

Wikiprint Book

Title: tap

Subject: TracMeteo - tap

Version: 19

Date: 11/28/2021 12:29:18 PM

Table of Contents

What is TAP	3
Environment setup	3
Derby setup	3
Start derby	3
Tomcat setup	3
Expose our datasource in Tomcat	3
Set the realm	3
Expose the datasource as JNDI resource	4
Start Tomcat	4
Initial TAP setup	4
Thredds and TAP deployment	4
First steps in TAP	4
System users	5
System groups	5
System roles	5
System policies	5
System messages	5
Thredds restricted dataset	6
Restrict dataset access	6
Allow access to a restricted dataset from TAP	6
Hidden modes selection	9
Issues	9

What is TAP

The aim of the *THREDDS Access Panel (TAP)* project lies in the idea of solving the huge problem existing in the *Unidata THREDDS* application with the management of users, roles and dataset access. The Thredds user authentication is delegated to Tomcat Basic which gets users and roles from a given Realm. In order to authorize users, THREDDS checks whether the dataset is restricted and if so, gets the user roles to make a decision. **TAP** manages the datasource mentioned to control user access to protected datasets.

Environment setup

TAP and *THREDDS* get users and roles from a database created specifically to model all the entities and processes involved. The main idea is to include easily a database instance to work with. Derby network allow users to access db instances even from outside when they are running and most people are familiar with it. To accomplish this step you need four main components:

- Derby 10.10.1.1 library
- Derby database
- Apache Tomcat 7.0.59
- jre 1.7.0_75
- TAP Web Application: [tap.war](#)

If you want to build it by yourself, you can [checkout the web application source](#) and follow these steps:

1. Create a deployment folder (eg. deployment_test)
1. Create an appz folder in deployment_test to place libraries.
1. Extract *Derby 10.10.1.1* library in /deployment_test/appz
1. Extract *Tomcat 7.0.59* or paste the customized *Tomcat* provided.
1. Place the Derby DB folder (derbydb) in \$CATALINA_HOME/content/data

Derby setup

To start *Derby* successfully add a socket permission in *JAVA 7* by including in *\$JRE_HOME/lib/security/java.policy* the following line:

```
permission java.net.SocketPermission "HOST:DERBY_PORT", "listen,resolve";
```

Start derby

```
$JRE_HOME/bin/java -jar $PATH_TO_DERBY_LIB/db-derby-10.10.1.1-bin/lib/derbyrun.jar server start -p DERBY_PORT -h 0.0.0.0 &
```

Tomcat setup

Although attached is a customized tomcat ready to run, you can use another and configure it following these steps

Expose our datasource in Tomcat

Tomcat gets users and roles from conf/tomcat-users.xml by default. We are going to change this in order to get users and roles from a given database. First of all, include both derbyclient.jar and derbynet.jar in \$CATALINA_HOME/lib. After doing that, we need to add a new resource called ?jdbc/admin? in [GlobalNamingResources?](#):

```
<Resource name="jdbc/adminDB" auth="Container" type="javax.sql.DataSource"
  factory="org.apache.tomcat.dbcp.dbcp.BasicDataSourceFactory"
  validationQuery="SELECT count(*) FROM users" maxActive="20" maxIdle="10" username="admin" password="adm!n"
  driverClassName="org.apache.derby.jdbc.ClientDriver"
  url="jdbc:derby://YOUR_HOST:YOUR_PORT/./content/data/derbydb" readOnly="false"/>
```

Set the realm

A Realm is a ?database? of usernames and passwords that identify valid users of a web application (or set of web applications), plus an enumeration of the list of roles associated with each valid user. The servlet container will be connected to the database and it also be aware of the username and the corresponding roles. Define this realm inside <Engine> in your server.xml:

```
<Realm className="org.apache.catalina.realm.DataSourceRealm" digest="MD5" debug="0" dataSourceName="jdbc/adminDB"
  userTable="USERS" userNameCol="USERNAME" userCredCol="PASSWORD" userRoleTable="V_USERS_ROLES" roleNameCol="ROLENAME"/>
```

Expose the datasource as JNDI resource

TAP gets the datasource from the JNDI resource "jdbc/adminDB". Add to \$CATALINA_HOME/conf/context.xml the following line:

```
<ResourceLink global="jdbc/adminDB" name="jdbc/adminDB" type="javax.sql.DataSource"/>
```

Start Tomcat

We strongly recommend you before run Tomcat, set JAVA_OPTS="-Xms256m -Xmx4096m -XX:+DisableExplicitGC -Dcom.sun.management.jmxremote -XX:PermSize=256m -XX:MaxPermSize=512m -XX:-UseGCOverheadLimit"

Run Tomcat

```
/bin/startup.sh
```

Check the Initialization

```
tail -f /deployment_test/apache-tomcat-7.0.59/logs/catalina.out
```

Initial TAP setup

You need to configure some files in order to adapt it for your needs. Shutdown Tomcat and go to /deployment_test/apache-tomcat-7.0.59/webapps/tap

