UNIVERSITY OF YORK

Service Builder

User Guide

Michael Weeks

4/1/2011

The Service Builder is a GUI-based application. It takes an executable that conforms to our specification, requests information on the application from the user, and wraps it up into a service that we can run our Virtual Research Environment (VRE) system. The Service Builder is still in development, though we are making it available to early adopters.

Contents

Setup and Configuration 3

Preparation 3

Installation 3

Configuration 3

Projects 4

Create New Project 4

Load Project 4

Project File Structure 4

Service Builder Process Flow 6

Metadata 7

General Service Metadata 7

Input Parameter Metadata 8

Add Input File 8

Add Command Line Option 9

Add Input Value 9

Add Output Filename 10

Add Input Drop-Down Selector 11

Output Parameter Metadata 12

Wrapping 13

Local Testing 14

Going Live with your Service 15

Setup and Configuration

Preparation

Prior to running the Service Builder, the service creator must prepare a system that is compatable with our system, in particular, our execution nodes and their environment. At the moment, we support the following Operating Systems on our execution nodes:

- Scientific Linux 4 x86 64-bit - This is in the process of being phased out;
- CentOS 5 Linux X86 64-bit;
- Windows Server 64-bit.

If the service requires some installed software to run, please contact our support team who will give details on currently-installed software. If the software you require is not available within our system, we will advise on whether we can add it.

Please note that preparation of a system is critical to the successful wrapping of the service. If your service is compiled for the wrong type of OS, or relies on installed software we don't have, it will fail to run correctly on our system.

Installation

The service builder is a single runnable jar file which has cross-platform compatibility. Copy the file onto the machine where you are going to create your services, and run it as you would an executable.

If your machine does not recognise the service Builder as a runnable jar, you can start the service builder via the java command, i.e.

```
Java  -jar  carmenServiceBuilder.jar
```

Configuration

Before a service can be wrapped, the Service Builder must be configured to match the platform's target architecture, and the wrapping method.

To configure the Service Builder, start it and go to the help/configure drop down menu. For the target architecture, we currently only use Windows server 64-bit, Scientific Linux 4, and CentOS Linux 5. We are in the process of phasing out Scientific Linux 4, and replacing it with CentOS Linux 5.

For the wrapping method, please make sure that "JavaClass" has been selected.

Projects

The Service Builder creates one service at a time, and is based around the project metaphor. The service creator can create a new project, and at every stage of the process, any information it has been given, and its state, are stored to local disk. At a later stage, the project can be reloaded in order to change some information, or redo a task.

Create New Project

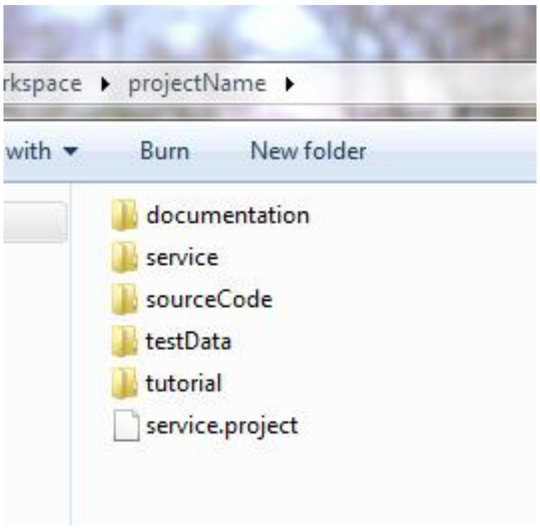
Select “create new project” from the project menu, and the new project panel appears. Enter the name for your project, and either select a location for your project or leave it as the default location. Press the **create project** button to initialise the project and to create the project structure.

Load Project

Previously created projects can be reloaded by selecting “open existing project” from the drop-down projects. A file browser appears allowing you to navigate to your project folder, and select the file called **service.project**. Selecting the file populates the Service Builder with the project information and state.

