Deliverable D5.3

Validation against confidential data

Grant Agreement	Health-F5-2008-200787
Acronym	OpenTox
Name	An Open Source Predictive Toxicology Framework
Coordinator	Douglas Connect

Contract No.	Health-F5-2008-200787	
Document Type:	Deliverable Report	
WP/Task:	WP5	
Name	Validation against confidential data	
Document ID:	OpenTox Deliverable Report WP5	
Date:	22/03/11	
Status:	Version 2.2	
Organisation:		
Contributors	Pantelis Sopasakis (PS)	NTUA
	Haralambos Sarimveis (HS)	NTUA
	Martin Gütlein (MG)	ALU-FR
	Tobias Girschick (TG)	TUM
	Fabian Buchwald (FB)	TUM
	Andreas Maunz (AM)	ALU-FR
	Barry Hardy (BH)	DC
	Andreas Karwath (AK)	ALU-FR
	Vedrin Jeliazkov (VJ)	IDEA

Distribution:	Public

Purpose of Document:	To disseminate results on procedures for the validation of confidential toxicity data within the OpenTox Framework

Document History:	1 - Initial draft prepared on Feb 8, 2011
	2 - Added validation examples on Feb 12 by MG
	3- Added descriptions of stand-alone applications on Feb 22 by PS
	4- First revision on Feb 22, 2011 by TG, FB
	5- Modifications following the comments by Tobias & Fabian
	6- Minor adjustments and reformatting of curl boxes by MG, Feb24
	7- Additions to local installation use case
	8- Review by BH
	9- Reply to BH's comments and modifications
	10-Final Review

Table of contents

Table of figures

Summary

In a framework like OpenTox where applications may involve confidential data, it is critical to address successfully the issue of protecting sensitive information from being accessed or copied by unauthorized users. With this functionality, OpenTox meets a key requirement imposed by the REACH legislation, according to which registrants may ask for protection of the security and confidentiality of the supplied information. OpenTox applications[1] protect confidential information from data leakage through identity and group-based policy controls, whose foundations are the authorization and authentication policies that have been developed and implemented in OpenTox services.

This report on "validation against confidential data" presents the work that has been accomplished within the OpenTox project on protecting confidential data from unauthorized access with emphasis on the validation services. The use case description and its implementation, based on a rigorous authentication and authorization strategy, are described. Examples on confidential datasets are presented, which illustrate not only the efficiency of the approach but also the consistency of the validation services with open standards, the adopted Restful web service architecture and the OpenTox ontology. An alternative implementation is also presented which maximizes protection of confidential data by providing a local standalone installation of the OpenTox services, not involving any transmission of confidential data over the Internet. An extended use case is described where the end user does not have access to confidential data but wishes to validate his algorithm or model against confidential data sets.

[1] For example see ToxCreate, http://toxcreate.org

1. Introduction

OpenTox validation service development aims to create a unified framework for evaluating (Quantitative) Structure-Activity Relationship i.e., (Q)SAR models and algorithms, to standardize the comparison routines, and to provide a robust and open platform for these comparisons. An extensive number of tools have already been implemented towards the accomplishment of these objectives[2]. For the integration of these validation and reporting tools, a common collaboration framework based on the RESTful web service architecture[3] has been utilized. For the successful deployment of OpenTox applications, it must be taken into account that toxicity data are often considered highly sensitive and confidential by their owners and unauthorized access to these data may expose them to risks. Additionally, OpenTox is a framework that meets the requirements of the Registration, Evaluation, Authorisation and Restriction of Chemicals (REACH) legislation[4]. Among them, data protection is of particular importance, if confidentiality is requested by the registrants. Towards this direction, mechanisms have been developed within OpenTox to control access to confidential information, ensure the maximum level of protection and minimize risks related to confidential data. This report describes the two approaches that have already been implemented: a) the on-line approach where access control and data protection is based on rigorous and strong authorization and authentication procedures, which have been gradually embedded in all OpenTox services and b) the off-line stand-alone approach which allows the user to download and locally install OpenTox services and does not involve any transfer of data over the Internet.

2. Validation against confidential data using authentication and authorization

Authentication[5] and Authorization[6] (abbreviated as A&A) form the core of network security with Accounting being the third 'A' of the trilogy[7]. **Authentication** is the process of trusting

[2] http://www.opentox.org/dev/documentation/components/componentsvalidation/

[3] http://www.opentox.org/dev/apis/api-1.1/Validation

[4] http://ec.europa.eu/enterprise/sectors/chemicals/reach/index_en.htm

[5] http://en.wikipedia.org/wiki/Authentication

[6] http://en.wikipedia.org/wiki/Authorization

[7] http://en.wikipedia.org/wiki/AAA_protocol

a user's alleged identity by requiring certain evidence such as pairs of id and password or attested digital certificates by some trusted authority. To put it simply, authentication is about confirming that the users are those that they claim to be. **Authorization** is a process that follows authentication and determines access privileges to the system including – but not limited to – retrieval of information from databases and use of web services or other functions of the system. So authorization determines whether a particular authenticated individual has the right to perform a given action and thus frames users with certain restrictions. Finally, **Accounting** refers to the tracking of actions of a particular authenticated user, for example the access and use history of particular services and the consumption of resources such as storage and computational usage.

2.1 REACH legislation and confidential data

As REACH comes into action, tens of thousands of data sheets regarding chemical substances along with safety and exposure information are going to be registered in a central database run by the European Chemicals Agency (EChA) in Helsinki[8]. The Agency acts as the central point in the REACH system: it manages the databases necessary to operate the system, co-ordinates the in-depth evaluation of suspicious chemicals and is building up a public database in which consumers and professionals can find hazard information.

According to REACH, the industries are assumed to shoulder the burden of managing the risks of the human contact with chemical substances (in food, cosmetics, etc.) and report to the EU accordingly. EChA publishes information it holds on registered substances free of charge on the internet. However, in certain cases, information can be withheld, if the registrant submitting the information also submits a justification as to why publishing the information would be potentially harmful to the commercial interests of the registrant or any other party concerned. EChA will not publish the information concerned, if justification is accepted as valid[9]. Towards this direction, REACH-relevant software frameworks such as OpenTox should take into account these confidentiality issues. In the light of these REACH issues, a robust authentication and authorization design is rendered a requirement for the OpenTox framework.

2.2 Authentication and authorization in OpenTox

Within OpenTox, the principles of network security are materialized by means of a central access control system based on Single Sign-On (SSO). Accounting is currently delegated to

8 http://echa.europa.eu/

9 http://echa.europa.eu/doc/reachit/dsm_16_confidentiality_claims.pdf

service providers according to their processing and storage resources. It is fundamental for a distributed system like OpenTox to provide a structured and robust access control system that enables administrators and system providers to:

- Flexibly specify and modify access privileges to users and user groups
- Segregate public and private data
- Protect users' private information such as passwords
- Build web services decoupled from the A&A infrastructure (administrative access to some database may not be necessary) or even provide completely public services without A&A

For these reasons, SSO was chosen as the security mechanism in OpenTox. The principles of SSO and how these bind with REST and OpenTox web services, was previously described in detail in the OpenTox Report on Tools for Access to Confidential Information[10]. The REST API for accessing the SSO infrastructure is described in the OpenTox API 1.2 at http://www.opentox.org/dev/apis/api-1.2/AA

2.2.1 Topological description of access control

The realization of access control in OpenTox is currently based on a central SSO server which is employed by individual web services to decide for a user’s access to them or to other services to which the former act as gateways or proxies. Figure 1 depicts the main concept and how services interact with the single access control manager when a single service is involved.

The client identifies itself providing an authentication token[11] to the OpenTox web service it wants to access. Tokens are generated by the SSO services upon request (over a secure TLS-encrypted connection[12], i.e. a connection using the Transport Layer Security protocol as described by the RFC-5246[13] specifications) of the user's identifier and password (user credentials) and have a certain lifetime. In the current implementation, tokens stay active for 24 hours unless they are invalidated by the client. The web service receives this token,

[10] http://www.opentox.org/data/documents/development/opentoxreports/opentoxreportd33/view?searchterm=D3.3

[11] http://en.wikipedia.org/wiki/Security_token

[12] Transport Layer Security on Wikipedia: http://en.wikipedia.org/wiki/Transport_Layer_Security

[13] http://tools.ietf.org/html/rfc5246

and using the SSO service, checks whether the token is valid (corresponds to a logged in user) and whether that user is granted the necessary privileges to perform the request. If authentication or authorization fails, a status code 401[14] is returned to the user along with an error report[15].

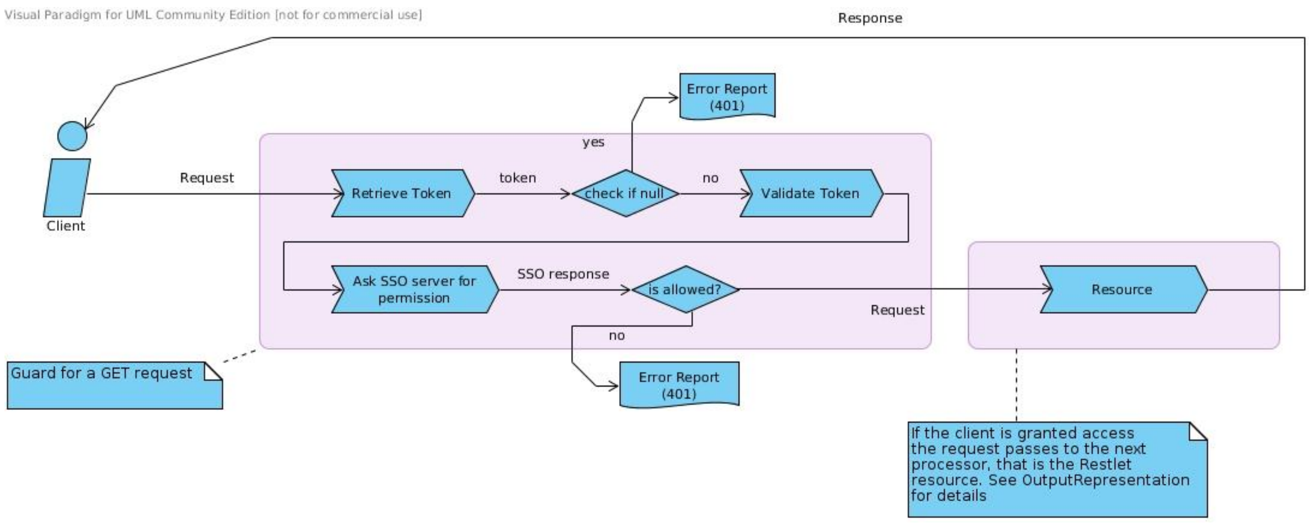

Visual Paradigm for UML Community Edition [not for commercial use]
Response
Error Report (401)
yes
Client
Request
Retrieve Token
token
check if null
no
Validate Token
Ask SSO server for permission
SSO response
is allowed?
Request
Resource
no
Error Report (401)
Guard for a GET request
If the client is granted access the request passes to the next processor, that is the Restlet resource. See OutputRepresentation for details

Figure 1: Protection of confidential information in the request-response chain

In case the initial client request induces a second request from the invoked service, this is always done on behalf of the user using the provided token. This token is passed to the next service(s) of the workflow and in case authorization fails somewhere in the middle, an error report is generated and propagated backwards to the client with a status code 401[16]. In the scheme described in Figure 2, service 1 passes to the remote service the token of the user that initiated the request. In this way, it is guaranteed that an end user will not access either directly or indirectly (through some other service) confidential data, unless he is authorized to do so.

