**Activity Instance Description Specification  
Version 1.0**

Status of This Document

This document provides information to the Grid community regarding the specification of the Activity Instance Description. Distribution is unlimited.

Copyright Notice

Copyright © Open Grid Forum (2009, 2010). All Rights Reserved.

# Abstract

The purpose of this document is to specify the syntax and semantics of the proposed Activity Instance Description. An *activity*, in this context, is a unit of work to be executed on a distributed system. It can be a job, a task, a data processing operation, a data access operation, an application execution, a program execution or a Web Service invocation. The Activity Instance Description captures the information related to this unit of work. Systems built based on this specification can provide a holistic view on an activity, for example for monitoring, auditing or checkpointing purposes. The specification pursues two different but complementary objectives: (i) it specifies a generic and extensible framework to capture the information related to an activity and (ii) it specializes this generic framework using a number of Open Grid Forum specifications.

Contents

Abstract 1

1. Introduction 3

1.1 Motivation 3

1.2 Definition of the Term Activity within the Scope of this Document 4

1.3 Information potentially captured by an Activity 4

1.4 Motivating Use Case – Delegating Activities between Schedulers 4

1.5 Notational Conventions 6

1.6 Namespaces 7

2. Scope of the Specification 7

2.1 On Basic Execution Service 7

2.2 On Job Submission Description Language 7

2.3 On Usage Records 8

3. Activity Instance Description Structure 8

4. The Abstract Activity Instance Element Set 9

4.1 The ActivityDocument Element 9

4.2 The ActivityDescription Element 10

4.3 The ActivityHistory Element 11

4.4 The ActivityHistoryEntry Element 12

4.5 The Status Element 13

4.6 The State Element 15

4.7 The Exception Element 15

4.8 The ActivityDefinition Element 16

4.9 The ActivityDependency Element 17

4.10 The ManagerReference Element 17

4.11 The ResourceUsage Element 18

5. The Open Grid Forum-related Activity Instance Element Set 19

5.1 The JobDefinition Element 19

5.2 The UsageRecord Element 20

5.3 The ActivityStatus Element 21

6. The Activity Instance Element Set not related to the Open Grid Forum 22

6.1 The Exception Element 22

6.2 The Identifier Element 23

6.3 The Reason Element 23

7. Security Considerations 24

8. Contributors 25

9. Intellectual Property Statement 25

10. Disclaimer 25

11. Full Copyright Notice 25

12. References 26

Appendix A The Abstract Activity Instance Description Schema 27

Appendix B Open Grid-Forum-related Activity Instance Schema 35

Appendix C Activity Instance Description Examples 35

# Introduction

The Activity Instance Description captures the information related to an *activity*—a unit of work processed in a distributed system. This description can contain whatever information is relevant to the application domain of the activity. This might be execution-related status data, quality-of-service information, or error messages. All this is captured in an activity instance document and therefore systems built based on the Activity Instance Description can provide a holistic view on an activity, for example for monitoring, auditing or checkpointing purposes.

This document standardizes the description of an activity instance. To this extend, the elements of the activity instance are syntactically and semantically defined. To achieve this, this document pursues two different but complementary objectives: (i) it specifies a generic and extensible framework to capture the information related to an activity (cf. and (ii) it specializes this generic framework using a number of Open Grid Forum (proposed) recommendations. However, this specification does not define how to create an activity instance and how to manage it, as this is seen as implementation-specific.

## Motivation

Information related to an *activity*, such as resource usage, security data, activity state, or data requirements is captured throughout the lifecycle of an activity using a variety of schemata. Furthermore, such information is stored in different ways and by different logical components. This dispersion of activity-related information leads to management, security and logistical overheads in discovering, accessing and using that information. Moreover, it results in an environment where activity information is managed by many systems.

The Job Specification Description Language [JSDL], for example, comprises a core Resource Request Language (RRL) and exists, as pictured in Figure 1, within an environment of other languages like a Scheduling Description Language (SDL), WS-Agreement (WS-AG) [WSAG], a Job Policy Language (JPL), and potentially many more. The activity instance can be used in this context to keep track all of the information related to a job described in JSDL (the activity) and trace historic information.

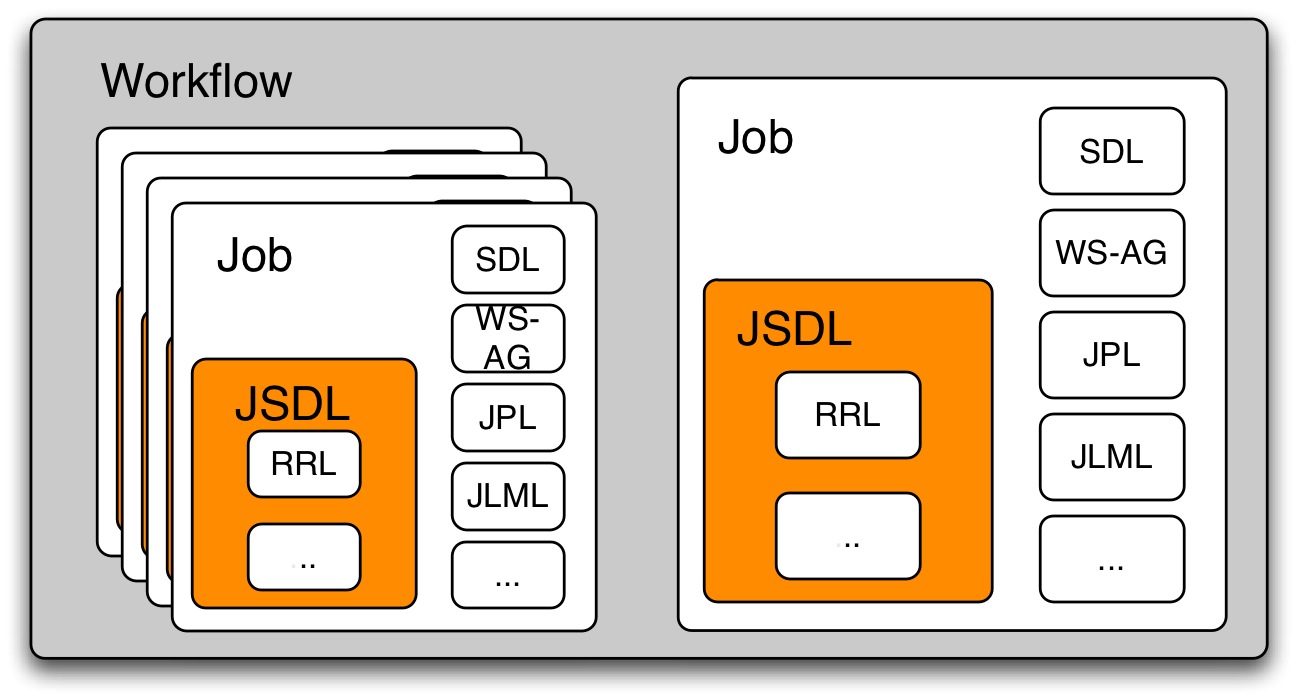


Figure 1. Relationship between JSDL and other specifications

## Definition of the Term Activity within the Scope of this Document

An *activity* is a unit of work to be executed on a distributed system. It can be a job, a task, a data processing operation, a data access operation, an application execution, a program execution, a Web Service invocation, or something else that a user or application needs to do, take care of, or execute.

An *activity* is atomic. This means that an activity is an indivisible unit of work from an activity management perspective. When you stop an activity, you stop all of it, not any one part of it. Moreover, activities can be composed together to form chains of activities that may be managed conditionally, sequentially, or in parallel. Activities can be the atomic nodes in a workflow. Further, they can be conditionally used to process data in a data centric process.

## Information potentially captured by an Activity

The specification provided by this document comprises activity-related information gathered from a number of use cases. Since the Activity Instance Description can potentially capture all information related to an activity and since the kind of information depends on the application domain, the specification includes a number of extension points.

These extension points may be used to add information like the following:

* Dependencies on data and other activities for the composition and management of activities for workflow, scheduling and brokering processes.
* Contextual information, such as:
  + domain (financial markets, weather forecasting, etc.),
  + security (who owns the activity, who is allowed to run it, etc.), or
  + SLAs, QoS and other related policies.
* Monitoring information, such as:
  + accounting, or
  + policy conformance.

## Motivating Use Case – Delegating Activities between Schedulers

This non-normative use case is included in this document to provide an example of what an activity can be and what steps can be included in processing it. The use case also serves as the source for the examples given in the normative sections that specify the XML representation of the Activity Instance Description.

### Actors

The following actors are involved in the delegation use case as shown in Figure 2:

* The *client*, which can be a user accessing the *primary scheduler* directly or a component doing it on the user’s behalf.
* The *primary scheduler* is the entity that receives the activity template and generates the activity instance.
* Depending on the outcome of the different delegations, one or more *secondary schedulers* to which the Activity is delegated.
* The *Basic Execution Service* (BES) [BES] represents the work unit’s execution endpoint. Executes the job related to the activity.
* The *activity store* is a potentially distributed instance where all information about the activity is stored.

### Activity Flow

A client sends an activity request to a scheduler describing the requirements of the submitted work unit[[1]](#footnote-1).

