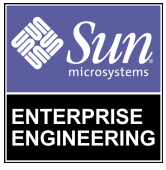
Policy-Based Networks

By Jean-Christophe Martin

Sun BluePrints™ OnLine - October 1999



http://www.sun.com/blueprints

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303 USA
650 960-1300 fax 650 969-9131

Part No.: 806-3718-10
Revision 01, October 1999

Policy-Based Networks

Most networking companies lately have been talking about policy-based networking, but there seem to be many different definitions of what a policy really is. In this months article, we'll explore the network policy concept in greater depth, and see how it is implemented in the Solaris(tm) Bandwidth Manager software.

Overview

Behind the term *policy,* most people put the concept of rules with various degrees of abstraction. To some, a policy might refer to a business policy or to company-wide rules, with a large scope, highly abstracted, and with little detail on implementation. To others, it might refer to low-level rules, like equipment configuration, with a narrow scope and no abstraction at all.

The concept of abstraction, in this context, refers to the ability to define the scope of a policy (or rule) without explicitly describing it. This is the main goal of policies. Therefore, this article assumes that the policies abstract enough details without being too difficult to apply (by being too generic).

The term *policy rule* will be used to refer to a high-level rule, and the term *configuration rule* will be used to refer to the lower level configuration derivative from a policy

Policy Definition

The notion of policy is difficult to define, so everybody seems to have their own definition. However, we can find some consensus:

- It is a rule set governing an entity behavior.
- This set is centrally defined, at an organizational level.
- This set follows a common information model.
- Each rule defines a scope, a mechanism, and actions.

An example of a network policy could be: “If the network resources are low at the end of the quarter, then restrict the WWW traffic.”

The scope of this policy is the network and more precisely, the WWW traffic. The mechanism is bandwidth allocation, and the action is to limit the network resources used by the WWW traffic during certain periods.

From this example, you can see that environmental or situational information is also used to trigger the rule. This information is usually carried by an event coming from the infrastructure or by evaluating conditions.

To summarize, we could say that a policy has the following attributes:

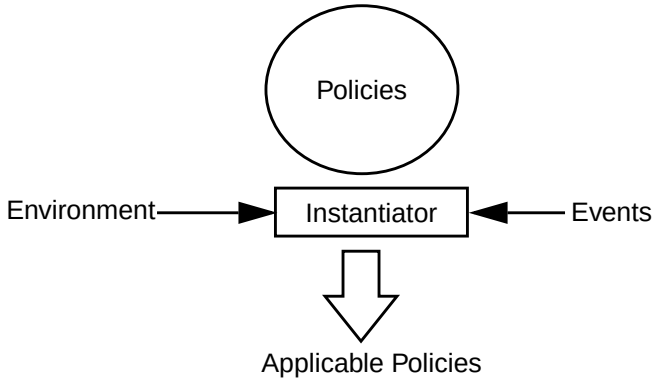
- A scope
- A mechanism
- One or several actions
- A triggering event or condition

Policy to Rule Derivation

Because the policies as defined by the administrator are not directly understood by the equipment or applications, it is necessary to process, or translate, them into configuration rules. At each stage, multiple policies can be in conflict. No simple way of resolving all conflicts has been found. Ultimately, it is the administrator’s responsibility to resolve them. The following steps are recursively applied to each policy until the rule can be understood by the targets:

Instantiation

Based on the previous definition, a policy is a generic rule that must be instantiated. Therefore, the first component is identified is an Instantiator. The policy instantiation can be based on received events or situational information.



The instantiator requires:

- The understanding of the environmental information such as schedules and time frames.
- The understanding of the events received and their impact on the policies.
- The knowledge of the information model.

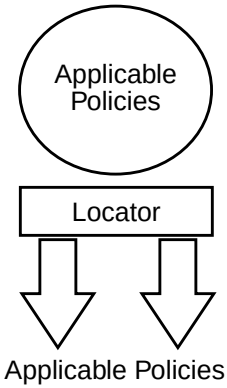
An example of the operations held in this phase are, for example, when a new user is created by an HR application:

- Identification of the New User event and its parameters.
- Selection of the policies impacted by this event.
- Evaluation of the environmental parameters and eventually refinement of the set of policies.

