A RESTful Approach to the OGSA Basic Execution Service Specification

Sergio Andreozzi
Università di Bologna
sergio.andreozzi@unibo.it
http://sergioandreozzi.com/

Moreno Marzolla
INFN Padova
moreno.marzolla@pd.infn.it
http://www.dsi.unive.it/~marzolla
Talk outline

- Introduction
  - Job Management in Grid systems
  - The Basic Execution Service and Job Submission Description Language specifications
  - REST
- RESTful BES
- BES extensions
- Conclusions
Job Management in Grid Systems

- Job Submission is one of the basic functionalities of any Grid system
- Users submit and manage jobs (or activities) through an Execution Service
  - Start/Stop execution
  - Pause/Resume execution
  - Query job status
Example: the gLite CREAM service

- Computing Resource Execution and Management

![Diagram of CREAM service]

- **CREAM**: Computing Resource Execution and Management
- **CEMonitor**: Local Resource Management System
- **Submit/Cancel/Suspend/...**: Job Queue
- **Async Job Status Change Notifications**
- **Worker Node**

---

S. Andreozzi, M. Marzolla  
ICIW 2009, Mestre, Italy
Achieving Interoperability with Open Standards

- Different Grids provide different interfaces for the job management service
  - Same functionalities, different names, datatypes...
- Job Submission Description Language (JSDL)
  - OGF Specification
  - XML-based notation for describing computational jobs and their resource requirements
- OGSA Basic Execution Service (BES)
  - OGSA Specification
  - WebService-based interface for job submission and management
  - Uses JSDL to describe activities
BES Port-types and operations

BES-Factory port-type
- CreateActivity
- GetActivityStatuses
- TerminateActivities
- GetActivityDocuments
- GetFactoryAttributesDocument

BES-Management port-type
- StartAcceptingNewActivities
- StopAcceptingNewActivities
<JobDefinition id="HelloWorldJob">
  <JobDescription>
    <JobIdentification>
      <JobName>Hello World</JobName>
      <Description>
        This job prints the Hello World message
      </Description>
      <JobProject>OMII-EU</JobProject>
      <JobAnnotation>uuid:090fe040e0</JobAnnotation>
    </JobIdentification>
    <Application>
      <ApplicationName>Echo</ApplicationName>
      <ApplicationVersion>1.0</ApplicationVersion>
      <Description>
        This application prints the Hello World message to the standard output
      </Description>
    </Application>
  </JobDescription>
</JobDefinition>
What is REST?

- REpresentational State Transfer

- It is an abstraction of the architectural elements within a distributed hypermedia system

- RESTful HTTP
  - XML + HTTP
RESTful HTTP data elements

- **Resource**
  - the conceptual target of a hypertext reference
- **Resource identifier**
  - URL, URN
- **Representation**
  - HTML document, JPEG image
- **Representation metadata**
  - media type, last-modified time
- **Resource metadata**
  - source link, alternates, vary
- **Control data**
  - if-modified-since, cache-control
Why RESTful HTTP?

- Considered somehow more simple and lightweight than WS
  - Uses standard HTTP methods
  - Reduces interoperability problems
- Many services offered both with WS and RESTful interfaces
  - Google GDATA API
  - Amazon S3
  - ...
Building a RESTful service

1. Model your system as a set of resources
2. Give each resource a URI
3. Define the operations on the resources
   - Operations can be performed using standard HTTP operations (GET, POST, DELETE...)
4. Design the representation served to the client
   - The same information can be represented as plain text, XML, ...
5. Define the error conditions to be handled
   - HTTP error codes
RESTful BES
Identify Resources and Resource Identifiers

- `/`
  - representation of the service capabilities (BES factory attributes document)
- `/status`
  - current status of the BES service
- `/activities/`
  - the list of all activities present in the service submitted to the given share (e.g., batch queue)
- `/activities/id`
  - the current representation of activity id
RESTful BES
Identify Resources and Resource Identifiers

- `/activities/id/submitted`
  - the JSDL document which has been used to instantiate activity `id`
  - In general, the execution service may return a different (processed) JSDL as the “current” JSDL

- `/activities/id/status`
  - the current status of activity `id`
**RESTful BES / Operations**

**CreateActivity**

**Client**

PUT /activities/ HTTP/1.1
Content-Type: application/xml
Content-Length: XXXX

...JSDL document here...

