Application Contents Service specification (proposed draft)

Status of This Memo

This is a sample version of the expected Application Contents Service (ACS) Specification and is created for the ACS Charter Discussion BOF at GGF12 in order to demonstrate the scope and outline of the spec. Our intention is to refine this document to be the final specification. As such, some details of the document remain yet to be discussed and clarified to reflect broader range of the comments and requirements.

Abstract

In order to install and operate complex systems such as three-tier systems more efficiently and automatically, it is necessary to specify and manage as a unit a lot of different application-related information, which we call Application Contents. Application Contents may include deployment contents such as program binaries, configuration data, and descriptions of the hardware resources typically required for application execution, and also various elements such as descriptions of procedures for lifecycle management and management policies typically applied to the running system.

In this document, we define Application Contents Service (ACS) to manage Application Contents. ACS is an OGSA service, which maintains Application Contents in a repository and provides functions to access them, retrieve their change histories and so on. We also define a standard format to archive Application Contents associated with a single application for registering Application Contents in an ACS and exchanging them between ACSs.

Contents

1. Introduction

OGSA-compliant Grid systems provide a rich set of functionalities such as dynamic resource allocation, provisioning, service level management, and automation of administrative tasks [OGSA doc]. In order to utilize these functionalities in executing applications on Grid systems, applications require certain preparation. That is, developers of Grid applications must develop various components including, but not limited to, deployment procedures, requirements on resources, and self-management policies, which are all handled by OGSA services, as well as program binaries, initial data, and configuration information, which are commonly necessary for application execution.

In this document, we use the term a Grid Application, or simply an application, to refer an application executable on Grid systems. And we use the term Application Contents for a set of information which collectively makes the application deployable, runnable on Grid systems and optionally help system do provisioning autonomously. The Application Contents include application binaries and related information; e.g. program binaries, configuration data, procedure descriptions for lifecycle management, requirements descriptions for the hardware and underlying middleware, policy rules, and anything needed to create a job instance on grid systems. They may be real entities or location pointers. On the other hand, the Application Contents don't include information updated by a job instance and information describing a status of a job instance. In addition, they don't include operating system and/or middleware binaries, since they are considered as a part of the resource provided by infrastructure. In any way, ACS doesn't interpret or execute information in the contents, rather just manage them for use by other OGSA-services.

More complicated application, such as three-tier system, makes it difficult to describe information handled by Grid systems for the purpose of deployment and execution management, and requires many files to be handled. Consistent management of Application Contents throughout the lifetime of application, including its version-upgrade, is difficult and prone to human errors.

Scenarios for resource sharing across sites and disaster recovery may be realized through the distributing a set of jobs among the separate grid systems if they are so designed as to be runnable in the distributed environment. For those jobs, automated exchange of Application Contents between Grid systems will facilitate effective deployment of the job. This doesn't necessarily include dynamic status for a job instance. There already exist many products for automated deployment of application on, and change management of, multiple hosts. However, those products don't target running in dynamic and heterogeneous environments, lacking in considerations about resource sharing across multiple administrative domains as one of the goals of OGSA. Some products, such as J2EE, utilize mechanisms for specific platforms, and some products take proprietary and/or nonpublic approaches. These products don't seem to enable exchange of Application Contents between heterogeneous Grid systems.

In this document, we define Application Contents Service (ACS) to address these problems. ACS is an OGSA service, which maintains Application Contents in a repository and provides functions to access them, retrieve their change histories and so on. We also define a standard format to archive Application Contents associated with a single application for registering Application Contents in an ACS and exchanging them between ACSs.

Since Application Contents are connected to each other on an application basis within a single archive file outside Grid systems or a logical unit inside Grid systems, ACS simplifies registration and updating processes of Application Contents and enables integrated management to maintain their consistency. The standard format to archive Application Contents also enables automatic administration of Grid Application including selection of Grid system to submit to. Moreover, by

enabling history reference and restoration of Application Contents, ACS can provide with assistance for troubleshooting of problems arising from version inconsistency.

In the rest of this chapter, we explain goals and non-goals of ACS. In §2, we analyze and define the set of requirements for ACS. Based on the requirements, we present the ACS concept and architecture in §3 and some example usage scenarios to shape the ACS image in §4. Then we define ACS specification that consists of Application Contents Service interface to handle Application Contents repository in §5 and Application Archive Format (AAF) to archive Application Contents in §6. We consider relationship with other existing technologies in §7 and security in §8. We are going to describe a sample ACS-compliant application in §9 and normative xsd and wsdl schema in §10 in a later version.

This specification is designed utilizing the experience in the Business Grid Project [BizGrid] in Japan, which makes reference with OGSA architecture. The details are to be determined at GGF ACS-WG in the future. There remain TBD items in this document and we collected up open issues in §11.

Most of the terms in this document are included in OGSA documents [OGSA doc][OGSA glossary][OGSA usecase]. See glossary section at the bottom of this document for some terms specific to this document.

1.1 Goals

This document is intended to:

- define the standard way to manage and handle Application Contents as a logical unit, so as to maintain their consistency, reduce management overheads, and enable automation throughout the lifetime of the application.
- define the standard format to archive Application Contents so as to simplify registration and updating processes and automate administration of application including selection of system to submit to,
- and standardize the common operations on Application Contents and their input/output format so as to cover the differences between heterogeneous systems and enable interoperating in the future.
-
-

1.2 Non-goals

ACS will define the specification necessary for reposition and management of Application Contents, but doesn't interpret or execute Application Contents themselves. That is, ACS handles Application Contents as opaque data. Information necessary for deployment and execution management of application need to be defined by OGSA services that process the information.

2. Requirements

In this chapter, we define a set of requirements that ACS is intended to address. The analysis is based on the demand of manipulating Application Contents inside/outside of the Gird system.

2.1 Functional Requirements

2.1.1 A consistent set of the Application Contents in a bundle

Business Activity Managers have to specify various kinds of configuration information to take advantage of the intelligent capabilities OGSA provides. The more complicated an application becomes, the larger number of Application Contents need to be handled associated with the application. It is required to manage a consistent set of Application Contents in a bundle to avoid troubles due to missing contents or version inconsistency. The dependency description here needs to be prepared by the Business Activity Managers and needs to be complete and static, leaving no system dependent factors for the interpretation of the information.

These troubles are likely to be caused when registering and updating applications. Though Application Contents are deployed on diverse and distributed resources and services, they should be submitted to Grid system through a single entry point in a unified manner. Change management and versioning are also desired to realize such functionalities as backup and rollback in case of errors while updating the application.

Archiving Application Contents into a single file makes it easier for both humans and systems to handle Application Contents and maintain the consistency. Various kinds of package formats targeting single- or multi- platforms are proposed and widely accepted today. In Grid, similarly, it will surely be effective to standardize an archive format which targets OGSA-compliant Grid systems and which aggregates a consistent set of files related to a single application.

