TINT Deployment Guide

John Chilton

December 16, 2009

Table of Contents

Prerequisites

Every configuration of TINT requires at least two pieces of software to be installed - a Java 5+ runtime and Apache Tomcat. An up-to-date Java runtime can be obtained here. The latest 5.5 branch of Apache Tomcat can be obtained here.

TINT requires more memory than Tomcat allows by default. Tomcat should be setup to increase the maximum amount of usable memory to at least 2 GB (or more if available). This can be done by setting the environment variable `CATALINA_OPTS` before starting Tomcat. For instance, to specify 2 GB of memory should be allowed `CATALINA_OPTS` should be set to `-Xmx2G`.

TINT is tested with Tomcat 5.5, but Tomcat 6 should also work for the Web GUI and web services (unless Globus secure transport is used.)

Depending on which components are being deployed additional prerequisites may be necessary. These will be covered as needed in this document.

Configuration Conventions

All of the files used to configure TINT are located in subdirectories of the directory `.tropix` beneath the home directory of the user running Tomcat. If Tomcat is running as the user `mary`, then this directory will likely be `C:\Users\mary\.tropix` on Windows systems and `/home/mary/.tropix` on Linux systems. If this directory is not manually created before TINT is started, then it will be created on start up.

The configuration directories for the various components of TINT will be referred to with titles of the form `tropix.name.config.dir` which corresponds to the directory `name` beneath the `.tropix` directory previously described. For instance, the directory that configures the Web GUI is referred to in this guide as `tropix.client.config.dir`. In the previous example this directory would be `C:\Users\mary\.tropix\client` on Windows systems and `/home/mary/.tropix/client` on Linux systems.

The various components of TINT discussed in this manual have different properties associated with them. These properties are generally read from a file named `deploy.properties` in the configuration directory corresponding to that component. For instance `login.timeout` is a property of the Web GUI component. To set the `login.timeout` property to `1000` one would create (or edit) a file named `deploy.properties` in the configuration directory `tropix.client.config.dir` and add the line `login.timeout=1000` to this file. These `deploy.properties` files are Java properties file. Each line should be of the form *`property=value`*.

Warning (Especially for Window's users): The backslash (\) character is an escape character in Java properties files. When a backslash should literally appear in a property value it must be escaped with another backslash. For instance, if one wanted to set the `storage.directory` property to `C:\foo\storage`, the line to add to `deploy.properties` would be `storage.directory=C:\\foo\\storage`.

Web GUI

Installation

From the TINT website download the file `tint.war`. While Tomcat is not running, place this file in the `webapps` directory of your Tomcat distribution.

Aside - The default URL for TINT will be `http://host:port/tint` (for instance `http://localhost:8080/tint`). If you wish to simplify this URL to `http://host:port/` delete the directory named `ROOT` from the `webapps` directory of the Tomcat distribution and rename `tint.war` to `ROOT.war`.

Warning About Firewalls - In addition to opening up a port in your firewall for the Tomcat container, the port 13139 should also be opened unless a TINT storage web service is going to be deployed. The default TINT deployment launches an embedded Jetty server that operates on port 13139 to handle file transfers.

Starting Tomcat now will start up TINT in its default state. In this state the TINT administrator may login to the site with the username `admin` and password `admin` and create other user accounts using the web interface. Users may upload files, create folders, and create wiki notes. Users may share these files with other users in shared folders and may preform searches.

If caGrid integration and/or access to proteomics tools is desired additional configuration is required.

Configuration

Common Configuration Properties

`modules`
: If set, this property should specify a comma separated list of the TINT modules to be activated. These modules are `SHARING`, `PROTIP`, `GENETIP`, `LOCAL_SEARCH`, `REQUEST`, `CATALOG`, `GRID_SEARCH`. Likely combinations to specify are as follows.

`modules=SHARING, LOCAL_SEARCH` - This is the default setting (i.e the property used if the `modules` property is not set in `deploy.properties`). Under this configuration, users can upload and share files, but no Protip (proteomics) services are enabled.