Configure WEB-INF/classes/global_variables.properties

```
recaptcha.verificationurl = http://www.google.com/recaptcha/api/verify
recaptcha.privatekey = YOUR_RECAPTCHA_PRIVKEY
recaptcha.publickey = YOUR_RECAPTCHA_PUBKEY

tap.baseurl = http://localhost:8080/tap
tap.managers.email = manager1@yourhost.com, manager2@yourhost.com #They will receive emails if people join groups
tap.email.noreply = no-reply@yourhost.com
```

You need to create a recaptcha [?https://www.google.com/recaptcha/admin#list](https://www.google.com/recaptcha/admin#list) and set both private and public keys in the previous file. You also need to set your base url like localhost:8080/tap or yourhost/tap, the managers emails which allow people to keep in touch of the group events (when user wants to join a group, etc)

Configure WEB-INF/classes/mail.properties

```
mail.port = port_number
mail.host = email_host
```

You can change the email templates optionally in WEB-INF/classes/templates

You can change the messages optionally in WEB-INF/classes/locale/messages.properties

Thredds and TAP deployment

Start the Tomcat instance and the Derby network instance. If Thredds is not able to start due to a directory error, please create a folder called ?thredds? in \$CATALINA_HOME/content to solve that issue.

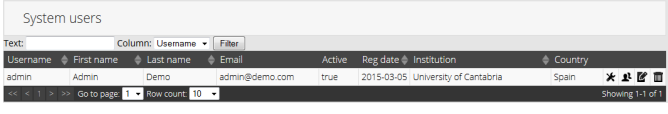
First steps in TAP

When the initial setup is finished TAP is ready to register users, send confirmation emails, let people join groups, etc. With your preconfigured TAP there is a Derby db with demo data. If you have deployed successfully both Thredds and TAP and they are running, go to [?http://yourhost/tap](http://yourhost/tap) and log in with the following credentials:

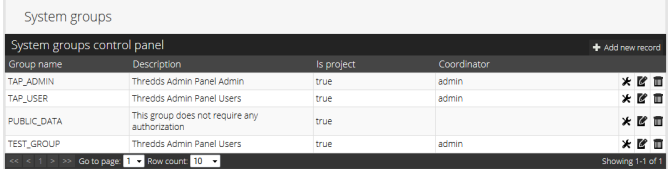
```
username: admin
password: adm!n
```

You will find new admin options in the main menu. From there you can control users, roles, groups, dataset policies and send messages to your users or a set of them.

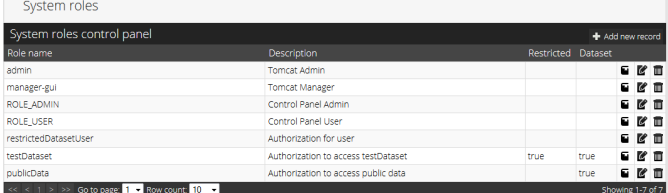
System users

 <p>The screenshot shows a table titled 'System users' with columns: Username, First name, Last name, Email, Active, Reg date, Institution, and Country. One row is visible with the following data: Username: admin, First name: Admin, Last name: Demo, Email: admin@demo.com, Active: true, Reg date: 2015-03-05, Institution: University of Cantabria, Country: Spain.</p>		<p>Operations:</p> <ol style="list-style-type: none"> User roles: Add or remove user roles directly. User groups: Add or remove user groups directly (Some requires user acceptance) Edit record: Edit user details. Delete record: Remove a user from the app and the relations with roles and groups.
---	--	---

System groups

 <p>The screenshot shows a table titled 'System groups control panel' with columns: Group name, Description, Is project, and Coordinator. One row is visible with the following data: Group name: TAP_ADMIN, Description: Thredds Admin Panel Admin, Is project: true, Coordinator: admin.</p>		<p>Operations:</p> <ol style="list-style-type: none"> Group roles: Add or remove roles from the group. Edit record: Edit group details. Delete record: Remove a group from the app and the relations with roles and users.
---	--	--

System roles

 <p>The screenshot shows a table titled 'System roles control panel' with columns: Role name, Description, Restricted, and Dataset. One row is visible with the following data: Role name: admin, Description: Tomcat Admin, Restricted: true, Dataset: true.</p>		<p>Operations:</p> <ol style="list-style-type: none"> Policy assignment: Add or remove policies from the role/dataset. Edit record: Edit role details. Delete record: Remove a role from the app and the relations with groups and users.
--	--	---

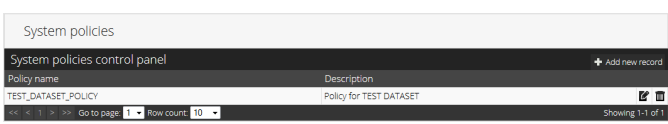
Roles have metadata. You can insert the following keys to show them:

1. url
2. type
3. isPrivate
4. label

Example:

```
label=System4;url=http://www.meteo.unican.es/tds5/catalogs/system4/System4Datasets.html;
```

System policies

 <p>The screenshot shows a table titled 'System policies control panel' with columns: Policy name and Description. One row is visible with the following data: Policy name: TEST_DATASET_POLICY, Description: Policy for TEST DATASET.</p>		<p>Operations:</p> <ol style="list-style-type: none"> Edit record: Edit policy details. Delete record: Remove a policy from the app and the relations with roles.
---	--	---

System messages

You can send messages to your users filtering by groups, newsletter, admins, etc.

