GXLA a Language for the Specification of Service Level Agreements

Badis TEBBANI and Issam AIB

Laboratoire Informatique de Paris 6 (LIP6)
Université de Paris 6
8 rue du capitaine Scott
75015 Paris France
(name.surname)@lip6.fr

Abstract. In this work we propose GXLA, a language for the specification of Service Level Agreements (SLA). GXLA represents the implementation of the Generalized Service Level Agreement (GSLA) information model we proposed in a previous work. It supports multi-party service relationships through a role-based mechanism. It is intended to catch up the complex nature of service interactivity in the broader range of SLA modeling of all sorts of IT business relationships. GXLA is defined as an XML schema which provides a common ground between the entities in order to automate the configuration. GXLA can be used by service providers, service customers, and third parties in order to configure their respective IT systems. Each party can use its own independent SLA interpretation and deployment technique to enforce the role it has to play in the contract. An illustrative VoIP service negotiation shows how GXLA is used for automating the process of SLA negotiation and deployment.

Keywords: *Self Configuration; Service Level Agreement; Policy Based Management; SLA Specification, Policy Specification.*

1 Introduction

Policy-Based Network Management (PBNM) is concerned with the use of rules to manage the configuration and the behavior of one or several entities in an IT infrastructure. A rule connects a set of conditions with a set of actions. When the conditions are verified, the corresponding actions are activated. The evaluation of the conditions can be triggered by some events.

In the context of service-oriented management, service level agreements (SLAs) are a very useful tool. Their necessity is continuously increasing, essentially for resource optimization and configuration automation. In the context of service provider networks, where collaboration and services exchange takes precedence, Policy-Based Management is necessary to react quickly to the changing environments without human intervention. There is currently no existing standard for specification of SLAs. Accordingly, there are many proposals that are more or less adapted to the information technologies (IT) context. The specification must be generic in order to be applicable to all possible scenarios and extensible to be able to add specific parameters.

Note that, in the literature, we can distinguish three domains of studies that overlap and are complementary. The success of a PBNM depends on these three solutions: (i) The representation and exchanges of the hardware configurations [1,2], where the stress is laid on the performance and the scalability of the PBM system, generally, it is supposed that the database is already filled by policy rules with respect to the SLAs. (ii) The second field of study is policy refinement, it concern representation, checking, validation, detection/resolution of the conflicts and the enforcement of the policies. This field overlaps with the first one especially in hardware configuration level. Besides it also proposes tools needed for the storage of policies [3,4]. (iii) The third field of study is about modeling the external world of the system, like the business environment, the service negotiation and the technical parameters describing the service [5,6]. In this field, a precise representation is needed in order to make services understood by all parties in both business and hardware level.

In this work, we focused on the last field of studies: modelling and formal specification of the service level agreements that make able to clearly describe the customer requirements and the objectives of the company. The majority of the existing specifications in the literature are specific to the Web services, and do not allow to describe several providers in the same service. We propose a language XML-based, role-oriented and multi-parties, who allows specifying several parties involved in the service (provider, subcontractors. . .) by associating each one to one role that it have played in the agreement. Our specification offers a simple interface to the users (or to the networks administrators) to request a service (or to introduce policies) in a business language, then it translates this contract into corresponding configuration policies.

The paper is organized as follows. In the next section, we will examine related work on the specification of SLAs. Existing work on the modeling of the policies will be presented in the third section. Our GSLA information model and its specification into the GXLA XML schema will be explained in the fourth section. The next section will show an example of execution to test our specification and we will conclude by some perspectives for our work.

2 Work on the Specification of Service Level Agreements

2.1 WSLA Web Service Level Agreement IBM2003

WSLA [5] is a good case of study where the steps and the course of the design of a language of specification of the SLA are explained. WSLA is XML-based. In WSLA (Fig. 1), the SLA class contains three parts: the first describes the parties involved in the service, the second contains one or more descriptions of the service and the third describes engagements. It defines signatory parties to be either a service provider or service customer. They are the main parties of the SLA and