[14] HTTP Status code 401 definition: http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2

[15] OpenTox specifications for Asynchronous Tasks and Error Reports: http://opentox.org/dev/apis/api-1.2/AsyncTask

[16] HTTP Status code 401 - Unauthorized: http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2

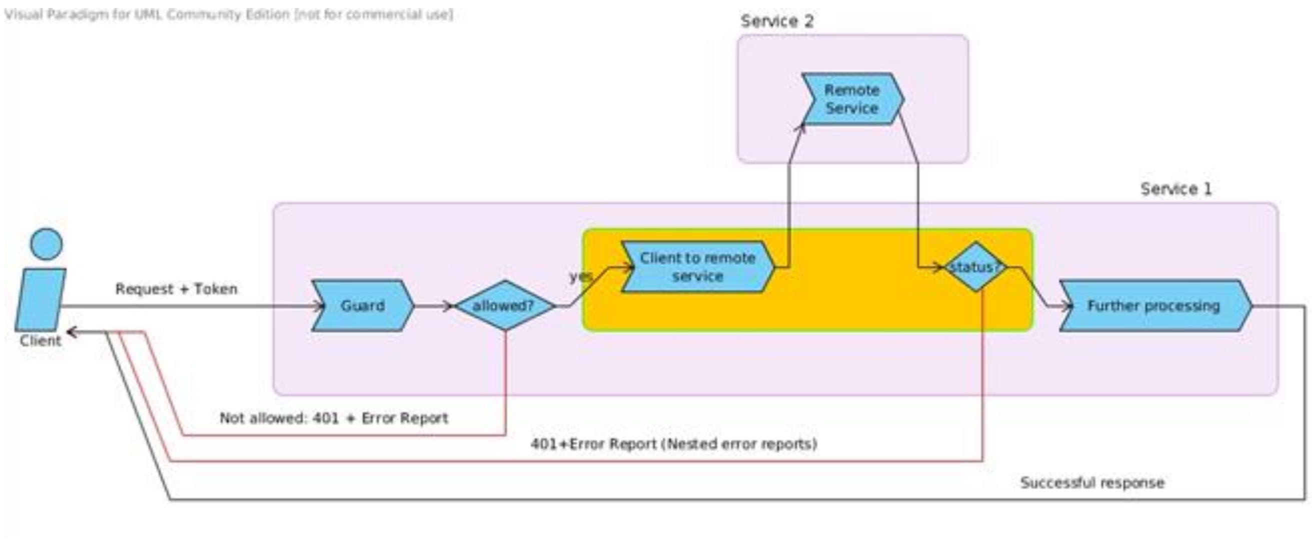


Figure 2: Protection of confidential data in a multi-service application

2.2.2 Managing access

Access to confidential data is secured by the SSO service. The way in which this service allows or blocks an action on an OpenTox web service is specified by the **policy** for the underlying resource. A policy over a resource (identified by its URI) defines to whom access is granted (Figure 3). The authorization policies for the central SSO server are defined by the creator of each resource. SSO policies specify restrictions on the REST level[17] and with respect to some HTTP method. These restrictions apply either on individual users or on groups.

From a programmatic point of view, a policy here is implemented as an XML file specifying explicitly to whom access is allowed and under which conditions. This way a policy defines rules that specify who or what can access these protected resources. The rules are, in effect, permissions describing when and how a user can perform an action on a given protected resource. A user can be an individual or a group. In general, the permissions define what a user can do to which resource and under what conditions.

For OpenTox, we provide a Policy Configuration Service (PCS) to define such preferences and manage the policies. The service allows any registered user to define, modify, and revoke permissions on specific resources (URIs). After creation, only the resource owner (the user who created it) can alter the policy.

[17] For a short explanation and reference to the REST commands, please see: http://www.opentox.org/dev/framework/restweb

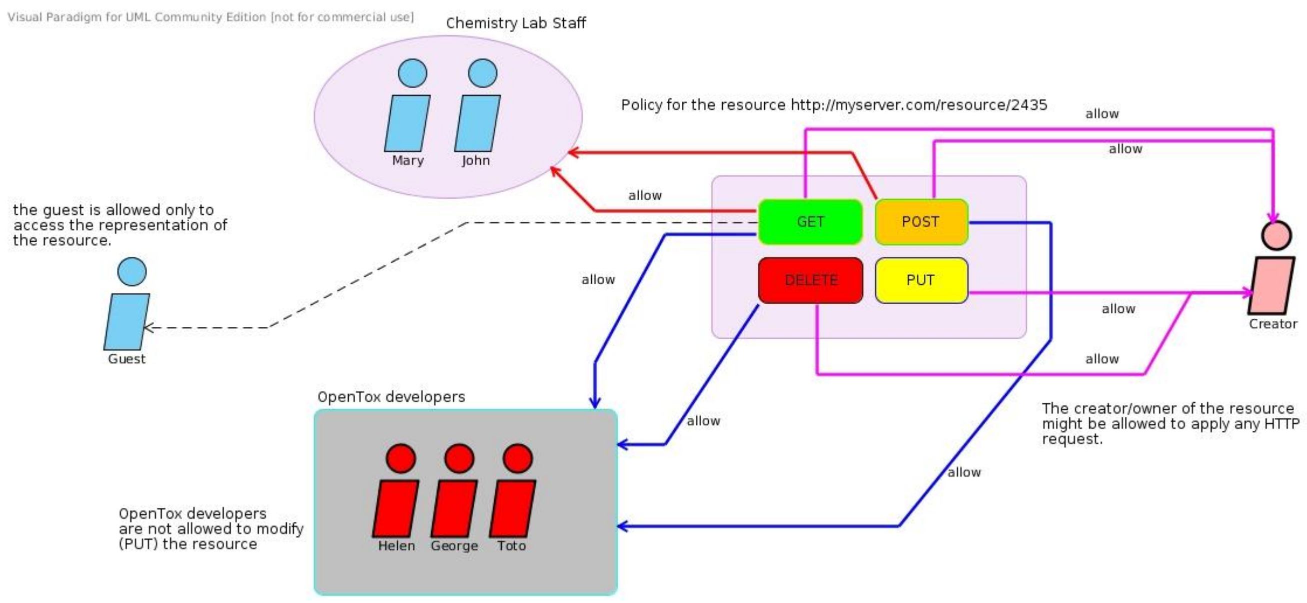

Visual Paradigm for UML Community Edition [not for commercial use]
Chemistry Lab Staff
Mary
John
Policy for the resource http://myserver.com/resource/2435
allow
allow
allow
GET
POST
DELETE
PUT
the guest is allowed only to access the representation of the resource.
Guest
allow
allow
allow
allow
Creator
OpenTox developers
allow
The creator/owner of the resource might be allowed to apply any HTTP request.
allow
OpenTox developers are not allowed to modify (PUT) the resource
Helen
George
Toto

Figure 3: Policy definition – A flexible way to assign privileges to individual users and groups

The policy for a new resource is created by the owner of the resource which is the individual that generates it using some web service. For example when a user uploads a new dataset to a dataset server (using POST) the PCS also creates a policy for it. The policy is created indirectly as the service that accepts the client's request also creates the policy for it. Finally, we provide an example policy XML which is POSTed to the SSO policy service to define access rules for the hypothetical resource http://opentox.org/s2 in Figure 4.

```
<?xml version="1.0" encoding="UTF-8"?>
<Policies>
<Policy name="s2_policy" createdby="id=amadmin,ou=user,dc=opensso,dc=java,dc=net"
lastmodifiedby="id=amadmin,ou=user,dc=opensso,dc=java,dc=net"
creationdate="1275290803394"2">
       <Rule name="s2 rule lastmodifieddate="1275290803394"
              <ServiceName name="iPlanetAMWebAgentService"/>
              <ResourceName name="http://opentox.org/s2"/>
              <AttributeValuePair>
                <Attribute name="POST"/>
                <Value>allow</Value>
              </AttributeValuePair>
              <AttributeValuePair>
                <Attribute name="GET"/>
                <Value>allow</Value>
              </AttributeValuePair>
       </Rule>
       <Subjects name="s2 subject 2" description="">
         <Subject name="amaunz" type="LDAPUsers" includeType="inclusive">
           <AttributeValuePair>
             <Attribute name="Values"/>
             <Value>uid=amaunz,ou=people,dc=opentox,dc=org</Value>
```

Figure 4: A policy in XML format

2.2.3 Policy Creation and management

When a client creates a resource, it should be able to specify a policy for it by passing some parameters to the corresponding service. This can be done for simplicity using POST parameters like "policy=public" or "allow_users_get=john,nick", "allow_users_post=nick" or "allow_groups_get=development,partner" etc. The client should be able to specify a policy by providing an XML document for it. To avoid passing the policy as a form parameter (in MIME-type application/x-form-urlencoded) a Header parameter can be used instead:

```
Policy = "Policy: <XML for policy>"
```

In Figure 5 the way policies are created is presented for the use case of model creation. The user that initiates the training will either provide a policy XML on the header of the request or the policy definition is delegated to the trainer. The default policy for models defines that only the creator is allowed to perform predictions and delete the model while the RDF representation of the model is publicly available. Once the policy for the model is created, only the creator is allowed to modify it and grant specific access to other users and/or groups.

