Analysis of the Globus Toolkit Grid Information Service

Giovanni Aloisio, Massimo Cafaro, Italo Epicoco, Sandro Fiore
HPCC, University of Lecce, Italy

Version 1.0

Abstract. This report is concerned with the analysis of current Globus Toolkit Grid Information Service (GIS) to assess its capabilities and limitations. A comparison of GIS capabilities and GridLab dynamic grid computing requirements will provide the foundations to build needed extensions.

1. Introduction

The grid computing model of computing [1] emerged recently as a new field distinguished from traditional distributed computing because of its focus on large-scale resource sharing and innovative high-performance applications. The grid infrastructure ties together a number of Virtual Organizations (VO) [2], that reflect dynamic collections of individuals, institutions and computational resources. Flexible, secure and coordinated resource sharing between VOs requires the availability of an information rich environment to support resource discovery and decision making processes. The Globus Toolkit [3] provides applications developers with the GIS, also called Metadata Directory Service (MDS) to support development of grid-aware applications, able to exploit the knowledge about current grid status and configuration to adapt themselves to changes in heterogeneous, dynamic environments.

Currently GIS infrastructure is based on two kind of Information Services, the Grid Resource Information Service (GRIS) and the Grid Index Information Service (GIIS). The GRIS supplies information about a specific resource, while the GIIS can be thought of as an aggregate directory
because a GIIS gathers information from multiple GRIS servers (belonging to the same VO or to multiple VOs) and supplies it as an aggregate collection of information. Thus the GIS is a hierarchical infrastructure with distributed services allowing for scalability and fault tolerance (a resource failure does not affect other resources).

The document is organized as follows. Section 2 recalls GIS requirements and Section 3 describes current approaches to GIS. Section 4 provides an overview of Globus GIS, Section 5 provides details about how the LDAP protocol is used in the Globus GIS and Section 6 recalls the Globus GIS schema. We present in Section 7 the results of our performance tests and conclude evaluating the Globus GIS in Section 8.

2. GIS requirements

As stated by the Global Grid Forum (GGF) [4] GIS Working group, a GIS infrastructure should provide:

- Efficient delivery of state information from a single source;
- Fault tolerance (component failure in grid environment is the rule);
- Distributed components for decentralized maintenance;
- Services that timestamp failures;
- Discovery and enquiry mechanisms;
- Robust authentication using Globus GSI;
- Timestamps and Time To Live (TTL) attributes for data objects.

The Globus GIS provide the following two protocols:

1. Enquiry protocol: Grid Resource Inquiry Protocol (GRIP)
2. Registration protocol: Grid Resource Registration Protocol (GRRP)

These two protocols foster interoperability between GIIS and GRIS servers. The GGF specification for GIS clearly states that GIS must provide support for Globus GRAM; a key point is that LDAP protocol must be implemented: it is mandatory to have an LDAP based query protocol. A GRIS server need to know the GIIS in which to register itself, and the GIIS itself provide a scope for searching (VO search scope).

The GRIP protocol is the core of the resource layer function of GIS; its security model must support GSI. It must supply a query function, must be deployed universally and should support distributed operations.

The GRRP protocol notifies one directory server about the existence of another; it is a Soft-state protocol (relationship by means of notifications, discarded after a long period of silence) and may be implemented on a variety of transport protocol (e.g. LDAP implementation coupled with SASL security).

It is worth noting here that timestamps and TTL estimates are not meaningful if a common timing mechanism is not available; thus the NTP protocol must be a mandatory resource level protocol inside a VO.
3. GIS approaches

Currently, there are two major approaches to GIS. The Globus one is based on a Directory infrastructure. Moreover a novel approach undertaken by a GGF WG will exploit a relational DBMS [5]. Also, a notification framework has been proposed, called the Grid Notification Framework (GNF) [6]. In this framework, periodic messages are dispatched by a declaring entity to one or more receiving entities.

4. The Globus GIS

The Globus GIS has been designed to achieve:

- Performance;
- Scalability;
- Security;
- Uniformity;
- Expressiveness;
- Extensibility;
- Multiple sources;
- Dynamic data;
- Flexible access;
- Deployability;
- Decentralized control.

Data representation and access in the Globus GIS are based on LDAP directory structure. The data model is able to encode the types of resources found in High Performance Distributed Systems, and the implementation reflect a set of strategies to meet requirements for performance, multiple data sources and scalability. LDAP directories are used to store and retrieve information. However, these are designed for information retrieval rather than for storing and updating information, and offer a static view of the data contained. Updates are simple without transactions. The LDAP directory service provide directories that can be accessed using a network protocol and include mechanisms for replication and data distribution through “referrals”.

The LDAP protocol is the network protocol for accessing the information stored inside the directories; the underlying information model defines the form and character of the information. Associated with LDAP is the concept of a namespace defining how information is referenced and organized. The Globus GIS is based on LDAP because it is an:

- Emerging distributed operation model defining how data may be distributed and referenced;
- Extensible protocol;
- Extensible information model.

The LDAP information model and namespace is based on entries. Entries store typed attributes, that can be single or multi-valued. Each entry has a Distinguished Name (DN), and the resulting hierarchy is called Directory Information Tree (DIT). Access to the data is based on the following:
1. **base DN** Where in the DIT to begin the search;
2. **filter** Specifies attribute types, assertion values and matching criteria;
3. **scope** Indicates how many levels of the DIT are searched, relative to the base DN.

