TBL Server Installation and Integration

Page Contents

1 Installation of the TopBraid Live Server
2 TBL Server Platform Considerations
    2.1 Application Server and Supported Platforms
    2.2 Suggested Java Configuration
    2.3 Operating Systems
        2.3.1 Installation on Linux
        2.3.2 Installation on Windows Server
        2.3.3 Java configuration
    2.4 Other Issues to Consider Prior to Installation
        2.4.1 Workspace location
        2.4.2 Security
        2.4.3 Internet Explorer and SSL
        2.4.4 Java EE memory management
3 TBL Server Installation
    3.1 Tomcat Installation Instructions
        3.1.1 Accessing the TBL application
        3.1.2 License registration
        3.1.3 TopBraid GUI apps require persistence configuration: Application data storage
        3.1.4 Configuring authentication
        3.1.5 Reloading a modified web.xml
    3.2 Advanced Equinox Features
        3.2.1 Modifying the configuration area
    3.3 Configure Logging
    3.4 Tomcat settings - other
4 LDAP Configuration
    4.1 Integration Instructions
        4.1.1 LDAP Configuration within Tomcat
            4.1.1.1 Identifying/Creating a Group and at Least One User in That Group
            4.1.1.2 Configuring Tomcat to Use LDAP
        4.1.2 Active Directory Integration within Tomcat
            4.1.2.1 Populate LDAP
            4.1.2.2 Configuring Tomcat to Use LDAP
        4.1.3 Configuring TBL to Use LDAP
    4.2 Starting an LDAP Service in Linux
        4.2.1 Identifying/Creating a Group and at Least One User in that Group
    4.3 LDAP Servers (Service Providers)
5 Additional Integration
    5.1 Authenticating Service Calls
        5.1.1 Authentication for TopBraid Suite Server
        5.1.2 End User Authentication
        5.1.3 Web Service Authentication
            5.1.3.1 Basic Authentication
            5.1.3.2 Form-Based Authentication
            5.1.3.3 Authenticating calls that use SPARQL's SERVICE keyword
            5.1.3.4 Role-Based Access Control (RBAC) in TopBraid

Installation of the TopBraid Live Server

The TopBraid Live Enterprise (TBL) Server is designed using Java servlet technology and deployed using a Web Application Server, also known as a Servlet Container, such as Apache Tomcat or Oracle WebLogic. Installation consists of deploying the WAR (Web Application Archive) file in the application server.

The WAR file is obtained by contacting TopQuadrant sales to receive the login/password for downloading the Enterprise Vocabulary Net install files. The install files consist of the WAR file for TopBraid Live (tbl.war) and the TBL Installation Guide (this document).

The installation of TopBraid Live (TBL) is slightly different depending on both the operating system and Web Application Server. Detailed instructions for Tomcat Installation Instructions , Installation Instructions for Linux and Installation Instructions for Windows Server are provided in the sections below. In general, installation consists of:

1. Deploy the TopBraid Live `tbl.war` file to the Tomcat server by either using Tomcat Manager or manually copying it to Tomcat's `webapps` directory.

2. (Optional) Setting the location of the TBL workspace. By default, the workspace is set to /var/lib/ontologies. Upon deployment of the (TBL) application, library files will be written to this folder, including TopBraid (system library for TopBraid Suite features), `server.topbraidlive.org` (TBL server files for sessions, etc.), and other TopBraid Suite features. This folder will also contain all projects uploaded to the server. See Methods to Deploy Projects to Server .

   The location of the workspace can be changed by editing the `init-param` element in $APP_HOME/webapps/tbl/tbl/WEB-INF/web.xml, where $APP_HOME is the home directory of the application server.

   If permissions for the user running the application server are insufficient to write to the /var/lib/ontologies folder, full deployment of TopBraid Live will not succeed. Edit the web.xml to a folder with write permissions and re-start the application server.
3. (Optional) Setting authentication and role-based access control through Tomcat or a LDAP server. These are explained in more detail throughout this document.

TBL Server Platform Considerations

These sections describe some aspects of the server platform to consider prior to installing the TBL server.

Application Server and Supported Platforms

The TBL server uses Apache Tomcat running on Java. Some persistence options use relational databases. For the details of your TBL server version, see the **Supported Platforms** page for compatible platform components (e.g., Java, database, etc.). This also includes sections with details and suggested configurations for installing on Linux or Windows Server.

Suggested Java Configuration

If you are using both TopBraid Live and another TopBraid server application on the same machine, we recommend that you run each in a separate JVM by running them with separate instances of Tomcat.

It is necessary to ensure that you have adequate physical memory to run TBL and that you have configured your Web Application Server to use it.

For the JVM 8, the recommended JAVA_OPTS (environment variable) parameter setting for heap space is:

```
-Xmx4g
```

The 4g value for Java heap space (-Xmx) represents a maximum allocation for Tomcat servers. If this figure is too low, then you might likely see a HTTP 500 server error on the first page accessed.

See the installation instructions below for your operating system for more details concerning how to set the parameters.

Operating Systems

Installation on Linux

This section describes details and suggested configurations when using Linux as the base operating system and Tomcat as the Web Application Server.

A suggested configuration for TopBraid Live is as follows:

Operating Systems:	Current version of Linux with 2.6 or higher kernel such as Red Hat 7 or Ubuntu 14.04 LTS
Java Runtime Environment:	Java 8
Web Application Server	Must support 2.4 release of the Java Servlet specification. Apache Tomcat 8 (see supported-platforms for specific versions)
Permissions	Full access to the Tomcat configuration area is required.

TopBraid Live also requires that the `nofile` limit be increased as it opens a large number of JAR files, and each TDB database also has a large number of files. The default is 1024 open files and sockets. Reset this by adding the following two lines to the /etc/security/limits.conf configuration file:

```
tomcat8         soft    nofile          2048
tomcat8         hard    nofile          8192
```