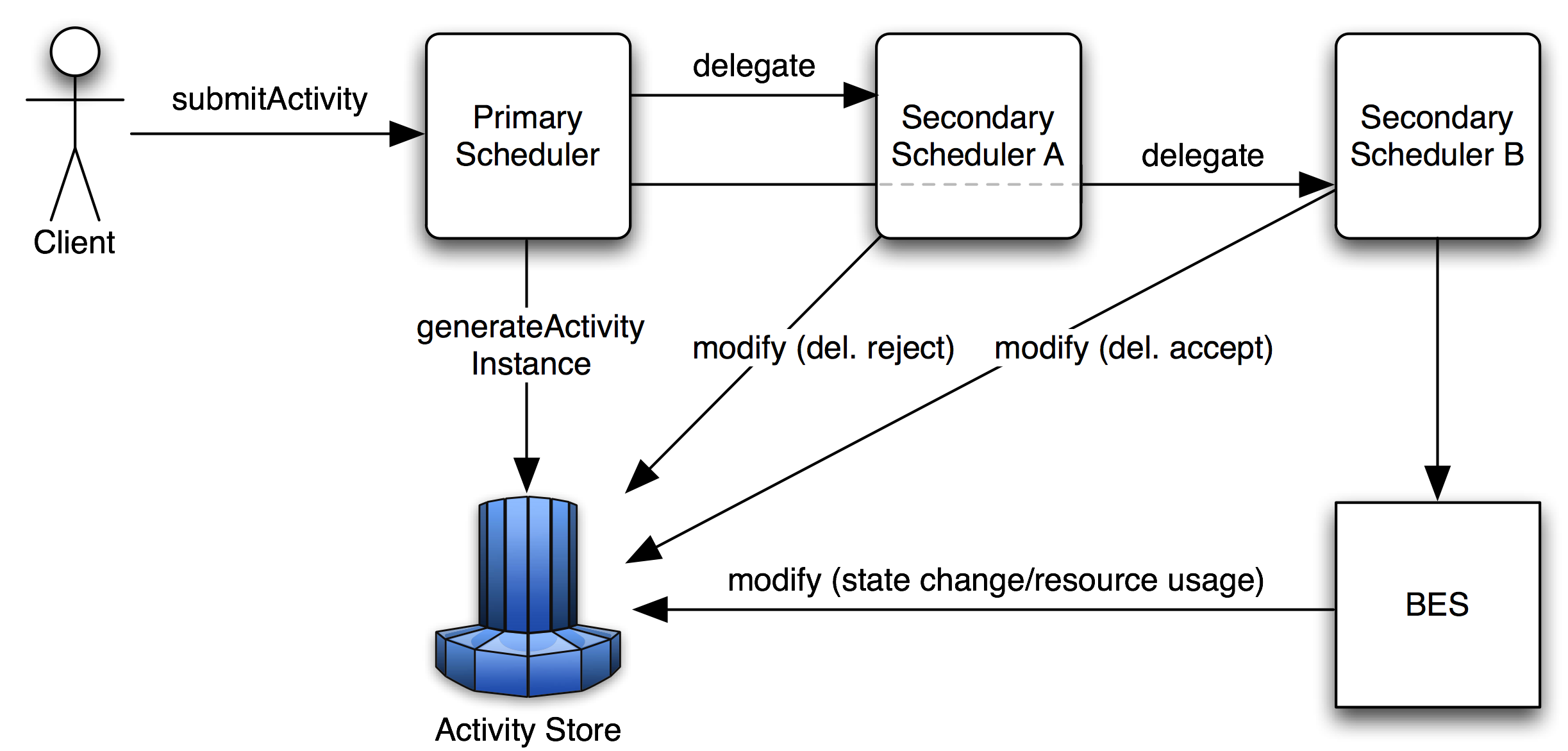


Figure 2. In this example use case and activity is delegated from one scheduler to others to be finally executed by a Basic Execution Service

The initially receiving (primary) scheduler takes the template and, if it is willing to handle it, creates an activity instance for it, storing the initial activity request and, if applicable, additional information. The latter should at least include a provenance record, which denotes that the current scheduler has taken over responsibility for the execution of the given activity. Other information may include scheduling attributes, dependencies on other activities, and the current state of the activity.

On activity delegation, the delegator acts like a client towards the potential delegate, offering the job to another scheduler. Again, if the delegate is willing to accept the job, it takes over responsibility and the provenance records and depending information (e.g. the expected BES) are updated. If necessary, the activity template is modified, as long as the manipulation history is kept. Such modifications may include, as depicted in Figure 2, whether a secondary scheduler rejects or accepts a delegation request, the state transition of the activity, or the resources consumed.

Throughout the whole process, state information is constantly updated in the activity instance. After activity completion, the resource consumption is written to the activity instance. The corresponding entries and dependent parts of the activity instance could then be marked final as to denote the completion of the activity.

### Sequence Diagram

Figure 3 shows the sequence of messages occurring in the example use case according to Figure 2. The primary scheduler accepts the Activity and, since it cannot process it (the reason for which is not of interest), delegates it first to the secondary scheduler A, which rejects the request, and then to secondary scheduler B, which accepts the request, schedules it and hands it over to the BES container for execution. During this process, the following (asynchronous) messages are sent to the activity store:

1. The primary scheduler informs the activity store about the new activity instance (following the acceptance of the activity request).
2. The secondary scheduler A informs the activity store about it rejecting the delegation request (using a modification message).
3. The secondary scheduler B informs the activity store about the acceptance of the delegation request.
4. The secondary scheduler B notifies the activity store about the hand-over of the activity to the BES container.
5. The BES container modifies the activity instance according to the state of the Activity’s execution, informs it about the resources used, etc.

The use case shows, for the sake of simplicity, only the steps until the activity is executed. Further steps that occur during the processing of the activity, like feed-back of results, are not show, nor are potential activity monitoring or assessment steps shown.

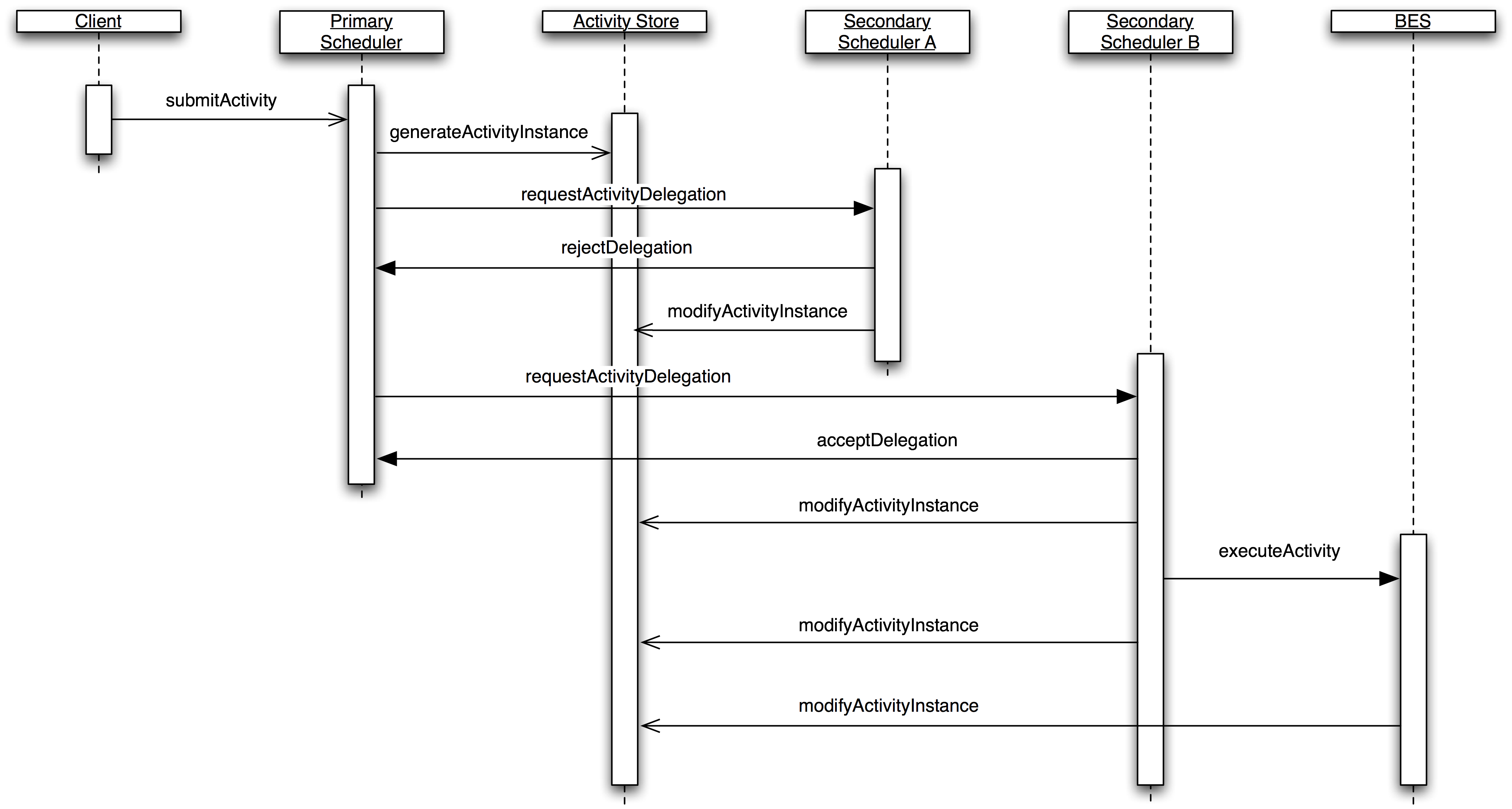


Figure 3. Sequence diagram of the activity delegation use case

## 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 [BRADNER].

This document describes XML Information Sets and inherits the square bracket notation of [INFOSET].

