

TDWG Life Sciences Identifiers Authority Setup Guide

Using the LSID Java Software Library

Date:

19 November 2007

Author:

Kevin Richards (Landcare Research, New Zealand)

Task Group:

TDWG Globally Unique Identifiers Task Group (GUID)

<http://www.tdwg.org/activities/guid/>

Abstract:

This document provides guidance on how to set up an LSID authority using the **LSID Java software library**. Readers must first follow the instructions on the companion document titled "[TDWG LSID Authority Setup Guide – Programming Language Independent Steps](#)" for guidance on the part of the process that is independent of the programming language used.

Status:

Accompanying (type 3) documentation for the [TDWG LSID Applicability Statement](#).



Copyright © Biodiversity Information Standards - TDWG (2007). Some Rights Reserved.
This work is licensed under a [Creative Commons Attribution United States 3.0 License](#).

To view a copy of this license, visit <http://creativecommons.org/licenses/by/3.0/us/> or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.

Disclaimer:

This document and the information contained herein are provided on an "AS IS" basis. TDWG MAKES NO WARRANTIES REGARDING THE INFORMATION PROVIDED, AND DISCLAIMS LIABILITY FOR DAMAGES RESULTING FROM ITS USE.

Table of Contents

Introduction.....	4
Preparation – Platform Independent Steps	4
Setup Outline.....	5
1. Install Java JRE & JDK	6
2. Install Eclipse development environment.....	6
3. Install Apache Tomcat web server	6
4. Download and install Java LSID toolkit	6
5. Create Java code project and classes	7
6. Configure the LSID service	8
7. Run the authority.....	8
8. Set up DNS	8
9. Finishing up and testing the Authority	8
Appendix A - Mapping Between LSID Parts and Java Code	9
Appendix B – Java code for the authority.....	9
Appendix C – The settings xml configuration file for the authority	11
Appendix D – Example xml responses from the authority.....	12

Introduction

This document provides guidance on how to set up an LSID authority using the **LSID Java software library**. Readers must first follow the instructions on the companion document titled "[TDWG LSID Authority Setup Guide – Programming Language Independent Steps](#)" which provides guidance for the part of the process of setting up an LSID authority that is independent of programming language used.

The instructions in this guide refer to version 1.1.4 of the LSID Perl software library, but should apply to previous versions of the libraries as well.

The environment used for this guide includes the Java Runtime Engine (JRE) 6.0, Eclipse Java development environment 3.3, Apache Tomcat 6.0, running on a Microsoft Windows Server, but could be followed for similar environments.

Preparation – Programming Language Independent Steps

Before setting up your LSID authority using the LSID Perl software library, make sure you have decided-

1. What categories of objects you will assign LSIDs to.
2. What **Namespace Identifications** you will use for each category listed above.
3. How to form the **Object Identification** for objects in each namespace.
4. Whether to use and how to form the optional **Revision Identification** for the objects in each namespace.
5. What **data** and **metadata** will be associated with each LSID.
6. What RDF representation will be used to express LSID metadata
7. What your LSID **Authority Identification** will be.

Also make sure that you have the **DNS SRV record** corresponding to your LSID authority identification set up and fully propagated.

For more information on how to make the decisions above or to set up DNS for work with your LSID authority, please see the [TDWG LSID Authority Setup Guide – Programming Language Independent Steps](#). If you are setting up an LSID authority for biodiversity information, please follow requirements and recommendations from the [TDWG LSID Applicability Statement](#) as well.

Setup Outline

The process of setting up an LSID Authority using the LSID Java software libraries is as follows:

1. Install Java JRE & JDK, if not already installed
2. Install Eclipse development environment
3. Install Apache Tomcat web server
4. Download and install the Java LSID toolkit
5. Create Java code project and classes
6. Configure the LSID service
7. Run the authority
8. Set up DNS
9. Finishing up and testing the Authority

1. Install Java JRE & JDK

Download/install online JRE (5 used in this example) from
<http://www.java.com/en/download/index.jsp>

Create an environment variable JRE_HOME to point to the JRE install directory, e.g. C:\Program Files\Java\jre1.6.0_03. This can be done by going to My Computer (on the desktop), right click, -> Properties -> Advanced -> Environment Variables.