Installation on Windows Server

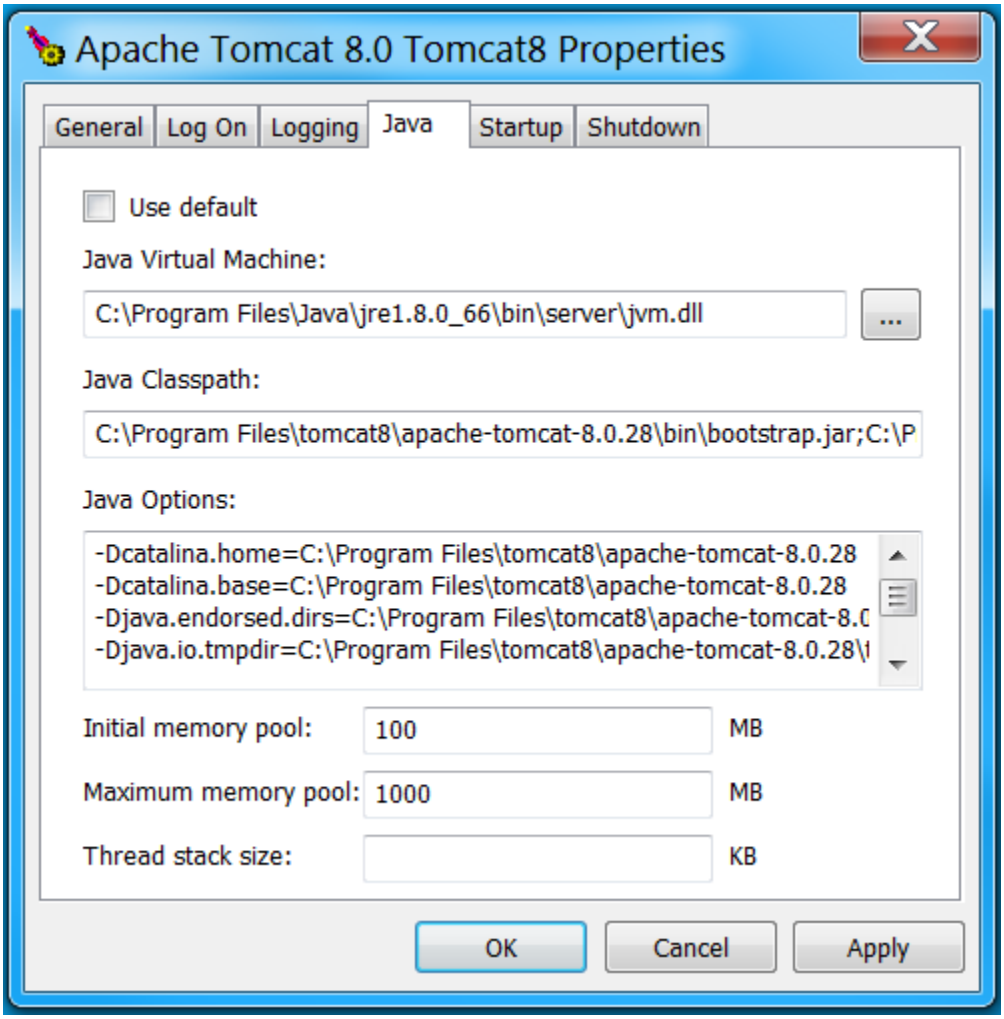
This section describes details and suggested configurations when using Windows Server as the base operating system and Tomcat as the Web Application Server.

A suggested configuration for TopBraid Live is as follows:

Operating Systems:	Windows Server, (e.g. 2008, 2012, 2016) Administrator access is required for installation.
Java Runtime Environment:	Java 8
Web Application Server	Must support 2.4 release of the Java Servlet specification. Apache Tomcat 8 (see supported-platforms for specific versions)

Earlier versions of Java/Tomcat 8 are expected to work but are known to have memory leaks that may affect some TopBraid Live functions.

Java configuration



The picture shows the use of the Tomcat properties window, which is reached by launching `tomcat8w.exe`. The permanent generation options should be added as shown, under `Java options:`. The memory options should be added under `Maximum memory pool:`. This step is performed before starting Tomcat.

Other Issues to Consider Prior to Installation

The following issues may need configuration on your system.

Workspace location

The workspace location is defined in the Basic Configuration section of section of Tomcat Installation Instructions - the Workspace location field.

TopBraid Live requires a workspace compatible with the Eclipse Equinox implementation of the OSGi R4 framework. (This is similar to the TopBraid Composer workspace.) The default location, which may be modified in TopBraid Live's `web.xml` file, is `/var/lib/ontologies`. This workspace location must be writable by the process running the Web Application Server.

Security

The Apache Tomcat $TOMCAT_HOME/conf/tomcat-users.xml file stores contain user IDs and passwords. Unless you've configured Tomcat to use some other authentication method (JNDI LDAP), you can use a user ID and password from this file.

Tomcat installations (and most Java EE servers) generally assume that authentication occurs from an outside source. Out of the box, Tomcat is configured to use tomcat-users.xml to load authentication information into memory. If, for example, LDAP is used instead, then a user trying to log into a Tomcat application will send information from the sign-in (either Basic or Form authentication) to the authentication service, which responds with role/user id information.

For more details, see Configuring authentication or LDAP Configuration. For information on access control to graphs in the workspace, see TBL Rights Management.

Internet Explorer and SSL

For your users using TopBraid applications from Internet Explorer over SSL, certain settings in your httpd.conf file will ensure that the application's behavior will be consistent with its use from other browsers. The following shows the end of a VirtualHost element in an httpd.conf file, with the IE-related lines in bold.

```
   SSLEngine On

   SSLCipherSuite ALL:!ADH:!EXPORT:!SSLv2:RC4+RSA:+HIGH:+MEDIUM
   SSLProtocol all -SSLv2

   SSLCertificateFile    /etc/ssl/certs/server.crt
   SSLCertificateKeyFile /etc/ssl/private/server.key

   SSLCertificateChainFile /etc/ssl/ca/sub.class1.server.ca.pem
   SSLCACertificateFile    /etc/ssl/ca/ca.pem
    SetEnvIf User-Agent ".*MSIE.*" is_ie nokeepalive ssl-unclean-shutdown downgrade-1.0 force-response-1.0
#Needed for IE Flash image loading bug Header set Cache-Control no-store env=is_ie Header unset Pragma
env=is_ie
</VirtualHost>
```

The SetEnvIf line sets a series of environment variables to true for user agents with MSIE in their name, and the two Header lines adjust the HTTP header that will be sent to the browsers.

Java EE memory management

TopBraid EDG is a Java EE application. To manage the memory, please set up monitoring software recommended by your organization.

Typical analysis for any memory issues are below:

- Stop and restart from Tomcat's admin module; after stopping TBL, wait five minutes and then select the find leaks option from Tomcat's admin page.
- The most severe leaks make it impossible to cleanly shutdown Tomcat and a "kill -KILL" command may be necessary.

To make a memory dump of your server, first find the process ID of your Tomcat process, and then issue the following command:

```
    jmap -dump:live,format=b,file=TB-dump.bin $PID
```

Then, compress the file TB-dump.bin.

TBL Server Installation

TopBraid Live runs on Web Application servers that support the 2.4 release of the Java Servlet specification. The following sections detail specific instructions for running TopBraid servers on Tomcat.

Tomcat Installation Instructions

The instructions give details for Tomcat, and may need variation depending on local setup. The details vary with other Web Application Servers and between different versions of Tomcat. Manual editing of the `web.xml` file is no longer supported. The following steps will create a correct `web.xml` file. Manual edits made for other applications are outside of the scope of this document, and not supported by TopQuadrant.

1. Copy the downloaded zip archive file named `TopBraidTBL-[NNN].zip` to any location.

   The WAR file `tbl.war` included in the zip is to be deployed on the Web Application Server. These instructions assume you do not rename this file.
2. Undeploy any current `tbl.war` instance.

