Peter Tröger, Hasso-Plattner-Institute
(Corresponding Author)
June 2012

OCCI-DRMAA

Status of This Document

Group Working Draft - Proposed Recommendation (GWD-R-P)

Document Change History

Copyright Notice

Trademark

All company, product or service names referenced in this document are used for identification purposes only and may be trademarks of their respective owners.

Abstract

This document is an extension specification in the *Open Cloud Computing Interface (OCCI)* document series. It describes an extension of the *OCCI Core Model* [**?**] for remote access to a distributed resource management (DRM) system over RESTful protocols such as HTTP. The wire protocol is defined by a OCCI rendering. All interface semantics in this specification are derived from the *Distributed Resource Management Application API Version 2* [**?**], so this document also represents a language binding in the DRMAA document series.

The intended audience for this specification are OCCI and DRMAA implementors. Based on this specification, OCCI implementors can extend their backend support to support DRM systems. DRMAA implementors can realize a remote version of their API implementation.

Notational Conventions

The key words "MUST" "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in RFC 2119 [**?**].

Contents

1 Introduction

The *Distributed Resource Management Application API Version 2 (DRMAA)* specification defines an interface for tightly coupled, but still portable access to Distributed Resource Management (DRM) systems. The scope is limited to job submission, job control, reservation management, and retrieval of job and machine monitoring information. The DRMAA root specification [**?**] describes the abstract API concepts and the behavioral rules of a DRMAA-compliant implementation. The programming language representation of the API is defined by a separate *language binding specification*.

The Open Cloud Computing Interface (OCCI) is a RESTful Protocol and API for all kinds of management tasks. The OCCI Core specification defines the OCCI Core Model [**?**]. It can be interacted with *renderings* (including associated behaviors) and expanded through *extensions*. For particular domain, extensions specify additional resource types, their attributes and the actions that can be taken on each resource type. OCCI makes an ideal interoperable boundary interface between the web and the internal resource management system of infrastructure providers.

This document acts as OCCI extension for the domain of such DRM systems that are covered by the DRMAA specification. It therefore acts both as OCCI extension and as DRMAA language binding. The OCCI DRMAA extension details how an OCCI implementation can provide control and monitoring functions for a DRM system as a Service API offering.

Due to the nature of this specification, no functional or protocol description is provided. All behavioral semantics of an implementation MUST be conformant to the DRMAA specification [**?**]. All syntactical aspects of the access protocol MUST be conformant to a chosen OCCI binding, such as the HTTP binding [**?**].

There are other relevant OGF standards for remote APIs in the area of job submission and monitoring. They typically focus on meta-scheduling, while this specification provides a tightly coupled DRM system access with a remote API. OGSA-BES [**?**] defines a service interface for similar job submission purposes, but with strong focus on the abstract notion of resources and WSRF standards [**?**]. OCCI-DRMAA interfaces may serve as backend for a OGSA-BES implementation. JSDL [**?**] is a specification for XML-based job description, but OCCI-DRMAA utilizes the more restrictive DRMAA job description scheme for implementing mandatory job attribute support.

2 Basic concepts

Add DrmaaReflective mapping

Consider DRMAAv2 errata

Table 1 describes the *Kind* instances for each of the OCCI-DRMAA entities.

All instantiatable DRMAA structures are represented as OCCI resources. Original DRMAA methods that return struct instances are mapped to OCCI or HTTP verb actions that return the location of a new struct resource instance. Clients MAY be allowed to create new struct resources by a POST request to the collection URI of the struct resource (e.g. POST /drmaa2/jobinfo/). By using an appropriate content type in a GET request on the struct instance, the client SHOULD be able to retrieve a serialized version of the struct instance.

The *JobInfo* structure of DRMAA represents a set of information items for a job. It is modeled as OCCI mixin, so the `jobinfo` mixin is a set of additional information attributes about `job` resources. This allows to use it both as reporting and filter configuration data structure. The reporting functionality comes from the combination of a `job` resource with the according `jobinfo` mixin. Filtering is supported by querying a `jobsession` resource that is connected with the `jobinfo` mixin for jobs under the given `jobinfo` filter constraints (see [**?**], Section 4.4.5).

DRMAA templates are data structures that express complex information entities as a whole. They might be modified by a DRM system after their creation, which makes them additional OCCI resources without actions.

The DRMAA session concept models the relationship of *Job* and *JobSession* instances. Similarly, it models the relation between *Reservation* and *ReservationSession* instances. In OCCI-DRMAA, these relationships are represented by OCCI links between the according resource entities. A `joblink` resource represents the connection of a job to it's job session. A `reservationlink` resource represents the connection of an advance reservation to it's reservation session.

Most of the enumeration members from the DRMAA specification are mapped directly to JSON strings. For the *JobTemplatePlaceholder* values, special mapping rules apply (see Table 3).

In adherence to the DRMAA specification, *reservation*, *reservationsession*, *reservationtemplate* and *reservationlink* only MAY be supported by the OCCI-DRMAA implementation. An implementation MUST either support all of these resource types, or none of them. The support MUST be discoverable through the OCCI Query Interface. All other *Kind* instances MUST be supported.