---

5. **The use of LDAP in the Globus GIS**

There are three different phases, called respectively *Initialization*, *Population* and *Querying*. In the *Initialization* phase, an information schema that describes the attributes associated with the different entry classes is used. The specification for this information schema utilizes the Grid Object Specification language (GOS) [7]; the latest available version is GOS v3.2. Recently, two new objects have been specified using GOS, namely the *people* [8] and *software* [9] objects. An XML binding for GOS is also available, MDSML v1 [10]. The Resource Description Framework (RDF) has been proposed as an alternative syntax for describing Grid resource Metadata [11].

In the *Population* phase, a framework to populate the directory with information, two different strategies can be used: a set of UNIX shell scripts (local Information providers) is called by the OpenLDAP back-end or a set of loadable modules allowing provider implementations to execute within the server, avoiding the overhead of server-side process creation. The configured Information Providers generate the information corresponding to the objects in the information schema in LDIF (LDAP Data Interchange Format), so that it can be readily published in the GIS.

The *Querying* phase is a framework to request information from directory services. A variety of APIs is available for C/C++, Java, Perl, etc.

LDAP assumes that all the information within a DIT subtree is provided by a single Information provider, however the Globus GIS addresses the need for multiple Information providers. The client/server architecture is augmented with caching mechanisms for timely responses. In the Globus GIS, Information providers can be specified on a *per attribute* basis. Moreover, GIS object class metadata contain Time-To-Live for attribute values and the update scope of attributes.

The TTL enable caching as follows:

- a TTL of 0 indicates that the attribute value can not be cached;
- a TTL of -1 indicates that the data is constant;
- a positive TTL values determine the mount of time that the attribute value is allowed to be provided out of the cache before refreshing.

In the Globus GIS implementation the update scope of an attribute limits the readers of an updated attribute value; standard LDAP implantations are not limited this way. Update scopes available include:

- Process;
- Computation;
- Global.

Process scope attributes are accessible only within the same process as the writer, Computation scope attributes can be accessed by any process within a computation and finally Global scope attributes can be accessed from any node or process on the Grid.
6. The Globus GIS Schema

The GIS schema provides a unique naming schema for the object classes specified, a specification of the attributes that an object class must or may contain, a data type for each attribute and the information needed to match a filter against the attributes found in an entry. The schema explicitly allows for inheritance between object classes, so that complex hierarchical relationships between object classes are possible. The GIS schema also reports assigned object identifiers (OIDs) [12].

Object Identifiers, commonly referred to as OIDs, are unique identifiers assigned to objects. They are used to uniquely identify many different types of things, such as X.500 directory object and attribute types. In fact, just about everything in the X.500 directory system is identified by an OID. OIDs are also used to uniquely identify objects in other protocols. OIDs are written as strings of dotted decimal numbers. Each part of an OID represents a node in a hierarchical OID tree. This hierarchy allows an arbitrarily large number of objects to be named, and it supports delegation of the namespace. For example, all the user attribute types defined by the X.500 standards begin with 2.5.4. The \texttt{cn} attribute is assigned the OID 2.5.4.3, and the \texttt{sn} attribute is assigned the OID 2.5.4.4. An individual subtree of the OID tree is called an \textit{arc}. Individual arcs may be assigned to organizations, which can then further divide the arc into \textit{subarcs}, if so desired. GGF is in charge of assigning OIDs for grid computing protocols.

Current GIS Schema follows. We recall here that ?, + and * in this context means respectively: optional, one or more instances, zero or more instances.

Object class Mds
Attribute type Mds-validfrom
Attribute type Mds-validto
Attribute type Mds-keepto?

Object class MdsHost
Attribute type Mds-Host-hn

Object class MdsHostNetNode
Attribute type Mds-Host-NetNode-hn
Object class MdsHostNode
Attribute type Mds-Host-Node-name

Object class MdsHostNodeGroup
Attribute type Mds-Host-Node-Group-name

Object class MdsDevice
Attribute type Mds-Device-name

Object class MdsDeviceGroup
Attribute type Mds-Device-Group-name

Object class MdsSoftware
Attribute type Mds-Software-deployment
Object class MdsComputer
Attribute type Mds-Computer-isa+
Attribute type Mds-Computer-platform+

Object class MdsComputerTotal
Attribute type Mds-Computer-Total-nodeCount

Object class MdsOs
Attribute type Mds-Os-name+
Attribute type Mds-Os-release+
Attribute type Mds-Os-version+

Object class MdsCpu
Attribute type Mds-Cpu-vendor+
Attribute type Mds-Cpu-model+
Attribute type Mds-Cpu-version+
Attribute type Mds-Cpu-features*
Attribute type Mds-Cpu-speedMHz*

Object class MdsCpuSmp
Attribute type Mds-Cpu-Smp-size+

Object class MdsCpuCache
Attribute type Mds-Cpu-Cache-l1kB*
Attribute type Mds-Cpu-Cache-l1kB*
Attribute type Mds-Cpu-Cache-l1dkB*
Attribute type Mds-Cpu-Cache-l2kB*

Object class MdsCpuFree
Attribute type Mds-Cpu-Free-1minX100+
Attribute type Mds-Cpu-Free-5minX100+
Attribute type Mds-Cpu-Free-15minX100+

Object class MdsCpuTotal
Attribute type Mds-Cpu-Total-count
Object class MdsCpuTotalFree
Attribute type Mds-Cpu-Total-Free-1minX100
Attribute type Mds-Cpu-Total-Free-5minX100
Attribute type Mds-Cpu-Total-Free-15minX100

Object class MdsMemoryRam
Attribute type Mds-Memory-Ram-sizeMB+
Attribute type Mds-Memory-Ram-freeMB+

Object class MdsMemoryRamTotal
Attribute type Mds-Memory-Ram-Total-sizeMB
Attribute type Mds-Memory-Ram-Total-freeMB

