



Axel Buecker
Craig Forster
Sridhar Muppidi
Borna Safabakhsh

IBM Tivoli Security Policy Manager

Introduction

In a growing number of enterprises, policies are the key mechanism by which the capabilities and requirements of services are expressed and made available to other entities. The goals that are established and driven by the business need to be implemented, managed, and enforced by the service-oriented infrastructure (SOA) consistently. Expressing these goals as policy and effectively managing this policy is fundamental to the success of any IT and application transformation, including SOA solutions.

First, a flexible policy management framework must be in place to achieve alignment with business goals and consistent security implementation. Second, common re-usable security services are foundational building blocks for SOA environments that provide the ability to secure data and applications. Consistent IT Security Services that can be used by different components of an SOA runtime are required. Point solutions are not scalable and cannot capture and express enterprise-wide policy to ensure consistency and compliance.

In this IBM® Redpaper publication, we discuss an IBM product-based end-to-end security policy management solution that is comprised of both policy management and enforcement using IT security services. We also demonstrate by means of customer scenarios how this standards-based unified policy management and enforcement solution can address authentication, identity propagation, and authorization requirements and thereby can help businesses demonstrate compliance, secure services, and minimize the risk of data loss.

Business context

Enterprises face new and emerging challenges in securing access to applications and services in today's IT environments, including:

- ▶ An increasing number of regulations that mandate access control requirements on both internal and external entities. An example of these mandates is contained in the Payment Card Industry Data Security Standard (PCI-DSS), which requires restricting access to the personal information of credit card holders on a "business need-to-know basis." Requirements such as these are becoming increasingly fine-grained (for example, privacy

constraints within e-health applications), and the consolidation of businesses requires unifying previously disjointed systems into a unified and compliant environment.

- ▶ Increasing demand to respond to business changes and reuse applications and services, and their sensitive data, which can increase application development and maintenance costs, particularly redundant application security development.
- ▶ The risk of intellectual property loss and threats to sensitive data drive a need to help ensure data-level entitlements and access control to greater numbers of more diverse users, including employees in various roles, contractors, business partners, and even competitors in some cases who are collaborating on projects together.
- ▶ Inconsistent access control policy across the various services and data sources in the enterprise can allow unintended access to business-sensitive data, which can put the enterprise in jeopardy as it seeks to satisfy compliance needs.

The adoption of SOA and Web 2.0 poses unique security policy management challenges, such as:

- ▶ The loose coupling of services and aggregate (mash-up) applications within and across the enterprise can create multiple policy management points, each of which might require its own administration.
- ▶ Security policies and configurations are typically tailored to individual products with platform-specific definitions. Manually managing these policies in a heterogeneous environment is error-prone and creates costly islands of security administration.
- ▶ The re-use of services within composite applications means increasing diversity of users of each service. A single piece of functionality can be re-used by many different parts of the business, rather than existing within the silo of a particular business application. As such, ensuring that users from different lines of business can access only appropriate data becomes more crucial and more difficult.
- ▶ Sensitive resources must be protected consistently across the enterprise, and in an SOA environment the context of each individual request should be part of the access control decision no matter how deeply it is buried within a given composite application.

Simple role-based access control (RBAC) is no longer sufficient. Today's enterprises demand the ability to better control access based on business rules and examining contextual data from authoritative sources within the enterprise. Such fine-grained access control requires improvements in the system used to protect sensitive data and resources.

A key challenge is how to apply policy consistently across the enterprise to comply with a growing number of regulations. Line of business (LOB) owners find it increasingly difficult to translate the organizational policy into technical policy that is suitable for consumption by IT operations. Security policy is no exception to this rule. Policies and configurations are currently platform-specific, with product and technology-specific capabilities and limitations.

Ensuring compliance with the organizational policy and external regulations across disparate and heterogeneous systems is time-consuming and costly. Audit records are an important piece of this puzzle, and recording who did what with which resources is a necessary first step.

This heterogeneity of platforms also places a large burden on the IT operations team that implements the security policy defined by the enterprise architects and business stakeholders. Each platform must be configured to adhere to the same business security policy, and as the size of the environment increases so too does the knowledge and time that is required. Each platform supports different security features, and the administrative capability and tooling support can vary greatly.

A manual process increases the risk of misconfiguring a service, leaving critical information either exposed or inaccessible. The distributed nature of the dependencies on an SOA environment means any security breach is no longer localized to a misconfigured system—data can be exposed in unforeseen ways.

In current environments, each application platform (for example, Java™, .NET, or mainframe) has its own security model and syntax and requires detailed technical skills to configure correctly. Consistency with the high-level requirements and between different platforms, and therefore overall compliance, is increasingly harder to achieve. Furthermore, the lack of traceability between business security requirements and the operational environment means that business people often cannot participate in the decision about which security measures to apply within the IT environment. IT architects desire the ability to capture, model, and analyze security policies that are consistent with the high-level requirements prior to delegating the policies to IT operations and security management.

A centralized policy-based administrative system built on open standards directly addresses these complex and diverse issues. Furthermore, centrally managed security services for cross-cutting tasks, such as authentication and authorization, allow integration with every platform into the unified system.

Policy-based approach

Policy management plays a key role in enabling governance in any service-oriented environment. SOA practices help businesses identify and focus on the key services of the business. Policies are the means by which services express conditions of use and manage the behavior of the supporting infrastructure. Adding policies that are external to the services themselves adds points of control and agility for business and IT, thereby making SOA more consumable and accelerating the adoption of SOA solutions.

A policy-driven approach is fundamental to the success of SOA. The goals established and driven by the business need to be implemented and enforced by the infrastructure. To achieve this implementation, you need an adequate policy management framework in place. For more information, see *SOA Policy Management*, REDP-4463.

IBM defined a policy reference model in *Understanding SOA Security Design and Implementation*, SG24-7310. This reference model has the following aspects to policy:

- ▶ Policy abstraction level

Policy encompasses all aspects of the solution life cycle. Thus, the different levels of abstraction include business, architectural, and operation policies.

- ▶ Policy management life cycle

The policy management life cycle helps guide the deployment of a policy-based information management system in an organization. Figure 1 shows the different stages of a *closed loop* methodology. Policies are authored based on a number of drivers—both business and technical. These policies are then published and transformed into a form that can be enforced. The enforcement is monitored and validated against the policies so that policies can be adjusted to better reflect business and technical drivers.

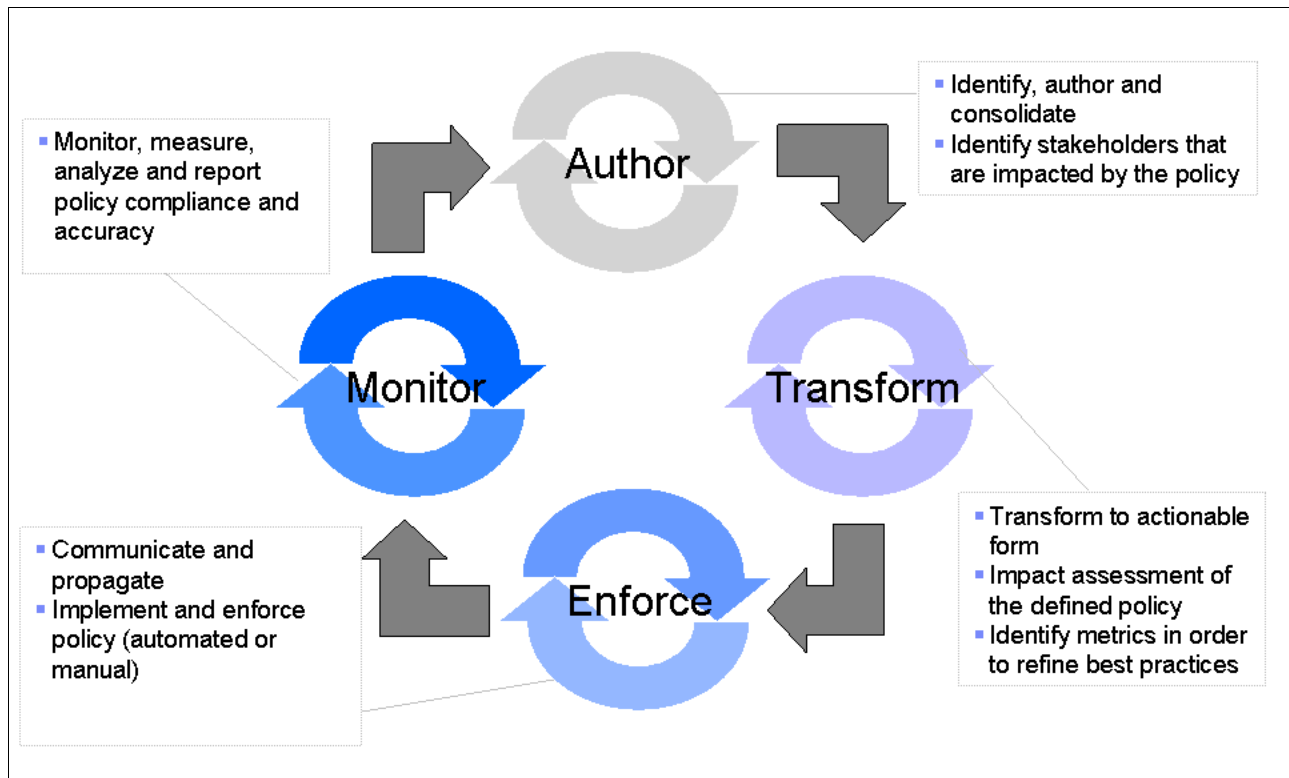


Figure 1 Policy life cycle stages

► Policy domains

Policy can be applied to multiple domains within an organization, including business, process, service, information, and non-functional domains.