□

The messages you send are customized. You don't need to include Hi, Dear nothing or goodbye. The template is the following:

```
Dear admin,

{YOUR MESSAGE}

Best regards,

TAP Manager
```

Thredds restricted dataset

Thredds allows users to restrict dataset access in two different ways:

- URL restriction using Tomcat: difficult to maintain. You need to restrict every dataset by URL and set the role in the web.xml
- Dataset restriction using TDS Catalog: most commonly used by adding an attribute on a dataset or datasetScan element in the TDS catalog. Eg, restrictAccess=?roleName?

If you set the mentioned attribute in a Dataset, users need two roles to access it: restrictedDatasetUser and roleName. That means, every user who wants to access to a restricted dataset needs the restrictedDatasetUser role by default and also the role of the dataset. Here is where TAP does the job for you.

Restrict dataset access

As we mentioned, TAP manages users, roles, groups, policies and their relationships. It is a common scenario to have a dataset with policies that prevent you of doing an illegal use of the data. In the given example you will find the default Thredds catalog example. The first step is to protect the dataset:

```
<dataset name="Test Single Dataset" ID="testDataset"
        serviceName="dap" urlPath="test/testData.nc" dataType="Grid" restrictAccess="testDataset" />
```

When a user tries to access this dataset, the Thredds authorizer expects to find the roles "testDataset" and "restrictedDatasetUser" in the user's granted authorities.

Allow access to a restricted dataset from TAP

To allow access to the previous dataset, create the role in /tap/admin/roles "Add new record":

✕
Add new record

Role name

Description

Restricted

Dataset

Fill the form:

- Name: testDataset
- Description: a description of the dataset
- Restricted: True if you want to moderate its access. Users will wait until a confirmation from an Admin or manager
- Dataset: True in this case. If you want to create an internal role in TAP set to false.

Create a policy in /tap/admin/policies "Add new record" :

×
Add new record

Policy name

Description

Description of the policy

Disable policy roles to users

Agreement

Source Save Undo Redo Find Link Image Table List Omega

B *I* ~~S~~ I_x

☰ ☷ ☰ ☷ ☰ ☷ ☰ ☷

Styles Normal Font Size A- A+ ↔

LICENCE AGREEMENT FOR ECMWF PRODUCTS IN ECOMS-UDG: <http://meteo.unican.es/ecoms-udg>

Interpretation. Words in the singular shall include the plural and vice versa and words used below shall mean:

Products: Data from operational and research seasonal re-forecasts. Real-time forecast data are not included.

ECMWF: The European Centre For Medium Range Weather Forecasts (ECMWF) whose principal place of business is at Shinfield Park, Reading, RG2 QAX, United Kingdom.

Registered users: users working on ECOMS (ie partners in SPECS, EUPORIAS and NAELIM projects) who

body p

Cancel
Save

Fill the form

- Name: TEST_DATASET_POLICY
- Description: Description of the policy
- Disable policy: ignore this in the creation. If you edit the policy and you want users accept again the agreement, set to true.
- Agreement: You can paste html directly by clicking in the source button.

Create a Group in /tap/admin/groups "Add new record":

□

After creating the Policy, associate it with the dataset (role) testDataset:

✕
Policies of role TEST_GROUP

Roles

Role selection

Assigned roles

Name: testDataset

Description: Authorization to access testDataset

✕ Remove

Fill the form

- Name: TEST_GROUP
- Description: Description of the group
- Is project: If the group represents a project like EUPORIAS, VALUE, CORDEX...
- Coordinator: If you want to delegate the user acceptance. TAP will send an email to the coordinator containing two links: accept or reject the user group request.

After creating the group, it is necessary to associate the dataset(role) testDataset to the group:

✕
Policies of role TEST_GROUP

Roles

Role selection

Assigned roles

Name: testDataset

Description: Authorization to access testDataset

✕ Remove

Hidden modes selection

Show dataset select to find groups which include selected datasets:

`/user/groups?dataset=ALL`

Show matched groups filtered by a given dataset:

`user/groups?dataset=system4`

Select groups

`user/groups?groupNames=EUPORIAS`

Issues

If you find any issue, problem or make a comment, please [drop us a ticket!!](#).

You can submit the ticket anonymously, but if you want to receive notifications on updates and feedbacks you can provide your e-mail address in the ticket form.

This is the list of tickets, so far:

<u>Ticket</u>	<u>Summary</u>	<u>Type</u>
#6113	Field Description max length when creating a group	defect
#6114	TAP does not allow to assign roles to a group	defect
#6117	Show dataset URL to allow access from R	enhancement