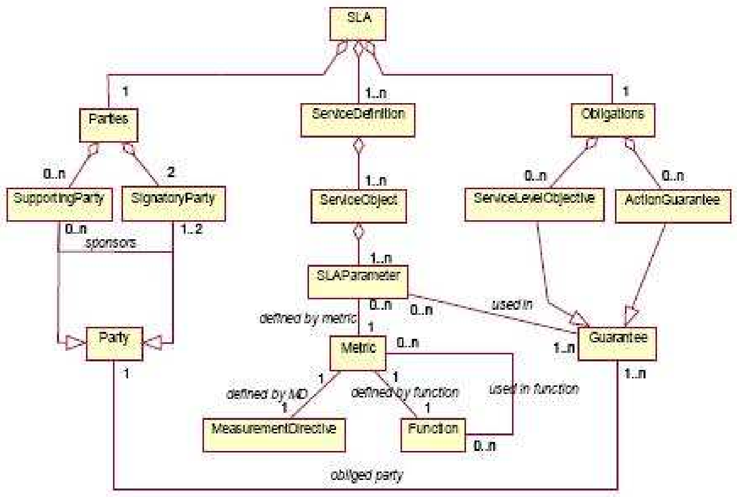


Fig. 1. Overview of main WSLA components [5]

are assumed to "sign" the SLA, bearing the final liability. Additional supporting parties can also exist and are sponsored by one or both of the signatory parties. The description of the service contains one or more operations which define one or more parameters. In their turn the Parameters are connected to metrics. A metric can be: (i)Connected to a directive which indicates how it must be measured. (ii) Connected to a function, in this case the metric one can be the combination of one or more other metric. Finally the Engagement class contains two classes of obligations: no or several SLO which guarantee a state of the SLA for one determined period, the other type of engagements is the actions promised in the SLA at the time of particular events.

The WSLA insists on the process of measurement of metric and articulates its specification to allow to the service provider (or customer) to sub-contract this process with a third company. WSLA is based on the descriptors of services Web WSDL, SOAP, UDDI which facilitates the description of the service and its indexing.

2.2 SLAng

SLAng is a SLA language, under development as an element of the project TAPAS in United College, London [6]. It is a XML-based language and which rests on the description languages of the Web services (WSDL) and the server applications (J2EE, CORBA).

SLAng defines seven types of SLAs. Initially, it divides the SLA into two main categories: (1) Horizontal SLA: the contract between two equal entities (of the same level of architecture), eg. Two applications. (2) Vertical SLA: the contract between two entities in different layers, eg. The networks layer with the application. SLAng conceives the SLA like a contract between strictly a customer and a provider without taking into account the other actors in the network. SLAng introduces the concept of client responsibility, provider responsibility and the mutual responsibility (eg. penalties). Each one of it describes the obligations of each part. Our GXLA extends this concept in the role and defines the responsibility or the guarantees for a service independently, to be able to then assign it to one or more parties.

3 Work on the Specification of Policies

3.1 PONDER Imperial College, London

The Ponder(Imperial College, London 1992-2005) offers a framework to specify access rights and management policies with an role-based access control [3,7]. Ponder is a declarative, object-oriented language, based XML and supports the typing and the combination of the policies. Ponder consider a policy as the choice of setting parameters of a network component, or the authorization access according under certain conditions. It defines several types of policies witch articulated on ECA-paradigm (Event, Condition, and Action).

Ponder defines four types of policies: (1) *Authorizations:* are essentially security policies related to access-control and specify what activities a subject is permitted or forbidden to do, to a set of target objects. They are designed to protect target objects so are interpreted by access control agents or the run-time systems at the target system. (2) *Obligations:* specify what activities a subject must do to a set of target objects and define the duties of the policy subject. Obligation policies are triggered by events and are normally interpreted by a manager agent at the subject. (3)*Refrains:* specify what a subject must refrain from doing and are similar to negative authorisation policies but are interpreted by the subject. (4) *Delegations:* specify which actions subjects are allowed to delegate to others. A delegation policy thus specifies an authorisation to delegate. These four types are the bases of all the policies, Ponder defines also four other types of composition of policies to group the policies belonging to the same field to simplify the specification: *groups*, *roles*, *relationships* and *management structure*. Ponder also defines the limits of the applicability of the policy and it introduces the concept of *Domains* to group the policies which concern, the same subject or the same geographical area.

In Ponder, the actions are achievable files, may be of the external scripts stored in precise fields. The access to these fields is controlled by the *Authorization* policies. Lastly, the Ponder specification is a very good candidate for the implementation of the SLAs. But, it must raise its level of abstraction to take into account the applicative needs and all the structure of the SLAs. Work on Ponder has been also stopped for two years and we do not see well how it will evolve.

Figure 2 show an example of a Ponder policy : The members of Agroup and Bgroup can accept carry out a videoconference with the personnel based in the United States with

```
inst auth+ VideoConf1 {
     subject  /Agroup + NYgroup;
     target    USAStaff-NYgroup;
     action    VideoConf(BW, Priority);
     when      Time.between(1600, 1800)
        and(Priority > 2);
        }
```

Fig. 2. Ponder policy Example

the group of New York. The constraint of the policy is made up here. The time constraint limits the service to apply only between 4:00pm and 6:00pm and the constraint of action indicates that the policy is valid only if the priority parameter is larger than 2.