When describing concrete XML schemas [SCHEMA1], [SCHEMA2], this specification uses the notational convention of WS-Security

[WSSEC]. Specifically, each member of an element’s [children] or [attributes] properties, is described using an XPath-like notation (e.g.: **/x:MyHeader/x:SomeProperty/@value1**). The use of **{any}** indicates the presence of an element wildcard (<xsd:any/>). The use of **@{any}** indicates the presence of an attribute wildcard (<xsd:anyAttribute/>).

Pseudo-schemas are provided for each component, before the description of the component. They use BNF-style conventions for attributes and elements: ‘?’ denotes zero or one occurrences; ‘\*’ denotes zero or more occurrences; ‘+’ denotes one or more occurrences. Attributes (other than the abstract and substitutes special attributes) are conventionally assigned a value that corresponds to their type, as defined in the normative schema.

<!-- sample pseudo-schema -->

<defined\_element>

required\_attribute\_of\_type\_string="xsd:string"

optional\_attribute\_of\_type\_int="xsd:int"? >

<required\_element />

<optional\_element />?

<one\_or\_more\_of\_this\_element />+

</defined\_element>

## Namespaces

This specification uses namespace prefixes throughout; they are listed in Table 1‑1. Note that the choice of any namespace prefix is arbitrary and not semantically significant.

Table 1‑1. Prefixes and namespaces used in this specification

|  |  |
| --- | --- |
| **Prefix** | **Namespace** |
| aid | http://schemas.ogf.org/jsdl/2010/06/activity-instance-description |
| aid-ogf | http://schemas.ogf.org/jsdl/2010/06/activity-instance-description-ogf |
| bes-factory | http://schemas.ggf.org/bes/2006/08/bes-factory |
| jsdl | http://schemas.ggf.org/jsdl/2005/11/jsdl |
| jsdl-posix | http://schemas.ggf.org/jsdl/2005/11/jsdl-posix |
| ur | http://schemas.ogf.org/urf/2003/09/urf |
| wsa | http://www.w3.org/2005/08/addressing |
| xsd | http://www.w3.org/2001/XMLSchema |
| xsi | http://www.w3.org/2001/XMLSchema-instance |

# Scope of the Specification

The Activity Instance Description specification defines the overall structure of the activity document and the semantics associated with each element. It is expected that a number of other specifications will provide the detailed information contained in each document element, depending on the domain in which the Activity Instance Description specification is used. An overview of specifications from the Open Grid forum that are used with the Activity Instance Description specification is given here.

## On Basic Execution Service

The Basic Execution Service (BES) [BES] defines a service to which clients can send requests to initiate, monitor, and manage computational activities. In addition to an information model and a set of port types, the BES specification also defines an extensible state modelfor activities. The Activity Instance Description specification uses the BES state model to record the various states the activity goes through during its lifetime.

## On Job Submission Description Language

The Job Submission Description Language (JSDL) [JSDL] is a language for describing the requirements of computational jobs for submission to resources, particularly in Grid environments where interactions between a number of different types of job management systems is common. A JSDL document may be transformed by intermediaries or refined further by information not available to the initial submitter of that job. Therefore in the context of the Activity Instance Description specification, JSDL can be used to describe both the initial submission that created the Activity Instance as well as the results of transformations to the activity definition due to processing steps like delegation or negotiation.

## On Usage Records

The Usage Record (UR) [UR] is an XML document language for describing units of accounting system data (e.g., batch scheduler log entries) in an interoperable exchange format. Usage records are focused on describing what a particular unit of work (e.g., computational job) actually consumed in terms of resources, and are used on Grids particularly when the organization that actually carried out the job is different from the organization that is paying for it, and are already widely used in production Grid deployments, such as TeraGrid and DEISA. Within the context of the Activity Instance Description specification, URs can be used to describe the resources (both hardware and software) that have been actually consumed/used by the Activity instance over time, as well as sample points of current allocation levels, allowing both system monitoring and accounting within the context of the overall activity lifecycle.

# Activity Instance Description Structure

An Activity Instance Description is organized as follows: The root element, *ActivityDocument*, contains an optional human readable description of the activity (i.e. the *Activity Description*); and a sequence of one or more historic entries, i.e. a list of *ActivityHistoryEntry* elements. Each *HistoryEntry* element contains information about the Activity at a specific point in time: status, definition, dependencies, reference to the activity’s manager, and resource usage. The pseudo schema definition is given below.

<ActivityDocument>

<ActivityDescription .../>?

<ActivityHistory .../>

<ActivityHistoryEntry>+

<xsd:any##other/>\*

<Status ...>

<ActivityDefinition ... />?

<ActivityDependency ... />?

<ManagerReference .../>

<ResourceUsage .../>?

</ActivityHistoryEntry>

<xsd:any##other/>\*

</ActivityDocument>

Normative definitions of the Activity Instance Description elements are given in Sections 4 and 5. The Activity Instance Description XML schema is listed in Appendix A, examples of Activity Documents are given in Appendix B. The examples given in the following sections are non-normative and are solely added for illustrative purposes.

The Activity Instance Description schema supports an open content model. Entities accessing information contained in Activity Documents may not necessarily recognize all the extensions used. Implementations SHOULD ignore all extensions they do not support.

# The Abstract Activity Instance Element Set

## The ActivityDocument Element

### Definition

This element is the root of a single Activity Instance Description, which contains an activity's meta-data and history and provides the entry point for every activity. While the meta-data part MAY carry information about the activity's creator, purpose, and references (i.e. to other activities), the history part SHOULD describe the full lifecycle of the activity.

### Multiplicity

The multiplicity of this element is one.

### Type

This is an xsd:complexType. It MUST support the following elements:

* *ActivityDescription* (cf. 4.2)
* *ActivityHistory* (cf. 4.3)

### Attributes

The following attributes are defined:

* *id* - An identifier for the activity, which MUST be globally unique. It is RECOMMENDED to use Universally Unique Identifiers (UUID) as described in RFC4122 [UUID].

### XML Representation

The *ActivityDocument* is rendered in XML as:

<ActivityDocument id=”xsd:string” xsd:any##other\*>

<ActivityDescription ... />?

<ActivityHistory ... />

<xsd:any##other/>\*

</ActivityDocument>

Where:

**/aid:ActivityDocument**

Represents the ActivityDocument element.

/**aid:ActivityDocument/ActivityDescription**

Represents the *ActivityDescription* element as defined in Section 4.2. This element MAY appear zero or one time.

**/aid:ActivityDocument/ActivityHistory**

Represents the *ActivityHistory* element as defined in Section 4.3. This element MUST appear exactly once.

### Example

The following example shows the *ActivityDocument* element, which carries an id attribute following the UUID specification. Examples for ActivityDescription and ActivityHistory are given in Section 4.2 and Section 4.3, respectively.

<ActivityDocument id=”ea196512-9cb7-4a14-91b0-2dde749a5f7d”>

<ActivityDescription> ... </ActivityDescription>

<ActivityHistory> ... </ActivityHistory>

</ActivityDocument>

### Extensibility

This element provides two extension points, one for element extensibility and one for attribute extensibility. The former extension point MAY be used to attach XML digital signatures [XMLDSIG] to an ActivityHistoryEntry element (see Section 7). Implementations SHOULD ignore unsupported extensions.

## The ActivityDescription Element

### Definition

This element contains a natural-language description of the activity and offers means for storing additional information on the activity for displaying purposes (e.g. in a user interface).

### Multiplicity

The multiplicity of this element is zero or one.

### Type

This type of this element is xsd:string.

### Attributes

No attributes are defined.

### Pseudo schema

The *ActivityDescription* is rendered in XML as:

<ActivityDescription> xsd:string </ActivityDescription>

Where:

**/aid:ActivityDocument/ActivityDescription**

Represents the ActivityDescription element.

### Example

<ActivityDescription>

This activity instance has been generated due to an activity request  
 submitted to the scheduling service with the  
 following URI: http://tempuri.org/services/activityscheduler. The  
 activity request has been received at 2010-05-10T11:11:11.11. The

activity instance has been created 2010-05-10T11:11:44.44 by the   
 organization’s activity store with the following URI:

http://tempuri.org/services/activitystore.

</ActivityDescription>

The example shows a human-readable description of an activity following the example given in Section 1.4. There, an activity request is accepted by a scheduler for processing, which then triggers the creation of an activity instance.

### Extensibility

This element provides no extension points.

## The ActivityHistory Element

### Definition

This element stores the history of an activity. Within it, a record of the activity’s history is kept track of. This record MUST contain one initial and, at most, one final record (see Section 4.4 for the *ActivityHistoryEntry* element and for the different category attributes). Note that, although a final record MAY be written already, the activity itself MAY still be modified.

### Multiplicity

The multiplicity of this element is one.