2.1.2 Exchangeability of the Application Contents

One major purpose of OGSA is sharing and utilizing resources owned and controlled by various organizations. Since Grid environment on which an application is executed may be heterogeneous and distributed across administrative domains, Application Contents need to be exchangeable between various implementations of the Grid systems. To realize exchangeability of Application Contents, the standardized manner of describing and exchanging them are required. The syntax and semantic definitions of each Application Content are being discussed in CDDLM-WG and to be discussed aftertime in some WGs (but not in ACS-WG) as substantiation of OGSA is proceeded. The definition of mechanisms for exchanging Application Contents between different Grid systems is within ACS scope and addressed in this document.

2.1.3 Reuse of the previous Application Contents

Considering situations such as repeating the same calculation with different parameters or performing similar services in parallel, Business Activity Managers may feel convenient if the Grid system allows them to create a new job reusing a running job or its subset. Grid system implementers, up to their ingenuity, may be able to develop more efficient systems that can save disk space if Application Contents are reused. The reuse of the previous Application Contents expands possibilities of Grid system in such areas as usability and implementation flexibility. In order to realize the reusability, the management of Application Contents must be independent of creation and execution management of jobs.

In case of creating similar jobs repeatedly, ACS should enable to create a new job reusing the Application Contents that are previously registered for the similar job. Business Activity Managers may desire to apply some updates on the registered Application Contents to create a different sort of job. Some job-specific parameters as well as Application Contents may be used to create jobs, but they are beyond ACS's scope. ACS provides the repository capability used by Job Manager, which is an OGSA service to control execution of jobs, to realize the reusability related to job creation.

The ACS doesn't specify how the above can be implemented, to reach higher efficiency; rather it merely specifies the common interface or harness of the Application Archive to enable this happen.

2.2 Non-functional requirements

2.2.1 Efficiency

The amount of each content and/or the whole size of an application may become extremely large depending on its property. Efficiency is one of the major keys in the standardization of manipulating Application Contents.

For example, efficiency is required when registering Application Contents to Grid systems or exchanging them within/across Grid systems. In this case, ACS interface is expected to support fragmentation regardless of its logical structure or to be flexible so that clients can select the best transport method supported by the implementation. In order to moderate the total size of Application Contents archive file, it may be useful to contain references to external entities as well as real entities in the Application Contents archive. However, in this case, consistency of the Application Contents including the referenced entities needs to be carefully maintained. When standardizing the method of managing change histories of Application Contents in a repository, it is expected to allow the implementers to contrive ways to eliminate redundancy and save disk space in the repository.

2.2.2 Extensibility

The standard specifications should define the minimum set of rules necessary for interoperability, and allow flexibility which enables the implementation to have its own extension. For example, the standard format to archive Application Contents should define the common required elements necessary for creating jobs as well as the optional elements to contain implementation-specific information.

2.2.3 Ease of use

It is one of the main goals of OGSA to reduce the administrative cost of application. Application Contents can be aggregated into a single file and submitted directly into Grid systems, which makes the application easy-to-use for Business Activity Managers. If application files are not aggregated so, Business Activity Managers would require technical knowledge as to where he should deploy each file.

2.2.4 Interaction with OGSA services

ACS is a building block to realize OGSA capabilities and will be designed in accordance with OGSA architecture. Therefore, ACS will be based on OGSA infrastructure service (XML, WSDL, WS-Addressing, WS-ResourceFramework, WS-Notification, etc...) and be designed to coordinate with, but avoid overlapping with, other OGSA services.

2.3 Security Consideration

Application Contents possibly include critical information such as business logics or privacy data. In the standardization of ACS, we must adequately take into account security issues. AAF, the standard archive format, needs a mechanism for preventing alteration. ACS, the OGSA service, needs the functionalities such as client authentication, access control, and so on. The mechanism of these functionalities should go with the OGSA security policies.

3. Architecture

3.1 Definition of Application Archive

As described so far, it is required to aggregate multiple Application Contents associated with an application and manage them as a unit. The conceptual unit is called Application Archive (AA) in this document.

AA includes the following components:

- AA Descriptor

  AA Descriptor is an XML document describing the AA's meta-information such as its identifier, properties and contents list. An AA always includes a single AA Descriptor.

- Signature

  Digital signature for detecting alteration of AA.

- Application Contents

  A various kinds of information that is used to create and operate a job by Job Manager or other OGSA services. Application Contents include, but not limited to, executable files (binaries or scripts), initial data, configuration information, procedures for installation, and administration policies.
  Application Contents allow entity and/or reference to external data storage area. The latter is useful in specifying logical relationships consistently about huge data such as user data.

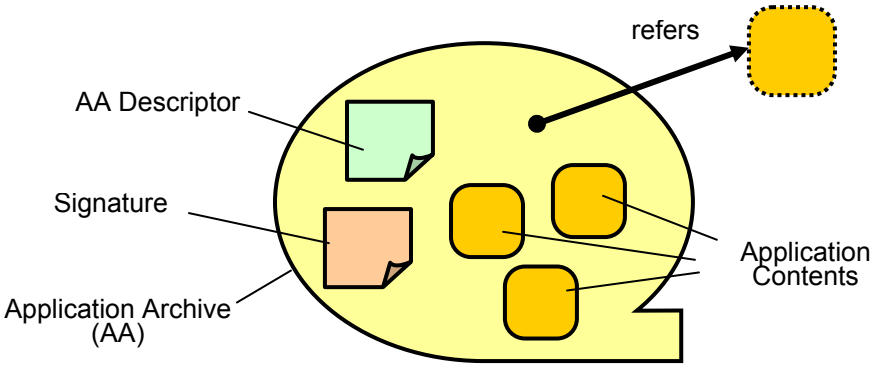

Figure 1: Structure of Application Archive

Outside Grid systems, AA is archived into a single file. Inside Grid systems, it is stored in a repository managed by ACS. How AA is physically stored in the repository depends on ACS implementations.

3.2 Identification and Reference of Application Archive

In an AA Descriptor, an AAID is specified for identifying the AA. AAID consists of name and version of the AA. AAID must be unique in the ACS which the AA is registered into.

Inside the Grid system, AA is handled as a WS-Resource. ACS which accepted registration of an AA publishes an endpoint reference to the AA and returns it to the ACS client. From then on, the ACS client can access the ACS (AA) by means of the endpoint reference. wsa:address element of the endpoint reference contains the address of the ACS keeping the AA, and wsa:ReferenceProperties element contains AAID.

ACS may optionally provide transfer of the archive between ACSs, to add more flexibility to the system for application deployment. When transferring an AA, acs:Originator element containing the address of the ACS from which the AA is transferred is added into wsa:ReferenceProperties element, which prevents duplication of identity.