3.2 WS-POLICY (IBM, Microsoft and al. 2003)

WS-Policy [8] is a framework which defines the basic structures for the description and the communication of the policies of the Web services. It forms a flexible and extensible grammar to express the needs, capacities and preferences of a service Web in a format XML. The policies are defined in WS-Policy like a collection of "alternatives". The alternative is a collection of assertions. The assertion is the basic unit of the policies. Each assertion belongs to a specific field. WS-Policy defines the applicability of the following policies: *Security*, *Privacy*, *Priorities application*...etc. The assertions can be gathered or combined by the operators <*wsp:All*>, <*wsp:ExactlyOne*> and <*Optional*>.

WS-Policy does not define the way in which the policies will be interfaced with the Web services; it leaves the free field to other specifications that define the way in which the policies are attached to the Web services for a better adaptability. For example, *WS-PolicyAttachement* is a specification which defines the association of WS-Policy with WSDL and UDDI. The principal idea of WS-Policy is to offer a grammar to describe the policies (preferences, capacities...) of each Web service and then to be able to exchange this information and to include/understand their semantics. WS-Policy is integrated in messages SOAP, this one being extensible to support new types of messages. Lastly, WS-Policy defines three operations for the treatment of the policies: *Normalize*, *Merge* and *Intersect*. These operations define a model to compile the policies by standardizing each policy initially and then to combine them according to a some rules.

Figure 3 represent two specific security policy assertions that indicate that two types of authentication are supported. A valid interpretation of the policy above would be that an invocation of a Web service contains one of the security token assertions specified.

```
<wsp:Policy>
   <wsp:ExactlyOne>
    <wsse:SecurityToken>
     <wsse:TokenType>wsse:
      Kerberosv5TGT</wsse:TokenType>
     </wsse:SecurityToken>
     <wsse:SecurityToken>
     <wsse:TokenType>wsse:
     X509v3</wsse:TokenType>
    </wsse:SecurityToken>
   </wsp:ExactlyOne>
</wsp:Policy>
```

Fig. 3. WS-Policy Example

4 The GSLA Framework

4.1 GSLA Information Model

A GSLA [9] is defined as a contract signed between two or more parties relating to a service relationship and that is designed to create a clear measurable common understanding of the role each party plays in the GSLA. A party role represents a set of rules which define the minimal service level expectations, and service level obligations that the party has with other roles and at which constraints. The constraints might be of any type and normally include contract scope (temporal, geographical, etc.), the agreed upon billing policies, as well as the expected behavior in case of abnormal service operation.

We identify in Fig. 4 the top-level components of the GSLA. A GSLA comprises a set of parties joined according to a certain schedule in order to realize the contract by playing each one or more roles. During the GSLA life cycle, a required behavior or constraint related to the GSLA is captured in the model by the abstract GSLAPolicy component. A *GSLARole* is modeled at first approximation by a set of GSLA rules, and as it participates in defining the behavior of the system, it is derived from the *GSLAPolicy* component. A Schedule component represents the temporal scope during which a GSLA component is valid. Finally, a GSLA object comprises one or more *Service Packages* to each of which is associated a *Service Package Objective* that some GSLA party is required to guarantee as is specified in its attached role(s).

A *service package* is an abstraction that enables different service elements (customer facing services) to be packaged together. Each *Service Element* is related to one or more *Service Resources*(SR). A service element is enabled through a set of service resources. A *service resource* is intended to be transparent to the customer and represents a basic provider resource, such as an email server, a network element, a processing server, a database, a shared file, or a stockpile.

We propose a way for modeling QoS that caters for the specification of both high-level business objectives as well as that of low-level resource QoS parameters. The *Service Package Objective* (SPO) component defines Service Level