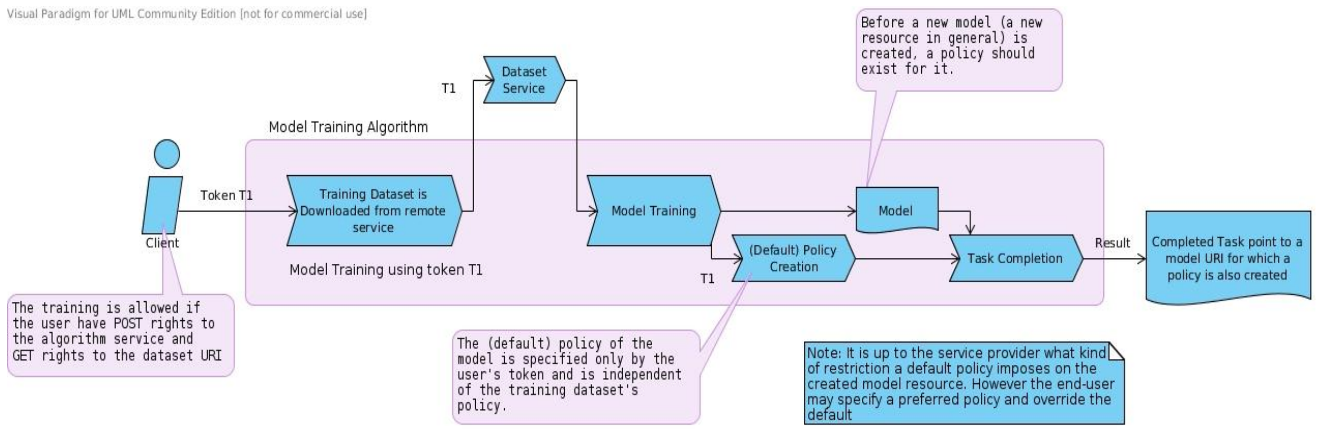

Visual Paradigm for UML Community Edition [not for commercial use]
Dataset Service
T1
Before a new model (a new resource in general) is created, a policy should exist for it.
Model Training Algorithm
Token T1
Client
Training Dataset is Downloaded from remote service
Model Training
Model
(Default) Policy Creation
T1
Task Completion
Result
Completed Task point to a model URI for which a policy is also created
Model Training using token T1
The training is allowed if the user have POST rights to the algorithm service and GET rights to the dataset URI
The (default) policy of the model is specified only by the user's token and is independent of the training dataset's policy.
Note: It is up to the service provider what kind of restriction a default policy imposes on the created model resource. However the end-user may specify a preferred policy and override the default

Figure 5: Policy creation in model training

2.2.4 Evaluation of the current access control system

The following evaluation not only provides an insight, from the web service developer point of view, regarding the access control system of OpenTox but also justifies the *raison d'être* for it. The adoption of SSO as an access control system for OpenTox offers the following features:

1. The web services are designed, implemented and deployed without the need for the maintenance of a local users' database. No administrators or privileged users are needed to deploy an OpenTox web service thus underlining the open nature of the framework since everyone can design and deploy an OpenTox-based service. Therefore, the web services are disengaged from the authentication and authorization infrastructure (A&AI). Phishing[18] opportunities are reduced to a minimum since users provide their credentials only once for every session.
2. Reduces password fatigue[19] as users are not required to remember as many pairs of username and password as the OpenTox web services they need to access. It also reduces the time that the user spends in entering passwords.
3. The client authenticates against the SSO service establishing an encrypted SSL/TLS connection and using it to pass the pair of username and password. No credentials are transferred over unencrypted connections and, more, these credentials are not passed to the individual services. The SSO service verifies the received credentials against the opentox.org's user database (LDAP[20]) which is not exposed to the network whatsoever.
4. The creator of a resource is responsible for its availability (public, private, etc). The data held by a web service and their flow to third party users or groups of such is fully controlled by the creator.

What is considered to be a possible drawback of this design approach is that the SSO server is the most critical node in the system. An outage of the SSO service will affect all services that depend on it.

2.2.5 OECD Principles

Validation services, even when running against confidential data, should satisfy the 5 OECD principles for (Q)SAR validation. In particular, OpenTox validation web services comply with OECD principles 3 and 4, even when confidential data are involved and A&A services are activated:

PRINCIPLE 3: "DEFINED APPLICABILITY DOMAIN"

18 Definition of phishing: http://en.wikipedia.org/wiki/Phishing

19 Definition of password fatigue: http://encyclopedia.thefreedictionary.com/password+fatigue

20 Project page of openLDAP: http://www.openldap.org/

OpenTox provides tools for the determination of applicability domains during the validation of (Q)SAR models against confidential datasets.

PRINCIPLE 4: "APPROPRIATE MEASURES OF GOODNESS-OF-FIT, ROBUSTENESS AND PREDICTIVITY"

OpenTox provides scientifically sound validation routines for the determination of these measures.

2.3 Use Case description

The problem arises when different or seemingly conflicting access privileges are expected to occur regarding models, datasets and other services. A client needs to validate a model against confidential data to which he might have no access. Regarding user privileges, the following alternative cases may occur:

	Access to the test dataset	
Access to the QSAR model	**Yes/Yes**	Yes/No
	No/Yes	**No/No**

The most representative cases are the Y/Y and the N/N case (as the Y/N and N/Y cases are actually sub-cases of the N/N case). In case that the user has access both to the test dataset and the QSAR model (Y/Y), the framework has to take care of the access privileges on any resources (datasets) created as predictions from the model so that confidential information will not leak from the validation service. Current access control infrastructure of OpenTox, combined with the REST architecture, caters for the protection of all these resources. In the second case, where the user has not access either to the model or to the test dataset, it becomes evident that a second user with enhanced privileges has to intervene and perform the validation on behalf of the first user exposing back to him just the validation report but no information regarding the test set and/or the model. For validation purposes OpenTox could provide a facility to test (Q)SAR models remotely against confidential datasets without getting access to the actual entries of the database to ensure security and confidentiality of proprietary data. This use case is an extended feature and is briefly described in Section 5 of this report.

2.3.1 Use Case implementation

Current OpenTox implementations support the case where the end user provides his own dataset or has access to the confidential dataset. When confidential data are held by public servers and these are to be used in a validation session, it should be clear which new resources that are created replicate some part of these data and under what kind of policies these resources are created. Validation lies in between all other OpenTox services and creates models and datasets on behalf of the end user. In Figure 6 the validation procedure is described regarding the service invocations involved.

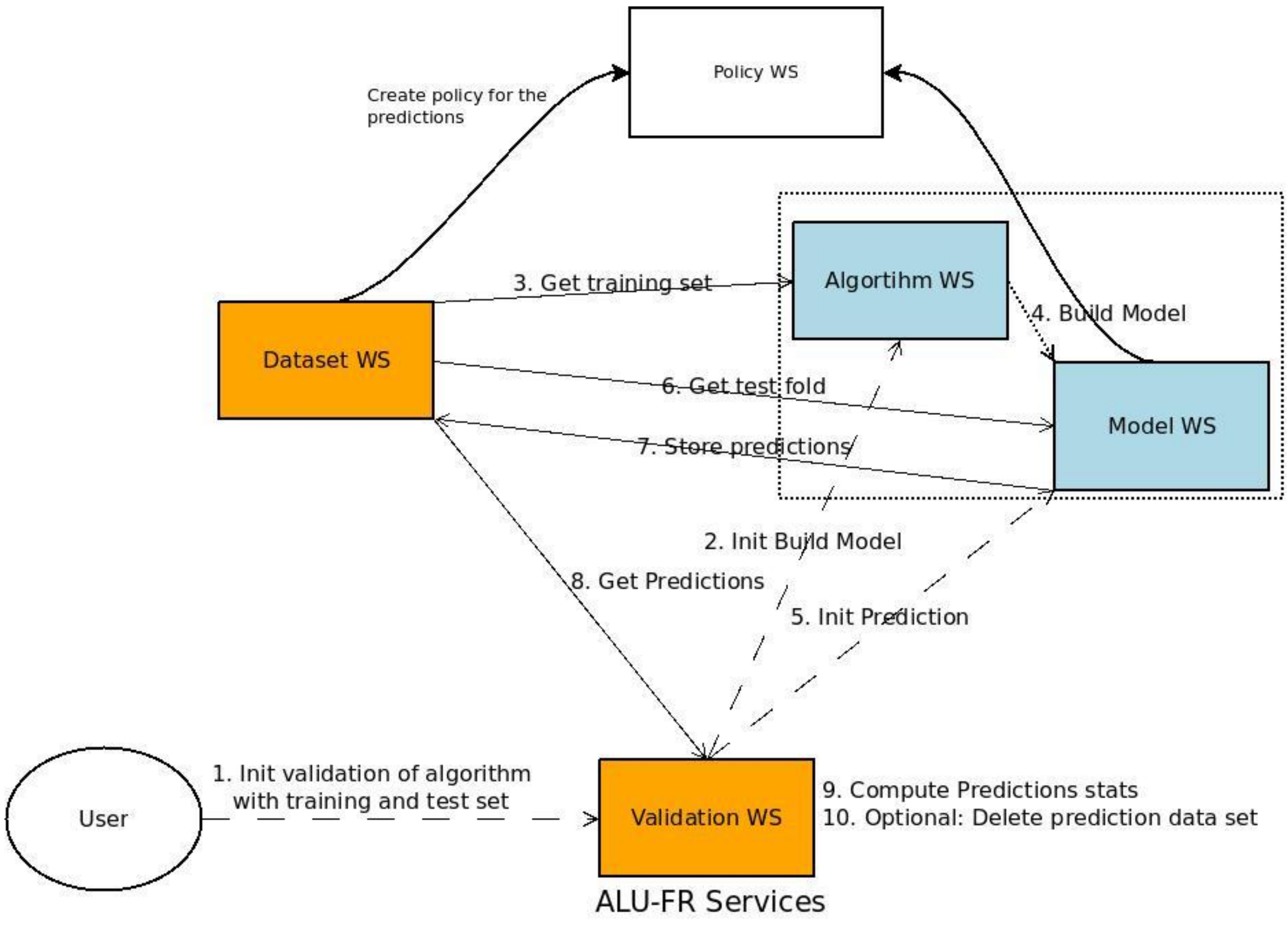


Figure 6: Topological description of an OpenTox-compliant validation service (interactions with other web services)

All service invocations mentioned above (validation request, model training, predictions) are performed using the end user's authentication token. In case a new resource is to be created, as for example in the case a model is trained or a dataset with predicted values is created on a dataset service, a policy is defined by the corresponding service that creates the resource (using again the user's token) and is POSTed to the policy service. All created models and datasets with predictions “belong” to the user that initiated the validation and only that user can amend their access options.