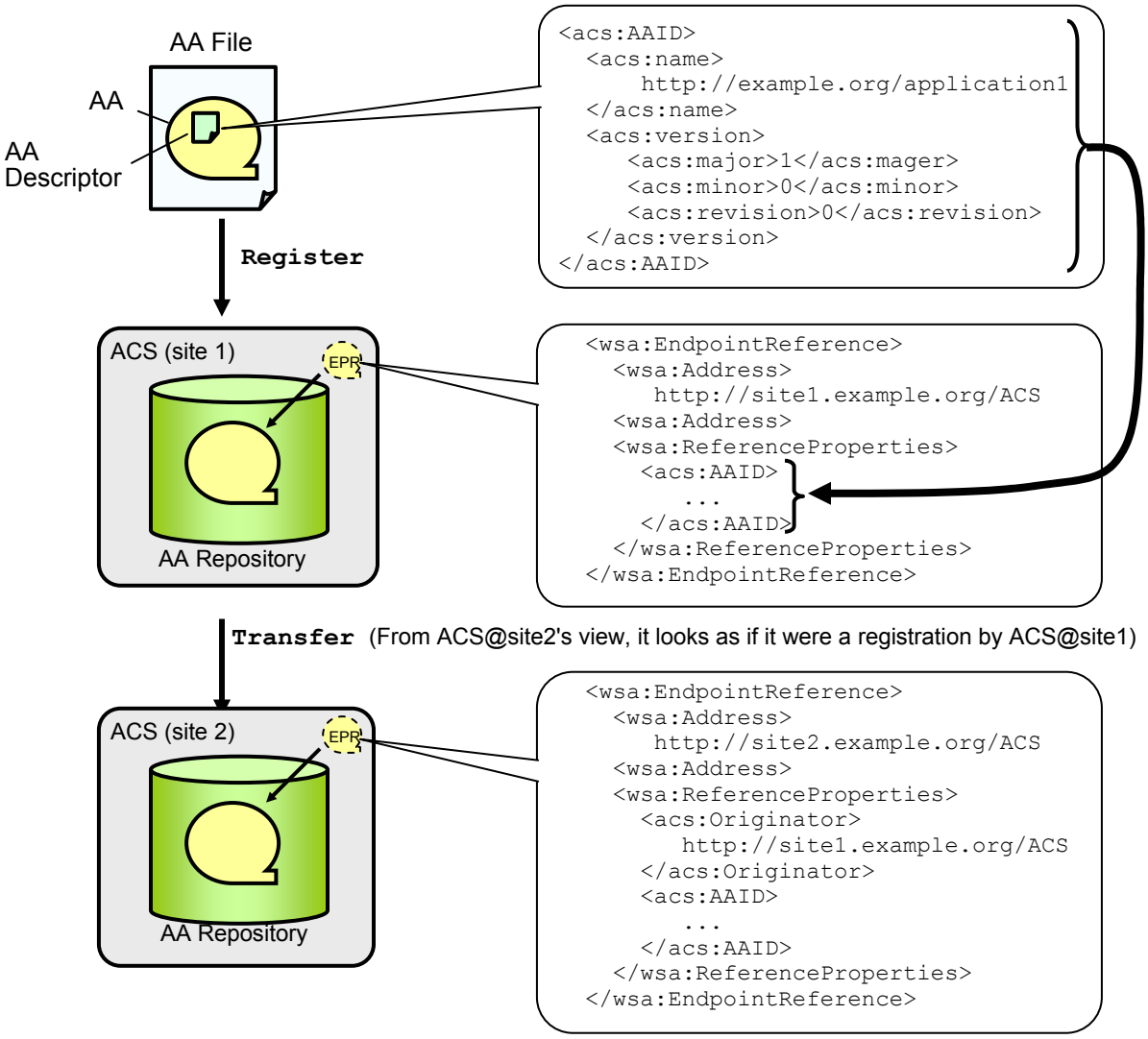

Figure 2: Identification of Application Archive

3.3 Job execution management and ACS

ACS is utilized by OGSA execution management services such as Job Manager and Deployment Service. This section examines how job execution management utilizes ACS.

Figure 3 illustrates the conceptual relationship between AA and job. Job keeps the endpoint reference to AA, that keeps up the relationship between the job and the AA. Multiple jobs may refer to the same AA. Job is considered to keep, except information contained in AA, its running state, log of operation by Business Activity Manager on the job, and so on, but such information is not managed by ACS.

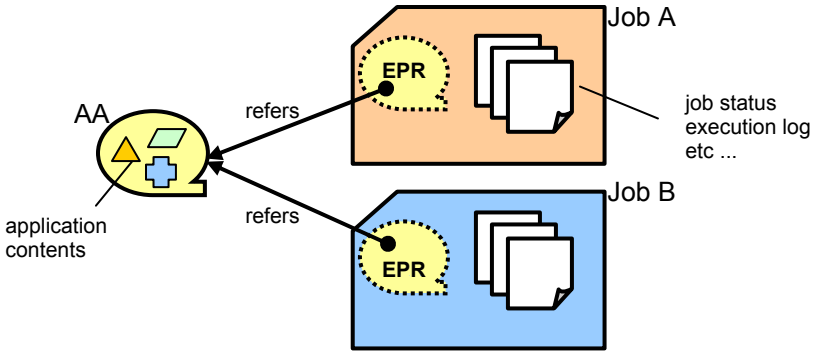

Figure 3: Conceptual relationship between AA and job

A Business Activity Manager sends a Register message to an ACS to register an AA into the ACS, and then the ACS creates an endpoint reference to the AA and returns it to the Business Activity Manager.
The Business Activity Manager sends the endpoint reference to a Job Manager to request a job creation. OGSA services get the endpoint reference from the Job Manager if necessary, and send a GetContent message to the ACS to get the Application Contents. OGSA services may inquire ACS only when they the information at the first time.
Figure 4 illustrates the relationship between services relevant to creation and deployment of job.

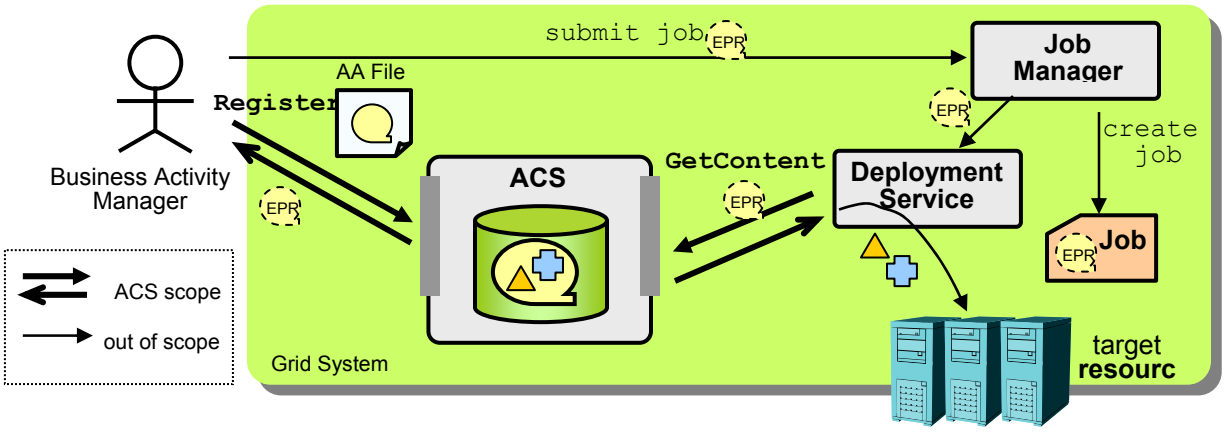

Figure 4: Relationship between ACS and job execution management

3.4 Change management of application archive

In order to update Application Contents in an AA, application developer creates an updating AA and Business Activity Manager registers it into ACS. The version from which the updating AA is updated (this is called the base version) is specified in the AA Descriptor of the updating AA. Unchanged contents may or may not be contained in the updating AA to minimize the size of it. AA containing only the difference from the base version is called a delta AA, and AA containing all the contents is called a full AA. In case of delta AA, ACS internally complements the Application Contents by means of those in the base AA.