2.1 Callback support

The DRMAA specification allows the DRMAA API implementation to offer push notification for applications. In OCCI-DRMAA, this is possible for clients by providing a URI as callback target in the *registereventnotification* action. If the provided URI declares the *http*protocol to be used, then the implementation MUST deliver the notification information via POST to the given endpoint. For all other protocols (i.e. mail://), the behavior is implementation-specific.

Define serialization of DrmaaNotification struct instance in the callback call

2.2 Exceptions

DRMAA defines a set of exceptions that may be thrown by API activities. In OCCI-DRMAA, the mapping of these exceptions depends on the chosen transport. In case the HTTP transport is used for the OCCI-DRMAA implementation, the error code mapping in Table 2 MUST be applied.

2.3 Type system

The OCCI core model supports the notion of action attributes as representation for parameters of an invocable operation (see [**?**], Section 4.5.4). OCCI attributes always have string values(see [**?**], Figure 2). On the other hand, DRMAA has a central definition of utilized parameter types. For this reason, Table 3 defines a mapping from DRMAA-IDL data types to a JSON-based string representation. Accordingly, constant values from DRMAA definitions are mapped to OCCI-DRMAA (see Table 4).

Add missing examples in Table, maybe based on OCCI JSON rendering ideas

DRMAAs notion of `UNSET` values is mapped to multiplicity of OCCI-DRMAA attributes. All attributes that may hold the value `UNSET` MUST be have a multiplicity $0 \dots 1$.

2.4 Capability check

The *SessionManager::supports()* method in DRMAA allows to check for the implementation support of optional features. The OCCI-DRMAA binding does not map this enumeration directly, since the different functionality checks are represented by OCCI-ish activities:

- *ADVANCE_RESERVATION*: The client is expected to ask the query interface if the *reservation* resource is supported.

  add example code

- *RESERVE_SLOTS*: The support for this capability is expressed by the *hasReserveSlots* attribute on the *drmaa2* resource (see Table 5).
- *BULK_JOBS_MAXPARALLEL*: The support for this capability is expressed by the *hasMaxParallel* attribute on the *drmaa2* resource (see Table 5).
- *RT_STARTNOW*: The support for this capability is expressed by the *hasStartNow* attribute on the *drmaa2* resource (see Table 5).
- *CALLBACK*: The client is expected to ask the query interface if the *registerEventNotification* action is supported by this implementation.

  add example code

- *JT_EMAIL, JT_STAGING, JT_DEADLINE, JT_MAXSLOTS, JT_ACCOUNTINGID, RT_DURATION, RT_MACHINEOS, RT_MACHINEARCH*: The client is expected to ask the query interface if the according job template mixin is supported.

  add example code

2.5 Blocking wait calls

DRMAA supports the notion of blocking status wait calls for both the *Job* and the *JobSession* interface. The OCCI-DRMAA extension re-models this approach with the concept of a wait handle URI. The call flow is described in Section 5.7 and Section 5.8. The wait action returns the location of the wait handle, which can be further used for polling GET requests to the server. The server MUST return one of these three possible error codes on such request:

- *Still waiting* (HTTP error 404): The blocking wait call is still running, no timeout occurred so far. The wait handle location remains valid.
- *Timeout* (HTTP error 410): The blocking call was terminated due to timeout. The wait handle location is now invalid.
- *Success* (HTTP error 301): The blocking call was terminated since the wait condition was fulfilled. The wait handle location is now invalid.

3 OCCI-DRMAA Resources

DRMAA interfaces represent activities on instantiatable entities. They are mostly modeled as OCCI resources:

- A `drmaa2` resource represents the container for all OCCI-DRMAA resources and the according functionalities.
- A `jobsession` resource acts as container for `job` resources and `jobarray` resources.
- A `reservationsession` resource acts as container for `reservation` resources.

- A `monitoringsession` resource acts as representation of information about the DRM system on provider side.
- A OCCI-DRMAA `job` resource represents one job in the underlying DRM system on provider side. Similarly, the `jobarray` resource represents a cluster of jobs.
- A `reservation` resource represents a successfully created advance reservation in the DRM system.

Add missing resources in the list

DRMAA interface methods that trigger state changes in the DRM system map to OCCI actions on OCCI resources. DRMAA functionalities that lead to the creation of instances represented by OCCI resources are available as OCCI resource creation activities (see also [**?**], Section 4.4.4 and [**?**], Section 3.4.4). DRMAA interface methods that return named instances (i.e. JobSession::getJobArray) are not translated to OCCI actions, since this kind of retrieval is possible by formulating a resource location string explicitely.

DRMAA IDL interface attributes map to OCCI attributes. The `readonly` modifier for DRMAA attributes translates to the immutability property. The concept of optional or possibly *UNSET* attributes in DRMAA is mapped to a OCCI attribute multiplicity of $0 \dots 1$. Id-based or name-based referencing of instances (e.g. of a DRMAA session) is replaced by URI-based referencing.

3.1 drmaa2

The DRMAA *SessionManager* interface is represented by the d́rmaa2ŕesource. The reference to this resource MUST be retrievable by a request to the OCCI query interface for the following OCCI *kind* scheme:

`http://schemas.ogf.org/drmaa2`

The result of this query interface call MUST be one OCCI *Kind* location. This location then is used to determine the location of the *drmaa2* resource, which MUST be a singleton too.