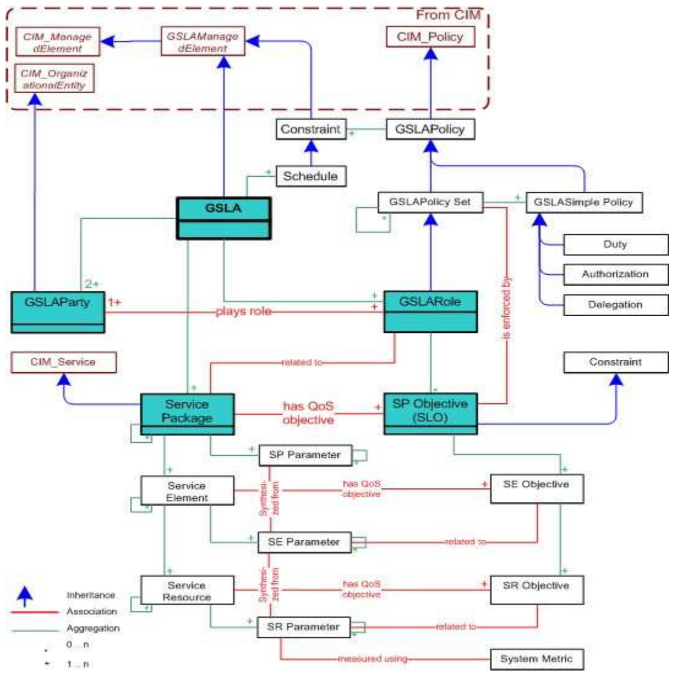


Fig. 4. GSLA [9]

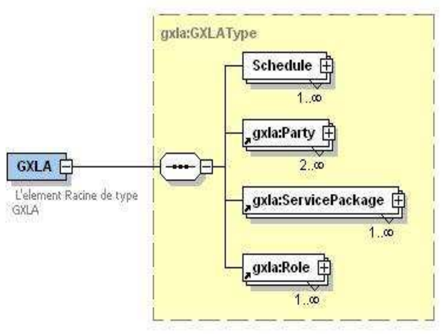
Objectives for one or more SPgs. Basically, an SPO is a constraint and we allow it to be defined in two possible ways: First, as a set of predicates or logical expressions over one or more SPg Parameters. This represents a high-level way of defining QoS objectives based on direct calculus made over high-level service parameters that are synthesized from basic System Metrics up through System Resource (SR) parameters and System Element (SE) Parameters. The other way around is to calculate the objectives based on QoS appreciations coming from subordinate *Service Element Objectives*. This represents the high-level compilation of low-level QoS appreciations. This latter approach reflects better the way users appreciate a given service infrastructure; i.e. by giving a final appreciation based on separate 'sub' appreciations over the different service components. In the GSLA information model, multiple party service relationships are supported and each party has a set of SLOs to assure and some behavior to follow with respect to the other parties. Also, to each SLO are generally associated rules (or policies) that define actions to take in case the SLO has not been respected or some threshold has been reached. We structure related SLOs and their corresponding policies into the Role class. Roles can be of several types: compulsory (such as supporting a specific routing scheme in an ad hoc network), optional (SuperNode in a p2p file sharing community), required by at least one party (Central Controller in an IBSS), or statically attached to a specific party (a VoD Provider) at the GSLA specification time.

A party behavior is captured within the *GSLARole* component. As Roles are ultimately translated into low level policies, a party behavior at the lowest view is modeled as a policy. A policy is of two types, either a duty or an authorization. A duty defines conditions that need to be met in order to execute some system operations. They represent the key to QoS policy specification in our model. A GSLARole is modeled through the set of duty and authorization policies having their subject domain the party or the group of parties that play that role; as well as the set of SLOs that it is required to ensure as part of its responsibilities in the SLA.

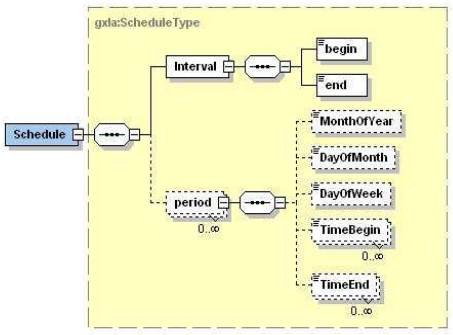
4.2 GXLA, an XML specification for the GSLA

The GXLA is a XML-Schema which implements the GSLA information model. It is usable at the same time by the service provider and the customer of the service in order to configure their respective systems. Moreover, it allows the internal use by the provider for compilation and the configuration of its system in order to provide the concluded services. The SLA in the GXLA is composed of four sections (see Fig. 5.a):

1. *Schedule*: represent the total temporal range of the contract. It is structured like a set of time intervals delimited by a beginning and an end (Fig. 5.b).