### Type

This is an xsd:complexType. It MUST support the following elements:

* *ActivityHistoryEntry*

### Attributes

No attributes are defined.

### Pseudo schema

The *ActivityHistory* is rendered in XML as:

<ActivityHistory>

<ActivityHistoryEntry/>+

</ActivityHistory>

Where:

**/aid:ActivityDocument/ActivityHistory** Represents the ActivityHistory element.

**/aid:ActivityDocument/ActivityHistory/ActivityHistoryEntry**

Represents the *ActivityHistoryEntry* element as introduced in Section 4.4. This element MUST appear at least once within the *ActivityHistory* element.

### Example

<ActivityHistory>

<ActivityHistoryEntry> ... </ActivityHistoryEntry>

<ActivityHistoryEntry> ... </ActivityHistoryEntry>

</ActivityHistory>

This example shows a history of an activity with currently only two entries. Following the example in Section 1.4, the history would contain exactly two entries after the first delegation attempt of the primary scheduler to secondary scheduler A. An example for the *ActivityHistoryEntry* is given in Section 4.4.

### Extensibility

This element provides no extension points.

## The ActivityHistoryEntry Element

### Definition

This element stores a single event in an activity's history. It denotes an event in the history of an activity, containing its properties at the time the event occurred. Every entry MUST contain at least a timestamp (as attribute), the status of the activity at this timestamp, and a WS-Addressing [WSADDR] endpoint reference to the managing service. Once an *ActivityHistoryEntry* is written, it MUST NOT be altered. Additional information about the respective activity has to be appended to the ActivityHistory by adding a new *ActivityHistoryEntry* element.

### Multiplicity

The multiplicity of this element is one or more.

### Type

This is an xsd:complexType. It MUST support the following elements:

* *Status*
* *ActivityDefinition*
* *ActivityDependency*
* *ManagerReference*
* *ResourceUsage*

### Attributes

The following attributes are defined:

* *timestamp* – the timestamp of the entry. It MUST appear exactly once. Its type is xsd:dateTime. It keeps the timestamp at which this event has occurred in the activity's history. The entries in the whole activity history SHOULD be ordered ascending to their timestamp.
* *category* – the category of an entry. The attribute MAY appear zero or one time. Its type is aid:ActivityHistoryEntryCategoryType. Denotes the category of this history record. The possible options are "initial", "intermediate", and “final”. Note that there MUST exist at least an initial and a final entry to describe the entire lifecycle of an activity. The semantics of the options is as follows:
  + *initial* denotes the initial history record for a given activity, which MUST be the first one created in the whole record. Note that this is not necessarily the first action taken on a certain activity instance; which events are to be recorded as a history record is implementation-specific. The initial ActivityHistoryEntry MUST NOT have an empty ActivityDefinition element.
  + *intermediate* denotes an intermediate history record for a given activity. Such entry MAY NOT be the first or last one created in the whole record.
  + *final* denotes the final history record for a given activity, which MUST be the last one created in the whole record. Note that this does not imply that the activity on its whole MAY NOT be modified any more.

### Pseudo schema

The *ActivityHistoryEntry* is rendered in XML as:

<ActivityHistoryEntry timestamp=”xsd:dateTime  
 category=”ActivityHistoryEntryCategoryType”?  
 xsd:any##other\*>

<Status/>

<ActivityDefinition/>?

<ActivityDependency/>?

<ManagerReference/>

<ResourceUsage/>?

</ActivityHistoryEntry>

Where:

**/aid:ActivityDocument/ActivityHistory/ActivityHistoryEntry**

Represents the ActivityHistoryEntry element.

**/aid:ActivityDocument/ActivityHistory/ActivityHistoryEntry/Status**

Represents the *Status* element as defined in Section 4.5. It MUST be present exactly once.

**/aid:ActivityDocument/ActivityHistory/ActivityHistoryEntry/ActivityDefinition**

Represents the *ActivityDefinition* element as defined in Section 4.8. It is an optional element, which MAY appear zero or one times.

**/aid:ActivityDocument/ActivityHistory/ActivityHistoryEntry/ActivityDependency**

Represents the *ActivityDependency* element as defined in Section 4.9. It is an optional element, which MAY appear zero or one times.

**/aid:ActivityDocument/ActivityHistory/ActivityHistoryEntry/ManagerReference**

Represents the *ManagerReference* element as defined in Section 4.10. It is an mandatory element, which MUST appear exactly once.

**/aid:ActivityDocument/ActivityHistory/ActivityHistoryEntry/ResourceUsage**

Represents the *ResourceUsage* element as defined in Section 4.11. It is an mandatory element, which MAY appear zero or one times.

### Example

<ActivityHistoryEntry timestamp=”2010-05-10T11:11:44.44”

category=”initial”>

<Status> ... </Status>

<ActivityDefinition> ... </ActivityDefinition>

<ActivityDependency> ... </ActivityDependency>

<ManagerReference> ... </ManagerReference>

<ResourceUsage> ... </ResourceUsage>

</ActivityHistoryEntry>

This example shows the *timestamp* and *category* attributes for the initially generated activity instance according to the example given in Section 1.4. All the other elements have examples in the respective sections below.

### Extensibility

This element provides two extension points, one for element extensibility and one for attribute extensibility. The former extension point MAY be used to attach XML digital signatures [XMLDSIG] to an ActivityHistoryEntry element (see Section 7). Implementations SHOULD ignore unsupported extensions.

## The Status Element

### Definition

This element stores the status of the activity with respect to the enclosing history record. The status of an activity comprises its current state (defined by an appropriate state model) and, if necessary, exception information. Every status record for an activity MUST contain at least the current state; if an exceptional condition occurs during the activity's lifetime, it SHOULD be also recorded here. Note that the existence of an exception entry is not necessarily coupled to a corresponding exceptional state; a possible connection between these is left to the implementor and SHOULD be described in the concrete state model's documentation.

### Multiplicity

The multiplicity of this element is one.

### Type

This is an xsd:complexType. It MUST support the following elements:

* *State*
* *Exception*

### Attributes

No attributes are defined.

### Pseudo schema

The *Status* is rendered in XML as:

<Status>

<State/>

<Exception/>?

</Status>

Where:

**/aid:ActivityDocument/ActivityHistory/ActivityHistoryEntry/Status**

Represents the Status element.

**/aid:ActivityDocument/ActivityHistory/ActivityHistoryEntry/Status/State**

Represents the *State* element as defined in Section 4.6. It is a mandatory element, which MUST appear exactly once.

**/aid:ActivityDocument/ActivityHistory/ActivityHistoryEntry/Exception**

Represents the *Exception* element as defined in Section 4.7. It is an optional element, which MAY appear zero or one time.

### Example

<Status>

<State> ... </State>

<Exception> ... </Exception>

</Status>

An example for *State* is given in Section 5.3 while Section 6.1 features an example for the *Exception* element.

### Extensibility

This element provides no extension points

## The State Element

### Definition

This element denotes details on the state of the activity with respect to the enclosing history record. More specifically, it stores a state model instance for the given activity state.

### Multiplicity

The multiplicity of this element is one.

### Type

This is an abstract type and has to be substituted by an appropriate definition (see Section 5.3).

### Attributes

No attributes are defined.

### Pseudo schema

The abstract *State* has no rendering:

<State abstract=”true”/>

Where:

**/aid:ActivityDocument/ActivityHistory/ActivityHistoryEntry/Status/State**

Represents the State element. Since the *State* element is an abstract XML element it  
MUST NOT occur in a XML document by itself. It MUST be substituted by a valid *State* substituent instead.

### Extensibility

States are specific to activities. The minimal definition provided in Section 5.3 SHOULD be used if there is no applicable specification.

## The Exception Element

### Definition

This element denotes details on an exception for the activity with respect to the enclosing record. More specifically, it stores an exception model instance for the given activity fault.

### Multiplicity

The multiplicity of this element is zero or one.

### Type

This is an abstract type and has to be substituted by an appropriate definition (see Section 6.1).

### Attributes

No attributes are defined.

### Pseudo schema

The abstract *Exception* has no rendering:

<Exception abstract="true"/>

Where:

**/aid:ActivityDocument/ActivityHistory/ActivityHistoryEntry/Status/Exception**

Represents the Exception element. Since the Exceptionelement is an abstract XML element it MUST NOT occur in a XML document by itself. It MUST be substituted by a valid Exceptionsubstituent instead.

### Extensibility

Exceptions are specific to activities. The minimal definition provided in Section 6.1 SHOULD be used if there is no applicable specification. Other possible substitutions may include, for example, SOAP faults.

## The ActivityDefinition Element

### Definition

This element stores the definition of an activity with respect to the enclosing history record. More specifically, it defines the requirements of an activity, for example, the description template used to create the activity. The initial definition (template) MAY change over time due to refinement of the requirements as a result of scheduling, delegation, or negotiation processes, etc. Therefore this element MAY appear in more than one *ActivityHistoryEntry*.

### Multiplicity

The multiplicity of this element is zero or one. Every ActivityHistory record MUST contain at least one *ActivityHistoryEntry* with a non-empty ActivityDefinition element. In particular, the “initial” ActivityHistoryEntry MUST NOT have an empty ActivityDefinition element.

### Type

This is an abstract type and has to be substituted by an appropriate definition (see Section 5.1).

### Attributes

No attributes are defined.

