Writing BASH Code for CARMEN Services

The Unix bash shell is a handy way to create services from unix commands and packages. It can also be used to create a service from a collection of tools from a range of different programming languages.

The following script is an example of a service for converting PDF documents into Postscript format, via use of the Linux pdf2ps utility. The service takes a single PDF input file and outputs a single postscript file.

```
# load input filename
ip=$1

# set output filename
ext=${ip##*.}
name=$(basename "$ip" "."$ext)
op=$name".ps"

# do conversion
pdf2ps "$ip" "$op"

# check program completed successfully
if [ $? -ne 0 ]
  then exit -1
fi

# inform the CARMEN platform of the output file
echo "<output>"$op"</output>"
```

Note the use of quotes to handle filenames containing spaces.

Some Useful Snippets

It's always good to check that the number of arguments passed to your script are correct, and to exit your script if it's incorrect.

```
no_of_expected_args=2
if [ $# -ne $no_of_expected_args ]; then
        echo "usage: $0 <input-file> <output-PNG-file>"
        exit
fi
```

If one of the commands within your script fails, it's important to exit from the script in a controlled way and return a non-zero value to indicate an error to the CARMEN platform. The variable "***$?***" is used to determine the exit status of the last executed function/program/command.

```
# error handling
if [ $? -ne 0 ]; then
   exit -1
fi
```

If you wish to append a string to the basename of a filename whilst keeping it's extension, the following technique can be used. This is useful to create an output file who's name is based upon the input file. For instance, if the input filename is called 'input.txt' and is in the bash variable 'inputFile', and you wish to append the string '_update' to the basename, and create a new filename called 'input_update.txt', then;

```
opFilename=${inputFile%.*}_clean.${inputFile#*.}
```