GFD-I.143
DRMAA-WG

Peter Tröger, Blekinge Institute of Technology
Martin v. Löwis, Hasso-Plattner-Institute
Enrico Sirola, StatPro
January 10, 2009

Distributed Resource Management Application API 1.0 – Python Language Binding

Status of This Document

This document provides information to the Grid community. Distribution is unlimited.

Copyright Notice

Abstract

This document describes the representation of the DRMAA 1.0 API in the Python programming language. It is based on the DRMAA 1.0 IDL recommendation (GFD-R-P.130), and maps the DRMAA IDL interface definition to specific Python language constructs.

Contents

1. Introduction

This document describes the representation of the DRMAA 1.0 API in the Python programming language. It is based on the DRMAA 1.0 IDL specification, which contains the detailed functionality description for each method of the API. The rules for language bindings are described in Section 2.2 of the DRMAA 1.0 IDL specification [GFD130].

2. Python Language Mapping for DRMAA

A Python module implementation can declare “DRMAA 1.0-compliance” if it realizes the API signature described in the following sections, and the functional behavior as described in [GFD130]. Additional module functionality beside the specified API is allowed, but must be clearly identifiable (e.g. by a function name convention).
The following table provides the basic mapping overview for the DRMAA IDL constructs to the Python programming language:

DRMAA 1.0 IDL specification	*DRMAA 1.0 Python binding*
`module` definition	Python module file named “drmaa.py”
`interface` definition	`class` definition
`enum` definition with enumeration members	`class` definition
`string` type	`str`
`long` type	`int`
`long long` type	`long`
`const` definition	Pre-defined `class` attributes
`boolean` type	`bool`
`[readonly] attribute` definition	`class` attributes
`exception` definition	Class definition derived from `Exception`, or mapping to in-build Python exception; `message` parameter maps to `args` attribute
`raises` clause	Not mapped
`valuetype` definition	`class` definition with `__cmp__(self,other)` method
`StringList`	Python list of `str` objects
`OrderedStringList`	Python list of `str` objects
`TimeAmount`	`long`
`Dictionary`	`dict`
`PartialTimestamp`	`str`