   - If hot deploy is enabled, delete `tbl.war` from the deployment directory and Tomcat will do the rest. The file to delete will be:

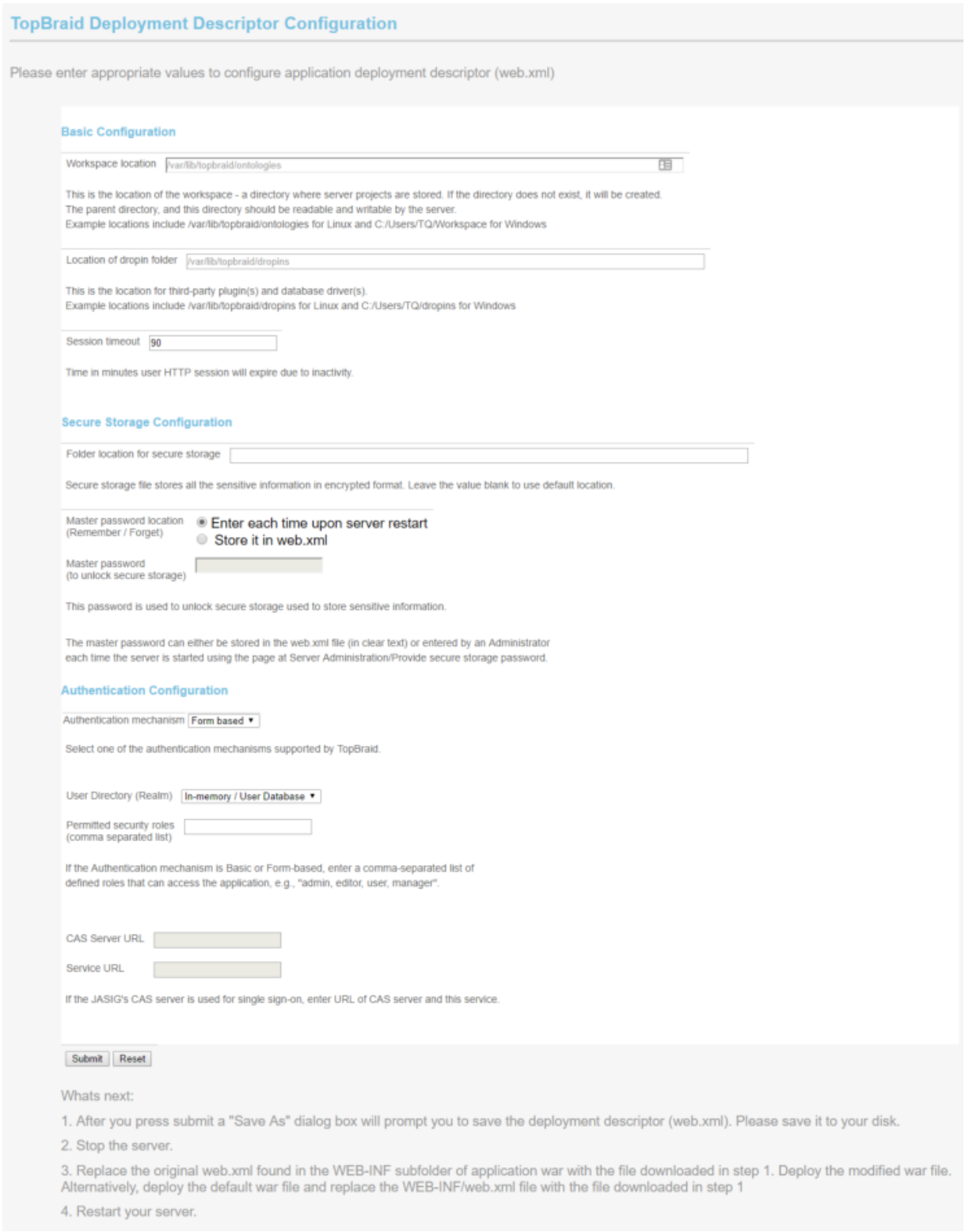
     `$TOMCAT_HOME/webapps/tbl.war`

     These paths may be different if your Tomcat installation is configured with a different deployment directory.
   - Without hot deploy, or if hot deploy fails, you may need to stop the server and then manually undeploy:

     For Tomcat, remove the following:

     - `$TOMCAT_HOME/webapps/tbl.war`
     - the directory `$TOMCAT_HOME/webapps/tbl`
     - the directory `$TOMCAT_HOME/work/Catalina/`*`[server_name]`*`/tbl`
3. In a separate terminal or window, open the log file for Tomcat in `$TOMCAT_HOME/logs/catalina.out`. This is the log file that prints startup messages for Tomcat.
4. Copy `tbl.war` to the deployment directory: `$TOMCAT_HOME/webapps`.
5. Check the log file. At this stage, relevant messages will include:

   ```
   Deploying web application archive tbl.war
   ```
6. Open the tbl console by directing a browser to the TopBraid Live server being installed: `http://`*`[Web-application-server-host]`*`:`*`[Web-application-server-port]`*`/tbl`. A window similar to the following will appear:



1. TBL uses *Secure Storage* to save passwords that it uses (e.g., for databases). If you provide a *Master password* (recommended), TBL will use it to encrypt its stored passwords. Otherwise, TBL only uses Base64 encoding (i.e., unencrypted). Saving the Master password in the TBL web.xml simply uses plain text, so you must ensure that access to the web.xml is restricted. Alternatively, instead of using the web.xml to save the Master password, it can be manually entered by an administrator each time the server is started: see *TBL Administration*, Access Control: Provide secure storage password.
2. Fill in the details on this page and "Submit".
3. A `web.xml` file will be generated and either displayed by your browser or saved to your local file system.

7. Stop Tomcat. Replace the `$TOMCAT_HOME/webapps/tbl/WEB-INF/web.xml` with the `web.xml` generated in the previous step.

8. Start Tomcat.

Accessing the TBL application

For access to the TBL Console, send your browser to the URL `http://`*`[Web-application-server-host]`*`:`*`[Web-application-server-port]`*`/tbl/tbl/`

(This corresponds to TopBraid Composer ME's `http://localhost:8083/tbl/` .)

NOTE: Until TBL completes the license registration step, only *one administrator* may log in.

