



NextGRID SLA Management Use Cases

Editors:	Bastian Koller	HLRS
	David Snelling	FLE
	Peer Hasselmeyer	NEC
	Kostas Tserpes	NTUA

Date	Author	Comments	Version	Status
07/09/2006	BK	First draft, missing parts	0.1	draft
10/05/2007	DS	Second draft	0.2	draft
31/07/2007	PH	Updated negotiation phase	0.3	draft
15/08/2007	PH	Added figures + description	0.4	draft
19/10/2007	BK	Update after internal review	0.5	draft
25/01/2008	KT	Inserted Evaluation special case	0.6	draft
27/02/2008	DS	Synced this version 0.6 with the SVN version 0.6 and reviewed the combined document.	0.7	draft
04/03/2008	KT	Followed DS guidelines for updating Section 2.2.1	0.8	Draft



- 1 INTRODUCTION..... 3**
- 2 USE CASES..... 4**
 - 2.1 Publishing and Discovery 5**
 - 2.2 Negotiation and Agreement..... 6**
 - 2.1.1 Evaluation..... 8
 - 2.2 Operation 9**
 - 2.2 Decommissioning..... 12**
- 3 REFERENCES..... 12**



1 Introduction

This document describes the basic set of use cases for Service Level Agreement (SLA) management in NextGRID. Along with the NextGRID SLA Profile document, it forms part of the NextGRID Generalised Specification for SLA management on a NextGRID compliant Grid.

Since NextGRID is fit for business, where guaranteed levels of service are the key factor, SLAs have become a key-architectural principle of the project. Thus, project partners have applied their experiences in using SLAs with customers, along with input from within the Grid community, to develop a machine-readable representation of an SLA as an XML schema and related management capabilities.

SLAs will help give customers confidence in the robustness, reliability, security and performance of the available services, while allowing providers to operate those services in an efficient and ultimately profitable manner. This must happen in a heterogeneous, loosely coupled and service oriented environment made up from multiple organisations, each owning and operating a segment of the infrastructure.

SLAs in NextGRID are managed based on the SLA lifecycle as defined by the TeleManagement Forum [1] and shown in Figure 1.

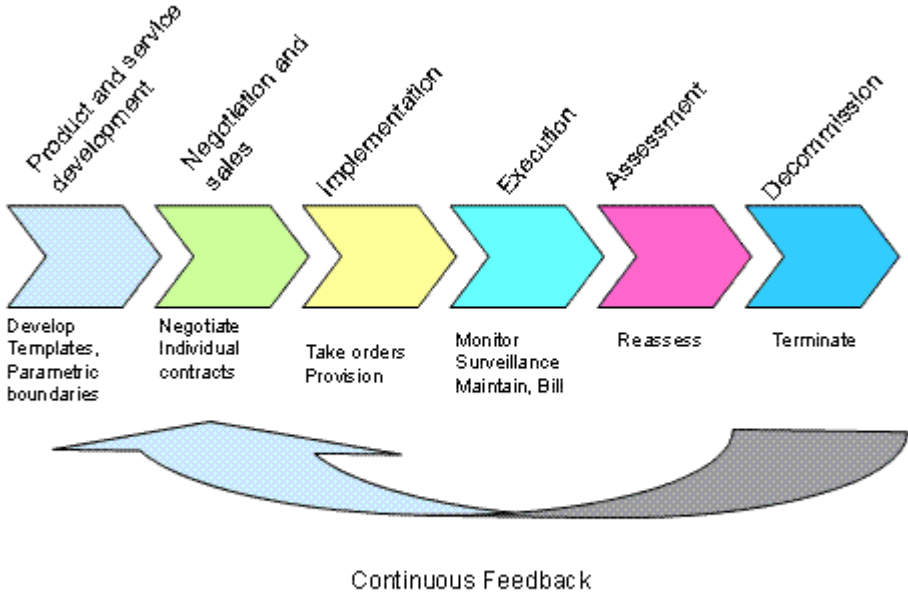


Figure 1 – The SLA lifecycle.



