# EMF Server Communication

## System Requirements

### Python:

The SMOKEServer must have a minimum of Python v2.3.

### tcsh:

The SMOKEServer must have tcsh installed.

### Java

Ideally the EMFServer and SMOKEServer should have common Java versions and locations.

### Home Directory

The ‘emf’ (Linux) user must have a home directory on the EMFServer.

### Shared Disks

Any shared (mounted) disks that are to be used by the EMF should have the same root on both the EMFServer and SMOKEServer. If the EMFServer local disk is called /data_emf, for example, then the disk mounted on the SMOKEServer should also be call /data_emf. In the Installation and Configuration instructions, review the Directory Permissions section.

## SMOKEServer to EMFServer Communication

### Setup

1. From the emf_installation_beta.tar.gz file (previously downloaded and untarred), place EMFClient directory and contents on a shared disk, as mentioned above (We refer to this location as <emf_client_dir> and the URL of your EMF server as <emf_server_host_name>)

2. Edit the EmfCmdClientEnv.csh script as follows:
   ```
   setenv EMF_HOME <emf_client_dir>
   setenv TOMCAT_SERVER http://<emf_server_host_name>:8080
   ```

3. Edit EmfTaskStatusClient.csh script as follows:
   ```
   setenv EMF_HOME <emf_client_dir>
   setenv TOMCAT_SERVER http:// <emf_server_host_name>:8080
   ```

4. Edit EmfRestoreClient.csh script as follows:
   ```
   setenv EMF_HOME <emf_client_dir>
   setenv TOMCAT_SERVER http://localhost:8080
   ```

### Test

Run EmfTaskStatusClient.csh script from the SMOKEServer

```
$ <emf_client_dir>/EmfTaskStatusClient.csh
```

A successful test means that the program has connected to the EMFServer and prints information on any tasks running on the EMFServer, for example:

```
        Starting EMF Task Manager Status Client


        =======================================
        Status of the CaseJobTaskManager


        Tasks in the Wait Table
        JobId,JobName,CaseId,CaseName,UserId,Ready?
        There are no tasks in the CaseJobTaskManager WaitTable
        =======================================
        Tasks in the CaseJobTaskManager RunTable
        JobId,JobName,CaseId,CaseName,UserId
        There are no tasks in the CaseJobTaskManager RunTable
        =======================================


        =======================================
        Status of the ExportTaskManager


        Tasks in the Wait Table
        UserId,DatasetNamevVersion
        There are no tasks in the ExportTaskManager WaitTable
        =======================================
        Tasks in the ExportTaskManager RunTable
```

```
UserId,DatasetName,Version
There are no tasks in the ExportTaskManager RunTable
=======================================


=======================================
Status of the ImportTaskManager


Tasks in the Wait Table
UserId, Dataset Name
There are no tasks in the ImportTaskManager WaitTable
=======================================
Tasks in the ImportTaskManager RunTable
UserId, Dataset Name
There are no tasks in the ImportTaskManager RunTable
=======================================
```

## EMFServer to SMOKEServer Communication

For the EMFServer to communicate properly with the SMOKEServer, the EMF will need to be able to ssh from the EMFServer to the SMOKEServer without being prompted for password authentication. This is achieved through the generation and ‘deployment’ of ssh keys. Once the keys has been generated (steps 1-4 below), the public key will need to be ‘deployed’ (as outlined below) to the SMOKEServer for each EMF user (steps 5-9). That is to say, if you have two EMF users, ‘userA’ and ‘userB’, steps 1-4 will only need to be performed once, while steps 5-9 will need to be performed *twice*. If a particular user is going to run SMOKE (or other model from the EMF), their EMF username *must* be the same as their username on the SMOKEServer.

If your EMFServer and your SMOKEServer are the same machine, we will use the same mechanism as above to run SMOKE. The emf user will use ssh to log onto the EMFServer/SMOKEServer via its public address.

**Complete the following steps:**

1. On the EMFServer, change to the ‘emf’ user:

```
$ su emf
```

2. Look at the contents of the .ssh directory in the ‘emf’ user’s home directory:
   ```
   $ ls ~/.ssh
   ```

   If you see a file named id_rsa.pub, then the ‘emf’ user already has a public key, and you can skip to step 5 below.

