Table of Contents

TriggerTool Development

General Info (mailing list, meetings, savannah, SVN, LS1, ...)

This page contains information for developers of the TriggerTool.

- Mailing list
  - Developers should sign up to the mailing list: atlas-triggertool-dev join archive. This is used to discus TriggerTool related issues, arrange meetings and recieves bug reports.
- Meetings
  - Developer meetings can be found in indico here (scheduled usually at 5pm on Monday, vidyo pin 2001).
  - UCL workshop: agenda coreSW summary
- Savannah
  - Any bugs relating to the trigger tool can be found here
- SVN
  - The code is split into two areas in svn:
    - GUI
    - Database
- Upgrades
  - Details of the LS1 upgrades can be found here: TriggerToolUpgrade

Stable P1 environment rules

Operations at P1 are picking up again. It is of the highest priority that we always

- have a working TriggerTool and DB at P1
- are able to fix any urgent problem or critical feature request right away

This implies that we always have a version in svn that corresponds to the version installed at P1. Therefore the new modus operandi should be as follows:

- Every change that impacts the operations version of the trigger tool has to be tested carefully for offline (MC, ATN, REPR) databases and online (ATONR) databases and its replica (ATLR)

- New versions of the TriggerTool/TrigDB need to go into the nightly releases so they are part of the ATN tests. As soon as a tag is created it should be requested in TagCollector for the nightlies.

- Developments of new features or larger bug fixes have to be moved to branches again and merged back into the trunk only when ready. For this ATLAS invented the devbranches, that don't have to follow the official naming policy. Please use them for any larger possibly unstable on the short term changes.

- Never commit anything to the svn trunk until you are completely convinced your change works, it does what it is supposed to do and doesn't break anything else. If you are not sure, it is better to use a branch.

- When something is done, please add an entry to ChangeLog and tag. Let's tag rather often, each fix/feature should get a new tag.

- Installation at P1 and on afs should happen with the svn tag number included and by setting soft links. That allows us to switch back in case something went wrong.

Getting started

- The TriggerTool code is easiest to develop and test within the netbeans IDE (the "Java SE" version is good enough for developing the TriggerTool).
    - to be able to install this locally on your machine you will need to have a JDK installed download here
    - details for running netbeans/Java on lxplus or working from remote
- Setup the view using:

```
WINDOW -> projects and WINDOW -> tasks
```

- You then need to check out the TriggerTool from SVN

```
TEAM -> subversion -> checkout
first get the TrigDb: svn+ssh://svn.cern.ch/reps/atlasoff/Trigger/TrigConfiguration/TrigDb
then the TriggerTool: svn+ssh://svn.cern.ch/reps/atlasoff/Trigger/TrigConfiguration/Trigge
```

- load the projects already configured for Netbeans. These are defined in the base folder of the svn repositories:

```
FILE -> Open Project and select TrigDb and again for TriggerTool.
```

    - Should you get errors of missing libraries from the above steps, follow:

```
projects -> triggertool (right click) -> set config -> customise -> libraries -> add
when done also delete the incorrect JAR
```

- to get the configuration to work on your computer you need to change the configuration
- don't edit an existing one rather create your own, which you can check into SVN if needed

```
projects -> triggertool (right click) -> Set Config -> Configuration -> working directory
on the far right next to the name dropdown box should be a button marked new
set your working directory to: /path/to/triggerTool/trunk/folder
and vm options: -DCORAL_AUTH_PATH="/path/to/TrigDB/trunk/folder"  -DCORAL_DBLOOKUP_PATH="/
```

- Now you should be able to run the TriggerTool
    - First select the `Main` configuration from the configurations drop down menu (see picture).
- To **run the TriggerTool** click on the green *play* button .
- To run in **debug** mode click on the debug button .

Request a DB and create the structure

- To be able to test your code you will need your own cern database:
  - go to: https://resources.web.cern.ch/resources/Manage/Oracle/Subscribe.aspx
  - subscribe to: oracle-general-purpose-users
  - once you are added to the egroup (you get an email) go to: https://resources.web.cern.ch/resources/Manage/Oracle/NewOracleAccount.aspx
  - choose a login (example: your afs username), database=devdb11 and description (example: db testing for the atlas trigger tool)
  - again you will get an email when created, change the password (pick something simple as you will be typing it on the command line)
- Now you are ready to create the DB structure:
  - To check the db exists, run this with the username/password replaced
  - `sqlplus username/password@devdb11`