Object class MdsMemoryVm
Attribute type Mds-Memory-Vm-sizeMB+
Attribute type Mds-Memory-Vm-freeMB+
Object class MdsMemoryVmTotal
Attribute type Mds-Memory-Vm-Total-sizeMB
Attribute type Mds-Memory-Vm-Total-freeMB

Object class MdsFs
Attribute type Mds-Fs-mount*
Attribute type Mds-Fs-sizeMB+
Attribute type Mds-Fs-freeMB+

Object class MdsFsTotal
Attribute type Mds-Fs-Total-count
Attribute type Mds-Fs-Total-sizeMB
Attribute type Mds-Fs-Total-freeMB

Object class MdsNet
Attribute type Mds-Net-name+
Attribute type Mds-Net-addr+
Attribute type Mds-Net-netaddr+
Attribute type Mds-Net-mtuB+

Object class MdsNetTotal
Attribute type Mds-Net-Total-count

Object class MdsService
Attribute type Mds-Service-type
Attribute type Mds-Service-protocol+
Attribute type Mds-Service-hn
Attribute type Mds-Service-port

Object class MdsServiceLdap
Attribute type Mds-Service-Ldap-suffix+
Attribute type Mds-Service-Ldap-timeout*
Attribute type Mds-Service-Ldap-sizelimit*

GIS object classes and their attribute types are reported in Appendix A.

The GIS object classes provide both static and dynamic information about the configuration and status of computational resources. However, other useful information for grid-aware applications is missing. As an example it would be highly desirable to access information about network parameters like bandwidth and latency between sites, or information related to software installed. This requires developing new Information Providers, applications that generate additional information to be published inside a GRIS server in LDIF format.

The GIS schema, by default is not used by GRIS servers to check for data consistency. Attribute types are correctly handled in current GIS implementation, due to the presence of data definition types: attribute syntax now includes int and float for numerical quantities instead of cis (case insensitive string).

The distributed model of operations is such that every computing resource runs a GRIS server and, by default, also a GIIS server. However, for simple production grids, only one GIIS server will be
chosen as the aggregate directory. The GRIS servers will be configured to register themselves with the GIIS. Registration messages are sent every 5 minutes using the GRRP protocol. Advanced grid environments can exploit complex topologies with hierarchical GIIS servers.

7. Performance of Globus GIS

To evaluate the performance of GIS infrastructure we did the following tests. Two applications were developed to contact a GIIS server and to query it using the Globus APIs. The two applications only differ in how they bind to the GIIS server. The former uses anonymous binding, the latter uses Simple Authentication and Security Layer (SASL) binding through GSI-GSS API, i.e. we wanted to evaluate not just how much time does it take to issue a query and get the server answer, but also how security mechanisms affect the GIS performance. To test the GIIS using SASL we configured properly our GIIS server installing a required Globus certificate needed for mutual authentication. It is worth noting here that the caching mechanism used by the GIS is also evident from our results.

We choose not to use the ldapsearch tool available in the Globus distribution because using it would have resulted in timings affected by an unnecessary overhead (initial setup, parameter options parsing etc) and also because real grid-enabled applications will exploit directly the Globus APIs.

To take into account the caching mechanisms, we proceeded issuing a first set of 20 queries to our GIIS server using the base scope, then we waited for 20 minutes in order for the GIIS cache to expire (by default the expiry period for almost all the information providers is set to 1 or 15 minutes; there are three information providers for which the cache expires after 12 hours, so that the information provided was considered as always available). Once again a new set of 20 queries was issued, this time with one as scope. After 20 minutes a final set of 20 queries was issued using the sub scope.

The experiment was repeated several times and during the day (peak time for computing resource with high workload) and late during the night (off-peak time, low workload), both for the anonymous and the SASL client applications the results were averaged. Here peak time refers to the time period from 9.00 am to 7.00 pm, and off-peak time refers to the time period from 7.00 pm to 9.00 am. Because during peak time a number of users is actually using the GIIS machine, we expect the workload on the machine to be “high”. Conversely, during off-peak time, the machine workload should be relatively “low”.

The configuration used for testing at the University of Lecce is a computational grid including seven machines running Linux with kernel 2.4.x; one of these resources is located in Hungary, so that we can actually simulate a very small Virtual Organization. The GIIS server is powered by an AMD Athlon 1 GHz with 256 MB of RAM.

Results are reported in the following figures (graphs are drawn using logarithmic scale). For each graph we also report the minimum, maximum and average values. As expected, the first query in each set takes significantly more time, due to the cache empty; also expected is the increase in response time when querying the GIIS with scope going from base to one and finally to sub or binding using SASL. The graphs also show that the performance of SASL binding during peak and off-peak time are almost the same.
Fig. 1

Peak Time - Anonymous Binding
sara.unile.it

MilliseCS

Scope BASE
Scope ONE
Scope SUB

Min Max Avg
Scope BASE 6 6393 326.9
Scope ONE 22 6126 329.1
Scope SUB 70 6631 401.95

Fig. 2

Peak Time - SASL Binding
sara.unile.it

MilliseCS

Scope BASE
Scope ONE
Scope SUB

Min Max Avg
Scope BASE 40 6466 364.4
Scope ONE 57 6199 368
Scope SUB 110 6887 452.9
Off-Peak Time - Anonymous Binding
sara.unile.it

![Graph showing performance metrics for three scopes: BASE, ONE, SUB.](image)

<table>
<thead>
<tr>
<th></th>
<th>Min</th>
<th>Max</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scope BASE</td>
<td>7</td>
<td>5047</td>
<td>259.85</td>
</tr>
<tr>
<td>Scope ONE</td>
<td>22</td>
<td>4544</td>
<td>249.9</td>
</tr>
<tr>
<td>Scope SUB</td>
<td>68</td>
<td>4620</td>
<td>299.4</td>
</tr>
</tbody>
</table>