The *open...* methods of the DRMAA *SessionManager* interface are not mapped to OCCI-DRMAA, since they translate directly to explicitely formulated resource locations.

The *destroy...* methods of the DRMAA *SessionManager* interface are represented by support for the DELETE verb on session resources.

Add example reference

The *close...* methods of the DRMAA *SessionManager* interface are also not mapped to OCCI-DRMAA, since OCCI assumes stateless clients as basic architectural concept.

The *drmaa2* resource supports the attributes as described in Table 5.

3.2 jobsession

Table 6 describes the actions available for a `jobsession` resource. Required attributes on actions are always mutable. Table 7 describes the attributes delivered on retrieval of a `jobsession` resource.

The original `JobSession::getJobArray` method is not represented as attribute or action. Instead, the server MUST support the retrieval of a specifically named `jobarray` resource based on the manual creation of an according URI beginning with *http://schemas.ogf.org/occi/drmaa#jobarray*.

The original `JobSession:getJobs` method is also not represented as attribute or action. Instead, the `jobsession` resource MUST return the references to all attached `job` resource instances as `joblink` OCCI links. This retrieval operation MUST also support a mixin-based filtering of this `job` set.

Not clear how the jobsession resource can transform to the jobsession+jobinfo resource for this activity.

On creation of a `jobsession` resource, the `contact` and / or the `sessionName` attribute MAY be provided by the client as part of the request. This maps to original DRMAA *SessionManager::createJobSession* method signature.

3.3 reservationsession resource

Table 8 describes the actions available for a `reservationsession` resource. Required attributes on actions are always mutable. Table 9 describes the attributes delivered on retrieval of a `reservationsession` resource.

The original `ReservationSession::getReservation` method is not represented as attribute or action. Instead, the server MUST support the retrieval of a specifically named `reservation` resource based on the manual creation of an according URI beginning with *http://schemas.ogf.org/occi/drmaa#reservation*.

The original `ReservationSession:getReservations` method is also not represented as attribute or action. Instead, the `reservationsession` resource MUST return the references all attached `reservation` resource instances as `reservationlink` OCCI links.

On creation of a `reservationsession` resource, the `contact` and / or the `sessionName` attribute MAY be provided by the client as part of the request. This maps to original DRMAA *SessionManager::createReservationSession* method signature.

3.4 monitoringsession resource

Table 10 describes the attributes delivered on retrieval of a `monitoringsession` resource. This resource has no actions defined.

Fix this, multiple queue attributes are not allowed in OCCI, use QueueInfoList rendering instead.

The original `MonitoringSession:getAllJobs` method is not represented as attribute or action. Instead, the `monitoringsession` resource MUST return the references to all known `job` resource instances as OCCI links. This retrieval operation MUST also support a mixin-based filtering of this `job` set.

The original `MonitoringSession:getAllReservations` method is also not represented as attribute or action. Instead, the `monitoringsession` resource MUST return the references all known `reservation` resource instances as OCCI links.

The original `MonitoringSession:getAllQueues` method is represented with the `queue` attribute. On retrieval of the `monitoringsession` resource, the implementation MUST return a `queue` attribute for each supported queue in the target system. This retrieval operation MUST also support the provisioning of `queue` attributes by the client for filtering of the result. Implementations MUST support the filtering by queue name. Implementations MAY support the filtering by implementation-specific *QueueInfo* attributes.

The original `MonitoringSession:getAllMachines` method is represented with the `machine` attribute. On retrieval of the `monitoringsession` resource, the implementation MUST return a `machine` attribute for each execution host in the target DRM system. This retrieval operation MUST also support the provisioning of `machine` attributes by the client for filtering of the result. Implementations MUST support the filtering by machine name. Implementations MAY support the filtering by other (DRMAA-)mandatory or implementation-specific *MachineInfo* attributes.

On creation of a `monitoringsession` resource, the `contact` attribute MAY be provided by the client as part of the request. This maps to original DRMAA *SessionManager::openMonitoringSession* method signature.

3.5 job resource

Table 13 describes the actions available for a `job` resource. Table 14 describes the attributes delivered for the `job` resource.

The DRMAA *sessionName* attribute was replaced by the `session` resource link, which fulfills the same purpose, but fits better to the OCCI semantics.

The `jobTemplate` and `session` attributes are mutable by the client, in order to allow the implicit triggering of job execution by creating a `job` resource. In this case, it MUST be ensured that the client provides both of them with valid values. All other attempts to create `job` resources from the client side MUST fail.

The `job` resource can be combined with the `jobinfo` mixin.

explain use cases

3.6 jobarray resource

Table ?? describes the actions available for a `jobarray` resource. Table ?? describes the attributes delivered for the `jobarray` resource.

The DRMAA *sessionName* attribute was replaced by the `session` resource link, which fulfills the same purpose, but fits better to the OCCI semantics.

The original `JobArray:jobs` attribute is not represented as attribute or action. Instead, the `jobarray` resource MUST return the references to all jobs in the array as OCCI `job` links.

