



DIRAC API

DIRAC Project

- ▶ **DIRAC API**
 - ▶ Why APIs are important?
 - ▶ Why advanced users prefer APIs?
 - ▶ How it is done?
 - ▶ What is local mode what is not?
- ▶ **Tutorial exercises**
- ▶ **What do you have learned?**



Why APIs are important?

- ▶ Application Programming Interface (API)
- ▶ The DIRAC API is encapsulated in several Python classes designed to be used easily by users to access a large fraction of the DIRAC functionality. Using the API classes it is easy to write small scripts or applications to manage user jobs and data.
- ▶ The DIRAC API provides a transparent and secure way for users to submit jobs to the Grid.
- ▶ Permit debug locally the programs before submit jobs to the Grid.
- ▶ While it may be exploited directly by users, the DIRAC API also serves as the interface for the Ganga Grid front-end to perform distributed user analysis for LHCb, for example.



Why advanced users prefer APIs

- ▶ API provides a simple, seamless interface to Grid resources to submit jobs that can be:
 - ▶ Single applications
 - ▶ Multiple steps of different applications
- ▶ Can perform a typical user analysis using understandable Python code



- ▶ The API allows creating DIRAC jobs using the Job object.
- ▶ Example:

```
from DIRAC.Interfaces.API.Job import Job
from DIRAC.Interfaces.API.Dirac import Dirac
dirac = Dirac()
j = Job()
j.setExecutable('ls',arguments='-al',logFile='ls.log')
dirac.submit(j)
```

Note that all the DIRAC API commands described here may also be executed directly from the Python prompt.

Setting the computing resources to use:

▶ **Operating system and system architecture:**

```
j.setSystemConfig("x86_64-slc5-gcc43-opt")
```

▶ **CPU requirement determined:**

```
j.setCPUTime(21600)
```

▶ **Site name:**

```
j.setDestination('LCG.KEK.jp')
```

▶ **Banned sites:**

```
j.setBannedSites(['LCG.CERN.ch','LCG.DESY.de'])
```



Configurable Job Parameters

▶ Multiple Jobs:

```
for i in range(20):  
    j.setName('MyJob_%d' % i)  
    dirac.submit(j)
```

▶ Log Level:

```
j.setLogLevel('DEBUG')
```

▶ Environment variables:

```
j.setExecutionEnv({'MYVARIABLE':'TOTO'})
```



▶ Input Sandbox

```
j.setInputSandbox(['Input.options','*.txt','lib/'])
```

▶ Output Sandbox

```
j.setOutputSandbox(['*.log','summary.data'])
```

▶ Input Data

```
j.setInputData(['/ilc/user/s/sposs/somefile'])
```

▶ Output Data

```
j.setOutputData(['output1.data','output2.data'],outputSE=['KEK-  
SRM'],outputPath='MyFirstAnalysis')
```





APIs Example

```
from DIRAC.Interfaces.API.Job import Job
from DIRAC.Interfaces.API.Dirac import Dirac
j = Job()
dirac = Dirac()
j.setName('MyFirstJob')
j.setOutputSandbox(['*.log', 'summary.data'])
j.setInputData(['/my/logical/file/name1',
               '/my/logical/file/name2'])
j.setOutputData(['output1.data', 'output2.data'],
               outputPath='MyFirstAnalysis')
j.setSystemConfig("x86_64-slc5-gcc43-opt")
j.setCPUTime(21600)
j.setDestination('LCG.KEK.jp')
j.setBannedSites(['LCG.CERN.ch', 'LCG.DESY.de'])
j.setLogLevel('DEBUG')
j.setExecutionEnv({'MYVARIABLE': 'TOTO'})
j.setExecutable('echo $MYVARIABLE')
print j._toJDL()
dirac.submit(j)
```



What is local mode? What is not?

- ▶ Local mode doesn't send the job to the Grid, the execution is not remote.
- ▶ The Local submission mode is a very useful tool to check the sanity of your job before submission to the Grid.
- ▶ The job executable is run locally in exactly the same way (same input, same output) as it will do on the Grid Worker Node.
- ▶ This allows to debug the job in a friendly local environment.





Local Mode Example

```
$ python
Python 2.5.5 (r255:77872, Mar 25 2010, 14:17:52) [GCC 4.1.2
20080704 (Red Hat 4.1.2-46)] on linux2 Type "help", "copyright",
"credits" or "license" for more information.
>>> from DIRAC.Interfaces.API.Dirac import Dirac
>>> from DIRAC.Interfaces.API.Job import Job
>>> j = Job()
>>> j.setExecutable('/bin/echo hello')
{'OK': True, 'Value': ''}
>>> Dirac().submit(j,mode='local')
```

- ▶ Exercises can be found at:
 - ▶ <https://github.com/DIRACGrid/DIRAC/wiki/JobManagementAdvanced>
- ▶ Run multiple Mandelbrot jobs, get output data files
- ▶ Prepare a Mandelbrot job with multiple input data files



What do you have learnt?

- ▶ How to submit jobs using APIs.
- ▶ Debug the payload before submitting jobs to the grid
- ▶ Handle job management using APIs

