Integrate Your Interaction Data into DASMIweb and Other DASMI Clients

This tutorial will guide you through the process of integrating your own data into DASMIweb or other DASMI clietns in just a few steps.

Introduction

DASMIweb is based on the Distributed Annotation System (DAS, http://www.biodas.org/wiki/Main_Page) and therefore retrieves all its data from different distributed servers. These servers, also referred to as sources, exhibit a defined interface that DASMIweb uses to query for particular interactions and annotations. The corresponding data is returned in a specific XML format (for technical details see http://www.dasregistry.org/extension_interaction.jsp). A list of all publically available servers is maintained at the DAS registry (http://www.dasregistry.org). In addition, sources can be directly incorporated into DASMIweb without registering them at the DAS registry.
In a nutshell, the process of incorporating new data into DASMIweb consists of two steps: Setting up a new DASMI server that provides the interaction data and registering the server at the DAS registry or the DASMIweb client.

1 Setting up a new server

Providing your own interaction data should not require you to create complex software. Therefore, several server libraries have been developed that take care of serving your data over the DAS protocol. Dazzle, written in Java, and ProServer, written in Perl, are the two most versatile DAS server libraries. In this tutorial, we will focus on Dazzle and provide step-by-step installation and configuration instructions. A chapter focusing on ProServer will follow; in the meantime visit http://www.sanger.ac.uk/Software/analysis/proserver/ for further information.

1.1 Installing the Dazzle DAS server

Dazzle is a DAS server written in Java. It requires a Java Runtime Environment version 1.5 or later (http://java.sun.com/javase/downloads/index_jdk5.jsp) and an application server like Apache Tomcat (http://tomcat.apache.org/) or Caucho Resin (http://www.caucho.com/). We have tested Dazzle with Tomcat versions 4.1 and 5.5 and will use Tomcat 5.5 throughout this tutorial. Please note that we cannot describe the detailed installation procedure of Tomcat here, as it depends on the Tomcat version and your operating system and would go far beyond the scope of this tutorial (detailed instructions for each Tomcat version and operating system are available at http://tomcat.apache.org/). Once you have installed Tomcat, you will find a directory named webapps within the main Tomcat directory. Dazzle has to be placed inside this directory to make it available to your Tomcat installation.
You can find a zipped archive of the Dazzle directory on our website at http://www.dasmi.de/download.php. If you wish to have a look at the source code, you can acquire it at http://www.derkholm.net/thomas/dazzle/. In order to “install” Dazzle, download the archive, place it inside the webapps directory, and unzip it there. A

directory named `das` should be created automatically. You can delete the zipped archive thereafter.

1.2 Configuring Dazzle to serve your interaction data.

To provide a better understanding of the following steps, we first briefly describe the basic structure of Dazzle. Dazzle comprises a number of `DataSource` classes that instruct the software how the data it should serve is organized and stored. There is, for example, one `DataSource` that can access data that is stored in specific relational database tables, one that can access data in flat GFF files, etc.
We currently provide two data source classes for common interaction file formats, `SIFReferenceSource` and `PSIMIReferenceSource`, which can be used to provide interaction data from Simple Interaction Format (SIF) and PSI-MI XML2.5 files, respectively. If your data is present in one of these formats, you only have to tell your Dazzle installation where to find it via the Dazzle configuration file. If your data files have a different format, you may contact us at `http://dasmi.de/contact.php` as we might be able to create a new `DataSource` class for you without much effort. Alternatively, you can create this class on your own (see `http://www.derkholm.net/thomas/dazzle/` for the Dazzle source code).

1.2.1 The `dazzlecfg.xml` configuration file

The Dazzle configuration specifies all data sources that Dazzle provides. This configuration is contained in the XML file `dazzlecfg.xml` that is located in the root directory of Dazzle (e.g. `<TOMCAT-HOME>/webapps/das/dazzlecfg.xml`). Each data source is defined via a `datasource` XML element. Below is an example of such an entry:

```
<datasource id="test"
    jclass="org.biojava.servlets.dazzle.datasource.EmblDataSource">
  <string name="name" value="Test seqs" />
  <string name="description" value="Evalutation sequences for promoter-finding
      software" />
  <string name="version" value="1.0" />
  <string name="fileName" value="test.embl" />
  <string name="stylesheet" value="test.style" />
</datasource>
```

A `datasource` entry contains an `id` attribute that defines a unique id and a `jclass` attribute that specifies the associated `DataSource` Java class with its full path. Depending on the `DataSource` class used, a number of additional parameters have to be made via `string` elements, each in the form of a `name`/`value` pair.
The configuration details of the two `DataSource` classes available for serving interactions from SIF and PSI-MI XML2.5 files are described in the next two sections.