Security is one aspect of policy. A number of critical factors can drive the need for better security policy life cycle management across IT environments:

► Apply policy consistently to check for compliance

When corporate policy is articulated, it needs to be incorporated into code, a configuration, or an intermediation of some kind in order to be effected on a service.

► Address risks that are associated with deploying inconsistent policies

Inconsistent policy deployment can pose significant operational risk, further driving the desire for security policy management.

► Enable a higher level of visibility and control to drive operational governance

Better visibility into and control of security policies can drive the need for security policy management.

Security policy management

Effective management of security policies requires holistically managing security policies throughout the life cycle of applications. Policies in the context of SOA are the means by which processes and services express the conditions for their use and manage the behavior of the underlying infrastructure to secure access to information, provide availability and retention, enable audit, and so on.

Security policy management begins with authoring security policies that are tailored to business needs. Policies are defined and managed centrally by *policy administration points* (PAPs). After the effective policy is obtained, the policy is distributed in a common format from these PAPs to *policy decision points* (PDPs) and *policy enforcement points* (PEPs). The distributed policies, in turn, are enforced by the infrastructure, possibly after being translated into product-specific configuration. Figure 2 shows this concept.

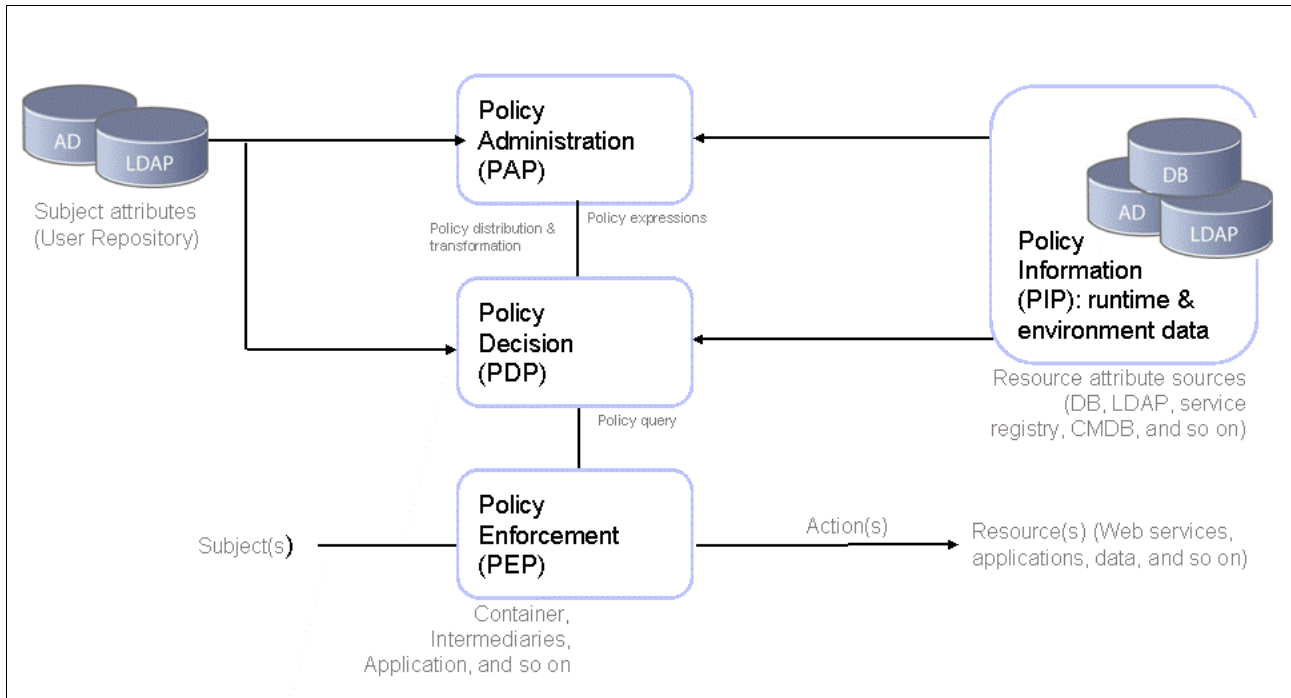


Figure 2 End-to-end security policy management

Representing security policy in open, standard formats has a number of important benefits. One benefit is to easily integrate with third-party products that support the same standards. Common standards for representing security policy include WS-SecurityPolicy¹ for message protection policy and OASIS eXtensible Access Control Markup Language (XACML)² for authorization policy.

WS-Policy and WS-SecurityPolicy provide descriptions of how service consumers and providers can specify requirements and capabilities in a Web services world. Policy assertions can be defined for use within SOAP messages. Assertions can cover the authentication schemes (the required security tokens and the encryption algorithms), the transport protocol selection, the privacy policy, as well as information related to the quality of service.

The XACML provides a markup language to specify access control policies. It can be used as a generic format to store policies, although it also provides a request and response model (based on XML format) for communication between enforcement and decision points.

¹ For more information about WS-SecurityPolicy, see:
<http://www.oasis-open.org/specs/index.php#ws-secpol>

² For more information about XACML, see:
<http://www.oasis-open.org/specs/index.php#xacml>

Policy administration

Policy administration addresses policy life cycle management—creating, modifying, maintaining, and deleting policies. As part of this life cycle, business policies are refined to service-specific policies such as security, performance indicators and metrics, trust policies, and so on. These policies, in turn, are enforced by the infrastructure when configured as requirements that the infrastructure should meet.

Traceability of policies from high-level business requirements to enforced configurations and runtime policy is key, primarily to track the goal of the policy and the basis for the runtime behavior. This ability can help trace and manage accountability for policy changes. You can keep track of applied and historic policies, as well as assess the compliance of lower-level policies against corporate policies. This traceability, version control, and compliance validation are key parts of an overall SOA governance solution.

When you deploy an application, you administer application-related policies to reflect any changes that might happen during the lifetime of the application. Changes to security policies include authorization policy changes (for example, adding new roles that can access the resources or assigning roles to new groups or users), user management changes (for example, users assigned to additional user groups), or other changes including audit requirements, and constraints such as integrity or confidentiality.

Underlying the policy management infrastructure is the ability to formally articulate policies. Policies can be articulated in terms of metadata about services, identities or other contextual information such as strength of authentication, time of the day, and so on. Therefore, articulating and managing metadata is very important.

Metadata can be about the target information content, the configuration and topology of those systems, or the applications that allow access to the data. For example, metadata can contain data classification levels, sensitivity of the data, value if lost, system configuration and topologies, and more. Metadata can also be made available through service registries and repositories.

Policy distribution and transformation

The security policies that are created need to be *distributed* to the enforcement and decision points within the infrastructure. The policies are defined centrally and are distributed to the enforcement points in a canonical format (for example XACML, WS-Policy, or WS-SecurityPolicy). The binding information to enforce the policies is also distributed appropriately. These policies are often then *transformed* at the enforcement point to a local representation so that they can be enforced.

Securing the distribution of the policies from the PAP to the PDP is critical in ensuring the security of the infrastructure itself. There can also be a policy within the PAP that controls the transformation and distribution of the policies. For example, one business might want policy updates sent out immediately whereas another business might want updates sent only during a specific maintenance window.

Policy decision and enforcement

Access to a resource is controlled by a *resource manager* that serves as a logical PEP. Administrators update security policies through the resource managers, or administer policies through PDPs. In a typical deployment, you can find several enforcement points. Each of these enforcement points can have its own mechanisms to enforce security for incoming requests.

The enforcement points rely on PDPs to make decisions. These PDPs interpret the security policies that are defined in the infrastructure to make policy decisions. Different security domains and application platforms have different requirements and thus might have different policy types and formats.

One challenge of having multiple PEPs and PDPs in the infrastructure is that they are often administered by different entities. Providing integrated and centralized decision capabilities reduces the administrative tasks related to policy management.

Policy context

The context in which a service is being accessed is as important as who is accessing it.³ The same service can be used both by an internal application as well as exposed to business partners, with different security requirements in each case.

You need to consider and evaluate the following aspects of context:

- ▶ Identity context

Information and attributes about the identity, such as the e-mail address, name, status, and so on.

- ▶ Service context

Information and attributes about the service or resource that is being accessed. For example service-level agreement, transaction amount, and so on.

- ▶ Request or environment context

Information and attributes about context of the environment, such as the time of the day, whether access is coming from an internal or external source, and so on.

- ▶ Business context

Context information such as emergency state, special permissions, and so on that can be either consulted from external rules engines or captured as policies.

The sources of such information are sometimes referred to as *policy information points* (PIPs).

Monitoring and reporting

As part of policy decision and enforcement, monitoring the behavior of system elements throughout the life cycle is critical to adhere to both changing corporate security policies as well as the discovery of vulnerabilities and new risks that can be identified through solution monitoring activities.

It is necessary to keep track of current policies, historic policies, and the assessment of compliance of lower level policies against corporate policies. Traceability of policies from high-level business policies to enforced configurations and runtime requirements is necessary to verify the basis for the runtime behavior. This ability to trace helps identify policy changes that occur and can help manage accountability of policy changes. Security policies need to be developed and deployed throughout the stages of the life cycle of an application.

³ Service in this context does not necessarily mean Web services, but rather a generic service such as Web service, J2EE™ service, .NET service, OS resource, Database element, and so on.

Policy modeling

Policy modeling addresses a key gap in the deployment of IT security measures at the company or line-of-business level. The gap exists between business process owners and IT professionals. Business process owners formulate high-level security requirements (for example, stating that personal information of customers needs to be protected). IT professionals then need to architect, implement, and manage security measures, such as access control, on a multitude of platforms (for example databases, file systems, Web Services, and applications).