(a) Main GXLA components

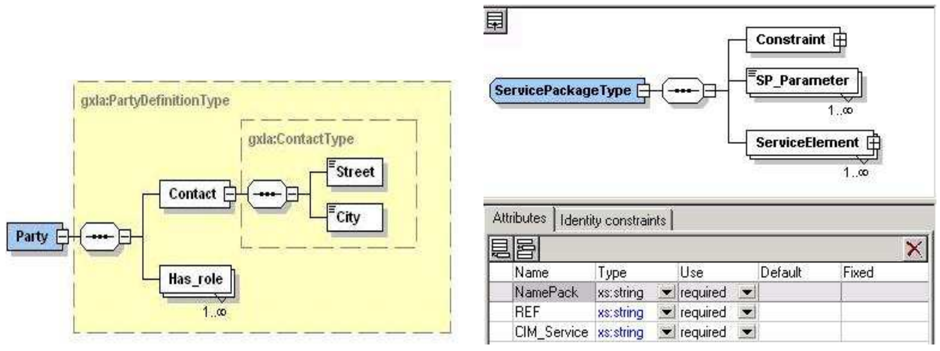


(b) Schedule

Fig. 5. Global GXLA(a) and GXLA.Schedule(b)

2. *GXLAParty*: represents the parties involved in the service. Unlike to the preceding specifications, the GXLA describes general information of each party without specifying its nature (provider or customer) in order to cover all the possible states in the services networks market. The element *Has_role* (Fig. 6.a) permit to attach each party to one ore several roles that it will play in the service.
3. *ServicePackage*: is an abstraction to describe one or more services (Fig. 6.b). It is composed of:

- *Constraint*: specify the temporal range of each service within the limit of the temporal range of the including components, in fact the *Schedule* constraint of the SLA.
- *SP-Parameter*: is the formalization of the high level parameters and represents the common sight of the service by all the participants. We define them in the GXLA like results of synthesis of metrics (*SE-Parameter*). *SP-Parameter* or *SE-Parameter* can be attached to one or more SLO which supervise QoS of the service.
- *ServiceElement*: is an abstraction to define the service in question, its operations, its constraints and its parameters. The GXLA makes it possible to describe the service with high level parameters (vocabulary of the contract) visible at the customers. Then, it attaches each service to one or more resources system (*ServiceResource*) in a transparent way to the customers. Indeed, each parameter of the service (*SE-Parameter*) is the result of a function or a directive on the metric ones of the system (eg. SNMP report). The metric can be atomic (eg. measure on a resource) or made up of several other metrics (eg. average of the number of server invocations).



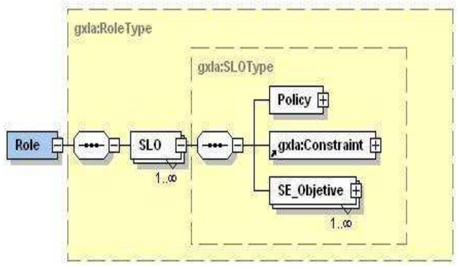
Name	Type	Use	Default	Fixed
NamePack	xs:string	required		
REF	xs:string	required		
CIM_Service	xs:string	required		

(a) Parties (b) Service Package

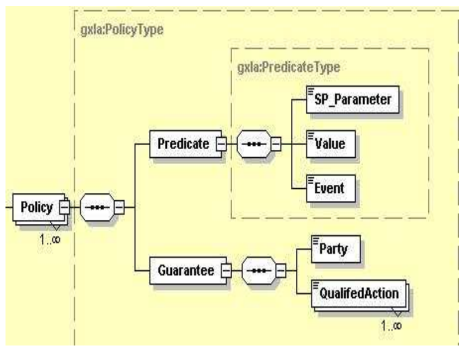
Fig. 6. GXLA.Parties(a) and GXLA.ServicePackage(b)

4. *Role*: For each service, QoS corresponds guaranteed by the provider and waited by the customer. The GXLA described in the *Role* element, the SLOs which will define the level of service necessary for each service (Fig. 7.a). A SLO, in the GXLA, is ensured by one or more policies (Fig. 7.b). The policies model in the GXLA are event-oriented, the *predicate* checking is done each time when event arrive; the *SP-Paramter* is compared to the *Value*. If the condition is verify, *QualifedAction* is triggered.
The policies have many types: (i) policies of obligation which define the *QualifedAction* and the party responsible for the role under certain conditions on *SP-Parameter* (eg. execution of a schell, notification in the case of

a violation or a value close to a threshold). Obligation policies are the key of the QoS guarantees.(ii) policies of authorization which define the access rights for certain actions on a system resource under certain conditions, in particular on the metric ones (eg. to change the size of a queue).