3.Validation examples using confidential data

Two different examples are presented in this section. The first example shows the seamless integration of the A&A concept into the OpenTox application ToxCreate. The user does not have to worry about security issues while he benefits from the comfort of a Graphical User Interface (GUI). The second example gives more technical insights: a remote confidential dataset is validated using the command line tool cURL[21]. This example emphasizes how confidentiality is guaranteed with locally distributed web services.

3.1 Validation against confidential data with ToxCreate

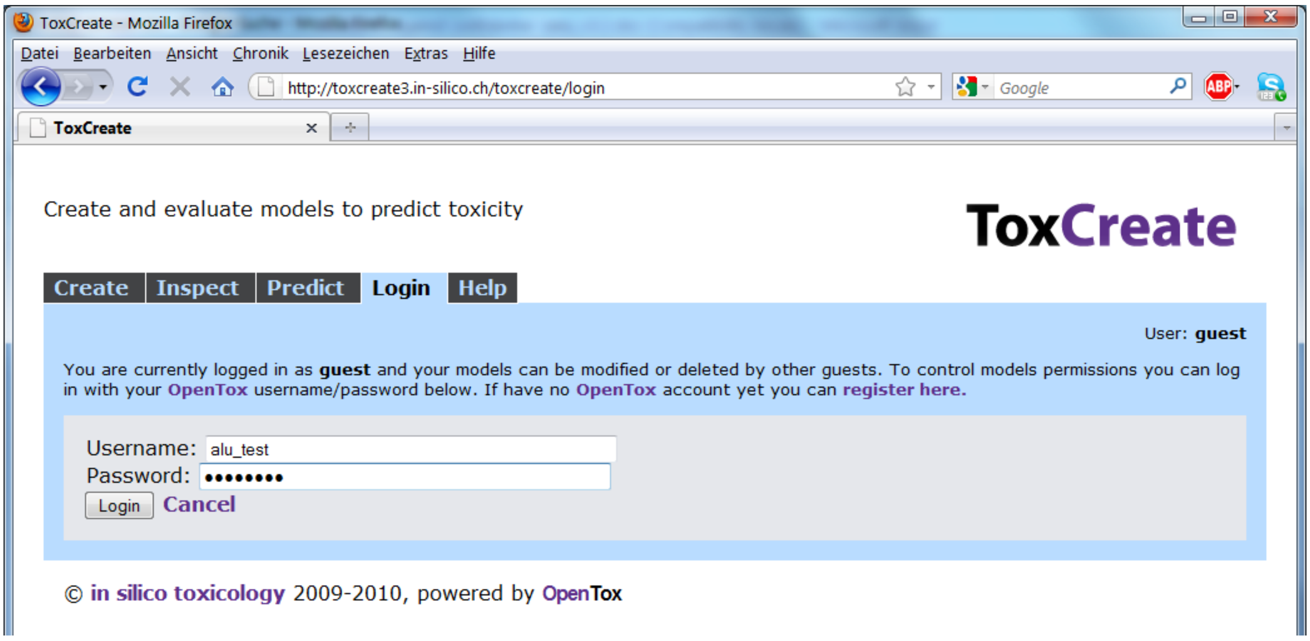
ToxCreate (www.toxcreate.org[22]) is a web-based application developed within the OpenTox framework. It is based on various OpenTox web services and provides model creation, validation and the prediction of compounds with the created models[23]. At this stage, ToxCreate only supports Lazar as a prediction algorithm and model but this will be extended in Spring 2011 to apply more generally for OpenTox algorithms. The user can use already created models, or upload a new dataset to train a new model. This example focuses on the latter use case: assuming that the data provided by the user is confidential, no other user should be able to access the uploaded dataset or resources created on the basis of this dataset, unless the creator provides an override to this default to specific users.

In ToxCreate the user is automatically logged in as guest. The application is organized with multiple tabs. The user can login at the 'Login' tab. This demonstration is performed with the test-user 'alu_test' (password is 'alu_test' as well).

[21] URL is a command-line tool serving as an HTTP client. See http://curl.haxx.se/

[22] The current production version of ToxCreate running at http://toxcreate.org will soon be updated with a version that supports A&A as described in this section. The new development version can be currently found at http://toxcreate3.in-silico.ch/toxcreate

[23] For more info on ToxCreate see http://opentox.net/dev/testing/testcasedevelopment/toxcreate


ToxCreate - Mozilla Firefox
Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe
http://toxcreate3.in-silico.ch/toxcreate/login
ToxCreate
Create and evaluate models to predict toxicity
ToxCreate
Create
Inspect
Predict
Login
Help
Login
Cancel

User: **guest**

You are currently logged in as **guest** and your models can be modified or deleted by other guests. To control models permissions you can log in with your **OpenTox** username/password below. If have no **OpenTox** account yet you can **register here**.

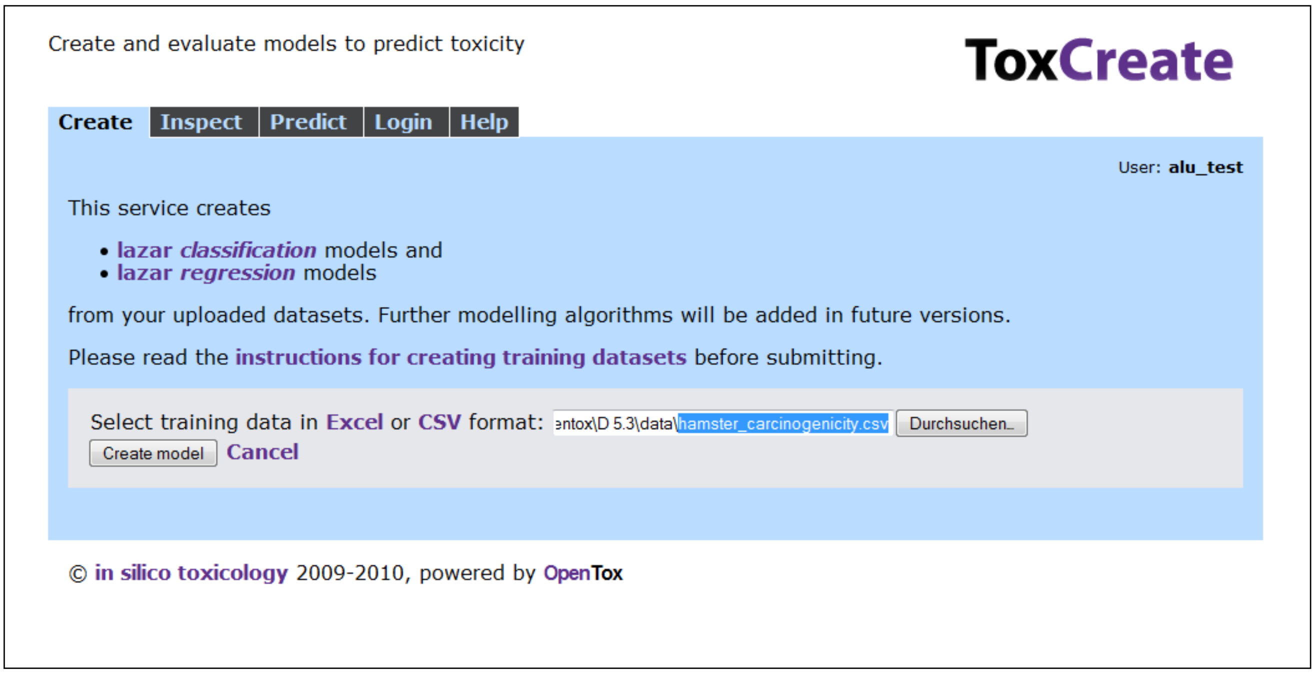
Username: alu_test
Password: ••••••••

© **in silico toxicology** 2009-2010, powered by **OpenTox**

Figure 7: Login screen of ToxCreate

After logging in (Figure 7), the current user switches from 'guest' to 'alu_test', as shown on the top right of the web page. It is now possible to safely upload the confidential dataset. For this example a publicly available dataset from the Carcinogenic Potency Database (CPDB) was chosen: the hamster carcinogenicity dataset contains 85 compounds[24]. A binary target variable indicates whether the compound is active or inactive. This dataset can be uploaded from your local hard drive at the 'Create' tab of ToxCreate (Figure 8).

24 Available at https://github.com/helma/opentox-test/blob/master/data/hamster_carcinogenicity.csv


Create and evaluate models to predict toxicity

ToxCreate

User: alu_test

This service creates

- lazar *classification* models and
- lazar *regression* models

from your uploaded datasets. Further modelling algorithms will be added in future versions.

Please read the **instructions for creating training datasets** before submitting.

Select training data in Excel or CSV format: entox\D 5.3\data\hamster_carcinogenicity.csv