In Figure 5, the delta AA contains the added content C and the updated content A only. When the delta AA is registered, AA repository adds the unchanged content B to it to complement the Application Contents. Note that the base AA must be registered before that.

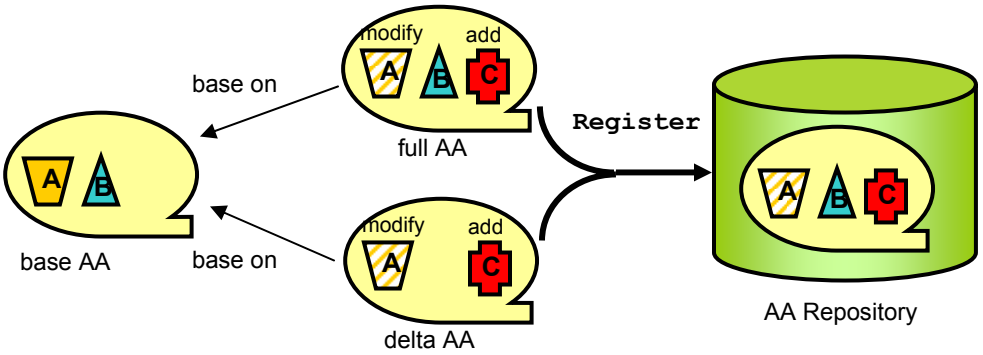

Figure 5: Delta AA and Full AA

ACS accumulates change history of AA. ACS generates a different endpoint reference for a different version of AA to identify it.

As is the case with job creation, registering a new version of AA and applying it into the running job are mutually-independent operations. In case of applying version updating into the running job, the running job needs to change the endpoint reference it has, but that process is out of ACS scope.

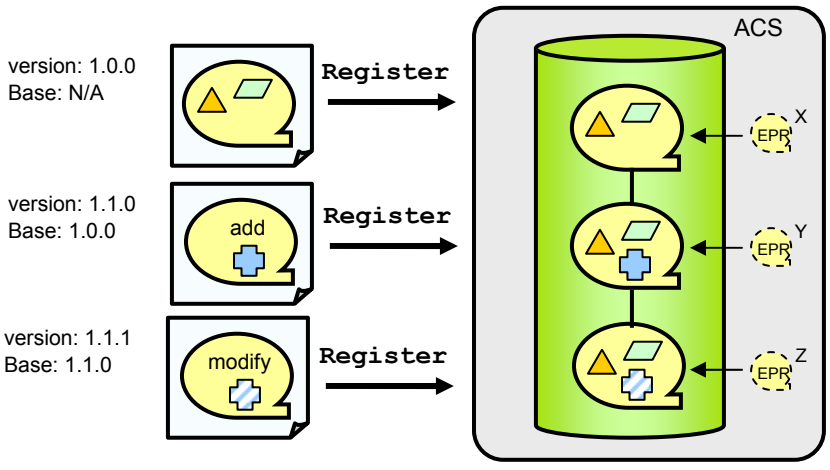

Figure 6: Version management of AA

3.5 Cooperation between multiple ACSs

In such scenarios as resource sharing across administrative domains and disaster recovery, applications spread out across multiple Grid systems. Job Manager can retrieve contents from ACS and then register it to the other ACS of another Grid system in response to Job Manager's requests. Alternatively, ACS may provide interface for the transfer to respond the Job Manager's requests. It is desirable anyway to make the transport detail transparent in interface, to allow implementation to be flexible on transport mechanism.

Figure 7 illustrates the cooperation model in latter case between multiple ACSs.

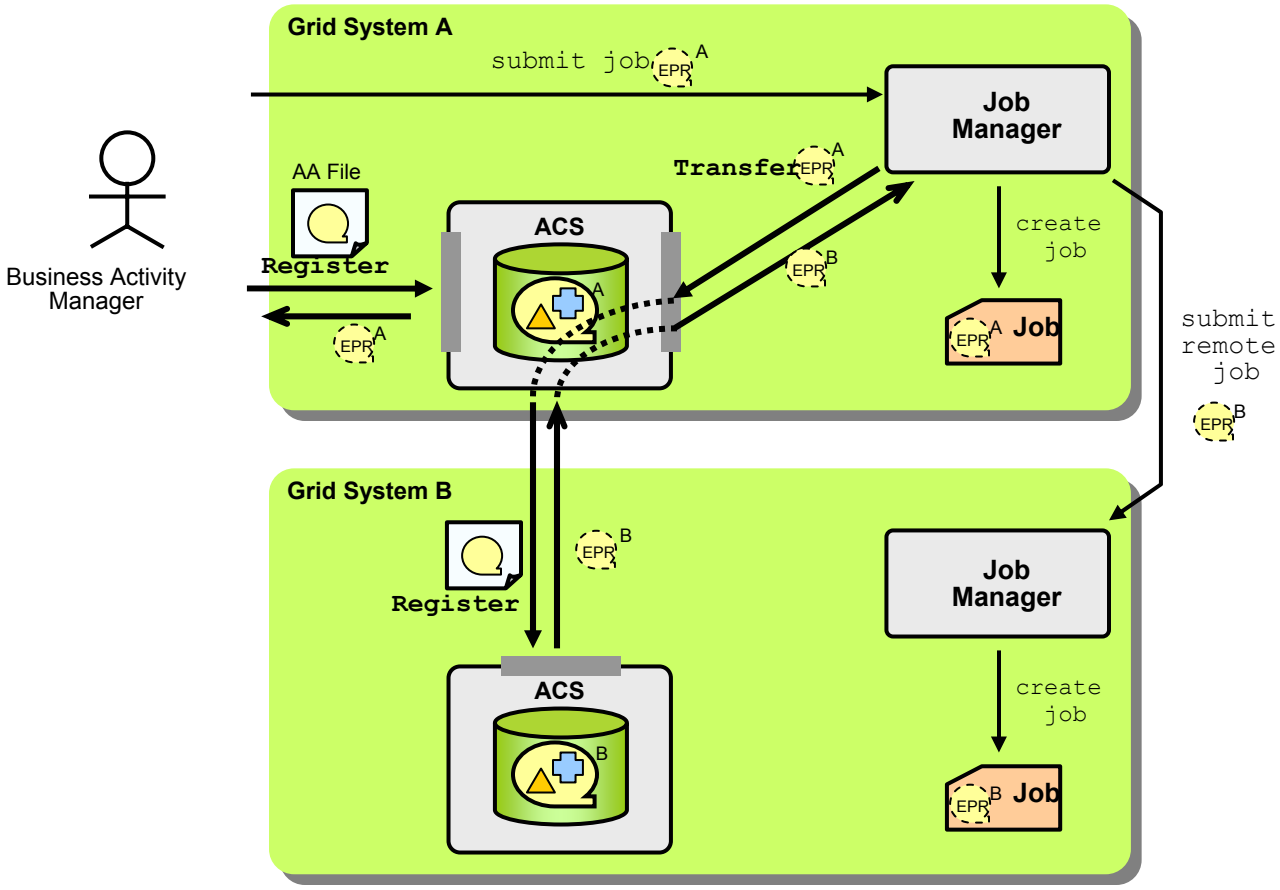

Figure 7: Cooperation model between multiple ACSs

3.6 Notification

ACS can produce event notifications on modification in AA repositories (initial registration, updating registration, and removal of AA) if necessary. Production and distribution mechanisms for event notification will be based on the Subscribe/Publish mechanism of WS-Notification. That is, such ACS inherits Producer interface defined by WS-Notification. This specification defines the event types (topics) supported by ACS, and structure of messages sent in case of events.

