IEEE/IFIP Network Operations and Management Symposium (NOMS2002), Florence, Italy, 15-19 April, 2002, pp. 213-218

Tools for Domain-based Policy Management of Distributed Systems

N. Damianou, N. Dulay, E. Lupu, M. Sloman
Department of Computing, Imperial College
180 Queen's Gate,London SW7 2BZ
UK
{ncd, nd, e.c.lupu, mss}@doc.ic.ac.uk

T. Tonouchi
Networking Laboratories
NEC Corporation
Japan
tonouchi@cw.jp.nec.com

Abstract

The management of policies in large-scale systems is complex because of the potentially large number of policies and administrators, as well as the diverse types of information that need to be managed. Appropriate tool support is essential to make management practical and feasible. In this paper we present the implementation of an integrated toolkit for the specification, deployment and management of policies specified in the PONDER language. PONDER policies provide a powerful framework for managing distributed systems which includes explicit domain-based subject and target specifications as well as a flexible life-cycle and deployment model. Domains, implemented using LDAP directories, are used for storing policies and grouping resources, people, and the entities which implement policy, thus facilitating the automated dissemination of policy information. The toolkit presented in this paper comprises: a policy compiler, used to generate implementation code for heterogeneous management and security platforms, a hyperbolic tree viewer for efficient manipulation of the domain structure and effective navigation across the domains, and various tools for deploying and managing the policy life-cycle.

Keywords

Policy Management Tools, Policy Based Management, Domain Based Management, Policy Specification Language, Policy Deployment.

1. Introduction

Policy-based management has become a widely employed and promising solution for managing enterprise-wide networks and distributed systems. It is largely supported by standards organizations such as the IETF and DMTF, and most network equipment vendors. The main benefits from using policy are improved scalability and flexibility for the management system. Scalability is improved by uniformly applying the same policy to large sets of devices while flexibility is obtained by separating the policy from the implementation of the managed system. Policy can be changed dynamically, thus changing the behaviour and strategy of the system, without modifying the implementation or interrupting the system's operation.

The recent emphasis on policy specification [26], information models for managed objects [7] and policy implementation for specific application areas [27, 29, 30], sometimes loses sight of the fact that management, even when policy-based, is an evolutionary process. Policy-based resource allocation, the association between policy and the devices on which it must be implemented and even the policies themselves are subject to frequent reviews and changes. To facilitate these activities it is necessary to use a policy language easily understandable by human administrators in conjunction with an integrated toolkit for the deployment, enforcement and coordination of policies within the system. The toolkit must further facilitate the policy specification and permit easy per user/per device review of policy.

PONDER [5, 6] is one of the few languages for specifying both security and management policies. The language is declarative and simple to use for human managers. Its design and deployment model are based on domain-based policy management [8] where policies apply to domains of managed objects. Domains are hierarchical and are similar to directories. They group objects according to various criteria such as geographical boundaries, object type, responsibility and authority [23]. The benefits of the domain-based approach are twofold: *i)* a policy applying to a domain will propagate to its sub-domains thus applying to large numbers of objects and providing scalability and, *ii)* when new objects are added or removed from the system they can simply be included or removed from relevant domains, without the need to modify the policies or manually manage the association between policy and managed objects.

Management of enterprise systems requires an integrated but decentralized administration of resources, people and corporate policies within large and complex organizational structures. Decentralized administration is both difficult and error prone. Administrators need to be isolated from the details of the underlying implementations and policy representations. Tools with the required abstraction and flexibility must allow for integrated administration of the structures and of the diversity of management information. They must provide support for specifying new policies and modifying existing ones, instantiating existing policy types, generating suitable code from the high-level policies, analysing policies for conflicts and managing the policy life-cycle.

In this paper we present a toolkit for specifying and managing PONDER policies which satisfies the requirements mentioned above. In conjunction with the language, the toolkit permits integrated administration of resources, people and policy information with automated policy deployment. The toolkit comprises a generic domain management tool, an Integrated Development Environment (IDE) with a policy compiler, as well as tools for managing policies and roles at runtime. We present the tools through a simple scenario of a backup and archiving system for a computer research institution. The following section describes the scenario. Section 3 describes the domain browser, and the policy IDE. In section 4 we describe tool support for managing the policy life-cycle, and section 5 presents support for role-based management. We include related work in section 6 followed by conclusions and further work.

2. Scenario

In this scenario, we consider the management of a backup and archiving system for a research institution with many autonomous operating units corresponding to the departments (inspired from a scenario in [11]). The central backup and archiving servers perform periodic backups (e.g., once a month) but departments may have their own backup servers for more frequent use (e.g., every evening). File servers are available for each department and it should be possible to specify backup strategies for individual users, based on their requirements (frequency, what data to backup, life of backed-up data).

It is important that the administration of managed data can be delegated on a hierarchical basis. Domains can be used to model a hierarchical structure of the system, which will also enable policy specification, and logically centralised system administration. Figure 1 shows a partial view of a domain structure for the system. The /staff subtree contains the system administrators as well as research and development staff subdivided by division and department. The /staff/admin subtrees are also subdivided by department, to reflect the fact that certain administrators are responsible for certain departments and users. The /system subtree contains the resources (files and data) partitioned to reflect departmental structure. It also contains the system servers (e.g. backup, account, mail etc). Finally, the /managementInfo subtree is used to group policies and other management specific information. Only policy administrators (/staff/admin/policy) and security administrators (/staff/admin/sec) are permitted to access management information.