    ◊ If you get to an sql prompt your DB exists 🙂
  - To now setup the structure you on lxplus you should check out trigDB:
  - Note do not setup an Atlas release first as this will give you the wrong schema!

```
svn co -r HEAD svn+ssh://svn.cern.ch/reps/atlasoff/Trigger/TrigConfiguration/TrigDb/
cd TrigDb/share/
./DBstartup -t oracle -u username -m devdb11 -p password
```

    ◊ Hopefully you see the lines `Table created` and `Commit complete` for each row. The only errors should be in drop table (as below) as nothing exists yet, **report if you get other errors**

```
DROP TABLE DBCOPY_SOURCE_DATABASE PURGE
           *
ERROR at line 1:
ORA-00942: table or view does not exist
```

- Now you can see the structure in the DB by doing:

```
rlwrap sqlplus username/password@devdb11
select table_name from user_tables;
describe l1_random;
```

  - Note that rlwrap is just an extra wrapper to help you use sqlplus (means you can use delete and the arrow keys as if its a normal command prompt)

- To reset the db use:
  - Prelim instructions:
  - Use the above DBstartup command
  - If nothing has changed in the schema then you should see no errors
  - If the schema has changed then you will get errors in certain tables, these tables can be removed by doing:

```
DROP TABLE insert_table_name CASCADE CONSTRAINTS ;
```

    ◊ whereas the DBstartup command normally does:

```
DROP TABLE insert_table_name PURGE;
```

Upload a menu

- Once you have the TriggerTool running and have your own DB you can try to upload a menu
- in your .triggertool file (normally in your home area) create the line for your DB:

```
Oracle,devdb11,insert_dbname,insert_username,insert_password,
```

- Now run the TriggerTool and follow:

```
Databases -> Oracle -> select your DB
```

- this should log you onto your DB
- note currently you have to have the pw in your .triggertool file
- Here you can again check your DB schema:

```
File -> Check Schema
```

- To upload a menu:

```
Load/Save -> Read XML
Click Browse on a menu xml file
Navigate to TrigDB -> XML
Choose the appropriate xml and fill in the remaining details (should be listed in menu fil
```

- more details... * If you have problems of getting a TriggerTool write lock while uploading (e.g. if you lose network connection) you can remove it in the TriggerTool. **Only perform this operation if you are certain that the write lock is in place by your lost connection. It could be in place due to another expert uploading which you will see as the user will be listed on the panel.** To see who has the write lock select View->Write lock. **Please contact the TriggerTool experts if you are uncertain if the lock is from you** as **removing this lock could harm your uploads or those from other experts**. If you are certain you can remove the lock by clicking **Force Unlock**.

Code development

For code that will interfere with other peoples work on the DB/GUI please create a branch to do your development. To do this follow:

- edit the ChangeLog in the latest trunk version to state which tag you are about to create

```
Tag as __TriggerTool_or_TrigDb__-XX-XX-XX to create branch point for YYYY developments
```

- now commit the ChangeLog

```
svn ci -m "Tag as __TriggerTool_or_TrigDb__-XX-XX-XX to create branch point for YYYY deve
```

- Using the outputted revision number then create the tag:

```
svn cp svn+ssh://svn.cern.ch/reps/atlasoff/Trigger/TrigConfiguration/__TriggerTool_or_Tri
```

- Now use this tag to create a branch:

```
svn cp svn+ssh://svn.cern.ch/reps/atlasoff/Trigger/TrigConfiguration/__TriggerTool_or_Tri
```

- Finally switch your version to this latest branch:

```
svn switch svn+ssh://svn.cern.ch/reps/atlasoff/Trigger/TrigConfiguration//__TriggerTool_o
```

    - Note this can also be done in netbeans by doing Subversion -> Copy -> Switch to Copy

Once the developments are finished in the branch you merge by following after checking your last changes into the branch:

- Switch your branch to the trunk

```
svn switch svn+ssh://svn.cern.ch/reps/atlasoff/Trigger/TrigConfiguration/__TriggerTool_or_
```

- Check the differences between your branch and the tag

```
svn diff svn+ssh://svn.cern.ch/reps/atlasoff/Trigger/TrigConfiguration/__TriggerTool_or_Tr
```

- Merge in these changes to your local trunk version

```
svn merge svn+ssh://svn.cern.ch/reps/atlasoff/Trigger/TrigConfiguration/__TriggerTool_or_T
```

- Check the resulting differences (svn diff)
- Next update the changelog with the details of what changed in your branch
- Finally check in the changes and create a new tag

```
svn cp svn+ssh://svn.cern.ch/reps/atlasoff/Trigger/TrigConfiguration/TrigDb/trunk/ -r new_
```

```
svn cp svn+ssh://svn.cern.ch/reps/atlasoff/Trigger/TrigConfiguration/__TriggerTool_or_Tri
```

Unit tests

- Unit tests should be run in NetBeans at critical points in development and also to inform or drive development.