Fig. 3

Off-Peak Time - SASL Binding
sara.unile.it

![Graph showing performance metrics for three scopes: BASE, ONE, SUB.](image)

<table>
<thead>
<tr>
<th></th>
<th>Min</th>
<th>Max</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scope BASE</td>
<td>41</td>
<td>5840</td>
<td>332.4</td>
</tr>
<tr>
<td>Scope ONE</td>
<td>55</td>
<td>5547</td>
<td>335.8</td>
</tr>
<tr>
<td>Scope SUB</td>
<td>110</td>
<td>5745</td>
<td>396.75</td>
</tr>
</tbody>
</table>

Fig. 4
The same experiment was repeated querying the following GIIS servers: lhun.ics.muni.cz in Czech and mishra.lpds.sztaki.hu in Hungary and, due to the configuration, only anonymous binding was possible. Moreover, no GRIS servers were registered with these GIIS servers. These machines are both running Linux with kernel 2.4.x; lhun.ics.muni.cz is powered by a Pentium 4 1.5 GHz with 256 MB of RAM; mishra.lpds.sztaki.hu is powered by a Cyrix 167 MHz with 128 MB of RAM. Results are reported in the following figures (graphs are drawn using logarithmic scale). For each graph we also report the minimum, maximum and average values. The behaviour of remote queries is almost the same as for the queries to the local GIIS server (sara.unile.it); the time needed to actually search a GIIS server (i.e., when the cache has expired), both local or remote, is an order of magnitude higher than the time needed to retrieve the information on the network, so that we can not really appreciate the fact that we are querying a remote GIIS server. When the cache has not expired instead, a local query takes significantly less time than a remote query. This is because in this case, the time needed for the query is much less than the time needed to retrieve the information, as shown in Figures 9 and 10 that present a comparison between the three GIIS servers.
Peak Time - Anonymous Binding
lhun.ics.muni.cz

![Graph showing peak time anonymous binding performance metrics](image)

<table>
<thead>
<tr>
<th>Scope</th>
<th>Min</th>
<th>Max</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scope BASE</td>
<td>149</td>
<td>4802</td>
<td>449.45</td>
</tr>
<tr>
<td>Scope ONE</td>
<td>185</td>
<td>4952</td>
<td>470.3</td>
</tr>
<tr>
<td>Scope SUB</td>
<td>279</td>
<td>5047</td>
<td>608.85</td>
</tr>
</tbody>
</table>

Fig. 5

Off-Peak Time - Anonymous Binding
lhun.ics.muni.cz

![Graph showing off-peak time anonymous binding performance metrics](image)

<table>
<thead>
<tr>
<th>Scope</th>
<th>Min</th>
<th>Max</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scope BASE</td>
<td>389</td>
<td>1085</td>
<td>485.75</td>
</tr>
<tr>
<td>Scope ONE</td>
<td>298</td>
<td>1215</td>
<td>537.85</td>
</tr>
<tr>
<td>Scope SUB</td>
<td>555</td>
<td>3479</td>
<td>900.5</td>
</tr>
</tbody>
</table>

Fig. 6
Fig. 7

<table>
<thead>
<tr>
<th>Scope</th>
<th>Min</th>
<th>Max</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scope BASE</td>
<td>329</td>
<td>4810</td>
<td>644.4</td>
</tr>
<tr>
<td>Scope ONE</td>
<td>536</td>
<td>5099</td>
<td>939</td>
</tr>
<tr>
<td>Scope SUB</td>
<td>485</td>
<td>4984</td>
<td>974.65</td>
</tr>
</tbody>
</table>

Fig. 8

<table>
<thead>
<tr>
<th>Scope</th>
<th>Min</th>
<th>Max</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scope BASE</td>
<td>189</td>
<td>4492</td>
<td>427.5</td>
</tr>
<tr>
<td>Scope ONE</td>
<td>264</td>
<td>4583</td>
<td>526.15</td>
</tr>
<tr>
<td>Scope SUB</td>
<td>373</td>
<td>4629</td>
<td>657.4</td>
</tr>
</tbody>
</table>
Peak Time - Anonymous Binding
Scope SUB

<table>
<thead>
<tr>
<th></th>
<th>Min</th>
<th>Max</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>sara.unile.it</td>
<td>70</td>
<td>6631</td>
<td>401.95</td>
</tr>
<tr>
<td>mishra.lpds.sztaki.hu</td>
<td>485</td>
<td>4984</td>
<td>974.65</td>
</tr>
<tr>
<td>lhun.ics.muni.cz</td>
<td>279</td>
<td>5047</td>
<td>608.85</td>
</tr>
</tbody>
</table>

Off-Peak Time - Anonymous Binding
Scope SUB

<table>
<thead>
<tr>
<th></th>
<th>Min</th>
<th>Max</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>sara.unile.it</td>
<td>68</td>
<td>4620</td>
<td>299.4</td>
</tr>
<tr>
<td>mishra.lpds.sztaki.hu</td>
<td>373</td>
<td>4629</td>
<td>657.4</td>
</tr>
<tr>
<td>lhun.ics.muni.cz</td>
<td>555</td>
<td>3479</td>
<td>900.5</td>
</tr>
</tbody>
</table>

Fig. 9

Fig. 10
8. Evaluation of Globus GIS and conclusions

Directory services, as pointed out by the GGF Relational GIS Working Group are not able to handle a growing number of data objects, especially when complex relationships between data objects exist. In particular, frequent updates to objects and their relationships are challenging. Another limitations of directory services is the lack of sophisticated queries and the ability of querying data streams. The directory service query language is both limited and restricted because it is essentially a procedural language, not a declarative one. This forces the user to know in advance explicitly the tree structure. However, the preceding remarks does not apply to the GridLab GIS we are going to provide for the following reasons.