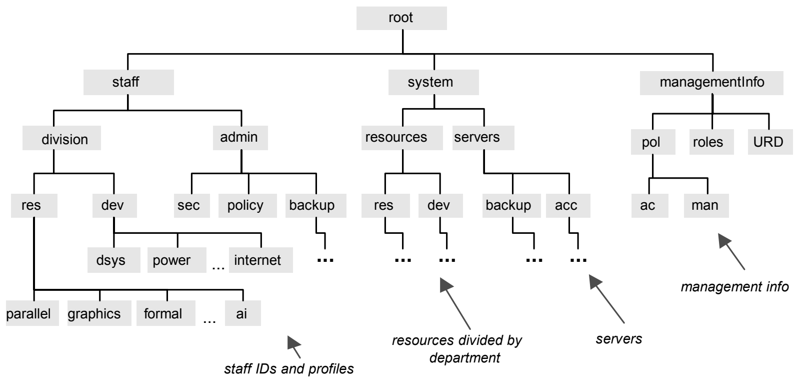


Figure 1 Partial Domain Structure

Security is an important aspect in the system. Authorized users are able to access their own files, those shared in their department, as well as those to which they have been explicitly granted access. The account servers hold account information for each user in the system. Users are not normally allowed to execute actions that are carried out for them by the administrators (e.g., regular backups). However, administrators can delegate those actions to the users they administer. The management system includes role-based management features. An account manager role is specified which defines the authorization and obligation policies associated with an account security manager. Similarly a backup administrator role is defined to which backup administrators are assigned.

3. Policy Administration Toolkit

The PONDER system architecture, shown in Figure 2, is derived from experience learnt from work on Policy-Based Management at Imperial College over the past 15 years. An initial notation for policy specification was described in [18], and an implementation of an agent for interpreting obligation policies was documented in [19]. Previous attempts at implementing a domain browser for navigating the domain structure have been presented in [9], while a simplified template-based policy editor can also be found in [18]. A separate standalone editor for specifying and viewing roles and relationships was presented in [16]. None of these papers related to the current version of the PONDER policy specification language which introduces modern object-oriented concepts into policy specification as described in [5] & [6]. An approach to deployment of policies for distributed management was outlined in [8]. In this paper we use the experience gained through earlier attempts at implementing those different standalone tools and components, and present an integrated toolkit to support the whole policy life-cycle relating to specifying and managing deployed policies. We combine the ideas presented in previous papers to implement a complete policy-based management platform which includes a comprehensive user interface built around a hyperbolic-tree based domain browser. The browser permits users to navigate through complex domain hierarchies of both policies and managed objects. The architecture also includes a compiler with various multiple 'back-ends' to generate, XML policy representations as well as management and security policy enforcement components targeted at many different platforms.

A Lightweight Directory Access Protocol (LDAP) server is used to implement the domain service, and Java RMI as the middleware for communication between the various system components. All tools are implemented in Java, and Swing is used for the graphical user interfaces. The enforcement components include Policy Management Components (PMCs) responsible for enforcing obligation and refrain policies, and Access Controllers (ACs) responsible for enforcing authorizations. The policy service is used to manage and coordinate access to policy objects stored in the domain service, and to instantiate new policies from existing policy types. The policy service creates a Policy Control Object for each policy which is distributed. The execution of policy life-cycle operations occurs through direct interaction with these control objects, which are also bound to the directory for persistence. For a more detailed description of the policy deployment model see [8].

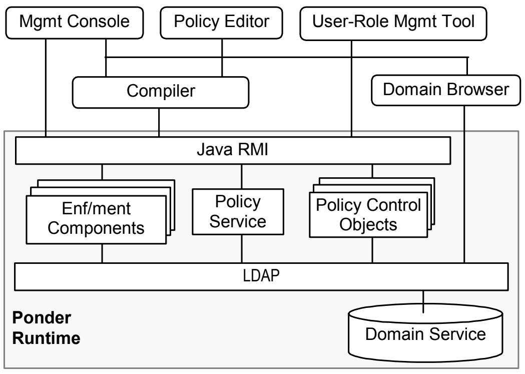


Figure 2 Management System Architecture

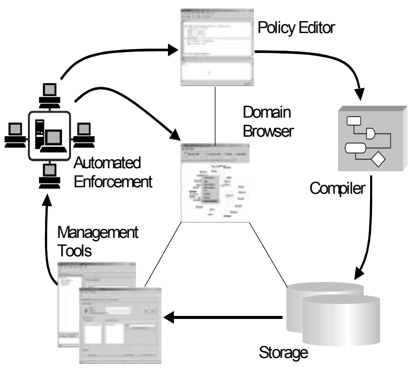


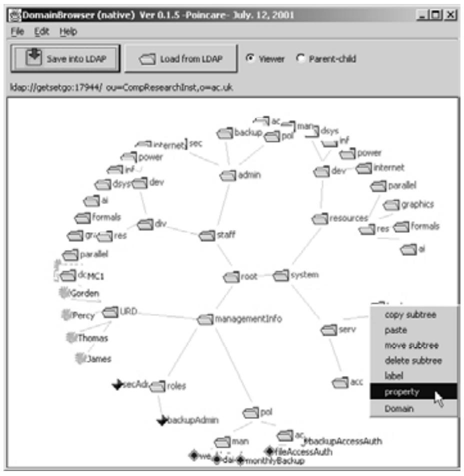
Figure 3 Policy Management Cycle

The domain browser provides a common user interface for all management interaction with objects stored in the domain service. Other tools interact with the domain browser to select objects from the domain service. Figure 3 shows the steps involved in managing a policy-based system. Policies and roles are created using the policy editor, compiled and stored in the domain service. The management console and user-role management tools can then be used to distribute policies and to activate/deactivate roles. The distribution and enforcement of policies is automated and the tools can be used in a distributed manner by a number of administrators.