- Before starting work, run the entire test suite:
    - Right-click on the project (TriggerTool or TrigDb) and click 'Test'.

    - The test output should look something like this:

    - Make sure the entire test suite passes before starting work.

- Tests should be maintained and written as part of development.
    - Check the classes and methods you are working on have test coverage. Look for the class' tests in 'Test Packages > package > ClassTest.java' and check that the test file covers the code you are working on.
        - To make a new test class for a completely uncovered class, right-click on the file in the navigator and click 'Tools > Create/Update Tests'. Leave the textboxes as their defaults and checkboxes as shown:

    - Add or modify tests as appropriate to ensure the class methods under development behave as expected. It may be useful to share instances between test methods or define helper functions. It might further be useful to refactor the class being tested in order to make it more easily testable.
    - A single test method or file can be run, or just the tests for a particular package. Run these tests often during development to check new or modified code.

- Before committing changes to the repository, run the entire test suite for the project. Make sure all tests pass before committing.

Building TriggerTool and using CMake

The following guide is based on more complete documentation for CMake in atlas at:

https://twiki.cern.ch/twiki/bin/view/AtlasComputing/CMakeTestProjectInstructions
https://twiki.cern.ch/twiki/bin/view/AtlasComputing/SoftwareDevelopmentWorkBookCMakeInAtlas

To build (tested on lxplus) do the following:

- Create working directory
- Within this directory create three subdirectories: 'source', 'build' and 'run'
- From within 'source' directory set up the release (currently this must be dev/devval). For example: asetup 21.X.Y-VAL,here,rel_5 (or most recently nightly)
- Check out packages with 'svnco', which is like pkgco.py but doesn't attach all the CMT stuff.

```
svnco TrigDb
svnco TriggerTool
```

- By default this gives you the trunk, checkout with full tag/branch name for other versions (svnco TriggerTool-XX-XX-XX)
- Go to build directory
    - From there run:

```
cmake ../source
```

    - Once this is complete run the rest of the build with:

```
make -j4
```

- Now that cmake has run once you don't have to re-run it for code changes, you just re-run the make itself.

ATN tests

- Before submitting a tag, various ATN tests should be run and confirmed that they pass in the nightly environment.
  - ♦ Setup the release environment and checkout the packages:

```
asetup AtlasCAFHLT,20.11.X.Y.Z-VAL,rel_4,here
pkgco.py -A TrigDb
pkgco.py -A TriggerTool
```

    - ◊ (use pkgco.py TriggerTool-04-XX-YY for a particular tag)
  - ♦ Compile with CMT

```
cd Trigger/TrigConfiguration/TriggerTool/cmt
cmt br make
cd ../../../../
```

  - ♦ Run our ATN tests

```
/afs/cern.ch/atlas/software/dist/nightlies/atn/atn Trigger/TrigConfiguration/Trigger
```

    - ◊ The tests should run in 5-10 mins and place output in NICOS_area/NICOS_TestLogWORK_RELEASE/Trigger_TrigConfiguration_TriggerTool__
  - ♦ Run other (trigger?) ATN tests

```
trigtest.pl --cleardir --test HLT_physicsV6_menu --rundir HLT_physicsV6_menu --conf
trigtest.pl --cleardir --test HLT_physicsV6_rerun --rundir HLT_physicsV6_rerun --con
trigtest.pl --cleardir --test CheckKeysV6 --rundir CheckKeysV6 --conf TrigP1Test.con
```

How to work remotely

- The above description works for those based at cern, for those working remotely you will need to use nomachine, or...

Tips for developers

- If you are developing a project that will take some time, best practice is to work in a branch
  - First create a tag from the trunk
  - Use this to then create the branch
  - Once you are done merge back into the trunk using:
- The TriggerTool-03-XX-YY tags are to be used as a legacy version for Run1 data

Running version from lxplus for run 2

To be able to run the latest version of the trigger tool directly on lxplus (affter checking out and making) you should set:

```
export TRIGGER_EXP_CORAL_PATH=~/<path to authentication XML files>
export USERTTPATH=~/<your triggertool working area>/InstallArea/share/lib/
export USERLOGDIR="$PWD/TriggerToolLogs/" # or update to the folder of your choice
export USERTTJAR="TriggerTool.jar" #leave as is
```

create dblookup.xml:

```
<?xml version="1.0" ?>
<servicelist>

<logicalservice name="TRIGGERDB_MARK">
   <service name="oracle://devdb11/mark"
accessMode="update" authentication="password"/>
</logicalservice>

</servicelist>
```

and create authentication.xml:

```
<?xml version="1.0"?>
<connectionlist>

<!-- THE ONLINE TriggerDB -->
<connection name="oracle://devdb11/mark">
  <parameter name="user" value="mark"/>
  <parameter name="password" value="password"/>
</connection>
</connectionlist>
```

now you will be able to run using:

```
<your triggertool working area>/Trigger/TrigConfiguration/TriggerTool/scripts/run_TriggerTool_Run
```