Project File Structure

When the service builder creates a project with the name “projectName”, it has the following structure:



The service.project file is an xml file contains saved information and state from the builder. This can be used to reload previously saved projects.

The sourceCode folder should contain the code you used to build the application. This should also contain the application as well. In this way, the whole project can be transferred to another machine and rewrapped, without getting a broken link to the original application.

The testData folder will contain any test data that can be used to test your application. This will be used by the builder to test your service (in the near future), and longer term, by the portal to perform automated service testing with.

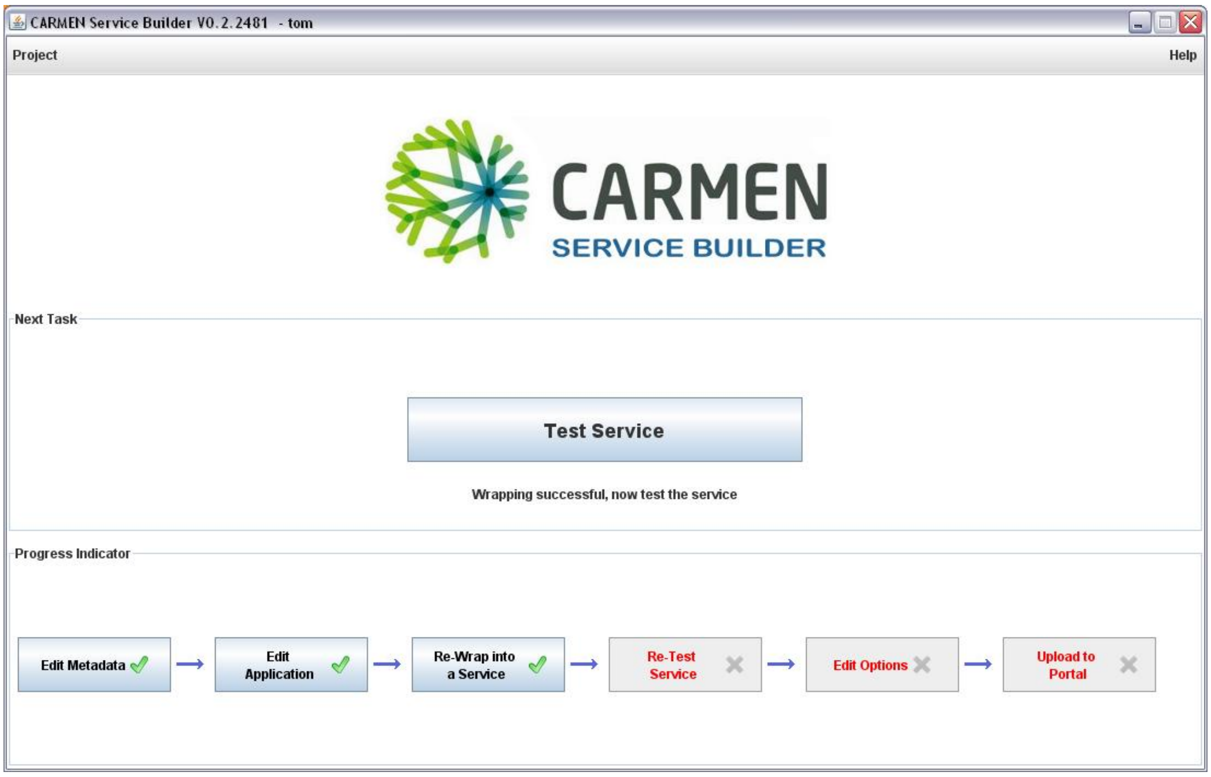
Any documentation describing the application, or how to rebuild it, can be placed in the docuumentation folder.

The tutorial folder is for future use when we can generate tutorial information regarding the application.

The service folder contains the wrapped application, and a jarred version of the wrapping. It also contains an xml file containing the service metadata.

Service Builder Process Flow

Once a project has been created or reloaded, the service builder displays the front panel view



Wrapping successful, now test the service

The view consists of two main panels that show the service wrapping process status.