The domain service is implemented using an LDAP directory with extensions to allow objects to be members of multiple domains. Thus, LDAP is used for both storing policies and for grouping subject/target objects whereas the IETF framework [27] uses directories only as policy repositories.

3.1. Domain Browser

The domain browser provides a common user interface for all aspects of an integrated management environment. It can be used to group or select objects for applying policy, to monitor them or to perform management operations, although the current implementation only supports policy management. The domain browser reads data from the domain service and provides a graphical tree-structured view of the directory structure. Usability is enhanced by customising pop-up menus according to the type of the object being selected. Furthermore, external tools can be invoked from within the domain browser for a specific managed resource or policy, depending on the current selected context. External tools also interface with the domain browser to allow for navigation or selection of objects from the domain service. For example, it is possible to specify a policy's subject and target domains by selecting them from the domain browser. The domain structure for the research institution of the scenario in section 2, is created using the domain browser as shown in Figure 4. Administrators can use the domain browser to manage the domain structure, group objects into domains to apply a common policy, modify or create new objects. Objects can represent users, roles, network components or manager agents.



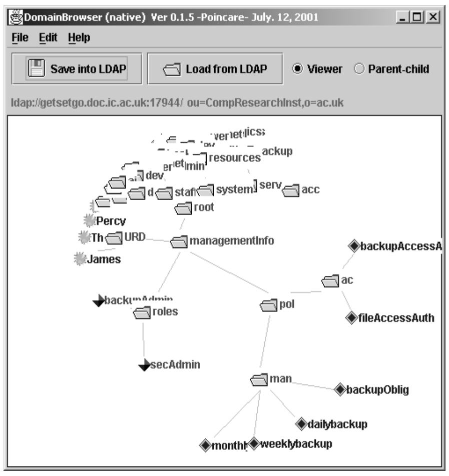


Figure 4 Domain Browser

Figure 5 Focusing a sub-tree

The domain structure can be very large, both in terms of number of objects within a domain as well as depth of the hierarchy. Thus, the challenge is to realise a tool which enables users to easily visualise and navigate the structure. We adopt a hyperbolic tree-mapping algorithm [15] which has two characteristics. First, the hyperbolic plane is a non-Euclidean geometry in which parallel lines diverge away from each other. Therefore, the circumference of a circle on the hyperbolic plane grows exponentially with its radius, which means that exponentially more space is available with increasing distance. In a 600 by 600 pixel window, a standard two-dimensional hierarchy browser can typically display 100 nodes whereas the hyperbolic browser can display about 1000 nodes, while providing more effective navigation of the hierarchy. The second characteristic guarantees that every sub-tree can be mapped congruously; the central angle of the sector that every sub-tree occupies is the same. The domain browser can thus display any part of the tree uniformly. This gives users a better feel of the entire domain structure, making it easier to perceive the context. We have experimented in the past with two-dimensional tree viewers, and have found it very difficult to display large domain structures. The domain structure of PONDER is not a pure tree but an acyclic graph [24] as it permits objects and sub-domains to be members of multiple parent domains.

Navigation is realised by moving the domain tree around the hyperbolic plane. Objects nearer to the centre of the display are enlarged and come into focus. For example, an administrator can focus on the policies sub-tree of the domain structure by selecting the /managementInfo/pol sub-tree and dragging it near the centre of the viewer, as shown in Figure 5. Sub-trees can be collapsed or expanded providing further flexibility in navigating large structures.

PONDER has composite policy structures (roles, relationships, management structures), which consist of sub-component policies. The domain browser displays a composite policy as a domain or sub-tree, whose children are the component policies.

Edited domain structures can be saved into the domain service or discarded by undoing the operations. Similarly, the latest domain structure information can be loaded from the domain service. The browser registers for update notifications with the domain service and the "Load" button is highlighted when the browser receives a notification that the domain structure has been modified. This function enables the cooperative use of the tool by several users.

We provide two implementations of the domain browser with the same API: one uses a program module we developed, and the other uses the Inxight Star Tree Software Development Kit [13]. Although the current version uses LDAP directories as the information repository, the browser implementation is not dependent on LDAP and data can be loaded from other sources as well.

3.2. Compiler Framework

The PONDER policy language provides reuse by supporting the definition of policy types to which any policy element can be passed as formal parameter. Multiple instances can then be created and tailored for a specific environment by passing actual parameters. We refer to policy instances as policies in this paper. PONDER policies can be mapped to low-level representations suitable for the underlying system or into XML for transfer around the network. Authorization policies can be mapped onto a variety of heterogeneous security platforms and mechanisms, such as firewalls, operating systems security, database security and Java authorizations. For example, if servers used to store data in the AI research group are Linux based while servers in other departments are Windows 2000 based, then appropriate code will be generated based on the type of server.

Dedicated code generators (compiler back-ends) must be implemented to translate the PONDER specification into the desired format. The compiler framework is designed for extensibility with custom code generators without recompiling the system.

