UDAP
Use Cases and Requirements

Editors:	A. Anjomshoaa	EPCC
	Henning Mersch	FZJ
	Ph. Wieder	FZJ

Date	Author	Comments	Version	Status
…	H. Mersch	Initial version	V0.1	Draft
14/05/2007	Ph. Wieder	Update	V0.2	Draft
09/2007	Ph. Wieder	Response to review	V0.3	Stable draft

1 Introduction

In the current state of Grid architectures, information about an activity is fragmented and dispersed. Activity information, such as resource usage, security data, activity state, data requirements, et cetera, is currently captured using a variety of schemata and stored in different ways and by different logical components.

This dispersion of activity information leads to management, security and logistical overhead in discovering, accessing and using that information. This uncoordinated scheme for finding activity information assumes a lot of Grid components and Grid system coordinators. It results in an environment where activity information is managed by many systems. This makes it difficult for Grid components that rely on that information to know where the information is, leading to them having to search for and to "keep an eye" on many sources of activity information.

The Universal Dynamic Activity Package (UDAP) aims to bring all of the information fragments that are associated with an activity, regardless of the various schemata that are used to describe and capture these fragments, into one logical package. It then aims to specify i) how to write activity information to that package, ii) how to query that package for activity information, ii) how to extract information from that package for read and update operations, and iv) how to insert new activity information into that package.

Using UDAP, any Grid component requiring activity information has a single source for retrieving and using that information. Grid components can subscribe to the interface of an entity managing an activity, so that they may be notified of changes in activity information, thus allowing them to fulfil their role with respect to that activity. The same Grid components can poll the UDAP management interface for activity information in read, update and append operations, in addition to or instead of subscribing for notifications.

The subscribe-notify capability of the UDAP components leads to many interesting use cases, some of which are introduced later in this document (see Section 3). If applied as a component central to a system, the variety of UDAP use cases is not limited. In contrast, looking at UDAP from a technical angle, the number of use cases is very much limited to what has said before: write, query, read, update, and insert information about an activity through a subscribe-notify mechanism. In general, it means that Grid components can simply wait for activity information generated events that would trigger particular actions. This leads to an event based activity management environment. Nevertheless, to outline potential usage scenarios of UDAP in a NextGRID context, this document describes a number of activity-related use case and the resulting requirements.

2 Components

The Universal Dynamic Activity Package (UDAP) is a scheme for managing all the information associated with an activity on a Grid. This in turn allows any Grid component or resource which is

concerned with that activity to access and use that information using the UDAP scheme to do its task.

2.1 Definition of Activity

An activity is a *unit of work* on a Grid. 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; something 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.

And 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.

Furthermore, an activity is considered to be a resource. The realised activity must implement the management interfaces of a resource so that it may be managed as such on a Grid. This leads to activities being dependent on and used by other resources. An activity may require another activity in its list of resource requirements. The statements: "depending on an activity" and "requiring an activity resource" are syntactically identical.

2.1.1 Activity Information

We take a holistic view of an activity. We consider all that there is to know about an activity:

- all of its requirements,
- all of its dependencies (on data and other activities) for the composition and management of activities for workflow, scheduling and brokering processes,
- all of its contextual information, such as:
  - topical domain (financial markets, weather forecasting, etc.),
  - security (who owns the activity, who is allowed to run it, etc.),
  - SLAs, QoS and other related policies, and
- all of its monitoring information, such as:
  - state,
  - history,
  - resource information,
  - accounting,
  - policy conformance, et cetera.

Activity information therefore includes such things as:

- resource requirements (hardware, software, services, data, etc.),
- dependencies on data and other activities,
- subsistence state,
- security and ownership context,

- migration rules and policies,
- SLA policies governing transactions,
- accounting and auditing,
- history and provenance, and
- any other information associated with the activity.

2.2 UDAP Concepts

The core of the UDAP model is the UDAP document. Its information is managed by a UDAP Manager and processed by UDAP clients. In the following, these concepts are described in detail. The respective implementations of the concepts are outlined in the "Resource Discovery using the UDAP System" recipe of the NextGRID cookbook [1]. Please note that the existing components are implemented according to the NextGRID Basic Profile [2] and that upcoming implementations have to follow the Basic Profile, too.

