

A Grid Information Service Schema for Grid Events

GridForum Performance Working Group
Warren Smith, Dan Gunter
October 3, 2000

1 Introduction

A grid event service provides the mechanisms needed to transmit events from where they are produced to where they are consumed. An overview of the basic architecture we have defined for this task is described in the Grid Monitoring Service Architecture document of the Performance Working Group. A Grid Information Service (GIS) [1] is used in several ways in this architecture. First, event producers use a GIS to advertise their existence and the types of events that they provide to event consumers. Consumers can then search the GIS to find producers that supply the information they are interested in. Second, a location is needed to store what we are calling an *event dictionary*: a database of commonly used event types and event elements. This dictionary gives producers and consumers of events from different organizations a common “language” to use when communicating with each other.

Recent activity in the Grid Forum organization indicates that the Grid community is in the process of adopting databases accessed using the Lightweight Directory Access Protocol (LDAP) [2, 3] as its standard for information services. In this document, we first describe how to use a GIS to store information about event producers and how consumers can use this information to find producers. Second, we describe how to use a GIS to store an event dictionary.

The next section provides some background information. Section 3 describes how providers of events can advertise their existence and the types of events they provide and consumers can find providers using a LDAP database. Finally, Section 4 discusses how an event dictionary can be implemented using a LDAP database.

2 Background

We define a very basic architecture in the Grid Monitoring Service Architecture working document. This architecture is shown in Figure 1 and consists of three components. Producers are entities such as a host monitor or application performance monitor that are generating events. A consumer is an entity that wishes to receive events from a producer. Finally, the directory service is used for consumers to find producers. The figure also mentions the most important piece of data in our design events. Events are time-stamped sets of information that are generated by producers and sent to consumers. The figure also illustrates what interfaces and data we need to define. We need to define the interface between producers and consumers. This interface consists of the protocol they use to communicate. The interface to the directory service is LDAP but we must define how our data is stored in the directory service. That is the focus of this document.

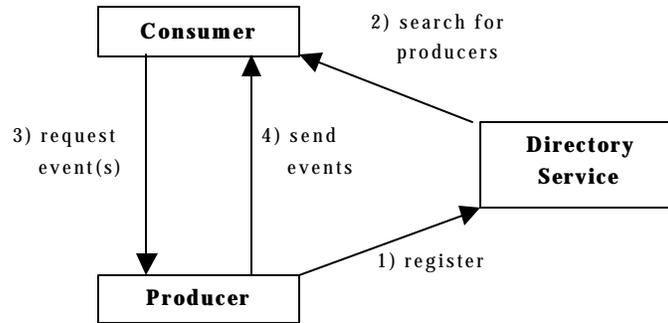


Figure 1. Our basic architecture for monitoring in computational grids.

At this point, we do not plan to store events in the information service. However, events are one of the central components of our monitoring architecture so it is good to define exactly what we think they are. An event consists of:

1. A type that identifies the event type object that constrains the event or the event may be untyped.
2. A set of elements. Each element has a unique name within the event and one or more typed values. Our current idea is to only have simple types. That is, types such as char, short, long, float, double, unsigned char, unsigned short, unsigned long, long double, and string. The consensus seems to be that having more complex types is not necessary at the current time. Each value may also have units and accuracy associated with it. There are several elements that are part of every event such as a timestamp.

Each event has a type. The type identifies an event type structure or object (we'll use object here). An event type object is a set of constraints on what elements an event may or must have and what type of values must be contained in these elements. An event type consists of a name and one or more element descriptions. Each element description consists of:

1. A name,
2. a type,
3. a text description,
4. a flag indicating if the element must be present in a valid event of this type.

To allow events to be understood, event types and their associated information must be propagated to consumers so that consumers can determine what events contain the information they need. One way we wish to support this propagation is to provide an *event dictionary*. An event dictionary is simply a listing of the common event types that are defined by various organizations. Such event dictionaries can be stored in an LDAP-based information service and we will address the issue of how this can be accomplished in this document. LDAP may be a useful way to store event dictionaries because an LDAP server can provide a structured set of event type information that is easily searched or dumped using LDAP client libraries.