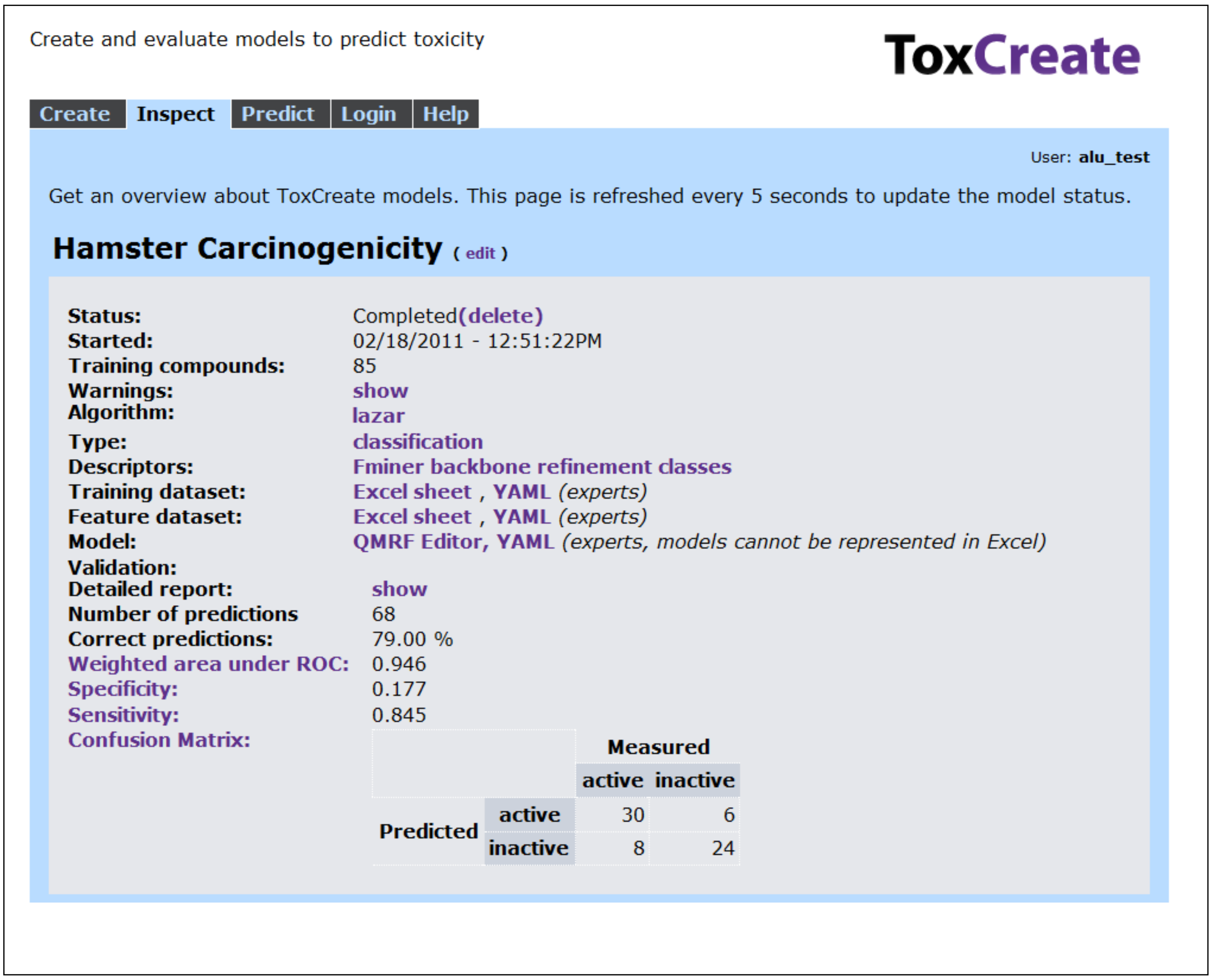
© in silico toxicology 2009-2010, powered by OpenTox

Figure 8: User interface provide by ToxCreate for uploading a training dataset

When the ‘Create model’ button is pressed, ToxCreate automatically switches to the ‘Inspect’ tab. In the background the dataset is uploaded and a model building and validation process is initialized:

- The dataset is uploaded to the dataset web service, and registered at the A&A server to allow access to user ‘alu_test’ (and the group of this user) only.
- Structural features are mined on this dataset and a lazar model is built. This model can be used later on to make predictions (‘Predict’ tab).
- A 10-fold cross-validation to evaluate the predictive power of this model on the dataset is performed. This splits the datasets into 10 folds, and repeatedly builds a model on 9 different folds of the 10 dataset folds. The resulting model is used to predict the test dataset (the fold that was left out when building the model). The final results of this cross-validation are shown in the validation section of the models properties. More details are available in the validation report.
- Finally a QMRF report is automatically created for this model. It contains meta-information on the trained model and the algorithm, the validation results, and other information about the model.

The results of these steps are gradually added to the new ‘Hamster Carcinogenicity’ model that is available on the ‘Inspect’ tab, until the status is finally set to completed (Figure 9).


Create and evaluate models to predict toxicity

User: **alu_test**

Get an overview about ToxCreate models. This page is refreshed every 5 seconds to update the model status.

Hamster Carcinogenicity (edit)

Status:	Completed(delete)
Started:	02/18/2011 - 12:51:22PM
Training compounds:	85
Warnings:	show
Algorithm:	lazar
Type:	classification
Descriptors:	Fminer backbone refinement classes
Training dataset:	Excel sheet , YAML *(experts)*
Feature dataset:	Excel sheet , YAML *(experts)*
Model:	QMRF Editor, YAML *(experts, models cannot be represented in Excel)*
Validation:	
Detailed report:	show
Number of predictions	68
Correct predictions:	79.00 %
Weighted area under ROC:	0.946
Specificity:	0.177
Sensitivity:	0.845
Confusion Matrix:	

		Measured	
		active	**inactive**
Predicted	**active**	30	6
	inactive	8	24

Figure 9: An overview of a QSAR model produced within the ToxCreate application

The user can now have a look at the detailed validation report, edit the QMRF report with the QMRF editor, or make predictions with the newly created model.

Note that the resources (dataset, model, validation report, etc...) are only available to user 'alu_test'. After logging out, the user 'guest' has no access to the newly created Model 'Hamster Carcinogenicity'; it is not available in the 'Inspect' tab of ToxCreate.

3.2 Validation against validation data using distributed web services

This example demonstrates how a dataset is protected by A&A and SSL when used for validation. The use case is a training-test-split validation. This is an established method to estimate the performance of a prediction model on unseen data[25]: the original dataset is split into a training data set and a test data set. The training data set is used to build a model. The model is then applied to make predictions on the unseen test dataset.

In the use case presented here, we use the well-known Caco-2 dataset[26]. The dataset consists of 100 organic molecules with a numeric endpoint (Caco-2 permeability, $\log P_{app}$). The dataset was uploaded to the AMBIT2 dataset service (It is available with A&A at https://ambit.uni-plovdiv.bg:8443/ambit2/dataset/R401560). 27 numerical features have been calculated using AMBITs descriptor calculation services[27]. A linear regression algorithm (https://ambit.uni-plovdiv.bg:8443/ambit2/algorithm/LR) was selected to predict the target variable. Both, the dataset as well the algorithm service are located in Sofia, Bulgaria. The validation server is located at the University of Freiburg, Germany.

We are executing this example with the command line tool curl (http://curl.haxx.se/), using the functionality specified in the OpenTox API[28]. Alternatively, the REST calls could be performed with any programming language that includes a REST library. To this end, the validation routines can be integrated into an application with a GUI (like in the ToxCreate example above).

Login:

The first step is to derive a subject-id from the SSO-server, for the user 'guest': [29]

[25] More extensive techniques like cross-validation should be preferred especially if the training dataset is small. The simpler training test split method is chosen for proof of concept.

[26] http://pubs.acs.org/doi/suppl/10.1021/ci049884m

[27] See for example: https://ambit.uni-plovdiv.bg:8443/ambit2/algorithm/org.openscience.cdk.qsar.descriptors.molecular.AtomCountDescriptor

[28] Documentation for the OpenTox API http://opentox.org/dev/apis/api-1.2

[29] The cURL calls (presented in purple boxes) can be copied to and executed with a command-line interface. The return value is marked in green (success) or orange (error) boxes.

The curl call returns the following:

```
curl -X POST -d "username=guest" -d "password=guest" http://opensso.in-silico.ch/opensso/identity/authenticate?uri=service=openldap
```

```
token.id= AQIC5wM2LY4SfczngIclWu3ztAWK7WKXHfAFK+CI8Rvf5zU=@AAJTSQACMDE=#
```

This string can now be used to identify the client as user 'guest' in the subsequent cURL calls. The access to all resources that are created with this subject-id will only be granted to user 'guest' (and the group of this user).

Start validation:

The validation is initialized with a HTTP POST call to http://opentox.informatik.uni-freiburg.de/validation/training_test_split. The parameters to control the routine are algorithm-URI, dataset-URI and prediction-feature. The subject-id is specified as additional header (with -H option):

```
curl -X POST -d algorithm_uri="https://ambit.uni-plovdiv.bg:8443/ambit2/algorithm/LR" -d dataset_uri="https://ambit.uni-plovdiv.bg:8443/ambit2/dataset/R401560" -d prediction_feature="https://ambit.uni-plovdiv.bg:8443/ambit2/feature/22190" http://opentox.informatik.uni-freiburg.de/validation/training_test_split -H "subjectid:AQIC5wM2LY4SfczngIclWu3ztAWK7WKXHfAFK+CI8Rvf5zU=@AAJTSQACMDE=#"
```

```
http://opentox.informatik.uni-freiburg.de/task/1004
```

The validation service returns the URI of a task object, while running the validation as an asynchronous background job. The result is stored in the task object when the job is finished.

Get validation result-URI:

Task resources are not protected, which is why the following cURL call does not need to include the subject-id:

```
curl http://opentox.informatik.uni-freiburg.de/task/1004 -H "Accept:application/x-yaml"
```

```
---
http://purl.org/dc/elements/1.1/title: Perform training test split validation
http://www.opentox.org/api/1.1#hasStatus: Completed
http://www.opentox.org/api/1.1#resultURI: http://opentox.informatik.uni-freiburg.de/validation/114
http://www.opentox.org/api/1.1#percentageCompleted: 100.0
[…]
```

Using cURL on the task-URI returns a list of its properties, including the field result-URI that contains the validation-URI.

Get validation result:

The following cURL call demonstrates that the validation result http://opentox.informatik.uni-freiburg.de/validation/114 is protected by the A&A routines:

```
curl http://opentox.informatik.uni-freiburg.de/validation/114
```

```
--- !ruby/object:OpenTox::ErrorReport
actor: http://opentox.informatik.uni-freiburg.de/validation/114
errorType: OpenTox::NotAuthorizedError
http_code: 401
message: Not authorized
[…]
```

Access is denied, and an error report is returned instead. However, we can access this resource when specifying the subject-id:

```
curl http://opentox.informatik.uni-freiburg.de/validation/114 -H
"subjectid:AQIC5wM2LY4SfczngIclWu3ztAWK7WKXHfAFK+CI8Rvf5zU=@AAJTSQACMDE=#"
```

```
---
http://www.opentox.org/api/1.1#model: https://ambit.uni-plovdiv.bg:8443/ambit2/model/35009
http://www.opentox.org/api/1.1#trainingDataset: http://opentox.informatik.uni-freiburg.de/dataset/452
http://www.opentox.org/api/1.1#predictionDataset: http://opentox.informatik.uni-freiburg.de/dataset/454
http://www.opentox.org/api/1.1#predictionFeature: https://ambit.uni-plovdiv.bg:8443/ambit2/feature/22190
http://www.opentox.org/api/1.1#numInstances: 32
http://www.opentox.org/api/1.1#testTargetDataset: https://ambit.uni-plovdiv.bg:8443/ambit2/dataset/R401560
http://www.opentox.org/api/1.1#validationType: training_test_split
http://www.opentox.org/api/1.1#testDataset: http://opentox.informatik.uni-freiburg.de/dataset/453
http://www.opentox.org/api/1.1#algorithm: https://ambit.uni-plovdiv.bg:8443/ambit2/algorithm/LR
http://www.opentox.org/api/1.1#regressionStatistics:
  http://www.opentox.org/api/1.1#rootMeanSquaredError: 0.627121310539419
  http://www.opentox.org/api/1.1#rSquare: 0.352408295243186
```

The validation object links to resources that have been used for validation, e.g., training and test data sets. The latter have been created by splitting the original dataset, and are located at Freiburg’s dataset service.