3.7 reservation resource

Table 15 describes the actions available for a `reservation` resource. Table 16 describes the attributes delivered for the `reservation` resource.

The DRMAA *sessionName* attribute was replaced by the `session` resource link, which fulfills the same purpose, but fits better to the OCCI semantics.

The `reservationTemplate` and `session` attributes are mutable by the client, in order to allow the implicit triggering of reservation requesting by creating a `reservation` resource. In this case, it MUST be ensured that the client provides both attributes with valid values. All other attempts to create `reservation` resources from the client side MUST fail.

3.8 JobTemplate

Table 17 describes the attributes delivered for the `jobtemplate` resource. The resource has no actions.

The implementation MUST render all `jobtemplate` attributes immutable when the template is linked from a `job` resource.

3.9 ReservationTemplate

Table 18 describes the attributes delivered for the `reservationtemplate` resource. The resource has no actions.

The implementation MUST render all `reservationtemplate` attributes immutable when the template is linked from a `reservation` resource.

4 OCCI-DRMAA Mixins

Describe missing struct resources

Describe optional job template attributes to be mixins, all optional. Leave open if a POST on job template resource adds automatically all mixins, or if the client is doing this. Inherit all mixins from an abstract optional_jt_attri mixin

Map job categories to empty mixins, which all inherit from an abstract job category mixin. This allows to distinguish the category mixins from other occi-drmaa mixins. If the user attaches such a job category mixin to a job template, ...

4.1 JobInfo

Table 19 describes the attributes delivered by the usage of the `jobinfo` mixin on a resource. There are no actions defined by this mixin.

Implementations SHOULD consider that some of the `jobinfo` attributes are mandatory on job information reporting (see [?], Section 5.5). All attributes MUST be optional on mixin-based job filtering.

5 Examples with HTTP

GFD-P-R.185 [?] describes the rendering of OCCI through a RESTful HTTP interface. The following example shows how typical OCCI-DRMAA interactions would be modeled based on this rendering.

5.1 Query DRMAA interfaces

```
> GET /-/ HTTP/1.1
> [...]
> Category: drmaa2;scheme="http://schemas.ogf.org/drmaa2"

< HTTP/1.1 200 OK
< [...]
< Category: drmaa2;scheme="http://schemas.ogf.org/drmaa2";class="kind";
rel="http://schemas.ogf.org/occi/core#resource";
            location="/drmaa2/";title="DRMAAv2 Interfaces";
            attributes="occi.drmaa2.drmsName occi.drmaa2.drmsVersion occi.drmaa2.drmaaName occi.drmaa2.

> GET /drmaa2/ HTTP/1.1
> [...]

< HTTP/1.1 200 OK
< Content-type: text/uri-list
< [...]
< http://example.com/drmaa2
```

5.2 Determine the DRM system information

```
> GET /drmaa2 HTTP/1.1
> [...]

< HTTP/1.1 200 OK
< [...]
< X-OCCI-Attribute: occi.drmaa2.drmsName="Platform LSF"
< X-OCCI-Attribute: occi.drmaa2.drmsVersion="{"major":"42", "minor":"0"}
< X-OCCI-Attribute: occi.drmaa2.drmaaName="Thijs's OCCI-DRMAA backend"
< X-OCCI-Attribute: occi.drmaa2.drmaaVersion="{"major":"2", "minor":"17"}
< [...]
```

5.3 Get all existing job sessions

```
> GET /drmaa2/jobsession/ HTTP/1.1
> [...]

< HTTP/1.1 200 OK
< Content-type: text/uri-list
< [...]
< http://example.com/drmaa2/jobsession/17
```

5.4 Create a job session

```
> POST /drmaa2/jobsession/ HTTP/1.1
> [...]
> X-OCCI-Attribute: occi.drmaa2.contact="headnode.testbed.platform.com"
> X-OCCI-Attribute: occi.drmaa2.sessionName="MyTestSession"
> [...]

< HTTP/1.1 201 CREATED
< [...]
< Location: http://example.com/drmaa2/jobsession/session1
< [...]
```

5.5 Submit a job

Step 1 - Create a `jobtemplate` resource:

```
> POST /drmaa2/jobtemplate/ HTTP/1.1
> [...]
> X-OCCI-Attribute: occi.drmaa2.remoteCommand="/bin/date"
> X-OCCI-Attribute: occi.drmaa2.machineOS="LINUX"
> X-OCCI-Attribute: occi.drmaa2.email=["peter@troeger.eu","tmetsch@platform.com"]
> X-OCCI-Attribute: occi.drmaa2.emailOnTerminated=true
> [...]

< HTTP/1.1 201 CREATED
< [...]
< Location: http://example.com/drmaa2/jobtemplate/template1
< [...]
```

Step 2 - Create a `job` resource:

```
> POST /drmaa2/job/ HTTP/1.1
> [...]
> X-OCCI-Attribute: occi.drmaa2.session="/drmaa2/jobsession/session1"
> X-OCCI-Attribute: occi.drmaa2.jobTemplate="/drmaa2/jobtemplate/template1"
> [...]

< HTTP/1.1 201 CREATED
< [...]
< Location: http://example.com/drmaa2/job/job43
< [...]
```