The **Next Task** panel (middle panel) shows the recommended next process to perform in the wrapping process. This advances as each stage is successfully completed.

The **Progress Indicator** panel (lower panel) illustrates the process flowchart and the current projects progress.

To advance along the flowchart you must use the large button on the next task panel, however you can at any point return to a previous process to amend any details. This will however reset your position in the flowchart and you will have to recomplete the subsequent steps.

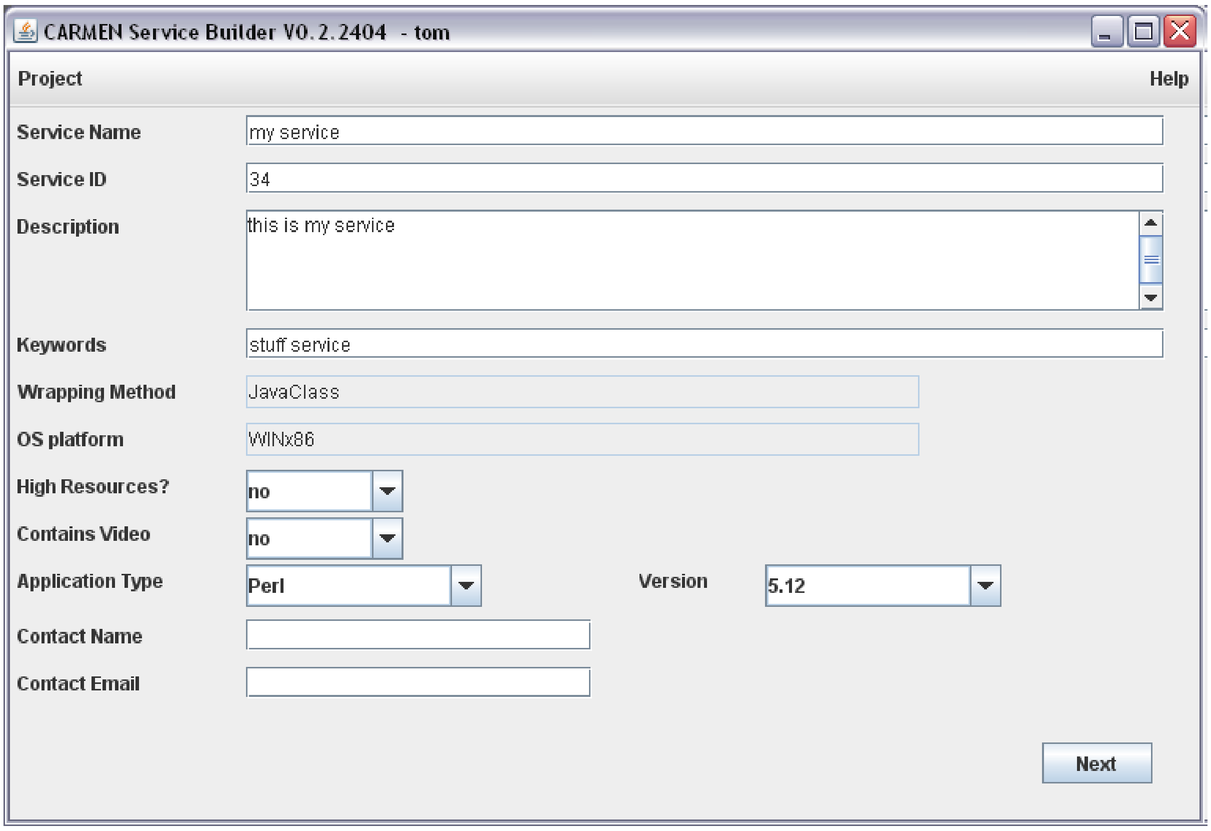
After each progression along the flowchart the state and metadata for the project is automatically saved to disk. Reloading an existing project from disk will set the flowchart to the last stored state.

Metadata

The metadata section of the Service builder consists of three forms; the service metadata, the input metadata and the output metadata.