4. Example Usage

This chapter examines how AA and ACS can be specifically utilized throughout the application lifecycle. The following scenarios are merely examples, and don't prescribe the usage of ACS.

4.1 Development and submission of Grid Applications

It is one of main motives to separate the role of Application Developer clearly from that of Business Activity Manager. In the existing systems, application deployment is often a very complicated task, and requires understanding of directory and file structure of the application and knowledge about OSs and middleware on which the application is deployed. With ACS, Business Activity Manager only has to obtain a single AA file from Application Developers and submit it directly into Grid system. He/She doesn't require deep technical knowledge about the application nor execution environment. Grid system extracts and utilizes the information contained in AA such as deployment procedures and self-management policies, to allocate resources, deploy program binaries and initial data, configure appropriately, and start a job automatically. When an error occurs or the workload increases, Grid system will perform activities according to the policies in AA. For example Grid system may notify the warnings to the Business Activity Manager or may allocate substantial or additional resources.

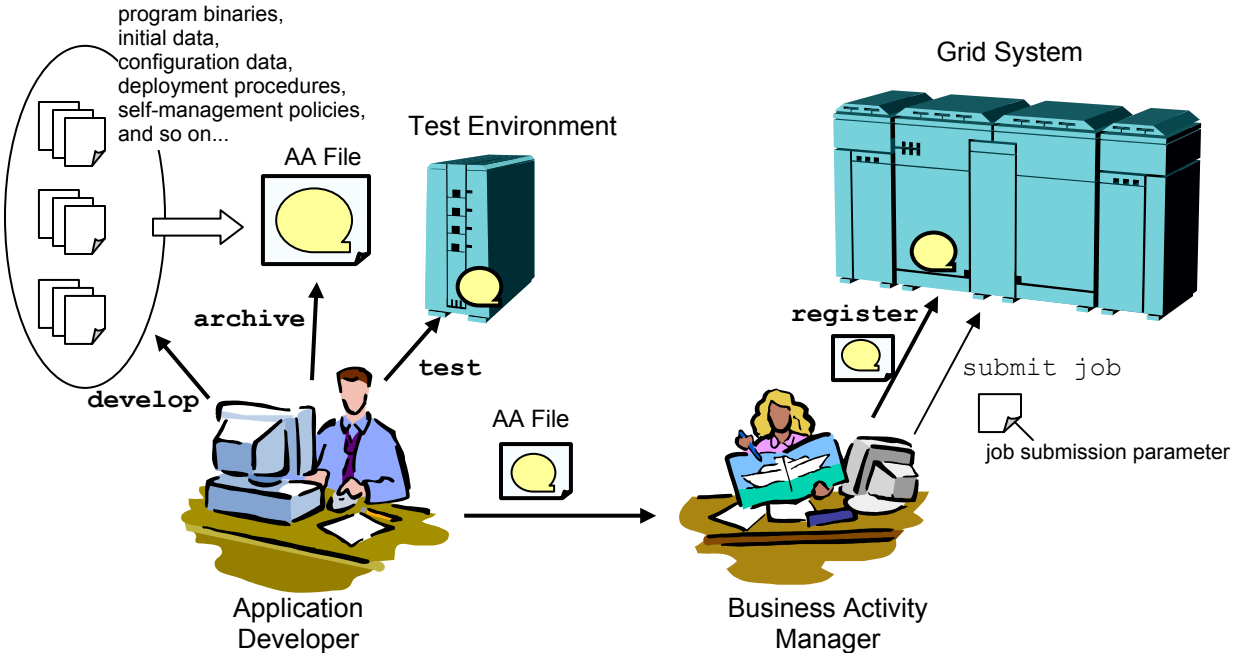

Figure 8: Application Developer and Business Activity Manager

Figure 8 illustrates the role of Application Developer and Business Activity Manager. Application Developer, according to Business Activity Manager's demand, develops Application Contents such as program binaries and initial data implementing business logics, configuration information, structure definition of application, deployment procedures, and self-management policies for attaining performance requirement. He/She assembles all the required files and archives them into a single according to AAF specification.

For example, in case of a web application consisting of web servers and a DBMS server, Application Contents such as WAR files, SQL for creating database, and initial data are deployed on execution environment. Procedures for deployment and configuration may contain steps to deploy WAR files on web servers, configure all servers, create database tables, and load initial data. Self-management policies may rule that another web server is to be added if the workload of web server exceeds a certain threshold.

Business Activity Manager logs into a Grid system and uploads the AA file to it using user interface provided by the system. He/she specifies the uploaded AA and job submission parameters such as reservation time to reserve a job. Some Grid system implementations may provide user interface which enables upload of AA file and job reservation in a single operation.

It is desirable to test AA enough before executing jobs. ACS can verify the format consistency of AA by checking the correspondence of the AA descriptor to the files in AA, but cannot detect erroneous omissions on the AA descriptor or errors in Application Contents. Application developer can test behavior of each application content by registering and executing the AA in a testing environment, or execute the AA beforehand with a testing configuration in the operational environment.

4.2 Updating of Grid Applications

The cases of making a change on running jobs include, for example, update of program binary (patch or function addition), update of self-management policies and so on. Business Activity Manager asks for Application Developer to develop an updating AA file. The manager uploads the updating AA file to the Grid system and applies the updating version to running jobs at an arbitrary time, similar to job creation.

In case of updating Application Contents to be deployed on execution environment, such as program binaries, the procedure for reflecting the change on executing environment must be described. There may be various steps, such as replacement of updated files, deletion of unnecessary files, SQL execution for changing structure of database tables. Application updating may require change of the job execution state during the updating process, such as job stop and screen display for maintenance. It depends on description format of deployment procedure whether updating process can be automated including change of the job execution state, or Business Activity Manager needs to change explicitly the job execution state.

4.3 Deletion of Grid Applications

In the case where Business Activity Manager registered an AA by mistake, he or she can specify the AA to delete it from ACS. It is possible to specify AAs of old versions or for testing which are no longer used in order to increase the free space of AA repository. It is desired to delete all the AAs collectively belonging to the same application when the jobs have been completed.