1.2.1.1 `SIFReferenceSource`

The Simple Interaction Format (SIF) is a basic file format to describe binary interaction data (the full specification is available at `http://www.cytoscape.org/cgi-bin/moin.cgi/Cytoscape_User_Manual/Network_Formats`). SIF files can be edited with an arbitrary text editor, are easily understandable, and can be readily imported into the network analyzing program Cytoscape (`http://www.cytoscape.org/`). Throughout this tutorial, we will use the exemplary SIF file `cltb.sif` with the following content:

cltb.sif:

```
1212 pp 3092
8218 pp 1212
1212 pp 1213
```

This file, stored in the same directory as the `dazzlecfg.xml` file, contains three binary protein interactions. The interactors are defined via Entrez Gene Identifiers. To serve these interactions with Dazzle, the following `datasource` entry has to be added to the `dazzlecfg.xml` configuration file:

```
<datasource id="cltb"
    jclass="org.biojava.servlets.dazzle.datasource.SIFReferenceSource" >
  <string name="name" value="CLTB" />
  <string name="description" value="My exemplary clathrin interactions" />
  <string name="version" value="default" />
  <string name="fileName" value=" C:/Tomcat/webapps/das/cltb.sif" />
  <string name="coordSys" value="Entrez,Gene_ID" />
</datasource>
```

We define `cltb` as the unique identifier for this `SIFReferenceSource` data source. The associated Java class has to be specified with its complete path, in this case `org.biojava.servlets.dazzle.datasource.SIFReferenceSource`. The attributes `name` and `description` are used to further characterize the interactions contained in the file, `fileName` contains the path to the SIF file, and `coordSys` the identifier system in which the interactions are specified. Other common identifier systems for describing interactions apart from `Entrez,Gene_ID` are `UniProt,Protein Sequence`, and `Pfam,Protein_Sequence`. See `http://www.dasregistry.org/help_coordsys.jsp` for a list of all currently available identifier systems. If your identifier system is not included in this list, you should contact us (`http://www.dasmi.de/contact.php/`).
SIF files are often accompanied by node and edge attribute files (file ending `.noa` for node attribute files, `.eda` for edge attribute files). As the name suggest, these files provide further information on interactors and interactions, such as names, symbols, and confidence scores (for detailed information on the structure of attribute files, see `http://www.cytoscape.org/cgi-bin/moin.cgi/Cytoscape_User_Manual/Attributes`). To further specify our three example protein-protein interactions, we add two node attribute files and one edge attribute file to the Dazzle root directory:

cltbSymbols.noa:

```
Symbol
1212 = CLTB
3092 = HIP1
1213 = CLTC
8218 = CLTCL1
```

cltbNames.noa:

```
FullName
1212 = clathrin, light chain (Lcb)
1213 = clathrin, heavy chain (Hc)
3092 = huntingtin interacting protein 1
8218 = clathrin, heavy chain-like 1
```

cltbConfidences.eda:

```
ConfidenceScore
1212 pp 3092 = 0.53
1213 pp 1212 = 0.99
```

The node attribute files `cltbSymbols.noa` and `cltbNames.noa` provide symbols and full names for the interactors, the edge attribute file `cltbConfidences` contains confidence scores for two interactions. Note that the first line of the files always contains the name of the attribute and will also be displayed within DASMIweb (Figure 7). To include the contents of these files into your Dazzle installation, you have to add the following line to the above `datasource` configuration:

```
<string name="attributeFiles" value=" C:/Tomcat/webapps/das/cltbNames.noa;
C:/Tomcat/webapps/das/cltbSymbols.noa;
C:/Tomcat/webapps/das/cltbConfidences.eda" />
```