General Service Metadata

Upon selecting Add/Edit Metadata from the front panel, the "Metadata Entry" panel appears. This panel is shown below, followed by a description of each field.



Service Name	my service
Service ID	34
Description	this is my service
Keywords	stuff service
Wrapping Method	JavaClass
OS platform	WINx86
High Resources?	no
Contains Video	no
Application Type	Perl
Version	5.12
Contact Name	
Contact Email	

Service Name - This is the name of the service as displayed by the portal to the user. It will be displayed in the list of services, either in favourites or in the search panel. The name needs to be a short snappy title, but descriptive enough to inform the user of it's functionality. Typically, up to 10 words long.

Service ID - This is the name of the XML and JAR file when wrapping is complete. This field will likely disappear in future, and the project name will be used for the filenames.

Description - A longer description of the service, which is displayed to the user via the portal. This field should contain information on the service that wouldn't fit in the service name field. The description can be upto several paragraphs long.

Keywords - A space-separated field containing a selection of keywords for use during search.

Wrapping Method - This field is uneditable - it is set when configuring the Service Builder.

OS Platform - This field is uneditable - it is set when configuring the Service Builder.

High Resources - If your service uses a lot of a specific server resource, such as RAM, then this field should be set to "yes". If selected, this service will restrict itself to a single execution node, and block other services from running on that node. This should only be set to yes in extreme circumstances, as it limits the processing for your service and the system in general.

Contains Video - On linux systems, the remote execution nodes do not have graphics displays (X-servers). Whilst not a problem for generating images (jpg, png, pdf, ps, etc), it can cause problems for movie generation. Setting this parameter to "yes" creates a virtual X-server on the execution node. It also sets the high resources field to high, limiting the number of services that can run concurrently.

Application Type and **Version** - Specifies the type of application that is being wrapped and the version to be used. The version numbers are the instances of the applications that we have available on our servers. If an application type or version is not available for your requirements, please contact our support staff.

Contact Name and **Contact Email** - The details of the person who has generated this service. This field will disappear in future, as the YouShare username of the person who deploys the service will be set when deploying the service.

Once this form is complete, hit "**Next**" at the base of the form to view or edit the input parameters.

Input Parameter Metadata

For each of the application's command line parameters, we must create one input parameter in the Service builder metadata.

We currently support four types of input parameter; file, value, command line option and NDF file. We will not discuss NDF file parameters in this document. The input value can take on quite a few forms as can be seen below.

From the input Parameter panel, you can add, edit or remove metadata for input parameters. By highlighting a parameter, and clicking the two arrow buttons on the right-hand side the order of the parameters can be re-arranged. As each input parameter is added to the metadata, an example of how the command should look is displayed at the base of the builder panel.