IBM Tivoli Security Policy Manager overview

IBM Tivoli® Security Policy Manager V7.0 is a standards-based application security solution. It provides centralized application entitlement management, SOA security policy management, and *security as run-time services* to strengthen access control for new applications and services, to help improve compliance, and to drive operational governance across the enterprise.

Tivoli Security Policy Manager enables application owners and administrators to externalize security and to simplify the management of complex authorization policies for new and existing applications, including customized applications. The benefits include:

- ▶ Ability to respond quickly to business changes through centralized application roles, entitlements and data-level access control
- ▶ Improved compliance and security management with roles-, rules-, and attributes-based access control

Tivoli Security Policy Manager also enables enterprise architects and security operations teams to centrally manage and enforce security policies for Web services resources across multiple policy enforcement points, including the WebSphere® DataPower® SOA appliances. The benefits include:

- ▶ Reduced manual, inconsistent, and costly administration of security policies at each policy enforcement point
- ▶ Operational governance with the ability to delegate and audit all changes to policies

IBM Tivoli Security Policy Manager capabilities

Tivoli Security Policy Manager provides a centralized security policy management and IT security services for modeling, authoring, transforming, enforcing, and monitoring security policies.

Tivoli Security Policy Manager reduces the risks and cost of managing security policy by centralizing management of the end-to-end security functions. As shown in Figure 3, at a high level, the product provides the following capabilities:

- ▶ The Policy Administration Point is the Tivoli Security Policy Manager and Console.
- ▶ The Policy Decision Point is the Tivoli Security Policy Manager Runtime Security Services.
- ▶ An example of a Tivoli Security Policy Manager Policy Enforcement Point is a plug-in for WebSphere Application Server for enforcing container level security.

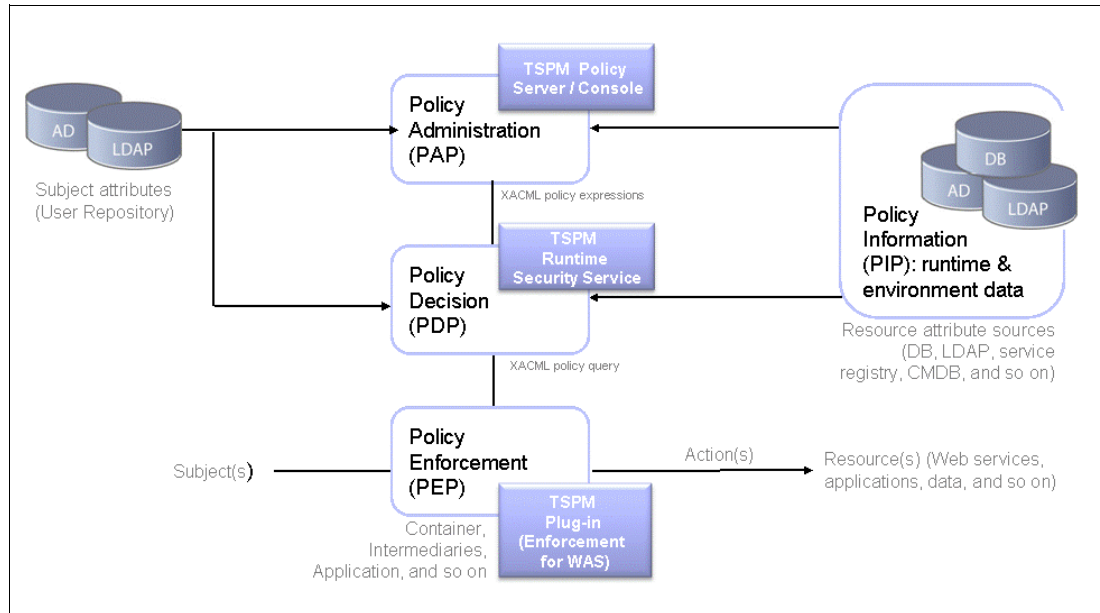


Figure 3 How Tivoli Security Policy Manager fits into the end-to-end policy management framework

Specifically, Tivoli Security Policy Manager provides the following capabilities:

- ▶ Policy administration point (centralized administration)

Delivered as the Tivoli Security Policy Manager Policy Server with the following features:

- Resource identification

Helps you to discover existing resources and their dependencies using existing repositories in an IT environment. Also includes any metadata that is associated with the resource.

- Resource classification

Enables you to classify resources, for the purpose of applying policy to a given resource, in ways that align with the business requirements.

- Policy authoring

Enables authoring of different security policies such as message protection and authorization policies to protect and govern services. As shown in Figure 4, authoring is based on the type of policies using an intuitive user interface to cater to both application owners and IT operations. Policies can be defined based on the different contexts, such as identity, service or resource, environment, and business.

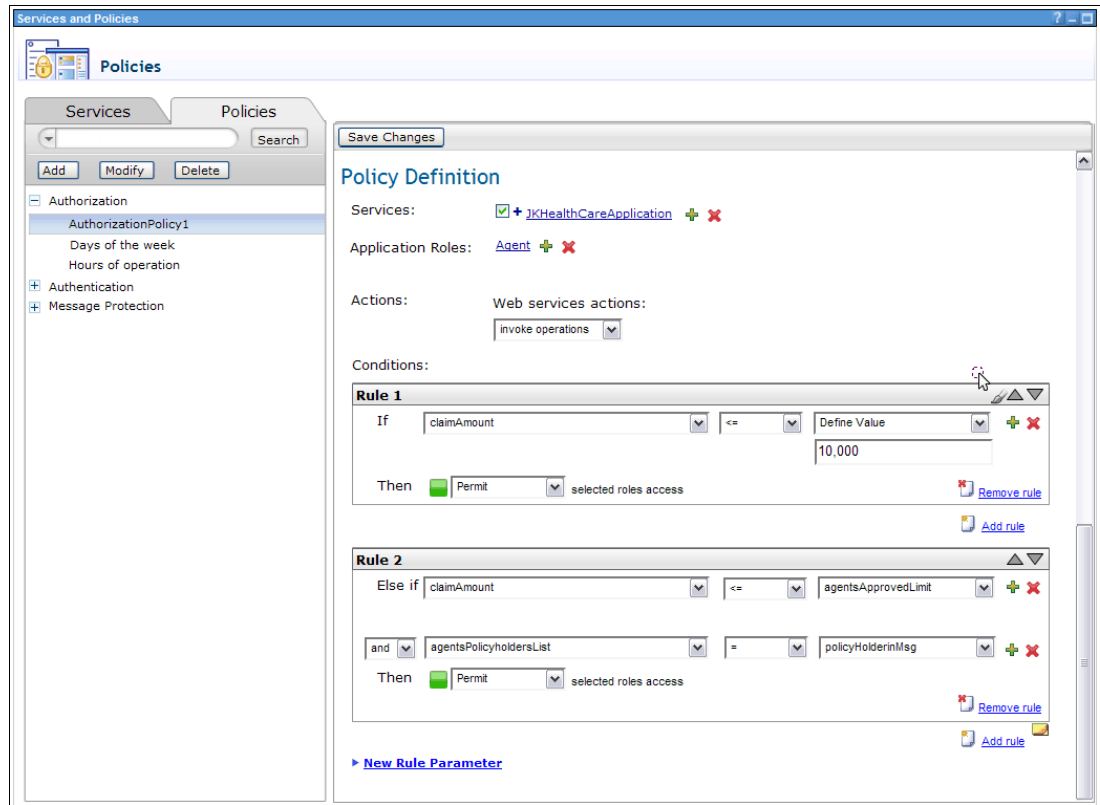


Figure 4 Authorization policy authoring in Tivoli Security Policy Manager

- Policy transformation

Enables automatic conversion of the policy to the format that is required by a given target system, which is XACML or WS-SecurityPolicy. In addition, multiple policies can be combined to derive effective policies both from inheritance as well as direct association.

- Secure policy distribution

Enables distribution of policy to stand-alone PDPs (such as Tivoli Security Policy Manager Runtime Security Services) as well as to logical decision points that are embedded inside enforcement points and resource managers. Tivoli Security Policy Manager uses a protocol based on WS-Notification and WS-MetadataExchange, using XML protection measures such as digital signatures to ensure the integrity of the transferred policy. Updates can be configured to be sent only over SSL for another layer of protection and confidentiality. The policies are secured both at the message level as well as by enforcing authentication and authorization so that only valid entities can get the policies.

- Delegated administration

Allows you to delegate administrative tasks to individuals based on roles and responsibilities. Roles that are shipped with Tivoli Security Policy Manager include Application Admin, Application Owner, Policy Operations, Policy Author, Auditor, IT Admin, and Role Admin. In addition, you can define new roles based on business requirements.