### Pseudo schema

The *ActivityDefinition* is rendered in XML as:

<ActivityDefinition abstract="true"/>

Where:

**/aid:ActivityDocument/ActivityHistory/ActivityHistoryEntry/ActivityDefinition**

Represents the ActivityDefinition element. Since the ActivityDefinitionelement is an abstract XML element it MUST NOT occur in a XML document by itself. It MUST be substituted by a valid ActivityDefinitionsubstituent instead.

### Extensibility

The definition provided in Section 5.1 SHOULD be used unless a more applicable specification is available.

## The ActivityDependency Element

### Definition

This element stores the dependency definitions for the activity with respect to the enclosing history record. More specifically, it describes links to associated activities within a dependency structure (such as a workflow). It does not have to contain the entire dependency structure, but the dependencies to other activities.

### Multiplicity

The multiplicity of this element is zero or one.

### Type

This is an abstract type and has to be substituted by an appropriate definition.

### Attributes

No attributes are defined.

### Pseudo schema

The abstract *ActivityDependency* has no rendering:

<ActivityDependency abstract=”true”/>

Where:

**/aid:ActivityDocument/ActivityHistory/ActivityHistoryEntry/ActivityDependency**

Represents the ActivityDependency element. Since the ActivityDependencyelement is an abstract XML element it MUST NOT occur in a XML document by itself. It MUST be substituted by a valid ActivityDependencysubstituent instead.

### Extensibility

No concrete extensions are provided for this element. It is however RECOMMENDED to use available standards.

## The ManagerReference Element

### Definition

This element keeps the endpoint reference of the activity's managing service at the time denoted by the enclosing record. The corresponding service SHOULD expose an interface for managing the activity's state, lifecycle, and execution.

### Multiplicity

The multiplicity of this element is one.

### Type

This is an external type. Refer to the WS-Addressing specification [WSADDR] for further details.

### Pseudo schema

The external *Manager Reference* has no rendering:

<ManagerReference external=”true”/>

Where:

**/aid:ActivityDocument/ActivityHistory/ActivityHistoryEntry/ManagerReference**

Represents the ManagerReference element.

### Extensibility

This element provides no extension points.

### Example

<wsa:EndpointReference>

<wsa:Address>http://tempuri.org/services/activitystore</wsa:Address>

</wsa:EndpointReference>

This endpoint shows the address of the activity store (see Section 1.4) formatted according to the Web Services Addressing standard.

## The ResourceUsage Element

### Definition

This element stores the resource usage for this activity with respect to the enclosing history record. It describes the resource consumption/usage of an activity, e.g., the number of CPUs used or maximum memory needed for some part of the activity. This element may appear multiple times for an activity, depending on the monitoring policies of the system generating them; the system may choose to perform averaging over the execution time, averaging over several periods that cover the execution time, sampling of the system, etc.

Concretizations of this element SHOULD describe what time instant or time period they apply to. Because several monitoring systems may be feeding usage information into the activity instance document, the time points/periods MAY be overlapping and MAY be non-contiguous.

Note that there is no requirement for the information in the activity instance document to be either accurate or timely.

### Multiplicity

The multiplicity of this element is zero or more.

### Type

This is an abstract type and has to be substituted by an appropriate definition (see Section 5.2).

### Attributes

No attributes are defined.

### Pseudo schema

The abstract *ResourceUsage* has no rendering:

<ResourceUsage abstract="true"/>

Where:

**/aid:ActivityDocument/ActivityHistory/ActivityHistoryEntry/ResourceUsage**

Represents the ResourceUsage element. Since the ResourceUsageelement is an abstract XML element it MUST NOT occur in a XML document by itself. It MUST be substituted by a valid ResourceUsagesubstituent instead.

### Extensibility

The definition provided in Section 5.2 SHOULD be used unless a more applicable specification is available.

# The Open Grid Forum-related Activity Instance Element Set

This section defines how existing Open Grid Forum specifications are used in an Activity Instance Description document as substituents for the respective abstract elements defined in Section 4.

## The JobDefinition Element

The *JobDefinition* element substitutes the abstract type *ActivityDefinition* (see Section 4.8) and defines the activity’s requirements using the Job Submission Description Language (JSDL) [JSDL]. All elements of the JSDL specification and extensions MAY be used when defining an Activity.

### Multiplicity

The multiplicity of this element is one.

### Type

The type of this element is jsdl:JobDefinition.

### Attributes

No attributes are defined.

### Pseudo schema

The *JobDefinition* is rendered in XML as:

<JobDefinition substitutes="aid:ActivityDefinition"/>

Where:

**/aid:ActivityDocument/ActivityHistory/ActivityHistoryType/JobDefinition**

Represents the JobDefinition element.

### Example

The following example is derived from Appendix 4 of GFD.136 [JSDL] and has been adapted to the needs of this specification.

<aid-ogf:JobDefinition>

<jsdl:JobDescription>

<jsdl:JobIdentification>

<jsdl:JobName> My gnuplot invocation</jsdl:JobName>

<jsdl:Description> Simple application invocation

</jsdl:Description>

</jsdl:JobIdentification>

<jsdl:Application>

<jsdl:ApplicationName>gnuplot</jsdl:ApplicationName>

<jsdl-posix:POSIXApplication>

<jsdl-posix:Executable>/usr/local/bin/gnuplot

</jsdl-posix:Executable>

<jsdl-posix:Argument>control.txt</jsdl-posix:Argument>

<jsdl-posix:Input>input.dat</jsdl-posix:Input>

<jsdl-posix:Output>output1.png</jsdl-posix:Output>

</jsdl-posix:POSIXApplication>

</jsdl:Application>

<jsdl:Resources>

<jsdl:IndividualPhysicalMemory>

<jsdl:LowerBoundedRange>1293942784.0  
 </jsdl:LowerBoundedRange>

</jsdl:IndividualPhysicalMemory>

<jsdl:TotalCPUCount><jsdl:Exact>1.0</jsdl:Exact>

</jsdl:TotalCPUCount>

</jsdl:Resources>

<jsdl:DataStaging>

<jsdl:FileName>control.txt</jsdl:FileName>

<jsdl:CreationFlag>overwrite</jsdl:CreationFlag>

<jsdl:DeleteOnTermination>true</jsdl:DeleteOnTermination>

<jsdl:Source>

<jsdl:URI>http://tempuri.org/~me/control.txt</jsdl:URI>

</jsdl:Source>

</jsdl:DataStaging>

</jsdl:JobDescription>

</aid-ogf:JobDefinition>

This JSDL job definition is the actual activity submitted in the first step to the primary scheduler (according to Figure 2).

## The UsageRecord Element

The *UsageRecord* element substitutes the abstract type *ResourceUsage* (as defined in Section 4.11) and defines a particle of usage information in a format that is compatible with the OGF Usage Record specification [UR]. All elements defined by the specification and extensions MAY be used when describing an activity’s resource usage. Where multiple Usage Records are created during the execution of an activity, multiple *UsageRecord* elements MAY be created within multiple *ActivityHistoryEntry* elements. Newer Usage Records do not necessarily make older ones obsolete. Consumer-side processing of Usage Records is outside the scope of this document. It is to be noted that according to the UR specification, the Usage Record’s ur:StartTime and ur:EndTime elements MUST be used to determine the periods that individual *UsageRecord* elements refer to.

### Multiplicity

The multiplicity of this record is zero or more (there may be an arbitrary number of usage records associated with an activity instance).

### Type

The type of this element is ur:UsageRecordType.

### Attributes

No attributes are defined.

### Pseudo schema

The *UsageRecord* is rendered in XML as:

<UsageRecord substitutes="aid:ResourceUsage">

...

</UsageRecord>

Where:

**/aid:ActivityDocument/ActivityHistory/ActivityHistoryType/**UsageRecord

Represents the *UsageRecord* element.

For further elements included in the *UsageRecord* element, please refer to the respective specification [UR].

### Example

This example is derived from Section 14.1 of GFD.98 [UR] and has been adapted to the needs of this specification.

<aid-ogf:UsageRecord>

<ur:RecordIdentity

ur:recordId="http://tempuri.org/mscf/colony/PBS.1234.0"

ur:createTime="2010-05-10T11:44:44.44" />

<ur:JobIdentity>

<ur:LocalJobId> PBS.1234.0 </ur:LocalJobId>

</ur:JobIdentity>

<ur:UserIdentity>

<ur:LocalUserId> scottmo </ur:LocalUserId>

</ur:UserIdentity>

<ur:Charge> 2870 </ur:Charge>

<ur:Status> completed </ur:Status>

<ur:Memory ur:storageUnit="MB"> 1234 </ur:Memory>

<ur:ServiceLevel ur:type="QOS"> Gold level </ur:ServiceLevel>

<ur:Processors> 1 </ur:Processors>

<ur:ProjectName> mscfops </ur:ProjectName>

<ur:MachineName> Colony </ur:MachineName>

<ur:WallDuration> PT1S </ur:WallDuration>

<ur:StartTime> 2010-05-10T11:22:22.22 </ur:StartTime>

<ur:EndTime> 2010-05-10T11:33:33.33 </ur:EndTime>

<ur:NodeCount> 1 </ur:NodeCount>

<ur:Queue> batch </ur:Queue>

<ur:Resource ur:description="quoteId"> 1435 </ur:Resource>

<ur:Resource ur:description="application"> gnuplot </ur:Resource>

<ur:Resource ur:description="executable"> gnuplot </ur:Resource>

</aid-ogf:UsageRecord>

## The ActivityStatus Element

The *ActivityStatus* element substitutes the abstract type State (see Section 4.5) and contains the activity’s state using the Basic Execution Service (BES) state model. It supports the same states and state extensibility model as the BES specification. For details refer to Section 4 of GFD.108 [BES].