Location

Knowing which policy set is applicable in a given condition is not enough. The process has to find which entities are responsible for enforcing the policy. The location phase, or locator, is responsible for identifying the entities based on

relationship between policies and entities, or between entities. These relationships are defined by the information model. The locator can also use topology information, or data path information to select the targets.



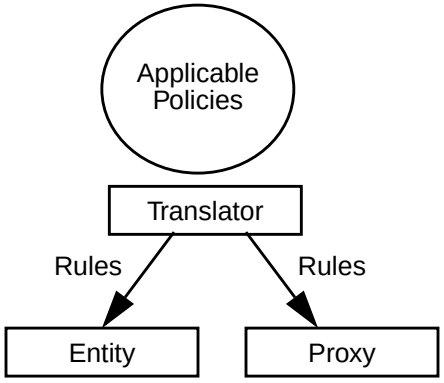
The locator requires:

- The knowledge of the scope of action for each entity (which type of mechanism an entity is able to use). This is sometimes referred to as role.
- The knowledge of the information model and the relationship it defines.
- The topology or data path or flows.
- The possibility to transfer the resulting subset of policies to the right entities.

Translation

When the set of applicable policies has been defined, these general rules must be translated into device / entity specific configuration actions. The next identified component is a Translator. This component is specific for each policy-enabled entity.

For active entities, the translator can be local. In other cases, a proxy service can be provided by the Translator to be able to use protocols like DHCP or SNMP to ultimately deliver the rules to the entity.



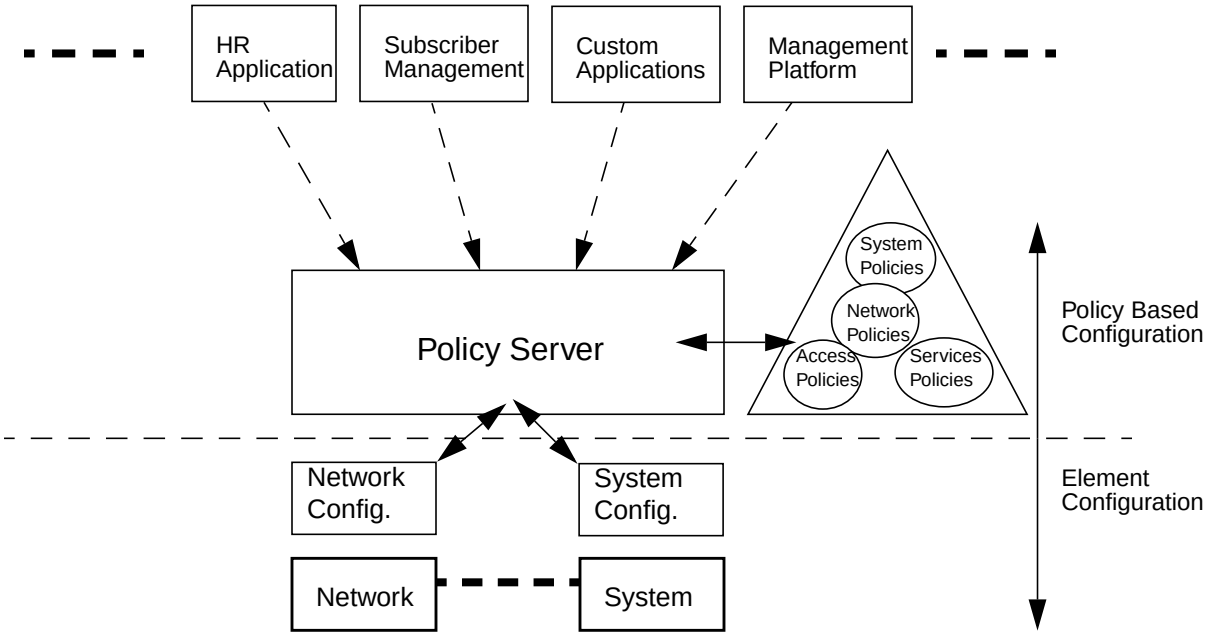
Policy and Directory Services Relationships

It is almost impossible to speak about policies without mentioning Directory Services. This is mainly because it is the natural repository for policies. Here are the main Directory Service features that are supporting policies:

- **It is a common name space**. The Directory Service provides a well defined syntax for objects, as well as a way to identify them with a distinguished name. It also defines a set of allowed operations. The benefit is that policies defined for a component can be reused or shared by other components.
- **It is hierarchical**. It can follow organizational structure and allows object grouping under branches. Hierarchies can be overloaded with specific semantics (departmental, device enclosure, etc). The benefit is that policies can have precedence defined by their position in the hierarchy. Their scope can also be constrained by their location in the hierarchy.
- **It can be distributed**. The Directory Services are, in essence, distributed databases with master and slave concepts. The replication can be total or partial. The benefit is that the policies can be located closer to where they will be applied.
- **It is a central repository**. All objects are located in a virtually unique repository. The benefit is that the policy is managed centrally, from a single point.

- **It already contains objects on which the policies apply**. The Directory Services are used for other purposes like authentication, corporate directory, or subscriber management. The benefit is that the policies can use all the already existing objects as part of their scope and help limit the duplication of information.
- **It allows extensible objects**. Each predefined object, such as a policy, can be extended by sub-classing. The benefit is that standard policies might be defined with room for vendor-specific enhancements without compromising interoperability.

Policy versus Element Configuration



As seen in the above schema, the policy-based configuration sits on top of the actual element configuration, provided that it fulfills some requirements. In no case is it meant to replace the already existing configuration schemas, but instead it enables centralized and distributed configurations.

The policies are stored in a Directory and managed by the Policy Server. The Policy Server is responsible for maintaining consistency between policies and converting the high-level requests or events from the applications into several element configuration modifications.

Policy Benefits

The addition of policy-based management on top of element configuration has the following benefits:

Central Management

By using the directory services, the configuration of multiple instances of the same entity, or the configuration of different entities can be done from a single tool, interfacing through the Lightweight Directory Access Protocol (LDAP) to the policy repository. Therefore, any device or system can be centrally configured with a limited additional cost.

Well-defined Interfaces

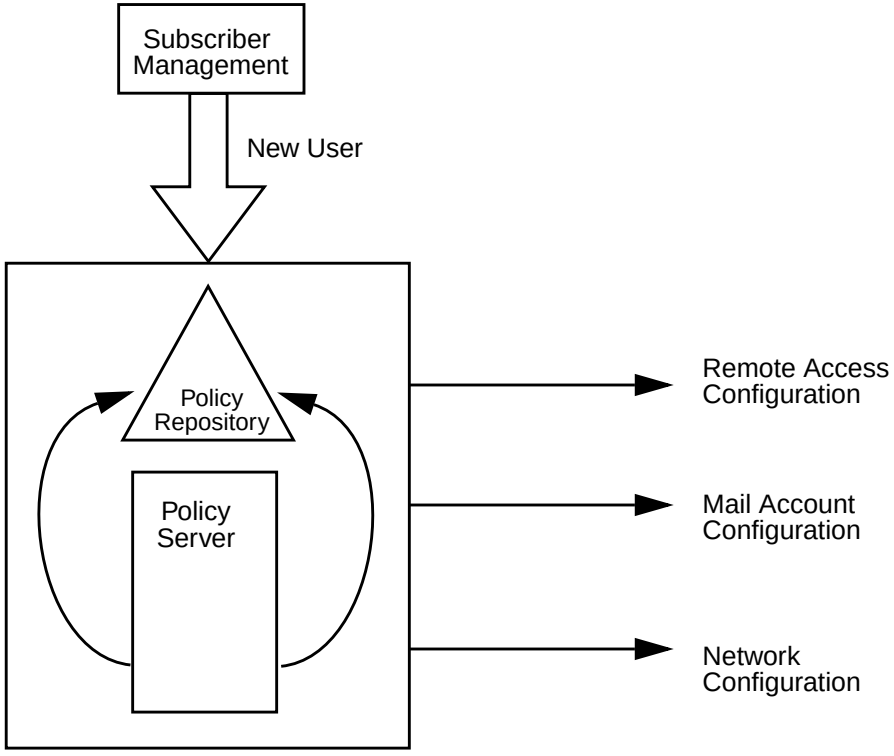
Given that the policies are stored in the directory services, any application could add new policies or look at existing policies. This, for example, could allow subscriber management software to add new customers to the directory and link them to predefined policies based on their Class of Service. It is also a way to help ensure consistent configuration across products using the same objects in their configuration, for example, a traffic description used both by a firewall and a bandwidth manager.