3. Make sure the .ssh directory is only ‘rwx’ (‘read/write/execute’) by the ‘emf’ user:
   ```
   $ chmod 700 ~/.ssh
   ```

4. To generate the ssh keys, execute:
   ```
   $ ssh-keygen -t rsa
   ```

   a. You will be prompted to “Enter file in which to save the key”. Hit **enter** to accept the default.
   b. You will then be prompted to “Enter passphrase”. Hit **enter** for no passphrase.
   c. You will finally be prompted to “Enter same passphrase again”. Hit **enter** for no passphrase.
   d. You will be given a message that looks something like:
      ```
      Your public key has been saved in
      /home/emf/.ssh/id_rsa.pub. The key fingerprint is …
      ```

   You should now see two new files in the .ssh directory:
   ```
   id_rsa (private key)
   id_rsa.pub (public key)
   ```

5. Copy the public key (id_rsa.pub) from your EMFServer to the SMOKEServer, as the EMF user (the login name of the user who will be running EMF jobs). For example:
   ```
   $ scp id_rsa.pub azubrow@anubis.cep.unc.edu:/home/azubrow/.ssh
   ```

6. On the SMOKEServer, as the EMF client user, append the id_rsa.pub file to the authorized_keys file in .ssh. For example:
   ```
   $ ssh azubrow@anubis.cep.unc.edu
   $ cat id_rsa.pub >> authorized_keys
   ```

   Note: If the authorized_keys file doesn’t exist, this will create it.

7. Change the permissions on the .ssh dir and the authorized_keys file:
   ```
   $ chmod 700 ~/.ssh
   $ chmod 640 authorized_keys
   ```

8. Delete the id_rsa.pub file

9. Test the connection from the EMFServer to the SMOKEServer:

    a. On the EMFServer, as the ‘emf’ user, ssh to the SMOKEServer. For example:
    ```
    $ ssh azubrow@anubis.cep.unc.edu
    ```

    b. You should NOT be prompted for a password.

## EMF Test Case

### Configuring the Case

1. Download test_EMF.tar.gz and untar the file.

2. If there is not a process queue on the SMOKEServer (e.g., PBS or LSF), copy our fake queue script to a central location. For example, as root:
    ```
    # cp test_EMF/qsub_emf /usr/local/bin
    # chmod a+rx /usr/local/bin/qsub_emf
    ```

3. Move or copy the test_EMF/test_scripts directory and contents to your shared disk (so that they can be accessed by both the EMFServer and SMOKEServer). Make sure the scripts are 'rx' for the emiss group. An example directory for these scripts is:
    ```
    /data4/test_EMF/scripts/test_scripts
    ```

4. Move or copy the test_EMF/test_case_setup to your import directory on the EMFServer. An example directory for the case imports is:
    ```
    /data3/imports/cases/test_case_setup
    ```

5. Modify test_case_setup/inputs/emf_jobheader_generic to match your system:
    ```
    source <emf_client_dir>/EmfCmdClientEnv.csh
    setenv EMF_CLIENT <emf_client_dir>/EmfCmdClient.py
    ```

6. Using the EMF client, import the test_case_setup/inputs/emf_jobheader_generic dataset:

    a. From the EMF client, launch the *Dataset Manager* (*Manage* > *Datasets* from the main menu)

    b. From the *Dataset Manage* window, select ‘EMF Job Header’ from the *Show Datasets of Type* pull-down. This should be empty.

    c. Click the **Import** button

   d. Browse to and select the newly modified emf_jobheader_generic file from step 3 above and click the **OK** button

   e. Click the **Import** button

   f. From the *Status* window, click the **Refresh** button periodically until you see a message stating the import completed (It should actually only require one click, as the import should be fast for such a small file)

   g. From the *Import Dataset* window, click the **Done** button

   h. From the *Dataset Manage* window, verify that your newly imported dataset is there (clicking the **Refresh** button if necessary)

   i. Close the *Dataset Manage* window

7. Import the test Case:
   a. From the EMF client, launch the *Case Manager* (*Manage* > *Cases* from the main menu)

   b. Click the **Import** button

   c. Browse to the test_case_setup directory and select the following files:
   `test_EMF_setup_test_setup_Inputs.csv`
   `test_EMF_setup_test_setup_Jobs.csv`
   `test_EMF_setup_test_setup_Summary_Parameters.csv`

   d. Click the **Import** button

   e. Verify that the message ’Successfully imported the case.’ appears at the top of the window

   f. Click the **Done** button

   g. From the *Case Manage* window, select ‘Demo’ from the *Show Cases of Category* pull-down and verify that your newly imported case (‘test EMF setup’) is there

