Totem Unified Database Access Service

System Prototype

Introduction

This prototype is introduced in order to familiarize with main concepts of using TUDAS system. We provide server application which is constantly working on pctotem34 machine, ready to handle client requests, as well as client-side libraries with simple examples written in all three supported languages (java, c++, python). In this document we would like to guide you through these client-side prototypes, showing how to install and run them. In the tutorial we use luminosity client example, but in this release there are two more examples : roman pot data management client and saving run information client. Steps to run these examples are similar to those from the tutorial.

Detailed description of prototype interfaces can be found in the documents :
https://twiki.cern.ch/twiki/pub/TOTEM/CompDBSoftware/RomanPotManagementInterfaces.pdf

https://twiki.cern.ch/twiki/pub/TOTEM/CompDBSoftware/LuminosityInterfaces.pdf

https://twiki.cern.ch/twiki/pub/TOTEM/CompDBSoftware/TudasDBPopInterfaces.pdf

Installing TUDAS

All you need to do is to checkout project from svn tag:

```
$ svn co svn+ssh://svn.cern.ch/reps/totem/trunk/tudas
```

Downloaded package includes all source code of our project. Essential part containing ready to use libraries and examples is located in directory:

```
$ ./tudas/release/
```

Here you can find client-side modules grouped in language specific directories.

Running Java example

1. Open your terminal and navigate to `java` directory:
   ```
   $ cd  ./tudas/release/java
   ```
2. Compile Example`LuminosityClient.java` attaching `tudas.jar` library:
   ```
   $ javac -cp ./lib/tudas.jar ExampleLuminosityClient.java
   ```
3. Run compiled class:
   ```
   $ java -cp .:./lib/tudas.jar ExampleLuminosityClient
   ```

Running python example

1. Open your terminal and navigate to directory that contains TUDAS. Then, naviagte to `python` directory:

```
$ cd  ./tudas/release/python
```

2. Update LD_LIBRARY_PATH variable:

```
$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$PWD/lib/ice
```

3. Update PYTHONPATH variable:

```
$ export PYTHONPATH=$PYTHOPATH:$PWD/lib/tudas:$PWD/lib/ice
```

4. Run example, i.e. example_luminosity_client:

```
$ python example_luminosity_client.py
```

Running c++ example

NOTE: All TUDAS c++ libraries were compiled on Scientific Linux CERN 5. If you would like to use another distribution to test our prototype, please contact us.

1. Open your terminal and navigate to python directory:
   ```
   $ cd  ./tudas/release/c++
   ```
2. Update LD_LIBRARY_PATH variable:
   ```
   $ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$PWD/lib
   ```
3. Compile i.e. example_luminosity_client.cpp:
   ```
   $ g++ -I./include -L./lib -ltudas ExampleLuminosityClient.cpp -
   o ExampleLuminosityClient
   ```
4. Run example_client:
   ```
   $ ./ExampleLuminosityClient
   ```

Using CMSSW module

In order to compile CMSSW with access to database, one should do following steps:

1. Create new CMSSW project workspace :

```
$ export RFIO_USE_CASTOR_V2=YES
$ export STAGE_HOST=castorpublic
$ export STAGE_SVCCLASS=default
$ export SCRAM_ARCH=slc5_amd64_gcc434
$ source /afs/cern.ch/cms/cmsset_default.sh
$ scram project CMSSW CMSSW_4_2_4
```

2. Check out version of CMSSW with Tudas from SVN branch:

```
$ svn co svn+ssh://svn.cern.ch/reps/totem/branches/
CMS_4_2_4_with_tudas CMSSW_4_2_4/src/
```

3. Install tool for database library:

```
$ cp /afs/cern.ch/exp/totem/soft/database/ice/tool/ice.xml
CMSSW_4_2_4/config/toolbox/slc5_amd64_gcc434/tools/selected
$ cd CMSSW_4_2_4/src/
$ eval `scram runtime -sh`
$ scram setup ice
```

4. Compile CMSSW framework.

```
$ scram b -j 4
```

If an error occurs, run following command until everything is OK (to exit, press Ctrl-C).

```
$ edmPluginRefresh
```

Finally:

```
$ scram b -j 4
$ scram b
```

5. Usage of example Producer:

```
$ cmsRun TotemUnifiedDatabaseAccessService/RPAlignmentESSource/test/test_cfg.py
```

Exceptions handling

With this release we provide exception handling mechanism which should be useful in case of potential problems with connection to server/database or some configuration errors. Each exception contains easy-to-understand error message in following format:

```
*** Totem Database Access Service Error ***

Information: methodThatFailed : Information about what happend
Reason: Why the problem appeared
Solution: What should I do to resolve the problem
****************************************
```

Important note: currently most of typical exceptions are handled, but sometimes (i. e. when there is a problem on database side) you might obtain “the unknown exception” with following message:

```
*** Totem Database Access Service Error ***

Information: methodThatFailed: The exception was thrown
Solution: Find more information about the exception in log file
****************************************
```

If you use Java client, there should be ***log*** directory with client logs, appended to file ***client.log***.
The detailed information about exception is stored here (and it is mainly Java stack trace). If you experience any strange problems, please send us this file.
In case of python and c++ clients, there is also the same information about unknown exceptions, but, for now, logs are not stored.