# SALSSA – Adding a new Code

The following is a guide on how to add a new code to SALSSA, using the computational chemistry code NWChem as an example. In general, adding a new code to SALSSA is composed of the following steps:

- **Register Code Processes/Files** associated with the new code.
- **Register a Job Script Template**, used by SALSSA to generate a Job Script for execution of the new code.
- Create a **Compute Machine Configuration** for launching a job.

Each of these steps are described in more detail below. The following assumes that NWChem has already been downloaded and installed. If you would like to follow along using NWChem, the link above provides access to downloads and installation instructions.

## Register Code Processes/Files

The first step in adding a new code is to register the code's new processes and file types with SALSSA. This registration process captures information about the code allowing SALSSA to evaluate dependencies, identify input and output file types, perform metadata extraction, and to correctly construct invocations of the new code.

Registration information is stored in *.ini* files located in either the User Registry or the System Registry. The System Registry contains registrations that come with SALSSA. The User Registry contains registrations defined by a user of SALSSA. Both System and User registries are loaded when SALSSA start up. If the same process or file type registration is discovered in both System and User registries, the User registry will take precedence. In this way, you can create a custom version of a System registration by adding it to your User Registry. The System Registry should not be modified directly.

Following are locations for the User Registry, based on platform (<*home*> is the user's home directory):

- Linux/Mac: <*home*>/*.salssa/registry*
- Windows: <*home*>*Application Data\salssa\registry*

Following are locations for the System Registry (<*install_dir*> is location where SALSSA was

installed):

- Linux/Mac: <*install_dir*>*/var/salssa/registry*
- Windows: <*install_dir*>*/Python25/var/salssa/registry*

In the case of NWChem, create a file called *nwchem.ini* in the User Registry. The only constraint on the file name is that it include the extension *.ini*. Add to it the following text (don't include the line numbers). This same file (without line numbers) can be found in the *Example Files* section below:

```
 1 ;
 2 ; NWChem File Types
 3 ;
 4 [application/text/nwchem/input]
 5 mime_key=NWCHEM_INPUT
 6 name=NWChem Input
 7 metadata_extractors = [
 8   Extractor("Title",  "^\s*title\s*\x22(.*)\x22.*$"),
 9   ]
10 filename_patterns = [ "*.nw" ]
11
12 [application/text/nwchem/output]
13 mime_key=NWCHEM_OUTPUT
14 name=NWChem Output
15 filename_patterns = [ "nwchem.out", ]
16 file_contents = [ "Northwest Computational Chemistry Package", ]
17
18 ;
19 ; NWChem Process
20 ;
21 [process/nwchem]
22 name=NWChem
23 process_category=NWChem
24 remotable=True
25 inputs=[
26   Input(mimetype="NWCHEM_INPUT", mandatory=True)
27   ]
```

```
28 widget_class=salssa.graph.nodes.SimNode
29 outputs = [
30   "NWCHEM_OUTPUT"
31   ]
32 primary_output_filename=nwchem.out
```

*nwchem.ini* contains three sections, where a section identifies the type of resource being registered. A section is started using bracket notation containing a mime type, which is used by SALSSA to identify the new process or file type. Lines 4, 12, and 21 each start a new section identifying an NWChem input file, an NWChem output file, and an NWChem process respectively. Each section contains key/value pairs called registration keys used to describe the particular process or file type. As stated earlier, if the same mime type appears in the System Registry, the User registry will override it. Following is a description of the different parts of the file, by line number:

1-3 Comment lines. The semi-colon character can be used to insert comments. Anything that appears after the semi-colon character is considered a comment.

4 Start of a new section, identified by the mime type *application/text/nwchem/input*. This section describes an NWChem input file.

5-10 Registration keys and associated values used to define the NWChem input file. Registration keys are described below.

12 Start of a new section, identified by the mime type *application/text/nwchem/output*. This section describes an NWChem output file.