2.2.1 UDAP Document

At the heart of the UDAP model is the UDAP Document (the relation between UDAP concepts is pictured in Figure 1). This conceptual document is the package that holds all the information associated with an activity. It needs a schema structure that allows it to classify the information that it holds for an activity, and to reflect the state of that activity at any point in time. This schema

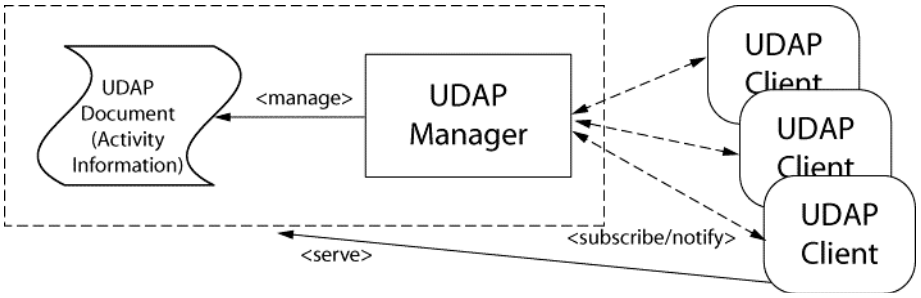

Figure 1 The figure shows the UDAP Manager in charge of a UDAP Document for an activity (represented by the dashed box) on a Grid. UDAP Clients concerned with that activity may subscribe

is defined as an XML Schema named "UDAP Schema" which is described in the respective NextGRID Generalised Specification [3].

2.2.2 UDAP Manager

The UDAP Manager is the entity that manages the information in the UDAP Document. The UDAP Manager should have a standardised public interface that allows any Grid component to invoke its management functions for read, update and append operations of activity information.

2.2.3 UDAP Client

A UDAP Client is any Grid component that uses and/or produces activity information. UDAP Clients invoke the public management interface of the UDAP Manager for read, update and append operations of activity information.

A UDAP Client can subscribe to the interface of a UDAP Activity Resource in order to receive notification of activity information based events. The subscription of a UDAP Client may be conditional, where the condition dictates the type of activity information based event that the client is interested in, e.g. “notify me if the state of the activity changes to running” or “notify me if the resource usage of the activity has exceeded the budget of the activity owner”.

2.3 UDAP as an Intermediary

An interesting feature of UDAP is that it acts as an intermediary[1] between activity owners and the service and resource providers that serve that activity (see Figure 2). This is as a result of the UDAP Activity Resource being at the centre of the interactions between the activity owner and the service providers.

The activity owner, who initiates the activity, must do so through a Grid component, such as a job submission client, or Grid portal. That Grid component, then, in turn is a UDAP Client and can subscribe to the UDAP Activity Resource that it has initiated for activity information based events.

This intermediary quality of UDAP leads to a symmetric, uniform interface for activity management, with the activity initiating Grid component on one side, and the service providers that serve that activity on the other, all interacting with and using the Activity Resource using the same interface.

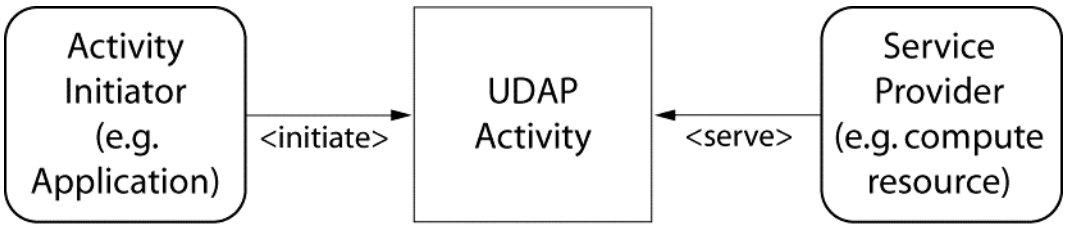

Figure 2 UDAP as an intermediary between an activity owner/initiator and the service

3 Use Cases

The potential use cases for UDAP are manifold and not limited by the model itself. In this section we list a number of these cases to give an idea of what can be achieved by the application of the UDAP model.

[1] The idea of an UDAP Intermediary has been massively used within the first realisation of the UDAP system, mainly to de-couple resource provider and resource consumer concerning the advertisement and request of resources. Please refer to **Fehler! Verweisquelle konnte nicht gefunden werden.** for a detailed description.

3.1 Job Submission

A Grid user, application or job submission client discovers a UDAP Factory from a registry. It then invokes that UDAP Factory to instantiate a new UDAP Activity Resource Service by submitting to it a description of the activity which includes the activity's requirements and any contextual and dependency information about that activity.

