2/algorithm/toxtreekroes
```
<table>
<thead>
<tr>
<th>Algorithm URI</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://apps.ideaconsult.net:8080/ambit2/algorithm/org.openscience.cdk.qsar.descriptors.molecular.HBondAcceptCountDescriptor">http://apps.ideaconsult.net:8080/ambit2/algorithm/org.openscience.cdk.qsar.descriptors.molecular.HBondAcceptCountDescriptor</a></td>
<td>Molecular HBondAcceptCountDescriptor</td>
</tr>
<tr>
<td><a href="http://apps.ideaconsult.net:8080/ambit2/algorithm/structurequality">http://apps.ideaconsult.net:8080/ambit2/algorithm/structurequality</a></td>
<td>StructureQualityDescriptor</td>
</tr>
<tr>
<td><a href="http://apps.ideaconsult.net:8080/ambit2/algorithm/struckeys">http://apps.ideaconsult.net:8080/ambit2/algorithm/struckeys</a></td>
<td>StructureKeysDescriptor</td>
</tr>
<tr>
<td><a href="http://apps.ideaconsult.net:8080/ambit2/algorithm/smartsprop">http://apps.ideaconsult.net:8080/ambit2/algorithm/smartsprop</a></td>
<td>SMARTSPropDescriptor</td>
</tr>
<tr>
<td><a href="http://apps.ideaconsult.net:8080/ambit2/algorithm/superservice">http://apps.ideaconsult.net:8080/ambit2/algorithm/superservice</a></td>
<td>SuperserviceDescriptor</td>
</tr>
<tr>
<td><a href="http://apps.ideaconsult.net:8080/ambit2/algorithm/mockup">http://apps.ideaconsult.net:8080/ambit2/algorithm/mockup</a></td>
<td>MockupDescriptor</td>
</tr>
</tbody>
</table>

**Example 19. Retrieve list of all algorithm URIs at TUM algorithm service**

```
Example 19. Retrieve list of all algorithm URIs at TUM algorithm service
```

```bash
curl -H "Accept:text/uri-list" http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm
```

```bash
http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/kNNclassification
http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/kNNregression
http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/PLSregression
http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/M5P
http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/GaussP
http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/FTM
http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/gSpan
http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/CDKPhysChem
http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/JOELIB2
http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/InfoGainAttributeEval
```

**Example 20. Retrieve list of all algorithm URIs at NTUA algorithm service**
http://opentox.ntua.gr:3000/algorithm/filter
http://opentox.ntua.gr:3000/algorithm/svm
http://opentox.ntua.gr:3000/algorithm/leverages
http://opentox.ntua.gr:3000/algorithm/mlr

Representation
As it can be noticed, there are multiple kinds of algorithms, some are descriptor calculation algorithms, and others provide access to different machine learning procedures or data preprocessing. The representation of algorithms is again defined by OpentoX ontology, where all algorithms are subclass of ot:Algorithm (Figure 6.).

![Figure 6 ot:Algorithm representation, Protégé screenshot.](image)

Algorithm name is defined by dc:title (Dublin Core namespace).

Parameters, supported by the algorithm are specified via object property ot:parameters and should be of class ot:Parameter (as defined in opentox.owl). These entries serve as a information what parameters are required in order to run the algorithm, the values itself should be provided by the client when initiating the calculations via POST.
Algorithm types are distinguished by means of Algorithm types ontology.

![Algorithm types ontology, Protégé screenshot.](image)

Algorithm types ontology is available at [http://opentox.org/data/documents/development/RDF%20files/AlgorithmTypes](http://opentox.org/data/documents/development/RDF%20files/AlgorithmTypes) and provides a hierarchical classification of algorithm types. Algorithm type in RDF representation is set by direct subclassing (rdf:type) of a class from the algorithm types ontology (ota: [http://www.opentox.org/algorithms.owl](http://www.opentox.org/algorithms.owl)), e.g. `<myalgorithm> rdf:type ota:Classification, Example 21.`

**Example 21.** Representation of a clustering algorithm. Note multiple inheritance from classes from Algorithm types ontology [http://www.opentox.org/algorithms.owl](http://www.opentox.org/algorithms.owl).

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:ot="http://www.opentox.org/api/1.1#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
```
Example 22. Representation of a classification algorithm.

```bash
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:ot="http://www.opentox.org/api/1.1#"
 xmlns:bx="http://purl.org/net/nknouf/ns/bibtex#"
 xmlns:owl="http://www.w3.org/2002/07/owl#"
 xmlns:othee="http://www.opentox.org/echaEndpoints.owl#"
 xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
 <owl:Class rdf:about="http://www.opentox.org/api/1.1#Algorithm"/>
 <owl:Class rdf:about="http://www.opentox.org/algorithmTypes.owl#Classification"/>
 <owl:Class rdf:about="http://www.opentox.org/algorithmTypes.owl#Supervised"/>
 <owl:Class rdf:about="http://www.opentox.org/algorithmTypes.owl#EagerLearning"/>
```
Example 23. Representation of a PLS regression algorithm.