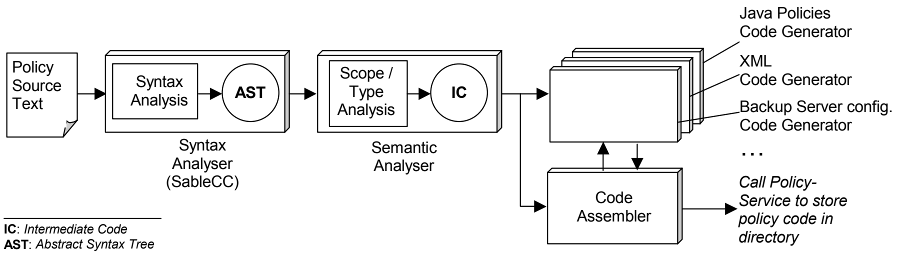

Policy Source Text
Syntax Analysis
AST
Syntax Analyser (SableCC)
Scope / Type Analysis
IC
Semantic Analyser
Java Policies Code Generator
XML Code Generator
Backup Server config. Code Generator
...
Code Assembler
Call Policy-Service to store policy code in directory
IC: Intermediate Code
AST: Abstract Syntax Tree

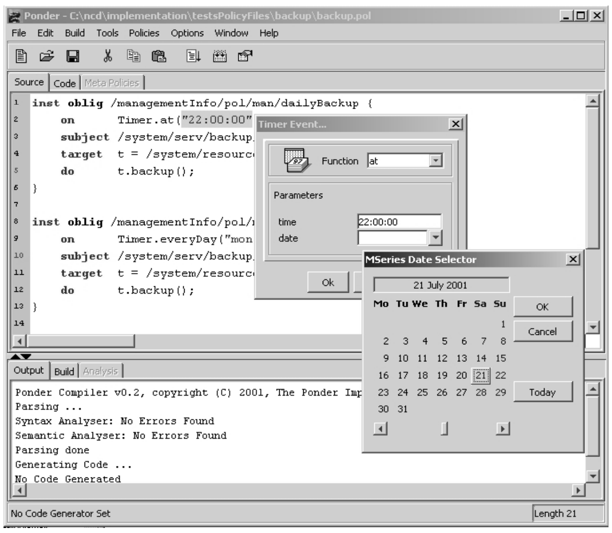
Figure 6 Compiler Framework

Figure 6 shows the main modules of the compiler which is based on a LALR(1) parser generated with SableCC [10], an object-oriented Java parser generator. The main phases of the compiler generate an intermediate code which is then passed on to all the code-generators added to the compiler. The code assembler module is responsible for coordinating the code generation phase, and for storing the generated code for a given policy in the directory service under the appropriate domain. Its implementation is specific to the underlying domain service. A Java code generator is included by default, which generates a Java class for each of the basic policies as defined in the deployment model for the PONDER language [8]. Preliminary implementations exist for translating PONDER policies onto various access control platforms. These include:

- A Java back-end which transforms PONDER authorization policies into access control policies for the Java platform. This has required several extensions to the Java security model in order to enable run-time PONDER policy evaluation, constraint checking and filtering [4].
- Code generators for translating PONDER authorization policies to Windows 2000 security templates and Firewall rules.
- Experimentation with mapping PONDER authorization policies to Linux access controls. System level scripts have been specified to program the Linux security kernel. A code generator translates PONDER policies into calls on those scripts.

3.3. Policy Editor

The policy editor tool (Figure 7) is integrated with both the domain browser and the PONDER compiler and provides an easy to use development environment for specifying, reviewing and modifying policies. Templates can be used to create policies easily. The domain browser can be invoked to select the subject and target domains for policies. Existing policies and policy types can be selected from the directory with the aid of the domain browser, loaded into the editor, modified, recompiled and stored back to the directory. Code generators added to the compiler framework, are accessible and can be enabled from within the editor to select the type of code to be generated.


Ponder - C:\ncd\implementation\testsPolicyFiles\backup\backup.pol
File Edit Build Tools Policies Options Window Help
Source Code Meta Policies
Timer Event...
Function at
Parameters
Ok
MSeries Date Selector
OK
Cancel
Today
Output Build Analysis

```
inst oblig /managementInfo/pol/man/dailyBackup {
    on      Timer.at("22:00:00"
    subject /system/serv/backup
    target  t = /system/resourc
    do      t.backup();
}

inst oblig /managementInfo/pol/
    on      Timer.everyDay("mon
    subject /system/serv/backup
    target  t = /system/resourc
    do      t.backup();
}
```

time 22:00:00
date

21 July 2001

Mo	Tu	We	Th	Fr	Sa	Su
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

```
Ponder Compiler v0.2, copyright (C) 2001, The Ponder Imp
Parsing ...
Syntax Analyser: No Errors Found
Semantic Analyser: No Errors Found
Parsing done
Generating Code ...
No Code Generated
```

No Code Generator Set
Length 21

Figure 7 Policy Editor

4. Policy Life-Cycle Management

4.1. Policy Dissemination

Policy types are compiled into Java policy classes by the PONDER compiler, and stored in the domain hierarchy. Instantiation of a basic policy type creates a Java policy object in the domain hierarchy to maintain a suitable representation of the policy attributes. Further, a policy control object is created to coordinate run-time access and dissemination of the policy: an authorization control object (ACO) for authorization policies, a refrain control object (RCO) for refrains and an obligation control object (OCO) for obligations.

