Writing Perl Script for CARMEN Services

This document gives examples on writing perl applications for Carmen Service Wrapping.

Below is an example of the framework required when writing a perl script as a command line, in preparation for wrapping as a Carmen Service. This example script takes two input arguments (input data filename and the name of the generated output filename) and outputs the name of the generated results file.

```
#!/usr/bin/perl
use strict;
use warnings;
my $numArgs = $#ARGV + 1;
if ($numArgs != 2 ) {
  print "\tincorrect number of arguments", "\n";
  print "\tusage:\t inputFilename outputFilename\n";
  exit(-1);
}

# process imput arguments
my $inputFile = $ARGV[0];
my $outputFile = $ARGV[1];

...
# put your algorithm here
...


print "<output>", $outputFile, "</output>", "\n";
exit(0);
```

Note: The above script is only an example of a 2-input single output program. The number of inputs and outputs is totally under the control of the application developer. For differing numbers of input arguments, simply process the input parameters to suit, where ARGV[i] is an array containing the command line parameters. i.e. for four inputs and two outputs;

```
my $numArgs = $#ARGV + 1;
if ($numArgs != 4 ) {
  print "\tincorrect number of arguments", "\n";
  print "\tusage:\t inputFilename_1 inputFilename_2 outputFilename_1
outputFilename_2\n";
  exit(-1);
}

# process imput arguments
my $inputFile_1 = $ARGV[0];
my $inputFile_2 = $ARGV[1];
my $outputFile_1 = $ARGV[2];
my $outputFile_2 = $ARGV[3];

...
# put your algorithm here
...
```

```
# multiple outputs printed to screen as a comma-separated string
print "<output>", $outputFile_1, ",", $outputFile_2, "</output>",
"\n";
exit(0);
```

Note that for multiple outputs, we need to concatenate the individual output strings into a comma-separated list and output it to the screen/stdout.

As mentioned before, the input parameters are passed from the command-line as a text string. Normally, any numeric parameters must be converted to the required data type (int, float, double, etc), however with Perl this is automatically handled.