- Provide auditing and logging for serviceability, reporting, and so on with standard reports.
- Change management and version control to keep track of policy changes, versioning of policies, and so on.
- ▶ Policy decision point (*Security as a Service*)
 - Provides the capabilities to handle security functions such as identity (IdAS), authentication (WS-Trust), authorization (XACML), and audit (CBE) in standards based services. These services are delivered as Tivoli Security Policy Manager Runtime Security Services and can be used by enforcement points to enforce policies that are defined centrally and distributed.
 - Tivoli Security Policy Manager provides not token and identity mediation as well as multiple levels of authorization enforcement with different granularity based on standards based interfaces.
 - For performance, scalability, and security reasons, you can deploy Tivoli Security Policy Manager Runtime Security Services in either local or remote mode. Thus, you can evaluate policies locally if needed.
- ▶ Policy enforcement point. In cases where needed, Tivoli Security Policy Manager provides enforcement points that fall into the following patterns:
 - Container based enforcement: The container is instrumented with a Tivoli Security Policy Manager plug-in (that is, a plug-in for WebSphere Application Server) to enforce policies without having to modify the applications.
 - Intermediary based enforcement: Intermediaries such as Web services gateway (that is, WebSphere DataPower) intercept the request and enforce the policies before providing access to the applications. The intermediaries can function both as PDP and PEP or just a PEP using an external PDP (such as Tivoli Security Policy Manager Runtime Security Services). Message protection policies can be enforced at message intermediaries, which are message handlers that can support WS-SecurityPolicy.
 - Application based enforcement: Applications have a choice of using the Java API based on JACC or the standard XACML over SOAP to get authorization decisions from Runtime Security Services and to enforce them.
- ▶ Integration with policy information points
 - Tivoli Security Policy Manager supports the standards-based Identity Attribute Service (IdAS) to integrate and use information from existing identity management, identity and attribute repositories, and rules engines. Using IdAS alleviates the need to work with each and every repository but can use a large pool of open source context providers to integrate with existing repositories.
 - As we mentioned in “Security policy management” on page 4, the context of the information can be based on identity, service, environment, and business.
- ▶ A policy design tool is provided as part of the Tivoli Security Policy Manager offering to help customers (for example, architects) model, analyze, and create policy templates and to check for Separation of Duties violations prior to using those templates within the application environment.

Architects can use this development tool to model and analyze the high-level security requirements in a business context and to create security policy templates in a standard format for use within the IT environment. Resources can be classified according to their business function using multiple taxonomies. Users can be assigned roles that capture the job functions that they are performing. Thus, the security template policies are captured with business-oriented resource classes and roles for ease of review with the business process owners. Policy design tool also provides the ability to import detailed security

policy information that previously resided on different platforms and provides the ability to assess a complete view of the security policy landscape. A number of analysis functions also help ensure that policies are consistent on the business level.

As shown in Figure 5, after policies are captured, modeled with roles, and verified using the policy design tool, you can export them as policy templates for use by Tivoli Security Policy Manager for application entitlement and SOA security operational policy management. Tivoli Security Policy Manager can use these templates to author and refine message protection and roles-, rules-, and attributes-based application entitlements and to transform and enforce access control across heterogeneous IT environment.

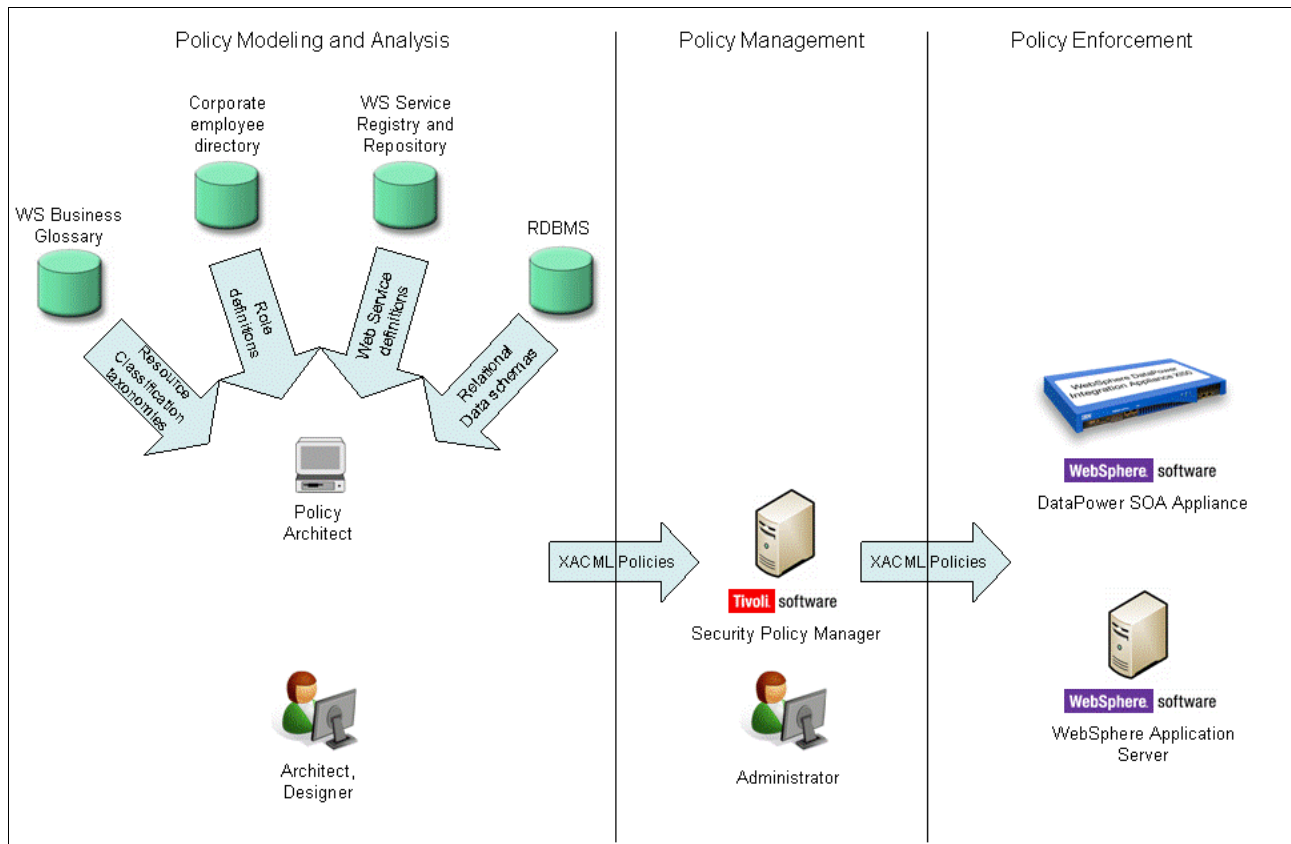


Figure 5 Policy modeling, management, and enforcement

In addition to these key features, Tivoli Security Policy Manager also integrates with IBM products for governance and provisioning. Web services can be discovered from WebSphere Service Registry and Repository or UDDI based service registries. The authored policies can be published back to the registry for governance.

Managing the security of large-scale SOA environments is a key capability of Tivoli Security Policy Manager. *Classifications* reduce the administrative overhead by allowing common security requirements to be configured once and applied to multiple services, ensuring that cross-cutting security requirements are applied consistently. Any service-specific policy is then combined with the classification's policy to form the effective policy for the service. A service can be included in multiple classifications.

For example, a *confidential information* classification can be created that contains both a message protection policy specifying encryption and an authorization policy containing certain role membership requirements. This classification can then be attached to any service deemed to be using confidential information.

Tivoli Security Policy Manager provides a powerful delegated administration security framework. Administrators can delegate responsibility for certain tasks to specific users. For example, the line-of-business owner can author attribute-based policy but then delegate the actual binding of that field to a concrete LDAP attribute to the IT operations team.

This policy management administration framework allows the appropriate individuals within the enterprise to handle the most appropriate tasks for their job role in a secure manner. High-level line of business owners do not need to know specific environment details (such as the IP address and bind credentials of an LDAP server). In addition, the IT operations team does not need to define security policy but simply ensure that it is implemented effectively.

After security policy is authored, it often requires the enforcement of the additional information for the policy by the policy enforcement points. Tivoli Security Policy Manager makes an important distinction between policy authoring and policy configuration, primarily based on the roles within your organization. Fundamentally, policy authoring refers to the user interaction that results in a policy that is standards-compliant and inter-operable but might or might not contain sufficient implementation detail to enforce on an enforcement point platform. Policy configuration refers to the additional level of detail that is required to augment the policy and to make it enforceable, potentially including platform-specific data based on the policy's ultimate target platform.

In most use cases, the policy configuration data is not specific to any PDP. For some use cases, however, configuration data is dependent specifically on the intended decision point. For example, DataPower might require the presence of a summary element at the top of the message protection policy document that describes the policy domain for the given policy. Different PEPs will often have different policy configuration requirements.

Every aspect of the Tivoli Security Policy Manager Management Server is built on a pluggable framework based on OSGi runtime.⁴ Business partners, OEM vendors, and consultants can easily build custom solutions based on Tivoli Security Policy Manager to provide unique value to their clients. In “IBM Tivoli Security Policy Manager architecture” we provide a more in-depth look at this framework.

Tivoli Security Policy Manager takes advantage of the capability of the existing investment in IBM products:

- ▶ The powerful capability for SOA governance of WebSphere Service Registry and Repository
- ▶ The flexibility and speed of SOA security provided by DataPower SOA appliances
- ▶ The provisioning and recertification provided by Tivoli Identity Manager

Other assets in the IT environment can also be used. Services can be discovered from registries that support UDDI, and integration with standard user registries, such as LDAP and Active Directory®, is supported. Tivoli Security Policy Manager integrates with the IBM Tivoli Common Reporting framework to provide reports on application roles and entitlements.

⁴ For more information about OSGi, see:

<http://www.osgi.org/Main/HomePage>

Tivoli Security Policy Manager uses the Equinox OSGi framework from Eclipse. For more information, see:

<http://www.eclipse.org/equinox/>

IBM Tivoli Security Policy Manager architecture

Tivoli Security Policy Manager architecture includes three easy to deploy components:

- ▶ The policy server or console (for PAP)
- ▶ The runtime security services (for remote or local-mode PDP)
- ▶ The policy enforcement plug-ins (PEP) for specific application deployments

We discussed the capabilities of each of these components in “IBM Tivoli Security Policy Manager capabilities” on page 8.

Tivoli Security Policy Manager is built on an entirely standards-based, scalable product architecture, which includes an Identity Attribute Service (IdAS) to integrate with existing identity management, existing identity and attribute repositories, rules engines, and so on. Tivoli Security Policy Manager is built on an Eclipse-based plug-in architecture and is extensible to support new resources, new policies, and new PDP and PIPs.