Figure 1 shows that the end-to-end lifecycle of an SLA consists of six stages. For the purposes of SLA management in NextGRID, however, we consolidate the *implementation*, *execution*, and *assessment* of the SLA lifecycle into one stage, which we name SLA Operations. Thus, the NextGRID SLA lifecycle consist of four stages:

1. SLA Publishing and Discovery;
2. SLA Negotiation;
3. SLA Operation; and
4. SLA Decommissioning.

2 Use Cases

The NextGRID SLA management use cases are aligned with the four stages of the NextGRID SLA lifecycle that were enumerated above. Thus, each of these four stages leads to a set of more detailed use cases, which are described in the sections below.

The main actors in these use cases are the service provider, from hereon referred to as the *provider* and the service consumer, from hereon referred to as the *consumer*. We do not distinguish between a customer and a consumer, and use “consumer” to mean the entity that is allowed to negotiate, agree on, and monitor SLAs. That entity is not necessarily the end-user of a service. It may delegate its negotiation rights to other actors, e.g. a negotiation *broker*. The provider and consumer are the actors to whom the guarantee terms, penalties, and premiums in the SLA relate.

For some of the use cases the roles of provider and consumer are obvious, predefined, and static. For all the use cases that are part of the SLA agreement process, the notions of provider and consumer are not relevant. Here, it is important to distinguish the *initiator* and the *responder*. This is due to these steps being symmetric in who starts the agreement protocol – it can be the consumer as well as the provider. Depending on the actual scenario, the roles of initiator and responder are assumed by the provider and the consumer as needed. It is also possible that the roles of initiator and responder are assigned to different actors in different use cases. For example, the consumer could be the initiator for template discovery, but the provider assumes the initiator role in the agreement use case.

Figure 2 shows an overview of the SLA management use cases.

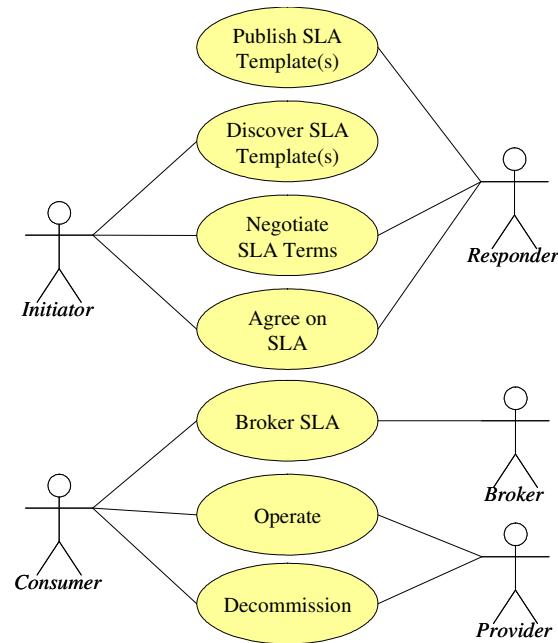


Figure 2 – SLA management use cases.

2.1 Publishing and Discovery

The NextGRID SLA framework assumes that consumers and providers either maintain and publish a list of SLA templates or create SLA templates on demand. An SLA template is the starting point for an agreed SLA. In NextGRID, SLA agreement is symmetric in that either a consumer or a provider can be the initiating party in the SLA agreement process. Thus, whichever party initiates the agreement process is responsible for choosing an initial SLA template as the agreement starting point. Thus, if the consumer initiates the agreement process, they can either use an SLA template of their own, or search for and discover a suitable SLA template from a candidate service provider. Symmetrically, if a provider is looking for custom, it can either advertise an SLA template relating to its service for a consumer to discover, or it can search and discover an existing SLA template from a candidate consumer, which it can agree to fulfil. It is easy to see that a consumer of a service from a service provider is the *provider* of custom to a *custom consumer*, i.e. a service provider, from a service provider's perspective. This leads to complete symmetry in the consumer-provider relationship.