<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:ot="http://www.opentox.org/api/1.1#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:bo="http://www.blueobelisk.org/ontologies/chemoinformatics-algorithms/#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:ota="http://www.opentox.org/algorithms.owl#">
  <owl:Class rdf:about="http://www.opentox.org/api/1.1#Algorithm"/>
  <owl:Class rdf:about="http://www.opentox.org/api/1.1#Parameter"/>
  <ot:Algorithm rdf:about="http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/PLSregression">
    <dc:identifier rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"/>
    <http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/PLSregression rdf:identifier/>
    <dc:title rdf:datatype="http://www.w3.org/2001/XMLSchema#string"/>
  </ot:Algorithm>
  <ot:parameters>
    <Parameter>
      <dc:title rdf:datatype="http://www.w3.org/2001/XMLSchema#string"/>
      <dc:description rdf:datatype="http://www.w3.org/2001/XMLSchema#string"/>
    </Parameter>
  </ot:parameters>
</rdf:RDF>
<ot:parameters>
  <ot:Parameter>
    <dc:description rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Whether to update the class attribute with the predicted value.</dc:description>
    <ot:paramScope rdf:datatype="http://www.w3.org/2001/XMLSchema#string">optional</ot:paramScope>
    <ot:paramValue rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">false</ot:paramValue>
  </ot:Parameter>
</ot:parameters>
<ot:parameters>
  <ot:Parameter>
    <dc:description rdf:datatype="http://www.w3.org/2001/XMLSchema#string">URI to the dataset service to be used</dc:description>
    <ot:paramScope rdf:datatype="http://www.w3.org/2001/XMLSchema#string">optional</ot:paramScope>
  </ot:Parameter>
</ot:parameters>
<ot:parameters>
  <ot:Parameter>
    <dc:description rdf:datatype="http://www.w3.org/2001/XMLSchema#string">URI to the dataset to be used</dc:description>
    <ot:paramScope rdf:datatype="http://www.w3.org/2001/XMLSchema#string">mandatory</ot:paramScope>
    <ot:paramValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"></ot:paramValue>
  </ot:Parameter>
</ot:parameters>
<dc:description rdf:datatype="http://www.w3.org/2001/XMLSchema#string">OpenTox REST interface to the WEKA PLS regression algorithm</dc:description>
<ot:parameters>
  <ot:Parameter>
    <dc:title rdf:datatype="http://www.w3.org/2001/XMLSchema#string">numComponents</dc:title>
  </ot:Parameter>
</ot:parameters>
An algorithm is applied by submitting HTTP POST to the algorithm URI and providing required parameters. A common required parameter is dataset_uri=http://host:port/dataset/{datasetid} ,
which specifies the data set to be operated on. Supervised learning algorithm require the the URI of the
dependent variable to be specified via parameter `prediction_feature`. The `dataset_service`
parameter is optional and used to specify where the final result will be stored. Recall HTTP POST in REST
style services is recommended to return URI of the result, and not the content of the result. Therefore,
the algorithm services are designed in a way to store the results into a dataset service and return the
URL of the resulted dataset. If the calculations are assumed to be time consuming, the algorithm service
might return OpenTox Task URI, instead of the dataset URI (Example 24)