Note that you can define as many attribute files as you want within one line, you only have to separate the paths to each file by semicolons.
Assuming that our Dazzle installation should only provide the SIF data source we defined in this example, the complete `dazzlecfg.xml` file would look like this:

```
<dazzle xmlns="http://www.biojava.org/2000/dazzle">
  <datasource id="cltb"
      jclass="org.biojava.servlets.dazzle.datasource.SIFReferenceSource" >
    <string name="name" value="CLTB" />
    <string name="description" value="My exemplary clathrin interactions" />
    <string name="version" value="default" />
    <string name="fileName" value="C:/Tomcat/webapps/das/cltb.sif" />
    <string name="coordSys" value="Entrez,Gene_ID" />
    <string name="attributeFiles" value=" C:/Tomcat/webapps/das/cltbNames.noa;
        C:/Tomcat/webapps/das/cltbSymbols.noa;
        C:/Tomcat/webapps/das/cltbConfidences.eda" />
  </datasource>
</dazzle>
```

1.2.2.2 `PSIMIReferenceSource`

PSI-MI XML2.5 is a widely used file format for the representation of molecular interactions (see `http://www.psidev.info/index.php?q=node/60` for a description of the format). The `PSIMIReferenceSource` enables Dazzle to serve interactions from PSI-MI files. PSI-MI XML2.5 files are rather complex and tool support has long been limited. Therefore, the `PSIMIReferenceSource` is still in development and currently only a minimal set of information is extracted from the PSI-MI file and made available by Dazzle. An exemplary `datasource` entry for the `PSIMIReferenceSource` is listed below:

```
<datasource id="17220478"
    jclass="org.biojava.servlets.dazzle.datasource.PSIMIReferenceSource">
  <string name="name" value="17220478" />
  <string name="description" value="Protein interaction from the experiment
      reported in the Pubmed entry 17220478." />
  <string name="version" value="default" />
  <string name="fileName" value=" C:/Tomcat/webapps/das/17220478.xml" />
  <string name="baseDir" value="C:/Tomcat/webapps/das/" />
</datasource>
```

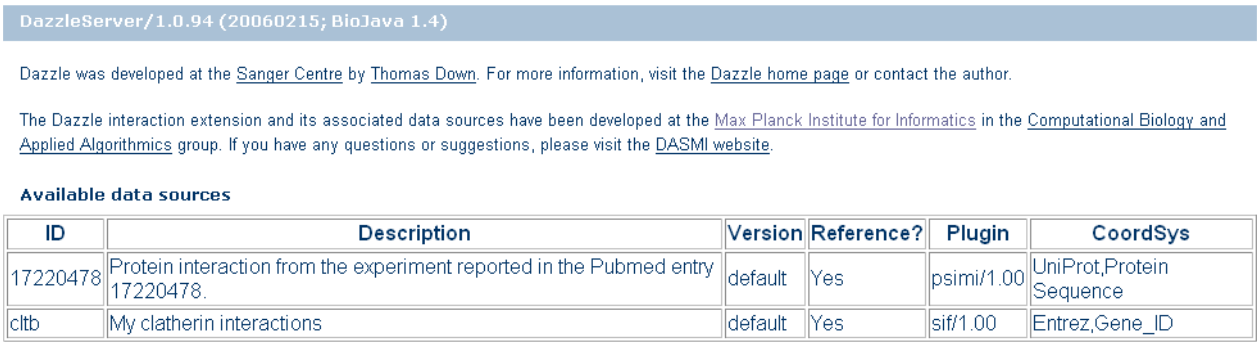
This entry enables Dazzle to serve all interactions from the PSI-MI XML2.5 file `17220478.xml` (downloaded from the IntAct database at `http://www.ebi.ac.uk/intact/`). The important difference to the `SIFReferenceSource` entry listed above is the different `jclass` attribute, now pointing to the Java class with the path `org.biojava.servlets.dazzle.datasource.PSIMIReferenceSource`. The `PSIMIReferenceSource` temporarily stores all interactions in an embedded Apache Derby database. The attribute `baseDir` specifies the location where this embedded databases is stored. Parsing the XML and storing all interactions in the embedded database can demand some time, depending on the size of the PSI-MI XML2.5 file. As already mentioned, this Java class is still in development and improvements to configurability and speed will be made. If you have any particular request, please contact us at `http://www.dasmi.de/contact.php/`.

