PyDMLite & DMLite Shell

MY OPENLAB SUMMER STUDENT PROJECT 2012

Georg Jahn
georg.jahn@cern.ch

openlab Summer Student Program 2012
CERN IT-GD-ITR

Outline

- The PyDMLite Python-Module
  - Implementation & Structure
  - How-To & Examples
  - Maintenance
- DMLite Shell
  - Implementation & Structure
  - Demo & Available Functionality
  - How to add a new command?
- Prospect

The PyDMLite Python Module

- Python bindings for the DMLite library
  - Implemented with Boost.Python.
  - Compliant with DMLite API version 20120817.
  - ~98% of the functionality available in Python 2.4.
  - Structure of the library untouched.
- Source available in the folder /python.
- Module name is pydmlite.

PyDMLite Structure

/python/

pydmlite.cpp	basic functionality, includes all other files
authnWrapper.cpp	wrapper for pure virtual classes of authn.h
authn.cpp	bindings for classes of authn.h
baseWrapper.cpp	~ of base.h
base.cpp	~ of base.h
...	...
helpers.cpp	functionality unrelated to any .h-file
CMakeLists.txt	make file with install functionality

PyDMLite How-To

- After installation via make install:
  - pydmlite.so file installed in Python module folder
- Accessible via import pydmlite:

```
% python
Python 2.4.3 (#1, Jun 19 2012, 13:51:22)
>>> import pydmlite
>>> pydmlite.API_VERSION
20120817
>>> dir(pydmlite)
['API_VERSION', 'Acl', 'AclEntry', 'Authn',
'AuthnFactory', 'BaseFactory', 'BaseInterface',
'Catalog', 'CatalogFactory', 'Chunk', ...
```

PyDMLite Example

- Naming scheme of the library is largely untouched, only small pythonic and indexing changes:

```
// C++ code
dmlite::SecurityContext root;
root.user["uid"] = 0u;
root.groups.push_back(group);

# Python code
root = pydmlite.SecurityContext()
root.user.setUnsigned("uid", 0)
root.groups.append(group)
```

PyDMLite Example

```
>>> pluginManager = pydmlite.PluginManager()
>>> pluginManager.loadConfiguration("/etc/dmlite.conf")
>>> root = pydmlite.SecurityContext() # build a Security Context
>>> group = pydmlite.GroupInfo()
>>> group.name = "root"
>>> group.setUnsigned("gid", 0)
>>> root.user.setUnsigned("uid", 0)
>>> root.groups.append(group)
>>> stackInstance = pydmlite.StackInstance(pluginManager)
>>> stackInstance.setSecurityContext(root)
>>> catalog = stackInstance.getCatalog() # access file catalog
>>> catalog.changeDir("/dpm")
>>> catalog.getWorkingDir()
'/dpm'
>>> catalog.create("test-file-a", 0755) # create a test file
>>> catalog.rename("test-file-a", "test-file-b")
>>> stat = catalog.extendedStat("test-file-b", False)
>>> stat.name
'test-file-b'
>>> stat.status
pydmlite.FileStatus.kOnline
>>> catalog.unlink("test-file-b")
```

Available on the wiki!

PyDMLite Maintenance

- Maintenance absolutely required!
  - When DMLite API changes, these changes have to be reflected in PyDMLite.
- File structure makes maintenance relatively easy.

DMLite Shell

- Goal:

  Easy testing & debugging of DMLite functionality.

- Vision:

  Bash-like interactive shell for DMLite commands.

- Outcome:

  DMLite shell (Wiki-Link)

  - Written in Python using PyDMLite.
  - Access to big parts of the DMLite functionality.
  - Can be found in folder /python/shell.

DMLite Shell Structure

/python/shell/

- dmliteshell.py

  Executable file creating a shell environment.

- dmliteinterpreter.py

  Actual interpretation and execution of the commands.

DMLite Demo

```
% ./dmliteshell.py
DMLite shell v0.1 (using DMLite API v20120817)
Using configuration "/etc/dmlite.conf" as root.
> ls /dpm/cern.ch/home/
atlas                (dir)
cms                  (dir)
dteam                (dir)
testers.eu-emi.eu    (dir)
> help
 acl          Sets or reads the ACL of a file.
 addreplica   Adds a new replica for a file.
 cd           Changes the current directory.
 chmod        Changes the mode of a file.
 …
```

DMLite Functionality

- Commands implemented so far give access to:
  - Basic folder navigation (ls, cd, pwd, mkdir, rmdir, ...)
  - Basic file manipulation (create, unlink, chmod, mv, ...)
  - Extended File Attributes (info, acl, comment)
  - Replica manipulation (addreplica, chreplica)
  - User / Group management (userinfo, groupinfo)

How to add a new shell command?

- Extremely easy to add in dmliteinterpreter.py:

```
class MkDirCommand(ShellCommand):
    """Creates a new directory."""
    def _init(self):
        self.parameters = ['ddirectory']

    def _execute(self, given):
        try:
            self.interpreter.catalog.makeDir(given[0], 0777)
        except Exception, e:
            return self.error(e.__str__() + given[0])
        return self.ok('Created directory.')
```

Prospect

- Everything can be found in the branch pydmlite.
- Both pyDMLite and DMLite shell working fine.
- Development on DMLite shell already lead to the discovery of several bugs.
- DMLite Shell scripting possible through command line options!