(a) Role

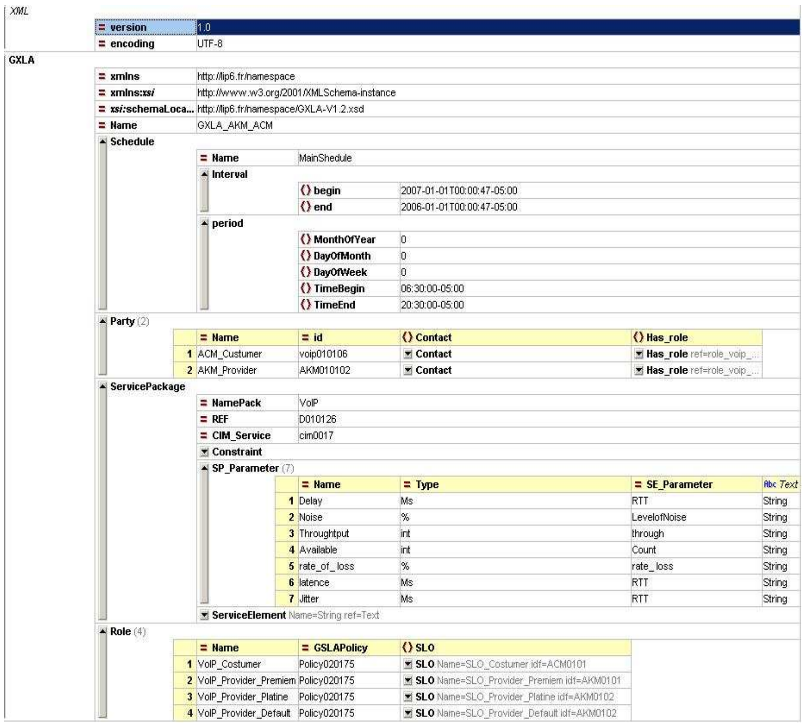


(b) Policy

Fig. 7. GXLA.Role(a) and GXLA.Policiy(b)

5 Use Case: VoIP Service

In this section, we consider a scenario where a customer requests a Voice over IP (VoIP) service. The context of our scenario includes a service provider owning a network with a geographic coverage and an underlying PBM system and it offers service through a Web site. Through the web site portal, the customer chooses his services and after signing the contract he starts to use them. The implementation comprises two parts: the customer part which is an Applet and the provider part which is a set of Servlets that intercept information coming from the customer and treat them. We test our specification through two aspects: service negotiation and automatic generation of suitable policies. The GXLA must allow collecting personal user information on one hand and the characteristics of the service on the other hand. In addition, starting from information on the level of quality of service desired by the customer, the GXLA is used along with policy generation templates to infer the suitable policies. For the creation of service level agreements, we prepared a generic contract GXLA (see Fig. 8) which comprises all the available services with all existing levels of available qualities of service . After the negotiation, the servlet creates an instance of the generic contract (GXLA) and personalizes it with: customer information, subcontractors (if they exist) and especially suitable policies. Tables 1 and 2 show some typical policies for the VoIP service with two levels of QoS. We considered in our implementation that the provider proposes its various services through a Web



XML

version	1.0
encoding	UTF-8

GXLA

xmlns	http://lip6.fr/namespace
xmlns:*xsi*	http://www.w3.org/2001/XMLSchema-instance
xsi:schemaLoca...	http://lip6.fr/namespace/GXLA-V1.2.xsd
Name	GXLA_AKM_ACM

Schedule

Name	MainShedule

Interval

begin	2007-01-01T00:00:47-05:00
end	2006-01-01T00:00:47-05:00

period

MonthOfYear	0
DayOfMonth	0
DayOfWeek	0
TimeBegin	06:30:00-05:00
TimeEnd	20:30:00-05:00

Party (2)

	Name	id	Contact	Has_role
1	ACM_Custumer	voip010106	Contact	Has_role ref=role_voip_...
2	AKM_Provider	AKM010102	Contact	Has_role ref=role_voip_...

ServicePackage

NamePack	VoIP
REF	D010126
CIM_Service	cim0017
Constraint	

SP_Parameter (7)

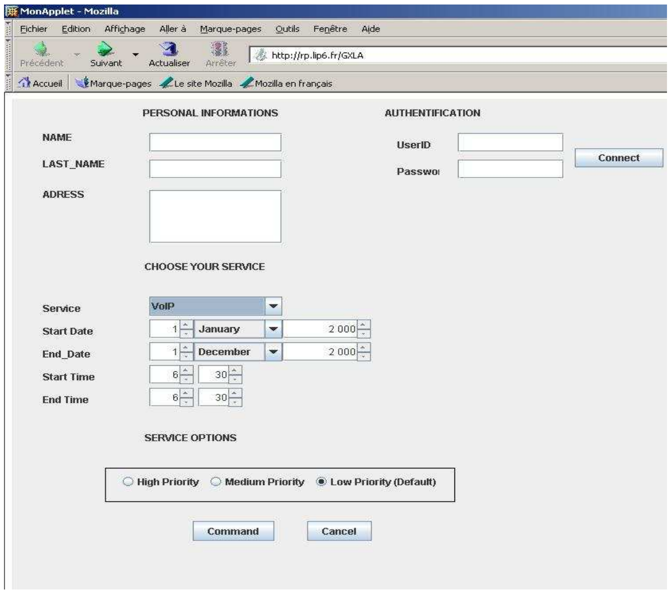
	Name	Type	SE_Parameter	*Text*
1	Delay	Ms	RTT	String
2	Noise	%	LevelofNoise	String
3	Throughtput	int	through	String
4	Available	int	Count	String
5	rate_of_ loss	%	rate_ loss	String
6	latence	Ms	RTT	String
7	Jitter	Ms	RTT	String