5.6 Fetch filtered list of jobs

```
> GET /drmaa2/job/ HTTP/1.1
> [...]
> X-OCCI-Attribute: occi.drmaa2.queueName="foo"
> [...]

< HTTP/1.1 200 OK
< Content-type: text/uri-list
< [...]
< http://example.com/drmaa2/job/job43
< http://example.com/drmaa2/job/job44
```

5.7 Wait for job start

```
> GET /drmaa2/job/job43?action=waitstarted HTTP/1.1
> [...]
> X-OCCI-Attribute: occi.drmaa2.timeout="..."
> [...]

< HTTP/1.1 202 ACCEPTED
< [...]
< Location: /drmaa2/job/job43/waithandle1
< [...]

> GET /drmaa2/job/job43/waithandle1 HTTP/1.1
> [...]
< HTTP/1.1 404 NOT FOUND

> GET /drmaa2/job/job43/waithandle1 HTTP/1.1
> [...]
< HTTP/1.1 410 GONE

> GET /drmaa2/job/job43/waithandle1 HTTP/1.1
> [...]
< HTTP/1.1 301 MOVED PERMANENTLY
< [...]
< Location: /drmaa2/job/job43
< [...]
```

5.8 Wait for some job to start

Can the jobs attribute be replaced with some OCCI links input ?

```
> GET /drmaa2/jobsession/js44?action=waitanystarted HTTP/1.1
> [...]
> X-OCCI-Attribute: occi.drmaa2.timeout="..."
> X-OCCI-Attribute: occi.drmaa2.jobs=["http://example.com/drmaa2/job/job44", "http://example.com/drmaa2/
> [...]

< HTTP/1.1 202 ACCEPTED
< [...]
< Location: /drmaa2/jobsession/js44/waithandle1
< [...]

> GET /drmaa2/jobsession/js44/waithandle1 HTTP/1.1
> [...]
< HTTP/1.1 404 NOT FOUND

> GET /drmaa2/jobsession/js44/waithandle1 HTTP/1.1
> [...]
< HTTP/1.1 410 GONE

> GET /drmaa2/jobsession/js44/waithandle1 HTTP/1.1
> [...]
< HTTP/1.1 301 MOVED PERMANENTLY
< [...]
< Location: /drmaa2/job/job42
< [...]
```

5.9 Query if advanced reservation is supported, negative answer

```
> GET /-/ HTTP/1.1
> [...]
> Category: reservationsession;scheme="http://schemas.ogf.org/drmaa2"

< HTTP/1.1 204 NOCONTENT
< [...]
```

Check OCCI compliance of negative query answer

5.10 Query if advanced reservation is supported, positive answer

```
> GET /-/ HTTP/1.1
> [...]
> Category: reservationsession;scheme="http://schemas.ogf.org/drmaa2"
```

```
< HTTP/1.1 200 OK
< [...]
< Category: reservationsession;
                   scheme="http://schemas.ogf.org/drmaa2";
                   class="kind";
                   rel="http://schemas.ogf.org/occi/core#resource";
                   location="/drmaa2/reservationsession/";
                   title="DRMAAv2 Advance Reservation Sessions";
                   attributes="occi.drmaa2.reservationsession.contact
                               occi.drmaa2.reservationsession.sessionName"
```

5.11 Query all existing reservation sessions

```
> GET /drmaa2/reservationsession/ HTTP/1.1
> [...]

< HTTP/1.1 200 OK
< Content-type: text/uri-list
< [...]
< http://example.com/drmaa2/reservationsession/rsess5
< http://example.com/drmaa2/reservationsession/rsess4711
< http://example.com/drmaa2/reservationsession/rsess42
```

5.12 Request an advance reservation

Step 1 - Create a ***reservationtemplate*** *resource*:

```
> POST /drmaa2/reservationtemplate/ HTTP/1.1
> [...]
> X-OCCI-Attribute: occi.drmaa2.startTime="2012-11-11T11:11:11"
> X-OCCI-Attribute: occi.drmaa2.endTime="2012-11-12T00:00:00"
> X-OCCI-Attribute: occi.drmaa2.minSlots=2
> X-OCCI-Attribute: occi.drmaa2.maxSlots=5000
> [...]

< HTTP/1.1 201 CREATED
< [...]
< Location: http://example.com/drmaa2/reservationtemplate/rtpl4711
< [...]
```

Step 2 - Create a ***reservation*** *resource*:

```
> POST /drmaa2/reservation/ HTTP/1.1
> [...]
> X-OCCI-Attribute: occi.drmaa2.session="/drmaa2/reservationsession/rsess5"
> X-OCCI-Attribute: reservationTemplate="/drmaa2/reservationtemplate/rtpl4711"
> [...]
```

```
< HTTP/1.1 201 CREATED
< [...]
< Location: http://example.com/drmaa2/reservation/rs99xy
< [...]
```