8. You will now need to modify the imported case to match your system

   a. Select the new case and click the **Edit** button

b. From the *Jobs* tab, edit the **Output Job Scripts Folder** field. An example directory is:
`/data4/test_EMF/scripts/cases/$CASE`

The EMF server will replace the variable $CASE by the case abbreviation. It will create this subdirectory when the first job is run. Make sure that the rest of the directory tree has been created and has the proper permissions (see Directory Permissions).

c. Select both jobs and click the **Edit** button (this will open two *Edit Case Job* windows)

d. For each job, modify:

   i. the **Executable** field to match the location of the test_scripts directory

   ii. the **Host** field to match the SMOKEServer host name. This is the URL or IP address of your SMOKEServer. The first time you do this, you will need to type into the field. Subsequent times, you can select the host from the pull-down. If your EMFServer and SMOKEServer are the same machine, set the host to the public address of the machine, *not* localhost.

   iii. the **Queue Options** field. If you are using a process queue on your SMOKEServer, you will provide the queue command and additional options here. An example using a PBS queue:

   ```
      qsub -o $EMF_JOBLOG $EMF_QUEUE_OPTIONS
   ```

   If you are not using a process queue, then you should use the fake queue script provided:

   ```
      qsub_emf $EMF_JOBLOG
   ```

   The variable $EMF_JOBLOG is automatically filled in by the EMF. This is the file that captures stdout and stderr of the job. The variable $EMF_QUEUE_OPTIONS is defined in the parameters tab.

   iv. click the **Save** button

e. From the *Inputs* tab, modify the **Input Folder** field. An example directory is:
`/data4/test_EMF/inputs/$CASE`

The EMF server will replace the variable $CASE by the case abbreviation. It will create this subdirectory when the first Job is run. Make sure that the rest of the directory tree has been created and has the proper permissions (see Directory Permissions).

f. Make sure that the 'emf job header' input is selected and click the **Edit** button

g. From the *Edit Case Input* window, click the **Select** button (next to the **Dataset** field)

h. From the *Select EMF Job Header* window, select the dataset 'emf_jobheader_generic' and click the **OK** button

i. From the *Edit Case Input* window, verify that the **Dataset** field has been correctly updated and click the **Save** button

j. If you are using a real process queue, go to the *Parameters* tab, select the 'EMF queue options' parameter and click the **Edit** button. If you are using the fake queue, the value of the 'EMF queue options' does not matter.

k. From the *Edit queue options* window, edit the **Value** field. If you are using a process queue, this is where you would place additional options such as user permissions, estimated time of run, etc. After editing the value, click **Save**.

## Running a Job

Once the Case (Jobs, Inputs, and Parameters) have been properly configured, to run a Job, simply return to the *Jobs* tab, select the Job you wish to run and click the **Run** button. Once the **Run** button has been clicked a confirmation dialog will appear; click the **OK** button.

## Job Status and Files

The EMF will generally not update a Job's state or status messages, so the user will be required to click the **Refresh** button from time to time. A successful run will go through the following progression:

1. *Not Started* - before a Job has been run
2. *Exporting* - EMF is exporting all the Inputs necessary for that specific Job
3. *Waiting* - Job is sitting in the EMF internal queue waiting for all exports to complete and/or and Job dependencies
4. *Submitted* - Job has been submitted to the SMOKEServer's process queue

5. *Running* - Job has started on the SMOKEServer

6. *Completed* - Job has completed its run on the SMOKEServer

If there is an error that occurs in the *Exporting*, *Waiting*, or *Running* stages, the status will be set to *Failed*.  If you have previously run a Job and do not change any Inputs in the interim, a second run of a Job will skip the *Exporting* stage.