13-16 Registration keys and associated values used to define the NWChem output file. Registration keys are described below.

18-20 Comment lines.

21 Start of a new section, identified by the mime type *process/nwchem*. This section describes an NWChem process. By convention, SALSSA assumes all process mime types start with *process*. Adherence to this convention is necessary for SALSSA to function properly.

22-32 Registration keys and associated values used to define the NWChem process. Registration keys are described below.

This is a simple example demonstrating the process of registering NWChem with SALSSA, however it only shows a subset of the possible registration keys. Following is the complete list of user registration keys and how they may be used in the registration process:

| **Registration keys shared between processes and file types** | | |
|---|---|---|
| **Key** | **Value Type** | **Description** |
| name | string | Name of the resource |
| mime_key | string | Defines a key that can be used in place of a mime type string in registration files. For large registration files, it may be easier to use a mime_key. |
| widget_class | string | Fully qualified python class used to render resource on the graph. Possible values include:<br>• *salssa.graph.nodes.SimNode*<br>A simulation node. Image and name will appear in the node representation.<br>• *salssa.graph.nodes.TextNode*<br>A text node. Only name will appear in the node representation. This is the default.<br>• salssa.graph.nodes.ImageNode<br>An image node. Only image will appear in the node representation. |
| **Registration keys for file types only** | | |
| **Key** | **Value Type** | **Description** |
| default_process | string | Either a process mime type or a known process mime_key, the indicated process will be invoked when "activating" (i.e. double-clicking) a document of this registration's mime type. |
| filename_patterns | list | A list of file name patterns. These patterns are used to automatically determine a file's mime type. Value is expressed in python list format. |
| file_contents | list | A list of regular expressions. These regular expressions are used to peek inside a file to determine its mime type. This is useful if there is not a unique file name pattern for the file type, and there is a well known string inside the file that can be matched with a regular expression. Only the first 1000 bytes of the file are looked at. Value is expressed in python list format. |
| metadata_extractors | list | A list of Extractor(name, regex) class initializations. The Extractor class is used to extract metadata from a file. Name is the name of the new metadata property, and regex is the regular expression used to extract the metadata. The example NWChem registration uses the following to extract a Title from an input file (an example input file *nitrogen.nw*, is included in the *Example Files* section at the end of this document): |

| | | |
|---|---|---|
| | | [<br>  Extractor("Title",<br>  "^\s*title\s*\x22(.*)\x22.*$"),<br>]<br><br>This example extracts the string between the pair of double quotes that follow the keyword *title.* |
| **Registration keys for processes only** | | |
| **Key** | **Value Type** | **Description** |
| application | string | Contains the absolute path location of a local binary/executable. Relative paths are not supported. If just the application name is provided, then the PATH environment variable is searched for the suitable application.<br>The application entry can be a comma-separated list, this allows an application to be registered on more than one platform.<br><br>Examples:<br>• /usr/bin/kedit<br>• C:\Program Files\Textpad 4\Textpad.exe<br>• tkdiff,c:\Program Files\TkDiff\tkdiff.exe |
| process_category | string | The category of a process type. Each new category will show up as a separate sibling in the task tree. You can further categorize items using the slash "/" character (e.g. "stomp/setup", "stomp/analysis") which will result in child folders within the task tree. |
| remotable | boolean | Indicates if the resource can be invoked and run remotely. If this is indicated, then there is no need to register the "application" field. This field will then appear as part of the Launch dialog with a target-machine-specific path for the application on the remote machine. |
| inputs | list | Identifies the file types that can be used as inputs. Note that the order of the inputs, if more than one is given, is enforced. The order will be the same when generating the command line invocation. Inputs are specified by name and take a number of keyword arguments.<br><br>Up to 7 key/value pairs can be provided per "Input" in no particular order. Only the "mimetype" is strictly required. Reasonable defaults are provided for the other fields. Following are the “Input” keys: |