Policy design tool is built on the Eclipse Rich Client Platform and offers a number of perspectives and views for modeling, analyzing, and debugging access control policies. The user interface design is easy to use for business users because it minimizes security-specific terminology and intuitively shows the effects of policies. The tool also contains an embedded XACML PDP so that the user can simulate access requests and debug the decision process that an operational PDP will carry out in a real deployment.

Using this policy design tool, architects can model and analyze the high-level security requirements, assign users to roles, and create template access control policies for resources, which include Web services and application data in relational databases. After review with business owners, these templates can be exported into Tivoli Security Policy Manager to author and refine both message protection and application entitlements (roles-, rules-, and attributes-based), transform the policies and enforce them across the IT environment.

At a high level, as illustrated in Figure 6, Tivoli Security Policy Manager Policy Server can import service definitions and metadata from repositories such as service registries and can also work with existing identity and access management systems. Policies can be authored by either the Tivoli Security Policy Manager Policy design tool (for modeling and simulation) and imported into Tivoli Security Policy Manager as templates or directly by Tivoli Security Policy Manager. These security policies are then securely distributed to various resources or Tivoli Security Policy Manager Runtime Security Services so that the resources can enforce the policies either directly or using Runtime Security Services. Runtime Security Services can integrate with existing identity and resource attribute sources, rules engines, and so on at runtime. Finally, as we described in “IBM Tivoli Security Policy Manager capabilities” on page 8, the Tivoli Security Policy Manager enforcement points are provided in situations where the resource itself does not have the capability to interpret standards-based security policies.

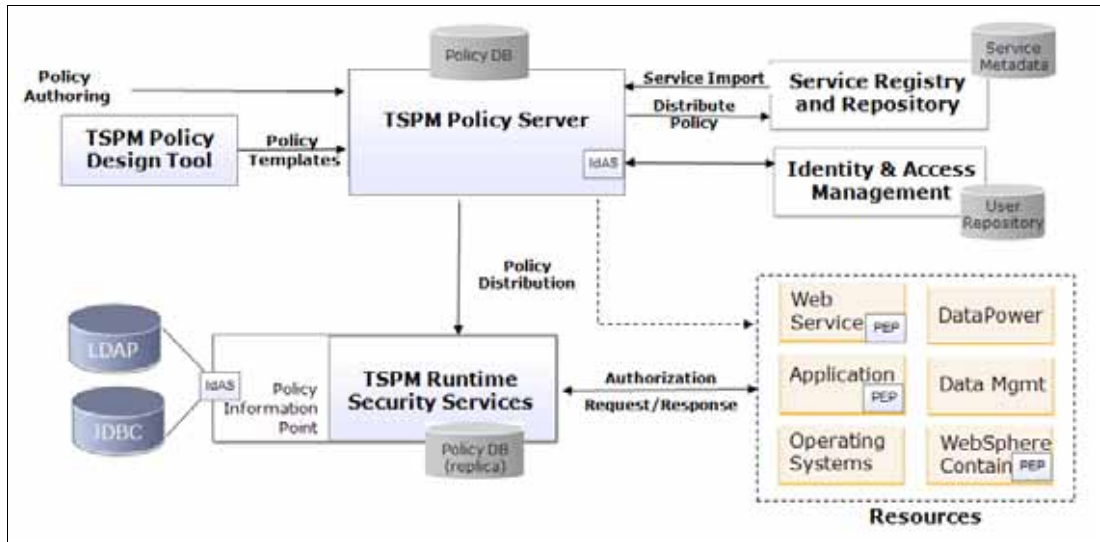


Figure 6 High-level architecture of Tivoli Security Policy Manager

The core of Tivoli Security Policy Manager consists of a policy administration framework and user interface, a policy repository to store policy data and metadata, and a security and delegation framework. Plug-ins of several different types, which exist on top of this framework, enable additional capabilities of Tivoli Security Policy Manager.

Tivoli Security Policy Manager uses JPOX to work with the underlying database.⁵ JPOX is an open-source fully-compliant implementation of the JDO and JPA specifications that provides transparent persistence of Java objects. It supports persistence to all of the major RDBMS on the market today.

Figure 7 shows the various architecture components of Tivoli Security Policy Manager.

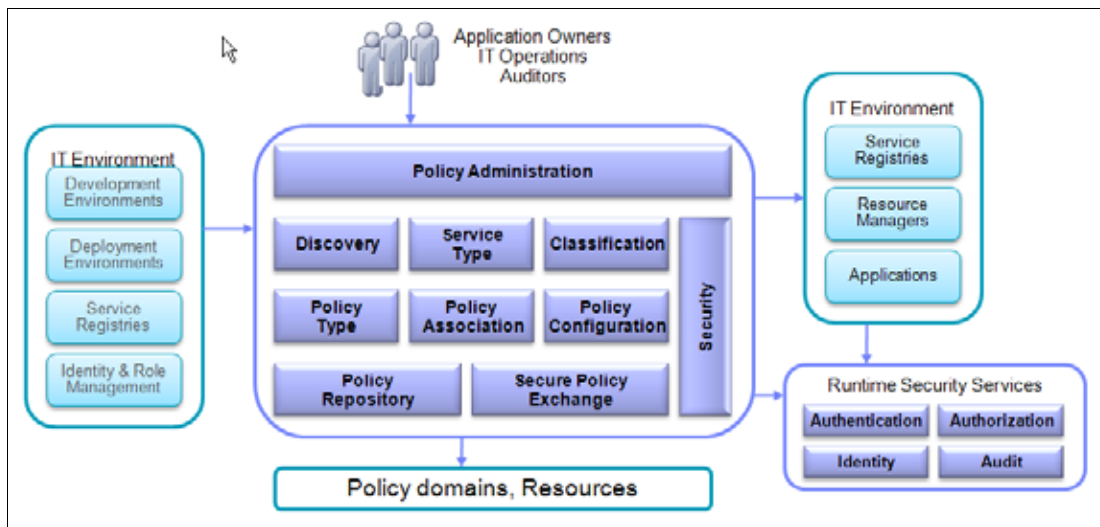


Figure 7 The components of Tivoli Security Policy Manager

⁵ For more information, see: <http://www.jpox.org>.

The plug-in framework is based on OSGi, with *extension points* defined that allow the addition of:

- ▶ New types of services (that is, Web services, J2EE, Portal, SharePoint®, Data, and so on)
- ▶ New types of policy to apply to services (that is, Authorization, Message Protection, and so on)
- ▶ New policy distribution targets (PDTs) to which to send policy (that is, DataPower, WebSphere Service Registry and Repository, Portal)
- ▶ New registries for storing service definitions and metadata (that is, WebSphere Service Registry and Repository, UDDI, MOSS, and so on)
- ▶ New types of user registries (that is, LDAP, DB2®, Active Directory, and so on)
- ▶ New types of password obfuscation plug-ins to allow custom obfuscation of passwords maintained by Tivoli Security Policy Manager

Most notably, the service type and policy type plug-ins define and manage the structure and semantics of instances of those kinds of services and policies. For example, the Web service type plug-in manages the processing of an imported WSDL to define an instance of a Web service, as well as any user interface pages specific to the contents of a Web service.⁶ Similarly, the authorization policy type plug-in manages the user interaction of authoring an authorization policy, as well as the import, calculation, and generation of the XACML authorization policy created and distributed by Tivoli Security Policy Manager.

The discovery capability employs service definition and other registry plug-ins to query the IT environment for service data and meta-data.

The distribution capability provides the standardized (WS-Notification and WS-MetadataExchange) secure policy exchange of notification and policy retrieval, while also allowing plug-ins to provide custom distribution logic, for example to distribute policy to custom policy distribution target using its available APIs.

Finally, you can deploy the Runtime Security Services in conjunction with the Tivoli Security Policy Manager Policy Server or throughout the IT environment to provide the suite of runtime security services described in “IBM Tivoli Security Policy Manager Runtime Security Services adoption” on page 18.

The underlying flexibility of the Tivoli Security Policy Manager architecture allows the product to be extended to support any new type of service to be protected, type of policy to be authored, type of registry to be queried, type of target to receive policy distributions, and so on. It allows of each of these types of plug-ins to contribute user interface panels to fully express their semantics and configuration requirements.

IBM Tivoli Security Policy Manager for Application Entitlements

In this offering, application owners and architects can author and evaluate policy for business applications, allowing business applications to externalize and delegate authorization and audit functionality to Tivoli Security Policy Manager. The policy becomes configured by an administrator, not hard-coded into an application, thus reducing the cost of managing entitlements throughout the enterprise and allowing line of business owners to ensure that the corporate organizational security policy is accurately reflected in their applications.

The authorization policy authored by Tivoli Security Policy Manager and evaluated by the provided Runtime Security Service is compliant XACML v2.0 policy. This compliancy allows

⁶ WSDL, or Web Service Definition Language, is the standard format for describing the capabilities of a Web service.

for extremely powerful authorization policy, which allows the business rules to be accurately reflected in the underlying security policy.

Three types of policy, and combinations thereof, are supported.

- ▶ *Role-based* policy allows access to be granted based on role membership, where membership is determined based on a condition such as the user being in a specified LDAP group.
- ▶ *Attribute-based* policy allows the security policy to use information from the IT environment. Attributes can be compared to static values or to each other in a series of rules, where each rule codifies a particular business rule or security requirement.
- ▶ *External rule-based policy* enables the use of logic from external systems to be used as part of the access control decision, allowing businesses to leverage their existing investments.

Every request for access is audited by Tivoli Security Policy Manager. Extensive information is recorded in each audit record, including the identity of the user, the resource that is accessed, the rule that permitted or denied access, any attributes from external sources that were used, and the machine that made the request for access. Audit records are stored in the Common Base Event (CBE) format, allowing for consumption and correlation for infrastructure monitoring software such as IBM Tivoli Security Information and Event Manager.