Once instantiated, a policy object can be *loaded* into its enforcement components, and once loaded, it can be *enabled.* An enabled policy can be *disabled* and later re-enabled, or disabled and then *unloaded* from its enforcement components. Unloaded (i.e. dormant) policies can be either re-loaded or deleted. Enable/disable are less expensive to implement when a policy is frequently stopped and later restarted. The policy control object co-ordinates these life-cycle policy operations, and acts as a centralized control point for managing concurrent and possibly conflicting requests from multiple policy administrators and from domain objects to which the policy applies. The control object does not participate in the policy enforcement process. Replication of policy control objects is possible, to enhance scalability.

Dynamic Changes to Subject/Target Domains and Objects

PONDER policies operate over sets of objects formed from domains using domain scope expressions (i.e. set expressions). Domains however, are not static, and when an object is added to a domain, loaded or enabled policies applying to that domain will need to be loaded and enabled on the new object. Thus, domains must maintain references, held in a multi-value attribute, to the policies applying to them. When a policy is loaded, its control object updates the entry of all domains to which it applies.

We use the Java Naming and Directory Interface (JNDI) event listener functionality to generate events on domain membership changes, which are sent to the policy control objects so that policies are added or removed accordingly from the enforcement components.

In the archiving scenario, the following policy applies to backup servers in the research division. The servers must perform a self-check of available disk space on the 1st of each month. When a new server that is installed in the system, is added under the /system/serv/backup/res domain, this policy and all others applying to the domain are automatically loaded and enabled on the server.

```
inst oblig selfDiskCheck {
        on        Timer.at("23:00:00", "01:*:*");
        subject   backServ = /system/serv/backup/res;
        do        backServ.checkDisk(); }
```

4.2. Management Console Tool

We have implemented a management console tool (Figure 9) for dynamically managing policies. The steps involved in using the tool in relation to the policy life-cycle are demonstrated in Figure 8. The tool has two main views:

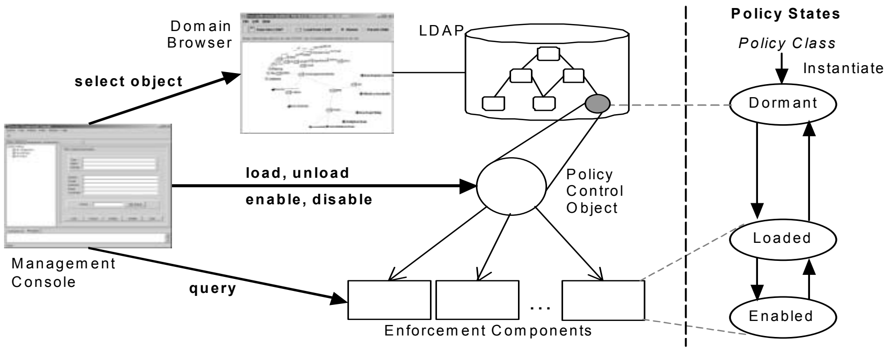
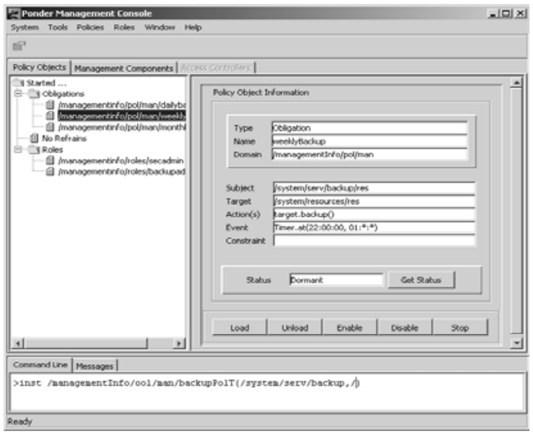


Figure 8 Managing the Policy Life-Cycle

In the **Policy Objects View**, a policy instance can be selected from the directory (using the domain browser) and loaded into the management console. Similarly, if a domain is selected all policy instances under that domain will be loaded into the management console in an expandable tree-navigator. Policies can then be selected and *loaded*, *unloaded*, *enabled* or *disabled* as needed. Details about the selected policy are displayed including the policy-status. When a new backup policy for a specific user is specified, a policy administrator uses the management console to select the policy from the directory, load it and enable it. Multiple management consoles could manage the same domain of policy objects, but LDAP does not support concurrency control.

In the **Enforcement Components View,** enforcement components can be selected and information about the policies loaded into them is displayed in a tabular format.

A **Command-line Window** can be used to type single-line commands to the PONDER compiler. This allows interactive instantiation of policy types.


Ponder Management Console
System Tools Policies Roles Window Help
Policy Objects
Management Components
Access Controllers
Started ...
Obligations
/managementinfo/pol/man/dailyb
/managementinfo/pol/man/weekl
/managementinfo/pol/man/monthl
No Refrains
Roles
/managementinfo/roles/secadmin
/managementinfo/roles/backupad
Policy Object Information
Get Status
Load
Unload
Enable
Disable
Stop
Command Line
Messages
Ready

Type: Obligation
Name: weeklyBackup
Domain: /managementInfo/pol/man
Subject: /system/serv/backup/res
Target: /system/resources/res
Action(s): target.backup()
Event: Timer.at(22:00:00, 01:*:*)
Constraint:
Status: Dormant

```
>inst /managementInfo/ool/man/backupPolT(/system/serv/backup,/)
```

Figure 9 The Management Console Tool

4.3. Conflict Analyzer

Having both positive and negative policies may result in conflicts which may lead to inconsistent or deadlock states within the system. A user-specified backup policy may conflict with backup policies specified by the administrator. In large-scale systems, where several administrators specify policies, conflict detection should be automated. We have implemented a conflict analysis tool, integrated with the policy editor, to perform static analysis of policies [17]. The tool also allows various forms of precedence to be specified to resolve policy conflicts. For example precedence can be given to negative policies, more specific ones or more recent ones. Policies stored in a sub-tree of the domain structure can be analyzed for conflicts or analysis can take place when storing new policies to see if they conflict with existing ones.