| | | |
|---|---|---|
| | | *mimetype* (required) : single string or list of strings . Mime type or mime type list of the input resource(s). Mimes must be quoted, see examples below:<br><br>• "application/text"<br>• ["application/text"]<br>• ["application/text",”another/mime"]<br><br>*mandatory* (optional, default True): a boolean value indicating whether or not the input type is Manadatory or Optional.<br><br>*flag* (optional, default ""): string. A command-line flag or parameter. For example, some input files require a flag such as "-i <input-file>".<br><br>*count* (optional, default 1) : integer . The count/number of inputs that are used. Acceptable values are integers. Any negative integer is interpreted to mean "one or more".<br><br>*help* (optional, default "") : string. A quoted help string which will be used by internal GUIs. This text will appear on the input wizard.<br><br>*required_name* (optional, default ""): string . Automatically rename the input file to this given name. Some file names may be arbitrarily named until used as inputs. This entry allows them to be renamed automatically. This is similar to symbolically linking to a file in another directory, such as "restart.10" to "restart".<br><br>*visible* (optional, default True) : boolean. Whether this input should appear on the command line. Some programs require to be run in the same directory as their inputs. If there weren't an Input entry, the software wouldn't know to use them, so this is how an Input can be specified without breaking the command line. |
| outputs | list | List of outputs by mime type string or mime_key. This list is used to tag the generated files with their proper mime type. |
| files_to_remove | string: comma delimited list | Comma delimited list of file patterns for files that should be removed after process completion. Pertains specifically to remotable processes but also applicable to any process. Default value is "". |
| output_file_patterns | string: whitespace delimited list | Whitespace delimited list of file patterns to save from the outputs of a process. This pertains specifically to (remote) processes but is also applicable to any process. Default value is * (i.e. all files are saved). |

| | | |
|---|---|---|
| primary_output_filename | string | Name of the primary output file for a process. |

## Register a Job Script Template

SALSSA provides a default job script template (the default job script template is shown in the *Example Files* section below). A job script template is used by SALSSA to generate a job script used to submit jobs for execution. However, if the default template is not adequate for your code, SALSSA supports the capability for a user to register their own custom template.

A job script template can apply to a specific code or a code running on a specific compute machine. Following is the order in which SALSSA will select a job script template (where the following step takes precedence over the previous. if found), in order to generate the resulting job script:

1. default template
2. code based template
3. code/compute machine based template

User's can register their job script template by adding it to the User Registry under the sub-directory *jobscripts*. Like process and file type registrations described above, user registered job script templates will take precedence over the System Registry. System registered job script templates are also under a sub-directory called *jobscripts* (see above for the location of the User and System Registries).

The job script template file name must adhere to one of the following naming rules, otherwise it will be ignored by SALSSA (this is also the order in which templates are discovered, as mentioned above):

1. File name is *default*. This file will be recognized as the the default template and will be used if no other template is discovered. If this is added to the User Registry, it will override the *default* job script template provided in the System Registry.
2. File name is <*mime type key*>, where <*mime type key*> is the last component of the processes mime type. This is a code based template. For example, the mime type value for an NWChem process is *process/nwchem*. The mime type key, in this case, is *nwchem*. A file named *nwchem* would be discovered by SALSSA.
3. File name is <*mime type key*>_<*hostname*> or <*mime_type key*>_<*hostname*>_*tasks*. This is a code/compute machine based template. <*mime type key*> is the same as above, <*hostname*> is the name of the compute machine, and the suffix *tasks* is used if the template applies to task parallel jobs. For example, for a single NWChem job running on salssa.pnl.gov, the file name *nwchem_salssa* would be discovered. If it were a task parallel job, then a file named *nwchem_salssa_tasks* would be discovered by SALSSA.
4. File name is <*mime type key*>_<*logical compute machine name*> or <*mime_type key*>_<*logical compute machine name*>_*tasks*. This is a code/compute machine based

template. <*mime type key*> is the same as above, <*logical compute machine name*> is the name a user assigns to a compute machine configuration in the launch dialog, and the suffix *tasks* is used if the template applies to task parallel jobs. For example, for a single NWChem job running on the compute machine configuration named *salssa_nw,* the file name *nwchem_salssa_nw* would be discovered. If it were a task parallel job, then a file named *nwchem_salssa_nw_tasks* would be discovered by SALSSA.