The EMFServer will create a Job wrapper script that sets up the necessary environment for the Job.  This wrapper script is a csh script that includes EMF related environment variables and environment variables from the Summary, Jobs, Inputs, and Parameters tabs.  As the Job wrapper script runs on the SMOKEServer, stdout and stderr are redirected to the Job specific log file.  Both of these files are tagged with job name, an ID, and a datestamp.  If a Job is rerun, a new Job wrapper script and log is created.  Based on the above directory structure, the Job wrapper scripts and log files would be located in:

```
/data4/test_EMF/scripts/cases/test_setup
```

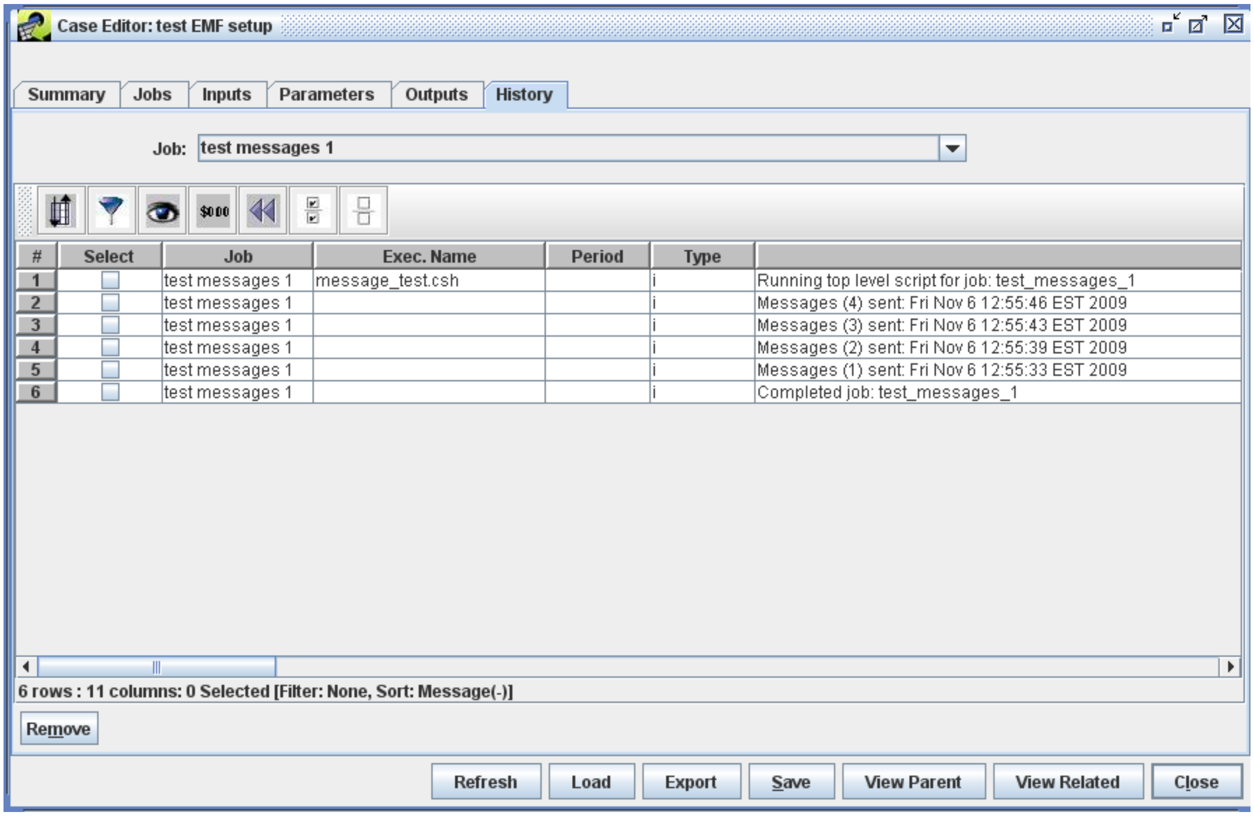
```
/data4/test_EMF/scripts/cases/test_setup/logs
```

### Expected Messages

More detailed information concerning a Job can be seen in the **History** tab.  The underlying scripts (the executable) have been developed to send back information on the progression of the program, status changes, and some error messages if they do occur.