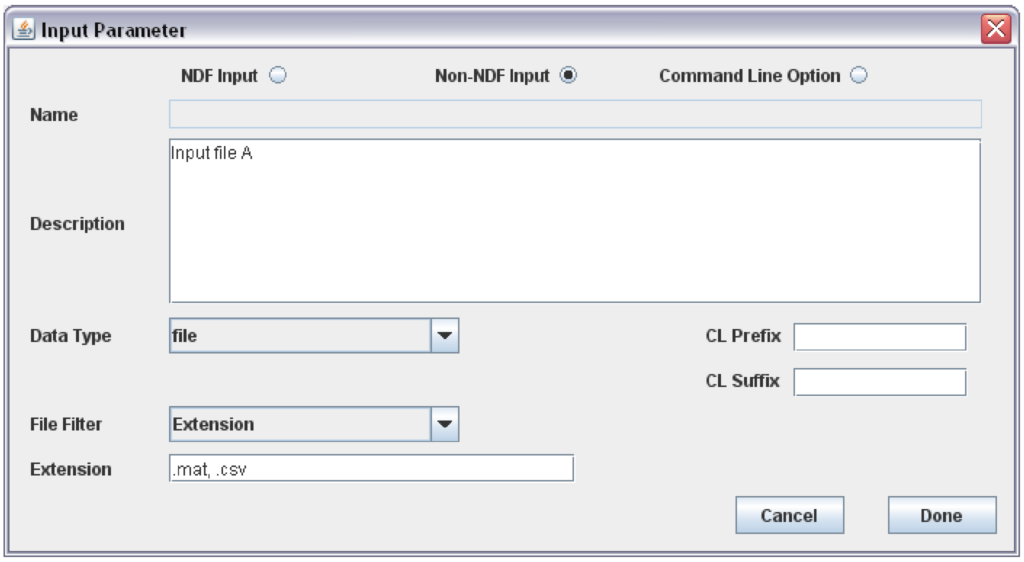
Add Input File

To set a parameter as an input file, click the add input button to display the Input parameter dialog panel.

To select a file input, click the "Non-NDF Input" radio button at the top of the panel, and select "file" from the **Data type** drop-down box.

Give the input a brief description which may be up to several sentences. This will be displayed to the service user on the portal when they try to run your service so it should be meaningful and informative.

Adding a file filter means that the input file browser only displays files of that type when running the service. This must be of the format ".mat", though multiple file extensions can be specified by using a comma-separated list, i.e. ".mat, .csv, .txt".



Non-NDF Input [✓]

Name:

Description: Input file A

Data Type: file

CL Prefix:

CL Suffix:

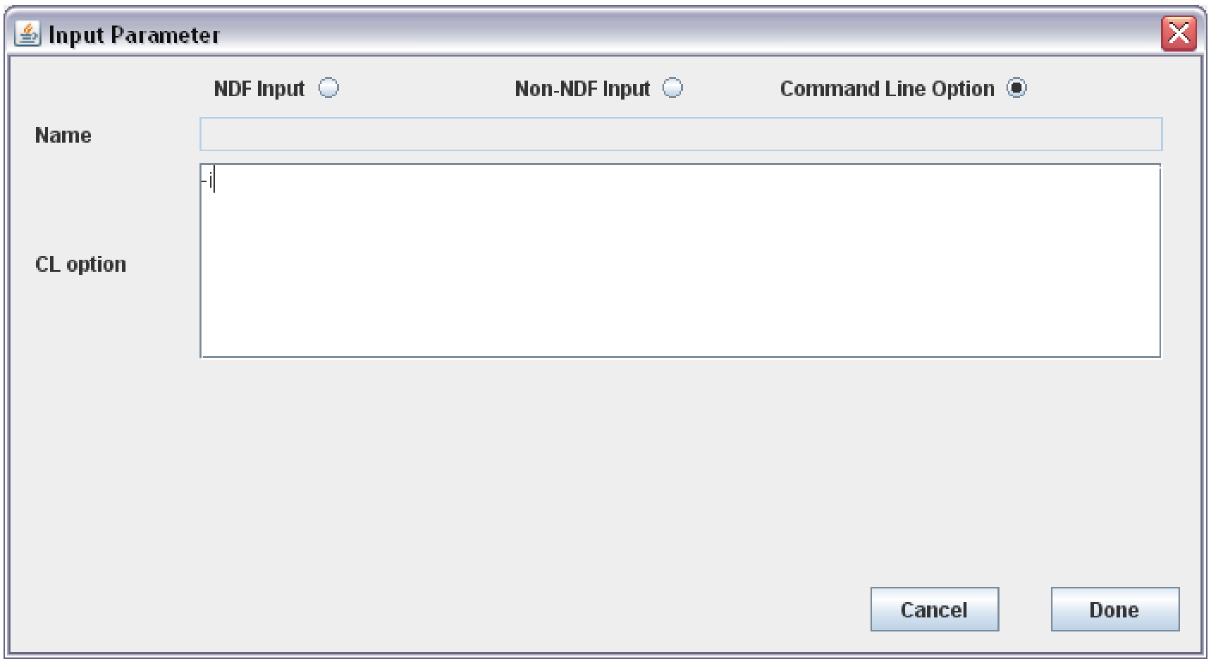
File Filter: Extension

Extension: .mat, .csv

The CL prefix and suffix options allow you to add an extra string on the command line. For instance if the command line demands that the input filename is prefixed with a "-i" option (i.e. abc.exe –i input.mat) then you can add "-i " in the **CL Prefix** text box.