4.4 Disaster Recovery

In the case where it becomes impossible to continue services in the main system because of disaster, it is demanded to move quickly into the alternative system and resume the services. The level at which the resumption is desired in case of disaster depends on the importance of the services, but generally, it is required to start up the alternative system in a short time, minimize the loss of managed data, and minimize the resources prepared for the alternative system in regular operation.

ACS can store Application Contents in an archived state into the alternative system, which reduces the resources required for the alternative system. Application Contents are automatically deployed on execution environment according to the specified procedure, which shortens the time required for starting up the alternative system. However, ACS handles only the static information contained in AA, and thus accumulated or altered data during operation must be transferred into the alternative system by means of separate replication systems.

In order to perform job management responding to disaster recovery, Business Activity Manager directs Grid system to transfer Application Contents into the alternative system. The alternative data center may be specified explicitly by Business Activity Manager, or may be discovered automatically by the Grid system. Job Manager transfers AAs referred to by the job into the alternative data center after the job is created or application updating is applied to the job.XXX

Job Manager, not ACS, must transfer management information of jobs into the alternative data center.

In case of disaster, Business Activity Manager directs the alternative data center to start jobs.

5. Application Contents Service interface specification

5.1 ApplicationRepository portType

A web service which implements ApplicationRepository portType accepts a Register request and creates a WS-Resource called ApplicationArchive.

5.1.1 Register

Service requestor sends the following message to the ApplicationRepository.

```
<Register>

  TBD

</Register>
```

[TBD: Details of Register request message.]

The ApplicationRepository verifies the format consistency of the accepted AA, checks there is no duplication of AAID, and stores it into the repository if there are no problems. When The ApplicationRepository completed the AA registration successfully, it generates the ApplicationArchive WS-Resource for the AA and returns its endpoint reference as following.

```
<RegisterResponse>
  <AAReference>
    <wsa:Address>
       http://www.example.org/ACS
    </wsa:Address>
    <wsa:ReferenceProperties>
      <AAID>
          ...
      </AAID>
    </wsa:ReferenceProperties>
  </AAReference>
</RegisterResponse>
```

In the following cases, the ApplicationRepository fails the AA registration and returns a error.

- Format error (AA does not conform to the defined syntax rules)
- Duplication of AAID
- The base archive of delta AA is not registered
- Security Error (details to be discussed)
- Repository runs out of space

5.2 ApplicationArchive portType

An ApplicationArchive is a WS-Resource which represents an AA in the repository.

5.2.1 Resource properties

ApplicationArchive has the following resource properties. These properties can be retrieved through the interface of WS-ResourceProperties .

- createDateTime
- buildInfo (creation information)
- description
- property
- contentsList

[TBD: Details of each resource property]

5.2.2 Remove

Service requestor sends the following message to the ApplicationArchive to remove the specified AA from the repository.

```
<Remove>

  ... TBD

</Remove>
```

[TBD: Details of Remove request message.]

If the ApplicationArchive accepts the Remove request, it removes the correspond AA from the repository, returns the following response message and destroy itself.

```
<RemoveResponse/>
```

In the following case, the ApplicationArchive fails the AA removal and returns a error.

- Security Error (details to be discussed)

5.2.3 GetContent

Service requester sends the following message to the ApplicationArchive to retrieve Application Contents.

```
<GetContent>

      TBD

</GetContent>
```

[TBD: Details of GetContent request message.]

The ApplicationArchive extract the specified content and returns the following response message.

```
<GetContentResponse>

      TBD

</GetContentResponse>
```

[TBD: Details of GetContent response message.]

In the following cases, the ApplicationArchive fails the content extraction and returns a error.

- Security Error (details to be discussed)
- No content corresponding to the specified content type

5.2.4 PutContent

[TBD: Need discussion on whether direct modification of Application Contents (not by delta AA) should be allowed or not]

5.2.5 Transfer

Service requester sends the following message to the ApplicationArchive to transfer the correspond AA to the specified remote ACS.

```
<Transfer>
  <Target>
    <wsa:Address>
      http://www.example.org/remoteACS
    <wsa:Address>
  </Target>
</Transfer>
```

The ApplicationArchive executes the transfer task and return the following response message including the endpoint reference of the newly created ApplicationArchive in remote ACS.

```
<TransferResponse>
  <RemoteAAReference>
    <wsa:Address>
      http://www.example.org/remoteACS
    <wsa:Address>
    <wsa:ReferenceProperties>
      <Originator>
        http://www.example.org/localACS
      </Originator>
      <AAID>
          ...AAID
      </AAID>
    </wsa:ReferenceProperties>
  </RemoteAAReference>
</TransferResponse>
```

In the following cases, ACS fails the ApplicationArchive transfer and returns a error.

- Security Error (details to be discussed)
- Unknown reason

5.3 Events

In case of changes on the repository managed by ACS, ACS can generate events. Supporting event notification is not mandatory for a Grid system, but if the system supports it, the system utilizes the message exchange mechanism in accordance with WS-Notification. In this case, ACS inherit the Producer interface.

If ACS supports event notification, it must support all of the following event types (topics).

- the case that updating an AA which have the specified AA as the base has been registered
- the case that the specified AA has been deleted

5.3.1 ApplicationArchiveUpdated event

[TBD: Details of the topic and event message.]

5.3.2 ApplicationArchiveDeleted event

[TBD: Details of the topic and event message.]

6. Application Archive Format specification

6.1 Structure

The following figure illustrates the logical layout of Application Archive.

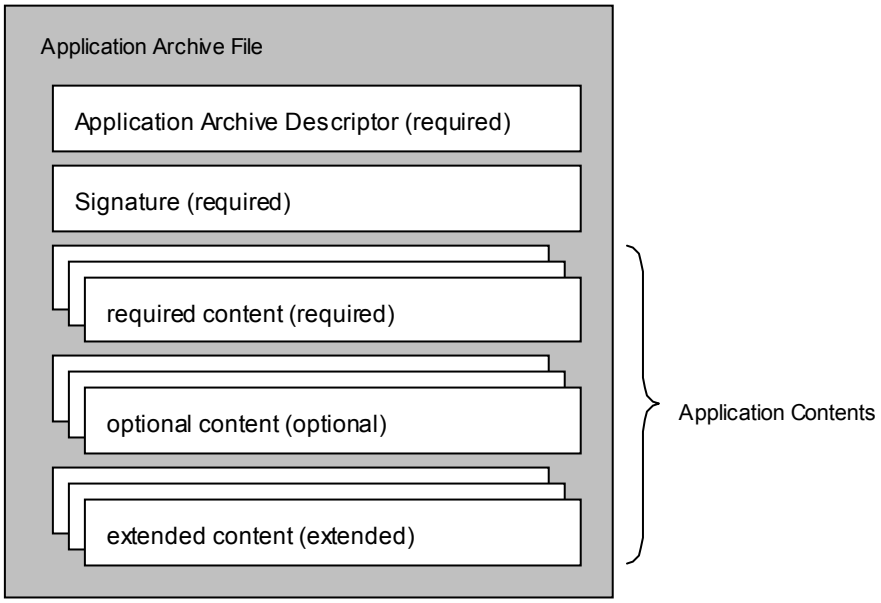

Figure 9: Logical layout of AA

- Application Archive Descriptor (AA Descriptor)

  AA Descriptor is an XML document describing information used by ACS to analyze the AA and verify the format validity of the AA. Specifically, list of Application Contents, identification information of the AA, author and created date and time of the AA, and so on, are described. This specification defines the schema of AA Descriptor.