The resulting Python module signature MUST be realized by all implementations. The `pass` statements should be replaced by an according implementation. It is RECOMMENDED to add the IDL specification description text as documentation to the DRMAA Python module functions.

```
"""This is drmaa.py, implementing the DRMAA Python language binding
   Visit www.drmaa.org for details"""

# Job control action
class JobControlAction:
    SUSPEND='suspend'
    RESUME='resume'
    HOLD='hold'
    RELEASE='release'
    TERMINATE='terminate'

# State of single job
class JobState:
    UNDETERMINED='undetermined'
    QUEUED_ACTIVE='queued_active'
    SYSTEM_ON_HOLD='system_on_hold'
    USER_ON_HOLD='user_on_hold'
    USER_SYSTEM_ON_HOLD='user_system_on_hold'
    RUNNING='running'
    SYSTEM_SUSPENDED='system_suspended'
    USER_SUSPENDED='user_suspended'
    USER_SYSTEM_SUSPENDED='user_system_suspended'
    DONE='done'
    FAILED='failed'

# State at submission time
class JobSubmissionState:
    HOLD_STATE='hold'
    ACTIVE_STATE='active'

class FileTransferMode:
    transferInputStream=False
    transferOutputStream=False
    transferErrorStream=False
    def __cmp__(self,other):
        pass

class Version:
    major='1'
    minor='0'
    def __cmp__(self,other):
        pass
    def __str__(self):
        return self.major+'.'+self.minor

class AlreadyActiveSessionException(Exception):
    pass
class AuthorizationException(Exception):
    pass
class ConflictingAttributeValuesException(Exception):
    pass
class DefaultContactStringException(Exception):
```

```
    pass
class DeniedByDrmException(Exception):
    pass
class DrmCommunicationException(Exception):
    pass
class DrmsExitException(Exception):
    pass
class DrmsInitException(Exception):
    pass
class ExitTimeoutException(Exception):
    pass
class HoldInconsistentStateException(Exception):
    pass
class IllegalStateException(Exception):
    pass
class InternalException(Exception):
    pass
class InvalidAttributeFormatException(Exception):
    pass
class InvalidContactStringException(Exception):
    pass
class InvalidJobException(Exception):
    pass
class InvalidJobTemplateException(Exception):
    pass
class NoActiveSessionException(Exception):
    pass
class NoDefaultContactStringSelectedException(Exception):
    pass
class ReleaseInconsistentStateException(Exception):
    pass
class ResumeInconsistentStateException(Exception):
    pass
class SuspendInconsistentStateException(Exception):
    pass
class TryLaterException(Exception):
    pass
class UnsupportedAttributeException(Exception):
    pass
InvalidArgumentException=TypeError
InvalidAttributeValueException=ValueError
OutOfMemoryException=MemoryError

class JobInfo:
    def __init__(self):
        self.resourceUsage={}
        self.jobId=''
        self.hasExited=False
        self.exitStatus=0
        self.hasSignaled=False
        self.terminatingSignal=''
        self.hasCoreDump=False
        self.wasAborted=False

class JobTemplate:
    HOME_DIRECTORY='$drmaa_hd_ph$'
```

```
    WORKING_DIRECTORY='$drmaa_wd_ph$'
    PARAMETRIC_INDEX='$drmaa_incr_ph$'
    # contains list of strings
    attributeNames=[]
    def __init__(self):
        self.remoteCommand="
        self.jobEnvironment={}
        self.workingDirectory="
        self.jobCategory="
        self.nativeSpecification="
        self.blockEmail=False
        self.jobName="
        self.inputPath="
        self.outputPath="
        self.errorPath="
        self.joinFiles=False
        # contains list of strings
        self.args=[]
        # contains one of the job submission state values, or None
        self.jobSubmissionState=None
        # contains list of strings
        self.email=[]
        # [[[[CC]YY/]MM/]DD]hh:mm[:ss] [{-|+}UU:uu]
        self.startTime="
        self.deadlineTime="
        # contains FileTransferMode instance, or None
        self.transferFiles=None
        # in seconds
        self.hardWallclockTimeLimit=0
        self.softWallClockTimeLimit=0
        self.hardRunDurationLimit=0
        self.softRunDurationLimit=0

class Session:
    TIMEOUT_WAIT_FOREVER=-1
    TIMEOUT_NO_WAIT=0
    JOB_IDS_SESSION_ANY='DRMAA_JOB_IDS_SESSION_ANY'
    JOB_IDS_SESSION_ALL='DRMAA_JOB_IDS_SESSION_ALL'
    version=Version()
    contact=''
    drmsInfo=''
    drmaaImplementation=''

    # no return value
    def initialize(self, contactString="):
        pass
    # no return value
    def exit(self):
        pass
    # returns JobTemplate instance
    def createJobTemplate(self):
        pass
    # takes JobTemplate instance, no return value
    def deleteJobTemplate(self, jobTemplate):
        pass
```

```
    # takes JobTemplate instance, returns string
    def runJob(self, jobTemplate):
        pass
    # takes JobTemplate instance and num values, returns string list
    def runBulkJobs(self, jobTemplate, beginIndex, endIndex, step):
        pass
    # takes string and JobControlAction value, no return value
    def control(self, jobName, operation):
        pass
    # takes string list, num value and boolean, no return value
    def synchronize(self, jobList, timeout, dispose):
        pass
    # takes string and long, returns JobInfo instance
    def wait(self, jobName, timeout):
        pass
    # takes string, returns JobState instance
    def jobStatus(jobName):
        pass
```

3. Rationale

A Python module is a single file encapsulating Python definitions and statements [PyTut]. The file name acts as parameter for the `import` statement, and therefore realizes the concept of a module name in the IDL sense. A DRMAA Python module SHOULD NOT be declared as part of a custom Python package, since this would lead to non-portable import statements in the application.

Interfaces in the sense of Java or C# are not a dedicated language construct in Python [PEP245]. Instead, a class encapsulates all DRMAA functions and attributes. The mapping does not imply an implementation strategy for the class. Implementations could use ‘old-style’ and ‘new-style’ classes as well as inheritance, as long as the instantiation procedure is not changed for the DRMAA-based application.

Enumerations are explicitly no part of the Python language [PEP354]. Since they are used as structuring concept for constants in the DRMAA IDL specification, they are mapped accordingly to classes.