### Multiplicity

The multiplicity of this element is one.

### Type

The type of this element is bes-factory:ActivityStatusType.

### Attributes

No additional attributes are defined.

### Pseudo schema

The *ActivityStatus* is rendered in XML as:

<aid-ogf:ActivityStatus substitutes="aid:State"/>

Where:

**/aid:ActivityDocument/ActivityHistory/ActivityHistoryType/Status/ActivityStatus**

Represents the ActivityStatus element.

### Example

The activity is in the BES Running state, and in an activity-specific sub-state of staging in files.

<aid-ogf:ActivityStatus state="Running">

<n00:Staging-In/>

</aid-ogf:ActivityStatus>

With respect to the example in Section 1.4, this state is reached after the activity has been handed over to the BES for execution.

# The Activity Instance Element Set not related to the Open Grid Forum

This section provides definitions for abstract element substitutions where no OGF specifications exist.

## The Exception Element

### Definition

The Exception element substitutes the *Exception* abstract type (see Section 4.7) and provides additional information about abnormal state change of the Activity. This is a basic definition as there is no applicable OGF specification at this time.

### Multiplicity

The multiplicity of this element is one.

### Type

This is an xsd:complexType. It MUST support the following elements:

* *Identifier*
* *Reason*

### Attributes

No attributes are defined.

### Pseudo schema

The *Exception* is rendered in XML as:

<Exception substitutes="aid:Exception">

<Identifier> xsd:string </Identifier>

<Reason> xsd:string </Reason>

</Exception>

Where:

**/aid:ActivityDocument/ActivityHistory/ActivityHistoryType/Status/Exception**

Represents the Exception element.

**/aid:ActivityDocument/ActivityHistory/ActivityHistoryType/Status/Exception/Identifier**

Represents the *Identifier* element as defined in Section 6.2. It is a mandatory element, which MUST appear exactly once.

**/aid:ActivityDocument/ActivityHistory/ActivityHistoryType/Status/Exception/Reason**

Represents the *Reason* element as defined in Section 6.3. It is a mandatory element, which MUST appear exactly once.

### Example

An exception was raised because the activity ran out of storage.

<aid-ogf:Exception>

<aid-ogf:Identifier> InsufficientStorage </aid-ogf:Identifier>

<aid-ogf:Reason> Storage quota reached </aid-ogf:Reason>

</aid-ogf:Exception>

## The Identifier Element

### Definition

This element identifies the raised exception by name. It provides information on the kind of exception raised. There are no format requirements.

### Multiplicity

The multiplicity of this element is one.

### Type

The type of this element is xsd:string.

### Attributes

No attributes are defined.

### Pseudo schema

The *Identifier* is rendered in XML as:

<Identifier> xsd:string </Identifier>

Where:

**/aid:ActivityDocument/ActivityHistory/ActivityHistoryType/Status/Exception/Identifier**

Represents the Identifier element.

### Example

A component of the activity could not write its data to storage due to insufficient storage.

<aid-ogf:Identifier> InsufficientStorage </aid-ogf:Identifier>

## The Reason Element

### Definition

This element provides additional information about the raised exception. There are no format requirements.

### Multiplicity

The multiplicity of this element is one.

### Type

The type of this element is xsd:string.

### Attributes

No attributes are defined.

### Pseudo schema

The *Reason* is rendered in XML as:

<aid-ogf:Reason> xsd:string </aid-ogf:Reason>

Where:

**/aid:ActivityDocument/ActivityHistory/ActivityHistoryType/Status/Exception/Reason**

Represents the Reason element.

### Example

The reason for the component failing to write to storage was that its quota was reached.

<aid-ogf:Reason> Storage quota reached </aid-ogf:Reason>

# Security Considerations

There are two key security considerations in relation to activity instance documents, being the privacy of the data within the document, and integrity of that data.

Because an activity instance document can contain much information that is in need of being secured, it is important that services and resources handling these documents take care to ensure that appropriate access controls are applied. The definition of such rules lies outside the scope of this specification, as is the description of how those rules are to be attached to the activity instance document.

Because an entire activity instance document, or parts thereof, may be passed between many systems between its originating source system and the eventual consumers of the data (e.g., a principal investigator or funding organization) and the fact that the document may be used for making decisions on payments for work done, it is important for the consumers of the activity instance document to be able to determine that the document they see is what was originally provided. The source system may attach XML digital signatures [XMLDSIG] to individual *ActivityHistoryEntry* elements; or to the overall *ActivityDescription* element. Signing the overall document ensures its integrity (as well as provides the ability to check who was responsible for creating it). It also has the effect of sealing that particular version of the document (though future versions of the document may also be created, at a cost of requiring some entity to recreate the signature once again).

This specification does not recommend any specific normalization or signing algorithms, though it is noted that algorithms that depend on the presence of ID attributes on elements or which depend on absolute XPath addressing [XPATH] are NOT RECOMMENDED as that makes those documents difficult to aggregate. It is RECOMMENDED that in order to gain maximal efficiency, originating source systems delay generating a signature for a document until they believe they have accumulated all the relevant *ActivityHistoryEntry* elements.

# Contributors

Donal Fellows

Research Computing Services  
The University of Manchester  
Devonshire House, Precinct Centre, Oxford Road  
Manchester M13 9PL

donal.k.fellows@manchester.ac.uk

Alexander Papaspyrou

Robotics Research Institute

TU Dortmund University

44227 Dortmund

alexander.papaspyrou@tu-dortmund.de

Andreas Savva

Cloud Computing Research Center

Fujitsu Laboratories

4-1-1, Kamikodanaka, Nakahara, Kawasaki City, Japan

Email: andreas.savva@jp.fujitsu.com

Philipp Wieder

Service Computing Group/ITMC

TU Dortmund University

44227 Dortmund

philipp.wieder@udo.edu

The authors would also like to thank Ali Anjomshoaa, Fred Brisard, Steve McGough, Neil Chue Hong, Shahbaz Memon, Shiraz Memon, Henning Mersch, Chris Smith, Wolfgang Ziegler and the people from the NextGRID project for their valuable input and the time they have committed to this specification or the foundations of it.

# 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.

# 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.

# Full Copyright Notice

Copyright (C) Open Grid Forum (2009, 2010). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the OGF or other organizations, except as needed for the purpose of developing Grid Recommendations in which case the procedures for copyrights defined in the OGF Document process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the OGF or its successors or assignees.

# References

[BES] Foster, I., Grimshaw, A., Lane, P., Lee, W., Morgan, M., Newhouse, S., Pickles, S., Pulsipher, D., Smith, C., and Theimer, M. OGSA® Basic Execution Service Version 1.0, Grid Forum Document, GFD.108, Open Grid Forum, Lemont, Illinois, U.S.A. August 2007.

[BRADNER] Bradner, S., Key Words for Use in RFCs to Indicate Requirement Levels, RFC 2119. March 1997.

[BRAY] Bray, T., Hollander, D., Layman, A., Tobin, R., and Thompson, H.S., Namespaces in XML 1.0 (Third Edition), W3C Recommendation. December 2009.

[INFOSET] Cowan, J. and Tobin, R., XML Information Set (Second Edition), W3C Recommendation. February 2004.

[JSDL] Anjomshoaa, A., Brisard, F., Drescher, M., Fellows, D., Ly, A., McGough, S., Pulsipher, D., and Savva, A., Job Submission Description Language (JSDL) Specification, Version 1.0, Grid Forum Document, GFD.136, Open Grid Forum, Lemont, Illinois, U.S.A. July 2008.

[SCHEMA1] Thompson, H.S., Beech, D., Maloney, M., and Mendelsohn, N., XML Schema Part 1: Structures (Second Edition), W3C Recommendation. October 2004.

[SCHEMA2] Biron, P. and Malhotra, A., XML Schema Part 2: Datatypes (Second Edition), W3C Recommendation. October 2004.

[UR] Mach, R., Lepro-Metz, R., Jackson, S., and McGinnis, L., Usage Record Format, Grid Forum Document, GFD.98, Open Grid Forum, Lemont, Illinois, U.S.A. February 2007.

[UUID] Leach, P., Mealling, M., and Salz, R., A Universally Unique IDentifier (UUID) URN Namespace RFC 4122. July 2005.

[WSADDR] Gudgin, M., Hadley, M, and Rogers, T. (eds.), Web Services Addressing 1.0 – Core (WS-Addressing), W3C Recommendation. May 2006.

[WSAG] Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T., Pruyne, J., Rofrano, J., Tuecke, S., and Xu, M., Web Services Agreement Specification (WS-Agreement), Grid Forum Document GFD.107; The Open Grid Forum, Joliet, Illinois, United States. 2007.

[WSSEC] Nadalin, A., Kaler C., Hallam-Baker, P., and Monzillo, R. (eds.), Web Services Security: SOAP Message Security 1.1, OASIS Standard. February 2006.