Interoperability

Policy-based management is a way to define standardized element configurations by abstracting vendor-specific parameters. This allows, for example, having a Brand X Policy Manager defining policies used by a Brand Y router, without knowing all the implementation details of the policies.

Added Ease of Use

Since the details of the configuration are hidden to the administrator (but still available if requested), the configuration is easy and identical across multiple system versions or vendor implementations. The configuration is not a component configuration anymore, but a service or logical entity configuration. For example, an ISP's user subscription (including disk quotas, QoS, remote access, mail setup, etc)

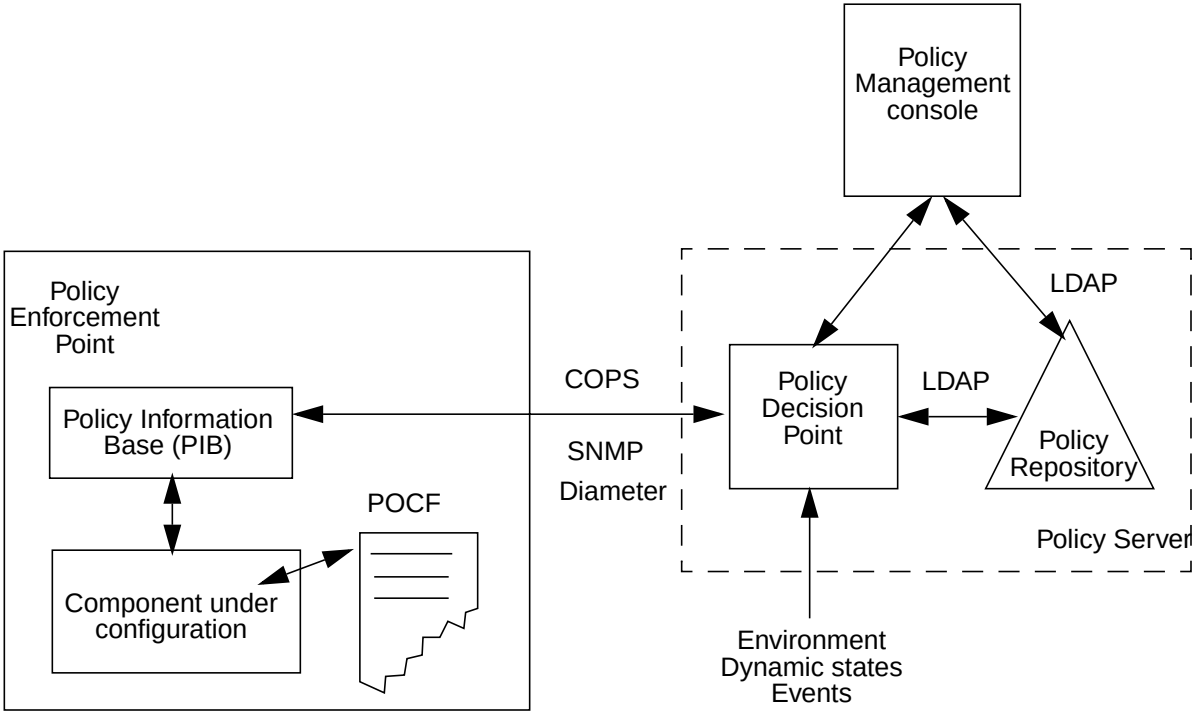


Policy-Based Management Model

Several working groups in the IETF, and the Directory Enabled Networks ad hoc working group (recently absorbed in the DMTF), have defined a model identifying entities involved in policy processing. These entities are applying the "Policy to Rule derivation" steps described earlier.

Overview

The following drawing outlines the various components and the protocols used to communicate between them.



COPS = Common Open Policy Service
POCF = “Plain Old Configuration File”

The Policy Enforcement Point may be physically split in multiple sub-components implemented on different systems.

Policy Repository

The repository of choice is a Directory Server with LDAP access. A schema for storing policies in such directories is under standardization at the IETF and in the DMTF. Basically, a policy is stored as multiple linked objects in the directory, each object has generic attributes and specialized attributes for QoS, Security, etc.