`modules=SHARING, LOCAL_SEARCH, PROTIP` - This will enable discovery and utilization of ProTIP services.

`modules=PROTIP` - This will enable discovery and utilization of Protip services, but turn off sharing and searching between users.

Information on other modules is beyond the scope of this document. If there is interest in the functionality they provide please contact the TINT developers.

`storage.directory`
If set, this property specifies where files uploaded to and created by TINT get placed. This defaults to the files directory beneath `tropix.storage.config.dir`.

This property is ignored if the TINT Web GUI is set to target a TINT Storage Service instead of managing files locally. For more information see `storage.type`.

`login.timeout`
If set, this specifies the amount of time in seconds a user can remain logged in before an automatic logout is triggered. This defaults to 43200 (12 hours).

Database Properties

Without any configuration, TINT will create many H2 databases in the TINT configuration directory for establishing the metadata store, persistent tracking of jobs (both client and service), etc.... H2 is a database engine that runs inside the Java virtual machine like other products such as HSQLDB or Apache Derby.

Performance is not critical for most of these databases the Java based default databases work just fine. The one exception to this is the metadata database (due to its complexity and frequency of use). It is recommend that this be configured to target a MySQL database. The following six parameters must be set when targetting a database other than the default H2 database. These properties should be set in the `deploy.properties` file of the configuration directory `tropix.metadata.config.dir`.

If the TINT Web GUI is going to be configured to target a Storage Service is going to deployed, an external database such as MySQL *must* be targetted so that both the Web GUI and the Storage Service can target the same database.

`metadata.db.url`
If set, this parameter should specify the JDBC URL of the database to target. The structure of a JDBC URL varies based on the database engine being targetted. For instance, if a MySQL database named `tintmetadata` is to be targetted where MySQL is running on port `3306` of host `localhost` then the JDBC URL would be `jdbc:mysql://localhost:3306/tintmetadata`.

`metadata.db.username`
Username used to access the database.

`metadata.db.password`
Password used to access the database.

`metadata.db.dialect`
If set, this specifies the Hibernate dialect corresponding to the database engine being targetted. For MySQL this should be `org.hibernate.dialect.MySQLInnoDBDialect`. Here is a list of dialects for other database engines.

`metadata.db.hbm2ddl`
: If set, this is the action performed by Hibernate at start up. Possible values are `validate`, `update`, `create`, and `create-drop`. For the typical MySQL setup this should just be set to `validate`.

`metadata.db.driver`
: This is the JDBC driver class Java loads to interact with the database. For MySQL this should be set to `com.mysql.jdbc.Driver`. For databases other than MySQL and H2, a jar file containing the driver class specified here should be placed in the `shared/lib` directory of Apache Tomcat so the class is available at runtime.

For a concrete example, if the user `mary` wanted to configure TINT to target a MySQL database named `tintmetadata` that is stored in a MySQL container running on host `localhost` under port `3306` and she was going to connect to that host using the username `mary` and password `pass123` she would create a file under her home directory with the path `.tropix/metadata/deploy.properties`. She would then populate this file with the following contents.

```
metadata.db.url=jdbc:mysql://localhost:3306/tintmetadata
metadata.db.username=mary
metadata.db.password=pass123
metadata.db.dialect=org.hibernate.dialect.MySQLInnoDBDialect
metadata.db.hbm2ddl=validate
metadata.db.driver=com.mysql.jdbc.Driver
```

When these database setttings are specified, the targetted database must be configured with the correct database structure. A MySQL script to execute to configure a database to hold TINT metadata can be found here. This script can potentially be modified to apply to other database engines.

caGrid Authentication

To allow TINT users to act as caGrid users TINT must be configured to use caGrid authentication mechanisms. This can be accomplished by creating and editing the file `authenticationSources.xml` in `tropix.client.config.dir`. The structure of this XML file is straight forward. The following XML is an example of such a files contents.

```
<authenticationSources
     xmlns="http://msi.umn.edu/tropix/client/authentication/config">
  <caGrid title="MSI Ldap"
          authenticationServiceUrl=
    "https://auth.msi.umn.edu:8443/wsrf/services/cagrid/AuthenticationService"
          dorianServiceUrl=
    "https://dorian.msi.umn.edu:8443/wsrf/services/cagrid/Dorian" />
</authenticationSources>
```