Create report:

We finally create a validation report from the validation resource:

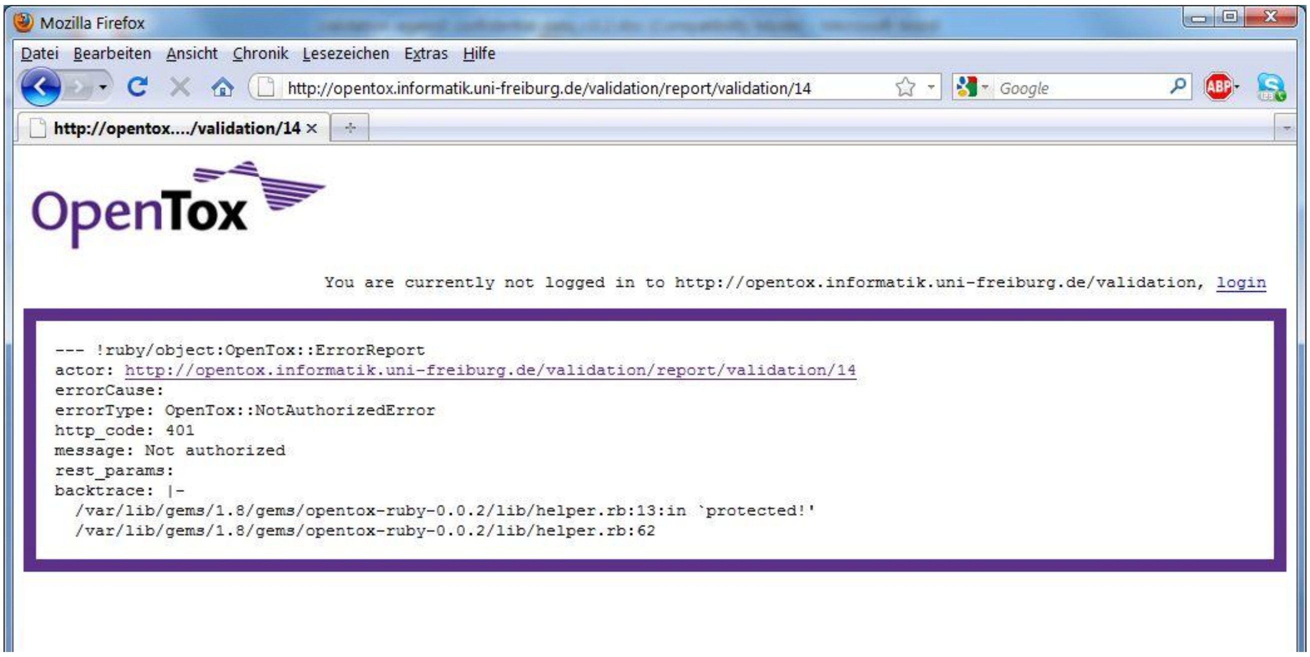
```
curl -X POST -d validation_uris="http://opentox.informatik.uni-freiburg.de/validation/114" http://opentox.informatik.uni-freiburg.de/validation/report/validation -H
"subjectid:AQIC5wM2LY4SfczngIclWu3ztAWK7WKXHfAFK+CI8Rvf5zU=@AAJTSQACMDE=#"
```

Again this call returns a task object first (skipped for simplicity). Accessing this task reveals the report-URI:

```
---
http://www.opentox.org/api/1.1#resultURI: http://opentox.informatik.uni-freiburg.de/validation/report/validation/14
[...]
```

Visit validation report with browser:

The validation report could be requested with cURL as well, but is best viewed with a web browser. As it is protected by A&A, access is denied for http://opentox.informatik.uni-freiburg.de/validation/report/validation/14 (Figure 10).


Mozilla Firefox
Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe
http://opentox.informatik.uni-freiburg.de/validation/report/validation/14
http://opentox..../validation/14
Google
OpenTox

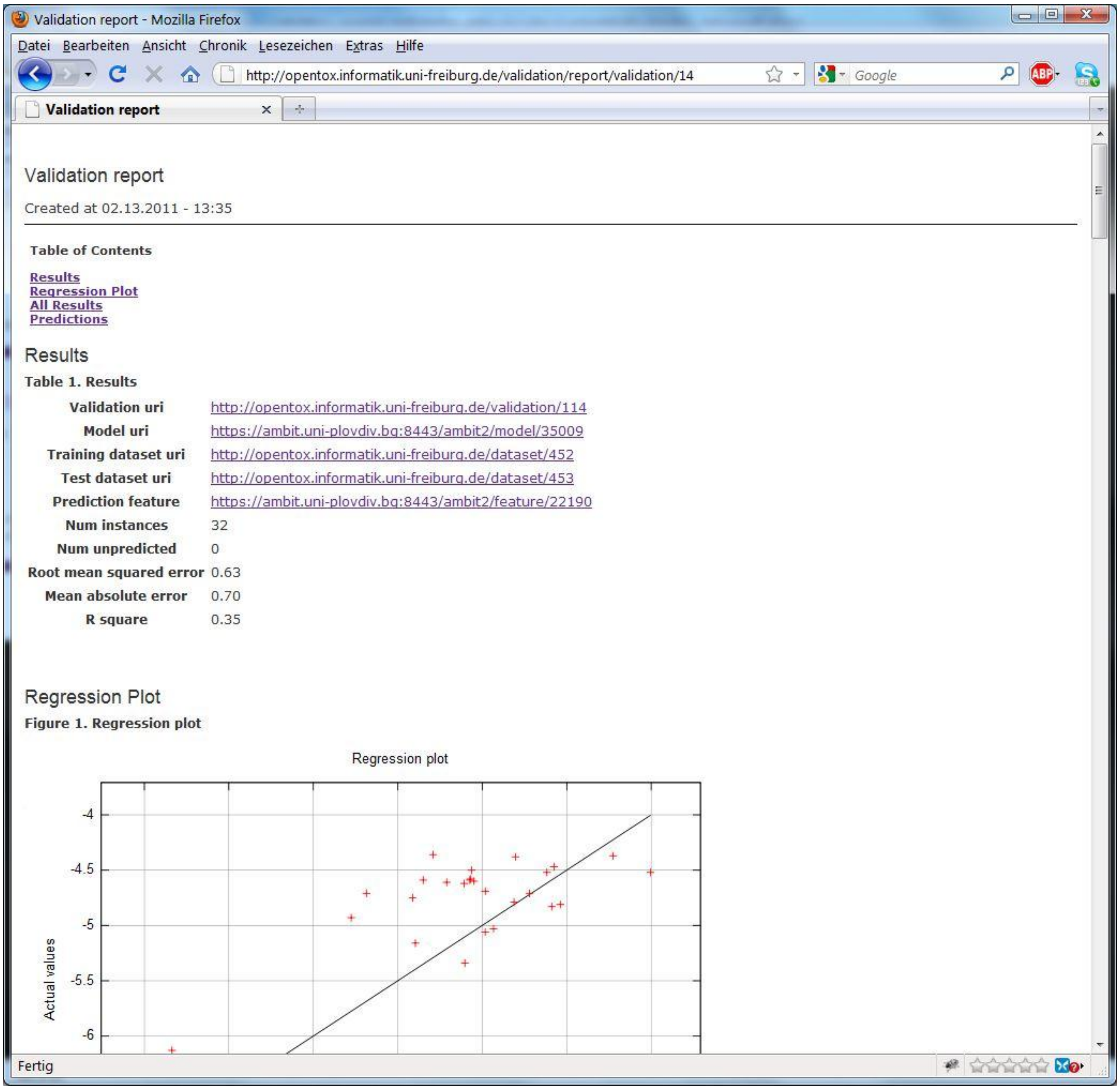
You are currently not logged in to http://opentox.informatik.uni-freiburg.de/validation, login

```
--- !ruby/object:OpenTox::ErrorReport
actor: http://opentox.informatik.uni-freiburg.de/validation/report/validation/14
errorCause:
errorType: OpenTox::NotAuthorizedError
http_code: 401
message: Not authorized
rest_params:
backtrace: |-
  /var/lib/gems/1.8/gems/opentox-ruby-0.0.2/lib/helper.rb:13:in `protected!'
  /var/lib/gems/1.8/gems/opentox-ruby-0.0.2/lib/helper.rb:62
```

Figure 10: Protected resources do not allow unauthorized access – access has been denied to a validation report

The user has to login as guest[30] (using the link at the top right of the browser, password is 'guest'). This stores the subject-id in a cookie that will identify the user to the validation web service with the web browser. Hence, the second attempt to visit http://opentox.informatik.uni-freiburg.de/validation/report/validation/14 is successful. The user is provided with the validation report (Figure 11).

[30] Login screen for validation service: http://opentox.informatik.uni-freiburg.de/validation/login



Validation report - Mozilla Firefox

Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe
http://opentox.informatik.uni-freiburg.de/validation/report/validation/14
Google
Validation report

Validation report

Created at 02.13.2011 - 13:35

Table of Contents

Results
Regression Plot
All Results
Predictions

Results

Table 1. Results

Validation uri	http://opentox.informatik.uni-freiburg.de/validation/114
Model uri	https://ambit.uni-plovdiv.bg:8443/ambit2/model/35009
Training dataset uri	http://opentox.informatik.uni-freiburg.de/dataset/452
Test dataset uri	http://opentox.informatik.uni-freiburg.de/dataset/453
Prediction feature	https://ambit.uni-plovdiv.bg:8443/ambit2/feature/22190
Num instances	32
Num unpredicted	0
Root mean squared error	0.63
Mean absolute error	0.70
R square	0.35

Regression Plot

Figure 1. Regression plot

Figure 11: Successful access to the validation report

4. Validation against confidential data using standalone version

4.1 Standalone Installation of OpenTox Web Services

A distributed system over the Internet has conveniency and extensibility advantages due to its high flexibility and easy integration of new services. However, its security is questioned by certain users, because even when encrypted, data are still transferred over the Internet and it might be possible for someone to eavesdrop the communication and steal sensitive information (Figure 12). The authorization and authentication strategy adopted in OpenTox and the overall security system provide a high level of protection of confidential data. We understand, however, that toxicity data can be considered highly confidential and sensitive by their owners and even a slight possibility of leakage might be an obstacle for potential end users of OpenTox services and applications.