5.13 Control a job

```
> POST /drmaa2/job/job77?action=suspend HTTP/1.1
> [...]

< HTTP/1.1 200 OK
< [...]
```

5.14 Fetch serialized information structure

Add example for content-type based serialization in GET request to structs

5.15 Get the list of machines

Step 1 - Create a ***`monitoringsession`*** *resource*:

```
> POST /drmaa2/monitoringsession/ HTTP/1.1
> [...]
> X-OCCI-Attribute: occi.drmaa2.contact="headnode.testbed.platform.com"
> [...]

< HTTP/1.1 201 CREATED
< [...]
< Location: http://example.com/drmaa2/monitoringsession/monitor8
< [...]
```

Step 2 - Fetch the ***`monitoringsession`*** *resource to get the machine list*:

```
> GET /drmaa2/monitoringsession/monitor8 HTTP/1.1
> [...]

< HTTP/1.1 200 OK
< [...]
< X-OCCI-Attribute: occi.drmaa2.machine="{"name":"exec1.testbed.platform.com","available":true,"sockets
< X-OCCI-Attribute: occi.drmaa2.machine="{"name":"exec2.testbed.platform.com","available":false,"socket
< [...]
```

6 Security Considerations

Security considerations from a DRM system point of view are clarified by the DRMAA root specification. An implementation MUST consider the regulations and security advices given there.

The DRMAA API does not specifically assume the existence of a particular security infrastructure in the DRM system. The scheduling scenario described herein presumes that security is handled at the point of interaction with the DRM system. It is assumed that credentials owned by the application using the API are in effect for the DRMAA implementation too, so that it acts as stakeholder for the application. This relays the responsibility of authentication to the OCCI rendering specification that is used to realize the wire protocol of an implementation.

DRMAA implementers SHOULD guard their product against buffer overflows that can be exploited through DRMAA enabled interactive applications or portals. Implementations of the DRMAA API will most likely require a network to coordinate subordinate DRM system requests. However, the API makes no assumptions about the security posture provided by the networking environment. Therefore, application developers SHOULD also consider the security implications of "on-the-wire" communications in this case.

7 Contributors

8 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.

9 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.

10 Full Copyright Notice

11 References

List of Tables

Term	Scheme	Title	Related Kind
jobsession	`<schema>/drmaa2#`	Job Session resource	`<schema>/occi/core#resource`
reservationsession	`<schema>/drmaa2#`	Reservation Session resource	`<schema>/occi/core#resource`
monitoringsession	`<schema>/drmaa2#`	Monitoring Session resource	`<schema>/occi/core#resource`
job	`<schema>/drmaa2#`	Job resource	`<schema>/occi/core#resource`
jobarray	`<schema>/drmaa2#`	Job Array resource	`<schema>/occi/core#resource`
reservation	`<schema>/drmaa2#`	Reservation resource	`<schema>/occi/core#resource`
jobinfo	`<schema>/drmaa2/job#`	Job Information Mixin	-
slotinfo	`<schema>/drmaa2#`	Slot Information resource	`<schema>/occi/core#resource`
queueinfo	`<schema>/drmaa2#`	Queue Information resource	`<schema>/occi/core#resource`
reservationinfo	`<schema>/drmaa2#`	Reservation Information resource	`<schema>/occi/core#resource`
machineinfo	`<schema>/drmaa2#`	Machine Information resource	`<schema>/occi/core#resource`
version	`<schema>/drmaa2#`	Version Information resource	`<schema>/occi/core#resource`
jobtemplate	`<schema>/drmaa2#`	Job Template resource	`<schema>/occi/core#resource`
reservationtemplate	`<schema>/drmaa2#`	Reservation Template resource	`<schema>/occi/core#resource`
joblink	`<schema>/drmaa2#`	Relation from `jobsession` to `job` resource	`<schema>/occi/core#link`
reservationlink	`<schema>/drmaa2#`	Relation from `reservationsession` to `reservation` resource	`<schema>/occi/core#link`

Table 1: The *Kind* instances defined for OCCI-DRMAA. The base URL *http://schemas.ogf.org* has been replaced with `<schema>` in this table for a better readability experience.

DRMAA Exception	OCCI-DRMAA HTTP Error Code
DeniedByDrmsException	401, if authorization is not available
DeniedByDrmsException	403, if authorization is available, but the operation is not allowed
DrmCommunicationException	500
TryLaterException	503, with retry header
TimeoutException	410
InternalException	500
InvalidArgumentException	400
InvalidSessionException	404
InvalidStateException	409
OutOfResourceException	503, without retry header
UnsupportedAttributeException	400
UnsupportedOperationException	405
ImplementationSpecificException	500

Table 2: Mapping of DRMAA exceptions to HTTP error codes.

DRMAA type	OCCI-DRMAA representation
string	JSON string Example: `"/bin/date"`
long / long long	JSON number Example: `42`
double	JSON float Example: `7.02`
boolean	JSON boolean, defaults to `false` if attribute is not set Example: `true`, `false`
struct	JSON dictionary with member names as keys Example: `{"machineName":"node1.drmaa2.org", "slots":42}`
Dictionary	JSON dictionary Example: `{"PATH": "/usr/bin", "OMP_NUM_THREADS": "64"}`
enum value	JSON string Example: `"RUNNING"`
JobTemplatePlaceholder	JSON string surrounded by ”$” Example: `"$HOME_DIRECTORY$"`
StringList	JSON array of strings Example: `["foo@example.com", "bar@example.com"]`
JobList	JSON array of `job` resource URIs Example: `["http://example.com/drmaa2/job/job43", "http://example.com/drmaa2/job/job46"]`
QueueInfoList	JSON array of strings Example:
MachineList	JSON array of strings Example:
OrderedStringList	JSON array of strings Example: `["node1.drmaa2.org", "node2.drmaa2.org"]`
OrderedSlotInfoList	JSON array of JSON dictionaries Example: `[{"machineName": "node1.drmaa2.org", "slots": 42}]`
AbsoluteTime	JSON string in ISO8601 format Example: `"2003-04-01T13:01:02"`
TimeAmount	JSON number representing seconds Example: `3600`

Table 3: The data types used for attributes in OCCI-DRMAA.