Policy Decision Point (or Policy Server)

The policy server is the nerve center of the policy processing. Its main functions (not in chronological order) are:

- Waiting for configuration requests from COPS or some other protocol.
- Monitoring the environment (i.e. time), listening to events (i.e. new user logged, router down).
- Take input and commands from the Policy Management Console.
- Process policies down to rules (potentially leaving some levels of abstraction).
- Resolve policy conflicts when possible, or report them to the console.
- Deliver rules in a form of PIB elements (Provisioning).

Policy Enforcement Point

The Policy Enforcement Point is where the policies are effectively enforced. This component can be embedded in a smart device or implemented as a proxy in front of a dumb device.The main actions taken by a PEP are:

- Request its configuration to a PDP (provisioning).
- Store this configuration in a PIB.
- Delegate any policy decision to a PDP (admission control).
- Report any error back to the PDP.
- Perform a last instantiation phase on the PIB to derive the actual element configuration.

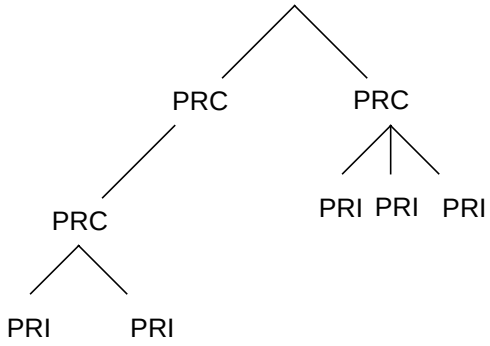
Policy Management Console

Through a graphical interface, the user can enter the policies in the directory and configure specific PDPs. It can be a dedicated console or part of a management console.

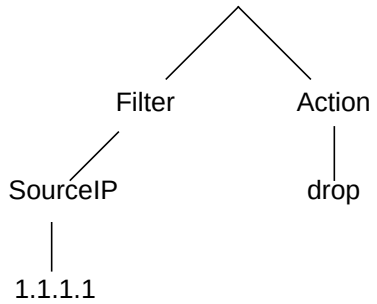
Policy Information Base (PIB)

The Policy Information Base is a local database for policy information. It uses a hierarchical structure similar to tables in SNMP's SMIv2. The branches are called Policy Rule Class (PRC) and the leaves are called Policy Rule Instance (PRI). PRIs

and PRCs are uniquely identified by PRIDs. PRIDs have a hierarchical structure of the form 1.3.4.2.7, where the first part identifies the PRC (e.g., 1.3.4) and the last part identifies the instance (e.g. 2.7).



The definition of what the PIB will look like is not yet final, but as an example, the following policy: “IF source IP address = 1.1.1.1 then drop” could be stored as follows:



The PIB does not duplicate the configuration rules. It must be generic across various entities. It must not contain any instance specific values like incoming interface numbers or process ids.

PDP to PEP Protocol

Multiple candidate protocols distribute policies between the PDP and the PEPs:

- COPS (common open policy service) is specifically designed for policy management.

- SNMP is a general purpose network management protocol that has been improved and can be a candidate for policy management.
- DIAMETER, which is an extension of RADIUS, a protocol for Remote Dial-In User authentication and accounting. DIAMETER is a general purpose protocol with flexible attribute set that can be customized for policy management.

LDAP Schema

Two standard bodies are defining the schema for policy repository. The DMTF (Service Working Group) is taking over the work from the Directory Enabled Networks Ad Hoc working group, and the IETF (Policy Framework Working Group) is working also on a schema specifically for QoS policies. The common parts of the schema are theoretically identical, and should remain identical. The DMTF will concentrate on extending the schema to areas outside of QoS.

A core schema has been defined, and all extensions are derived from the objects defined in the core. In the core schema, a policy is defined as multiple object classes:

- PolicyRule: This class represents the “If Condition then Action” semantics associated with a policy.
- PolicyCondition: The purpose of a policy condition is to determine whether or not the set of actions (contained in the PolicyRule that the condition applies to) should be executed or not
- PolicyAction: The purpose of a policy action is to execute one or more operations that will affect network traffic and/or systems, devices, etc. in order to achieve a desired policy state. Solaris Bandwidth Manager and Policies

Solaris™ Bandwidth Manager Software and Policies

The Solaris Bandwidth Manager (SBM) software has two main parts:

- A kernel module responsible for the network policy enforcement, composed of a packet classifier, scheduler, and marker.
- A policy agent, based on the Java Beans(TM) component architecture model, composed of several policy Beans implementing policies based on various triggering events (Directory Events, Network Events, etc).