Another object to discuss is what we call an event description. An event description is the information a consumer provides to a producer to describe the events it is interested in. For example, a consumer may want to know information about the status of a process. The consumer must therefore provide information such as the process ID to monitor and how often to monitor it. An event description consists of a set of parameters and a filter. The parameters are passed to a producer and are input parameters that

determine how and which events a producer generates. The filter is then used by the producer to determine which of these events should be sent to a consumer. Each parameter description consists of

1. A name
2. a type
3. a text description
4. a flag indicating if the parameter is required in this type of event description.

The other part of an event description the filter. The filter is a logical expression that describes under what conditions an event that has been generated should be sent to a consumer. For example, send an event if the status of a process is that it doesn't exist, or send an event if the CPU load goes over 2.0.

We also need to define what information about producers should be represented in our information service so that consumers can find the producers they are interested in. A producer can be described with:

1. A list of the names of the event types it provides.
2. Constraints on how the provider provides events. This includes information such as:
 - a. Communication protocols,
 - b. authentication methods,
 - c. encryption techniques,
 - d. who has access,
 - e. minimum and maximum event rates,

2.1 LDAP

Recent projects to provide computational grid services and recent activity in the Grid Forum indicate that the community is adopting distributed databases accessed using the Lightweight Directory Access Protocol as a primary source of information in computational grids. LDAP is a standard protocol to access directory services that has become a de facto standard. It was originally intended to access X.500 directories using a simpler interface than the X.500 Directory Access Protocol (DAP). While very few of the current LDAP servers are implemented using X.500, this original intention of LDAP has defined some of the assumptions about how the data accessed through LDAP is organized.

LDAP assumes that the entries in the database are organized in a tree, similar to the DNS namespace. A LDAP namespace has a root and for computational grids, the root we use is "o=Grid". Our namespace, called the Directory Information Tree (DIT) branches out below this root by adding organizations, organizational units, resources, etc. An example DIT is shown in Figure 2. The unique name for the "service=JobManager" entry on the lower left side is "service=JobManager, hn=evelyn.nas.nasa.gov, ou=Ames Research Center, o=National Aeronautics and Space Administration, o=Grid". It's a rather long unique name, but you can see how it is formed by walking up the tree to the root. To use a LDAP directory service, we must define where our monitoring entries should be placed in this DIT.