2. Install Eclipse development environment

Download Eclipse (version 3.3 used in this example) from-
<http://www.eclipse.org/downloads/>

Unzip into a directory of choice, e.g. c:\eclipse

3. Install Apache Tomcat web server

Download Apache Tomcat (version 6 used in this example) from
<http://tomcat.apache.org/download-60.cgi>

Unzip into a directory of choice, e.g. c:\tomcat

4. Download and install Java LSID toolkit

Download LSID java toolkit from Source Forge -
http://sourceforge.net/project/showfiles.php?group_id=130827&package_id=144604

Unzip into a directory of choice, e.g. c:\lsid-toolkit

Copy the /com.ibm.lsid.server/authority directory from the toolkit to the webapps directory under Tomcat, eg c:\tomcat\webapps\authority

Download the required jar files for the Authority from
<http://lsids.svn.sourceforge.net/viewvc/lsids/trunk/lsid-java/com.ibm.lsid.client/lib/>

Put into a directory of choice, eg c:\lsid\jars

Copy the jar files to the authority lib directory in the Tomcat webapps directory, e.g.:
c:\tomcat\webapps\authority\WEB-INF\lib

5. Create Java code project and classes

Java project in eclipse:

1. Start eclipse – run eclipse.exe in the Eclipse directory.
2. Select New > Project...
3. Select the Java > Java Project type project. Select Next
4. Type in the name of the project, eg ExAuthority. Select Next.
5. Select Libraries tab, the Add External Jars and browse to the folder where the LSID authority Jars were downloaded to
6. Add all the Jar files
7. Select Finish.

Create Java class:

1. In Eclipse, right click the src folder of the java project, in the Package Explorer pane. Select New > Class
2. Type in the name of the class, eg ExAuthority
3. Browse to the class to inherit from (superclass) - type in SimpleResolutionService (it should appear automatically)
4. Select Finish
5. Open the class file (eg ExAuthority.java, in the Package Explorer pane)
6. There are several functions that need implementing, depending on what you would like your authority to do:
 - a. doGetMatadata – returns a metadata response for metadata calls to your authority. This will be implemented in the majority of cases
 - b. getServiceName – always implemented (simply returns the name of your authority service)
 - c. hasData – implemented if you want to return data from your authority
 - d. hasMetadata – implemented if you want to return metadata from your authority (most cases). Returns true if the requested LSID has metadata in your database
 - e. validate – implemented if you need to validate or check any details about particular requests
 - f. initService – implemented if you need to do anything on start-up of your service/authority
 - g. getData – implemented if you want to return data from your authority

The code that was written for this example is given in appendix B.

Build Authority:

1. Implement the required functions for your authority (most typical cases will include getServiceName, doGetMetadata and hasMetadata)
2. Build happens automatically (if eclipse is setup that way – this is default)
3. Copy the built class file to authority classes directory, eg
c:\tomcat\webapps\authority\WEB-INF\classes
The class file will be built to the Eclipse workspace directory – something like c:\Documents and Settings\[user]\workspace\ExAuthority\bin

6. Configure the LSID service

Create a copy of the default-services.xml that is located in the webapps\authority\services directory. Eg exauthority-services.xml.

Edit the services to reflect your own map and service elements. Put the details of your class within these elements.

Remove any other services.xml files in this directory.

For more details of the services xml file, see appendix C.

7. Run the authority

1. Start Tomcat web server (/bin/startup.bat of the Tomcat directory)
2. Test the authority by entering <http://localhost:8080/authority> into a browser. A WSDL (xml) response should be returned.
3. Test an LSID call, eg
<http://localhost:8080/authority/metadata/?lsid=urn:lsid:exauthority.org:abc:123>
The **urn:lsid:exauthority.org:abc:123** part will depend on the LSID format you have setup.