Example 24. Building J48 decision tree classification model

```bash
* About to connect() to opentox.informatik.tu-muenchen.de port 8080 (#0)
* Trying 131.159.28.16... connected
* Connected to opentox.informatik.tu-muenchen.de (131.159.28.16) port 8080 (#0)
> POST /OpenTox-dev/algorithm/J48 HTTP/1.1
> User-Agent: curl/7.18.2 (x86_64-pc-linux-gnu) libcurl/7.18.2 OpenSSL/0.9.8g zlib/1.2.3.3 libidn/1.8 libssh2/0.18
> Host: opentox.informatik.tu-muenchen.de:8080
> Accept: */*
> Content-Length: 201
> Content-Type: application/x-www-form-urlencoded
> HTTP/1.1 202 Accepted
HTTP/1.1 202 Accepted
< Date: Sat, 31 Jul 2010 14:46:38 GMT
Date: Sat, 31 Jul 2010 14:46:38 GMT
< Location: http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/task/acdf6eac-d5a2-402c-a4e2-06cd7e3ca1b5
Location: http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/task/acdf6eac-d5a2-402c-a4e2-06cd7e3ca1b5
< Accept-Ranges: bytes
Accept-Ranges: bytes
< Server: Noelios-Restlet-Engine/1.1.snapshot
Server: Noelios-Restlet-Engine/1.1.snapshot
< Content-Type: text/uri-list;charset=ISO-8859-1
Content-Type: text/uri-list;charset=ISO-8859-1
< Content-Length: 99
Content-Length: 99
<
* Connection #0 to host opentox.informatik.tu-muenchen.de left intact
* Closing connection #0
http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/task/acdf6eac-d5a2-402c-a4e2-06cd7e3ca1b5
```

When task URI is returned, the returned status code is HTTP 202 Accepted, instead of HTTP 200 OK.
This tells the client the processing is not completed and the client need to poll the task URI until OK code
is returned (Example 25)
Example 25. Building J48 decision tree classification model

```
$ curl -iv -H "Accept:text/uri-list" http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/task/acdf6eac-d5a2-402c
  * About to connect() to opentox.informatik.tu-muenchen.de port 8080 (#0)
  * Trying 131.159.28.16... connected
  * Connected to opentox.informatik.tu-muenchen.de (131.159.28.16) port 8080 (#0)
> GET /OpenTox-dev/task/acdf6eac-d5a2-402c-a4e2-06cd7e3ca1b5 HTTP/1.1
> User-Agent: curl/7.18.2 (x86_64-pc-linux-gnu) libcurl/7.18.2 OpenSSL/0.9.8g zlib/1.2.3.3 libidn/1.8 libssh2/0.18
> Host: opentox.informatik.tu-muenchen.de:8080
> Accept:text/uri-list
>
< HTTP/1.1 200 OK
HTTP/1.1 200 OK
< Date: Sat, 31 Jul 2010 14:47:22 GMT
Date: Sat, 31 Jul 2010 14:47:22 GMT
< Location: http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/model/TUMOpenToxModel_j48_48
Location: http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/model/TUMOpenToxModel_j48_48
< Vary: Accept-Charset, Accept-Encoding, Accept-Language, Accept
Vary: Accept-Charset, Accept-Encoding, Accept-Language, Accept
< Accept-Ranges: bytes
Accept-Ranges: bytes
< Server: Noelios-Restlet-Engine/1.1.snapshot
Server: Noelios-Restlet-Engine/1.1.snapshot
< Content-Type: text/uri-list;charset=UTF-8
Content-Type: text/uri-list;charset=UTF-8
< Content-Length: 86
Content-Length: 86
<

* Connection #0 to host opentox.informatik.tu-muenchen.de left intact
* Closing connection #0
```

The final result, returned by Example 25 is the URI of the new model [http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/model/TUMOpenToxModel_j48_48](http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/model/TUMOpenToxModel_j48_48). This URI could be then used to obtain prediction results, as will be demonstrated in the Model section of this document.