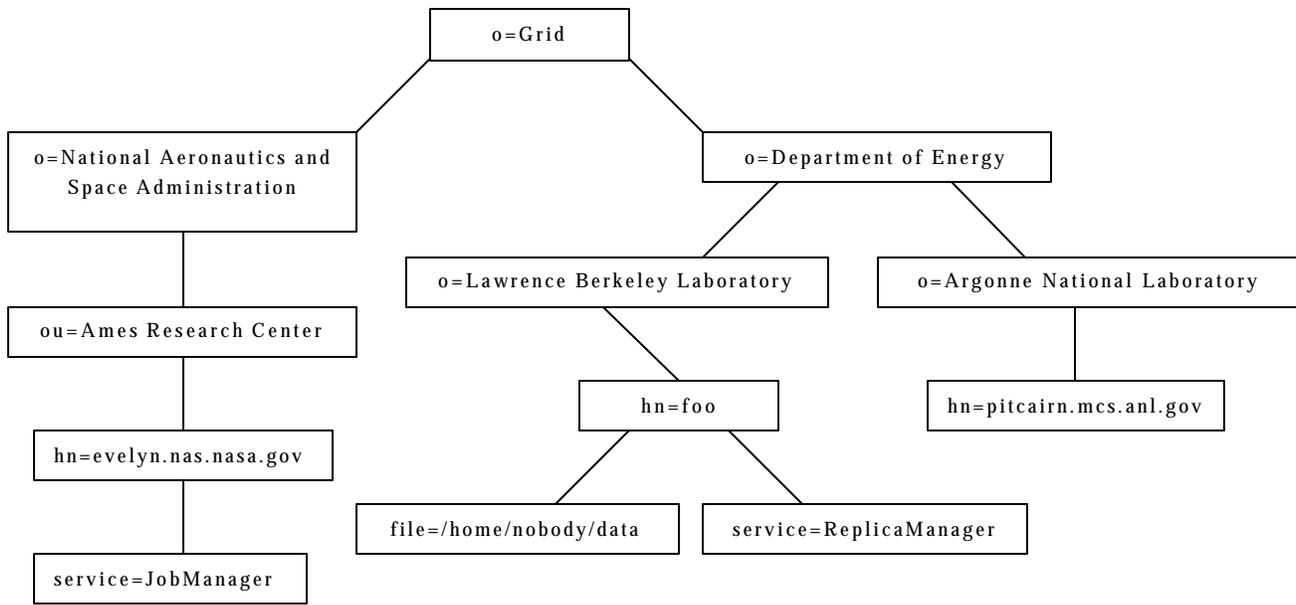


Figure 2. An example directory information tree.

All of the entries in an LDAP database are defined to be instances of one or more “types” The LDAP “types” are called object classes. An object class consists of a name, all of the elements (<name, value> pairs) that must be present in an instance of that object class, and all of the elements that may be present in an instance of that object class. An object class can also be derived from an existing object class. This means that the derived object class contains all of the must and may elements of its parent. Other must and may elements can then be added in the child class.

3 LDAP Object Classes

This section describes the object classes we have defined that allows consumers to find producers and allows us to maintain a dictionary of event types. We currently plan to use four different LDAP object classes. They are:

EventProducer. The EventProducer class is the object class associated with an event Producer. It contains information about how to contact the producer.

EventAccess. The EventAccess class describes who can receive which events from the Producer using what protocols, authentication methods, and security methods. Instances of this class are associated with the EventProducer instances.

EventType. The EventType class provides a description of one type of event that is provided by a Producer.

EventElementType. The EventElementType object class describes one element that may, must, or cannot be contained in an event that is an instance of this description. Instances of this class are associated with EventType instances.

EventParameterType. This object class describes a parameter that can be provided by a consumer when the consumer is requesting events from a provider. Instances of this class are associated with EventType class (Can different producers could have different parameters for an event?).

Table 1 through Table 5 provide a detailed description of these four object classes.

Table 1. EventProducer LDAP objectclass.

Name	Type	Required or Optional	Number of Values	Description
ProducerName	String	Required	1	A unique name for this event producer under the entity where it sits in the DIT.
HostName	String	Required	1	The host name the Producer is running on.
Port	Unsigned int	Required	1	The port the Producer is listening to.

Table 2. EventAccess LDAP objectclass.

Name	Type	Required or Optional	Number of Values	Description
AccessName	String	Required	1	A unique name for this event access instance under it's event producer.
Event	DN	Optional	0+	The distinguished names of the events that the access object applies to. If no event names are specified, the access object applies to all accesses to the provider.
Authentication	String	Optional	1+	A list of authentication methods that can be used. If none are specified, there are no required authentication methods.
Service	String	Required	1+	A list of services provided (subscription, query, ...)
ProtocolVersion	String	Required	1	The version number of the messaging protocol used to encode events and other messages.
CommunicationProtocol	String	Required	1+	A list of communication protocols that can be used (TCP, UDP, IP Multicast)
AllowedUsers	String	Optional	1+	A list of users that are allowed to receive events from this Producer. Wildcards are allowed. The format of the user IDs depends on the authentication method being used.

Table 3. EventType LDAP objectclass.

Name	Type	Required or Optional	Number of Values	Description
EventName	String	Required	1	The name of the event.

Table 4. EventElementType LDAP objectclass.

Name	Type	Required or Optional	Number of Values	Description
ElementName	String	Required	1	The name of the element.
Type	String	Required	1	The type of the element (char, short, int, long, float, double, unsignedChar, unsignedShort, unsignedInt, unsignedLong, longDouble, string).
Description	String	Optional	1	A textual description of the element.
Units	String	Optional	1+	A list of possible units that can be used for the value of this element in an event.
Required	Boolean	Optional	1	If the element is required to be set in a valid event.