Example response xml is given in appendix D.

Tip : If you make changes to the java class and copy it back to the webapps/classes directory you will need to restart Tomcat. You may also need to clear the Tomcat cache so you get the correct responses (the easiest way to do this is to delete the contents of the Tomcat/work/Catalina directory).

8. Set up DNS

To have your LSIDs resolve automatically over the Internet, you will need to set up a DNS record to point to your resolver server. For more details about this part of LSID setup, see the associated document [TDWG LSID Authority Setup Guide – Programming Language Independent Steps](#).

9. Finishing up and testing the Authority

The document [TDWG LSID Authority Setup Guide – Programming Language Independent Steps](#) provides instructions to ensure the authority is performing correctly, including:

1. Testing the LSID Authority
2. Advertising the LSID Authority
3. Tagging objects in your information systems with LSIDs.

Appendix A - Mapping Between LSID Parts and Java Code

Each part of your LSIDs is mapped into the Java code as follows:

Network Identifier (NID) – the “urn:lsid” prefix; it is fixed and unchanged for all LSIDs.

Authority Identification – defined by the Map element in the settings.xml file, eg:

```
<map name="ex">
    <pattern auth="exauthority.org" ns="*" />
</map>
```

Namespace Identifications – also defined in the settings.xml file as shown above (the ns attribute of the pattern element).

Object Identification – defined by the parameter used in a lookup, eg a SQL query parameter, a filename, etc, for a particular object.

Revision Identification –the revision is ignored in this example, but the revision could be mapped and processed separately for more complex databases.

Appendix B – Java code for the authority

The following code is a simple example of a Java LSID authority that loads an RDF response from the associated file for a given LSID. A similar authority could be created that accesses a databases via SQL statements that pass the requested LSID as a parameter.

```
public class ExAuthority extends SimpleResolutionService
{
    @Override
    public MetadataResponse doGetMetadata(LSIDRequestContext arg0, String[]
arg1)
        throws LSIDServerException
    {
        //return the metadata for the requested LSID
        // Get the RDF metadata from the file with the associated name -
        [LSID object].rdf

        InputStream str = null;
        try
        {
            InputStream inp = new FileInputStream(arg0.getRealPath() +
"\\files\\" + arg0.getLsid().getObject() + ".rdf");
            int len = inp.available();
            byte[] data = new byte[len];
            inp.read(data);

            str = new ByteArrayInputStream(data);
        }
        catch(Exception ex)
        {
```

```

        str = new StringBufferInputStream(ex.getMessage());
    }

    MetadataResponse resp = new MetadataResponse(str,
java.lang.System.currentTimeMillis() + 1000000, "text/xml");

    return resp;
}

@Override
protected String getServiceName()
{
    // return the name of the service
    return "ExAuthorityService";
}

@Override
protected boolean hasMetadata(LSIDRequestContext arg0)
{
    //Return true if this service can process the given LSID
    // and there is metadata for that LSID
    //In this simple example we are just returning true
    // if the file exists
    String fName = arg0.getRealPath() + "\\files\\" +
arg0.getLsid().getObject() + ".rdf";

    if (new File(fName).exists())
    {
        return true;
    }
    return false;
}
}

```

Appendix C – The settings xml configuration file for the authority

The following xml defines the resolution services that are available at the associated authority.

This is a simple example. More settings are available for advanced configuration – see the documentation that comes with the Java LSID code (docs/html/server.html#configimpl).

The main elements are:

Map - this defines the mapping between an LSID pattern and a particular service. This is done by specifying the authority and namespace parts of the LSIDs that can be handled by the service (* = any namespace). Eg the map below handles all LSIDs for the LSID authority name “exauthority.org”, any namespace, eg urn:lsid:exauthority.org:abc:123, urn:lsid:exauthority.org:example:001

Service – this defines the service name and LSID functionality that is implemented by this service. The functionality is defined in the components elements of the service.