Example 26. Calculating XLogP descriptors

```
$ curl -X POST
- d "dataset_uri=http://apps.ideaconsult.net:8080/ambit2/dataset/54" -iv
  * About to connect() to apps.ideaconsult.net port 8080 (#0)
  * Trying 93.123.36.100... connected
  * Connected to apps.ideaconsult.net (93.123.36.100) port 8080 (#0)
> POST /ambit2/algorithm/org.openscience.cdk.qsar.descriptors.molecular.XLogPDescriptor HTTP/1.1
> User-Agent: curl/7.18.2 (x86_64-pc-linux-gnu) libcurl/7.18.2 OpenSSL/0.9.8g zlib/1.2.3.3 libidn/1.8 libssh2/0.18
> Host: apps.ideaconsult.net:8080
> Accept: */*

> Content-Length: 62
```
A Task URI is returned http://apps.ideaconsult.net:8080/ambit2/task/dcbd4145-b03f-45a8-9183-ef4759ea06e3, we'll query the task again to find out if it is completed:
$ curl -v http://apps.ideaconsult.net:8080/ambit2/task/dcbd4145-b03f-45a8-9183-ef4759ea06e3
  * About to connect() to apps.ideaconsult.net port 8080 (#0)
  * Trying 93.123.36.100... connected
  * Connected to apps.ideaconsult.net (93.123.36.100) port 8080 (#0)
  > GET /ambit2/task/dcbd4145-b03f-45a8-9183-ef4759ea06e3 HTTP/1.1
  > User-Agent: curl/7.18.2 (x86_64-pc-linux-gnu) libcurl/7.18.2 OpenSSL/0.9.8g zlib/1.2.3.3 libidn/1.8 libssh2/0.18
  > Host: apps.ideaconsult.net:8080
  > Accept: */*
  >
  < HTTP/1.1 200 OK
  < Content-Language: en
  < Date: Sat, 31 Jul 2010 16:56:13 GMT
  < Vary: Accept-Charset, Accept-Encoding, Accept-Language, Accept
  < Accept-Ranges: bytes
  < Server: Restlet-Framework/2.0m6
  < Content-Type: application/rdf+xml;charset=UTF-8
  < Content-Length: 1785
  <
  <rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:ot="http://www.opentox.org/api/1.1#"
    xmlns:bx="http://purl.org/net/nknouf/ns/bibtex#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:otee="http://www.opentox.org/echaEndpoints.owl#"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:ota="http://www.opentox.org/algorithmtypes.owl#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
    <owl:Class rdf:about="http://www.opentox.org/api/1.1#Task"/>
    <owl:DatatypeProperty rdf:about="http://www.opentox.org/api/1.1#resultURI" />
    <owl:DatatypeProperty rdf:about="http://www.opentox.org/api/1.1#percentageCompleted" />
    <owl:DatatypeProperty rdf:about="http://www.opentox.org/api/1.1#hasStatus" />
    <ot:Task rdf:about="http://apps.ideaconsult.net:8080/ambit2/task/dcbd4145-b03f-45a8-9183-ef4759ea06e3">  
      <ot:resultURI rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI" />
      <ot:percentageCompleted rdf:datatype="http://www.w3.org/2001/XMLSchema#float"/>
      >0.0</ot:percentageCompleted>
      <ot:hasStatus rdf:datatype="http://www.w3.org/2001/XMLSchema#string" />
      <Completed rdf:about="http://www.w3.org/2001/XMLSchema#dateTime">
        >1280595219987</Completed>
      <dc:date rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">
        >1280595219987</dc:date>
      <dc:title rdf:datatype="http://www.w3.org/2001/XMLSchema#string" />
      <ApplyXLogP to http://apps.ideaconsult.net:8080/ambit2/dataset/54</dc:title>
    </ot:Task>
  </rdf:RDF>
  * Connection #0 to host apps.ideaconsult.net left intact
  * Closing connection #0
Upon completion, the task returns the URI of the result, a dataset with XLogP values


Documentation
Further documentation and examples are available at http://opentox.org/dev/apis/api-1.1/Algorithm.

Task
Provides different representations for chemical compounds with a unique and defined chemical structure.

Documentation

http://opentox.org/dev/apis/api-1.1/AsyncTask

Model
Provides different representations for QSAR/toxicology models. Models are the output/result of learning algorithms. To make use of a model for prediction, it is necessary to have a dataset with compatible descriptors/features. If the dataset_service parameter is stated, a new dataset will be created if on the other hand the result_dataset parameter is stated, the stated dataset will be updated with the predicted feature values. In other words, a new "column" for the predictions is added to the input dataset. If none of the two parameters is given, a default dataset service is used and a new dataset is created.