1. GridLab GIS will handle a small number of data objects providing the information needed for dynamic grid computing scenarios we envision.
2. These data object will not be related by complex relationships; we expect them to be almost independent.
3. We do not depend on sophisticated queries; the LDAP filtering mechanism augmented with a small amount of additional post-processing suffices for GridLab purposes.
4. Data stream information providers, if needed (this will be evaluated in WP 10 Task 2), will be specified, implemented and tested to assess their performance; we defer this task to WP 10 next activities. It is worth noting here that, in case, data stream information providers can be queried directly instead of integrated inside GIS infrastructure, allowing for a complimentary approach.
5. Current GIS performance, as stated by our results, demonstrates the feasibility of the Globus LDAP based GIS.
6. Even though the LDAP query language is procedural in nature, we believe that knowing in advance the DIT structure is not a limitation. Indeed, a schema exist to fulfil this requirement, so that extending the Globus GIS schema to include additional information and making it publicly available solves the problem.

The Globus GIS infrastructure was evaluated to assess its capabilities and limitations. Extensive review and tests lead us to the conclusion that the Globus GIS is suitable as a base Information Service infrastructure, due to its features like security, performance, extensibility, scalability deployability and decentralized maintenance. Finally, GIS also offers flexible access through both available tools and APIs.

Acknowledgements

The authors gratefully acknowledge the help of the following people: Ewa Deelman, Zoltan Balaton and Petr Holub.
References


Appendix A

The Globus GIS schema

Object class Mds (OID 1.3.6.1.4.1.3536.2.6) represents the abstract root of all MDS data types requiring dynamic metadata.

Attribute type Mds-validfrom (OID 1.3.6.1.4.1.3536.2.6.0.1 from class Mds) is a required single-valued attribute, using attribute syntax gentime, and represents the global time at which the object and its information is first valid.

Attribute type Mds-validto (OID 1.3.6.1.4.1.3536.2.6.0.2 from class Mds) is a required single-valued attribute, using attribute syntax gentime, and represents the global time at which the object information is no longer valid.

Attribute type Mds-keepto (OID 1.3.6.1.4.1.3536.2.6.0.3 from class Mds) is an optional single-valued attribute, using attribute syntax gentime, and represents the global time at which the object should be deleted. The purge time should always be later than the expiration time, if it exists. The existence of a purge time hints that some information in the object may be useful even when the object is invalid, i.e. that the invalid object is better than no information at all.

Object class MdsHost (OID 1.3.6.1.4.1.3536.2.6.1.1) inherits Mds and represents a networked computer and its various service access points, e.g. a workstation, SMP parallel system, or the front-end server for a distributed parallel machine.

Attribute type Mds-Host-hn (OID 1.3.6.1.4.1.3536.2.6.1.1.0.1 from class MdsHost) is the required single-valued RDN attribute, using attribute syntax cis, and contains the fully-qualified public hostname for a networked computer.

Object class MdsHostNetNode (OID 1.3.6.1.4.1.3536.2.6.1.1.1) inherits Mds and represents a networked computing element of a distributed parallel machine, e.g. a node that has a public network interface.

Attribute type Mds-Host-NetNode-hn (OID 1.3.6.1.4.1.3536.2.6.1.1.1.0.1 from class MdsHostNetNode) is the required single-valued RDN attribute, using attribute syntax cis, and contains the fully-qualified public hostname for a networked computing element, e.g. the public hostname of a compute node.

Object class MdsHostNode (OID 1.3.6.1.4.1.3536.2.6.1.1.2) inherits Mds and represents a hidden computing element of a distributed parallel machine, e.g. a node that does not have a public network interface.

Attribute type Mds-Host-Node-name (OID 1.3.6.1.4.1.3536.2.6.1.1.2.0.1 from class MdsHostNode) is the required single-valued RDN attribute, using attribute syntax cis, and contains the locally unique name of a hidden computing element within a service or management scope, e.g. a node label or number within a distributed parallel machine.

Object class MdsHostNodeGroup (OID 1.3.6.1.4.1.3536.2.6.1.1.2.1) inherits Mds and represents a collection of computing elements, e.g. a set of nodes that are designated for a particular operational role.

Attribute type Mds-Host-Node-Group-name (OID 1.3.6.1.4.1.3536.2.6.1.1.2.1.0.1 from class MdsHostNodeGroup) is the required single-valued RDN attribute, using attribute syntax cis, and contains the locally unique name of a group of networked computing elements within a service or management scope, e.g. a compute partition within a distributed parallel machine.
Object class MdsDevice (OID 1.3.6.1.4.1.3536.2.6.1.2) inherits Mds and represents a local user-visible physical or logical device on a computing element, e.g. a cpu, filesystem, or network interface.

Attribute type Mds-Device-name (OID 1.3.6.1.4.1.3536.2.6.1.2.0.1 from class MdsDevice) is the required single-valued RDN attribute, using attribute syntax cis, and contains a locally unique name for a device on a computing element.

Object class MdsDeviceGroup (OID 1.3.6.1.4.1.3536.2.6.1.2.1) inherits Mds and represents a group of related local devices, organized for conveniently scoped searches.

Attribute type Mds-Device-Group-name (OID 1.3.6.1.4.1.3536.2.6.1.2.1.0.1 from class MdsDeviceGroup) is the required single-valued RDN attribute, using attribute syntax cis, and contains a locally unique name for a group of devices on a computing element.