The UDAP Factory performs a search on its list of subscribers (subscribed UDAP Clients) to find suitable ones in order to fulfil the activity. If it finds suitable UDAP Clients that match the requested activity's requirements, it then instantiates a WS-RF UDAP Activity Resource.

The UDAP Factory then notifies the suitable UDAP Clients it has found of the activity's existence, sending them the Activity Resource Service's EPR. The UDAP Clients identified can then contact the new activity resource instance to find out further details of its requirements. If they are able to fulfil those requirements, an SLA will be agreed between each UDAP Client and the activity resource. The activity can then be served by those UDAP Clients that have an SLA in place with it.

3.2 Service and Resource Discovery

Taking up a common scenario in Grids, a client-side agent requests compute resources and expects a list of potential service providers back from the UDAP system.

Within the setup procedure, the UDAP Manager is started and waits for incoming subscription requests. UDAP Subscribers can subscribe themselves to the UDAP Manager, passing as arguments their EPR address [4] and an XPATH [5] query expression about their capabilities for processing UDAP Documents, i.e. the service or resource capabilities they provide.

Following this, the UDAP system is set up and can be used by UDAP Generators. A UDAP Generator retrieves information about the resources to query from some other component like an end user interface or a broker. The UDAP Generator then generates, based on the information it has retrieved, a UDAP Document and submits this to the UDAP Manager, which implicitly means it will subscribe to changes on that UDAP Document. The UDAP Manager will now execute all capability descriptions of the UDAP Subscribers to the UDAP Document and – in case they match – inform the UDAP Subscriber with a reference to the UDAP Document.

The UDAP Subscriber could now retrieve this UDAP Document and process it by interpreting the content. Afterwards, the UDAP Subscriber will modify the UDAP Document at the UDAP Manager. In a common scenario, where services or resources are to be discovered based on their capabilities, this would mean to insert one or more service provider EPRs.

Every time a modification is made to the UDAP Document, the (implicitly subscribed) UDAP Generator will be informed about the changes, which will be service provider EPRs for our concrete scenario. At any time new UDAP Subscribers could be added to the UDAP Manager, which enhances the facilities of the overall system.

3.3 Job Management

State Monitoring and Control

Once an activity resource has been instantiated and has a set of UDAP Clients working on fulfilling its requirements, it needs to be managed. It should be the remit of a Job Manager component to manage, monitor and control that activity.

The Job Manager becomes a UDAP Client. It subscribes to the UDAP Activity Resource Service for event notifications, so that it can perform activity management operations conditional on changes in the activity's state and information, as specified in the activity's requirements.

In managing the activity, the Job Manager may perform job control functions based on conditional events generated by the activity resource.

Job Check-pointing and Migration

This is an example of a control function that may be performed by a Job Manager. If the Job Manager has the capability, it may be able to checkpoint and migrate the activity it is in charge of.

Such a capability may be explicitly required by an activity and specified as a requirement in its activity information set. Further, the requirement for check-pointing and migration and the ability of a Job Manager to deliver that capability may be explicitly captured in an SLA between the instantiated UDAP Activity Resource Service being managed and the Job Manager UDAP Client managing it.

Job Termination

This is another example of a control function that may be performed by a job manager.

3.4 Supporting an SLA Framework

The NextGRID SLA framework [6] requires that each pair of interacting Web Services must have in place an SLA that governs their interaction in terms of guaranteed quality of service.

In the UDAP scheme, this requirement leads to there being an agreed SLA in place between each UDAP Client-UDAP Activity Resource Service pair. This SLA would set out the levels of service and quality expected from a UDAP Client when serving a UDAP activity. It will also set out the conditions on the activity under which it will be served, such as the required security conditions being fulfilled and an appropriate price being paid for the service provided.

An activity being served by many UDAP Clients would have an agreed SLA in place with each UDAP Client.

3.5 Supporting Data Centricity

Data centricity may be defined as the set of operations that need to be conditionally performed on some data of interest. These operations include:

- producing,
- storing,
- transporting,
- filtering,
- transforming,
- processing,
- accessing,
- reading,
- writing,
- updating,
- archiving,
- et cetera.

When user or application requirements are data centric, "activity operations", such as those listed above, must be performed on the specified data as a result of events generated due to previous operations performed on that data, or due to the state of that data.

This leads to a workflow of activity nodes, that are conditionally executed based on events generated as a result of previous operations.