Add Command Line Option

This is another variation on the CL Prefix option shown above. By selecting the command line option radio button at the top, you can enter the command line option in the CL option box.



Command Line Option [✓]

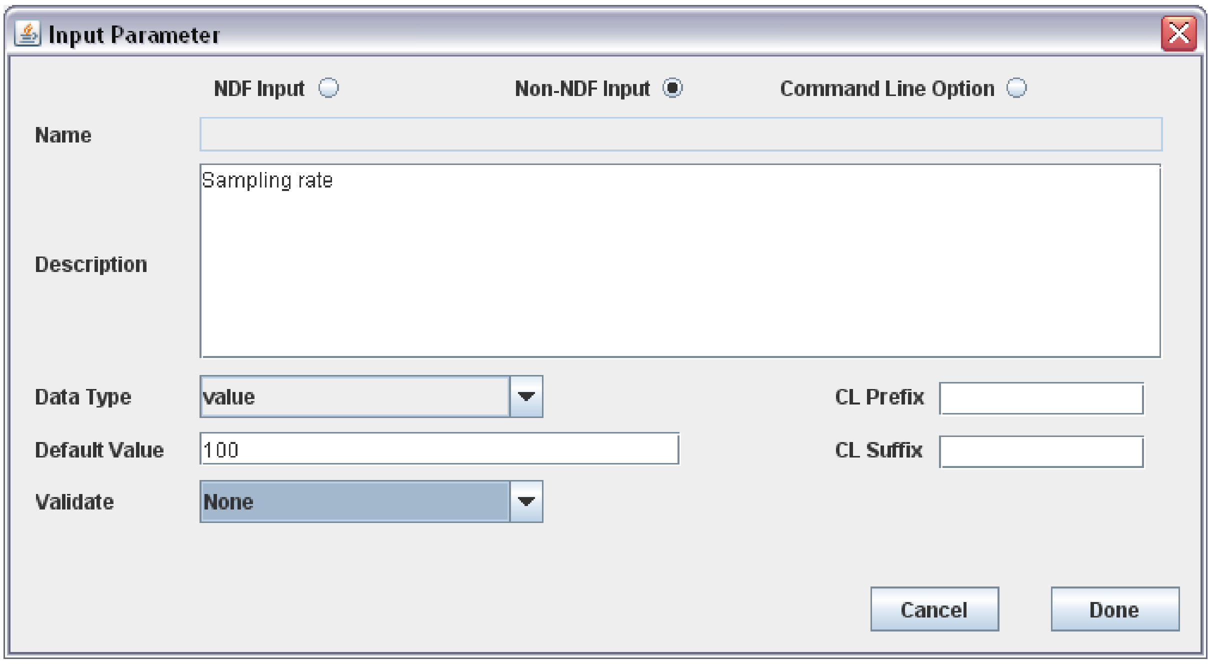
Name:

CL option: -i

Add Input Value

For value input parameters select the Non-NDF radio button at the top of the screen and select "value" from the **data type drop-down** box. Enter a description of the input value as described above.

A default value can be set, for instance in this case we have set the value of 100. So when this service is executed, the Sampling rate field will be pre-populated with the value 100, though the user can change it to whatever they wish.