ServiceElement Name=String ref=Text

Role (4)

	Name	GSLAPolicy	SLO
1	VoIP_Costumer	Policy020175	SLO Name=SLO_Costumer idf=ACM0101
2	VoIP_Provider_Premiem	Policy020175	SLO Name=SLO_Provider_Premiem idf=AKM0101
3	VoIP_Provider_Platine	Policy020175	SLO Name=SLO_Provider_Platine idf=AKM0102
4	VoIP_Provider_Default	Policy020175	SLO Name=SLO_Provider_Default idf=AKM0102

Fig. 8. A Global Specification of a VoIP with three levels of quality of service


MonApplet - Mozilla
Fichier Edition Affichage Aller à Marque-pages Outils Fenêtre Aide
Précédent Suivant Actualiser Arrêter
http://rp.lip6.fr/GXLA
Accueil Marque-pages Le site Mozilla Mozilla en français

PERSONAL INFORMATIONS

NAME

LAST_NAME

ADRESS

AUTHENTIFICATION

UserID

Passwo

Connect

CHOOSE YOUR SERVICE

Service	VoIP		
Start Date	1	January	2 000
End_Date	1	December	2 000
Start Time	6	30	
End Time	6	30	

SERVICE OPTIONS

High Priority Medium Priority [✓ Low Priority (Default)]

Command Cancel

Fig. 9. User interface

site. The customer thus has a convivial interface which posts the services that a customer can request without any skills in computer networks. This interface (Fig. 9), which conforms the GXLA, permits the customer to introduce his personal information, desired service(s) and level of quality of service. In a step forward, the provider receives the customer request under the shape of a XML file, respecting GXLA specification, which is parsed in order to extract necessary information. Then, using XSL, the provider produces a HTML version of the contract including details like the service(s) price which is send back to the customer permitting him to choose whether to accept or refuse the agreement. If he accepts, the contract is agreed then the provider proceeds to turn up the service and notify the customer.

Through this example, we showed the role of GXLA in the process of automatic services negotiation. Once the contract is concluded, the provider must parameterize its equipment in an optimal way in order to honour the contract. Next paragraph is dedicated to show the role of GXLA in such a procedure and especially in the translation of the SLA agreement into service level objectives. The provider translates the agreed contract according to GXLA specifications:

1. He will initially define the role of each participant: itself and the customer.
2. The provider by defining the role of each one lays down the policies which must be applied in order to guarantee the contract. The policies in our example are deduced from to rules illustrated in table 1 and 2.

	Predicate	Unit	Action	Etat
P1	Delay > 99	μs	Notification	Active
P2	Jitter > 9	μs	Re-connect + Notification	Active
P3	Rate-of- loss > 0,98 then	%	Gold + Notification	Active
P4	Noise > 5	%	Penalties = Penalties+1 %	Active

Table 1. Example of the policies for the VoIP-Premiem.

	Predicate	Unit	Action	Etat
P1	Delay > 199	μs	Notification	Active
P2	Jitter > 19	μs	Re-connect + Notification	Active
P3	Rate-of- loss > 1,98	%	Gold + Notification	Active
P4	Noise > 10	%	Penalties = Penalties+1 %	Active

Table 2. Example of the policies for the VoIP-Platine.

– P1: is a policy which takes care the *time of transmission* parameter, we check if it does not exceed certain thresholds. If the value of the parameter exceed, an alarm is trigger to the provider.

– P2: is a policy that observes *Jitter parameter*. If it exceeds certain thresholds the provider prefers to notify and to reconnect the customer.
– P3: is a policy which takes care the *rate of loss* parameter. If it exceeds certain thresholds, the provider rocks the flow of the customer in the priority flow.
– P4: is a policy that observes the *rate of noise* parameter, if it exceeds certain thresholds, the sum of the consequently penalties augment.