When a user attempts to login to TINT they will be presented with a drop down list of authentication sources, one for each caGrid element that appears between the outer `authenticationSources` element in this file. The `title` attribute specifies a simple description that the TINT user will see in the drop down box. `authenticationServiceUrl` and `dorianServiceUrl` should specify the URLs caGrid Authentication and Dorian services used for authentication and proxy creation. Dorian services are also authentication providers so both URLs may be set to the URL of a Dorian service.

TINT can operate a dual fashion that allows caGrid authentication sources as well as the default local authentication mechanism. This is generally discouraged because locally authenticated user will not have a grid credential, hence certain services may not work for him or her. To enable this add `>local /<` as another child element of the `authenticationSources` element.

Other caGrid Integration Properties

To set any of these remaining properties add them to the `deploy.properties` file of `tropix.client.config.dir`.

`hostname`
: If set, this should specify the hostname or IP address of the host machine. If a storage service is *not* configured an embedded Jetty server is used to transfer files two and from analytical services. For this to work properly TINT must know the hostname or IP address of the host for the Web GUI. Java has some guess of this IP address and this guess is used by default, but unfortunately on certain machines this address is configured incorrectly - instead evaluating to an IP address that is only valid internally such as `127.0.0.1`. Hence this should be specified unless a Storage Service is configured (`storage.type=cagrid`).

`index.service.url`
: To enable use of a caGrid Index Service this parameter should be set to the URL of that Index Service. A common value might be the caGrid training Index Service, this URL is `http://index.training.cagrid.org:8080/wsrf/services/DefaultIndexService`.

  The URL for a different target grid can be found in the `service_urls.properties` file for that grid. This file may be found in the `repository/caGrid/target_grid/[your chosen grid]` directory of your caGrid distribution.

`unindexed.service.urls`
: TINT maintains a list of services it adds to the list obtained from the Index service (if specified). These extra services can be specified as a comma separated list via this parameter.

`blocked.service.urls`
: If set, this property should be specified as a comma separated list of URLs. Services appearing in this list will *NOT* be displayed to the user even if they appear in the specified Index Service.

`proxy.lifetime`
: Lifetime in seconds of proxies requested from configured Dorian services. This defaults to 43200 (12 hours).

`storage.type`
: Set this property to `cagrid` to specify that files should be stored and transferred via a TINT Storage Service.

  Employing a Storage Service ensures that all data that is transferred to and from analytical services is done so over a Globus secured https protocol. This also unfortunately increases the complexity of a TINT deployment. A storage service URL must be specified and a Credential Delegation Service (CDS) must be configured so that analytical services can be permitted to act as the submitting user when transferring data. The CDS in turn requires that the Web GUI and any analytical service it interacts with be configured with host credentials. Finally, a list of these host identities for trusted analytical services must be configured. This last requirement severely limits the usefulness of the Index Service because the static nature of this list clashes with the otherwise useful dynamic nature of the Index service.

`storage.service.url`
: This property specifies the URL of the TINT Storage Service that should be used if the property `storage.type` is set to `cagrid`.

  If the TINT Storage Service is deployed using a WAR file as described in this document into a Tomcat container configured with Globus security on host `foo.edu` using the default port of 8443, this URL

would be `https://foo.edu:8443/caGridTransfer/services/cagrid/TropixStorage`.

`host.credential.cert`, `host.credential.key`
: Operations such as creating delegated credentials require the Web GUI to have a host credential. Such host credentials can be configured with this parameter. If set, these two parameters should specify the absolute path to the certificate and key of a host credential. Information on obtaining a host credential can be found here

`delegated.credential.lifetime`
: If set, this should specify the lifetime in seconds of a delegated credential. This defaults to 300 (5 minutes).