1.3 Testing your Dazzle DAS server

Once you have configured all data sources in your `dazzlecfg.xml` file, you may start Tomcat. If you have not specified it otherwise during the installation process, Tomcat will be available at `http://localhost:8080/` and should present you a welcome page. If this does not happen, please check the console for obvious errors.

As we unzipped the Dazzle archive into the subdirectory `das` within Tomcat `webapps` directory, Dazzle will be available at the URL `http://localhost:8080/das/`. (It is a convention to choose the directory name `das`, you may, however, choose a different name.) Note that the trailing slash is mandatory. Especially when used in conjunction with the Apache web server, a missing slash will result in an “URL not found error”.

If all configurations went fine, you should see a Dazzle welcome page with an overview of all currently available sources like in Figure 1. If not, please check your Tomcat logs and the console for obvious errors like reports of malformed XML elements. If this does not help, you may contact us at `http://dasmi.de/contact.php`. Please note that all data sources are initialized upon the first Dazzle request. If you have configured a `PSIMIReferenceSource` associated with a large XML file, this startup process can take several minutes.


DazzleServer/1.0.94 (20060215; BioJava 1.4)

Dazzle was developed at the Sanger Centre by Thomas Down. For more information, visit the Dazzle home page or contact the author.

The Dazzle interaction extension and its associated data sources have been developed at the Max Planck Institute for Informatics in the Computational Biology and Applied Algorithmics group. If you have any questions or suggestions, please visit the DASMI website.

Available data sources

ID	Description	Version	Reference?	Plugin	CoordSys
17220478	Protein interaction from the experiment reported in the Pubmed entry 17220478.	default	Yes	psimi/1.00	UniProt,Protein Sequence
cltb	My clatherin interactions	default	Yes	sif/1.00	Entrez,Gene_ID

Figure 1. Welcome page of the Dazzle DAS server providing two exemplary data sources.

To test, for example, the cltb data source for interactions, attach the query string `cltb/interaction?interactor=1212` to the Dazzle URL, resulting in the complete URL `http://localhost:8080/das/cltb/interaction?interactor=1212`, and hit enter. If your

browser presents you an XML output like in Figure 2, your interaction DAS server is ready to be used within DASMIweb.

```
- <DASINT xsi:schemaLocation="http://dasmi.de/dasint.xsd">
  - <INTERACTOR id="1212" shortLabel="CLTB" dbSource="" dbVersion="" dbAccessionId="1212" dbCoordSys="Entrez,Gene_ID">
      <DETAIL property="FullName" value="clathrin, light chain (Lcb)"/>
    </INTERACTOR>
  - <INTERACTOR id="8218" shortLabel="CLTCL1" dbSource="" dbVersion="" dbAccessionId="8218" dbCoordSys="Entrez,Gene_ID">
      <DETAIL property="FullName" value="clathrin, heavy chain-like 1"/>
    </INTERACTOR>
  - <INTERACTOR id="1213" shortLabel="CLTC" dbSource="" dbVersion="" dbAccessionId="1213" dbCoordSys="Entrez,Gene_ID">
      <DETAIL property="FullName" value="clathrin, heavy chain (Hc)"/>
    </INTERACTOR>
  - <INTERACTOR id="3092" shortLabel="HIP1" dbSource="" dbVersion="" dbAccessionId="3092" dbCoordSys="Entrez,Gene_ID">
      <DETAIL property="FullName" value="huntingtin interacting protein 1"/>
    </INTERACTOR>
  - <INTERACTION name="1212-3092" dbSource="" dbSourceCvId="" dbVersion="" dbAccessionId="">
      <DETAIL property="BPscore" value="0.53"/>
      <PARTICIPANT id="1212"/>
      <PARTICIPANT id="3092"/>
    </INTERACTION>
  - <INTERACTION name="1212-8218" dbSource="" dbSourceCvId="" dbVersion="" dbAccessionId="">
      <PARTICIPANT id="8218"/>
      <PARTICIPANT id="1212"/>
    </INTERACTION>
  - <INTERACTION name="1212-1213" dbSource="" dbSourceCvId="" dbVersion="" dbAccessionId="">
      <DETAIL property="BPscore" value="0.99"/>
      <PARTICIPANT id="1212"/>
      <PARTICIPANT id="1213"/>
    </INTERACTION>
</DASINT>
```

Figure 2. Dazzle's XML response to the example query `http://localhost:8080/das/cltb/interaction?interactor=1212`.