The *discover SLA templates* use case in SLA management arises as a result of an initiating SLA agreement party choosing to discover an existing candidate SLA template from another party. This is as opposed to the, in this case, initiating party choosing to create their own SLA template. In the latter case, the template discovery step is omitted and the SLA agreement process directly starts with the *negotiate SLA terms* use case.

For ease of describing the discovery use case, let us assume that the initiating SLA agreement party is a consumer. In order to discover a service provider's (i.e. the responder's) SLA templates, the consumer must first choose an appropriate service provider. This is where service discovery takes place, i.e. in order to find an appropriate service provider, the consumer must search for one that is capable of fulfilling its requirements. Once an appropriate service provider has been identified, the consumer can then look up the SLA Template Repository [2] of that service provider for one or more SLA templates. The message flow for looking up templates is shown in Figure 3.

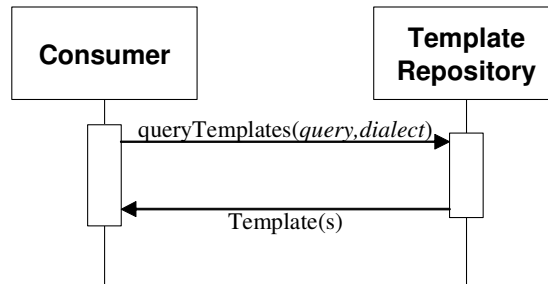


Figure 3 – The template lookup protocol

Service providers may be chosen in one of two ways:

- **From a known list of service providers:** The consumer can directly contact service providers that offer the desired service, from a known list of service providers, to get their SLA templates from the provider's Template Repository.
- **From a public registry of service providers:** Service providers can publish their services in public registries. To find a service provider that can offer their desired service, a consumer submits the requirements of that desired service to the registry in order to find candidate service providers whose service descriptions match those requirements. If matches are found, the registry then sends a list of suitable candidate service providers back to the consumer, who can then contact the service providers for their SLA templates.

In both cases, SLA templates need to be published before consumers can retrieve them. In the *publish SLA templates* use case, a service provider takes the steps necessary to make SLA templates available to consumers. The way to do this is to add the templates to the service provider's Template Repository. Later on, candidate consumers will get the published templates from that repository. Publishing templates works as shown in Figure 4.

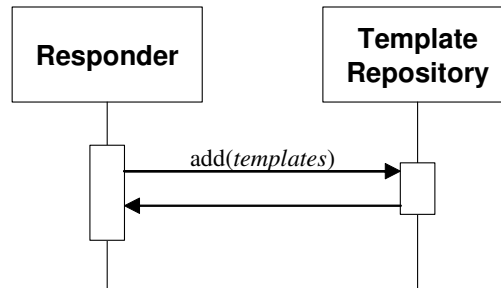


Figure 4 – The template publishing protocol

A relationship between a template repository and a service provider can be established in many ways and depends on the particular deployment of the components. This use case document does not mandate the use of a particular way of establishing and publishing such a relationship.

2.2 Negotiation and Agreement

Before a service can be provided in a relationship between any two parties in NextGRID, i.e. a consumer and a service provider, an SLA must be in place for that service delivery. This SLA will be binding on both parties, with a specified, quantifiable set of guarantee terms that relate to them and the delivery of service from one to the other.

Reaching an agreed-upon SLA can be separated into two steps: negotiation of the terms in an SLA and agreeing to the negotiated terms. These are two separate use cases as agreement can happen without prior negotiation and negotiation does not need to be followed by agreement. Both steps can be initiated by either the provider or the consumer.

