AN ARCHITECTURE FOR PROVISIONING IP SERVICES IN AN OPERATIONS SUPPORT SYSTEM

Ajita John[1], Binay Sugla[2], Hari Krishnan, Edwin Park, Arni Raghu, Roshan Sequiera, Ajay Wanchoo
[1]ajita@avaya.com, [2]sugla@dset.com, *DSET Corporation, Shrewsbury, NJ*

Abstract: This paper discusses an architecture for a provisioning system that meets the challenges currently facing service providers in a service and subscriber-based OSS environment. The architecture makes a clear separation between a provisioning core, which is a general framework for provisioning services, and service definitions that model the provisioning view of a service. The architecture is distributed, scalable and extensible and is especially suited for scenarios where a large number of services is expected to be offered, deployed, and managed. The service definitions can be used by the other OSS components to correlate information to provide complete device-to-service-to-subscriber diagnostics for faults, performance degradations, and accounting. It is argued that this approach leads to natural, efficient and effective management solutions.

Key words: Provisioning, IP services, Operations Support System, OSS, VPN

1. INTRODUCTION

The evolution of Operations Support Systems (OSS) from basic device management components to service and subscriber-based ones has come with complexities in widely differing services, standards, and vendors and complexities in interactions between different OSS components. These complexities have made the provisioning and maintenance of IP services such as VPNs, firewall-based security, and web-based hosting a major hurdle. The major issues in provisioning are providing support for multiple services, dependencies among services, multi-vendor equipment, transaction and rollback, seamless interaction with other OSS components, scalability and customization requirements in different service provider environments. This paper discusses these issues and a provisioning architecture to

address them. The architecture consists of service definitions that model the provisioning view of a service and a provisioning core that uses distributed service agents. The service definitions can be used by the other OSS components to correlate information to provide complete device-to-service-to-subscriber diagnostics for faults, performance degradations, and accounting.

With respect to related work, the representation of services has been explored in [1], [2]. Several approaches for provisioning services such as service object-based approach [3], profile-based service provisioning [4] and model-based configuration [5] have been looked at. This work undertakes a comprehensive provisioning architecture for multiple IP services in the context of the current challenges faced by service providers. While the notion of distributed service agents is not new, the contribution of this work is to show how the use of a distributed architecture and an effective decoupling of service definitions and the provisioning core can address the current challenges of IP service provisioning.

2. ISSUES IN PROVISIONING

A provisioning system provisions services on the service infrastructure of a service provider. The service infrastructure includes devices such as web servers, email servers, AAA servers, firewalls and routers. Services may be server-based services such as web hosting or network-based services such as VPNs. The architecture for a provisioning system should support multiple services on a single platform and the easy addition of new services on the same platform. It should be possible to modify the implementation of a service without affecting other services, compose existing services into new ones as in combining an authorization service (from a AAA server), a DNS service, and the provisioning of a web server into a web service. It should also support multi-vendor solutions. A provisioning system should easily interface with other OSS components. The provisioning system forms a key component of a modern OSS by providing the models of a service that should follow well-structured definitions that other OSS components can decipher. IP services such as VPNs may span the domains of multiple providers. The provisioning system for the service provider that accepts service requests should be able to negotiate with the provisioning systems of other service providers, under the agreements they have between them. The provisioning of a service may also require calls to legacy or proprietary systems. The system should be scalable in the number of simultaneous users of the system, the throughput of service requests, the number of supported service infrastructure elements such as routers, and the number of subscribers and services. The provisioning system should also enable transaction support for service requests by rolling back changes to a device if any part of the service request fails. However, the service provider may still want control over the rollbacks and hence it cannot be hard-coded into the system.

3. PROVISIONING SYSTEM ARCHITECTURE

Our architecture is separated into a *provisioning core* that is responsible for handling service requests and *service definitions* that model the provisioning view of

services. The provisioning core interprets service definitions to provision requests for services. A service may be as complex as an MPLS VPN service or it may be a component of a complex service such as the configuration of an edge router or it may be an operational service such as sending email.