Before you can register your server, you have to determine its public URL. If your server runs on your computer, you can use services like `http://www.whatsmyip.org/`. If you replace the "localhost" in our prior example with your IP, you should still be able to see the same output as in Figure 2. Ff you are within a larger institute you will most likely not be able to provide servers from your local computer to the public due to security restrictions. But in those cases it is rather likely that you have an administrator who will help you with all the Tomcat related work and eventually provide you a valid URL of your server.

2 Registering your server

Depending on whether your data should be publically available to everybody or not, you have two options for the server registration:

- In case the data should be **publically available** to everyone who uses DASMIweb or other DASMI clients, you have to register it at the DAS registry (see 2.1).
- If your data should not be **available** to everyone who uses DASMIweb or other DASMI client, but **only to you**, you have to register it locally at DASMIweb (see 2.2).

Please note that the registration at the DAS registry is permanent (at least until you remove the source from the registry), while the registration within DASMIweb has to be renewed each time you visit DASMIweb after a longer period.

2.1 Registering your server at the DAS registry

The DAS registry (`http://www.dasregistry.org/`) is the central repository of DAS servers. That means that once you registered your server at the DAS registry it will be available to all DASMI clients und users. The DAS registry is well documented, thus we will not explain every detail of the registration process here.

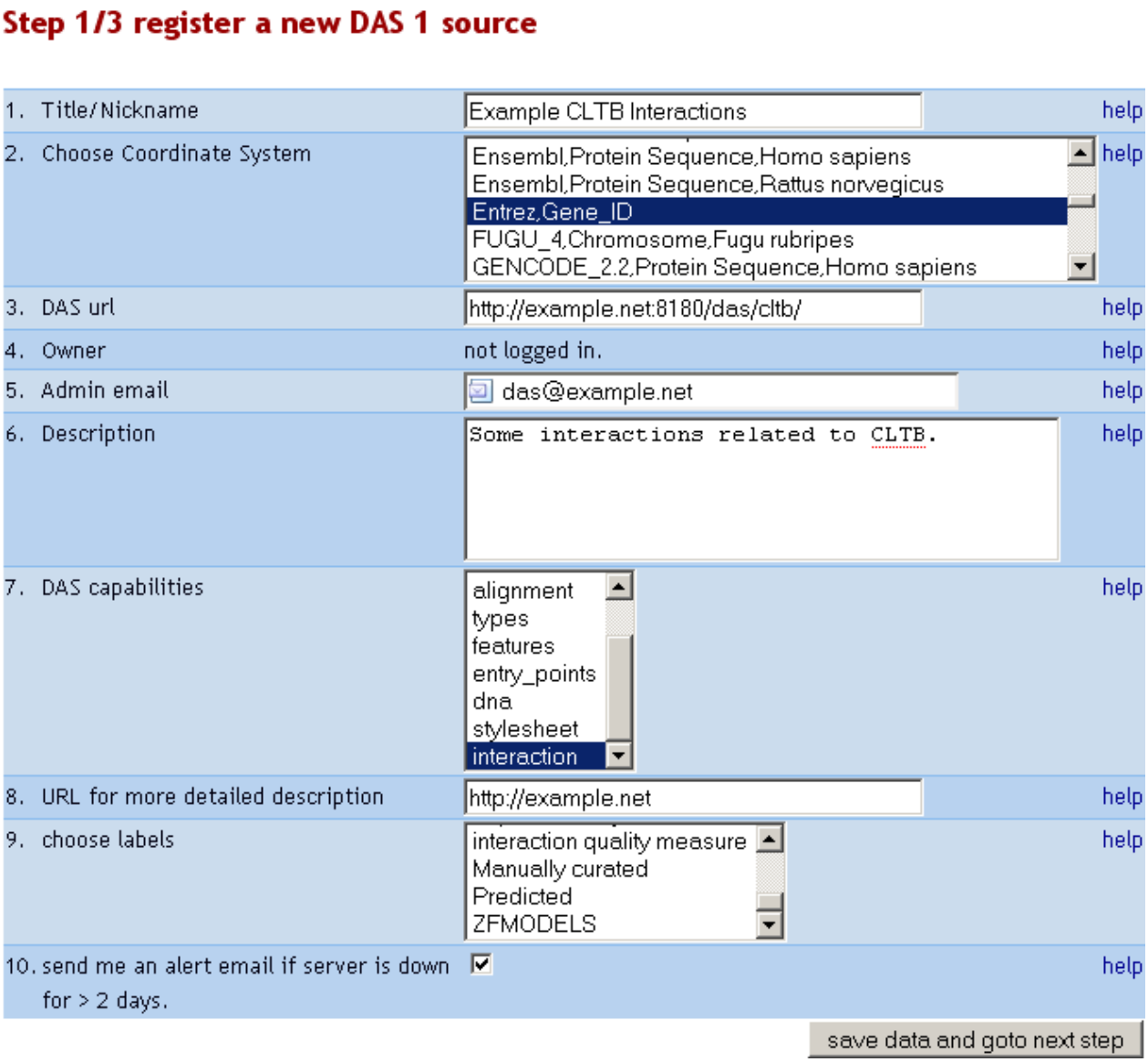


Figure 3. Step one of the registration process at the DAS registry. Note that these are just example values that have to be replaced with your correct values. If you choose the label “interaction quality measure” in point 9, your server will be used as a quality measure by DASMIweb (see 2.3 for more details).

In the DAS registry menu select "register new" or navigate to `http://www.dasregistry.org/registerService.jsp`. On the following page you can choose if you want to login with an OpenID account (recommended, see `http://en.wikipedia.org/wiki/OpenID` for more details) or register the source without logging in. Figure 3 shows your example values for the example data source that we defined and tested above.