The negotiation use case is all about creating the contents of an SLA that both parties can agree on. In this use case, the parties negotiate the terms of the SLA, which specify the level of service that is required by a consumer and delivered by a service provider, as a quantifiable set of metrics.

The agreement use case is about reaching an actual SLA from a predecessor of an SLA, called *SLA proposal* in the following. Creating such an SLA requires both parties to commit to the same SLA contents. Commitment in NextGRID is shown by digitally signing an SLA proposal. As soon as both parties have signed an SLA proposal (i.e. they have committed to the contents of the SLA), it becomes an actual SLA, called *SLA instance*.

In general, the agreement step is preceded by a negotiation phase. The consumer chooses a starting point – either from a set of SLA templates or a custom-made SLA proposal. It adjusts the SLA contents to its liking and sends the SLA proposal to the provider. The provider is responding with another SLA proposal that contains terms that are changed or added according to the policies and possibilities of the provider. The exchange of SLA proposals continues until one of the parties either terminates negotiation or enters the agreement phase. Negotiation can continue for an arbitrary amount of time. For practical reasons, the time or the number of proposal exchanges should be restricted.

Entering the agreement phase is signified by one party sending a binding *offer* to the other. A binding offer is an SLA proposal that is digitally signed by one party. The transmission of such an offer by the signing party (the initiator) is the first step in the agreement protocol. This step is either followed by the rejection of the offer or by the creation of an SLA. Rejection of an offer can be followed by another negotiation phase. The creation of the SLA happens when the other party (the responder) signs the offer as well. The result is an SLA instance that has been signed by both participants showing that both parties commit to the SLA. The initiator can be either the provider or the consumer. The decision of who takes on that role is independent of previous decisions, including who started negotiation, who supplied templates, etc. The agreement protocol is shown graphically in Figure 5. The states that an SLA proposal transitions through during the execution of that protocol are shown in Figure 6.

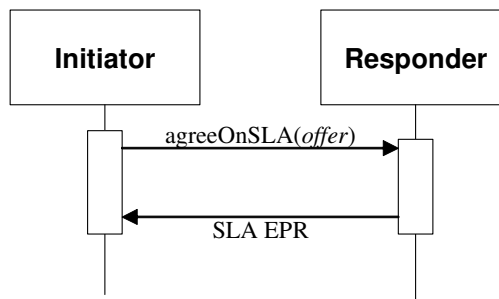


Figure 5 – The agreement protocol

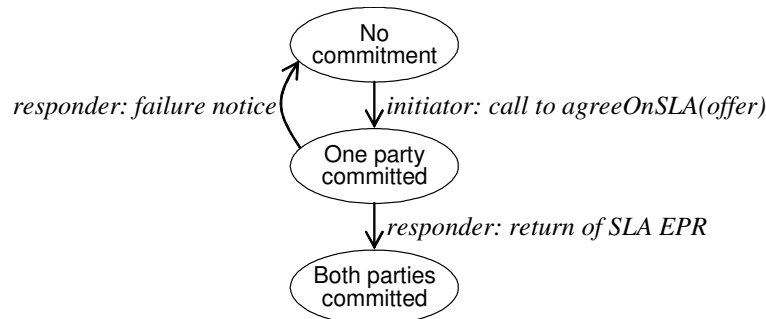


Figure 6 – SLA states and transitions

A special case of the negotiation and agreement phase is the *discrete service quality model*. In this model, the negotiation phase is skipped and both parties directly proceed from template discovery to the SLA agreement step. The reason for skipping negotiation is that in this model, terms in SLA templates are deemed fixed and are non-negotiable. Consumers can therefore either accept the contents of an SLA template or look for an alternative service provider. This model is termed discrete service quality model as service providers publish a number of SLA templates with different service qualities. These templates mark discrete points in the continuous service quality space. Note that symmetry exists in this case as well as the more complicated case, e.g. either party may publish non-negotiable SLA templates

To summarise, there are four variations of an SLA document that are involved in the negotiation and agreement process. The parties start with an *SLA template*, which is then modified into an *SLA proposal*. The latter can be modified further into another SLA proposal, a step in the negotiation that can be undertaken multiple times by either party. The SLA proposal is followed by an *SLA offer* which is that proposal digitally signed by one party. The final result is an *SLA instance* document which has guarantee and obligation terms relating to both parties, which specifies penalties for defaulting on those terms, and which is signed by both partners.