The job script template itself is a shell script file containing SALSSA variables, which are substituted with actual values when the job script is generated. In the case of NWChem, create a file called *nwchem* in the User Registry under the sub-directory *jobscripts*, and add to it the following (this file is also provided in the *Example Files* section below, without line numbers):

```
 1 #!/bin/sh
 2 # This is an example job salssa script template
 3 # for NWChem
 4
 5 # Run dos2unix on all text files
 6 for f in *
 7 do
 8   if [ `file $f |grep -c text` != "0" ]; then
 9      dos2unix $f 2> /dev/null
10   fi
11 done
12
13
14 # In the simple case, the program simply runs with well known input
15 # and outputs which have been registered with SALSSA
16 ##code## >& nwchem.out
17
18 # The status must be written to a file named status
19 echo $? > status
```

There are three important sections of this template that need mentioning, and are described below, by line number:

5-11 Converts a job's text files from DOS to Unix format. This must be included in your template if running SALSSA on a windows platform.

16 This is the invocation command. SALSSA's job script generator will replace the variable *##code##* with the binary executable path provided in the launch dialog along with input parameters based on the the process registration. For example, based on the NWChem registration in the previous section, using *nitrogen.nw* as the input file, and using an executable binary path of */apps/nwchem/bin/nwchem*, the value of *##code##* would be:

*/apps/nwchem/bin/nwchem nitrogen.nw*

The new value for this line would be:

*/apps/nwchem/bin/nwchm nitrogen.nw >& nwchem.out*

*nitrogen.nw* is provided in the *Example Files* section at the end of this document.

19 The status value of the invocation command must be recorded in a file called *status*. SALSSA uses this to report status of the job to the user.

This is a fairly simple template, demonstrating a single execution of NWChem and only containing the one SALSSA variable *##code##*. Examples of more complex templates can be found in the System Registry. The other SALSSA variables that may be included in a job script template are described below:

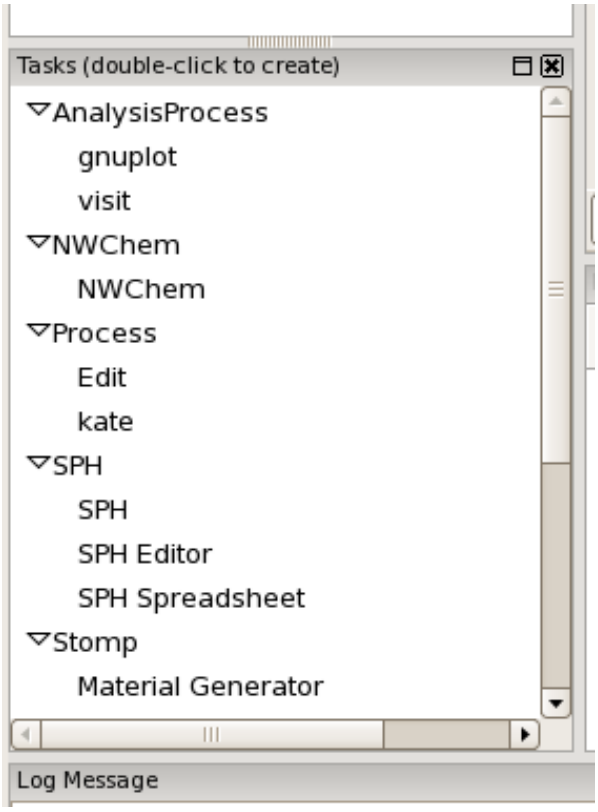
| **SALSSA Variables for all job types** | |
|---|---|
| **Variable** | **Description** |
| ##account## | Allocation account |
| ##code## | Invocation command |
| ##nodes## | Number of nodes to use |
| ##ppn## | Processors per node |
| ##procs## | Total number of processors requested |
| ##runDir## | Directory location where code will run |
| ##walltime## | Wall time limit |
| **SALSSA Variables for Task Parallel jobs** | |
| **Variable** | **Description** |
| ##dirList## | List of run directories for tasks |
| ##nodesTask## | Number of nodes to use per task |
| ##ntasks## | Number of tasks |
| ##procsTasks## | Total number of processors requested per task |

