PRINCESS Challenge Problem Toolkit

Michael Reposa, Steve Marotta

27-MAR-2018

Version 1, Phase 2 (v1.2.x)

1 Overview

The PRINCESS Challenge Problem Toolkit (CPT) is a set of applications and libraries supporting PRINCESS challenge problem solution development, testing, and the evaluation by MIT Lincoln Labs. The CPT consists of:

1) Evaluation Manager Java Library
2) REMUS Manager Java Library
3) Mock Test Harness Web Service Application
4) Evaluation Web Service Application
5) Metron REMUS Simulator Application
6) REMUS Viewer Application

To facilitate distribution, the CPT is part of the PRINCESS source tree. The latest version is available in

\princess\cptoolkit\latest

You will need the following third-party software installed to use the CPT:

- Java 8
- Scala 2.12
- Apache ActiveMQ 5.14

The following sections describe each CPT component in depth.

1.1 Evaluation Manager Java Library

All PRINCESS challenge problem solutions use the Evaluation Manager library's API to communicate and log runtime state and data with the Lincoln Labs Test Harness during the evaluation. For instance, it informs the Test Harness when a challenge problem solution has started, when it has completed, and when there is a violation of intent.

At runtime, the Evaluation Manager library depends on the Mock Test Harness Web Service.

The Evaluation Manager supports communication with the Test Harness in one of two ways: either integrated directly into your challenge problem using the TestHarnessAdapter Java API, or invoked from Windows or Linux challenge problem scripts at the command line (using th.bat or th.sh). The usage for the command line version is below:

```
USAGE

TestHarnessAdapter COMMAND where COMMAND is one of the following:

        READY                         system is ready to start the evaluation
```

`PERTURBATION`	`system has detected a perturbation requiring adaptation`
`ADAP_STARTED`	`system has entered an adaptation phase`
`ADAP_COMPLETED`	`system has completed an adaptation phase`
`INTENT true|false`	`system has an intent pass/fail value for the test harness`
`CHECKSUM value`	`system has a checksum value to send to the test harness`
`RESULTS results_file_path`	`system has results file to upload to the test harness`
`DONE rms_error start_date end_date`	`system has results for the test harness`
`DATA_ERROR error_message`	`system encountered an error processing input data`
`STOPPED`	`system is finished processing and is exiting`

See th-test.bat (or th-test.sh) for specific examples of working with the TestHarnessAdapter via the command line.

Note that the TestHarnessAdapter API provides a variety of general Test Harness methods. It is important that our team work with Lincoln Labs to map the expected communication workflow to the Test Harness for each challenge problem solution and determine which methods each challenge problem solution should be invoking. Document this workflow in the same document describing the challenge problem solution.

1.2 REMUS Manager Java Library

Challenge problem solutions that need to communicate with the simulated REMUS platform use the REMUS Manager library's API. It listens for messages from the appropriate Metron REMUS Simulator JMS topics and provides concrete data structures for the message content it sends and receives.

At runtime, the REMUS Manager library depends on the Metron REMUS Simulator.

1.3 Mock Test Harness Web Service Application

The Mock Test Harness Web service is a Grizzly Java application that mimics the basic functionality provided by the Lincoln Labs Test Harness during evaluation. The purpose is to support end-to-end local testing of challenge problem solutions. When the Mock Test Harness receives a request, it tests to see if the JSON content of the request meets the minimum requirements specified by Lincoln Labs for format and parameters. Use the Mock Test Harness Web service during local testing of your challenge problem solution to ensure it is sending well-formed requests to the service at runtime.