2.2.1 Evaluation

A special case of the negotiation agreement that is strongly related to the *discrete service quality model* is the *SLA Template Evaluation*. The selection process of an SLA includes the evaluation of the SLA terms according to each involved party's demands. The provider must be able to publish SLAs that are matching his capability to deliver QoS (Quality of Service) at a certain level, whereas the consumer must be able to agree on SLAs that match his QoS requirements. The QoS level is defined by the *SLA terms*, the parameters used for reaching an agreement that are usually closer to the consumer's understanding.

In order to evaluate the success rate of an SLA, the parties must keep a historic record of the *SLA violations* that possibly occurred. Measuring the success rate of an SLA can give valuable insight as of what selection policies to deploy.

Of course, given the dynamicity of the SLAs each one of them is unique; not all the SLA parameters can be the same in two SLAs. Therefore, no party can assess the SLAs itself but rather it can assess the SLA terms. The *SLA Templates* from which the respective SLAs are derived can be used as a reference point in order to have an understanding as to which violation on a term came from which SLA.

The evaluation is different for every category of actor and it is based on each one's knowledge. The provider has direct access to the infrastructure therefore can evaluate SLA terms mapping them in low level infrastructure parameters. Whenever the infrastructure is not delivering the required QoS the provider understands that a violation

occurred to an SLA term. On the other hand, the consumer can only use his experience in order to evaluate a violation. If the result of an invoked service does not match the consumer's understanding of QoS, then he can directly map it to a violation of an SLA term. Therefore the consumer uses his own Quality of Experience (QoE).

The evaluation process is therefore symmetric. The provider evaluates the existing SLA Templates that are most likely to match the consumer's demands in terms of QoS and selects them so as to publish them or even creates new. The consumer evaluates the published SLA templates according to his own experience and selects one. The procedure then is to negotiate and agree upon the selected template and conclude to an SLA. SLA Template Evaluation works as shown in Figure 7.

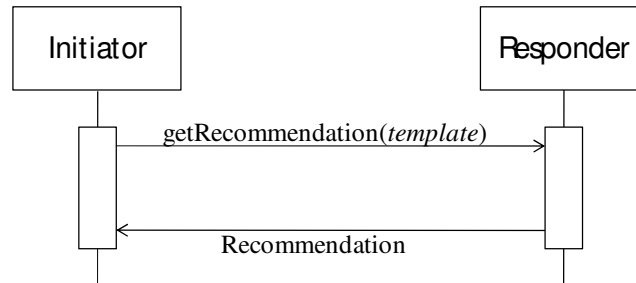


Figure 7: The Evaluation Protocol

Negotiation can be a tedious process for a service consumer, in particular when a large number of providers offer potentially matching services. The burden of negotiating with multiple providers and selecting the most appropriate one can be offloaded to a negotiation broker. The broker works on behalf of the consumer. It collects SLA templates from a set of potential providers, orders them according to their fulfilling the requirements of a consumer, and then tries to negotiate an SLA with the most appropriate provider. The broker can be operated by the same entity as the consumer, but it may also be provided by an external third party that specializes in brokering services. For template retrieval and negotiation, the broker uses the protocols described above. The protocol between a broker and a consumer is shown in Figure 8.