Object class MdsSoftware (OID 1.3.6.1.4.1.3536.2.6.1.3) inherits Mds and represents an instance of installed (deployed) software on a computing platform, e.g. a particular version of software available for use.

Attribute type Mds-Software-deployment (OID 1.3.6.1.4.1.3536.2.6.1.3.0.1 from class MdsSoftware) is the required single-valued RDN attribute, using attribute syntax cis, and contains a locally unique name for an instance of deployed software.

Object class MdsComputer (OID 1.3.6.1.4.1.3536.2.6.2.1) inherits Mds and represents auxiliary information about a shared-memory computing element (standalone or part of a distributed parallel computer).

Attribute type Mds-Computer-isa (OID 1.3.6.1.4.1.3536.2.6.2.1.0.1 from class MdsComputer) is a required multi-valued attribute, using attribute syntax cis, and informally names the Instruction Set Architecture (ISA) of the computing element.

Attribute type Mds-Computer-platform (OID 1.3.6.1.4.1.3536.2.6.2.1.0.2 from class MdsComputer) is a required multi-valued attribute, using attribute syntax cis, and informally describes the platform type of the computing element.

Object class MdsComputerTotal (OID 1.3.6.1.4.1.3536.2.6.2.1.1) inherits Mds and represents summary information about one or more shared-memory computing elements.

Attribute type Mds-Computer-Total-nodeCount (OID 1.3.6.1.4.1.3536.2.6.2.1.1.0.3 from class MdsComputerTotal) is a required single-valued attribute, using attribute syntax int, and indicates the number of computing elements being represented.

Object class MdsOs (OID 1.3.6.1.4.1.3536.2.6.2.3) inherits Mds and represents auxiliary information about the installed operating system (OS).

Attribute type Mds-Os-name (OID 1.3.6.1.4.1.3536.2.6.2.3.0.1 from class MdsOs) is a required multi-valued attribute, using attribute syntax cis, and informally names the OS using a vendor-specific convention.

Attribute type Mds-Os-release (OID 1.3.6.1.4.1.3536.2.6.2.3.0.2 from class MdsOs) is a required multi-valued attribute, using attribute syntax cis, and informally names the OS release using a vendor-specific convention.

Attribute type Mds-Os-version (OID 1.3.6.1.4.1.3536.2.6.2.3.0.3 from class MdsOs) is a required multi-valued attribute, using attribute syntax cis, and informally names the OS or kernel version using a vendor-specific convention.

Object class MdsCpu (OID 1.3.6.1.4.1.3536.2.6.2.4) inherits Mds and represents auxiliary information about general-purpose processors (CPUs) in an SMP computing element.

Attribute type Mds-Cpu-vendor (OID 1.3.6.1.4.1.3536.2.6.2.4.0.1 from class MdsCpu) is a required multi-valued attribute, using attribute syntax cis, and informally names the CPU vendor.
Attribute type Mds-Cpu-model (OID 1.3.6.1.4.1.3536.2.6.2.4.0.2 from class MdsCpu) is a required multi-valued attribute, using attribute syntax cis, and informally names the CPU model.

Attribute type Mds-Cpu-version (OID 1.3.6.1.4.1.3536.2.6.2.4.0.3 from class MdsCpu) is a required multi-valued attribute, using attribute syntax cis, and informally names the CPU version or stepping.

Attribute type Mds-Cpu-features (OID 1.3.6.1.4.1.3536.2.6.2.4.0.4 from class MdsCpu) is an optional multi-valued attribute, using attribute syntax cis, and informally names optional CPU features.

Attribute type Mds-Cpu-speedMHz (OID 1.3.6.1.4.1.3536.2.6.2.4.0.5 from class MdsCpu) is an optional multi-valued attribute, using attribute syntax int, and indicates the clock speed of a CPU.

Object class MdsCpuSmp (OID 1.3.6.1.4.1.3536.2.6.2.4.1) inherits Mds and

Attribute type Mds-Cpu-Smp-size (OID 1.3.6.1.4.1.3536.2.6.2.4.1.0.1 from class MdsCpuSmp) is a required multi-valued attribute, using attribute syntax int, and indicates the number of CPUs in an SMP configuration.

Object class MdsCpuCache (OID 1.3.6.1.4.1.3536.2.6.2.4.2) inherits Mds and

Attribute type Mds-Cpu-Cache-l1kB (OID 1.3.6.1.4.1.3536.2.6.2.4.2.0.1 from class MdsCpuCache) is an optional multi-valued attribute, using attribute syntax int, and indicates first-level unified cache size (in kilo-bytes) of a CPU.

Attribute type Mds-Cpu-Cache-l1ikB (OID 1.3.6.1.4.1.3536.2.6.2.4.2.0.2 from class MdsCpuCache) is an optional multi-valued attribute, using attribute syntax int, and indicates first-level instruction cache size (in kilo-bytes) of a CPU.

Attribute type Mds-Cpu-Cache-l1dkB (OID 1.3.6.1.4.1.3536.2.6.2.4.2.0.3 from class MdsCpuCache) is an optional multi-valued attribute, using attribute syntax int, and indicates first-level data cache size (in kilo-bytes) of a CPU.

Attribute type Mds-Cpu-Cache-l2kB (OID 1.3.6.1.4.1.3536.2.6.2.4.2.0.4 from class MdsCpuCache) is an optional multi-valued attribute, using attribute syntax int, and indicates second-level unified cache size (in kilo-bytes) of a CPU.