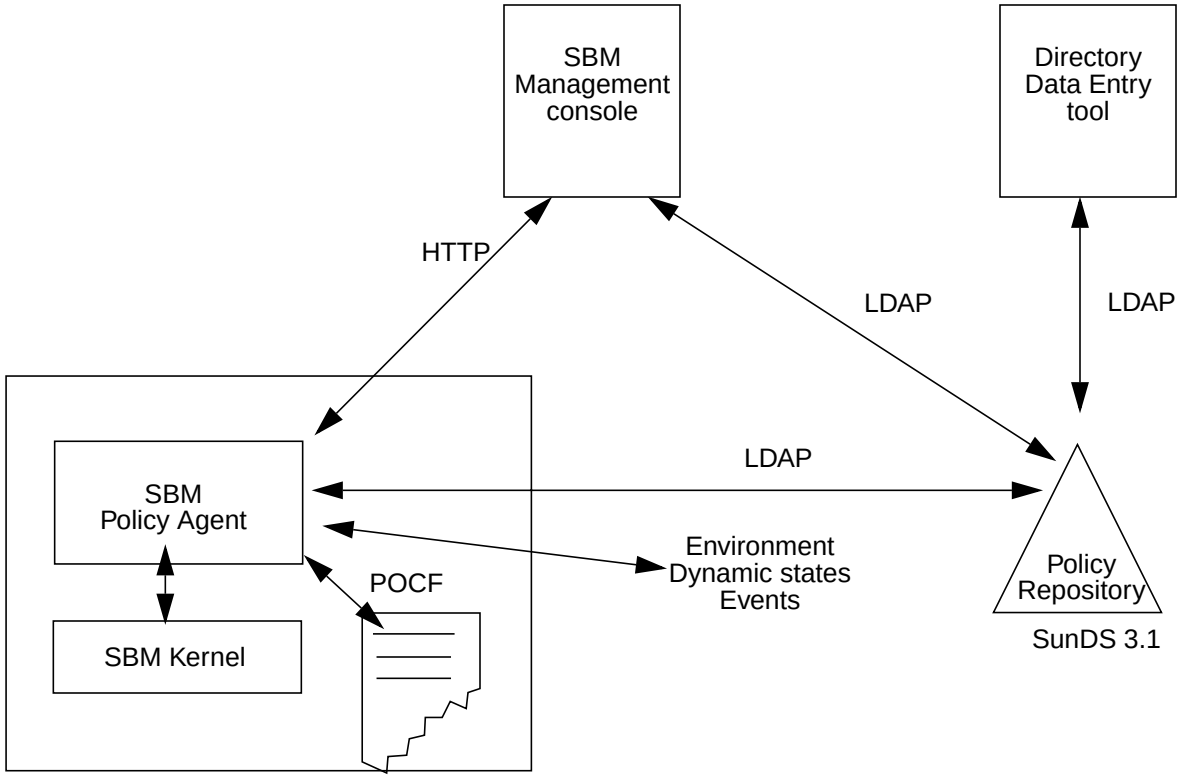
The policy agent is implementing all the steps described previously:

- An Instantiator, selecting the applicable policies.

- A Translator, converting the policies to low level configuration.

The location is not necessary since the target PEP (SBM) is retrieving only the policies that apply to its location.

The following drawing shows the overall architecture:



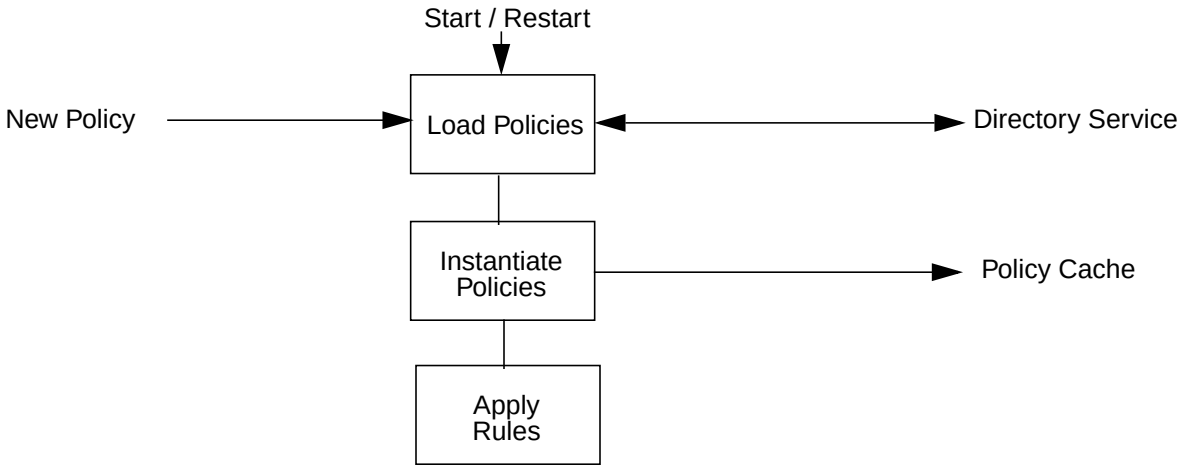
POCF = “Plain Old Configuration File”

The schema used in SBM correspond to the DEN base schema.

This model does not totally match what has been defined today by the IETF, but it is close and already implements many of the new concepts. Note that the SBM policy agent is both the PEP and the PDP.

Resource Provisioning

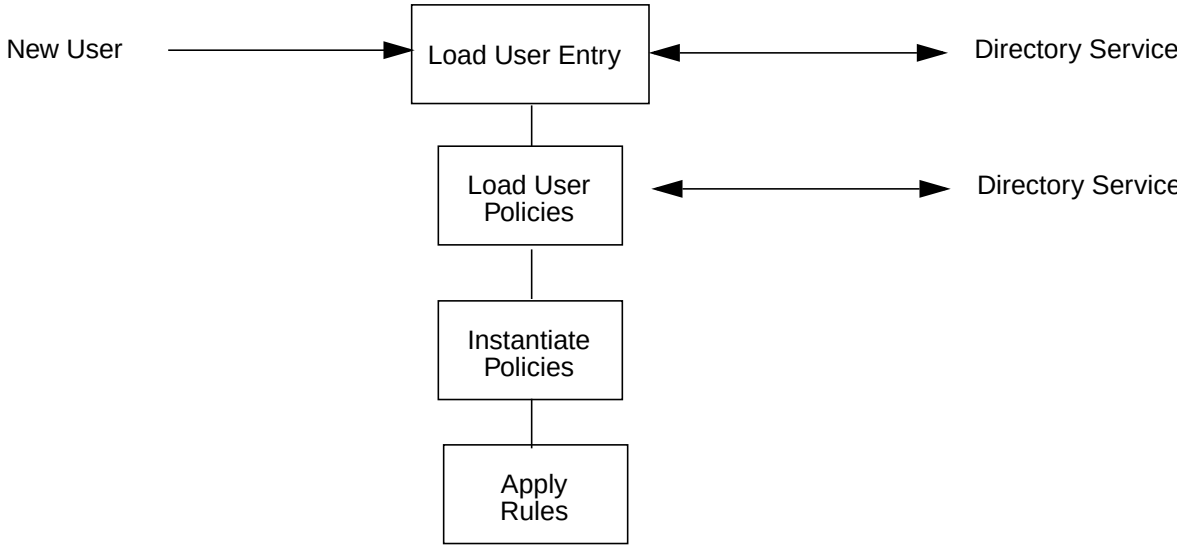
On startup, the SBM policy agent loads a set of policies from the Directory Service. The sequence of operations is described by the following graph:



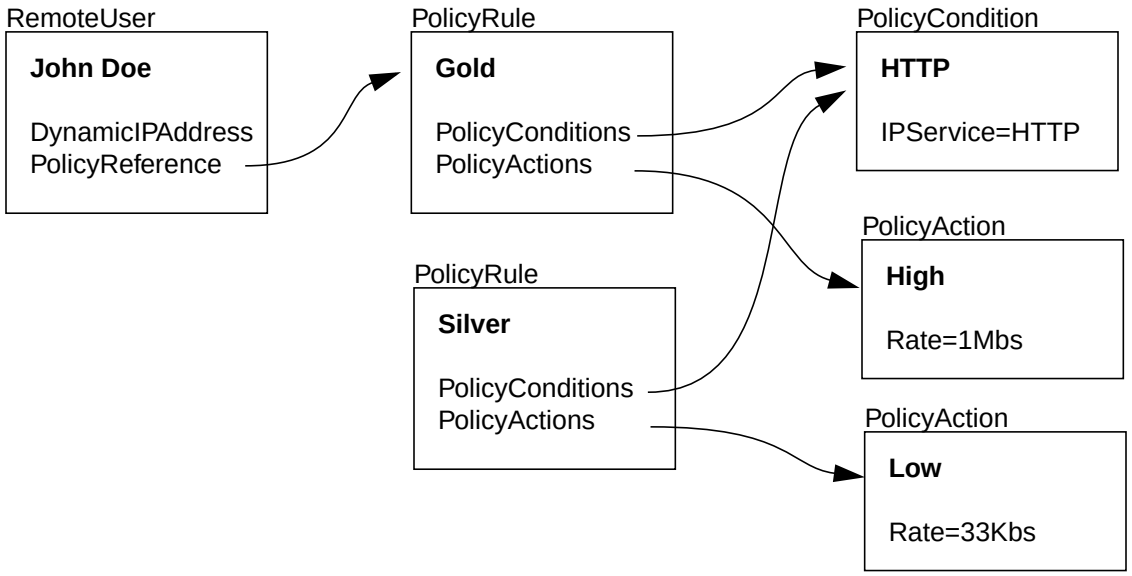
A special mechanism is implemented in the SBM Policy Agent to monitor the Directory Service modifications and to resynchronize the configuration when requested.

Remote Access User Policy

When used in an ISP's point of presence, the SBM policy agent can dynamically instantiate policies for a user connecting. The graph is:



A simplified example of the schema used to implement the policy “if user is John Doe and traffic is HTTP, then QoS is high” is show in the following illustration:



Through the RADIUS protocol, the user John Doe is authenticated with the Directory Service and an IP address is dynamically allocated and stored in the user entry. This entry modification triggers an event sent to the SBM policy agent notifying a new user login. The SBM policy agent retrieves the reference to the policy applicable to the user and instantiates the policies with the dynamic IP address. This is translated into the configuration, and the kernel of SBM is asked to give all HTTP traffic to or from this IP address the rate configured in the action part of the policy.

As only a pointer in the user entry selects the applicable policy, it’s easy to downgrade the user’s QoS just by changing the pointer in its directory entry to the silver policy, for example. Also, it’s possible to change the rate allocated to all gold users just by changing the PolicyAction linked to the gold policy.

Reference Documents

Internet Drafts.

- “A Framework for Policy-based Admission Control”, R. Yavatkar, R. Guerin, D. Pendarakis, 05/17/1999.

- “The COPS (Common Open Policy Service) Protocol”, Shai Herzog, A. Sastry, R. Rajan, Ron Cohen, J. Boyle, David Durham, 02/25/1999.
- “COPS Usage for Policy Provisioning “, R. Yavatkar, Shai Herzog, Francis Reichmeyer, David Durham, 06/29/1999.
- “Policy Action Classes for Differentiated Services and Integrated Services”, R. Yavatkar, George Powers, Dinesh Verma, Michael See,Jean-Christophe Martin, R. Rajan, S. Kamat, R. Chaudhury, 04/06/1999.
- “Policy Framework Core Information Model”, Ed Ellesson, John Strassner, Robert Moore, 05/18/1999.
- “Terminology for describing network policy and services “, Ed Ellesson, John Strassner, 06/29/1999.
- “Quality of Service Policy Information Base”, Keith McCloghrie, Andrew Smith, Scott Hahn, Kwok Chan, Mike Fine, 06/30/1999.

Directory Enabled Ad Hoc Working Group Documents.

- “Directory Enabled Networks - Information Model and Base Schema, Draft v3.0c5. Steven Judd, John Strassner, 08/29/1998.

Author’s Bio: Jean-Christophe Martin

Jean-Christophe is the architect of the Solaris Bandwidth Manager software. He is also an actibve member of the IETF.