`cds.service.url`
: If set, this should specify the URL of a trusted Credential Delegation Service (CDS). If this is set, TINT will create a delegated credential reference before submitting jobs and fetching results that will allow the target analytical service to act the user. This is necessary if a TINT Storage Service is being targetted (`storage.type=cagrid`). A common value for this property might the URL for the CDS service deployed on the caGrid traning grid - this URL is `https://cds.training.cagrid.org:8443/wsrf/services/cagrid/CredentialDelegationService`. The URL for a particular target grid can be found in the `service_urls.properties` file for that grid as discussed previously [5].

`cds.allowed.parties`
: This should be set if `cds.service.url` is set. This should be a comma separated list of all of the grid identities that are allowed to receive delegated credentials. For instance if the Web GUI should create delegated credentials for analytical services that have grid identities of `/C=US/OU=MSI/CN=host/h1.edu` and `/C=US/OU=MSI/CN=host/h2.edu`, then this parameter should be set as follows:

  ```
  cds.allowed.parties=/C=US/OU=MSI/CN=host/h1.edu,/C=US/OU=MSI/CN=host/h2.edu
  ```

  Note: Sometimes grid identities appear with commas between the different parts of the identity (e.g. `C=US,OU=MSI,CN=host/h1.edu` instead of `/C=US/OU=MSI/CN=host/h1.edu`). The form with commas should not be used in this list. Any commas should be replaced with backslashes so the identity is in the same form as the examples shown above.

  The grid identity for a particular certificate can be found using the Globus tool `grid-cert-info` found in `bin` of the Globus distribution that is installed as part installing caGrid. To use this program simply execute from the commmand line with parameters `-file /path/to/cert`.

Proteomics Analytics (ProTIP)

There are currently 6 proteomics tools that may be integrated with TINT.

`Sequest`
: http://fields.scripps.edu/sequest/

`X! Tandem`
: http://www.beavisinformatics.com/TANDEM/

`OMSSA`
: http://pubchem.ncbi.nlm.nih.gov/omssa/

`Scaffold`
: http://www.proteomesoftware.com/

`iTraq Quantification`
: This is a tool developed at the University of Minnesota that performs iTraq quantification on Sequest runs post procecessed through Scaffold.

`RAW Extraction`
: This is an abstraction over two different tools for converting Thermo Finnigan RAW files into MzXML files. Either can be targetted using this component.

Installation

The various ProTIP services can be installed from WAR files. On the TINT website two WAR files can be found for each service, one suitable for deployment into a stock Tomcat container and one suitable for deployment into a Tomcat container configured with Globus security. The installation process for these services is similar to that of the TINT Web GUI. Obtain the WAR file corresponding to the TINT service you desire from the TINT website, drop that file in the `webapp` directory of your Tomcat container, configure the service, and finally start up Tomcat.

Configuration

Each service type has its own configuration directory. These are of the form `tropix.`*`servicetype`*`.config.dir`, where *`servicetype`* can be `sequest`, `xtandem`, `omssa`, `scaffold`, `rawextract`, or `itraqquantitation`. Note we use the term quantification in certain places and quantitation in others, sorry for this confusion, be careful to follow this manual carefully when either term is referenced.

The remainder of this section discusses various properties that may (or must) be set via the `deploy.properties` file in the respective configuration directory.

Shared Configuration

The properties in this subsection can be set for each service.

`advertise.container.hostname`
: If set, this should specify the hostname or IP address of the host machine. This is used when registering this service with the caGrid Index Service. Setting this property is not necessary if `advertise.perform=false`. Java has some guess of this IP address and this guess is used by default, but unfortunately on certain machines this address is configured incorrectly - instead evaluating to an IP address that is only valid internally such as `127.0.0.1`.

`advertise.index.service.urls`
: If set, this property should specify a comma separated list of the URLs of the caGrid Index Services to register this service with. By default, the caGrid training Index Service is used. This can be overridden with this parameter or disabled with `advertise.perform`.

`advertise.perform`
: Set this property to `false` to disable Index Service registration.

`advertise.container.port`
: If a grid service is being deployed into a Tomcat container configured to run on a port other than `8080` for normal Tomcat containers or 8443 for secured containers this property must specify that port. This corrects the URL that the service registers with the Index Services specified via `advertise.index.service.urls`.