IBM Tivoli Security Policy Manager for SOA

In this offering, Tivoli Security Policy Manager provides an end-to-end solution for managing the security policy protecting Web services, including using existing enforcement run times. Web services can be discovered from a registry such as WebSphere Service Registry and Repository or UDDI, standards-based policy can be authored and attached to these services, and the resulting policy can be published back to the registry or distributed to decision or enforcement points for runtime evaluation.

The message protection policy authored by Tivoli Security Policy Manager is compliant WS-SecurityPolicy 1.2 policy, which allows for extremely powerful encryption, signature, and token policies that allow the business drivers to be reflected accurately in the underlying security policy.

Both message protection and authorization policy can be authored by Tivoli Security Policy Manager for Web services. All the capabilities of the authorization policy described in “IBM Tivoli Security Policy Manager for Application Entitlements” on page 16 are available for use with Web services. Furthermore, when authoring policy for Web services, the message content (that is, the SOAP Body element) is available for inspection at run time, which allows business rules such as “users can access only their own personal information” to be represented in Tivoli Security Policy Manager simply and clearly.

The IBM Security Policy Manager runtime security services used in the previous solution are not a prerequisite for this solution. The DataPower SOA appliances can consume the message protection and authorization policy produced by Tivoli Security Policy Manager, adding the centralized management, control, and reporting capabilities described previously. When integrated with DataPower, the authorization policy can use attributes presented by the security token (such as a SAML 2.0 token) used to authenticate to the protected Web service.

IBM Tivoli Security Policy Manager Runtime Security Services adoption

Tivoli Security Policy Manager provides a suite of standard security functions that are exposed by consumable Web services. These security services are called the *Runtime Security Services* and include the following services as highlighted in Figure 7 on page 15:

- ▶ Authentication
- ▶ Authorization
- ▶ Identity
- ▶ Audit

Tivoli Security Policy Manager Runtime Security Services provide these four reusable security services, although for the initial release the audit service is for internal use only.

The authorization engine is a high-performance implementation of the XACML 2.0 standard, which is authored by the Organization for the Advancement of Structured Information Standards (OASIS). The XACML engine is fully conforming with version 2.0 of the specification, and has been publicly demonstrated at previous interoperability events organized by OASIS.

Runtime Security Services can be embedded in WebSphere Application Server to provide *local mode* authorization and audit capability. Each Runtime Security Services client instance can be configured to have only the policy applicable to the local machine distributed to it, allowing for scalability in large deployments.

A key feature of the authorization engine is the ability to retrieve information from authoritative sources in the IT environment. This capability makes attribute-based authorization policy so powerful. Identity attributes are provided by the Higgins Identity Attribute Service (IdAS). Customers and business partners can write or use existing *context providers* for IdAS to allow integration with the data stores that are not provided with the solution. Support for querying databases using JDBC™ is also provided.

Another source of information that can be used during the authorization decision is the content of the actual message passed to the service. XPath expressions can be specified to extract data from XML content, such as the SOAP body, and the extracted data can be used as an attribute. Combining this functionality with data retrieved from the IT environment allows for powerfully simple policy to be expressed and enforced.

For example, a policy protecting customer information might be to allow customers to access their own data but not anyone else's data. For this example, assume the Web service's SOAP body contains an element that represents customer information by a unique identifier that is different from the authenticated user name. The policy can state that the unique identifier can be extracted from the message body using XPath and compared to the currently authenticated user's unique identifier extracted from an LDAP server, allowing access only if the values match.

Runtime Security Services allows the delegation of an authorization decision to an *external rule*. External rules are custom code that make an authorization decision based on appropriate logic, perhaps calling an external system in the process. This decision is then combined with the other rules that may be part of the policy to form the final result.

Taking advantage of your existing IT environment

Tivoli Security Policy Manager integrates with the following IBM and third-party products to provide a complete policy management solution, thus taking advantage of the investment in your existing IT infrastructure:

► **Service Registries, such as IBM WebSphere Service Registry and Repository**

Web service definitions can be discovered and imported from service registries into Tivoli Security Policy Manager. UDDI-based registries are supported, as well as IBM WebSphere Service Registry and Repository.

WebSphere Service Registry and Repository helps maximize the business value of your SOA. It is an industry leading solution that enables you to easily and quickly publish, find, enrich, manage, and govern services and policies in your SOA. Authored policies can be published from Tivoli Security Policy Manager back into WebSphere Service Registry and Repository to be integrated into the governance process defined for your environment.

For more information, see:

<http://www.ibm.com/software/integration/wsrr/>

► **XML appliances, such as IBM WebSphere DataPower SOA Appliances**

XML appliances are specialized devices to secure, accelerate, and route XML. They are commonly used as boundary security devices, placed in the DMZ to protect Web services in an SOA.

WebSphere DataPower SOA Appliances are a key element in the holistic approach to SOA from IBM. These appliances are purpose-built, easy-to-deploy network devices to simplify, help secure, and accelerate your XML and Web services deployments.

DataPower appliances are a supported policy distribution target for Tivoli Security Policy Manager, allowing Tivoli Security Policy Manager to centrally manage and author security policy for enforcement by these appliances.

For more information, see:

<http://www.ibm.com/software/integration/datapower/>

► **Application servers, such as IBM WebSphere Application Server**

Application servers are platforms on which business applications run, commonly built on well-defined platforms and languages such as J2EE and .NET.

WebSphere Application Server drives business agility by providing developers and IT architects with an innovative, performance-based foundation to build, reuse, run, integrate, and manage SOA applications and services. Tivoli Security Policy Manager has two integrations with WebSphere Application Server:

- The Tivoli Security Policy Manager Authorization API described in “IBM Tivoli Security Policy Manager capabilities” on page 8
- A JAX-RPC Web services enforcement point

The Runtime Security Services Client can run in local-mode inside WebSphere Application Server for both these integrations, allowing scalable and high-performance authorization decisions.

For more information, see:

<http://www.ibm.com/software/webservers/appserv/was/>

- ▶ **User and role management systems, such as IBM Tivoli Identity Manager**

User and role management systems provide governance and automation for the provisioning and deprovisioning of users.

Tivoli Identity Manager helps enterprises strengthen and automate internal controls that govern user access rights. It provides a secure, automated, and policy-based solution that helps effectively manage user privileges across heterogeneous IT resources. Tivoli Identity Manager can be used to provision application roles to Tivoli Security Policy Manager, enabling a view into the application entitlements that are enabled by provisioning users into organizational roles.

For more information, see:

<http://www.ibm.com/software/tivoli/products/identity-mgr/>

- ▶ **Log management and event correlation systems, such as IBM Tivoli Security Information and Event Manager**

Tivoli Security Information and Event Manager provides centralized log management, event correlation, policy compliance dashboard, and a reporting engine. It provides visibility into your security posture, controls the cost of demonstrating compliance, and reduces the complexity of managing a heterogeneous IT infrastructure. All authorization requests are audited by Runtime Security Services in the CBE format, ready for consumption by Tivoli Security Information and Event Manager.

For more information, see:

<http://www.ibm.com/software/tivoli/products/security-info-event-mgr/>

- ▶ **Existing user data stores, using the Higgins Identity Attribute Service**

In addition to integrating with these products, the Higgins Identity Attribute Service (IdAS) provides a standardized plug-in point for integrating custom context providers, which allows you to use existing IT assets that the solution does not support as sources for attribute data at run time.

For more information, see:

http://wiki.eclipse.org/Identity_Attribute_Service

Customer deployment scenarios

In this section, we outline two business scenarios and how you can use Tivoli Security Policy Manager to address them.

Application entitlements management use case

This scenario provides an example of using Tivoli Security Policy Manager to centrally manage application entitlements.

Scenario description

In this scenario, an insurance firm has a significant portfolio of business applications built by outsourced development contracts. The Director of Application Development highlighted the cost of developing security components for these applications, because each application is using custom security code. The manager is concerned about the lack of control of the security of their applications. However, modifying the application's security policy to meet new business requirements is costly and time-prohibitive, and auditing who has accessed sensitive data is impossible.

The development manager is looking for a centralized set of security services to provide the common functionality across all the business applications. The firm is also looking to take its investment in a commercial business rules engine from the security framework. Various existing applications use this rules engine extensively, so they are hesitant to attempt to remodel these rules in a separate system.

Solution

To regain control of the business application's security logic, this company uses the provided *Tivoli Security Policy Manager Authorization API based on JACC* to externalize and delegate the authorization logic. Then, they can manage the security logic centrally, independent of the application itself.

Using the API from within the business application, rather than doing URL-based access control at the Web layer, allows the company to use application-specific context as part of the decision process. Application-specific attributes can be passed easily to the policy decision point, and the policy authored by Tivoli Security Policy Manager can examine these attributes in a similar manner to those attributes explicitly retrieved from an external source such as an LDAP server.

The company also uses the *external rule* capability of Tivoli Security Policy Manager to take advantage of the business rules engine that is already in place. Tivoli Security Policy Manager provides an API to author custom pieces of code that can then be instantiated with parameters specified in a configuration file, given a name, and invoked at run time.

Figure 8 shows a simplified overview of this solution. In this diagram, and all subsequent diagrams, the dashed lines illustrate configuration flows, and the solid lines illustrate runtime flows. J2EE applications on WebSphere Application Server can call a local Runtime Security Services client instance, which can use decisions from external rules to permit or deny access.