## Compute Machine Configuration

At this point NWChem should be registered with SALSSA, this includes process and file types along with a job script template. In order to launch an NWChem job, a compute machine will need to be configured as part of the launch process.

The following steps outlined below, are a guide to creating a new NWChem job and the configuration of a compute machine supporting its launch:

1. Startup SALSSA, create a new Study and navigate to it.
2. At this point you should see NWChem in the Tasks panel. It may look similar to the screen shot below:




   Double-click the NWChem task to start the wizard.

3. Follow the wizard instructions, stop when you reach the *Launch Page*. When asked to provide an input file, use the sample *nitrogen.nw* provided in the *Example Files* section at the end of this document.
4. On the *Launch Page*, you will need to configure the launch to use the machine where NWChem was installed. In this example, a workstation configuration is used, as opposed to a queued machine. Select either a system registered workstation (e.g.

*penrose*) or user configured workstation and click the *Add* button.

5. Fill in all fields to complete configuration. Ensure *Machine Name* field is the compute machine where NWChem was installed and *NWChem Binary Path* field contains the absolute path location of the NWChem binary on the compute machine.
6. Click *Finish*. This will launch the NWChem job. Job status and results will be reported in SALSSA.

## Example Files

nwchem.ini (NWChem registration file)

```
; NWChem File Types
;
[application/text/nwchem/input]
mime_key=NWCHEM_INPUT
name=NWChem Input
metadata_extractors = [
  Extractor("Title",  "^\s*title\s*\x22(.*)\x22.*$"),
  ]
filename_patterns = [ "*.nw" ]

[application/text/nwchem/output]
mime_key=NWCHEM_OUTPUT
name=NWChem Output
filename_patterns = [ "nwchem.out", ]
file_contents = [ "Northwest Computational Chemistry Package", ]

;
; NWChem Process
;
[process/nwchem]
name=NWChem
process_category=NWChem
remotable=True
inputs=[
  Input(mimetype="NWCHEM_INPUT", mandatory=True)
  ]
widget_class=salssa.graph.nodes.SimNode
outputs = [
  "NWCHEM_OUTPUT"
  ]
primary_output_filename=nwchem.out
```

nitrogen.nw (sample NWChem input file)

```
title "Nitrogen cc-pvdz SCF geometry optimization"
geometry
  n 0 0 0
  n 0 0 1.08
end
basis
  n library cc-pvdz
end
task scf optimize
```

default (SALSSA's default jobscript template)

```
#!/bin/sh
# This is a default job salssa script config file

# Run dos2unix on all text files
for f in *
do
  if [ `file $f |grep -c text` != "0" ]; then
     dos2unix $f 2> /dev/null
  fi
done



# In the simple case, the program simply runs with well known input
# and output file names
# The status must be written to a file named status
##code##
echo $? > status
```

nwchem (sample jobscript template for nwchem)

```
#!/bin/sh
# This is an example job salssa script template
# for NWChem

# Run dos2unix on all text files
for f in *
do
  if [ `file $f |grep -c text` != "0" ]; then
     dos2unix $f 2> /dev/null
  fi
done



# In the simple case, the program simply runs with well known input
# and outputs which have been registered with SALSSA
##code## >& nwchem.out

# The status must be written to a file named status
echo $? > status
```