`host.credential.cert`, `host.credential.key`
: If set, these two parameters must specify the fully qualified path to the certificate and key of the host credential to be used for this service. This host credential is needed resolve delegated creden-

tial references with a CDS. Setting these is necessary if clients to this service will be using the Storage Service to transfer data. The host identity of this credential should also be added to the `cds.allowed.parties` property of Web GUIs submitting jobs to this service.

`queue.staging.path`
Each job must be staged before execution. This involves creating a directory for the external program to run in and setting up various input files. This property controls the directory in which these staged directories will be created. This defaults to whatever the Java system property `java.io.tmpdir` resolves to, which will likely be `CATALINA_HOME/temp` (i.e. the temp directory of your Tomcat container).

`queue.staging.clean`
This can be set to `true` or `false` and controls whether staged files are deleted after they are no longer needed. This defaults to `false`, but it can be set to `true` if there are configuration problems that need to be debugged.

`service.name`
If set, this overrides the name of the service that gets publicized in the service metadata.

`service.researchcenter.path`
Part of the metadata exposed for caGrid services is a description of the research center hosting the service. TINT services leave this metadata empty by default but this behavior can be overridden with this parameter. To set this metadata copy the following XML fragment into a file, fill in the empty XML attributes, and set the `service.researchcenter.path` property to the fully qualified path of this newly created XML file.

```
<ResearchCenter displayName="" shortName=""
    xmlns="gme://caGrid.caBIG/1.0/gov.nih.nci.cagrid.metadata.common">
  <Address country="" locality="" postalCode=""
           stateProvince="" street1="" street2=""/>
  <pointOfContactCollection>
    <PointOfContact affiliation="" email="" firstName=""
                    lastName="" phoneNumber="" role=""/>
  </pointOfContactCollection>
</ResearchCenter>
```

Sequest Configuration

`sequest.path`
This must be set. It should be set to the fully qualified path of a Sequest executable.

`sequest.output.type`
If set, this parameter controls how Sequest logs are parsed to determine job progress. This defaults to `STANDARD_OUTPUT`, which corresponds to the output Sequest 28, if the PVM version of Sequest 27 is used this parameter should be set to `PVM_OUTPUT`.

X! Tandem Configuration

X! Tandem is distributed as part of the TINT distribution and will be configured automatically. These parameters are only necessary if you want to point TINT at a manually installed version of X! Tandem.

Ubuntu Linux Users: Due to library incompatibilities the version of X! Tandem that is bundled with ProTIP will not work out of the box on Ubuntu Linux systems. X! Tandem can be manually installed and targetted with the `xtandem.path` parameter mentioned below. The latest version of X! Tandem for various operating systems can be downloaded here. Alternatively, the version of X! Tandem that is bun-

dled with ProTIP will work in Ubuntu if a a symbolic link is created via the following command `sudo ln -s /usr/lib/libexpat.so /usr/lib/libexpat.so.0` and `libstdc++` is installed. A `.deb` package for `libstdc++5` can be found here. `.deb` files can be installed via the command `sudo dpkg -i /path/to/file`.

`xtandem.path`
: If set, this should be set to the fully qualified path of an X! Tandem executable.

`xtandem.xsl.path`
: If set, this should be set to the fully qualified path of the XSL file to apply to X! Tandem output files. Scaffold requires that the XSL file to use is the one named `tandem-style.xsl` in the `bin` directory of the X! Tandem distribution.

OMSSA Configuration

Like X! Tandem, OMSSA is distributed with TINT and will be installed automatically. Because OMSSA does not accept FASTA files directly, TINT targets a script that wraps OMSSA and the BLAST tool `formatdb`. `formatdb` is required to convert FASTA files into a format consumable by OMSSA.

There are three parameters involved in the above process that can be overridden. The wrapper script is automatically created if `omssa.path` is not set. If `omssa.path` is not set and the script is created, it will attempt to read `omssa.home` and `blast.home`. These should point to the directory which contains the omssa and blast distributions respectively. Both of these parameters are also optional and these programs will be automatically installed for your platform if needed.