Once the **History** tab has been selected, select the Job in which you are interested from the **Job** pull-down (selecting ‘All’ will display all messages). Clicking the **Refresh** button from time to time will display any new messages if the Job has not yet completed the run.

Below is a screenshot of the expected messages from a successful run of the "test messages 1" Job:

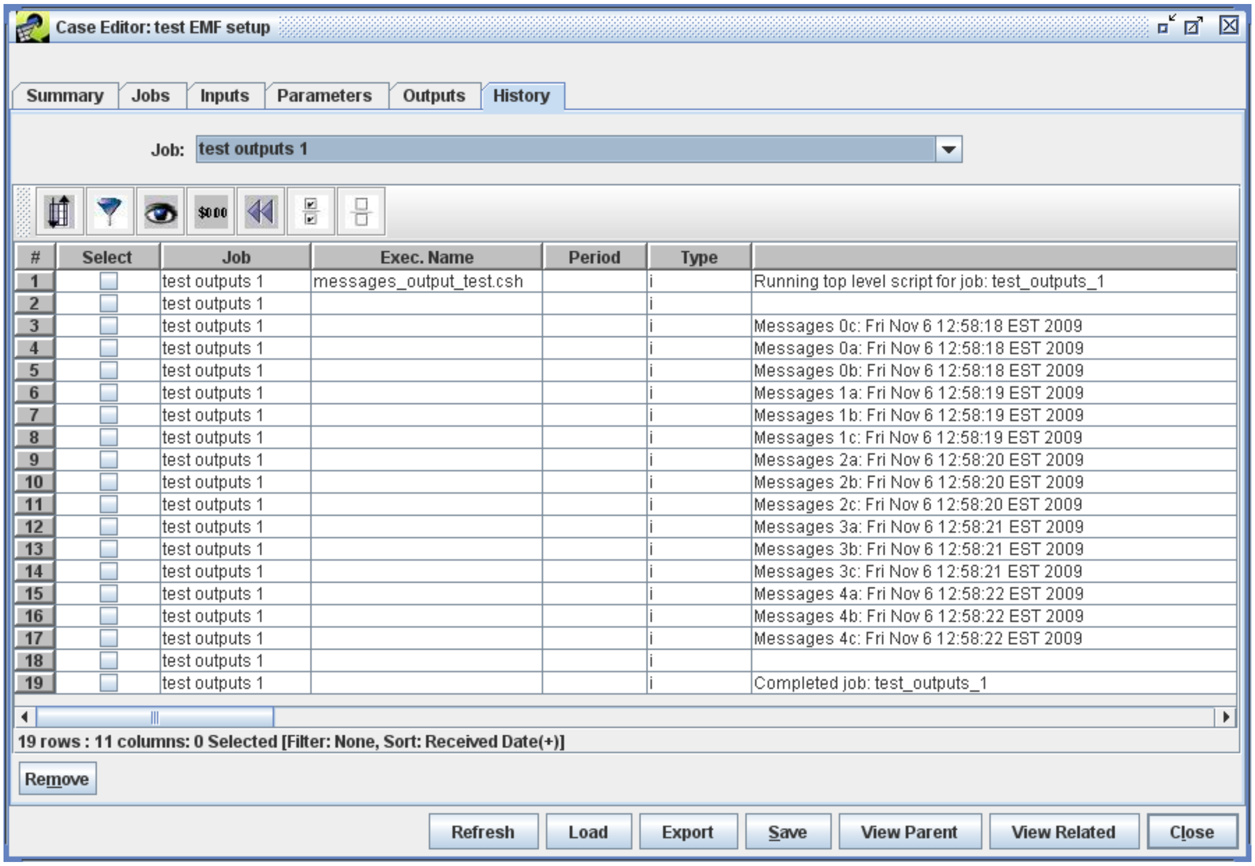


Job: test messages 1

| # | Select | Job | Exec. Name | Period | Type | |
|---|---|---|---|---|---|---|
| 1 | □ | test messages 1 | message_test.csh | | i | Running top level script for job: test_messages_1 |
| 2 | □ | test messages 1 | | | i | Messages (4) sent: Fri Nov 6 12:55:46 EST 2009 |
| 3 | □ | test messages 1 | | | i | Messages (3) sent: Fri Nov 6 12:55:43 EST 2009 |
| 4 | □ | test messages 1 | | | i | Messages (2) sent: Fri Nov 6 12:55:39 EST 2009 |
| 5 | □ | test messages 1 | | | i | Messages (1) sent: Fri Nov 6 12:55:33 EST 2009 |
| 6 | □ | test messages 1 | | | i | Completed job: test_messages_1 |