On the first visit, TopBraid Live itself is initialized, and you should see entries like the following added to the log:

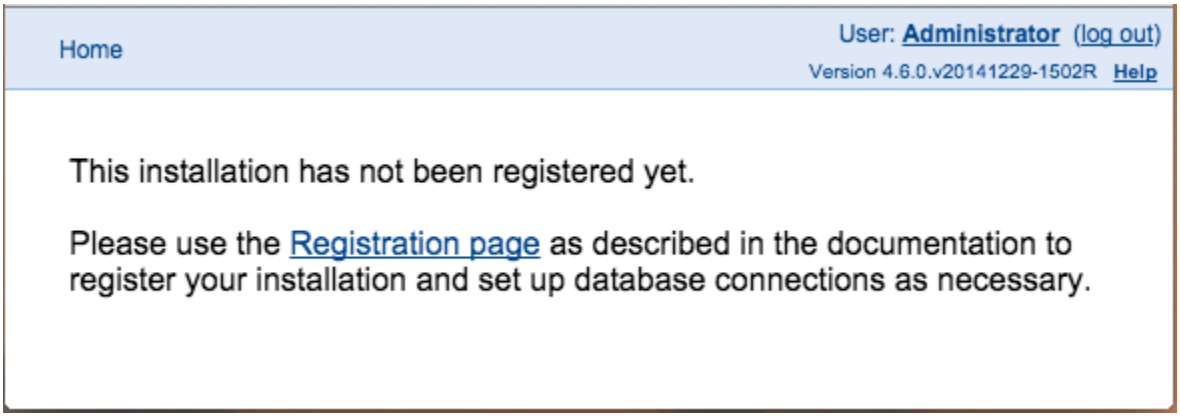
```
  09-Sep-2016 13:06:36.478 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Server
version: Apache Tomcat/8.0.33
  09-Sep-2016 13:06:36.479 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Server built:
Mar 18 2016 20:31:49 UTC
  09-Sep-2016 13:06:36.480 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Server
number: 8.0.33.0
  <etc., etc.>
  09-Sep-2016 13:07:13.371 INFO [main] org.apache.coyote.AbstractProtocol.start Starting ProtocolHandler
["http-apr-8080"]
  09-Sep-2016 13:07:13.384 INFO [main] org.apache.coyote.AbstractProtocol.start Starting ProtocolHandler
["ajp-apr-8009"]
  09-Sep-2016 13:07:13.395 INFO [main] org.apache.catalina.startup.Catalina.start Server startup in 36001
ms
```

If there are permission problems with the workspace, this step will not work. If the workspace does not exist, or is an empty directory, it will be created and populated.

The project `server.topbraidlive.org` and several others will be created in the workspace. If not, there were initialization problems and the TopBraid server will not work properly. In a Linux installation, the most common source of problems are folder permissions for the process running the Web Application Server. Try removing the workspace folder and reloading `web.xml`, making sure that the process has write access to the parent folder of the workspace folder.

License registration

After these steps, log into the TBL server. A message will appear that point to the registration page, as shown in the following screen shot:



1. Click on the `Registration Page` link. A page will appear stating that there are "No currently registered products!"
2. Click on the `Change or Update Registration Information` link. Use the `Choose File` button to choose the license file (.lic file) sent by TopBraid support when the license was issued.

A dialog will appear, verifying that the license was updated. The licenses and their expiration date will be displayed. This page is always available through the Server Administration tab and choosing the Server Administration then the Product Registration links.

TopBraid GUI apps require persistence configuration: Application data storage

For GUI applications (excluding TopBraid Live), an administrator must configure the persistence installation for graph data, following these steps: Teamwork Platform Parameters: Application data storage.

The TBL server-based application does not include the sample data that is included in TopBraid Composer-Maestro Edition (TBC-ME). If desired, the examples can be deployed from TBC-ME to the server by clicking on the sample project in TBC-ME's Navigator window and choosing Export... > TopBraid Composer > Deploy Project to TopBraid Live Server. Permission will default to "Administrator" for the sample project. Please update this in the role management page if needed.

Configuring authentication

The web.xml file created in the Tomcat Installation Instructions contains configuration settings based on how you filled out the form at installation. Use either the tomcat-users configuration, here, or the LDAP configuration, below. If using **In Memory / User Database**, please ensure that your <TOMCAT_ROOT>/conf/tomcat-users.xml is setup and users belong to appropriate roles. An example is below:

```
  <role rolename="Admin"/>
  <role rolename="Manager"/>
  <role rolename="Editor"/>
  <role rolename="Viewer"/>
  <user username="Admin_user" password="password32" roles="Admin,Manager"/>
  <user username="Editor_user" password="password54" roles="Editor"/>
  <user username="Guest" password="password76" roles="Viewer"/>
```

It is imperative that your web.xml security roles match the roles defined in this tomcat-users.xml file.

The LDAP Configuration section of this installation guide has information on using TopBraid Live with the LDAP.

Reloading a modified web.xml

With Hot Deploy

If hot deploy is enabled, then this step is already done. A Tomcat log file will show something like the following:

```
May 23, 2009 2:13:23 PM org.apache.catalina.startup.HostConfig checkResources
INFO: Undeploying context [/tbl]
May 23, 2009 2:16:28 PM org.apache.catalina.startup.HostConfig deployWAR
INFO: Deploying web application archive tbl.war
osgi> May 23, 2009 2:18:26 PM org.apache.catalina.startup.HostConfig checkResources
INFO: Reloading context [/tbl]
May 23, 2009 2:18:26 PM org.apache.catalina.core.StandardContext stop
INFO: Container org.apache.catalina.core.ContainerBase.
   [Catalina].[localhost].[/tbl] has not been started
osgi> May 23, 2009 2:18:29 PM org.apache.catalina.core.StandardContext start
INFO: Container org.apache.catalina.core.ContainerBase.
   [Catalina].[localhost].[/tbl] has already been started
```

This may take a couple of minutes.

Other Web Application Servers will report the same behavior in different ways.

Without Hot Deploy

Go to the console for Tomcat or the equivalent if using another WebApplication Server. Find the `tbl` application and either *Reload* or *Stop* and then *start* the application.

Advanced Equinox Features

Many other Equinox features can be accessed by adjusting the Equinox command line or the `config.ini` file. (See below for more on this.) Many different options for Eclipse are documented. Some of them may be useful in TopBraid Live. However, very few have been tested. Please contact TopQuadrant support (support@topquadrant.com) if you would like more information about using any of these options with TopBraid Live.

Modifying the configuration area

Equinox, like Eclipse, uses a configuration area to store some system files. By default this is created in a temporary area inside the Web Application Server. It can be deleted and recreated using the Equinox redeploy option from the TopBraid Live Administration menu. Using Tomcat, a possible path for the configuration area is:

`work/Catalina/localhost/tbl/eclipse/configuration/`

To use a configuration area other than the default it is necessary to modify `web.xml` and initialize the new configuration area. (This is done automatically if using the default configuration.)

Using a custom configuration area is useful if, for example:

- You wish to customize the initialization of the configuration.
- There are difficulties using the configuration area within the temporary files provided by the web application server. For example, a lock file is used in the configuration area, which is incompatible with certain NFS arrangements.

The Steps

1. Initialize New Config Area.
    1. Create a new directory.
    2. Copy `webapps/tbl/WEB-INF/eclipse/configuration/config.ini` into the new directory.
    3. Verify that the permissions and owner are appropriate. (The process running the web application server must be able to both read and write the configuration area.)
2. Modify the `commandline` param of the TopBraid Live `web.xml` to include the `-configuration` option pointing to the new directory. For example:

```
  <init-param>
 <param-name>commandline</param-name>
 <param-value>
 -data /var/lib/ontologies
 -configuration /var/lib/config320
 </param-value>

 </init-param>
```

You will find a sample `web.xml` file with the name `web-config.xml` in the `installation` directory of your TopBraid Live distribution zip file.