- Signature

  Digital signature for detecting AA alteration and spoofing.

- Application Contents

  Component of the application such as executable binaries and initial data. Application Contents are classified into the following three categories: 1) Required Contents, which are required for the application, 2) Optional Contents, which are optional for the application, and 3) Extended Contents, whose type can be extended by users.

6.2 physical format

AA must be archived into a single file in ZIP form. AA Descriptor must be named "descriptor.xml" and located directly below the root directory. The Signature files must be located directly below META-INF directory which is located directly below the root directory.

6.3 AA Descriptor

6.3.1 Overview

The following figure illustrates the structure of XML elements in AA Descriptor.

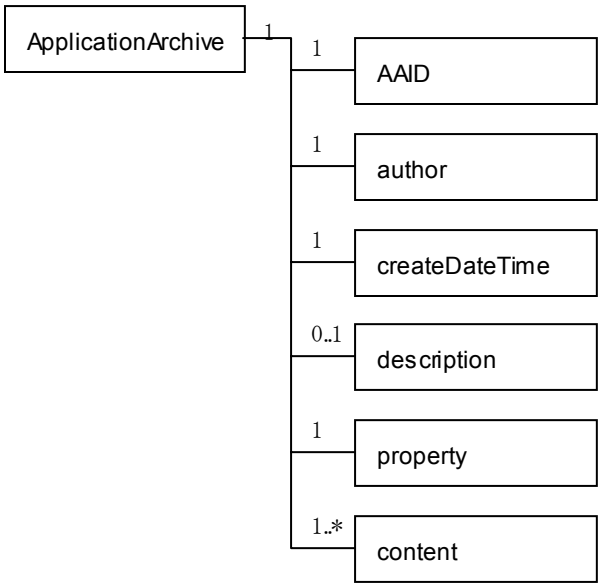

Figure 10: structure of AA Descriptor

6.3.2 ApplicationArchive

This is the root (document) element of AA Descriptor.

6.3.3 AAID

Information for identifying the AA is specified. It has the following child elements.

- name (REQUIRED) xsd:anyURI

  URI which represents the AA name. This is for grouping the same applications with different versions. Updating AA must have the same AA name as the base version of AA.

- version (REQUIRED)

  Version information of AA, which has the following child elements.
  - major (REQUIRED) xsd:nonNegativeInteger

- minor (REQUIRED) xsd:nonNegativeInteger
- revision (REQUIRED) xsd:nonNegativeInteger

The meaning of the numbers in the version is not defined. Unused number is also allowed.

AAID must be unique in an ACS. It is responsibility of AA author (or registerer) to guarantee the uniqueness of AAID.

The following is a sample of AAID.

```
<AAID>
 <name>
    http://www.foo.org/sample-application
 </name>
 <version>
  <major>1</major>
  <minor>0</minor>
  <revision>2</revision>
 </version>
</AAID>
```

6.3.4 author

Human-readable information about AA author.

- name (OPTIONAL) xsd:string

  The name of the author.

- description (OPTIONAL) xsd:string

  The description about the author.

- location (OPTIONAL)

  The address of the author, which has the following child elements.
  - country (OPTIONAL) xsd:string
  - address (OPTIONAL) xsd:string

The following is a sample of author.

```
<author>
    <name>Keisuke Fukui</name>
    <description>member of Fujitsu Laboratory</description>
    <location>
        <country>Japan</country>
        <address>Yokohama</address>
    </location>
</author>
```

6.3.5 createDateTime

Created date and time of AA, whose type is xsd:DateTime.
The following is a sample of createDateTime.

```
<createDateTime>2004-08-01T00:00:00</createDateTime>
```

6.3.6 description

Human-readable description about the AA, whose type is xsd:string.
The following is a sample of description.

```
<description>
  sample application of foo organization
</description>
```

6.3.7 property

Property of the AA is specified. It has the following child elements.

- type (REQUIRED)

  Type of the AA. Either "full" (AA contains all the Application Contents) or "delta" (AA contains only the differences from the base version) can be specified.

- baseversion (OPTIONAL)

  The base version of AA is specified only in case of updating AA. In this case, this element is required independently of the type of AA (full/delta). This element consists of the three numbers (major, minor, revision).

- scopeList (OPTIONAL)

  Declaration of labels given to Application Contents. The meaning of each label is defined not by ACS, but by the services which extract the individual Application Contents and interpret them. This element has more than 1 scope child elements, each of which contains the name of a label.

The following is a sample of property for full AA.

```
<property>
 <type>full</type>
</property>
```

The following is a sample of property for delta AA.

```
<property>
 <type>delta</type>
```

```
 <baseversion>
   <major>1</major>
   <minor>0</minor>
   <revision>1</revision>
 </baseversion>
 <!-- list of scope (optional) -->
 <scopeList>
   <scope>basic</scope>
   <scope>ext1</scope>
   <scope>ext2</scope>
 </scopeList>
</property>
```

6.3.8 content

Information about a single Application Content contained in AA.

The content element has the following attributes.

- type (REQUIRED)

  The type of Application Content. The following strings are reserved for the type (for details, see §6.4), but other string may be specified.

  TBD

- name (REQUIRED)

  The name of Application Content. The name is used as a key for extracting the Application Content, and must be unique in the AA.

The content element has the following child elements.

- file (REQUIRED)

  Information about a single file contained in the Application Content. This element contains the name of the file (absolute pathname from the directory where the AA is decompressed). If the value of the externalReference attribute is "true", this element contains the URL of an external file. This element has the following attributes.

  - externalReference (OPTIONAL)

    It is set to "true" when referring an external file outside the AA. The default value is "false".

  - scpose (OPTIONAL)

    One of the scope defined in the list in property element may be specified If the file belongs to a certain scope.

  - operation (OPTIONAL)

This attribute, used in the case of updating AAs, specifies the type of update operation on the base AA. If this attribute is omitted, the file is considered to be unchanged from the base AA. The following values can be specified:

- add (the file is added)
- update (the file is updated)
- delete (the file is deleted)

The following is a sample of content.

```
<content type="dataDeployment" name="dataDeployForSample">
  <file>./dataDeploy/deploy.zip</file>
  <file scose="ext1">./dataDeploy/deploy-ext1.zip</file>
   <file externalReference="true"
        operation="add">http://foo.com/sample.txt</file>
</content>
```

6.4 Application Contents

The following is a list of the Application Content TYPES. The repository does not interpret the contents in any way.

- Application Configuration Description (REQUIRED)

  It describes the logical structure of the application. Specification of this description will be defined by other standard, e.g. CDDLM-WG of GGF.

- Procedures for lifecycle management operation (REQUIRED)

  Such procedures define how to initialize, start and terminate the application. They are deployed to the appropriate location, e.g., hosting environment, as part of job instantiation. The procedures are executed when there is a change in the job lifecycle state. The job lifecycle state is now under discussion in CDDLM-WG.