The UDAP scheme enables an environment to be built, where activities may be composed together and conditionally executed by a data centric job manager (a workflow engine) based on events, in order to meet the needs of a data centric process.

3.6 Accounting

When an activity is being fulfilled by one or more UDAP Clients, there are accounting requirements that govern the use of the resources.

Accounting information may be kept and updated in the activity's information directory (the WS-Resource Properties (WS-RF) Document [7]). When new accounting information is available, events can be generated and sent to subscribed UDAP Clients which are interested in that information.

3.7 Archiving

When an activity is being fulfilled by one or more UDAP Clients, there may be provenance and archiving requirements for that activity.

The UDAP schema allows for provenance and archiving of activity information to be captured in the schema. However, this may not be practical. As activities progress in their life, the provenance and archiving data that they carry may blow up and hinder performance of the UDAP Activity

Resource Service. This should be borne in mind when implementing the UDAP Manager and UDAP Document concepts.

A good way to capture provenance and archiving information about an activity is to have a service that is dedicated to undertaking provenance and archiving. The implementation of this service may be specialised to do provenance and archiving in whatever way or form and using whichever schema. The important point to bear in mind is that this provenance and archiving service can be a UDAP Client which can subscribe to a UDAP Activity Resource to receive notification every time a piece of information about the activity changes (that information being captured in the Resource Properties Document of a WS-RF UDAP Activity Resource Service).

Provenance and archiving have a number of use-cases in themselves. For example, it may be necessary to undertake provenance and archiving for fault tolerance, in case of catastrophic failure of services or resources that are serving an activity. Under such a circumstance, the provenance and archiving data will act as a snapshot of the "memory footprint" of the activity's information and state and can be used for the activity to be migrated to other services or resources.

4 UDAP Requirements

UDAP manages various information belonging to (as prior defined) Activities, thus it has to be informed and used to manage this information form various components. Nevertheless, all components interacting with UDAP will directly benefit from its capabilities like acting as intermediary. Thus we expect UDAP to be one of the central components of a NextGRID implementation.

4.1 Scalability

For the UDAP Framework, the crucial component when it comes to scalability, is the UDAP Manager. Due to the fact it keeps track of the inserted UDAP Activities, we currently think about a scalable system of UDAP Managers. Another UDAP Manager could subscribe for newly and/or otherwise un-processable UDAP Activities and will be notified to process a UDAP Activity from the first UDAP Manager. This would lead to a kind of a *Peer-to-Peer*–Network of UDAP Managers, which is envisaged to scale very well. Due to the subscription/notification mechanism used, such a UDAP Manager network should be not too difficult to realise. Please note that this feature is not yet planned to be implemented to the UDAP Framework, but initial design considerations are under way.

4.2 Security

UDAP Clients that are able to fulfil the requirements of an activity, must match the requirements stated in the activity's security context. The security requirements will most likely be bipartite, and so the activity must match the requirements of the UDAP Clients.

The ability to satisfy bipartite security requirements between the activity and a UDAP Client should be captured in the SLA between them.

Furthermore, details of the fulfilment of an activity's security requirements by UDAP Clients must be put into that activity's information set (the UDAP Document rendered as the WS-RF Resource Properties Document).

5 References

[1] K. Tserpes, NextGRID Cookbook, NextGRID, March, 2007.

[2] V. Li and D. Snelling, NextGRID Basic Profile, NextGRID, June, 2006.

[3] A. Anjomshoaa and Ph. Wieder, UDAP Schema, NextGRID Generalised Specification, NextGRID, May, 2007.

[4] M. Gudgin and M. Hadley (eds.), Web Services Addressing 1.0 – Core, W3C Candidate Recommendation, W3C, 17 August 2005. http://www.w3.org/TR/2005/CR-ws-addr-core-20050817/.

[5] A. Berglund, S. Boag, D. Chamberlin, M. F. Fernández, M. Kay, J. Robie, and J. Siméon, XML Path Language (XPath) 2.0, W3C Candidate Recommendation, W3C, June, 2006. <http://www.w3.org/TR/xpath20/>.

[6] B. Koller, Ph. Masche-Pakkala, B. Mitchell, and H. Mizani, NextGRID SLA Framework, NextGRID Project Output, P4.5.6, September, 2006.

[7] S. Graham and J. Treadwell (eds.) Web Services Resource Properties 1.2 (WS-ResourceProperties), OASIS Standard, 1 April 2006. http://docs.oasis-open.org/wsrf/wsrf-ws_resource_properties-1.2-spec-os.pdf