DataTurbine Matlab Toolkit

Prototype Release, 2/15/2013
Matt Miller, Cycronix

Introduction

These notes, along with the code, are in preliminary draft form.

The DT Matlab Toolkit is a collection of Matlab M-file functions and scripts that implement a data access interface from Matlab to DataTurbine.

Tutorial

This introductory tutorial is comprised of test scripts. Run them, observe their behavior, and study the source code to understand the data structures and how the functions work. Matlab command-line help is available for each associated Toolkit function.

In preparation for the tutorial:

- Run a DT server on localhost machine. java -jar rbnb.jar
- Set Matlab classpath to find DT rbnb.jar file. javaaddpath('rbnb.jar')
- Set your Matlab file path to find your DT Toolkit M-files

Run the following tutorial test scripts as follows.

1. Issue the following command:
>> testplot

See how it plots live (built-in Metrics) data. This will run for 60 seconds then stop. Note how newest overlapping data is retrieved with the DTget function.

2. Issue the following two commands in sequence:
>> testput
>> testget

The first puts data to DT with DTput. The second retrieves it with DTget. Note how the DT structure identifies server, source, and channels. Note how the DT channel structure is an array of channel names on input, and an array of channel structures with data on output.

3. Issue the following command:
>> testnext

This retrieves live (Metrics) data, back-to-back without overlap. It utilizes the DTnext function. Note how each new call uses the prior DT structure to manage consecutive frames. DTget is simpler to use than DTnext, but lacks control of overlapping data.

4. Examine the 'testTrout.m' script. It is an enhanced version of the testnext.m script to sequentially fetch data from the TroutBog data source. Edit it as necessary to specify the IP address and timing

parameters. The default script simply prints some status each iteration; you may want to enhance this logic to save or send data to an archive.

Data Structures

There are two primary data structures for handling DT data. From the Matlab command prompt, type "help DTstruct" or "help DTchan" for descriptions of each.

DTstruct

The DT structure defines the DataTurbine, its data source, time reference, and has a sub-structure channel array per the following example.

```
DT = DTstruct()
  Return template DT structure:
      DT.server = 'localhost';
      DT.source = '_Metrics';
      DT.start = 0;
      DT.duration = 0;
      DT.reference = 'newest';
      DT.chan = '*';
```

DTchan

The DT channel structure is either a single channel name string, or an array of structures defining channel name, time, data, and metadata per the following example.

```
chan = DTchan()
  Return default DTchan structure:
      chan(1).name = 'MemoryUsed';
      chan(1).time = 0;
      chan(1).data = 0;
      chan(1).type = 'int64';
      chan(1).mime = 'binary';
      chan(1).meta = 'null';
```

Functions

The 'DT' functions comprise the DT Matlab Toolkit interface. Help for each function is available from the Matlab command prompt.

DTput

```
nput = DTput( DT [, ncache, narchive, amode] )

   Flush data to DataTurbine. Input is 'DT' structure.
   DT structure parameters:
       DT.server, e.g. 'localhost';
       DT.source, e.g. 'mysource';
       DT.chan(), as follows:

   DT.chan is struct array with names and data, e.g.:
       chan(1).name = 'mychan';
       chan(1).data = 0;

   Optional ringbuffer-settings:
       ncache - cache frames, default=10
       narchive - archive frames, default=1000000
       amode - archive mode, 'none', 'create', 'append' (default)

   Notes:
       Timestamps are all current time, zero duration

       Call DTput() with no arguments to disconnect. Otherwise
       re-uses prior connection for efficiency.
```

DTget

```
[ DTgot nchan ] = DTget( DT )
   Get data from DataTurbine. Input and output is 'DT' structure.
   optional second output param is number of fetched channels
   DT structure parameters (with defaults):
       DT.server = 'localhost';
       DT.source = '_Metrics';
       DT.start = 0;
       DT.duration = 0;
       DT.reference = 'newest';
       DT.chan = '*';

   On input, DT.chan is a string (or struct array with names).
   On output, DT.chan is array of DTchan structures, e.g.:
       chan(1).name = 'MemoryUsed';
       chan(1).time = 0;
       chan(1).data = 0;
       chan(1).type = 'int64';
       chan(1).mime = 'binary';
       chan(1).meta = 'null';
```

DTfetch

```
DTgot = DTfetch(DT)
   Wrapper to fetch array of DT structures.  See DTget()
```

DTnext

```
[ DTgot nchan ] = DTnext( DT [, delay, timeout] )
  Get 'next' data after previous fetch without overlap.
  Initialize with DT.reference 'newest', 'oldest', or 'absolute',
  after that call with DT = DTprevious.

  Parameters:
  DT - DT structure defining what to get
  delay - polling interval to check for new data
  timeout - give up waiting after this long
```

DTlist

```
[ DTgot nchan ] = DTget( DT )
   Get list of channels from DataTurbine. Input and output is 'DT' structure.
   optional second output param is number of listed channels
   DT structure parameters (with examples):
       DT.server = 'localhost';
       DT.source = '_Metrics';
       DT.chan = '*';

   On input, DT.chan is a string (or struct array with names).
   On output, DT.chan is array of DTchan structures, e.g.:
       chan(1).name = 'MemoryUsed';
```