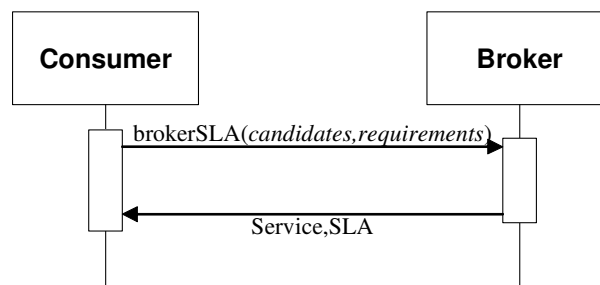


Figure 8 – The brokering protocol

The consumer sends its requirements on an SLA to the broker, along with a set of potential service provider candidates. The broker tries to establish an SLA that fulfills all the given requirements. If SLA establishment was successful, the SLA as well as the service provider that this SLA relates to are communicated back to the consumer.

2.3 Operation

Once a suitable service provider and SLA template have been discovered and an SLA agreement reached, an SLA instance will be ready to help govern the provision of the service. The purpose of this agreed SLA instance is to enable the service provider to perform automated, SLA-driven service management using SLA operations. This



should be facilitated by a management system that can automatically respond to changes in the values of SLA metrics as represented by the SLA terms relating to the delivered service. To this end, NextGRID has developed and brought together a set of technologies to facilitate SLA operations.

According to the NextGRID SLA architecture, an agreed SLA instance is managed collaboratively by both parties. In managing the SLA instance, the consumer is represented by a *Consumer Smart Bit* and the provider is represented by a *Provider Smart Bit*. These “smart bits” are SLA management systems which are proprietary to the party that they represent. Thus, the Provider Smart Bit manages the SLA instance with respect to the requirements of the service provider in managing the service being delivered, and in managing the provider’s infrastructure and resources for the delivery of that service. Likewise, the service consumer’s Consumer Smart Bit will manage the SLA instance in order to meet their obligations as stated in the SLA instance, including the payment of penalties to the service provider and receipt of compensation from service providers, should either party not meet their SLA obligations, respectively.

The main aim at this operations stage of the SLA lifecycle is to move from the requirements of the customer, encoded as a set of business level objectives using the NextGRID SLA schema [3], to a set of resource requirements and a strategy for the automated deployment and management of the resources, as encoded using the NextGRID policy schema.

Although NextGRID does not specify the design details of the “smart bits” introduced above, it does specify the SLA operations that are required to manage a SLA instance. SLA operations start by ensuring that the SLA instance, which has been agreed according to a set of business level objectives, can be mapped to the service and resource fabric layer of the service provider. Thus, the Provider Smart Bit must have a translation capability that is able to undertake the mapping between the business layer and the resource fabric layer for the terms in an agreed SLA. Since it is not desirable for the consumer to be exposed to the details of the SLA terms pertaining to the resource fabric layer, the Provider Smart Bit must handle these terms in such a way that the business level terms of the SLA reflect the state of the resource layer correctly, and thus present an accurate state of the system, in terms of QoS, at the business level that the consumer can understand.

This operations stage of the SLA lifecycle involves a number of sub-stages that are defined by a number of activities. These sub-stages are:

- **Mapping to a Standard Operating Portfolio (SOP):** The service offering in the SLA are mapped to relevant elements of the service provider’s SOP by the Provider Smart Bit. These elements may include the application servers and database servers as well as the presentation layers for the standard three-tier application. This operation will retrieve a default set of technical configuration and management details for the service, which can be refined in further stages.
- **Applying organisational governance policies:** The set of technical parameters built from the elements of the service provider’s SOP will contain default management information that will define a set of policy decisions to deal with issues that may occur during runtime. At this stage these details are refined to fit the governance model, i.e. how the service provider prioritises service delivery, of the provider.
- **Taking count of legal and regulatory issues:** A service provider must always take note of the legal and regulatory obligations requested of them in a SLA. These may vary according to the consumer’s geographical location and business domain. This will place extra logging and auditing requirements on the provider and require extra configuration for these requirements to be added to those already identified in the previous stage.