Components – Contains the functionality elements for the service. The elements shown are “auth” (main authority function) and “meta” (metadata function – i.e. this service will return metadata for a given LSID). Other functionality is available (data, authentication and assigning).

```
<deployment-descriptor xmlns="http://www.ibm.com/LSID/Standard/rsdl">
  <maps>
    <map name="ex">
      <pattern auth="exauthority.org" ns="*" />
    </map>
  </maps>
  <services>
    <service name="ExAuthorityService" >
      <components>
        <auth map="ex" type="class">ExAuthority</auth>
        <meta map="ex" type="class">ExAuthority</meta>
      </components>
    </service>
  </services>
</deployment-descriptor>
```

Appendix D – Example xml responses from the authority

Standard authority response (for a url like <http://exauthority.org:8080/authority/>)

```
<?xml version="1.0" encoding="UTF-8" ?>
<wsdl:definitions
  targetNamespace="http://www.omg.org/LSID/2003/Standard/WSDL"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:asb="http://www.omg.org/LSID/2003/AuthorityServiceSOAPBindings"
  xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
  xmlns:tns="http://www.omg.org/LSID/2003/Standard/WSDL"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:ahb="http://www.omg.org/LSID/2003/AuthorityServiceHTTPBindings">
  <wsdl:import
    namespace="http://www.omg.org/LSID/2003/AuthorityServiceHTTPBindings"
    location="LSIDAuthorityServiceHTTPBindings.wsdl" />
    <wsdl:import
      namespace="http://www.omg.org/LSID/2003/AuthorityServiceSOAPBindings"
      location="LSIDAuthorityServiceSOAPBindings.wsdl" />

      <wsdl:service name="AuthorityServiceSOAP">
        <wsdl:port name="SOAPPort" binding="asb:LSIDAuthoritySOAPBinding">
          <soap:address location="http://exauthority.org:8080/authority/" />
        </wsdl:port>
      </wsdl:service>

      <wsdl:service name="AuthorityServiceHTTP">
        <wsdl:port name="HTTPPort" binding="ahb:LSIDAuthorityHTTPBinding">
          <http:address location="http:// exauthority.org:8080" />
        </wsdl:port>
      </wsdl:service>

</wsdl:definitions>
```

Example RDF metadata response. This metadata is for a taxonomic name using the TDWG LSID vocabularies (see <http://wiki.tdwg.org/twiki/bin/view/TAG/LsidVocs> for more details).

```
<?xml version="1.0" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:TaxonName="http://rs.tdwg.org/ontology/voc/TaxonName#"
  xmlns:ns="http://purl.org/dc/elements/1.1/"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:Common="http://rs.tdwg.org/ontology/voc/Common#">

  <TaxonName:TaxonName rdf:about="urn:lsid:exauthority.org:a:123">

    <ns:Title>Amanita abrupta Peck1897</ns:Title>
    <owl:versionInfo>1.1.2.1</owl:versionInfo>
    <TaxonName:nameComplete>Amanita abrupta</TaxonName:nameComplete>
    <TaxonName:genusPart>Amanita</TaxonName:genusPart>
    <TaxonName:specificEpithet>abrupta</TaxonName:specificEpithet>
    <TaxonName:authorship>Peck</TaxonName:authorship>
    <TaxonName:basionymAuthorship>Peck</TaxonName:basionymAuthorship>
    <TaxonName:year>1897</TaxonName:year>
    <Common:publishedIn>(1897) (1897)</Common:publishedIn>
    <TaxonName:nomenclaturalCode
      rdf:resource="http://rs.tdwg.org/ontology/voc/TaxonName#ICBN" />
    </TaxonName:TaxonName>

  </rdf:RDF>
```