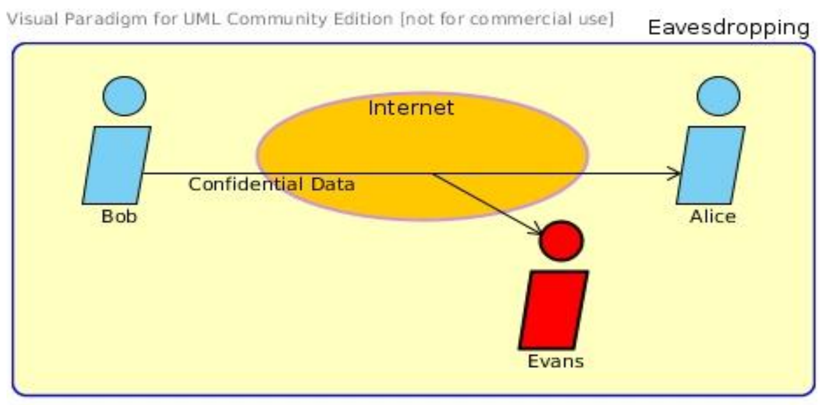

Visual Paradigm for UML Community Edition [not for commercial use]
Eavesdropping
Internet
Confidential Data
Bob
Alice
Evans

Figure 12: Eavesdropping of sensitive information

In order to minimize as much as possible the risk of data leakage, OpenTox offers an alternative implementation of the use case, which is based on a stand-alone local installation of some or all the OpenTox services. The alternative approach is considered as the most secure way to seal data, namely to protect them physically prohibiting any kind of interaction with others and restraining their mobility within an isolated system. Complete physical isolation of the system means that the application runs on a machine either disconnected from the Internet (or any other network) or protected by means of firewalls. A virtual private network can also be established (Figure 13), again isolating the nodes of the distributed application from the rest of the network and also protecting the client-server communication using secure cryptographic tunnelling protocols.

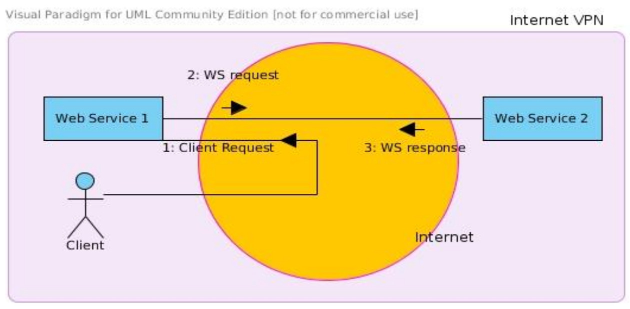


Figure 13: Topological structure of a virtual private network established over the Internet or some Local Area Network

The local installation of OpenTox web services along with web interfaces that facilitate their consumption is feasible since all projects under OpenTox are freely distributed (executables, documentation and source code) and are available on-line from the www.opentox.org website[31]. All service providers offer the ability to download and install individual web service implementations locally either as standalone applications or in a Servlet container (such as Apache Tomcat[32]). In the case of a servlet container, the web service implementations come as “web archive” files (.war).

In all cases documentation is provided regarding the installation of prerequisites such as the **MySQL database server** or a **J2EE-compatible servlet container** such as Apache Tomcat.

4.2 Standalone Installation of particular services

This section describes how a user can install locally three OpenTox services namely AMBIT, Jaqpot, and ToxCreate. The AMBIT web service package is one of the several existing independent implementations of the OpenTox Application Programming Interface and is built according to the principles of the Representational State Transfer (REST) architecture. The Open Source Predictive Toxicology Framework, developed by partners of the EC FP7 OpenTox project, aims at providing a unified access to toxicity data and predictive models, as well as validation procedures. This is achieved by i) an information model, based on a common OWL-DL ontology; ii) links to related ontologies; iii) data and algorithms, available through a standardized REST web services interface, where every compound, data set or predictive method has a unique web address, used to retrieve its Resource

31 OpenTox downloads: http://www.opentox.org/downloads

32 Apache Tomcat home page: http://tomcat.apache.org/index.html

Description Framework (RDF) representation, or initiate the associated calculations. The Jaqpot web services are OpenTox API 1.2-compliant web services. Jaqpot is a web application that supports model training and data preprocessing algorithms such as multiple linear regression, support vector machines, neural networks (an in-house implementation based on an efficient algorithm), an implementation of the leverage algorithm for domain of applicability estimation and various data preprocessing algorithms such as PLS and data cleanup. Jaqpot also comes with a web service for storing BibTex[33] entries which become also available in JSON and RDF formats. Jaqpot provides asynchronous execution of tasks submitted by users, authentication, authorization and accounting mechanisms powered by OpenSSO and two monitoring access points mounted at /monitoring and /status.

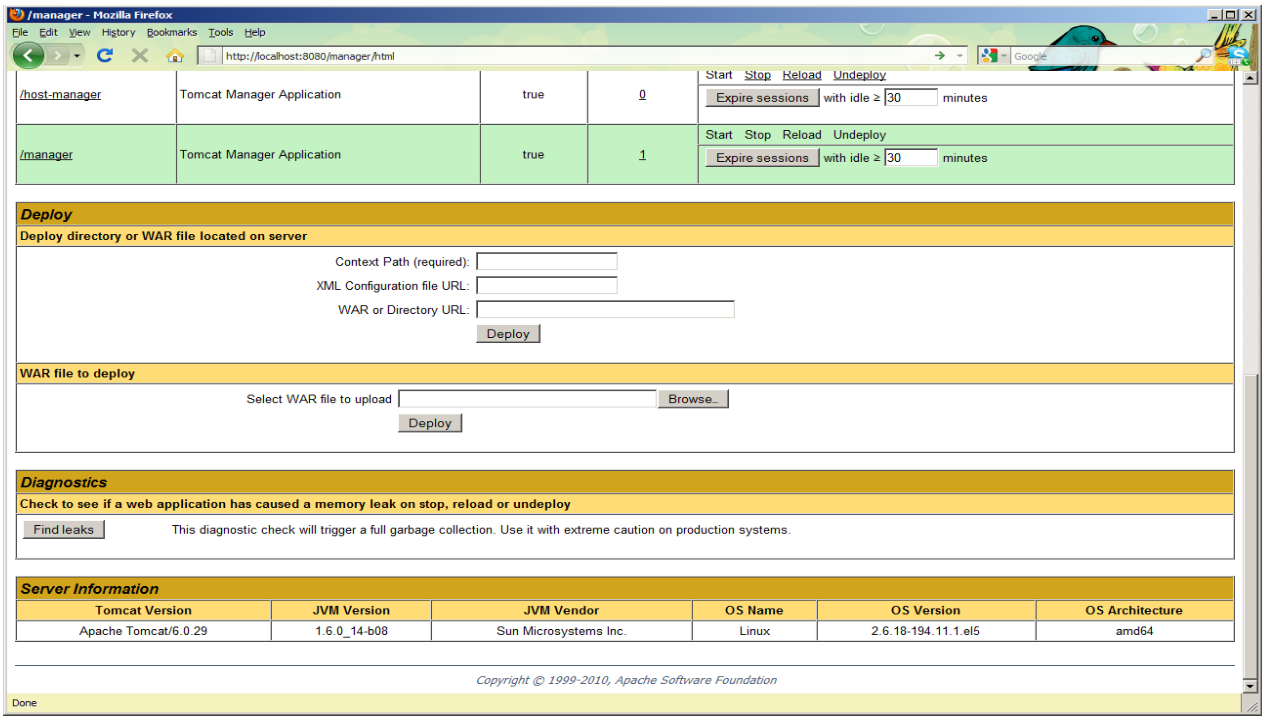
ToxCreate is a QSAR web application that has been developed in OpenTox. It derives nearest neighbours of the query structure and uses those to learn a model. Currently, it is being extended to accommodate any OpenTox-compliant model and dataset service.

4.2.1 AMBIT

The user downloads the AMBIT 2.0 application (http://www.ideaconsult.net/downloads/ambit2/ambit2.war) and saves the file ambit2.war. With his web browser, he navigates to http://localhost:8080, and clicks on “Tomcat Manager” in the Administration box at the top-left of the screen. He is prompted to enter the user name and password of the Tomcat manager/administrator he has set up. On the manager page, he scrolls to the bottom and finds the box entitled “WAR file to deploy”[34].

[33] BibTeX specifications online: http://www.bibtex.org/

[34] More documentation regarding deployment on a Tomcat servlet container can be found online at http://tomcat.apache.org/tomcat-6.0-doc/deployer-howto.html



/manager - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://localhost:8080/manager/html

/host-manager	Tomcat Manager Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/manager	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

Deploy

Deploy directory or WAR file located on server

Context Path (required):

XML Configuration file URL:

WAR or Directory URL:

Deploy

WAR file to deploy

Select WAR file to upload Browse...

Deploy

Diagnostics

Check to see if a web application has caused a memory leak on stop, reload or undeploy

Find leaks This diagnostic check will trigger a full garbage collection. Use it with extreme caution on production systems.

Server Information

Tomcat Version	JVM Version	JVM Vendor	OS Name	OS Version	OS Architecture
Apache Tomcat/6.0.29	1.6.0_14-b08	Sun Microsystems Inc.	Linux	2.6.18-194.11.1.el5	amd64

Copyright © 1999-2010, Apache Software Foundation

Done

Figure 14: Screenshot from the deployment of AMBIT on a tomcat servlet container

Under “WAR file to deploy”, he clicks “Browse...”, finds ambit2.war and clicks “Deploy”. Following these steps, he has successfully installed the AMBIT 2.0 implementation of the OpenTox REST API. If he next navigates to http://localhost:8080/abmit2 he should see the welcome screen of AMBIT2. As explained in the installation instructions for AMBIT 2.0 (ambit.sourceforge.net), this release (September 2010) comes without an embedded database (Figure 14).