Running run 2 version at point 1

* ssh -XY atlasgw

* pick a computer:

    - trigger desk is: pc-atlas-cr-35 or pc-atlas-cr-trg
    - pub machine: pc-atlas-pub-01

* trigger tool is here: /det/tdaq/hlt/trigconf/TriggerTool/Run2

    - built from: /afs/cern.ch/user/a/attrgcnf/TriggerTool/build
    - (using build_Run2.sh)

* script to run TT is: /det/tdaq/scripts/start_trigger_tool_interactive_run2

TMC installation

For the LevelOneCentralTriggerMenuTest it is required to setup the L1CT release to get the TriggerMenuCompiler.

- on lxplus:
  - setup the current L1CT release: `source /afs/cern.ch/atlas/project/tdaq/level1/ctp/setup/setup-l1ct.sh`
  - start the TriggerTool: `/afs/cern.ch/user/a/attrgcnf/TriggerTool/run_TriggerTool_CTPExperts.sh`
- at P1:
  - setup the current L1CT release: `source /det/ctp/setup/setup-l1ct_cmake.sh`
  - start the TriggerTool: `/det/tdaq/scripts/start_trigger_tool`

Previous version of twiki, to be merged above

Below is the previous version of this documentation. This page is currently being updated as part of the LS1 activities as documented on the twiki: TriggerToolUpgrade

Expand Previous: Hide Previous

How to Compile

using `ant`

To compile using ant, you need to cd to the ant dir `cd ant` and run the ant compile and build task `ant dist -DDBSchemaBaseDir=/Path/To/TrigDb/.`

Example: Hide Example
Example:

```
atlaslap05:${TriggerTool} $ cd ant
atlaslap05:${TriggerTool}ant $ ant dist -DDBSchemaBaseDir=/Path/To/TrigDb/
Buildfile: /Users/tiago/atlas/TriggerTool/TriggerTool-02-02-47/ant/build.xml
     [echo] Directory for TriggerDB schema /Users/tiago/atlas/TriggerTool/TrigDb/
     [echo] Path /Users/tiago/atlas/TriggerTool/TriggerTool-02-02-47/ant
     [echo] Java ${env.JAVA_HOME}

init:
     [exec] Setting TriggerTool version to: TriggerTool-02-02-47
[...]
dist:
      [jar] Building jar: /Users/tiago/atlas/TriggerTool/TriggerTool-02-02-47/lib/TriggerTool.jar
[...]
BUILD SUCCESSFUL
Total time: 21 seconds
```

using `cmt`

Todo...

Test the TriggerTool / run JUnit

Netbeans

1. Open class `triggertool.junit.BasicTestSetUp`. Either click on "Test Packages -> triggertool.test -> BasicTestSetup.java or press "ctrl+shift+o" and type "BasicTestSetup"
2. Check the password for the replica DB in `BasicTestSetUp.connectToORACLEonline()`. Eventually you'll need to set to correct pwd here.
3. Open the class `triggertool.junit.TriggerToolTestSuite`
4. Select the configuration `JUnitTest`
5. Right click on `TriggerToolTestSuite` and select "Test File".
6. After 2 minutes you should see something like in the picture below. If you see anything red means that some of the tests failed. If so you'll need to check which one failed and why.

Command Line: `ant + JUnit`

There is a `ant` target defined, which will generate and run the JUnit tests. To use it do:

1. Change to the ant directory: `${TriggerTool}> cd ant`
2. Compile the TriggerTool: `${TriggerTool}/ant> ant dist -DDBSchemaBaseDir=/Path/To/TrigDb/`
3. Run the JUnit test: `${TriggerTool}/ant> ant test`