Configure Logging

If you encounter server startup problems, the following settings in Apache Tomcat's conf/logging.properties file can give you a better picture of what's going on:

```
handlers = 1catalina.org.apache.juli.FileHandler, 2localhost.org.apache.juli.FileHandler,
  3manager.org.apache.juli.FileHandler, 4host-manager.org.apache.juli.FileHandler,
  5tbl.org.apache.juli.FileHandler, java.util.logging.ConsoleHandler
5tbl.org.apache.juli.FileHandler.level = FINE
5tbl.org.apache.juli.FileHandler.directory = ${catalina.base}/logs
5tbl.org.apache.juli.FileHandler.prefix =tbl.
org.apache.catalina.core.ContainerBase.[Catalina].[localhost].[/tbl].level = INFO
org.apache.catalina.core.ContainerBase.[Catalina].[localhost].[/tbl].handlers = 5tbl.org.apache.juli.
FileHandler
```

Keep in mind that:

- The handlers setting would replace the existing one. It is shown as three lines above, but should be one line in your logging.properties file.
- TopBraid Suite messages, including those related to startup issues, are sent to their own tomcat log file in $TOMCAT_HOME/logs/tbl.log.

Tomcat settings - other

A tomcat SSL connector, without an exclusion rule will require some characters be encoded where a flat http connector will not. This can cause problems with embedded links.

The relaxedQueryChars exclusion rule can be set to relaxedQueryChars="[]|{}^\`"<>"

.

LDAP Configuration

This section describes how to use the TBL server with LDAP authentication.

Integration Instructions

Authentication is the process by which users log on to TBL. It is possible, but not advised, to run TBL without authentication. In this case, all users get full administrative privileges.

TBL uses the Tomcat web-app server for authentication. This is intended to help implementation of policies such as single sign-on. TBL provides no further features for single sign-on. If this is a policy mandated for your environment, first find some other Java EE based service and understand which Web Application Server they are using, and how they authenticate users in accordance with the single sign-on policy. Then contact TopQuadrant with this information for advice as to how to configure TBL in a similar way.

This section provides instructions for integrating TBL with LDAP in Linux and Active Directory in Windows Server.

LDAP Configuration within Tomcat

These instructions cover the configuration of LDAP within Tomcat. For any other web application server, there could be some differences. Note that the exact formats used will depend on your LDAP configuration. Also, although the description below has a step to initiate an LDAP service, normally this is not required because you will be using a pre-existing LDAP service for configuring Tomcat and TBL.

Identifying/Creating a Group and at Least One User in That Group

In the sample configuration routine shown below, we will set the security to permit a specific group to access TBL. This group needs a name, which is configured within LDAP. You may already have an appropriate group.

Configuring Tomcat to Use LDAP

Next, we need to configure tomcat to connect to LDAP. This is specified in this document : https://tomcat.apache.org/tomcat-8.5-doc/realm-howto.html#JNDIRealm

This is the difficult step. It requires adding an entry to `server.xml`, found in the `conf` folder for Tomcat. For our example, the entry is as follows:

```
<Realm   className="org.apache.catalina.realm.JNDIRealm" debug="99"
   connectionURL="ldap://localhost:389"
   userPattern="uid={0},ou=people,dc=tqinc,dc=info"
   roleBase="ou=groups,dc=tqinc,dc=info"
   roleName="cn"
   roleSearch="(memberUid={1})"
   />
```

inside of the Engine element. Note that in this expression, `{0}` and `{1}` both stand in for the username (they are written literally as `{0}` `{1}` in the web.xml file), but the actual expressions such as `uid={0},ou=people,dc=tqinc,dc=info` will be different depending on your LDAP service.

For reference, this is for group configuration that looks like this in LDAP:

```
keefe@keefetq:~$ ldapsearch -xLLL -b "dc=tqinc,dc=info" -s sub "(memberUid=keefe)"
dn: cn=acme,ou=groups,dc=tqinc,dc=info
objectClass: posixGroup
cn: acme
gidNumber: 10001
description: Group account
memberUid: keefe
```

If your LDAP directory is setup in a straightforward way, it should be easy to modify the search expression appropriately. However, you may need assistance from someone who knows your LDAP system.

Three pieces of information are needed:

1. A base pattern to identify groups in LDAP, (in this case: `ou=groups,dc=tqinc,dc=info`).
2. A property at which the rolename we will use in securing TBL may be found, in this case:`cn`
3. A search filter that indicates role membership, in this case : `(memberUid={1})` where `{1}` is the username.

Active Directory Integration within Tomcat

These instructions concern integrating Active Directory within Tomcat using the LDAP protocol. Note that the exact format depends on your LDAP configuration.

Populate LDAP

The security to permit a specific group of users to must be set to access TBL. This group needs a name, which is configured within LDAP. You may already have an appropriate group.

1. Create users in Active Directory for TBL use. If existing users need to use the TBL, then ignore this step.
2. Create a TopBraid TBL group named **acme** and associate users with this group in Active Directory using the wizard.

Configuring Tomcat to Use LDAP

Next, configure tomcat to connect to LDAP. This is specified in this document : https://tomcat.apache.org/tomcat-8.5-doc/realm-howto.html#JNDIRealm

This is the difficult step. It requires adding an entry to `server.xml`, found in the `conf` folder for Tomcat. For our example, the entry is as follows:

```
<Realm className="org.apache.catalina.realm.JNDIRealm"

               connectionURL="ldap://localhost:3268"
               authentication="simple"
               referrals="follow"

               connectionName="jeremy@tblive.tqinc.info"
               connectionPassword="enter password here"

               userSearch="(sAMAccountName={0})"
               userBase="CN=Users,DC=tblive,DC=tqinc,DC=info"
               userSubtree="true"

               roleSearch="(member={0})"
               roleName="cn"
               roleSubtree="true"
               roleBase="CN=Users,DC=tblive,DC=tqinc,DC=info"

               debug="99"
        />
```

inside of the Engine element.

In the *userSearch* expression, `{0}` stands for the username as typed by the end-user, whereas in the *roleSearch* expression, `{0}` stands for the distinguished name found from the user look up. (These are written literally as `{0}` in the server.xml file), but the actual expressions such as `(sAMAccountName={0})` and `CN=Users,DC=tblive,DC=tqinc,DC=info` will be different depending on your LDAP service. With Active Directory, `(sAMAccountName={0})` and `roleSearch="(member={0})"` should generally be correct.

If your LDAP directory is setup in a straightforward way, it should be easy to modify the search expression appropriately. However, you may need assistance from someone who knows your LDAP system.