To start the Mock Test Harness Web service application, run the following script:

- Windows Command Prompt
    a. cd princess\cptoolkit\latest\mockwebservice
    b. grizzly-service.bat
- Linux Terminal Window
    a. cd princess/cptoolkit/latest/mockwebservice
    b. ./grizzly-service.sh
- Linux Background Process
    a. cd princess/cptoolkit/latest/mockwebservice
    b. nohup ./grizzly-service.sh > thws.log &

1.3.1 Configuration Files

Configuration files allow you to tailor the JSON responses the Test Harness supplies in response to /ready and /path requests made by PRINCESS challenge problems. Configuration files are text files that contain the JSON the Test Harness would normally supply, making it easy to change this data for testing. The configuration files are located in

mockwebservice\config\default\

cp1-initialParams.json

cp2-initialParams.json

cp3-initialParams.json

cp3-batteryPerturbations.json

The file names must stay the same, so either you can edit the default files in place or you make a copy of the \default directory and then edit the files as needed. You can select the configuration directory to use by sending a /config request to the Mock Test Harness Web service via the Test Harness Adapter (this makes it convenient to set up unit tests). Once you set the configuration directory, the Web service will continue to load files from that location until it is changed or the Web service restarts.

1.4 Evaluation Web Service Application

The Evaluation Web service is a Grizzly Java application that provides a means for Lincoln Labs to determine if the PRINCESS system is still running. The purpose is to support the /alive HTTP request endpoint that can be queried to see if any error has occurred that has left the system in an unusable state. Requests to this endpoint can come at any time during an evaluation run, so PRINCESS updates its runtime state in real time by examining errors and exceptions reported during initialization and at runtime.

Request Method	**URL**
GET	http://localhost:8081/princess/alive
Response Type	**Response Details**
text/plain	TRUE if running as expected, FALSE otherwise

To start the Evaluation Web service application, run the following script:

- Windows Command Prompt
  c. cd princess\cptoolkit\latest\mockwebservice
  d. evaluation-service.bat
- Linux Terminal Window
  c. cd princess/cptoolkit/latest/mockwebservice
  d. ./evaluation-service.sh
- Linux Background Process
  c. cd princess/cptoolkit/latest/mockwebservice
  d. nohup ./evaluation-service.sh > evws.log &

1.5 Metron REMUS Simulator Application

Our partners at Metron are implementing and maintaining a Hydroid REMUS 600 simulator Java application. The simulator loads a JSON scenario file and then plays it back by generating realistic data for the REMUS underwater motion model, sensors, and ground truth. The simulator regularly broadcasts this data as messages to several JMS topics for consumption by external applications. Java or Scala applications interested in integrating this simulated data can use the REMUS Manager library.

To start the REMUS Simulator application, **first start the Apache ActiveMQ broker**, then run the following script:

- Windows Command Prompt
    a. cd princess\cptoolkit\latest\remus-simulator\bin
    b. simulation.bat
        i. Starts the UI
    c. simulation.bat scenario_file
        i. Automatically loads and runs scenario_file
- Linux Terminal Window
    a. cd princess/cptoolkit/latest/remus-simulator/bin
    b. ./simulation
        i. Starts the UI
    c. ./simulation scenario_file
        i. Automatically loads and runs scenario_file
- Linux Background Process
    a. cd princess/cptoolkit/latest/remus-simulator/bin
    b. nohup ./simulation scenario.json > /remus-sim.log &

1.6 REMUS Viewer Application

The REMUS Viewer application visualizes a running Metron REMUS Simulator scenario. It displays such things as the areas of current and the REMUS path origin and destination. As the scenario plays and the REMUS moves, it will render tracks for the ground truth location and the location as determined by the adapting PRINCESS Kalman Filter.

To start the REMUS Viewer application, start the Apache ActiveMQ broker and run the following script:

- Windows Command Prompt
    a. cd princess\cptoolkit\latest\remus-viewer
    b. remus-viewer.bat
- Linux Terminal Window
    a. cd princess/cptoolkit/latest/remus-viewer
    b. ./remus-viewer.sh
- Linux Background Process
    a. cd princess/cptoolkit/latest/remus-viewer
    b. nohup ./remus-viewer.sh > viewer.log &

To view a running Metron REMUS Simulator scenario with REMUS Viewer, do the following:

1) Start the Apache ActiveMQ broker

2) Start the REMUS Viewer application
    a. Load the Metron REMUS scenario file you wish to execute
3) Start the Mock Test Harness Web service
4) Start the PRINCESS REMUS Client
    a. This is currently run from your IDE
5) Start the scenario