The successful output of looks like

Show Output Hide Output

```
atlaslap05:ant tiago$ ant test
Buildfile: /Users/tiago/atlas/TriggerTool/TT/ant/build.xml
     [echo] Directory for TriggerDB schema ${DBSchemaBaseDir}
     [echo] Path /Users/tiago/atlas/TriggerTool/TT/ant
     [echo] Java ${env.JAVA_HOME}

maketest:
   [delete] Deleting directory /Users/tiago/atlas/TriggerTool/TT/build
    [mkdir] Created dir: /Users/tiago/atlas/TriggerTool/TT/build
    [unzip] Expanding: /Users/tiago/atlas/TriggerTool/TT/external/junit-4.8.2.jar into /Users/tia
    [javac] /Users/tiago/atlas/TriggerTool/TT/ant/build.xml:42: warning: 'includeantruntime' was
    [javac] Compiling 6 source files to /Users/tiago/atlas/TriggerTool/TT/build
      [jar] Building jar: /Users/tiago/atlas/TriggerTool/TT/lib/TriggerToolTester.jar
     [copy] Copying 1 file to /Users/tiago/atlas/TriggerTool/TT/scripts
     [copy] Copying 1 file to /Users/tiago/atlas/TriggerTool/TT/scripts

test:
     [echo] Start JUnit testing of TriggerTool.
    [junit] Testsuite: triggertool.junit.TriggerToolTestSuite
    [junit] Tests run: 8, Failures: 0, Errors: 0, Time elapsed: 157.763 sec
    [junit] ------------- Standard Error -----------------
    [junit] Mar 24, 2011 11:37:33 AM triggertool.Connections.ConnectionManager connect
    [junit] INFO:  Connecting to jdbc:oracle:thin:@atlr1-v.cern.ch:10121:atlr1
    [junit]  System user
    [junit]  DB username atlas_conf_trigger_v2_r
    [junit]  Role        User
    [junit]  online   = false
    [junit]  onlineDB = false
    [junit] Mar 24, 2011 11:37:33 AM triggertool.Connections.ConnectionManager connect
    [junit] FINE: jdbc:oracle:thin:@atlr1-v.cern.ch:10121:atlr1
    [junit] Mar 24, 2011 11:37:33 AM triggertool.Connections.ConnectionManager connect
    [junit] FINE: atlas_conf_trigger_v2_r
    [junit] Mar 24, 2011 11:37:34 AM triggertool.Connections.ConnectionManager isDBVersionCompati
    [junit] FINE: TriggerDB version combatibility: true
    [junit] Mar 24, 2011 11:37:34 AM triggertool.junit.TriggerToolTestSuite setUpClass
    [junit] INFO:  CONTROL PATH ../scripts/
    [junit] Mar 24, 2011 11:37:38 AM triggertool.HLTRecords.HLTTriggerMenu getAllAlgorithms
    [junit] INFO: getAllAlgorithms query = SELECT HCP_ALIAS, HCP_ID, HCP_MODIFIED_TIME, HCP_NAME,
    [junit] Mar 24, 2011 11:37:38 AM triggertool.HLTRecords.HLTTriggerMenu getAllAlgorithms
    [junit] FINE:  Execute Query T = 174.375ms
    [junit]
    [junit] Mar 24, 2011 11:40:11 AM triggertool.HLTRecords.HLTTriggerMenu getAllAlgorithms
    [junit] FINE:  Execute While = 152.751181s
    [junit]  Iters = 490
    [junit]  tIter = 311737.104us
    [junit]
    [junit] Mar 24, 2011 11:40:11 AM triggertool.junit.HLTTriggerMenuTest testMenuLoadAgorithms
    [junit] INFO:  Testing TriggerTool ver trunk Thu Mar 24 11:36:31 CET 2011 agaist control file
    [junit]  Time to load all Algos  : 153.240592sec
    [junit]  SIZE=1698
    [junit]
    [junit] ------------- ---------------- ---------------
```

```
    [junit]
    [junit] Testcase: testGetPrescales took 2.539 sec
    [junit] Testcase: testControlFile took 0.001 sec
    [junit] Testcase: testPrescales took 0.03 sec
    [junit] Testcase: testDirectQuery took 0.405 sec
    [junit] Testcase: testControlFile took 0 sec
    [junit] Testcase: testIsConnected took 0 sec
    [junit] Testcase: testIsMenuOk took 0.14 sec
    [junit] Testcase: testMenuLoadAgorithms took 153.305 sec

BUILD SUCCESSFUL
Total time: 2 minutes 42 seconds
```

If you get something like `[junit]  FAILED` or `BUILD FAILED`, then something went wrong with the test and you'll need to investigate why :(.

Major updates: -- AdamJamesBozson - 18-Apr-2017 -- MarkStockton - 19-May-2014 -- TiagoPerez - 24-Mar-2011

%RESPONSIBLE% WillPanduroVazquez
%REVIEW% **Never reviewed**

This topic: Sandbox > TriggerToolDevelSandbox
Topic revision: r1 - 2017-04-18 - AdamJamesBozson