The provisioning core is a distributed framework for provisioning. It consists of entities called *service agents*. Each service agent is a provider of one or more services and accepts requests for provisioning these services. A service agent may, in turn, send requests for services to other service agents. Service agents register with a registry to advertise their identity, the services they are providing, and any distinguishing characteristics about the services provided. Depending on the needs of the service provider, there may be several provisioning systems, each composed of a set of service agents, for offering several services. Service agents may be deployed at different locations. For instance, a service agent at the Network Operation Center may request a service agent at a customer premise location to provision the customer's firewall. The communication between the service agents is secure and can traverse firewalls. Communication is asynchronous so that requesting service agents are not blocked until requests are provisioned.

Each service agent exports a set of interfaces for the services it provides. Each interface consists of: (1) a service-independent part containing the type of request (add/delete/edit), subscriber information etc. and (2) a service-dependent part (described below) containing the service parameters that have to be provisioned for the service. Recent industry efforts such as OSSJ [6] can be used for the service-independent part of the interfaces.

Service definitions model the provisioning view of a service by defining its data and process models. The three major components of a service definition are: (1) Service parameters capture the data that comes as part of a service order for that service. Some parameters for a VPN service are the VPN Identifier, the end points of the VPN, and the requested bandwidth. (2) Process model captures the decomposition of a service request into requests for sub-services that are provided by other service agents or the invocation of a software module for the allocation of system resources such as IP addresses. In addition, the process model specifies the transactional semantics for the service: what actions need to take place in the event of a failure returned by the invocation of a sub-service. (3) Service Instance is the information that needs to be stored in a service inventory for a service request after it is provisioned. It includes service parameter values and is associated with the subscriber for whom the service is provisioned.

After receipt of a service request, it is translated into requests for sub-services as specified by the process model. Replies from these sub-requests need to be managed and rollbacks specified by the transactional model may need to be invoked. After completion of this process, a service instance may be created in the service inventory. A workflow engine may be needed to manage the process model if the services provided by the service agent are composed of sub-level services with notions of transactions. Examples of this may be the IPSec VPN or MPLS VPN services.

The advantages of the proposed architecture are numerous. New services can be added without changing the provisioning core. Existing service definitions may be

customized or composed to form new services without changing the provisioning core. The composition of services enables multi-vendor solutions to be supported and enables external provisioning systems to be invoked as a sub-service. Calls to proprietary or legacy systems from the process model of a service can be accommodated. Scalability requirements on a parameter (such as users, network devices) can be met through the distributed architecture where a group of service agents can collectively provide a service. The group may act as a cluster where load balancing over the parameter is done. Modifying the process model can accommodate the need for customization of rollbacks in transactions.

A provisioning system incorporating the proposed architecture and based on the J2EE specification is in place at DSET Corp. The provisioning core has been implemented using java. Communication between the different service agents is done through the Java Messaging System (JMS). The agent and service interfaces are represented using XML and the process model is represented using XSL. The service and subscriber information is partitioned between an LDAP-based directory and a relational database.

4. CONCLUSIONS

This paper discusses the issues in building a provisioning system for multiple IP services in an OSS. It lays out an architecture based on service definitions that model the provisioning view of services and a provisioning core that interprets the service definitions to provision service requests. The paper highlights how a flexible, extensible, and scalable architecture addresses many of the difficult issues faced in building a provisioning system. Additionally, a service definition approach helps other components in the OSS to infer service relationships between the subscribers and service infrastructure. This enables the offer of an end-to-end OSS architecture where faults and degradations from the devices flow into fault and performance management systems that correlate information using the service definitions to report affected subscribers and services.

REFERENCES

[1] Common Information Model (CIM). Core Model, DMTF, 1998.

[2] Serivce Level Management for Enterprise Networks. L. Lewis. Artech House, ISBN I-58053-016-8, 1999.

[3] Provisioning Voice Over Packet Networks: A Metadata Driven, Service Object-Based Approach. Jung Tjong, Prakash Bettadapur and Alexander Clemm. Proc. of the 12th International Workshop on Distributed Systems: Operations and Management DSOM'2001 France, Oct. 2001. Eds: O. Festor and A. Pras.

[4] Profile-based Subscriber Service Provisioning. F. Shen, A. Clemm. Proc. of the 8th Network Operations and Management Symposium, NOMS 2002, April 2002.

[5] Model-based Configuration of VPNs. I. Luck, S. Vogel, H. Krumm. Proc. of the 8th Network Operations and Management Symposium, NOMS 2002, April 2002.

[6] OSSJ: Java Specification for Operations Support Systems. Sun Microsystems.