IBM Platform Symphony RFE55194 and RFE12598 Readme File

About RFE55194 (Replacing hardcoded information with variables in `client.py`)
RFE55194 removes hardcoded user names, passwords, and application names in `client.py` and adds additional exceptions for the user. With this feature, you can configure the user name, password, and application name using variables. This feature also provides JobCancelled and UnhandledRemoteError exceptions.

About RFE12598 (Resolving conflicts for STDOUT and STDIN in Python wrapper (pythonwrapperservice.py) with SWIG generated C ++ library)
RFE12598 provides socket communication between service side processes. This feature allows service side standard ouput and standard input messages.

Readme file for: IBM® Platform Symphony
Product/Component Release: Symphony 6.1.1
Update Name: Replacing hardcoded information with variables in client.py, and resolving conflicts for STDOUT and STDIN in Python wrapper (pythonwrapperservice.py) with SWIG generated C ++ library
Fix ID: sym-6.1.1-build-240101-pimco
Publication date: 29 Aug 2014
Last modified date: 29 Aug 2014

1 Scope

Applicability	
Operating system	Linux2.6-glibc2.3-x86_64
Python version	2.4.5 64 bit, 2.7.2 64 bit
Symphony version	6.1.1
Cluster types	This feature applies to a single grid cluster or DE.
Other	This feature applies to SOAM.
Dependencies	
File system	This feature has no requirement on the file system type.
<Other>	Python 2.4.5 64 bit or Python 2.7.2 64 bit version must be installed on the client host and all compute hosts.
Limitations	
<Limitation>	None.
Known Issues	None.

2 Configuration

2.1 Prerequisites

Python 2.4.5 64 bit or Python 2.7.2 64-bit must be installed on the client host and all compute hosts.

2.2 Installation files

This package includes the following files:

File name	Description
`sympython-wrapper-lnx26-lib23-x64-6.1.1_240101.tar.gz`	This package contains this new feature for Linux x86_64.

2.3 Installation procedure

1. Download the `sympython-wrapper-lnx26-lib23-x64-6.1.1_240101.tar.gz` package from the IBM web site.
2. Decompress the package: `sympython-wrapper-lnx26-lib23-x64-6.1.1_240101.tar.gz.`

   For example, run:
   ```
   tar xzvf sympython-wrapper-lnx26-lib23-x64-6.1.1_240101.tar.gz
   ```
3. Copy the `python files` to `$GRID_DIR_LOCATION` and replace the exising files:
   ```
   cp client.py $GRID_DIR_LOCATION/grid/
   cp service/pythonwrapperservice.py $GRID_DIR_LOCATION/grid/service/
   cp service/invokeservice.py $GRID_DIR_LOCATION/grid/service/
   ```

2.4 Configuration procedure

2.4.1 Configure RFE55194

Set variables for the user name, password, and application name:

1. Open a user module, such as `test_apply.py`.
2. Configure the application name. The default is `"PythonApp"`.

   For example:
   ```
   grid.client.APPLICATION_NAME = "PythonApp"
   ```
3. Configure the user name and password. The default for both is `"Guest"`.

   For example:
   ```
   grid.client.USERNAME = "Guest"
   grid.client.PASSWORD = "Guest"
   ```
4. Save the file.

Add custom code for `client.py`:

1. Create a new file in `GRID_DIR_LOCATION/grid/` (for example, called `custom.py)`, and add your code in this file
2. Add `import custom` in `client.py` to load the `custom.py` module.

2.4.2 Configure RFE12598

1. Go to the service directory and deploy the service package:
   a) `cd service`
   b) `./makepackage.sh`
2. Set the USE_SOCKET environment variable to Y in the "Service" section of application profile.
   For example:

   <env name="USE_SOCKET">Y</env>
3. Register the application:
   `soamreg ../PythonApp.xml`

2.5 Verification procedure

1. Configure the application name:
   a. Open the `sample/test_apply.py` file in the grid location:
      `vi sample/test_apply.py`
   b. Add `grid.client.APPLICATION_NAME = “”AppName”`

   For example:

```
...
def main_function():
  grid.client.USE_LOCAL_MAP_APPLY = False
  grid.client.APPLICATION_NAME = “AppName”
       submitter = grid.client.Client(max_slots=1,
   import_local_env=False, debug_level=logging.INFO,
   debug_stdio=sys.stdout)
...
```

   c. Save the file.
   d. Register the application:
      `soamreg AppName.xml`
   e. Run the user module:
      `Python sample/test_apply.py`

2. Configure the user name and password:
   a. Open the `sample/test_apply.py` file in the grid location:
      `vi sample/test_apply.py`
   b. Add `grid.client.USERNAME = "Guest" grid.client.PASSWORD = "Guest"`

   For example:

```
...
def main_function():
  grid.client.USE_LOCAL_MAP_APPLY = False
  grid.client.USERNAME = "Guest"
  grid.client.PASSWORD = "Guest"
       submitter = grid.client.Client(max_slots=1,
   import_local_env=False, debug_level=logging.INFO,
   debug_stdio=sys.stdout)
...
```

   c. Save the file.
   d. Run the user module:
      `Python sample/test_apply.py`

3. Add custom code for `client.py`:

a. Create a new file in the grid location:

```
vi custom.py
```

b. Add custom code to `custom.py` such as JobCancelled and UnhandledRemoteError exceptions.

For example:

```
class JobCancelled(Exception):
    pass

class UnhandledRemoteError(Exception):
    pass

    def __str__(self):
        if False:
            self.args = None
        if len(self.args)>1 and type(self.args[-1]) == str:
            return  str(self.args[:-1])+'\n'+str(self.args[-1])
        else:
            return super(UnhandledRemoteError,self).__str__()
```

c. Add import custom to `client.py`:

```
...
import custom
...
```

d. Call the custom code in `map_apply.py`:

```
...
raise grid.custom.JobCancelled, 'test'
...
```

4. Echo messages to stdout on the service side:

a. Create `test.py` under grid location:

```
#!/usr/bin/python
import grid.client
import logging
import sys
def f(x):
    import os
    print "echo hello"
    return x

c = grid.client.Client()
for x in c.imap_apply(f, (range(1,3))):
    print 'hello'
    print x
```

b. Run `test.py` to ensure you can successfully print the output:

```
# python test.py
hello
1
hello
2
# cat $EGO_TOP/soam/work/PythonApp_21760_WorkerService.out
echo hello
```

3 Usage

3.1 How the RFE55194 feature works

This feature provides three variables to configure the application name, user name, and password. Configure them in the user modules as follows:

```
grid.client.APPLICATION_NAME = "AppName"
grid.client.USERNAME = "Guest"
grid.client.PASSWORD = "Guest"
```

The default value of `APPLICATION_NAME` is `"PythonApp"`. The default value of `USERNAME` and `PASSWORD` are both `"Guest"`.

3.2 How the RFE12598 feature works

This feature provides socket communication between service side processes.
Originally, on the service side, STDOUT and STDIN are used to exchange data in a special format; echoing messages to STDOUTbreaks that protocol. After enabling this feature, echoing message to STDOUT or STDIN will not impact workload.

4 Copyright and trademark information