Object class MdsCpuFree (OID 1.3.6.1.4.1.3536.2.6.2.4.3) inherits Mds and

Attribute type Mds-Cpu-Free-1minX100 (OID 1.3.6.1.4.1.3536.2.6.2.4.3.0.1 from class MdsCpuFree) is a required multi-valued attribute, using attribute syntax float, and indicates the 1-minute average processor availability for an SMP computing element, which is the difference between the available CPUs and the average runnable task count during that time.

Attribute type Mds-Cpu-Free-5minX100 (OID 1.3.6.1.4.1.3536.2.6.2.4.3.0.2 from class MdsCpuFree) is a required multi-valued attribute, using attribute syntax float, and indicates the 5-minute average processor availability for an SMP computing element, which is the difference between the available CPUs and the average runnable task count during that time.

Attribute type Mds-Cpu-Free-15minX100 (OID 1.3.6.1.4.1.3536.2.6.2.4.3.0.3 from class MdsCpuFree) is a required multi-valued attribute, using attribute syntax float, and indicates the 15-minute average processor availability for an SMP computing element, which is the difference between the available CPUs and the average runnable task count during that time.

Object class MdsCpuTotal (OID 1.3.6.1.4.1.3536.2.6.2.4.4) inherits Mds and represents summary information about one or more SMP computing elements.

Attribute type Mds-Cpu-Total-count (OID 1.3.6.1.4.1.3536.2.6.2.4.4.0.1 from class MdsCpuTotal) is a required single-valued attribute, using attribute syntax int, and indicates the total number of CPUs in all computing elements.

Object class MdsCpuTotalFree (OID 1.3.6.1.4.1.3536.2.6.2.4.4.2) inherits Mds and
Attribute type Mds-Cpu-Total-Free-1minX100 (OID 1.3.6.1.4.1.3536.2.6.2.4.4.2.0.1 from class MdsCpuTotalFree) is a required single-valued attribute, using attribute syntax float, and indicates the 1-minute average processor availability for all computing elements, which is the difference between the available CPUs and the average runnable task count during that time.

Attribute type Mds-Cpu-Total-Free-5minX100 (OID 1.3.6.1.4.1.3536.2.6.2.4.4.2.0.2 from class MdsCpuTotalFree) is a required single-valued attribute, using attribute syntax float, and indicates the 5-minute average processor availability for all computing elements, which is the difference between the available CPUs and the average runnable task count during that time.

Attribute type Mds-Cpu-Total-Free-15minX100 (OID 1.3.6.1.4.1.3536.2.6.2.4.4.2.0.3 from class MdsCpuTotalFree) is a required single-valued attribute, using attribute syntax float, and indicates the 15-minute average processor availability for all computing elements, which is the difference between the available CPUs and the average runnable task count during that time.

Object class MdsMemoryRam (OID 1.3.6.1.4.1.3536.2.6.2.5.1.1) inherits Mds and represents the physical random-access memory (RAM) in a computing element.

Attribute type Mds-Memory-Ram-sizeMB (OID 1.3.6.1.4.1.3536.2.6.2.5.1.1.0.1 from class MdsMemoryRam) is a required multi-valued attribute, using attribute syntax int, and represents the configured RAM size (in mega-bytes).

Attribute type Mds-Memory-Ram-freeMB (OID 1.3.6.1.4.1.3536.2.6.2.5.1.1.0.2 from class MdsMemoryRam) is a required multi-valued attribute, using attribute syntax int, and represents the unallocated RAM size (in mega-bytes).

Object class MdsMemoryRamTotal (OID 1.3.6.1.4.1.3536.2.6.2.5.1.1.1) inherits Mds and represents summary information about the physical random-access memory in one or more computing elements.

Attribute type Mds-Memory-Ram-Total-sizeMB (OID 1.3.6.1.4.1.3536.2.6.2.5.1.1.1.0.1 from class MdsMemoryRamTotal) is a required single-valued attribute, using attribute syntax int, and represents the total configured RAM size (in mega-bytes) for one or more computing elements.

Attribute type Mds-Memory-Ram-Total-freeMB (OID 1.3.6.1.4.1.3536.2.6.2.5.1.1.1.0.2 from class MdsMemoryRamTotal) is a required single-valued attribute, using attribute syntax int, and represents the total unallocated RAM size (in mega-bytes) for one or more computing elements.

Object class MdsMemoryVm (OID 1.3.6.1.4.1.3536.2.6.2.5.1.2) inherits Mds and represents the disk-based virtual memory (VM) in a computing element.

Attribute type Mds-Memory-Vm-sizeMB (OID 1.3.6.1.4.1.3536.2.6.2.5.1.2.0.1 from class MdsMemoryVm) is a required multi-valued attribute, using attribute syntax int, and represents the configured VM size (in mega-bytes).

Attribute type Mds-Memory-Vm-freeMB (OID 1.3.6.1.4.1.3536.2.6.2.5.1.2.0.2 from class MdsMemoryVm) is a required multi-valued attribute, using attribute syntax int, and represents the unallocated VM size (in mega-bytes).

Object class MdsMemoryVmTotal (OID 1.3.6.1.4.1.3536.2.6.2.5.1.2.1) inherits Mds and

Attribute type Mds-Memory-Vm-Total-sizeMB (OID 1.3.6.1.4.1.3536.2.6.2.5.1.2.1.0.1 from class MdsMemoryVmTotal) is a required single-valued attribute, using attribute syntax int, and represents the total configured VM size (in mega-bytes) for one or more computing elements.

Attribute type Mds-Memory-Vm-Total-freeMB (OID 1.3.6.1.4.1.3536.2.6.2.5.1.2.1.0.2 from class MdsMemoryVmTotal) is a required single-valued attribute, using attribute syntax int, and represents the total unallocated VM size (in mega-bytes) for one or more computing elements.