Three pieces of information are needed:

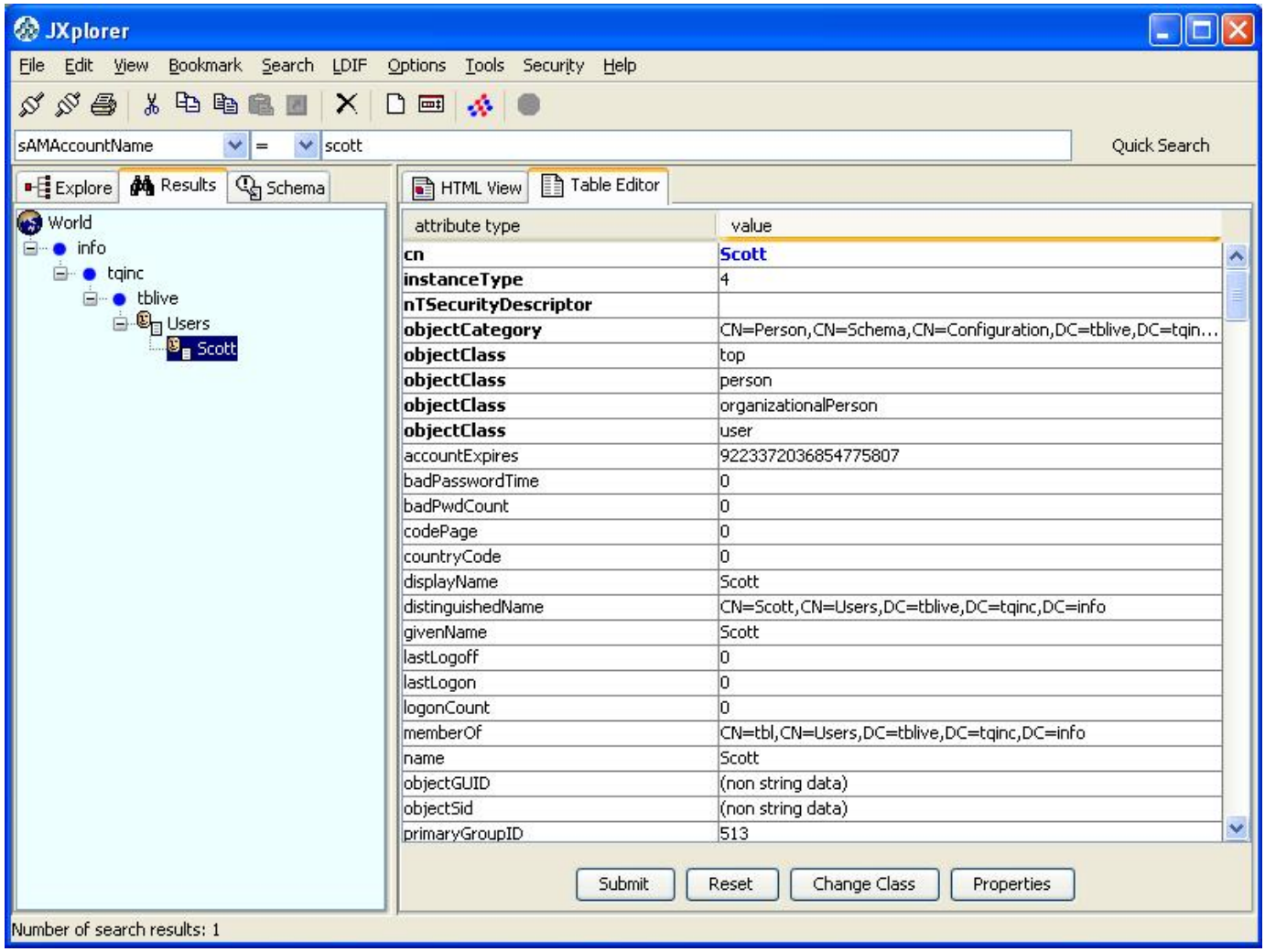
1. A base pattern to identify groups in ldap, (in this case: `CN=Users,DC=tblive,DC=tqinc,DC=info`).
2. A property at which the rolename we will use in securing TBL may be found, in this case:`cn`
3. A search filter that indicates role membership, in this case : `(member={0})` where `{0}`is the distinguishedName (and {1} is the username).

This can also be seen using an LDAP explorer tool something similar to JXplorer

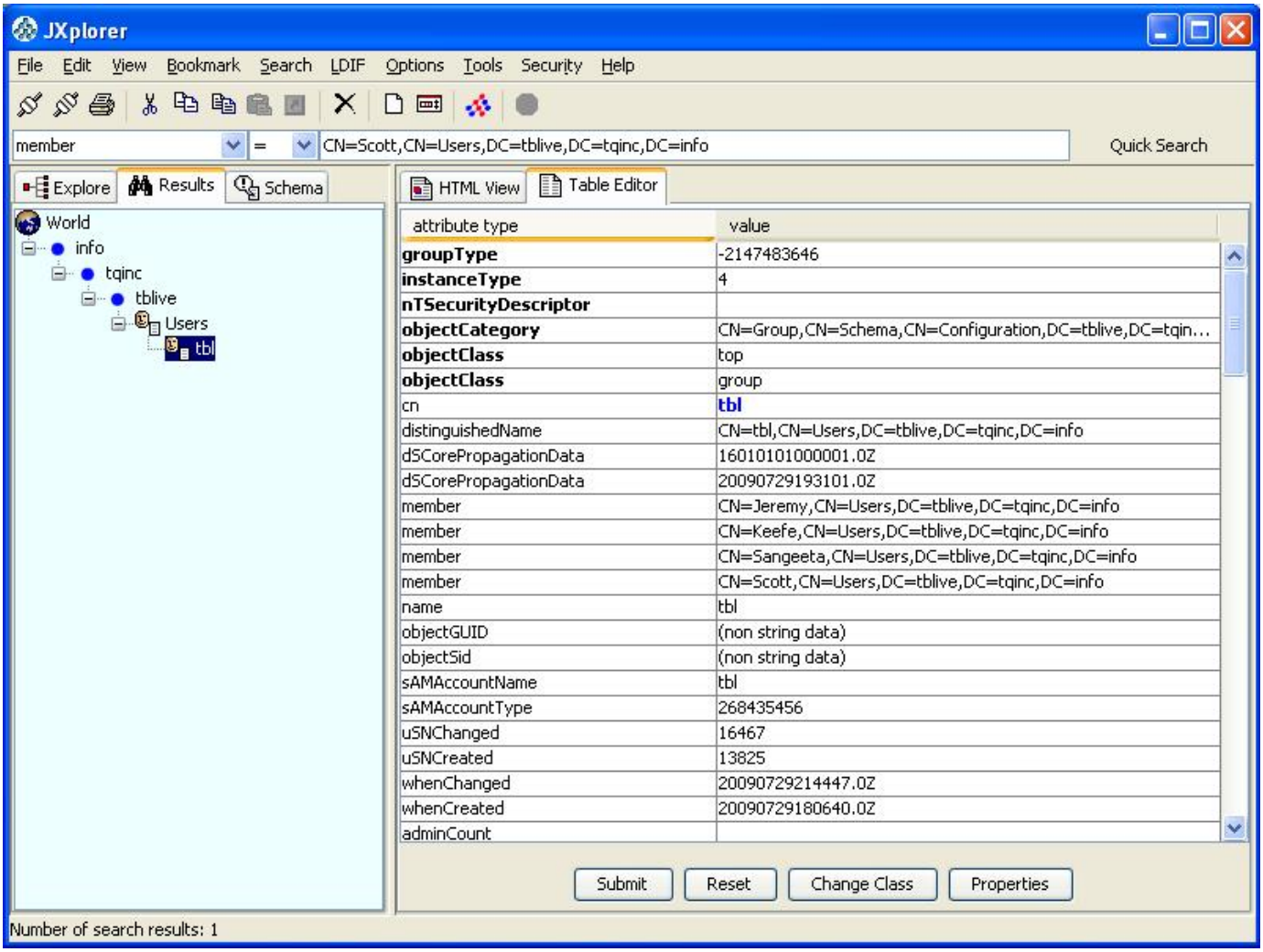
For reference, the screen-shot below shows how the Active Directory group configuration looks like in JXplorer:

Connecting to LDAP:

User record for user Scott:



Group record belonging to user Scott:



Configuring TBL to Use LDAP

Finally, add a security constraint to TBL in the `web.xml file`:

```
<security-constraint>
  <display-name>Example Security Constraint</display-name>

  <web-resource-collection>
  <web-resource-name>Protected Area</web-resource-name>
<!-- Define the context-relative URL(s) to be protected -->
  <url-pattern>/*</url-pattern>
<!-- If you list http methods, only those methods are protected -->
  <http-method>DELETE</http-method>

  <http-method>GET</http-method>
  <http-method>POST</http-method>
  <http-method>PUT</http-method>
  </web-resource-collection>

  <auth-constraint>
<!-- Anyone with one of the listed roles may access this area -->
  <role-name>acme</role-name>
  </auth-constraint>
</security-constraint>

<!-- Default login configuration uses basic authentication -->

<login-config>
  <auth-method>BASIC</auth-method>
  <realm-name>TopBraid</realm-name>
</login-config>
```

```
<!-- Security roles referenced by this web application -->

<security-role>
  <role-name>acme</role-name>
</security-role>
```

Starting an LDAP Service in Linux

This is intended as a quick guide. More detailed instructions can be found elsewhere.

1. Install LDAP. For an example of installation documentation, see the documentation for the Ubuntu OpenLDAP Server.
2. Configure LDAP by running through the setup wizard and setting up a domain name and related information. For example:
   ```
   domain = tqinc.info
   organization name = tq
   ```
3. Populate LDAP

   The example below, which uses the ldapadd utility, is based Ubuntu's OpenLDAP configuration guide, replacing all example.com references with the domain specified above: tqinc.info.

   ```
   keefe@keefetq:~$ ldapadd -x -D cn=admin,dc=tqinc,dc=info -W -f test.ldif
   Enter LDAP Password:
   adding new entry "ou=people,dc=tqinc,dc=info"
   adding new entry "ou=groups,dc=tqinc,dc=info"
   adding new entry "uid=john,ou=people,dc=tqinc,dc=info"
   adding new entry "cn=example,ou=groups,dc=tqinc,dc=info"
   ```

Identifying/Creating a Group and at Least One User in that Group

In this stage, we'll set the security to permit one or more specific groups to access TBL. This group needs a name, which is configured within LDAP.

First we install the LDAP scripts and add a group and a user,remembering to configure the LDAP scripts first. With a different LDAP server, this step will be different.

```
keefe@keefetq:~$ sudo ldapaddgroup acme
Error adding group acme to LDAP
```

The error message shows that the LDAP scripts must be configured first:

```
keefe@keefetq:~$ sudo gedit /etc/ldapscripts/ldapscripts.conf
keefe@keefetq:~$ sudo sh -c "echo -n 'monkey' > /etc/ldapscripts/ldapscripts.passwd"
keefe@keefetq:~$ sudo chmod 400 /etc/ldapscripts/ldapscripts.passwd
keefe@keefetq:~$ sudo ldapaddgroup acme
Successfully added group acme to LDAP
keefe@keefetq:~$ sudo ldapadduser keefe acme
Successfully added user keefe to LDAP
keefe@keefetq:~$ sudo ldapsetpasswd keefe
Changing password for user uid=keefe,ou=people,dc=tqinc,dc=info
New Password:
Retype New Password:
Successfully set password for user uid=keefe,ou=people,dc=tqinc,dc=info
keefe@keefetq:~$
```

LDAP Servers (Service Providers)

The preceding LDAP sections configure TBL to use of LDAP for authentication: user login identity and roles. To further configure TBL for using additional LDAP information, e.g., email, see Server Administration > Setup - Server Configuration Parameters > LDAP Servers (Service Providers).

Additional Integration

Authenticating Service Calls

When accessing TopBraid services via RESTful web service calls, authentication of the invoking entity needs to be considered. There are a few options and tradeoffs when using the application server underlying TopBraid. The following sections provide an overview of Java EE and choices available for authentication of services.

Authentication for TopBraid Suite Server

All authentication issues for TopBraid Suite are handled by using standard Java EE web container methods. Once inside the container, users are free to design their own RBAC designs, such as TopBraid TBL's *User Roles* management for vocabulary or asset models. Some of the details will depend on the authentication Realm set up for Tomcat - i.e. LDAP, AD, etc. However, authentication is always a handshake between the entity requiring access and the web container (e.g. Tomcat), not TopBraid Suite applications.

Authentication works the same across all HTTP methods.

The following outlines some basic concepts and suggestions for handling authentication from user and services. There are a number of ways to support authentication, and many organizations choose to build their own. Organization-specific IT policies will largely dictate interactions with the web container to authenticate users and services. Therefore, TopQuadrant can only play an advisory role for getting started with authentication issues.

Consult Java EE tutorials for a good starting point on authentication methods. The authentication method is chosen in the initial Deployment Descriptor Configuration wizard (see Tomcat Installation Instructions), which sets the login-config/auth-method element in the web.xml file. Only one authentication form can be used per server. TopBraid supports Form, Basic and no authentication:

1. **No Authentication.** No user id or password is required to invoke services. This is useful for Linked Open Data applications and applications are are used openly behind firewalls. This should only be used for read-only servers and the administrator should ensure that the *Enable SPARQL Updates* parameter in Server Administration > Setup - Server Configuration Parameters > Advanced Parameters is set to `false`, which is the default that blocks updates.
2. **Form-based authentication.** This is the best choice for all UI-based applications, such as TBL and SWA/SWP-based applications. Form-based authentication will display a form for entering authentication, pass the challenge to the authentication agent, and respond with the challenge result. On logout, the user is logged out from the container.
3. **Basic authentication.** This should be used for access via 3rd party entities, such as web services. It is not convenient for user-based authentication because the user cannot log out of the system without closing the browser. It is recommended to use Basic authentication when the server is used for web services and an administrator needs to occasionally log in to perform maintenance.

