General Code Writing Rules for CARMEN Services

Wrapping CARMEN Services is an automated process that is performed Using the CARMEN service builder tool. The user describes their application (inputs, outputs, descriptions,etc), adds their application in the form of a binary or script, and the builder tool creates the service in the form of JAR and XML files. These files are then be deployed on the portal so that they can become live.

However, there are a few simple rules that must be obeyed when creating your application in order for it to be wrapped, and subsequently invoked, successfully. This document describes these rules.

In order to wrap legacy applications we require that they are written as a command-line application. Therefore, the general rule set is:

- The application will be running as a service on a remote CARMEN server. Therefore, the user will have no interactivity with the application. As such the application must be non-interactive; i.e. no key presses, mouse click's, GUI's, etc.;
- Input parameters must be passed into the application as string arguments. There is no limit (found so far) on the number of inputs, though large numbers of inputs can make service invocation a little tedious. A ten input service is workable but a bit tedious;
- The application code must process the input parameters, converting them into whatever data-type (int, float, double, etc) is required by the algorithm;
- The required output of the service must be printed to the screen/stdout surrounded by <output></output> tags. The contents of the tags will be captured by the java wrapper and will become the service output, and passed to the portal for data staging. So if the output of the service is to be a filename, the application must print the full name (including file extension) of the output filename to stdout, otherwise the portal will not be able to find the file. If multiple outputs are required, the output data must be printed as a comma-seperated string, i.e. <output>filea.txt,fileb.jpg,10</output>
- All screen output is captured by the service wrapper and displayed in an error file should problems occur. Therefore use debug and error capturing code liberally as this may be the only clue should your program fail when run as a service;
- File handling must be performed within the application, but using filenames passed into the application as input parameters, and passed out of the application as output parameters. It is important to use the full filename as this is used by the CARMEN Portal for data staging. For instance, if your service creates a file called "plot.jpg", and

you output "plot" from the service, the CARMEN Portal will not be able to find the file "plot.jpg" and will not be able to save it in your user space;
- When working with files, no paths must be used. Write your code to use the current working directory.
- The servers that run the service will have no graphics capability, i.e. no X-server session. Applications that create graphical images must preferably be able to run without graphical displays. We can incorporate virtual x-server support into your service wrapper, however this is not a recommended option as it severely limits scalability.
- The application writer should preferably write a wrapper around their algorithm. In this way they can keep their original algorithm and run it in the usual way;
- It is recommended that applications contain error checking if possible, and return 0 (zero) if successful, and a non-zero integer value if unsuccessful. If the service detects a non-zero return value from the application, it will inform the user (via the portal) that the application failed.
- Test your code thoroughly before committing code for wrapping. Debugging a wrapped application can be extremely difficult; therefore it is good practise to test the tool before wrapping. Since we are in effect building a command-line tool, it can be simply tested as such prior to wrapping. We have had code submitted for wrapping that does not work. We could actually wrap some broken code successfully but it would only ever fail and is a complete waste of time;
- Code can consist of multiple files, i.e. script files, libraries, etc. These must be able to run from a single folder or folder hierarchy.

For services that produce output data files, we recommend that the output files should only contain new data. In theory, the input information could be incorporated into the output file too, but in practise, this would add a lot of redundant data to the datastore, especially if the input data is large and the service is executed a number of times with the same input data file, i.e. in a parameter sweep scenario. We occasionally get requests for the service to append the output data into the input data file. This isn't possible as the input file the service processes, is a copy of the original data in the data repository. The original file also may not belong to the user running the service, so there will be a permissions problem, and there may be issues regarding metadata.

Whilst always good to add input validation to applications that will be wrapped as CARMEN services, there are several levels of input validation that are applied at the portal level. When wrapping applications, each parameter can be individually specified for input validation. This includes;

- Whether an input is a file or a value;
- If the input is a file, it's file extension can be specified;
- If the value is a numeric value, it can be specified as a floating point, integer, maximum and/or minimum values, etc.

There are also several input validation checks inside the application's service wrapper. Since the service is unaware of the context of the data, it can only check that input entries are not empty (null), or if it is a file that the file actually exists on the server.

We can currently wrap Matlab, Python, R, and any language that compiles to an executable (i.e. C/C++, Java, etc). Basically anything that can be run as a command line.