Object class MdsFs (OID 1.3.6.1.4.1.3536.2.6.2.5.2) inherits Mds and represents a information about a configured filesystem in a computing element.
Attribute type Mds-Fs-mount (OID 1.3.6.1.4.1.3536.2.6.2.5.2.0.1 from class MdsFs) is an optional multi-valued attribute, using attribute syntax cis, and indicates the mounted pathname to the filesystem.

Attribute type Mds-Fs-sizeMB (OID 1.3.6.1.4.1.3536.2.6.2.5.2.0.2 from class MdsFs) is a required multi-valued attribute, using attribute syntax int, and represents the formatted filesystem size in mega-bytes.

Attribute type Mds-Fs-freeMB (OID 1.3.6.1.4.1.3536.2.6.2.5.2.0.3 from class MdsFs) is a required multi-valued attribute, using attribute syntax int, and represents the unallocated filesystem size in mega-bytes.

Object class MdsFsTotal (OID 1.3.6.1.4.1.3536.2.6.2.5.2.1) inherits Mds and represents summary information about the filesystems in one or more computing elements.

Attribute type Mds-Fs-Total-count (OID 1.3.6.1.4.1.3536.2.6.2.5.2.1.0.1 from class MdsFsTotal) is a required single-valued attribute, using attribute syntax int, and indicates the total number of filesystems being reported.

Attribute type Mds-Fs-Total-sizeMB (OID 1.3.6.1.4.1.3536.2.6.2.5.2.1.0.2 from class MdsFsTotal) is a required single-valued attribute, using attribute syntax int, and represents the total formatted filesystem size (in mega-bytes) for all filesystems.

Attribute type Mds-Fs-Total-freeMB (OID 1.3.6.1.4.1.3536.2.6.2.5.2.1.0.3 from class MdsFsTotal) is a required single-valued attribute, using attribute syntax int, and represents the total unallocated filesystem size (in mega-bytes) for all filesystems.

Object class MdsNet (OID 1.3.6.1.4.1.3536.2.6.2.6.1) inherits Mds and represents information about network interfaces in a computing element.

Attribute type Mds-Net-name (OID 1.3.6.1.4.1.3536.2.6.2.6.1.0.1 from class MdsNet) is a required multi-valued attribute, using attribute syntax cis, and informally names a network interface.

Attribute type Mds-Net-addr (OID 1.3.6.1.4.1.3536.2.6.2.6.1.0.2 from class MdsNet) is a required multi-valued attribute, using attribute syntax cis, and indicates the IP address of a network interface.

Attribute type Mds-Net-netaddr (OID 1.3.6.1.4.1.3536.2.6.2.6.1.0.3 from class MdsNet) is a required multi-valued attribute, using attribute syntax cis, and indicates the IP network address associated with a network interface.

Attribute type Mds-Net-mtuB (OID 1.3.6.1.4.1.3536.2.6.2.6.1.0.4 from class MdsNet) is a required multi-valued attribute, using attribute syntax int, and indicates the maximum transmission unit size (in bytes) associated with a network interface.

Object class MdsNetTotal (OID 1.3.6.1.4.1.3536.2.6.2.6.1.1) inherits Mds and represents summary information for all network interfaces a computing element.

Attribute type Mds-Net-Total-count (OID 1.3.6.1.4.1.3536.2.6.2.6.1.1.0.1 from class MdsNetTotal) is a required single-valued attribute, using attribute syntax int, and indicates the total number of network interfaces being reported.

Object class MdsService (OID 1.3.6.1.4.1.3536.2.6.2.7.1) inherits Mds and represents a network-accessible service in a way that can be discovered by remote clients.

Attribute type Mds-Service-type (OID 1.3.6.1.4.1.3536.2.6.2.7.1.0.1 from class MdsService) is a required single-valued attribute, using attribute syntax cis, and represents the protocol that is used by the service, e.g. the mechanism portion of the service URL.

Attribute type Mds-Service-protocol (OID 1.3.6.1.4.1.3536.2.6.2.7.1.0.2 from class MdsService) is a required multi-valued attribute, using attribute syntax cis, and represents the protocol version(s) that is(are) used by the service.
Attribute type Mds-Service-hn (OID 1.3.6.1.4.1.3536.2.6.2.7.1.0.2 from class MdsService) is a required single-valued attribute, using attribute syntax cid, and represents the FQDN hostname of the host running the service, e.g. the host part of the service URL.

Attribute type Mds-Service-port (OID 1.3.6.1.4.1.3536.2.6.2.7.1.0.3 from class MdsService) is a required single-valued attribute, using attribute syntax int, and represents the TCP port number of the service, e.g. the port number part fo the service URL.

Object class MdsServiceLdap (OID 1.3.6.1.4.1.3536.2.6.2.7.1.1) inherits Mds and represents a service accessible by the LDAP protocol.

Attribute type Mds-Service-Ldap-suffix (OID 1.3.6.1.4.1.3536.2.6.2.7.1.1.0.1 from class MdsServiceLdap) is a required multi-valued attribute, using attribute syntax cid, and represents the root(s) of the logical service(s) in the LDAP server DIT.

Attribute type Mds-Service-Ldap-timeout (OID 1.3.6.1.4.1.3536.2.6.2.7.1.1.0.2 from class MdsServiceLdap) is an optional multi-valued attribute, using attribute syntax cid, and suggests a maximum LDAP operation timeout during normal function.

Attribute type Mds-Service-Ldap-sizelimit (OID 1.3.6.1.4.1.3536.2.6.2.7.1.1.0.3 from class MdsServiceLdap) is an optional multi-valued attribute, using attribute syntax cid, and suggests a maximum LDAP result size during normal function.