The DAS registry has a built-in testing functionality for DAS servers. This is very useful, as it will notify you via email if your server has been unavailable for more than two days. In the next step you are therefore asked to provide a test range (Figure 4). We use the same identifier as we used for testing in Figure 2.

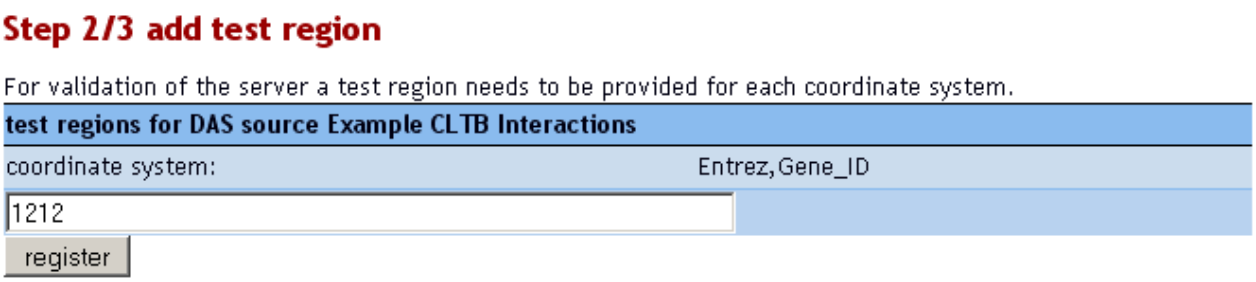


Figure 4. Step two of the registration process at the DAS registry.

If the DAS registry is able to retrieve a valid response from the server with the values we entered in the prior steps, it will register the source and provide you a key that you will need to delete the source, as shown in Figure 5. Upon its successful registration, the source will be available within DASMIweb and all other DASMI clients without further configuration efforts (Figure 7 (a)).

Figure 5. Step three of the registration process at the DAS registry. The key has been made unreadable, even though the example source has been immediately deleted after its registration.

2.2 Registering your server at DASMIweb

In case you want your data not to by seen be everyone who uses DASMIweb or other DASMI clients, you have to register it at DASMIweb instead of the DAS registry. Please note that the server that you have set up in step 1 is still publically accessible to everyone knowing the URL. But if you do not register the server at the DAS registry, nobody apart from you will know of its existence. However, we are working on further

increasing data protection by implementing an IP filter that will only allow certain “users” like DASMIweb to access the data from your server.
To add a new source, navigate in the DASMIweb client (`http://www.dasmi.de`) to the Configuration Panel by clicking on the “Show Source Configuration Button” in the top left Query Panel. If you have already performed a query within your current session, the Configuration Panel will initially show you all sources with the same identifier system. In this case, you have to click the “Add New Source” tab in the top of the Configuration Panel. Once you entered all your data, your Configuration Panel should look like Figure 6.