Once created, the contract is stored in a XML-database. An agent is then dedicated to derive the policies from each contract and stores them in the policies database (Policy repository). Then, the PEP (Edge-Router) controls the user access by requesting the PDP (central controller) which responds back with the policies to be applied for the corresponding user. Once policies are stored in the database, the steps of consistency checking and conflict detection are to be done, independently of the technical enforcement policies.

During the use case implementation, we note:

– GXLA is sufficient to describe the services provider environment and the case of the request/negotiation of services. The provider has at the end of the negotiation a concise specification of the contract, with a specification of the suitable policies.
– The fact that we chose XML like language of specification enabled us to benefit from many tools available for the treatment: XML-SPY or Eclipse to edit and validate our specification, XSLT style sheets to convert our XML files into other formats like posting the contract to the customer; JAVA especially the Document Object Model (DOM) library for the parsing and the instantiation of XML files.
– However, future work is essential to define the semantics of the policies. Right now, the policies semantic (parameters, values and actions) are defined informally, witch has turned out to be weakness. It is necessary to define for each service a number of parameters with the corresponding thresholds.

6 Conclusion

In this work, we presented GXLA, an XML-based language which implements the GSLA information model for the specification of the service level agreements (SLA). GXLA is role-oriented, each role includes a set of SLOs and rules which characterize each party's behavior in the SLA. We illustrated the usability of the GXLA through a VoIP service negotiation along with the generation of the full contract script and its enforcing policies. The specification of the GXLA language represents the first step in the automation of service-oriented management. Ongoing work considers the refinement of the policy generation process and the actual configuration of the IT platform to enable the contracted SLAs for real applications.

References

1. Boutaba R, and Al.: Extending COPS-PR with Meta-policies for Scalable Management of IP Networks. (2002) International Journal on Networks and Systems Management (special issue on Management of Converged Networks, vol.10, No.1, pp.91-106.
2. Kamel, H., al.: Designing Scalable on Demand Policy-based Resource Allocation in IP Networks. (IEEE Communications Magazine)
3. Sloman, M., al: The Ponder Policy Specification Language. Proc. Policy 2001: Workshop on Policies for Distributed Systems and Networks, Bristol, UK, 29-31 Jan.2001, Springer-Verlag LNCS 1995, pp. 18-39 (2001)
4. Moses T, and Al.: extensible access control markup langage(xacml). (Feb. 2003) Technical report, OASIS, version 1.0.
5. Ludwig, H., Keller, A., Dan, A., P.King, R., Franck, R.: Web Service Level Agreement (WSLA)
   Language Specification. IBM Corporation (2003)
6. D.Davide Lamanna, J.S., Emmerich, W.: SLAng : A Langage for Defining Service Level Agreements. (2003) IEEE FTDS.
7. Dulay, N., Damianou, N., Lupu, E., Sloman, M.: A policy language for the management of distributed agents. In: AOSE. (2001) 84–100
8. Bajaj, S., Al.: Web Services Policy Framework (WSPolicy). (IEEE Consumer Communications and Networking Conference) IBM, Microsoft, and al. september 2004.
9. Issam Aib, and al.: The Generalized Service Level Agreement Model and its Application to the SLA Driven Management of Wireless Environments. (ACIT 2004)
10. Aib, I., Agoulmine, N., Pujolle, G.: A Multi-Party Approach to SLA Modeling Application to WLANs. IEEE Consumer Communications and Networking Conference (Jan 2005)
11. Aib, I., al.: Capturing Adaptive B2B Service Relationships Management throug a Generalized SLA Information Model. HPOVUA (2004)
12. Heiko Ludwig, Alexander Keller, A.D.R.K.: A Service Level Agreement Language for Dynamic Electronic Services. (IEEE Network 2002)
13. Verma, D.C.: Simplifing network administration using policy-based management. IEEE Network (2002)
14. Sahai, A.: Automated sla monitoring for web services (2002)
15. Machiraju, V., Sahai, A., van Moorsel, A.: Web service management network: An overlay network for federated service management (2002)
16. TeleManagementFORUM: SLA Management Handbook GB917. (juin 2001)
17. Ganz, A., Ganz, Z., Wongthavarawat, K.: Multimedia Wireless Networks: Technologies, Standards and QoS. Prentice Hall PTR (2004)
18. Distributed Management Task Force, I.: Common information model (cim), policy model white paper (18 June 2003) CIM Version 2.7.
19. Network Working Group, R..: Policy core information model (February 2001) Version 1 Specification.
20. IETF NETCONF Working Group: (http://www.ietf.org/html.charters/netconf-charter.html)