End User Authentication

If user log-in is required, then it is best to use Form authentication. In this case the application (`tbl/tbl`) controls the login and logout pages that are displayed to the user. When a user logs out, they are logged out from the web application container.

Some user information is cached in the `/server.topbraidlive.org/dynamic/users.ttl` file of the active TopBraid workspace. This information is passed to TopBraid from the web application container at login. Helper functions are available from any SPARQL context, including `smf:currentUserName()`, `smf:hasCurrentUser()`, and `smf:userWithName()`.

Web Service Authentication

Web services invocation on servers requiring authentication can use Basic or Form-based authentication. Basic authentication using HTTPS encryption is recommended.

Basic Authentication

Basic authentication should be used for access by web services. Basic authentication relies on a Base64 encoded 'Authorization' header whose value consists of the word 'Basic' followed by a space followed by the Base64 encoded name:password. This is better suited for service-only access because the only way a user can log out is to shut down their browser. The URL with header information can be submitted with authentication information. Here's an example using cURL to access a SPARQLMotion service named ‘DisplaySimpleHtml’ in TopBraid:

```
curl -H "Authorization: Basic c2NvdHQ6MTIzNDU=" -X POST "http://localhost:
8080/tbl/tbl/sparqlmotion?id=DisplaySimpleHtml"
```

GET works as well. The base 64 in the middle translates to the uid:pwd string "scott:12345".

Some other examples of service calls with parameters (be sure to encode the URL before submitting as a web service call) :

```
curl -H "Authorization: Basic c2NvdHQ6MTIzNDU=" -X POST "http://localhost:
8080/tbl/tbl/sparql?default-graph-uri=http://topbraid.org/examples
/kennedys&format=application/sparql-results+json&query=SELECT ?s WHERE {?s
a <http://www.w3.org/2002/07/owl#Class>}"
```

```
curl -H "Authorization: Basic c2NvdHQ6MTIzNDU=" -X POST "http://localhost:
8080/tbl/tbl/sparql?query=SELECT ?s WHERE { GRAPH <http://topbraid.org
/examples/kennedys> {?s a <http://www.w3.org/2002/07/owl#Class>} }"
```

wget has a similar protocol:

```
wget --save-cookies="/tmp/cookies" --header "Content-type: application/x-
www-form-urlencoded" --post-data="j_username=scott&j_password=tomcat"
http://localhost:8080/tbl/j_security_check
```

For secure access, web services should use HTTPS encryption.

Form-Based Authentication

Access using Form-based authentication, while not recommended, is possible using cookies generated by the server. One method is to respond to the challenge with a hardcoded URL with a valid user id and password. The general form of this response is:

```
http://[host]:[port]/tbl/j_security_check?J_username=[username]&j_password=
[password]
```

Another method is to request an HTTP cookie that can be used in subsequent requests. The following is an example script for accessing a TopBraid form-based server. In summary, the script interaction has three parts:

1. The client requests a cookie, e.g., `loginRequestCookie`, to use for logging in using `GET` or `POST`.
2. The client logs in, and if successful, the client will receive another cookie, e.g., `loginSuccessCookie`, which is saved for subsequent requests.
3. The client uses the `loginSuccessCookie` for subsequent TopBraid service requests, such as the SPARQL endpoint call in the example.

Assuming the following is defined in a file named `authenticateCallService.sh`:

```
#!/bin/bash

#The script needs the servername:portnumber, username and password
respectively to work correctly.

if [[ $# -ne 3 ]] ; then
echo 'Invalid command. Please run authenticate script in following format .
/authenticateCallService <servername:port> <username> <password>'
    exit 1
fi

curl -c ~/loginRequestCookie -X POST "http://$1/tbl/tbl"   -D ~
/firstReqHeaders > /dev/null

#The server sends a new cookie as a response to this request. Use that
cookie for subsequent requests.
curl -b ~/loginRequestCookie -X POST "http://$1/tbl/tbl/j_security_check" -
H "Context-Type: application/x-www-form-urlencoded" --data
"j_username=$2&j_password=$3"  -L "http://$1/tbl/tbl" -c ~
/loginSuccessCookie -D ~/secReqHeaders
```

```
curl -b ~/loginSuccessCookie -X POST "http://$1/tbl/tbl/sparql" -G --data-
urlencode "query=SELECT * WHERE{?s a <http://www.w3.org/2002/07
/owl#Class>} "  --data-urlencode "default-graph-uri=http://topbraid.org
/examples/kennedys" --data "format=json" -D ~/thirdReqHeaders

#Logout when done so as not to exhaust the user limit on the license
curl -b ~/loginSuccessCookie -X POST "http://$1/tbl/tbl/purgeuser" -D ~
/logoutReqHeaders
```

The script can be invoked by the command:

```
./authenticateCallService <servername:port> <username> <password>
```

For secure access, web services should use HTTPS encryption.

Authenticating calls that use SPARQL's SERVICE keyword

To enable the use of the SPARQL SERVICE keyword to retrieve data from SPARQL endpoints that require authentication, add an entry to secure storage of the following form:

```
username@localhost:8080/tbl/tbl/sparql : password
```

For a query like the following, TopBraid TBL will check whether there is a user@uri key in secure storage that ends with the given URI, and returns the first one.

```
SELECT ?fname
WHERE {
        SERVICE <https://localhost:8080/tbl/tbl/sparql> {
                GRAPH <http://topbraid.org/examples/kennedys> {
                        ?subject <http://topbraid.org/examples
/kennedys#firstName> ?fname
                }
        }
}
```

This is used to generate the credentials for basic authentication.

See Password Management for more information about managing passwords in secure storage.

Role-Based Access Control (RBAC) in TopBraid

TopBraid does not directly handle user authentication, which is performed by an authentication layer, such as LDAP or Active Directory, the interacts with the web application container (Tomcat). Once a user has passed the authentication challenge from the web application, the user id and roles are cached in TopBraid. An important implication is that the TopBraid infrastructure cannot know about the existence of a user until they have successfully logged in to the web container.

Once a user has successfully passed authentication by the web container, user information is cached in a file named `/server.topbridlive.org/dynamic/users.ttl` in the TopBraid workspace. Useful SPARQL functions for access to user information accessible by TopBraid include the following:

- `smf:currentUserName`: Gets the name of the user that is currently logged into TopBraid. Should be preceded by `smf:hasCurrentUser` to avoid exceptions.
- `smf:userWithName`: Converts a user name into a URI resource, following the default settings in TopBraid. Often used in conjunction with `smf:currentUserName()`.

- `teamwork:currentUserHasPrivilege`: Checks whether the currently logged in user has a given privilege, specified by a role property. The current user must have that role or a sub-property thereof, for the given governed resource. The query will be executed on the given team graph.

Please see Help > TopBraid Composer > SPARQL Function Reference in TopBraid Composer for more information on these and other functions.

Note that there is no connection between TopBraid and LDAP/AD/other authentication systems. TopBraid caches some user information when an entity is authenticated that can be accessed by TopBraid solutions.