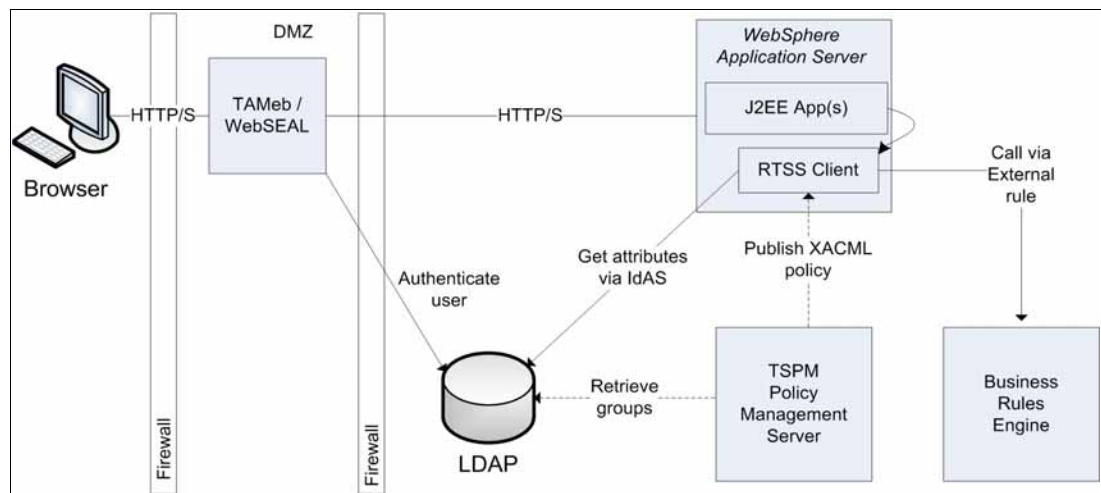


Figure 8 Application entitlements management solution

The following steps show a step-by-step iteration of this solution:

1. Applications running on WebSphere Application Server are modified to call the Tivoli Security Policy Manager Authorization API for the relevant platform. The API can serialize the current user's identity and group information automatically.
2. Application-specific attributes can be added to the request for use in the policy. Application owners develop a series of *context handlers*, which automatically extract the information from the appropriate data structures within the application. For example, an

application working with insurance claims can pass the last date that a claim was made as the attribute `lastClaimDate`.

- Each application has a descriptive name and a unique identifier that is used to partition the policy into separate domains. To represent the policy for a business application in Tivoli Security Policy Manager, a new service of type *Application* is created with the service name and identifier for the application.
- A set of appropriate actions for this application are then defined. For example, an application working with insurance claims can define *approve*, *deny*, *create*, *view*, *update*, and *delete*.
- Each resource is identified by a unique identifier, and a number of actions can be performed on each resource. For example, the business application might perform the action *read* on the resource *customer-information*.
- After the actions are defined, the application's policy attachment points are defined in a tree hierarchy. These elements directly correlate to the resources that are specified by the API at run time. For example, the *customer-information* resource can be placed within a *customer data* tree branch along with other resources such as *claim-history* and *payment-history*.
- A policy can then be attached to any point in the tree of application resources. Classifications can also be used to attach policy for common authorization requirements.
- After the policy is defined and policy attributes are mapped to concrete sources from the IT environment, the policy can be distributed to the Runtime Security Services instances that are configured in the environment.
- Applications not running on WebSphere Application Server can call a remote instance of the Runtime Security Services authorization service using the XACML over SOAP protocol as illustrated in Figure 9.

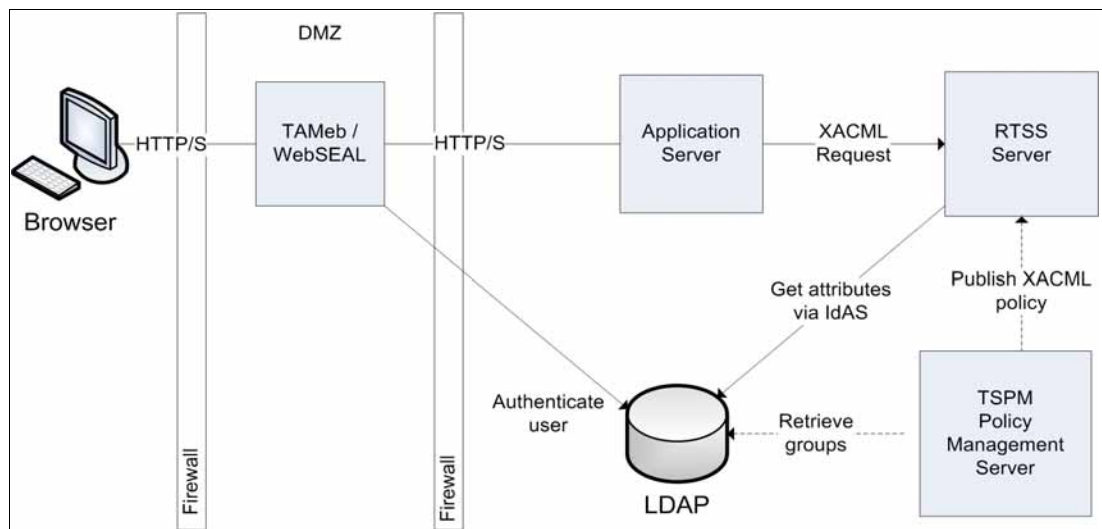


Figure 9 Application entitlements management solution, continued

10. The policy for each application can be distributed separately, allowing each Runtime Security Services server or Runtime Security Services client instance to contain only the relevant policy as illustrated in Figure 10. This partial replication of policy allows for large environments to be managed in a scalable and efficient way.

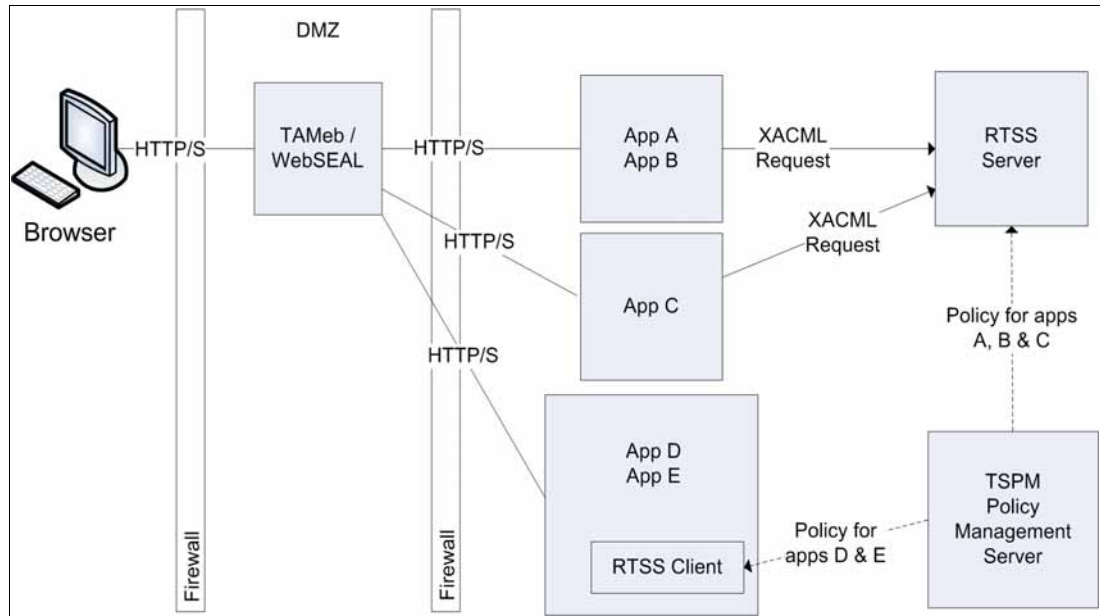


Figure 10 Application entitlements management solution, continued

SOA security policy management use case

In this scenario, we address security policy management issues in an SOA environment.

Scenario description

In this scenario, a large financial services firm has a growing SOA deployment, including a number of internal applications that are composed of reusable services. This company is looking to make a subset of these services available to their business partners but is concerned about the security implications of doing so.

The Chief Security Architect needs to ensure that:

- ▶ Messages containing confidential information cannot be viewed by unauthorized parties.
- ▶ All messages cannot be modified in transit.
- ▶ Messages are received only from business partners with the appropriate permission.

To fulfill these requirements, the architect must define an appropriate combination of message protection and authorization policies. The architect also needs to ensure that these policies are part of the service life cycle and are enforced accurately in the IT environment.

The IT operations team needs to translate the architect's policies into operational policy accurately and configure the environment. They need to reduce administrative overhead to ensure that allowing their business partners access to these services is cost-effective.

Solution

The company uses Tivoli Security Policy Manager to define and distribute the message protection and authorization policy and uses IBM WebSphere DataPower SOA Appliances as

the enforcement point for both types of policy, running in the DMZ to protect the internal resources.

The company uses WebSphere Service Registry and Repository as the definitive store for service definitions in the enterprise. These definitions are discovered by Tivoli Security Policy Manager, and authored message protection policy is distributed back to WebSphere Service Registry and Repository because WS-SecurityPolicy. DataPower retrieves updates for WS-SecurityPolicy automatically from WebSphere Service Registry and Repository, ensuring that the service is protected.

There are two options for enforcing the access control policy authored by Tivoli Security Policy Manager. The first option is to perform the authorization on the DataPower appliance itself. The appliances contain an XACML decision engine, and Tivoli Security Policy Manager provides an integration that allows policies to be distributed to the appliance and used by this on-board PDP as shown in Figure 11.