- Self-Management Policies (OPTIONAL)

  These are policies that should be followed when an event, such as a failure or a load change, occurs. The policies are registered with the appropriate service, e.g., the fault management service, as part of job instantiation. A policy is applied when the specified event occurs to the job.

- Application Programs and Initial Data (OPTIONAL)

  These components are to be deployed on the allocated resources by Deployment service that is being discussed at CDDLM-WG. This data does not have to be in any specific format.

- Extension data. (Extension)

  Repository contents are extensible. Any domain or system specific data may be added.

[TBD: Whether each item is required or optional]
[TBD: Another items may be added to this list.]

6.5 Signature

[The following is signature mechanism in JAR specification. We expect that AAF signature will be similar to this]

Signature files is created in the so-called MANIFEST method and stored in AA. The following is the procedure for creating signature files.

(1) For each file in the archive (including AA Descriptor), compute the cryptographic hash value of the file in an uncompressed state, regarding the file as a byte array.

(2) Enumerate in the manifest file (MANIFEST.MF) the filenames and the hash values of the files above. (The manifest file originally contains other information, but it is unnecessary for the purpose of signature.)

(3) Create a SF file (.SF) in text format by extracting filenames and hash values from the manifest file and adding the pre-defined header information. (Except the header informaiton, the content of the manifest and SF files can be considered to be the same.)

(4) Create a signature file (.RSA) by executing RSA digital signature on the SF file.

(5) Locate the three files above directly below META-INF directory which is located directly below the root directly of the archive.

7. Relationship with other technologies

There are many systems that partially overlap ACS. In this chapter, we note our understanding for them and relationship with those and ACS.

7.1 Installation packages

7.1.1 Packages for installation on OS

There are package formats on various operating systems such as PKG on UNIX, RPM on linux, MSI on Windows. ACS handles these archive files as one of Application Contents.

7.1.2 Packages specific to middleware

There are the package formats to deploy applications on specific middleware, or hosting environments. Examples are jar, war and ear files on application server and gar files on globus toolkit. Grid applications are likely to be developed in these formats. ACS handles these archive files as one of Application Contents.

7.1.3 Solution Installation

The structure of the Solution Installation and AAF looks similar, in which both make use of the XML document in descriptors, and incorporate multiple set of application binaries to be bundled in one. However, we see two are different in scopes of target application and purposes. Solution Installation is mainly focus on pre-packaged ISV software and doesn't scope a submitting them to grid systems;

rather they are for the integration of the group of the pre-packaged software. It also focus on the automated installation but not scope working with certain Grid services, e.g. binding between a job and resource, agreement between consumer and provider, job scheduling, and so on. On the other hand, ACS aims at handling the application tailored or customized to the user requirement, and executed in Grid systems.

Solution Installation is an emerging specification submitted to W3C in June, 2004, which aims to be platform-independent standard. The standardization process is yet to be started. Thus, at this moment, we are not sure if there are possibilities of its future enhancement to include our purpose, scope and requirements. These points need to be communicated and clarified in the working group activities.

7.2 Software provisioning systems

There are a products or services for provisioning software on multiple hosts. Most of these technologies have sophisticated and wide-ranging functionalities and have shown their effectiveness especially in maintaining a large number of machines in organizations or data centers. Their implementations, however, are developed separately and thus different each other. They are not focused in standard and service-oriented approach, which is encouraged by GGF OGSA.

Here the standard is needed for grid applications collecting the luminance of those independent technologies, while following the advocate of the GGF OGSA.

7.3 Other standard bodies

7.3.1 DCML

TBD

8. Security Considerations

TBD

9. Samples

TBD.

10. Schema Definition

10.1 application archive descriptor (xsd)

TBD.

10.2 application repository interface (wsdl)

TBD.

11. Open Issues

- More appropriate name and its abbreviation for Application Archive (AA). [3.1 Definition of Application Archive]
- Details of Register interface, and method of uploading the entity to be registered. [5.1.1 Register]
- Details of Remove interface. [5.2.2 Remove]
- How to specify the Application Content in GetContent interface. [5.2.3 GetContent]
- Whether the PutContent interface is necessary or not. [5.2.4 PutContent]
- Details of events. [5.3.1 event of AA updating and 5.3.2 event of AA deletion]
- List up and details of types of Application Contents. [6.4 Application Contents]
- Clean up the relationship with DCML. [7.3.1 DCML]
- Details of security [8 Security Consideration]
- Details of samples [9 Samples]
- Details of schema definition [10 Schema Definition]

Author Information

[Contact information for authors.]

Glossary

Grid System	OGSA-compliant Grid middleware consisting of OGSA services.
Grid Application	An application executable on Grid systems.
Application Contents	A set of information which collectively makes the application ready to run on Grid systems and to generate an expected output.
Application Archive	A consistent unit of Application Contents associated with a single application.

Intellectual Property Statement

The GGF 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 GGF Secretariat.

The GGF 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 GGF Executive Director.

Full Copyright Notice

References

[OGSA doc]
"The Open Grid Services Architecture", Version 1.0, I.Foster , D.Berry, A.Djaoui, A.Grimshaw, B.Horn, H.Kishimoto , F.Maciel, A.Savva, F.Siebenlist, R.Subramaniam, J.Treadwell, J.Von Reich, https://forge.gridforum.org/projects/ogsa-wg

[OGSA glossary]
"Open Grid Services Architecture: Glossary of Terms"", Version 1.0, I.Foster , D.Berry, A.Djaoui, A.Grimshaw, B.Horn, H.Kishimoto , F.Maciel, A.Savva, F.Siebenlist, R.Subramaniam, J.Treadwell, J.Von Reich,
https://forge.gridforum.org/projects/ogsa-wg

[OGSA usecase]
"Open Grid Services Architecture Use Cases", March 4, 2004, I. Foster, D. Gannon, H. Kishimoto, Jeffrin J. Von Reich
http://www.ggf.org/documents/GWD-I-E/GFD-I.029.pdf

[BizGrid]
“Business Grid Middleware Goals and Status”, January 20, 2004, Hiro Kishimoto, Takashi Kojo, Fred Maciel, , http://www.globusworld.org/program/slides/3c_1.pdf

[WS-N]
"Publish-Subscribe Notification for Web services", Version 1.0, May 3, 2004, Graham, S. (ed), Niblett, P. (ed), Chappell, D., Lewis, A., Nagaratnam, N., Parikh, J.Patil, S., Samdarshi, S., Sedukhin, I., Snelling, D., Tuecke, S., Vambenepe, W., Weihl, B.
http://www.oasis-open.org/committees/download.php/6661/WSNpubsub-1-0.pdf

[WS-RF]
"The WS-Resource Framework", Version 1.0, March 2004, Czajkowski, K., Ferguson, F. D., Foster, I., Frey, J., Graham, S., Sedukhin, I., Snelling, D., Tuecke, S. and Vambenepe, W..
http://www.oasis-open.org/committees/download.php/6796/ws-wsrf.pdf