Datacasting Feed Publishing Tools Tutorial

Introduction

This document is intended to provide a walkthrough of the process of setting up a Datacasting Feed. There are a number of assumptions made in this tutorial, so here they are up front:

- You have a stream of data that is being made available for users to access via a URL on a regular basis.

- You have reviewed the Datacasting XML Specification (have a look at http://datacasting.jpl.nasa.gov/) to determine which metadata is required or recommended for publishing a Datacasting Feed.

- You have considered what additional metadata you wish to distribute in the customElement tags and know the data type of that metadata.

- You have a process (which we'll refer to as ingestion) that automatically readies the data for distribution.

- During your ingestion process, you have access to all relevant metadata that you wish to publish in the Datacasting Feed or you are prepared to write your own scripts, programs, or processes to extract that metadata. This includes the URL at which your data will be available for download.

- You are prepared to create a simple plain-text file for each ingested data item that contains all the relevant metadata for that data. You can create this file all at once or in stages by appending information as it becomes available.

- You have a publicly accessible web server to place the feed's XML file on. This server needs to be accessible from wherever you generate the feed so that the feed can be automatically updated as you add more items. You could run a scheduled task to copy the XML file from wherever it is generated over to a web server.

- You have Python 2.5 or higher installed on your system.

- You have the tool xmllint installed (if not, edit the config-sample.cfg to set useXmllint = False)

- You are working in a Unix-type environment. Although the Python code should work in Windows if Python and other necessary tools are installed, this code has only been tested on Mac OS X, Linux, and Solaris.

The Feed Publishing Tools get all feed specific information including the name of the output feed and the directories in which the feed items will be stored from a single configuration file. You must create **one** configuration file **for each feed** (see server/config-sample.cfg) that you wish to publish. A Datacasting feed is

produced in two steps: item creation and feed generation. The two stages can be run sequentially or asynchronously. The first process, IngestItem.py, takes the metadata in a simple plain-text format (see server/ref-doc-item.txt for a commented example) and turns it into a snippet of XML that will be included in the final feed. This process is run every time you complete the ingestion of a new data item. The second process, GenerateFeed.py, maintains a queue of items to be published and publishes those items in an XML feed file that you serve on the web.

Homework

Understanding Your Users

The purpose of producing a Datacasting Feed is to announce to your users when you have new data available and to allow them to decide on which data items to download. Before publishing your feed, think about what metadata your users might need to make their decision.

- Do they care about the geographic bounding box?
- Do they need to know the value of some quality measure?
- Do they want some statistical measure (min, max, or mean) of the data?
- Does your data cover a particular phenomenon that you wish to call out?

Some of the metadata is standard in the Datacasting spec, but you may need to create custom elements to contain others. If you are creating custom elements, you should think about the type of the data you need to represent (integer, float, date, string, etc.).

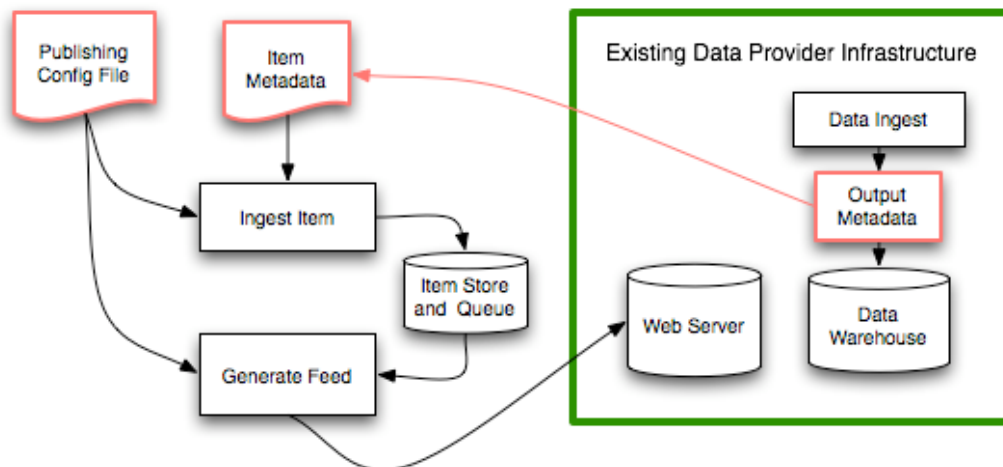## *Understanding the Datacasting Feed Publishing Tools*

Figure 1: Datacasting Feed Publishing Tools Scenario

In the figure shown above, the green box represents your existing infrastructure. The red boxes are files or scripts that you create.

The Publishing Configuration File (a fully commented sample is provided in server/config-sample.cfg) contains all the information about your feed including the definitions of any custom elements that you choose to add. Read through the sample configuration file and determine how to best fill it out for your feed. You need to create one configuration file for each feed you wish to publish.