A Model is created by applying an ot:Algorithm with specific parameters to a training ot:Dataset.

Representation

- Model URI is defined by its URI
- Model Name is defined by dc:title property
- Model creator might be defined by dc:creator property
- The date of Model creation is defined by dc:date property
- The Algorithm defined by ot:algorithm object property
- The independent variables are instances of ot:Feature defined by ot:independentVariables property (can be multiple)
- The dependent variables are are instances of ot:Feature and are defined by ot:dependentVariables property (can be multiple)
- The variables, where prediction results will be stored, are are instances of ot:Feature and are defined by ot:predictedVariables (can be multiple)
- Parameters are defined by ot:parameters
- The training Dataset is an instance of ot:Dataset and defined by ot:trainingDataset
Figure 8: Model representation, Protégé screenshot.

An example representation of the model, built in the Algorithm section is presented at Example 27. The output is huge, since the model use more than 200 independent variables.

Example 27. Retrieve representation of model http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/model/TUMOpenToxModel_j48_48 in RDF/XML format

```
curl -H "Accept:application/rdf+xml" http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/model/TUMOpenToxModel_j48_48
```
<ot:independentVariables>
</ot:independentVariables>

<ot:independentVariables>
</ot:independentVariables>

<ot:independentVariables>
</ot:independentVariables>

<ot:independentVariables>
</ot:independentVariables>

<ot:independentVariables>
</ot:independentVariables>

<ot:independentVariables>
</ot:independentVariables>

<ot:independentVariables>
</ot:independentVariables>

<ot:independentVariables>
</ot:independentVariables>

<ot:independentVariables>
</ot:independentVariables>

<ot:independentVariables>
</ot:independentVariables>

<ot:independentVariables>
</ot:independentVariables>

<ot:independentVariables>
</ot:independentVariables>

<ot:independentVariables>
</ot:independentVariables>

<ot:independentVariables>
</ot:independentVariables>

<ot:independentVariables>
</ot:independentVariables>

<ot:independentVariables>
</ot:independentVariables>

<ot:independentVariables>
</ot:independentVariables>

<ot:independentVariables>
</ot:independentVariables>

<ot:independentVariables>
</ot:independentVariables>

<ot:independentVariables>
</ot:independentVariables>
<ot:independentVariables>  
</ot:independentVariables>

<!-- Snippet of repeated independent variables and parameters... -->

<ot:algorithm>
  <ot:parameters>
    <ot:Parameter>
      <dc:title rdf:datatype="http://www.w3.org/2001/XMLSchema#string">minNumObj</dc:title>
      <ot:paramValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int">2</ot:paramValue>
    </ot:Parameter>
    <ot:Parameter>
      <dc:title rdf:datatype="http://www.w3.org/2001/XMLSchema#string">seed</dc:title>
      <ot:paramValue rdf:datatype="http://www.w3.org/2001/XMLSchema#double">1</ot:paramValue>
    </ot:Parameter>
    <ot:Parameter>
      <ot:paramValue rdf:datatype="http://www.w3.org/2001/XMLSchema#double">0.25</ot:paramValue>
    </ot:Parameter>
    <ot:Parameter>
      <ot:paramValue rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">false</ot:paramValue>
    </ot:Parameter>
  </ot:parameters>
</ot:algorithm>
Example 28. Retrieve representation of model http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/model/TUMOpenToxModel_j48_48 in text format

Example 7 displays an OpenTox feature with title “XLogP” and identified by the URI
**Documentation**

http://opentox.org/dev/apis/api-1.1/structure

**Example:** Retrieve compound as SMILES

```
$ curl -H "Accept:chemical/x-daylight-smiles" http://apps.ideaconsult.net:8080/ambit2/compound/1
```

**Validation**

Provides different representations for chemical compounds with a unique and defined chemical structure.

**Documentation**

http://opentox.org/dev/apis/api-1.1/Validation

**Example:**

http://opentox.informatik.uni-freiburg.de/validation/examples