6 rows : 11 columns: 0 Selected [Filter: None, Sort: Message(-)]

Below is a screenshot of the expected messages from a successful run of the "test outputs 1" Job:



Job: test outputs 1

| # | Select | Job | Exec. Name | Period | Type | |
|---|---|---|---|---|---|---|
| 1 | □ | test outputs 1 | messages_output_test.csh | | i | Running top level script for job: test_outputs_1 |
| 2 | □ | test outputs 1 | | | i | |
| 3 | □ | test outputs 1 | | | i | Messages 0c: Fri Nov 6 12:58:18 EST 2009 |
| 4 | □ | test outputs 1 | | | i | Messages 0a: Fri Nov 6 12:58:18 EST 2009 |
| 5 | □ | test outputs 1 | | | i | Messages 0b: Fri Nov 6 12:58:18 EST 2009 |
| 6 | □ | test outputs 1 | | | i | Messages 1a: Fri Nov 6 12:58:19 EST 2009 |
| 7 | □ | test outputs 1 | | | i | Messages 1b: Fri Nov 6 12:58:19 EST 2009 |
| 8 | □ | test outputs 1 | | | i | Messages 1c: Fri Nov 6 12:58:19 EST 2009 |
| 9 | □ | test outputs 1 | | | i | Messages 2a: Fri Nov 6 12:58:20 EST 2009 |
| 10 | □ | test outputs 1 | | | i | Messages 2b: Fri Nov 6 12:58:20 EST 2009 |
| 11 | □ | test outputs 1 | | | i | Messages 2c: Fri Nov 6 12:58:20 EST 2009 |
| 12 | □ | test outputs 1 | | | i | Messages 3a: Fri Nov 6 12:58:21 EST 2009 |
| 13 | □ | test outputs 1 | | | i | Messages 3b: Fri Nov 6 12:58:21 EST 2009 |
| 14 | □ | test outputs 1 | | | i | Messages 3c: Fri Nov 6 12:58:21 EST 2009 |
| 15 | □ | test outputs 1 | | | i | Messages 4a: Fri Nov 6 12:58:22 EST 2009 |
| 16 | □ | test outputs 1 | | | i | Messages 4b: Fri Nov 6 12:58:22 EST 2009 |
| 17 | □ | test outputs 1 | | | i | Messages 4c: Fri Nov 6 12:58:22 EST 2009 |
| 18 | □ | test outputs 1 | | | i | |
| 19 | □ | test outputs 1 | | | i | Completed job: test_outputs_1 |

19 rows : 11 columns: 0 Selected [Filter: None, Sort: Received Date(+)]