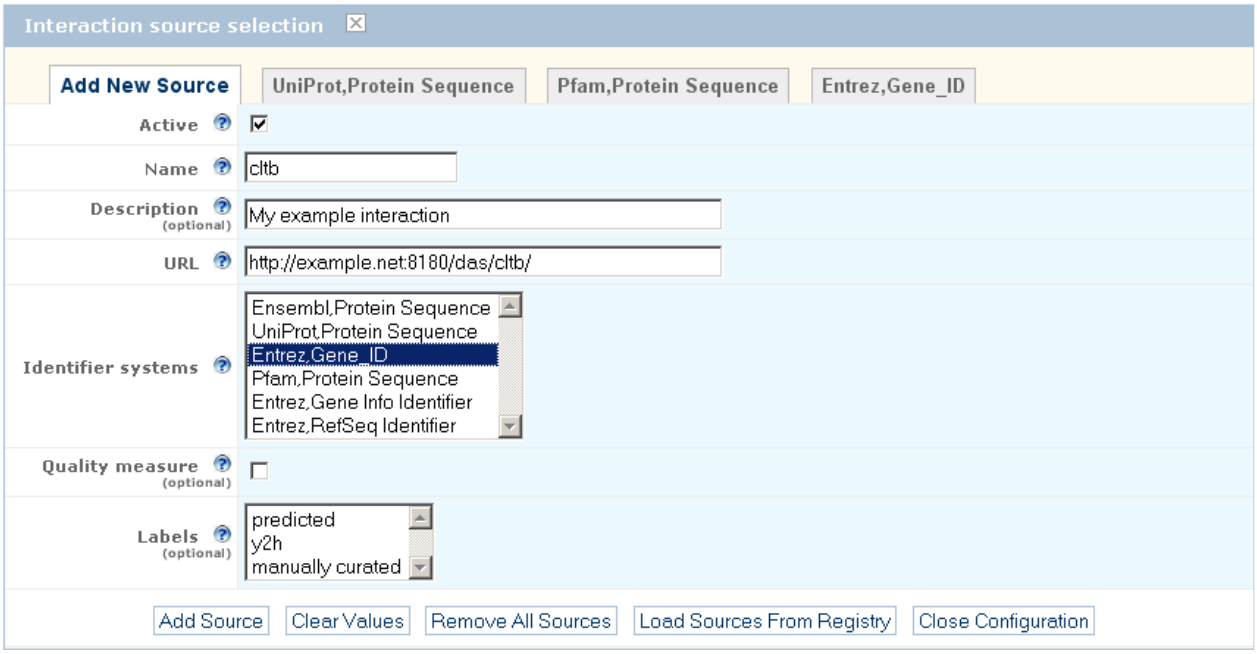


Figure 6. Registration of a server at DASMIweb. Please notr that the URL in this example is wrong and has to be replaced with the correct URL of your server. If the field right beside of “Quality measure” would be checked the server would be used as a quality measure instead of a “normal” interaction server (see 2.3 for more details).

After you click the “Add Source” button, you server will be registered and will be included in all future queries (Figure 7(a)).

2.3 Register your server as a quality scoring server

In our prior SIF data source example (see 1.2.1.1) we included an edge attribute file that contained two confidence scores. We might therefore not want to include our server to provide novel interaction data, but to score the confidence of other interaction servers. For example, DASMIweb presents the results of “normal” interaction servers as squares in an interaction table (Figure 7(a)). In contrast, the results of servers that provide quality measures are not presented as new squares, but are used to color existing squares (Figure 7(b)).
In order to register your server as a quality measure, you only have to make minimal changes to the registration processes described in 2.1 and 2.2.

- If you want to register your interaction quality server at the DAS registry, you have to select the label “interaction quality measure” (point 9 in Figure 3).
- If you want to register your scoring server at DASMIweb, you have to hit the checkmark at the option “Quality measure” (Figure 6).