[XMLDSIG] Eastlake, D., Reagle, J., Solo, D., Hirsch, F., and Roessler, T. (eds.), XML Signature Syntax and Processing (Second Edition), W3C Recommendation. June 2008.

[XPATH] Berglund, A., Boag, S., Chamberlin, D., Fernández, M.F., Kay, M., Robie, J., and Siméon, J., XML Path Language (XPath) 2.0, W3C Recommendation. January 2007.

1. The Abstract Activity Instance Description Schema

<?xml version="1.0" encoding="UTF-8"?>  
<xsd:schema attributeFormDefault="unqualified"  
elementFormDefault="qualified"  
targetNamespace="http://schemas.ogf.org/jsdl/2010/06/activity-instance-description" version="1.0"  
xmlns:aid="http://schemas.ogf.org/jsdl/2010/06/activity-instance-description"  
xmlns:wsa="http://www.w3.org/2005/08/addressing"  
xmlns:xsd="http://www.w3.org/2001/XMLSchema"   
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

<xsd:annotation>  
 <xsd:documentation xml:lang="en">  
 The OGF takes no position regarding the validity or scope of  
 any intellectual property or other rights that might be claim  
 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.  
  
 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.

Copyright (C) Open Grid Forum (2010). All Rights Reserved. This  
 document and translations of it may be copied and furnished to  
 others, and derivative works that comment on or otherwise  
 explain it or assist in its implementation may be prepared,  
 copied, published and distributed, in whole or in part, without  
 restriction of any kind, provided that the above copyright  
 notice and this paragraph are included on all such copies and  
 derivative works. However, this document itself may not be  
 modified in any way, such as by removing the copyright notice  
 or references to the OGF or other organizations, except as  
 needed for the purpose of developing Grid Recommendations in  
 which case the procedures for copyrights defined in the OGF  
 Document process must be followed, or as required to translate  
 it into languages other than English.  
  
 The limited permissions granted above are perpetual and will  
 not be revoked by the OGF or its successors or assignees.  
 </xsd:documentation>

<xsd:documentation xml:lang="en">  
 Abstract Activity Instance Description schema document  
 according to the Activity Instance Description Specification  
 Version 1.0 (GFD.X)  
  
 Authors:  
 Alexander Papaspyrou, TU Dortmund University  
 Philipp Wieder, Service Computing Group, TU Dortmund University  
 </xsd:documentation>  
</xsd:annotation>  
  
<xsd:import  
 namespace="http://www.w3.org/2005/08/addressing"  
 schemaLocation="http://www.w3.org/2005/08/addressing/ws-addr.xsd"/>   
  
<xsd:element abstract="true" name="State">  
 <xsd:annotation>  
 <xsd:documentation xml:lang="en">  
 Stores a state model instance for the given activity state. In  
 its abstract form, the state model is arbitrary. This is an  
 abstract type and has to be substituted by an appropriate  
 definition (see also GFD.X, Sections 4.6 and 5.3).  
 </xsd:documentation>  
 </xsd:annotation>  
</xsd:element>

<xsd:element abstract="true" name="Exception">  
 <xsd:annotation>  
 <xsd:documentation xml:lang="en">  
 Stores an exception model instance for the given activity  
 fault. This is an abstract type and has to be substituted by a  
 appropriate definition (see also GFD.X, Sections 4.7 and 6.1).  
 </xsd:documentation>  
 </xsd:annotation>  
</xsd:element>  
   
<xsd:complexType name="ActivityStatusType">  
 <xsd:annotation>  
 <xsd:documentation xml:lang="en">  
 Describes the status of the activity with respect to the  
 enclosing history record. The status of an activity comprises  
 its current state (defined by an appropriate state model) and,  
 if necessary, exception information. Every status record  
 for an activity MUST contain at least the current state; if an  
 exceptional condition occurs during the activity's lifetime, it  
 SHOULD be also recorded here. Note that the existence of an  
 exception entry is not necessarily coupled to a corresponding  
 exceptional state; a possible connection between these is left  
 to the implementor and SHOULD be described in the concrete  
 state model's documentation.  
 </xsd:documentation>  
 </xsd:annotation>  
 <xsd:sequence>  
 <xsd:element ref="aid:State">  
 <xsd:annotation>  
 <xsd:documentation xml:lang="en">  
 Denotes details on the state of the activity instance  
 with respect to the enclosing history record. See also  
 abstract element State and GFD.108, Section 6.6.  
 </xsd:documentation>  
 </xsd:annotation>  
 </xsd:element>  
 <xsd:element ref="aid:Exception" minOccurs="0">  
 <xsd:annotation>  
 <xsd:documentation xml:lang="en">  
 Denotes the exception raised for the activity instance  
 with respect to the enclosing record. See also abstract

element Exception.  
 </xsd:documentation>  
 </xsd:annotation>  
 </xsd:element>  
 </xsd:sequence>  
</xsd:complexType>

<xsd:element abstract="true" name="ActivityDefinition">  
 <xsd:annotation>  
 <xsd:documentation xml:lang="en">  
 Defines the requirements of an activity, for example the  
 description template used to create the activity. The initial  
 definition (template) MAY change over time due to refinement of  
 the requirements as a result of scheduling, delegation, or  
 negotiation processes, etc. Therefore, this element MAY appear  
 in more than one ActivityHistoryEntry. This is an abstract type  
 and has to be substituted by an appropriate definition (see  
 GFD.X, Sections 4.8 and 5.1).  
 </xsd:documentation>  
 </xsd:annotation>  
</xsd:element>

<xsd:element abstract="true" name="ActivityDependency">  
 <xsd:annotation>  
 <xsd:documentation xml:lang="en">  
 Describes links to associated activities within a dependency  
 structure (such as a workflow). This is an abstract type and  
 has to be substituted by an appropriate definition (see GFD.X,  
 Section 4.9).  
 </xsd:documentation>  
 </xsd:annotation>  
</xsd:element>

<xsd:element abstract="true" name="ResourceUsage">  
 <xsd:annotation>  
 <xsd:documentation xml:lang="en">  
 Describes the resource consumption/usage of an activity, e.g.,  
 the number of CPUs used or maximum memory needed for some part  
 of the activity. This element may appear multiple times for an  
 activity, depending on the monitoring policies of the system  
 generating them; the system may choose to perform averaging  
 over the execution time, averaging over several periods that  
 cover the execution time, sampling of the system, etc.  
 Concretizations of this element SHOULD describe what time

instant or time period they apply to. Because several  
 monitoring systems may be feeding usage information into the

activity instance document, the time points/periods MAY be

overlapping and MAY be non-contiguous. Note that there is no   
 requirement for the information in the activity instance  
 document to be either accurate or timely. This is an abstract  
 type and has to be substituted by an appropriate definition  
 (see GFD.X, sections 4.11 and 5.2).  
 </xsd:documentation>  
 </xsd:annotation>  
</xsd:element>

<xsd:simpleType name="ActivityHistoryEntryCategoryType">  
 <xsd:annotation>  
 <xsd:documentation xml:lang="en">  
 Describes the category of a historic entry for an activity. The  
 possible options are "initial", "intermediate", and "final".   
 </xsd:documentation>  
 </xsd:annotation>  
 <xsd:restriction base="xsd:string">  
 <xsd:enumeration value="initial">  
 <xsd:annotation>  
 <xsd:documentation xml:lang="en">  
 Denotes the initial history record for a given activity,  
 which MUST be the first one created in the whole record.  
 </xsd:documentation>  
 </xsd:annotation>  
 </xsd:enumeration>  
 <xsd:enumeration value="intermediate">  
 <xsd:annotation>  
 <xsd:documentation xml:lang="en">  
 Denotes an intermediate history record for a given  
 activity. Such entry MUST NOT be the first one  
 created in the whole record.  
 </xsd:documentation>  
 </xsd:annotation>  
 </xsd:enumeration>  
 <xsd:enumeration value="final">  
 <xsd:annotation>  
 <xsd:documentation xml:lang="en">  
 Denotes the final history record for a given activity.  
 Note that this does not imply that the activity on its  
 </xsd:documentation>  
 </xsd:annotation>  
 </xsd:enumeration>  
 </xsd:restriction>  
</xsd:simpleType>

<xsd:complexType name="ActivityHistoryEntryType">  
 <xsd:annotation>  
 <xsd:documentation xml:lang="en">

Denotes an event in the history of an activity, containing its  
 properties at the time the event occurred. Every entry MUST  
 contain at least a timestamp (as attribute), the status of the  
 activity at this timestamp, and a WS-Adressing endpoint  
 reference to the managing service.  
 </xsd:documentation>  
 </xsd:annotation>  
 <xsd:sequence>  
 <xsd:any namespace="##other" processContents="lax" minOccurs="0"  
 maxOccurs="unbounded">  
 <xsd:annotation>  
 <xsd:documentation xml:lang="en">  
 Provides an extension point for additional elements in  
 the ActivityDocumentType. Implementations SHOULD ignore  
 unsupported extensions.  
 </xsd:documentation>  
 </xsd:annotation>  
 </xsd:any>  
 <xsd:element name="Status" type="aid:ActivityStatusType">  
 <xsd:annotation>  
 <xsd:documentation xml:lang="en">  
 Stores the status of the activity instance at the time  
 described by the surrounding history entry. The status   
 of an activity comprises its current state   
 (defined by an appropriate state model) and, if  
 necessary, exception information. Every status record for

an activity MUST contain at least the current state; if

an exceptional condition occurs during the activity's

lifetime, it SHOULD be also recorded here. Note that the

existence of an exception entry is not necessarily

coupled to a corresponding exceptional state; a possible  
 connection between these is left to the implementor and  
 SHOULD be described in the concrete state model's  
 documentation. See also ActivityStatusType and GFD.X,