The IDL `long` type represents a 32-bit signed value and therefore maps to the Python `int` type, which is promised to have at least 32-bit of precision [PyLib]. The Python `long` type is promised to have unlimited precision, and therefore represents the 64-bit IDL `long long` and `TimeAmount` type.

Python has no support for declaring constant attributes. The IDL `const` type is therefore mapped to a pre-defined attribute in the according scope. Default values for the constant / enumeration / value type attributes are partially adopted from the DRMAA 1.0 C language binding specification [DRMAAC10].

The DRMAA function `Session.init()`was renamed to `Session.initialize()`, in order to avoid confusion with the Python constructor method `__init__()`. The method was intentionally NOT mapped to the Python `__init__()` function. The DRMAA `initialize()` function explicitly attaches the application to the DRM system, which is not necessary in all cases (e.g. reading the library version number).

Some of the DRMAA IDL exceptions are mapped to built-in Python exceptions, as suggested by Section 5 of [GFD130]. Even though Python exception names usually end in “…`Error`” and not in “…`Exception`”, the original names are kept. This makes the resulting source code more familiar to experienced DRMAA developers, and eases the re-use of existing documentation. The

binding does not rely on a common exception base class (“`DrmaaException`”), since the module scoping already ensures encapsulation.

Partial timestamp objects are mapped to native Python strings, without a separate type definition in module scope. The in-built Python types for date and time representation are not usable here, due to the partial time stamp feature of DRMAA 1.0 (see Section 6 of [GFD130]).

According to the discussion in Section 8.2 of [GFD130], this language binding maps implementation-specific job template attributes to additional class data attributes [PyTut].

The `runBulkJobs` method in the IDL specification has three numerical parameters to express a range of `int` values. In Python, this is usually expressed by a numerical list created by the `range()` function. Since the increment value for a list generated by `range()`cannot be determined in a reliable way, the language binding sticks with the original syntax.

The IDL `valuetype` is mapped to a Python class, even though a Python `dict` might look appropriate, especially with the demand for comparability (see Section 2.2 of [GFD130]). The main reason is the restriction to the defined set of attributes, while a dictionary type would allow the DRMAA library implementation to support more keys.

4. Python 3.0 compatibility

Python 3 removes the in-built `int` type in favor of `long`, which changes the mapping described in Section 2 accordingly. However, the DRMAA Python module API does not change for a Python 3 implementation.

5. Example application

```
from drmaa import *

s=Session()
print('DRMAA library v%s'%s.version)
s.initialize()
print('DRM system used: '+s.drmsInfo)
jt=JobTemplate()
jt.remoteCommand='/bin/sleep'
jt.args=['10']
jname=s.runJob(jt)
s.control(jname, JobControlAction.TERMINATE)
jinfo=s.wait(jname, Session.TIMEOUT_WAIT_FOREVER)
if (jinfo.wasAborted):
    print('Job never ran')
if (jinfo.hasExited):
    print('Job exited with '+jinfo.exitStatus)
if (jinfo.hasSignaled):
    print('Job was signalled with '+jinfo.terminatingSignal)
s.exit()
```

6. Security Considerations

The language binding specification does not imply security demands upon an implementation. According to the functional behavior, GFD.130 rules and best practices apply also here.

7. Intellectual Property Statement

The OGF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the OGF Secretariat.

The OGF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this recommendation. Please address the information to the OGF Executive Director.

8. Disclaimer

This document and the information contained herein is provided on an “As Is” basis and the OGF disclaims all warranties, express or implied, including but not limited to any warranty that the use of the information herein will not infringe any rights or any implied warranties of merchantability or fitness for a particular purpose.

9. Full Copyright Notice

10. References

[DRMAAC10] Andreas Haas (Ed.). Distributed Resource Management Application API C Bindings v1.0. February 2005

[GFD130] Peter Tröger, Daniel Templeton, Roger Brobst, Andreas Haas, Hrabri Rajic. Distributed Resource Management Application API 1.0 - IDL Specification (GFD.130). Proposed Grid Recommendation. Open Grid Forum, April 2008

[PEP245] Michel Pelletier. Python Interface Syntax (PEP 245). January 2001

[PEP354] Ben Finney. Enumerations in Python (PEP 354). December 2005

[PyLib] Python Software Foundation. The Python Standard Library - Built-in Types. Release 2.6. October 2008

[PyTut] Guido van Rossum, Fred L. Drake. Python Tutorial. February 2008