`omssa.path`
: If set, this should be set to the fully qualified path of the wrapper script for OMSSA and BLAST.

`omssa.home`
: If set, this should be set to the fully qualified path of an OMSSA installation.

`blast.home`
: If set, this should be set to the fully qualified path of a BLAST installation.

Scaffold Configuration

To integrate Scaffold with TINT, you will need to purchase a license and install ScaffoldBatch from Proteome Software. Information related to purchasing ScaffoldBatch can be found here.

Be sure that ScaffoldBatch has been activated with a working key before TINT is started. This can be done by starting ScaffoldBatch from the command-line without any parameters. You will then be prompted for a license key.

`scaffold.path`
: This must be set. This should be set to the fully qualified path of a Scaffold Batch executable.

RawExtract Configuration

This component can be used to wrap external programs that convert Thermo Finnigan .RAW files into MzXML files. Currently two tools are supported - ReAdW and msconvert. Both of these are part of the Transproteomics Pipeline (TPP).

`rawextract.path`
: If set, this should be set to the fully qualified path of either the `msconvert.exe` or `ReAdW.exe` executable.

If this parameter is not set and TINT is running in a Windows environment, `ReAdW.exe` will be installed and targetted automatically. This requires that Thermo's Xcalibur software has been previously installed and the DLL file `XRawfile2.dll` is available to `ReAdW.exe`. For more information see this page.

If this parameter is not set and TINT is running in a Linux environment, `ReAdW.exe` will be extracted into the directory `tropix.rawextract.config.dir` as well as a script that uses Wine to call `ReAdW.exe`. For this script to work, Wine should be installed and Thermo's Xcalibur software should be installed inside of Wine.

The process of installing Xcalibur under Wine is likely going to vary based on the version of Wine and Xcalibur being used. At MSI, we were able to install Xcalibur 2.0.7 with the stock version of wine for Ubuntu 9.10. To install this simply install wine (`sudo apt-get install wine`), navigate to the `Xcalibur 2.0.X` directory of the Xcalibur CDROM (`cd /media/cdrom/Xcalibur\ 2.0.7`), launch `setup.exe` (`wine setup.exe`) and select all of the default options in the graphical installer that is launched.

ITraq Quantification Configuration

No additional configuration is necessary.

Storage Service

Installation

Unlike the other WAR files discussed so far the WAR file for the Storage Service may only be deployed into a Tomcat container with Globus security configured. The easiest way to install and configure such a container is to use the caGrid installer. TINT currently supports only caGrid 1.2, information on installing caGrid 1.2 can be found here.

Once the a secure container has been configured, the Storage Service may be installed via the WAR file `caGridTransfer.war`, available on the TINT website. Once this WAR files has been downloaded, the Storage Service can be installed just like all of the other TINT components. Move the WAR file into the `webapp` directory of the destination Tomcat container, configure the service, and start up Tomcat.

Unlike the other TINT WAR files, `caGridTransfer.war` must not be renamed. The storage service currently can only be installed in the `caGridTransfer` webapp directory of its Tomcat container.

Configuration

The Storage Service has only a few interesting configuration parameters. It shares these parameters with the TINT Web GUI.

A metadata database must be configured. This should be configured in the `deploy.properties` file of `tropix.metadata.config.dir` just like the Web GUI. The properties to be place in this file were previously documented here.

If a `deploy.properties` file exists in `tropix.storage.config.dir` the storage service attempts to read the property `storage.directory`. This property and its default value is documented here. Note that the Storage Service reads this property from the directory `tropix.storage.config.dir` instead of `tropix.client.config.dir` like the Web GUI uses.

A Final Note

Additional components are available for TINT that are beyond the scope of this document. These include components for searching and downloading data over caGrid, advertising and requesting lab services over caGrid, and integration with Globus GRAM services. The deployment process for these services is a bit rougher and the applicability more narrow, but if you are interested in deploying these components please contact the Tropix developers for more information.