Table 5. EventParameterType LDAP objectclass.

Name	Type	Required or Optional	Number of Values	Description
ParameterName	String	Required	1	The name of the parameter.
Type	String	Required	1	The type of the parameter (same as the Type in the EventElementType class).
Description	String	Optional	1	A textual description of the parameter.
Units	String	Optional	1+	A list of possible units that can be used for the value of this element in an event.
Required	Boolean	Optional	1	If the element is required to be set when requesting events of the associated type.

4 LDAP Directory Information Tree

The next issue we will address is where the objects we just described live in the directory information tree. The first thing we note is that there may be many producers of events of a certain type. This indicates that it would be wasteful to duplicate this event type information under each producer in the DIT. Second, instances of both the EventElementType and the EventParameterType are associated with an instance of the EventType class. This indicates that there should be a subtree rooted at an EventType

instance that has children that are instances of the EventElementType and the EventParameterType classes. An example of this sub-tree is shown in Figure 3.

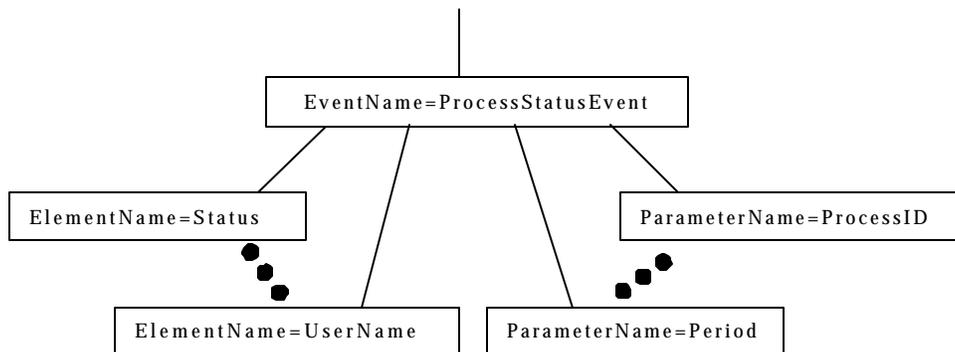


Figure 3. Event type sub-tree in the directory information tree.

The other two object classes we defined in the previous section are the EventProducer and EventAccess objects. Each producer instance will have one or more access instances associate with it. It therefore makes sense to have a sub-trees that are EventProducer instances, each of which have one or more EventAccess instances as children. An example of this sub-tree is shown in Figure 4.

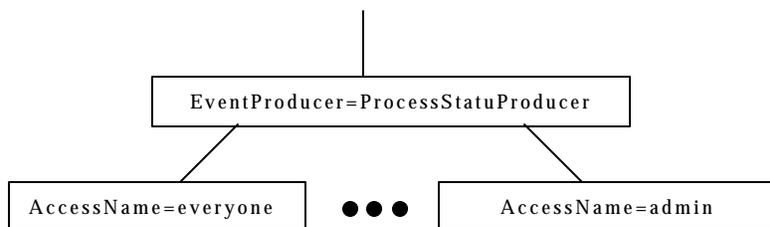


Figure 4. Event producer sub-tree in the directory information tree.

The next question to address is where should the sub-trees rooted at EventType instances and EventProducer instances should be placed? The EventType instances are not associated with any particular host or producer, but they are associated with an organization that defines them. This leads us to propose that EventType sub-trees be placed in the DIT under the organizations that define them. EventProducer instances are running on a host so they are associated with both a host and an organization. The EventProducer sub-trees can therefore be children of either host instances or organizational instances. We propose that they be children of host instances since other grid services (such as Globus GRAMs) are placed in the DIT as children of the host they are running on.