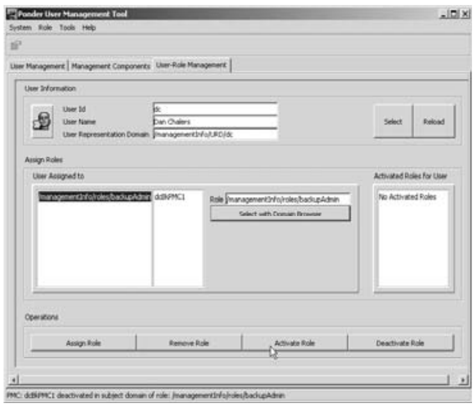
5. Role-based Management

In PONDER policies pertaining to a position in an organization have a common subject domain and can be grouped into a role such as: operator, security administrator or a nurse in a hospital [6]. Roles can also be used to group policies for a particular type of automated management agent such as backup agent. Roles can be specified using the policy editor and then compiled and stored in the domain service as a composite role object c.f. a domain. Policies inside the role are stored as subentries of the role. This allows navigation of the role contents in the same way that domains are navigated. Instantiating a role type creates a new object for the role instance, and instantiates all the policies contained in the role type.

Roles provide a grouping and abstraction mechanism which simplifies management and can be selected from the domain service using the management console in the same way as policy objects are selected. Control operations can then be performed on the role object, without needing to access the individual policies of the role. The corresponding role control object takes care of loading, unloading, enabling, or disabling the policy instances inside the role to all the enforcement components.

5.1. Role Assignment

An important aspect of role-based management is that of assigning users to roles, and activating/deactivating roles. A *user representation domain* (URD) is a persistent representation of a human user in the system which contains two different kinds of objects: one or more policy management components (PMC's) and a user profile object (UPO). PMC's can act as proxies for the user to allow access to resources permitted by the authorization policies or they can act as automated agents, interpreting obligation policies on the user's behalf. A user is assigned to a role by including a PMC from the user's URD into the subject domain of the role. PMCs are application specific and different PMCs can be used to support security management, backup administration etc. if the user is assigned to different roles. The UPO contains information about the roles and domains to which a user's PMCs have been assigned. Note that automated agents implementing policies are also PMCs which can be assigned to roles but do not represent a human user.


Ponder User Management Tool
System Role Tools Help
User Management | Management Components | User-Role Management
Select
Reload
Select with Domain Browser
Assign Role
Remove Role
Activate Role
Deactivate Role

User Information
User Id: dc
User Name: Dan Chalers
User Representation Domain: /managementInfo/URD/dc

Assign Roles
User Assigned to: /managementInfo/roles/backupAdmin ddBKPMC1
Role: /managementInfo/roles/backupAdmin
Activated Roles for User: No Activated Roles

Operations

PMC: ddBKPMC1 deactivated in subject domain of role: /managementInfo/roles/backupAdmin

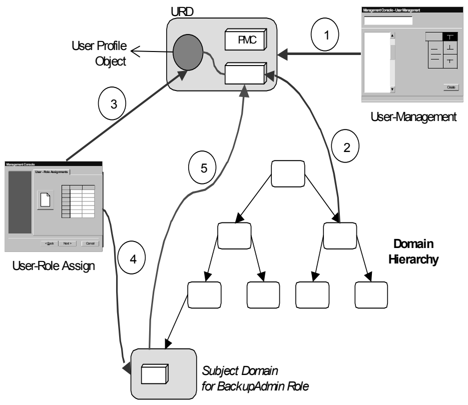


Figure 10 User-Role Management Tool **Figure 11** User-Role Management Steps

Figure 11 shows the steps involved in assigning users to roles. These steps are summarized below in relation to the user-role management tool (Figure 10), which is used to manage users for which policies are specified in the system. The tool consists of three sub-views:

In the **User-Management View** new users can be created, assigned and removed from domains. Creating a user means, creating a URD for the user, and a UPO, which is stored inside the URD. Assigning a user to a domain implicitly creates a reference in the selected domain, which points to a policy management component (PMC) within the URD. This corresponds to steps 1 and 2 in Figure 11.

In the **Management Components View** new policy management components can be created (or deleted) for users and stored in the users' URD. This is also related to step 1 of Figure 11. PMCs stored in domains (URDs or other domains), can be started and stopped using the user management tool remotely onto any host. This requires a PMC-Server to be running on the remote host. For example, when a new backup administrator for the research division of our scenario is added to the system, a corresponding URD will be created under /managementInfo/URD, and a backup-enabled PMC will be instantiated in the URD. The administrator is then assigned to /staff/admin/backup/res, and his PMC is started on the research network.