Input Parameter

NDF Input ○ Non-NDF Input ● Command Line Option ○

Name	
Description	Sampling rate
Data Type	value
Default Value	100
Validate	None
CL Prefix	
CL Suffix	

Add Output Filename

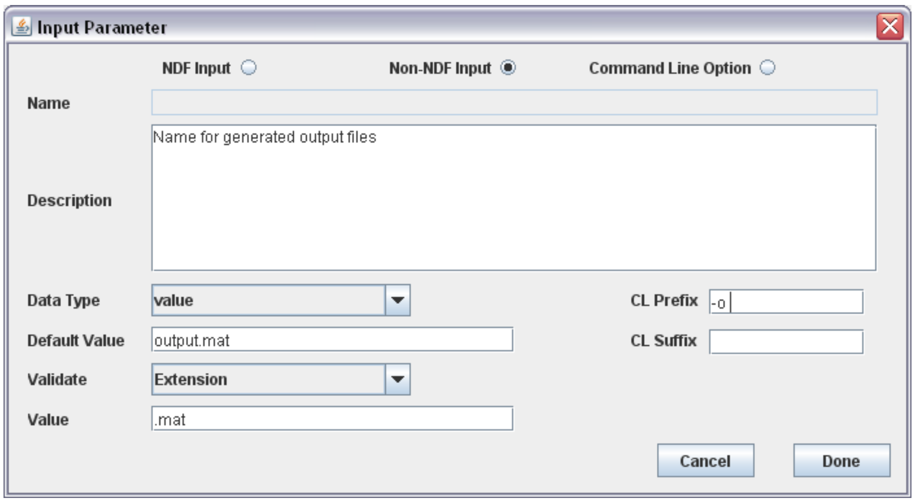
Sometimes you may wish the service to ask the user for the name of any output files that are generated. In this case, the input is set as a value as it is simply a name to be used by your service, and NOT an actual file. If this is set incorrectly to a **file**, the portal will request the user to select an actual file that exists in the system and so the service will fail.

You have several options here, all of which are dependent upon the operation of the application being wrapped.

Set the data type to **value** and leave the remaining fields unspecified. When this service executes, any string value entered here will cause your service to generate files of that name (should your application work in that manner).

You may add a default value if you want to make it easier for the service users at execution time.

If you want the users to specify an extension at runtime, you can set the **validate** button to “Extension” and setting the extension type (one extension only) in the value box. If the user misses off the extension at runtime, the portal will check this and append it if needed.


Input Parameter

NDF Input ○ Non-NDF Input ● Command Line Option ○

Name			
Description	Name for generated output files		
Data Type	value	CL Prefix	-o
Default Value	output.mat	CL Suffix	
Validate	Extension		
Value	.mat		

Alternatively, if a service generates multiple files, it is sensible to only ask for a basename, and for your application to use that to generate its output files.

Add Input Drop-Down Selector

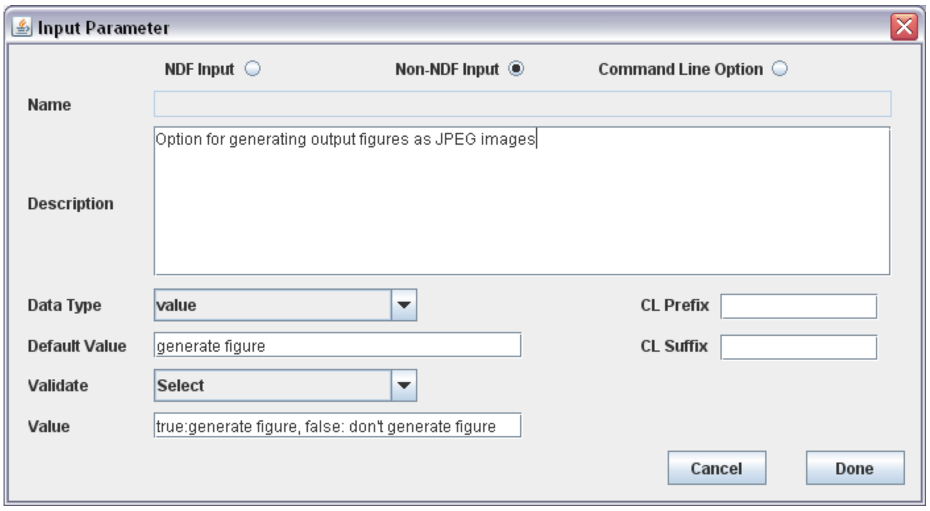
If the range of input values is known, you can set up the input so that a drop-down box is displayed on the portal for that input at runtime.