DRMAA constant	OCCI-DRMAA representation
ZERO_TIME	JSON number `0`
INFINITE_TIME	JSON number `-1`
NOW	JSON string `"now"`

Table 4: The constants used in OCCI-DRMAA.

Attribute	Type	Multiplicity	Mutability
occi.drmaa2.drmsName	string	1	Immutable
occi.drmaa2.drmsVersion	DRMAA Version	1	Immutable
occi.drmaa2.drmaaName	string	1	Immutable
occi.drmaa2.drmaaVersion	DRMAA Version	1	Immutable
occi.drmaa2.hasReserveSlots	Boolean	1	Immutable
occi.drmaa2.hasMaxParallel	Boolean	1	Immutable
occi.drmaa2.hasStartNow	Boolean	1	Immutable

Table 5: Attributes of the drmaa2 resource

Action term	Attributes	Attribute type	DRMAA equivalent
runjob	jobTemplate	jobtemplate URI	JobSession::runJob()
runbulkjobs	jobTemplate	jobtemplate URI	JobSession::runBulkJobs()
	beginIndex	long	
	endIndex	long	
	step	long	
	maxParallel	long	
waitanystarted	job	job URI	JobSession::waitAnyStarted()
	timeout	DRMAA TimeAmount	
waitanyterminated	job	job URI	JobSession::waitAnyTerminated()
	timeout	DRMAA TimeAmount	

Table 6: Actions available for a jobsession resource

Attribute	Type	Multiplicity	Mutability
occi.drmaa2.contact	string	1	Mutable
occi.drmaa2.sessionName	string	1	Mutable
occi.drmaa2.jobCategory	string	0..1	Immutable

Table 7: Attributes of the jobsession resource

Action term	Required attributes	Attribute type	DRMAA equivalent
requestreservation	reservationTemplate	reservationtemplate URI	ReservationSession::requestReservation()

Table 8: Actions available for a reservationsession resource

Attribute	type	Multiplicity	Mutability
occi.drmaa2.contact	string	1	Mutable
occi.drmaa2.sessionName	string	1	Mutable

Table 9: Attributes of the reservationsession resource

Attribute	Type	Multiplicity	Mutability
occi.drmaa2.contact	string	1	Mutable
occi.drmaa2.queue	DRMAA QueueInfo	0..*	Mutable
occi.drmaa2.machine	DRMAA MachineInfo	0..*	Mutable

Table 10: Attributes of the monitoringsession resource

Action term	Required attributes	Attribute type	DRMAA equivalent
`suspend`	-	-	Job::suspend()
`resume`	-	-	Job::resume()
`hold`	-	-	Job::hold()
`release`	-	-	Job::release()
`terminate`	-	-	Job::terminate()
`waitstarted`	timeout	DRMAA TimeAmount	Job::waitStarted()
`waitterminated`	timeout	DRMAA TimeAmount	Job::waitTerminated()

Table 11: Actions available for a `job` resource

Attribute	Type	Multiplicity	Mutability
`occi.drmaa2.jobId`	string	1	Immutable
`occi.drmaa2.session`	`jobsession` URI	0..1	Mutable
`occi.drmaa2.jobTemplate`	`jobtemplate` URI	1	Mutable
`occi.drmaa2.state`	Enum (DRMAA JobState)	1	Immutable
`occi.drmaa2.substate`	string	0..1	Immutable

Table 12: Attributes of the `job` resource

Action term	Required attributes	Attribute type	DRMAA equivalent
`suspend`	-	-	JobArray::suspend()
`resume`	-	-	JobArray::resume()
`hold`	-	-	JobArray::hold()
`release`	-	-	JobArray::release()
`terminate`	-	-	JobArray::terminate()

Table 13: Actions available for a `jobarray` resource

Attribute	Type	Multiplicity	Mutability
`occi.drmaa2.jobArrayId`	string	1	Immutable
`occi.drmaa2.session`	`jobsession` URI	0..1	Immutable
`occi.drmaa2.jobTemplate`	`jobtemplate` URI	1	Immutable

Table 14: Attributes of the `jobarray` resource

Action term	Required attributes	Attribute type	DRMAA equivalent
terminate	-	-	Reservation::terminate()

Table 15: Actions available for a `reservation` resource

Attribute	Type	Multiplicity	Mutability
`occi.drmaa2.reservationId`	string	1	Immutable
`occi.drmaa2.session`	`reservationsession` URI	0..1	Mutable
`reservationTemplate`	`reservationtemplate` URI	1	Mutable