**Server**

HTTP/1.1 201 Created
Location: /activities/ACT0001

HTTP/1.1 501 Not Implemented
Content-Type: application/xml
Content-Length: XXXX

...Unrecognized JSDL element here...

HTTP/1.1 503 Service Unavailable

Activity Successfully Created
Unsupportd element in JSDL
Service not accepting activities
GET /activities/ACT0001 HTTP/1.1

HTTP/1.1 202 Accepted
Content-Type: application/xml
Content-Length: XXXX

...Current JSDL here...

HTTP/1.1 403 Not Found

Activity found

Unknown Activity ID
RESTful BES / Operations

GetActivityStatuses

GET /activities/ACT0001/status HTTP/1.1

HTTP/1.1 202 Accepted
Content-Type: application/xml
Content-Length: XXXX

...ActivityStatusResponse XML doc here...
RESTful BES / Operations

GetFactoryAttributesDocument

Client

GET /?schema=glue HTTP/1.1
Accept: application/xml

HTTP/1.1 202 Accepted
Content-Type: application/xml
Content-Length: XXXX

...service description in GLUE format and XML rendering...

Server

GET /?schema=glue HTTP/1.1
Accept: text/plain

HTTP/1.1 202 Accepted
Content-Type: text/plain
Content-Length: XXXX

...service description in GLUE format and plain text rendering...

Get Service info in XML rendering of GLUE information model

Use GLUE information model with plain text rendering
### RESTful BES / Operations

**IsAcceptingNewActivities/SetAcceptingNewActivities**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Client</strong></td>
<td><strong>Server</strong></td>
</tr>
</tbody>
</table>

**GET /status HTTP/1.1**

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: XXXX

<?xml version="1.0"?>
<serviceStatus status="open"/>
```

**PUT /status HTTP/1.1**

```
PUT /status HTTP/1.1
Content-Type: application/xml
Content-Length: XXXX

<?xml version="1.0"?>
<serviceStatus status="closed"/>
```

HTTP/1.1 200 OK
BES extensions
Change status of an activity

POST /activities/ACT0001/status HTTP/1.1
Content-Type: application/xml
Content-Length: XXXX

...StatusChangeRequest XML doc here...

HTTP/1.1 202 Accepted
Content-Type: application/xml
Content-Length: XXXX

...StatusChangeResponse XML doc here...
BES extensions
Delete (purge) an Activity

Client

DELETE /activities/ACT0001 HTTP/1.1

HTTP/1.1 202 Accepted
Content-Type: application/xml
Content-Length: XXXX

...DeleteResponse XML doc here...

Server
BES batch operations

- Some BES operations can operate on multiple activities
  - e.g., `GetActivityStatuses(A_1, A_2, \ldots, A_N)`

- Encoding multiple activities ID in a single URI

  ```
  GET /activities/ACT001;ACT002;ACT003/status HTTP/1.1
  ```

- Encoding multiple result codes in a single HTTP response

  ```
  HTTP/1.1 202 Accepted
  Content-type: application/xml
  Content-length: XXXX
  
  XML result data structure
  ```
BES extensions

- **Idempotent execution**
  - Makes CreateActivity idempotent

  ```
  POST /activities/ HTTP/1.1
  Pragma: IdempotentActivityID=ABCD0100
  ...
  ```

- **Lifetime Management**
  - Allows clients to specify the maximum lifetime for an activity

  ```
  POST /activities/ HTTP/1.1
  Pragma: InitialTerminationTime=20090524
  ...
  ```

Client-generated nonce
Conclusions

- The BES specification has been defined in term of WS technologies
  - WS are not the only way of implementing the Service Oriented Architecture (SOA) paradigm
- We showed how BES can be implemented as RESTful HTTP
  - Reduces complexity
  - BES operations naturally map to HTTP operations
  - BES extensions can be supported as well
  - Not possible to generate interfaces from WSDL
  - Difficult to encode vector operations
    - Identify multiple activities with a single URI
    - Multiple HTTP return status codes
Conclusions

Thank you!