To do this, again select **non-NDF Input**, and **value**, and then choose “select” from the validate drop-down box. Then the options you want the portal to display must be listed in the value box as a comma-separated list. A default may be specified by adding the specified option in the default value field.

Sometimes, the options required by the application might be meaningless to the person running the service. For instance, the options “true/false” maybe used by the aplication to perform some task, but it may make more sense to the service user to display an alias such as “draw figure/no figure” instead. This can be achieved by setting the list as:

optionA :aliasA, optionB:aliasB, . . .

And any default values set to the required alias


Input Parameter

NDF Input ○ Non-NDF Input ● Command Line Option ○

Name			
Description	Option for generating output figures as JPEG images		
Data Type	value	CL Prefix	
Default Value	generate figure	CL Suffix	
Validate	Select		
Value	true:generate figure, false: don't generate figure		

Once you are happy with the input parameters, you can either hit ***next*** to advance towards the output parameter panel, or go ***back*** to view the general metadata.

Output Parameter Metadata

Output parameters are simpler and simply require you to give a description to the output and and set the output data type of NDF, File or value.

Note that as you add outputs, at the bottom of the output parameter screen, there is an example of the output tags that your application must generate. The forms of these must match exactly in numbers and type.

Once you are happy with the output parameters, you can either hit ***next*** to return to the front panel, or go ***back*** to view the input parameter metadata.

Wrapping

Once the metadata and application has been selected, the wrapping stage can be selected.

The wrapping panel opens and performs a series of checks. If these requirements are ok, the “click to generate service” button will be enabled. Clicking the button generates the service, and indicates whether the service wrapping has been successful or not. Upon completion, clicking the button again makes a log appear giving details of the wrapping process.

If the wrapping was successful, the **NEXT** button will be enabled and will return you to the front panel view. The service now exists as two files (a jar and an xml file) in your project/service folder.

If the wrapping failed, hit the cancel button and fix the problem that was specified in the wrapping log.

Local Testing

The builder will have the ability to test a wrapped application in the near future, however, at the moment this must be done from the console.

Open a console and go into the service/app directory. Here you should see your application and a runApplication script. You can run your application directly from here; this will test out any dependencies that may be missing. You may have to move any input data into this directory first.

Check that the application prints the required outputs to the screen in <output></output> tags, with multiple outputs separated by a comma ‘,’. Also that any filenames in the output actually match with the ones produced on disk.

Alternatively, you can run the script, which is the first part of the wrapping. This script sets up any environment variables, and then calls the target application. To run the script,

```
run_serviceApplicationScript.bat  .  .  file parameter list
```

Note the two dots at the beginning, these just set path information for our test to the current working directory.

Once the script has completed, check that it prints the correct outputs to the screen in <output></output> tags. Also that any filenames in the output actually match with the ones produced on disk.

If they run ok, we can check that the full wrapping works. Move to the classes folder (“cd ../classes/”). Then run “java Test” – this should print out a list of the expected parameters. Make sure you have any input files present, and run the application. This time the output tags should be stripped, but the outputs data still present. Again check that any output files exist on disk. This tests that the wrapping code is actually working, however there could still be an error in the metadata, such as describing an output file as a value. This can only be found by running the service on the portal, or in the builder when this feature becomes available.

Going Live with your Service

Once you have created your service, and it has been tested successfully, it can be uploaded to the YouShare system. This will be achieved in future via the Service Builder application.

Currently service upload is achieved by YouShare administrators. Inside the service folder of your project, there will be a JAR file (the implementation) and and XML file (the metadata). Send these to the service admin team and they will register your service with the live system. You also need to specify whether you want the service to be private to you or public, though the service will be owned by you, so you will be free to change permissions once it's live.