The final question that needs to be answered is should this monitoring information be placed in the general DIT with all of the other information (see Figure 2) or should there be a separate sub-tree for this information? This is a difficult question and we should see what groups with similar problems (such as the data grid) are doing. Are we going to have separate servers for the monitoring information? If so, then we can do whatever we want. If we are going to use the general DIT, then there will be issues to address. For example, obtaining authorization to write monitoring information to the servers and how much of an administrative hassle this will be for us and the administrators. This will probably be more difficult using the general GIS directory tree than our own specialized sub-tree. An example of how we could have a specialized monitoring sub-tree is shown in Figure 5. We believe that the better approach is not to have a specialized sub-tree, but to use the general DIT and those of us who want them can support and use specialized GIS servers to store event data.

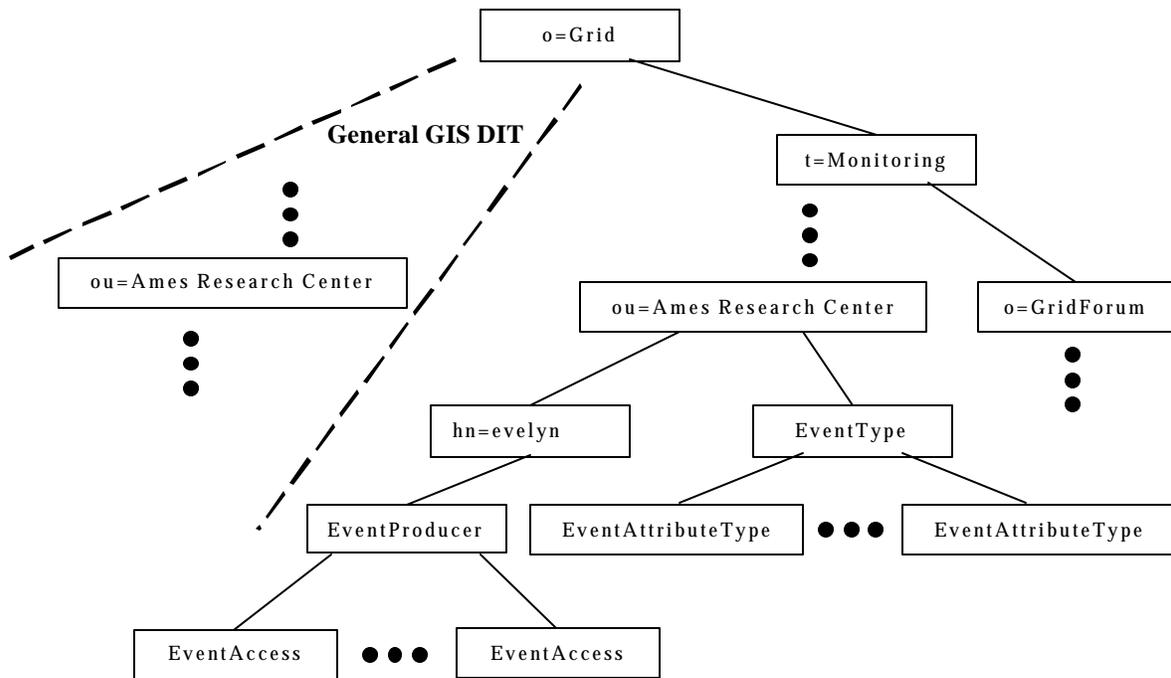


Figure 5. Directory Information Tree with a sub-tree for all monitoring information. This would allow there to be a hierarchy of servers just for event types.

References

- [1] S. Fitzgerald, I. Foster, C. Kesselman, G. v. Laszewski, W. Smith, and S. Tuecke, "A Directory Service for Configuring High-Performance Distributed Computations," presented at 6th IEEE International Symposium on High Performance Distributed Computing, 1997.
- [2] T. Howes and M. Smith, *LDAP: Programming Directory-Enabled Applications with Lightweight Directory Access Protocol*: Macmillan Technical Publishing, 1997.
- [3] T. Howes, M. Smith, and G. Good, *Understanding and Deploying LDAP Directory Services*: MacMillan Technical Publishing, 1999.