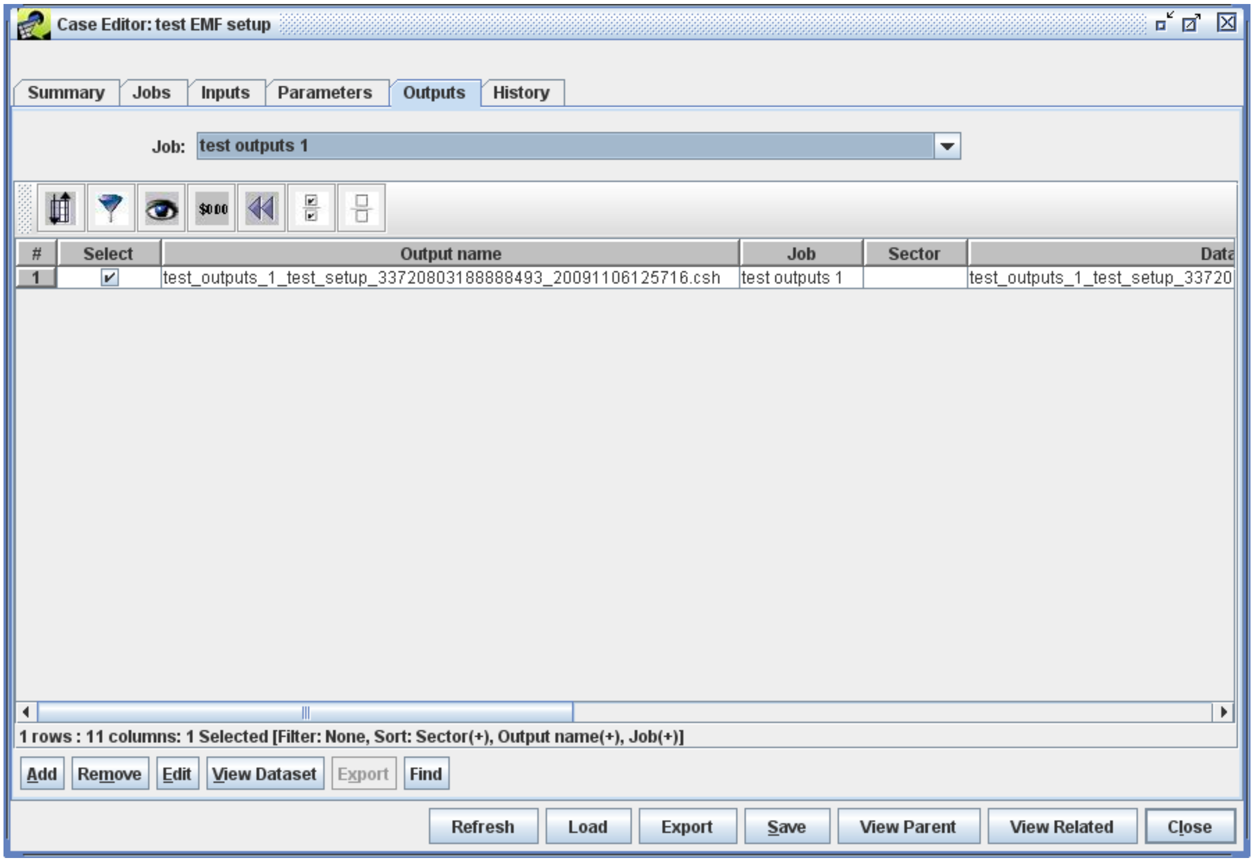
## Expected Outputs

In addition to messages, a Job can import new datasets into the EMF.  These new datasets are associated with Case outputs (similar to how a dataset is associated to an Input).

Once the **Output** tab has been selected, select the Job in which you are interested from the **Job** pull-down (selecting ‘All’ will display all outputs). Clicking the **Refresh** button from time to time will display any new outputs if the Job has not yet completed the run.

The "test messages 1" Job does not produce any outputs.

Below is a screenshot of the expected outputs from a successful run of the "test outputs 1" Job:



Job: test outputs 1

| # | Select | Output name | Job | Sector | Data |
|---|---|---|---|---|---|
| 1 | ☑ | test_outputs_1_test_setup_33720803188888493_20091106125716.csh | test outputs 1 |  | test_outputs_1_test_setup_33720 |

1 rows : 11 columns: 1 Selected [Filter: None, Sort: Sector(+), Output name(+), Job(+)]

## Troubleshooting

The most common source of errors is a problem with user permissions. Both the emf user on the EMFServer and the user who is actually running the Job wrapper script on the SMOKEServer need permission to read, write, and execute in the common directories and files.

Another possible problem is that the ssh-keys have not been properly setup. Make sure for each new user who plans to run on the SMOKEServer that the emf user can login via ssh without a password.

Other areas to search for the cause of an error:

1. The History tab. For known errors, the underlying scripts will attempt to send back error information.
2. The EMF status window. This will generally indicate errors in exporting the Inputs.
3. The Job wrapper log file.
4. Other log files created. For example, SMOKE intermediary log files.

If none of these sources help identify the cause of an error, consult the apache-tomcat log, /usr/local/apache-tomcat-6.0.20/logs/catalina.out