An empty database can be created using cURL (curl.haxx.se). On many linux systems, cURL can be easily installed from a package repository using a standard package manager. It can also be downloaded from http://curl.haxx.se/download.html. Under Windows, there are two options for using cURL: 1) installing cURL natively, preferably using the most recent generic Win32 version: http://curl.haxx.se/download.html, or 2) installing the VMWare Player (see online http://www.vmware.com/products/player/) and running a small Linux environment (http://www.maunz.de/opentox/dsl-4.1.zip) under Windows (after installing VMWare Player and unpacking the dsl-4.1.zip file, just double-click the dsl-4.1.vmx file). Under Linux, after installing cURL, the following command can be typed as root in a console:

```
$ curl -X POST -d "dbname=ambit2" -d "user=mysqladminuser" -d
"pass=mysqladminpass" http://localhost:8080/ambit2/admin/database
```

4.2.2 Jaqpot

A user can download and install Jaqpot on their local machine following these instructions (Tested on Ubuntu, Debian and Mac OS X operating systems). First, he needs to install in his system the following:

1. MySQL database server and client
2. Maven2
3. Git

The dependencies #2 and #3 are optional but will facilitate a lot the installation of Jaqpot. The commands needed for downloading the latest version of Jaqpot are the following:

```
$git clone git://github.com/alphaville/jaqpot.git
$cd jaqpot/
$mvn clean package tomcat:run
```

and Jaqpot will start on port 8080 (see http://localhost:8080/jaqpot).

4.2.3 ToxCreate

ToxCreate is distributed as a ready-to-use virtual machine (appliance). This offers several advantages over traditional installers:

- The appliance is deployed anywhere in minutes, not even administrative privileges are needed on a Windows machine[35].
- The appliance can be deployed as is in virtual servers or cloud-based services. By extracting the filesystem from the virtual hard disk, the Linux operating system is also installed quickly on a physical machine.

A local installation of ToxCreate is a fully functional QSAR solution and offers complete privacy. Installation instructions and downloads can be found on:

https://github.com/helma/opentox-documentation/wiki/Installation-of-IST-OpenTox-webservices

[35] See VirtualBox portable version, http://www.vbox.me

5. Further work

5.1 Description of the extended use case

The extended use case of “validation against confidential data” aims at the creation of validation reports for datasets that are not accessible by any means to a user. No prediction data or parts of the initial dataset should be exposed to the user and any intermediate resources related to the test data should be protected in such a way that the end user is denied access. A validation report for the model is created along with a proper policy that allows the end user to access it. Whether the confidential data should be available for validation is to be decided by the data owner. If yes, then a *trusted proxy service* is endowed with (GET) privileges on these data, *i.e.* is given privileged credentials. The proxy retains a database relating confidential dataset URIs with credentials to access them. End users submit tasks to that service to perform a validation for a dataset to which they don't have access.

5.2 Implementation of the extended use case

The extended use case can be implemented using a proxy service which acts as a broker. Having enhanced privileges compared to the end user, this broker service can perform all necessary actions on behalf of the user and return back only what is allowed. The privileged credentials are handed to the webmasters of the proxy service once they have presented their privacy policy to the data owner and upon common agreement. The proxy service providers assure that the data will not leak and also provide a transparent framework since the source code of the service is publicly available.

The proposed structure is presented in Figure 15. Being generic enough, it can be used to tackle similar problems where the user needs to access part of the results, that only a more privileged user can access or acquire. The end user provides its token *T1* to the proxy service along with a validation request. The client is not actually aware whether it refers to an actual validation service or to a proxy since the request is identical with the difference that the actual service is protected behind the proxy. The proxy service acquires a privileged token *T2* that corresponds to the test dataset URI that the user has provided. If the proxy service does not have any stored credentials for the test dataset, then it tries to use the token *T1* (i.e. in that case *T2* = *T1* which will be successful only in the case that the end user has access to the test dataset and the model). The proxy service has the option to deny access to the end user specifying some SSO policy for its URI while the validation service may have a different policy itself. This way it is possible to control which users can use the proxy service and/or the validation service directly. Furthermore service providers might consider hiding completely the actual validation service behind a firewall for additional security thus allowing it to be accessible only via the proxy.

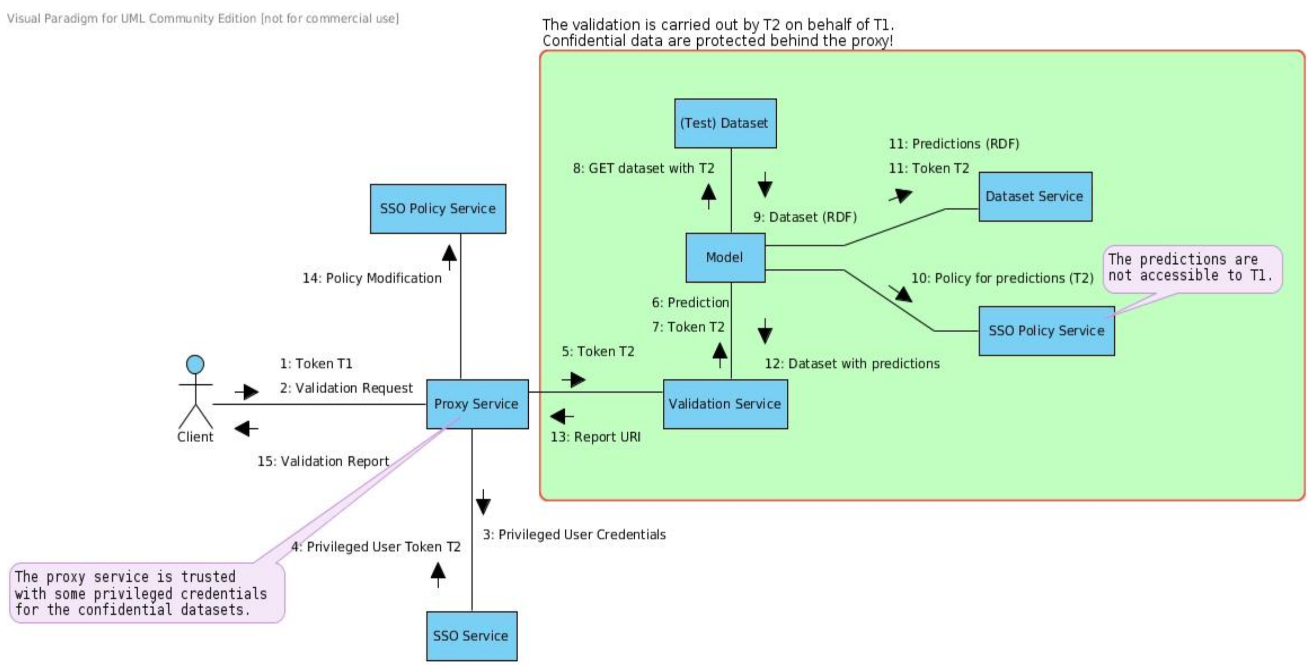

Visual Paradigm for UML Community Edition [not for commercial use]
The validation is carried out by T2 on behalf of T1.
Confidential data are protected behind the proxy!
(Test) Dataset
8: GET dataset with T2
9: Dataset (RDF)
11: Predictions (RDF)
11: Token T2
Dataset Service
SSO Policy Service
14: Policy Modification
Model
10: Policy for predictions (T2)
The predictions are not accessible to T1.
SSO Policy Service
6: Prediction
7: Token T2
5: Token T2
12: Dataset with predictions
1: Token T1
2: Validation Request
Client
Proxy Service
Validation Service
13: Report URI
15: Validation Report
3: Privileged User Credentials
4: Privileged User Token T2
The proxy service is trusted with some privileged credentials for the confidential datasets.
SSO Service

Figure 15: Privileges proxy acts as a broker of a portion of confidential results

Figure 16 shows that it is possible that the end user is denied access to the proxy service and in the presence of a firewall no validation is possible and the validation service is utterly protected from unwanted requests (even though it might not make much sense in denying access to a user to the validation service).

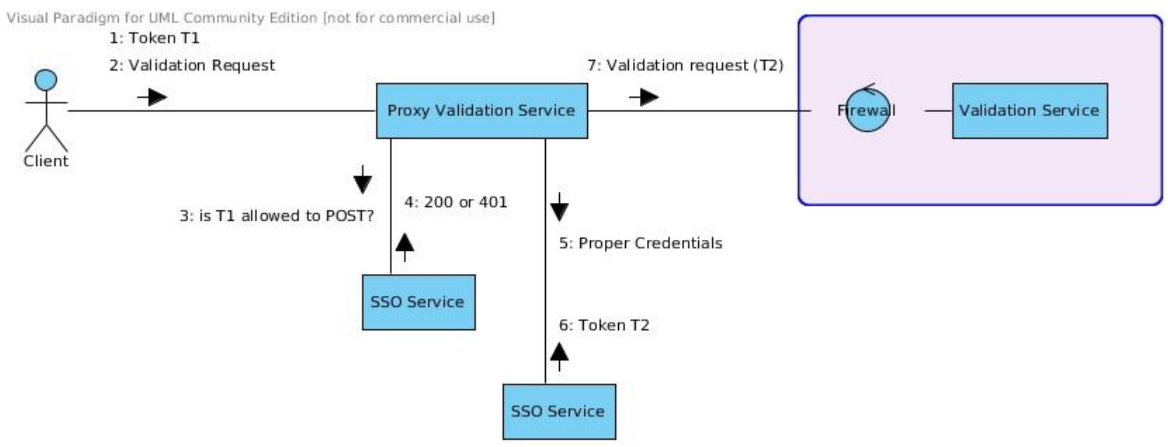


Figure 16: Firewalls enhance the protection of sensitive information from leakage

Finally one more alternative is that the validation service can address the proxy in case it receives a 401 from the test dataset resource acting on behalf of the user (*T1*). This scheme has its advantages as it can control the validation of data on two independent layers. Users address the actual validation service in all cases, which takes on the task of accessing the trusted proxy service only if a 401 is received while processing the request on behalf of *T1*. In case of an initial 401, the validation service attempts to find an authentication token *T2* and retry the request. In this case proxy service providers can consider hiding the proxy service behind a firewall.

6. Conclusions

This report summarizes the work that has been accomplished within the OpenTox Framework on the protection of confidential data and in particular on providing validation services against confidential data. Based on a description of the relevant use case, but also on technical details about the implementation strategy and several examples on particular datasets, the report illustrates that maximum protection has been accomplished using the OpenTox authorization and authentication services. This capability should have value in the context of the increasing acceptance of cloud computing approaches by industry. A top priority requirement by some end users for further reducing data leakage risk was a driving force for developing stand-alone versions of OpenTox applications. A challenging task for future implementations is to give end users the chance to test their models and algorithms against confidential data without having direct or indirect access to the actual endpoint values.