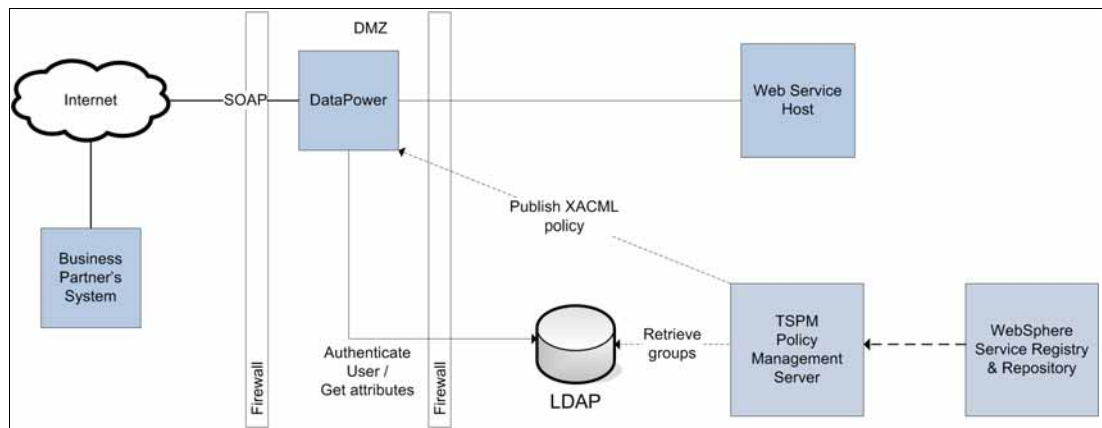


Figure 11 SOA security policy management solution

The second option is to delegate the authorization from DataPower to a remote Runtime Security Services instance, using the natively supported XACML request and response protocol. If the policy requires context information from custom PIPs or JDBC data sources or if it includes external rules, then using a Runtime Security Services server is the only option. Figure 12 shows this deployment.

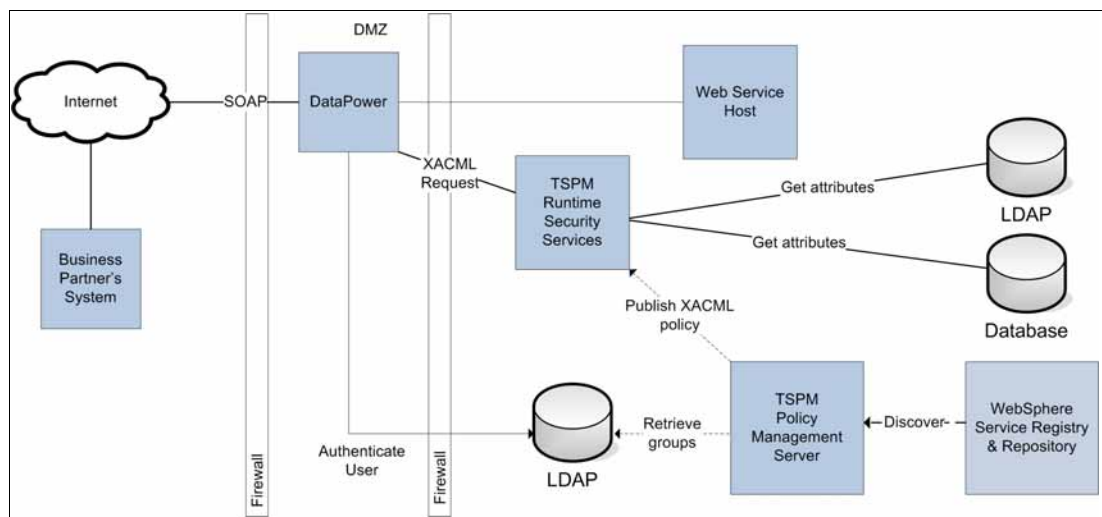


Figure 12 SOA security policy management solution, continued

For this scenario, the on-board PDP is used to evaluate the policy. No external rules or custom data sources are used, and Tivoli Security Policy Manager provides an integration for DataPower to retrieve attributes from an LDAP server, so the on-board policy decision point meets the requirements of the scenario.

The company uses a number of advanced capabilities to enhance the security of the Web services:

- ▶ Classifications are used to manage common policy based on the business requirements.
- ▶ The business partner's account status is extracted from an LDAP server to ensure that only active accounts are allowed access.

The following steps show a step-by-step iteration of this solution:

1. The Tivoli Security Policy Manager Policy Server and Console is used to define the required security policy.
2. Services are discovered from WebSphere Service Registry and Repository and imported into the Tivoli Security Policy Manager console.
3. Classifications are created for each group of security policy requirements and are associated with combinations of message protection and authorization policy depending on the particular business requirement, as follows:
 - A *Confidential* classification containing a message protection policy specifying that all messages are encrypted, and an authorization policy ensuring that callers are in the role *Confidential Allowed*
 - A *Protect integrity* classification containing a message protection policy specifying that all messages must be signed
 - An *Account active* classification containing an authorization policy specifying that any authenticated partner can access the service, as long as the business partner's *account-active* attribute is set to true.
4. The message protection policy is exported back to WebSphere Service Registry and Repository. The DataPower appliances then subscribe to updates from WebSphere Service Registry and Repository, and the updated policy is applied automatically. This setup is the recommended setup for this scenario and is illustrated in Figure 13.

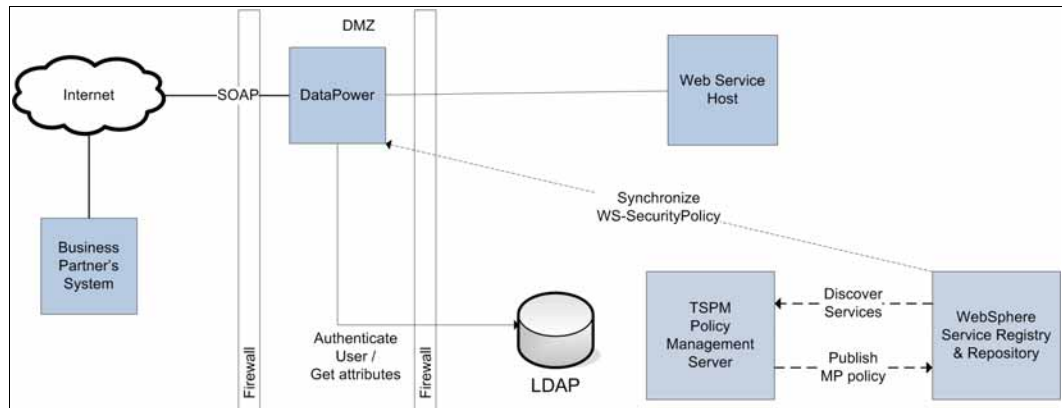


Figure 13 SOA security policy management solution - contd.

5. The IT Operations staff configures the authorization policy to map the information specified by the security architect to concrete values. This involves binding the *account-active* attribute to an LDAP attribute on a specific server, including specifying the

credentials used to bind to the LDAP server. In this scenario, this binding is done on the DataPower appliance.

6. After the authorization policy is authored and configured, it is distributed to the DataPower appliance for enforcement at run time.

Conclusion

Increasing external regulations and the adoption of SOA create new challenges for business securing access to their services and applications in today's IT environment. By externalizing and centrally managing security policy, businesses can ensure that policy aligns with business goals and is applied consistently across the enterprise.

IBM Tivoli Security Policy Manager is a standards-based solution that provides centralized application entitlements management, SOA security policy management, and "security as runtime services." It strengthens access control for applications, improves compliance, and drives operational governance across the enterprise. Discussing relevant scenarios and their solutions, this paper has shown how IBM Tivoli Security Policy Manager addresses these challenges in entitlements management and SOA security policy management.

The team that wrote this paper

This paper was produced by a team of specialists from around the world working with the International Technical Support Organization (ITSO).

Axel Buecker is a Certified Consulting Software IT Specialist at the International Technical Support Organization, Austin Center. He writes extensively and teaches IBM classes worldwide on areas of Software Security Architecture and Network Computing Technologies. He holds a degree in computer science from the University of Bremen, Germany. He has 22 years of experience in a variety of areas related to Workstation and Systems Management, Network Computing, and e-business Solutions. Before joining the ITSO in March 2000, Axel worked for IBM in Germany as a Senior IT Specialist in Software Security Architecture.

Craig Forster is a Staff Software Engineer with IBM Tivoli Software Security at the Gold Coast in Australia. He joined IBM in 2005 after graduating from the University of Queensland holding a Bachelor of Engineering (Software) and Bachelor of Science (Computer Science). His areas of expertise include Web services security, J2EE security, and authorization best practices.

Sridhar Muppidi is a Senior Security Architect at IBM Software Group. As a Senior Technical Staff Member, he drives security architecture and design activities across IBM security products. He has the responsibility for SOA Security Architecture and SOA security solutions. He is a lead architect for IBM Security Policy Management solution, His responsibilities also include providing secure solutions to enterprises, working on new product development, and representing IBM in standards activities. He holds a Ph.D. in Computer Science and has published extensively.

Borna Safabakhsh is a software developer in the U.S. He joined IBM in 2004 after graduating from the Georgia Institute of Technology with a Master in Computer Science. His areas of expertise include plug-in based architectures, enterprise software security, and software engineering.

Thanks to the following people for their contributions to this project:

Debbie Willmschen, editor
International Technical Support Organization

Ravi Srinivasan, Eric Wood, Neil Readshaw, Liz Hughes
IBM

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

This document REDP-4483-00 was created or updated on February 17, 2009.

Send us your comments in one of the following ways:

- ▶ Use the online **Contact us** review Redbooks form found at:
ibm.com/redbooks
- ▶ Send your comments in an e-mail to:
redbooks@us.ibm.com
- ▶ Mail your comments to:
IBM Corporation, International Technical Support Organization
Dept. HYTD Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400 U.S.A.




Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

DataPower®
DB2®

IBM®
Redbooks (logo) ®

Tivoli®
WebSphere®

The following terms are trademarks of other companies:

J2EE, Java, JDBC, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Active Directory, SharePoint, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.