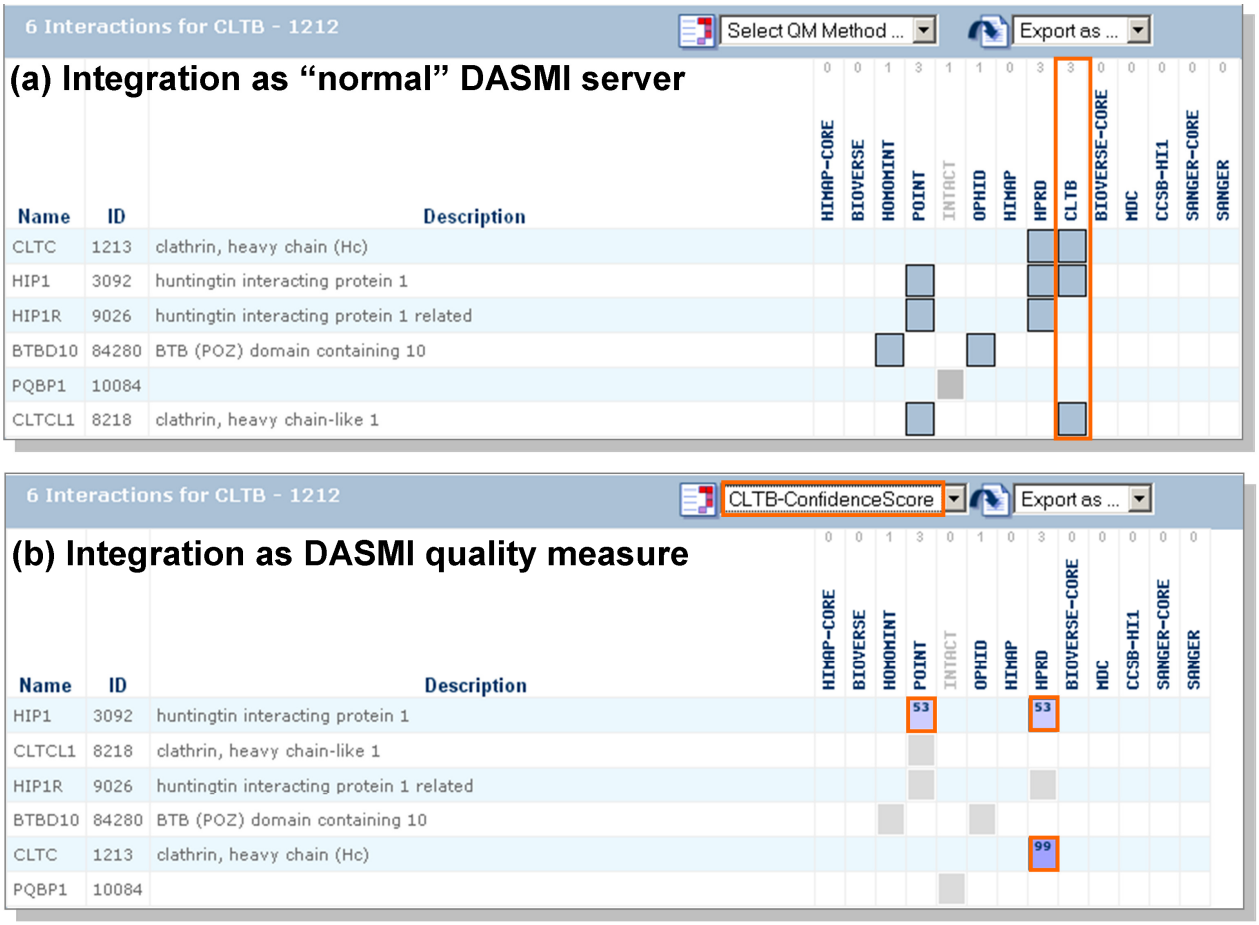


Figure 7. Integration of a DASMI server as normal server and quality measure. **(a)** If the DASMI server has been registered without the label “interaction quality measures” (Figure 3) or the respective “Quality measure” checkmark (Figure 6), its results will be present in the usual fashion as squares. **(b)** In case the server has been registered as a quality measure, its results will not be visualized by new squares, but can be used to color existing squares according to the scores.