Table 16: Attributes of the `reservation` resource

Attribute	Type	Multiplicity	Mutability
`occi.drmaa2.remoteCommand`	string	0..1	Mutable
`occi.drmaa2.args`	DRMAA OrderedStringList	0..1	Mutable
`occi.drmaa2.submitAsHold`	boolean	0..1	Mutable
`occi.drmaa2.rerunnable`	boolean	0..1	Mutable
`occi.drmaa2.jobEnvironment`	DRMAA Dictionary	0..1	Mutable
`occi.drmaa2.workingDirectory`	string	0..1	Mutable
`occi.drmaa2.jobCategory`	string	0..1	Mutable
`occi.drmaa2.email`	DRMAA StringList	0..1	Mutable
`occi.drmaa2.emailOnStarted`	boolean	0..1	Mutable
`occi.drmaa2.emailOnTerminated`	boolean	0..1	Mutable
`occi.drmaa2.jobName`	string	0..1	Mutable
`occi.drmaa2.inputPath`	string	0..1	Mutable
`occi.drmaa2.outputPath`	string	0..1	Mutable
`occi.drmaa2.errorPath`	string	0..1	Mutable
`occi.drmaa2.joinFiles`	boolean	0..1	Mutable
`occi.drmaa2.reservationId`	`reservation` URI	0..1	Mutable
`occi.drmaa2.queueName`	string	0..1	Mutable
`occi.drmaa2.minSlots`	long	0..1	Mutable
`occi.drmaa2.maxSlots`	long	0..1	Mutable
`occi.drmaa2.priority`	long	0..1	Mutable
`occi.drmaa2.candidateMachines`	DRMAA OrderedStringList	0..1	Mutable
`occi.drmaa2.minPhysMemory`	long	0..1	Mutable
`occi.drmaa2.machineOS`	Enum (DRMAA OperatingSystem)	0..1	Mutable
`occi.drmaa2.machineArch`	Enum (DRMAA CpuArchitecture)	0..1	Mutable
`occi.drmaa2.startTime`	DRMAA AbsoluteTime	0..1	Mutable
`occi.drmaa2.deadlineTime`	DRMAA AbsoluteTime	0..1	Mutable
`occi.drmaa2.stageInFiles`	DRMAA Dictionary	0..1	Mutable
`occi.drmaa2.stageOutFiles`	DRMAA Dictionary	0..1	Mutable
`occi.drmaa2.resourceLimits`	DRMAA Dictionary	0..1	Mutable
`occi.drmaa2.accountingId`	string	0..1	Mutable

Table 17: Attributes of the `jobtemplate` resource

Attribute	Type	Multiplicity	Mutability
`occi.drmaa2.reservationName`	string	0..1	Mutable
`occi.drmaa2.startTime`	DRMAA AbsoluteTime	0..1	Mutable
`occi.drmaa2.endTime`	DRMAA AbsoluteTime	0..1	Mutable
`occi.drmaa2.duration`	DRMAA TimeAmount	0..1	Mutable
`occi.drmaa2.minSlots`	long	0..1	Mutable
`occi.drmaa2.maxSlots`	long	0..1	Mutable
`occi.drmaa2.jobCategory`	string	0..1	Mutable
`occi.drmaa2.userACL`	DRMAA StringList	0..1	Mutable
`occi.drmaa2.candidateMachines`	DRMAA OrderedStringList	0..1	Mutable
`occi.drmaa2.minPhysMemory`	long	0..1	Mutable
`occi.drmaa2.machineOS`	Enum (DRMAA OperatingSystem)	0..1	Mutable
`occi.drmaa2.machineArch`	Enum (DRMAA CpuArchitecture)	0..1	Mutable

Table 18: Attributes of the `reservationtemplate` resource

Attribute	Type	Multiplicity	Mutability
`occi.drmaa2.jobId`	string	0..1	Mutable
`occi.drmaa2.exitStatus`	long	0..1	Mutable
`occi.drmaa2.terminatingSignal`	string	0..1	Mutable
`occi.drmaa2.annotation`	string	0..1	Mutable
`occi.drmaa2.jobState`	Enum (DRMAA JobState)	0..1	Mutable
`occi.drmaa2.jobSubState`	string	0..1	Mutable
`occi.drmaa2.allocatedMachines`	DRMAA OrderedSlotInfoList	0..1	Mutable
`occi.drmaa2.submissionMachine`	string	0..1	Mutable
`occi.drmaa2.jobOwner`	string	0..1	Mutable
`occi.drmaa2.slots`	long	0..1	Mutable
`occi.drmaa2.queueName`	string	0..1	Mutable
`occi.drmaa2.wallclockTime`	DRMAA TimeAmount	0..1	Mutable
`occi.drmaa2.cpuTime`	long	0..1	Mutable
`occi.drmaa2.submissionTime`	DRMAA AbsoluteTime	0..1	Mutable
`occi.drmaa2.dispatchTime`	DRMAA AbsoluteTime	0..1	Mutable
`occi.drmaa2.finishTime`	DRMAA AbsoluteTime	0..1	Mutable

Table 19: Attributes of the `jobinfo` mixin