In the **User-Role Management View** users can be assigned and removed from roles. This simply updates the list of roles to which the user is assigned within the UPO (steps 3 and 4). A PMC representing the user in the role must then be selected. Roles assigned to a user can be selectively activated and deactivated. Activating a role creates a reference in the role's subject domain which points to the PMC in the user's URD (step 5). This implements the RBAC concept of sessions [22]. The new backup administrator in our scenario can be assigned to the backupAdmin role by selecting from the domain browser the appropriate URD, PMC and the backupAdmin role. The role can then be activated and deactivated with a click of a button by selecting it from the list of roles assigned to the user.

6. Related Work

Unlike the PONDER framework, which covers a wide range of policies with customized enforcement, the majority of existing policy-based tools for networks and distributed systems management concentrate on a specific area, primarily quality of service or access control management. Verma [29] describes a QoS tool used to specify Service Level Agreements (SLAs) and to manipulate SLA related information in a tabular format. The tool transforms high-level policy information into device configurations, and stores them in an LDAP directory. Another tool, presented in [20], focuses solely on template-based refinement of policies from high-level goals.

Existing work within the RBAC community is limited to specifying access control configurations in terms of roles. A centralized tool, presented in [28], translates access control configuration from the RBAC framework to the target's native security mechanism, which is then transported to the target. Another web-based tool, presented in [2], allows administrators to specify roles, role hierarchies and constraints.

In Policy Based Networking most of the tool support comes from industry and is based on the IETF policy framework. The majority of these tools are specific to quality of service management [3, 12, 14, 21] but some also include access control configuration for routers, switches and firewalls [1, 25]. The user interface usually comprises a policy-editing tool, which uses a tabular view of policies. Although visual tools are included by some vendors, most of this work focuses on managing individual network elements. Scalability to enterprise wide management is not obvious as the dissemination of policies to specific elements is performed manually whereas in PONDER this is automated based on domain membership. For comparison of PONDER with other policy specification languages and approaches see [5].

7. Conclusions and Further Work

In this paper we have presented a prototype implementation of a management toolkit, for integrated management of distributed systems. The toolkit is based on the use of domains which support scalability in that policies can be specified for hierarchical groups of objects and deployed automatically. We have used the experience gained through earlier attempts at implementing various tools and components of the architecture, to provide for the implementation of a policy-based management platform. We have experimented with various forms of domain visualization as the traditional 'Windows Explorer' view does not scale for large acyclic graphs of directories/domains, although it is useful for small sub-trees. The hyperbolic tree-based view gives a 'fisheye' focus onto specific sets of domains and permits easy navigation of very complex structures. We think that integrating all management tools around the domain browser to give a common 'look and feel' is the right approach but so far we have only concentrated on tools for specification, analysis, dissemination and control of policy.

PONDER supports a number of composite policy concepts [5, 6] although only roles are discussed in this paper. Being able to specify all the policies relating to a position in an organization or for an automated agent and then creating multiple instances of these roles, provides a powerful mechanism for specifying policies in large-scale systems. Composite policies are viewed in the same way as domains with the domain browser and both composite policy types and instances can be easily opened and controlled.

The PONDER deployment model is implemented using Java and LDAP and uses JNDI to interface with LDAP directories. We are currently working on extending the domain browser with an optional explorer-like view for selected domains. This will enable easier selection and manipulation of objects inside specific domains. Extensions to the domain browser include drag-and-drop functionality to assign users to domains and roles, and optionally viewing relationships between policies, such as delegation-authorization policy relationships or subject and target relationships.

The enforcement of PONDER authorisation policies on various security platforms needs to be investigated further; different security platforms have different enforcement semantics and some may include restrictions which prevent direct mapping of certain features of PONDER policies onto these platforms. Future work will evaluate more closely the degree to which the same authorisation policy can be enforced on a variety of security architectures and platforms. In addition we are working on mapping PONDER policies to QoS rules for configuring a DiffServ-enabled network. It has been quite easy to produce back-ends for the Ponder compiler, targeted to different platforms. We definitely believe that a common high-level declarative language for different applications of policy-based management is the right approach.

The paper describes what we consider to be the minimum requirements for a toolkit for policy life-cycle management – a high-level language and editor for specifying policies, a compiler for translating policies into enforcement components targeted to specific platforms, a browser to view and manipulate complex domain structures of policies and objects, and an automated approach to dynamically deploying, enabling, disabling and replacing policies in the distributed components that will interpret them. Other components which have been implemented, but not described here, include policy conflict analysis and resolution.

The policy-management toolkit needs to be developed further. Tools for the refinement of high-level policy specifications (goals, SLA's, etc) are a primary objective. As the refinement process is not expected to be fully automated this will require interactive tool support. We are also investigating the possibility for providing further analysis by simulating the execution of policies. An integrated environment for animating the simulation and viewing the results will be part of such a task.

Acknowledgments

We gratefully acknowledge the support of the EPSRC for research grants GR/L96103 (SecPol) and GR/M 86019 (Ponds), BT for support on the Alpine project, and NEC Corporation for funding Mr Tonouchi's visit to the UK.

References

[1] Allot Communications, Policy Based Networking Solution, Available electronically at: http://www.allot.com/pdf/company/pbn_solution.pdf

[2] Barkley, J., et al. Role-Based Access Control for the Web. In Proceedings of CALS Expo International & 21st Century Commerce 1998: Global Business Solutions for the New Millennium

[3] Cisco Systems Inc., Cisco Assure Policy Manager, Available electronically at: http://www.cisco.com/warp/public/cc/pd/nemnsw/cap/index.shtml, June 2000.