The Item Metadata file is generated by you during or after your ingest process. In the figure, it is shown as being between Ingest and putting the data in its final location; however, you can create the item file at any time that the metadata for a new data item is available to you. As mentioned above, the Item Metadata must include the URL from which your data can be downloaded. There is a fully commented sample of the item file in server/ref-doc-item.cfg. Read through this sample item file to see what metadata you need to collect about your new data items in addition to the Custom Elements that you define.

You will need to create "item store" and "queue" directories for each feed that you are publishing. The paths to these directories are set in the configuration file. These directories are used by Ingest Item and Generate Feed to organize and queue the items for publishing. You can also specify a "queuing policy" in the configuration file that will control which items appear in your feed. RSS feeds generally only show the most recent items; the Feed Reader tracks all the items that have appeared in the feed in the past.

If you distribute different types of data, use only one feed for each type of data. Each Datacasting Feed should contain data of only one format. If you distribute data from two different instruments, have one feed for each instrument. This allows your users to subscribe to only the feeds that they care about.

Getting Started

In all likelihood, you've already unpacked the package and looked through the files, so bear with me as I lay out the process in detail.

- Unpack the Datacasting Feed Publishing Tools in a location where your operational tools can access it (on a shared file-system or in /usr/local)

- Read the README.txt file; there might be some interesting stuff there.

- From the directory that the README.txt file is in:

    o source ./setup.csh

    o cd test

    o ./testDatacasting

- Assuming that the above test went well, have a look in testDatacasting, there is some good info in the comments.

    o If the test didn't go well, the most likely cause is an old Python version or missing xmllint. You can edit the config.cfg to set useXmllint = False and try again or upgrade your Pyton to 2.5 or higher.

- Return to the directory that the README.txt file was in (cd ..).

- ./make-executables will create CSH scripts, IngestItem and GenerateFeed which can be linked to a bin directory and require no special paths to find the required python modules.

- Run: IngestItem –h and GenerateFeed –h so that you can get a feel for the command-line parameters.

- Link IngestItem and GenerateFeed to a bin directory (like /usr/local/bin). Using links (ln –s) allows you to upgrade the software in the future and simply replace the links. For example:

  ln -s $PWD/IngestItem /usr/local/bin/IngestItem
  ln -s $PWD/GenerateFeed /usr/local/bin/GenerateFeed

  will create links in /usr/local/bin. To upgrade, you just delete those links and recreate them to point to the newer version.

The datacasting tools are now installed. The next step is setting up a feed.

Setting up a Datacasting Feed

Let's say that you have an ops account (/home/ops) that owns the data ingestion process. Here's how you'd set up:

- Create a subdirectory to house your datacasting feeds:
  - cd /home/ops
  - mkdir datacasts
  - cd datacasts
  - mkdir feed1
  - cd feed1
  - mkdir items-xml queue

- Create config file; start by copying the config-sample.cfg
  - cp /*<path to tools>*/server/config-sample.cfg config.cfg

- Customize the config file. Do not keep any of the default values, use ones pertinent to your new datacasting feed. Remember that you need to define all of your custom elements here. You also need to provide a full path for the output XML feed. That location needs to be on a web-server so that your users can read your feed. For now, put it in the current directory: ./test-feed.xml

- Remember to point the config file to your new items-xml and queue directories.

- Create an item file by hand for one of your data items. Start with the ref-doc-item.txt as a template, but, as with the config file, replace all the fields with ones that reflect your data.

- Try ingesting this hand made item
    - IngestItem –c /home/ops/datacasts/feed1/config.cfg my_test_item.txt
    - Keep in mind that the publishing tools do not trap errors well. If you have syntax errors in your files, the failure messages will be quite difficult to read. This will be fixed in the next release.

- Try generating the feed
    - GenerateFeed –c /home/ops/datacasts/feed1/config.cfg

- Your feed should be sitting in the current directory. Try copying it into a web accessible location and accessing it with the Datacasting Feed Reader or any other RSS compliant feed reader.

- If you've run into issues that you haven't been able to debug, please email datacasting@list.jpl.nasa.gov and we'll be happy to help you out.

- If you've successfully generated a feed, start working on the scripts to generate your item text files automatically as part of your ingest process.

- Once you have automatically created an item file, call IngestItem on it. You may delete item text files once they are ingested, as they are stored as XML in the items-xml directory (or use the –e option to delete them automatically, see IngestItem –h)

- Once new items have been ingested, you call GenerateFeed to create the feed. You should probably make sure that the output XML file is being put on a web-server and is named correctly (test-feed.xml likely isn't the name you want, maybe myInsturment-datacast.xml makes more sense).

- If you've made it this far, congratulations, you have a functioning Datacasting Feed. Go ahead and create more feeds; have fun!

- Email us at datacasting@list.jpl.nasa.gov and let us know about your feed so that we can add it to our feed directory.