- **Taking count of historic information:** It is important that a provider uses previous experiences of offering similar services to optimise their service delivery. To this end, the QoS history, and the policies that were managing the system when the recorded QoS was delivered, should be consulted to further refine SLA threshold levels. A database can be used to store this historic data.
- **Creating the service model:** Once all this information has been collected, the service is ready to be mapped to the service provider's infrastructure. This will involve creating a graph to describe the deployment of the service and management functionality, and obtaining the physical addresses and locations of the allocated hardware. This is the final step before deploying the service components onto the infrastructure and starting service delivery.

To facilitate the mapping of SLA terms from the business level to the resource fabric layer, NextGRID has prototyped a SLA intermediate language (SLA-IL). In addition to helping map business objective terms to resource fabric requirements, the SLA-IL helps to define the method for the collection and combination of monitoring information in order to determine if a threshold that indicates a SLA violation has been breached. Monitoring information collected is then passed to an *evaluator* component that instigates specific actions in response to SLA violations, according to a set of policies relating to the use of specific monitoring information or a combination of that information. Policy actions include, for example, requests for further resources to avoid further SLA breaches and penalty payments. The evaluator is constrained by a resource *allocator* component which has a global view of all resource allocations in the infrastructure and understands the SLAs associated with these allocations.

The SLA-IL is used, in a set of mappings, to describe SLAs at the resource management level. The SLA-IL framework defines an XML-based language as well as a runtime engine for monitoring and auditing of SLAs. The language helps to link the business level objectives expressed using the SLA schema, and the SLA management actions expressed as monitoring tasks. These monitoring tasks are flexible, configurable, and enable the monitoring system to log the necessary information and to generate events if required.

The three main elements in the SLA-IL are *Measurement*, *Computation*, and *Violation*. Monitoring rules, the Measurements, can be associated with higher-level aggregations and filters, the Computations, and verified by audit instructions, the Violations. A violation is used to check whether a term in an SLA is fulfilled.

A measurement describes what information to extract from one or more sources (Soap messages, management system, the IP stack, etc...) and when to extract it. The measurement then packages this information and forwards it. A computation describes the filtering and aggregation to be applied to the received information from measurements and other computations. It describes also when to do the filtering and the basis of the aggregation (time based or volume based). A violation receives data from measurements and computations and verifies that it is within a defined range.

In addition to these three elements, there are four supporting elements in the SLA-IL. *Schedules* describe the generation of time events. Measurements and computations can be based on these time events to start an action. *Runtimes* specify the machines on which measurements, computations, and audits are instantiated and performed. *Loggers* state the machine that does the logging. *Message Identifications* describe which messages to capture for analysis.

The SLA-IL is supported by a *runtime engine* that performs the monitoring and auditing activities. The NextGRID SLA framework is composed of an *XML Parser* for SLA documents, an *Object Model* for validating the SLA documents and also for directly writing SLA documents, and an *Object Model Compiler* for configuring the runtime engine for monitoring and auditing SLA documents.



The elements in the SLA-IL can be combined to build a flexible description for a monitoring scenario. The root element is SLA, which may contain up to seven sub-elements, which describe the monitoring requirements.

2.4 Decommissioning

Successful provision of a service is defined by the business level objectives in the SLA instance that governs that service's provision. Once a service has been successfully delivered, the service provider and consumer are deemed to have fulfilled their obligations and the provider can then decommission the SLA. This may occur while the SLA instance is still valid, i.e. before its default lifetime as stated using the SLA schema.

3 References

- [1] TeleManagement Forum, SLA Management Handbook Member Evaluation Version 2.0, July 2004
- [2] NextGRID SLA Template Repository Profile, NextGRID Project, October 2007
- [3] NextGRID SLA Schema, NextGRID Project, July 2007
- [4] NextGRID SLA Negotiation Schema, NextGRID Project, August 2007