[4] Corradi, A., N. Dulay, R. Montanari, and C. Stefanelli. Policy-Driven Management of Agent Systems. In Proceedings of Policy Workshop 2001. 29-31 January 2001. HP Labs, Bristol, UK, Springer-Verlag.

[5] Damianou, N., N. Dulay, E. Lupu, and M. Sloman. The Ponder Policy Specification Language. In Proceedings of Policy Workshop 2001. 29-31 January 2001. HP Labs, Bristol, UK, Springer-Verlag.

[6] Damianou, N., N. Dulay, E. Lupu, and M. Sloman. Ponder: A Language for Specifying Security and Management Policies for Distributed Systems. The Language Specification - Version 2.3. Research Report DoC 2000/1, Imperial College of Science Technology and Medicine, Department of Computing, London, 20 October, 2000.

[7] DMTF Distributed Management Task Force, Inc., Common Information Model (CIM) Specification, version 2.2, available from http://www.dmtf.org/spec/cims.html, June 14, 1999.

[8] Dulay, N., E. Lupu, M. Sloman, and N. Damianou. A Policy Deployment Model for the Ponder Language. In Proceedings of 7th IFIP/IEEE International Symposium on Integrated Network Management (IM'2001): Integrated Management Strategies for the New Millennium. 14-18 May 2001. Seattle, Washington, USA

[9] Fosså, H. and M. Sloman. Interactive Configuration Management for Distributed Object Systems. In Proceedings of First International Enterprise Distributed Object Computing Workshop (EDOC'97). October 1997. Gold Coast, Queensland, Australia, pp. 118-128, IEEE.

[10] Gagnon, E. SableCC, An Object-Oriented Compiler Framework. Master of Science, School of Computer Science, McGill University, Montreal.

[11] Hegering, H.-G., S. Abeck, and B. Neumair, Integrated Management of Network Systems: Concepts, Architectures and Their Operational Application, ed. D. Clark. 1999: Morgan Kaufmann Publishers. 651.

[12] Hewlett-Packard Company, PolicyXpert, Available electronically at: http://www.openview.hp.com/products/policyexpert/index.asp, 2001.

[13] Inxight Software Inc., Inxight Web-Page,http://www.inxight.com.

[14] IP Highway Ltd, Policy Management Console, Available electronically at: http://www.iphighway.com/, 2001.

[15] Lamping, J., R. Rao, and P. Pirolli. A Focus+Context Technique Based on Hyperbolic Geometry for Visualizing Large Hierarchies. In Proceedings of CHI 95, Available electronically at: http://www.acm.org/sigchi/chi95/Electronic/documnts/papers/jl_bdy.htm.

[16] Lupu, E.C. A Role-Based Framework for Distributed Systems Management. Ph.D. Thesis, Department of Computing, Imperial College, London, U. K.

[17] Lupu, E.C. and M.S. Sloman, Conflicts in Policy-Based Distributed Systems Management. To appear in IEEE Transactions on Software Engineering - Special Issue on

Inconsistency Management, 1999.
[18] Marriott, D.A. Policy Service for Distributed Systems. Ph.D. Thesis, Department of Computing, Imperial College, London, U. K.
[19] Marriott, D.A. and M.S. Sloman. Implementation of a Management Agent for Interpreting Obligation Policy. In Proceedings of 7th IFIP/IEEE International Workshop on Distributed Systems Operations Management (DSOM'96). October 1996. L' Aquila, Italy
[20] Mont, M.C., A. Baldwin, and C. Goh. POWER Prototype: Towards Integrated Policy-Based Management, Extended Enterprise Laboratory, HP Laboratories, Bristol, 18 October 1999.
[21] Orchestream Ltd, Orchestream Service Activator, Available electronically at: http://www.orchestream.com/, 2001.
[22] Sandhu, R.S., E.J. Coyne, H.L. Feinstein, and C.E. Youman, Role-Based Access Control Models. IEEE Computer, 1996. 29(2): p. 38-47.
[23] Sloman, M. and K. Twidle, Domains: A Framework for Structuring Management Policy, in Chapter 16 in Network and Distributed Systems Management (Sloman, 1994ed). 1994a. p. 433-453.
[24] Sloman, M.S., Policy Driven Management for Distributed Systems. Journal of Network and Systems Management, 1994b. 2(4): p. 333-360.
[25] Solsoft Inc., Solsoft NP: Policy Management for Enterprise Network Security, Available electronically at: http://www.solsoft.com/
[26] Stone, G.N., B. Lundy, and G.G. Xie, Network Policy Languages: A Survey and a New Approach. IEEE Network January/February 2001, 2001.
[27] Strassner, J., E. Ellesson, B. Moore, and A. Westerinen, Policy Core Information Model - Version 1 Specification, RFC 3060, Available from http://www.ietf.org, February 2001.
[28] Thomsen, D., D. O'Brien, and J. Bogle. Role Based Access Control Framework for Network Enterprises. In Proceedings of 14th Annual Computer Security Applications Conference. December 1998
[29] Verma, D., M. Beigi, and R. Jennings. Policy Based SLA Management in Enterprise Networks. In Proceedings of Policy Workshop 2001. 29-31 January 2001. HP Labs, Bristol, UK, Springer-Verlag.
[30] Virmani, A., J. Lobo, and M. Kohli. Netmon: network management for the SARAS softswitch. In Proceedings of 2000 IEEE/IFIP Network Operations and Management Seminar (NOMS 2000). April 2000. Hawaii