Section 4.5.  
 </xsd:documentation>  
 </xsd:annotation>  
 </xsd:element>  
 <xsd:element ref="aid:ActivityDefinition" minOccurs="0">  
 <xsd:annotation>  
 <xsd:documentation xml:lang="en">  
 Stores the definition of the activity with respect to  
 the enclosing history record. See also  
 ActivityDefinition and GFD.X, Section 4.8.  
 </xsd:documentation>  
 </xsd:annotation>  
 </xsd:element>  
 <xsd:element ref="aid:ActivityDependency" minOccurs="0">  
 <xsd:annotation>  
 <xsd:documentation xml:lang="en">  
 Stores the dependency definitions for the activity with  
 respect to the enclosing history record. See also  
 ActivityDependency and GFD.X, Section 4.9.  
 </xsd:documentation>  
 </xsd:annotation>  
 </xsd:element>  
 <xsd:element name="ManagerReference"  
 type="wsa:EndpointReferenceType">  
 <xsd:annotation>  
 <xsd:documentation xml:lang="en">  
 Keeps the endpoint reference of the activity's managing  
 service at the time denoted by the enclosing record. The  
 corresponding service SHOULD expose an interface for  
 managing the activity's state, lifecycle, and execution.   
 See also GFD.X, Section 4.10.  
 </xsd:documentation>  
 </xsd:annotation>  
 </xsd:element>  
 <xsd:element ref="aid:ResourceUsage" minOccurs="0"  
 maxOccurs="unbounded">  
 <xsd:annotation>  
 <xsd:documentation xml:lang="en">  
 Stores the resource usage for this activity with respect  
 to the enclosing history record. See also ResourceUsage  
 and GFD.X, Section 4.11.  
 </xsd:documentation>  
 </xsd:annotation>  
 </xsd:element>  
 </xsd:sequence>  
 <xsd:attribute name="timestamp" type="xsd:dateTime"  
 use="required">  
 <xsd:annotation>  
 <xsd:documentation xml:lang="en">  
 Keeps the timestamp at which this event has occurred in  
 the activity's history. The entries in the whole activity  
 history SHOULD be ordered ascending to their timestamp.   
 See GFD.X, Section 4.4.4,  
 </xsd:documentation>  
 </xsd:annotation>  
 </xsd:attribute>  
 <xsd:attribute name="category"  
 type="aid:ActivityHistoryEntryCategoryType" use="optional">  
 <xsd:annotation>  
 <xsd:documentation xml:lang="en">  
 Denotes the category of this history record. See  
 ActivityHistoryEntryCategoryType and GFD.X, Section  
 4.4.4, for possible values.  
 </xsd:documentation>  
 </xsd:annotation>  
 </xsd:attribute>  
 <xsd:anyAttribute namespace="##other" processContents="lax"/>  
</xsd:complexType>

<xsd:complexType name="ActivityHistoryType">  
 <xsd:annotation>  
 <xsd:documentation xml:lang="en">  
 Keeps track of the record of the activity's history. This  
 record MUST contain one initial record and, at most, one final  
 record see also ActivityHistoryEntryCategoryType). Note that,  
 although a final record MAY have been written already, the  
 activity document MAY still be modified.  
 </xsd:documentation>  
 </xsd:annotation>  
 <xsd:sequence>  
 <xsd:element name="ActivityHistoryEntry"  
 type="aid:ActivityHistoryEntryType" maxOccurs="unbounded">  
 <xsd:annotation>  
 <xsd:documentation xml:lang="en">  
 Stores a single event in an activity's history. It  
 denotes an event in the history of an activity,  
 containing its properties at the time the event

occurred. Every entry MUST contain at least a   
 timestamp (as attribute), the status of   
 the activity at this timestamp, and a WS-Addressing  
 [WSADDR] endpoint reference to the managing service.  
 Once an ActivityHistoryEntry is written, it MUST NOT  
 be altered. Additional information about the  
 respective activity has to be appended to the  
 ActivityHistory by adding a new ActivityHistoryEntry  
 element. See also ActivityHistoryEntryType and GFD.X,  
 Section 4.4.  
 </xsd:documentation>  
 </xsd:annotation>  
 </xsd:element>  
 </xsd:sequence>  
</xsd:complexType>  
  
<xsd:complexType name="ActivityDocumentType">  
 <xsd:annotation>  
 <xsd:documentation xml:lang="en">  
 Contains an activity's meta-data and history and provides a

root element for every activity. While the meta-data part MAY  
 carry information about the activity's creator, purpose, and  
 references (i.e. to other activities), the history part SHOULD  
 describe the full lifecycle of the activity.  
 </xsd:documentation>  
 </xsd:annotation>   
 <xsd:sequence>   
 <xsd:element name="ActivityDescription" type="xsd:string"  
 minOccurs="0">  
 <xsd:annotation>  
 <xsd:documentation xml:lang="en">  
 Contains a natural-language description of the activity  
 and offers means for storing additional information on  
 the activity for displaying purposes (e.g. in a user  
 interface). See also ActivityDescriptionType and GFD.X,  
 Section 4.2.  
 </xsd:documentation>  
 </xsd:annotation>  
 </xsd:element>  
 <xsd:element name="ActivityHistory"  
 type="aid:ActivityHistoryType">  
 <xsd:annotation>  
 <xsd:documentation xml:lang="en">  
 Stores the history of an activity. See also  
 ActivityHistoryType and GFD.X, Section 4.3.  
 </xsd:documentation>  
 </xsd:annotation>  
 </xsd:element>  
 <xsd:any namespace="##other" processContents="lax" minOccurs="0"  
 maxOccurs="unbounded">  
 <xsd:annotation>  
 <xsd:documentation xml:lang="en">  
 Provides an extension point for additional root elements  
 in the activity document. Implementations SHOULD ignore  
 unsupported extensions.  
 </xsd:documentation>  
 </xsd:annotation>  
 </xsd:any>  
 </xsd:sequence>  
 <xsd:attribute name="id" type="xsd:string">  
 <xsd:annotation>  
 <xsd:documentation xml:lang="en">  
 An identifier for the activity, which MUST be globally

unique. It is RECOMMENDED to use Universally Unique

Identifiers as described in RFC 4122, "A Universally Unique  
 IDentifier (UUID) URN Namespace". See also GFD.X, Section  
 4.1.4.  
 </xsd:documentation>  
 </xsd:annotation>  
 </xsd:attribute>  
 <xsd:anyAttribute namespace="##other" processContents="lax">  
 <xsd:annotation>  
 <xsd:documentation xml:lang="en">  
 Provides an extension point for additional root attributes  
 in the activity document. Implementations SHOULD ignore  
 unsupported extensions.  
 </xsd:documentation>  
 </xsd:annotation>  
 </xsd:anyAttribute>  
</xsd:complexType>

<xsd:element name="ActivityDocument" type="aid:ActivityDocumentType">  
 <xsd:annotation>  
 <xsd:documentation xml:lang="en">  
 The document root of a single activity instancewhich contains   
 an activity's meta-data and history and provides the entry   
 point for every activity. While the meta-data part MAY carry   
 information about the activity's creator, purpose, and   
 references (i.e. to other activities), the history part   
 SHOULD describe the full lifecycle of the activity. See also  
 ActivityDocumentType and GFD.X, Section 4.1.  
 </xsd:documentation>  
 </xsd:annotation>

</xsd:element>

</xsd:schema

1. Open Grid-Forum-related Activity Instance Schema
2. Activity Instance Description Examples

<ActivityDocument id=”ea196512-9cb7-4a14-91b0-2dde749a5f7d”>

<ActivityDescription>

This activity instance has been generated due to an activity request  
 submitted to the scheduling service with the  
 following URI: http://tempuri.org/services/activityscheduler. The  
 activity request has been received at 2010-05-10T11:11:11.11. The

activity instance has been created 2010-05-10T11:11:44.44 by the   
 organization’s activity store with the following URI:

http://tempuri.org/services/activitystore.

</ActivityDescription>

<ActivityHistory>

<ActivityHistoryEntry

timestamp=”2010-05-10T11:11:44.44” category=”initial”>

<Status> ... </Status>

<ActivityDefinition> ... </ActivityDefinition>

<ActivityDependency> ... </ActivityDependency>

<ManagerReference> ... </ManagerReference>

<ResourceUsage> ... </ResourceUsage>

</ActivityHistoryEntry>

</ActivityHistory>

</ActivityDocument>

1. Please note that the activity request can be submitted in any supported format, for example JSDL. In general, the client does not have to be aware of the concept of an activity instance as specified by this document at all. To this end, the term activity is used in two different ways: the unit of work submitted by the client (an arbitrary unit of work or activity) and the activity instance compliant with the activity instance description specified within this document [↑](#footnote-ref-1)