Publishing and Consuming Services with Policy Manager®

SOA | software™

Publishing and Consuming Services with Policy Manager

1.0
September, 2013

Copyright

Trademarks

SOA Software, Inc.

SOA Software, Inc.
12100 Wilshire Blvd, Suite 1800
Los Angeles, CA 90025
(866) SOA-9876
www.soa.com
info@soa.com

Disclaimer

Contents

Chapter 1 | Products Overview 8

Chapter 2 | Concepts Overview 10
- Organizational Structure and IDs 10
  - Organization Tree Functionality 11
- Workbench Objects 12
  - Workbench Object Relationships 14
- Containers 15
  - SOA Containers 16
  - SOA Container Architecture 19
  - SOA Container Deployment Modes 20
    - Standalone Proxy Deployment 20
    - SOA Container Deployed on Consumer Node 21
    - SOA Container Deployed on Provider Node 21
    - Provider Side SOA Container in WebSphere 21
    - Consumer Side SOA Container in WebSphere 22
  - Network Director (ND) 22
  - Agents (TC Server, WebSphere) 23
  - Policy Manager for IBM WebSphere DataPower (PMDP) 23
  - Delegates 23
- Services 24
  - Access Points 24
  - Schemas 24
  - Interfaces 24
  - Bindings 24
- Policies 25
  - Operational 25
  - Compliance 26
  - QoS 27
- Contracts 27
  - Provided Contracts 29
  - Consumed Contracts 29
  - Contract Scope 29
  - Consumer Identities 29
  - Contract Metadata 30

Chapter 3 | Capabilities 31
- Search 31
- Publishing 31
- Workflow 31
- Mediation 32

Monitoring 32
Alerts 32
Logs 32
Real-Time Charts 33
Historical Charts 33
Dependencies 33
Security 34
Auditing 34
Categorization/Classification 34

Chapter 4 | Managing Users and Security 36
User Roles and Responsibilities 36
Workbench Security 36
Identity Categories 36
Consumer Identities 36
End User Identities 36
Policy Manager Users 37
Service Identities 37
Container Identity 37
Access Control Policies 37
Access Control Model 37
Access Control Roles 38
Access Control Privileges 39
Administrator Role and Privilege 41
Default Role Definitions 42
What are the Role Definitions? 42
How do I View a Summary of current Roles? 42
Role Definitions Portlet 44
How do I Add a Role Definition? 44
How do I Modify a Role Definition? 48
How do I Delete a Role Definition? 52
Role Memberships 53
How do I View a Summary of current Role Memberships? 53
How do I Manage a Role? 54
Creating and Managing Keys and Certificates 56
Certificate/Key Management Options 56
Generate Options 56
Import Options 58
Export Options 59
Delete Options 61
Key Management for Users 61
Key Management for Services 64

Chapter 5 | Auditing Alerts and Security Activities 65
How do I Perform an Audit Trail Search? 65
How do I Configure an Audit Trail Status? 67
How do I View Audit Trail Details? 67
How do I Export an Alert Audit Trail? 68
How do I Schedule an Alert Audit Trail Job? 71
Security Audit Trails 72
How do I Configure a Security Audit Trail Status? 72
How do I View Security Audit Trail Details? 72
How do I Export a Security Audit Trail? 73
How do I Schedule a Security Audit Trail Job? 75

Chapter 6 | Publishing Services 77
Physical Services 77
Virtual Services 77
Hosting a Service 80
Attaching Policies 94
Managing Schemas 95
How Do I Add New Schemas? 95
Managing Interfaces 98
How do I Import an Interface into Policy Manager? 98
How do I Model an Interface using Existing Schemas or Interfaces? 99

Chapter 7 | Configuration Migration 108
What Information Models does the Configuration Migration Process Support? (Organization, Service, Contract, Policy) 108
What is the General Process for Exporting and Importing Information Models? 108
To export a policy 109
To import a package 114
Services Workflow 116
How do I Configure Workflow for a Service? 116

Chapter 8 | Policies 119
Adding Policies (General Process) 121
Policy Use Cases 130
WS-Security Username/Password Use Case 131
X-509 Authentication Use Case 132
Basic Authentication Using WS-Security Policy Use Case 132
Basic Authentication Using HTTP Security Policy Use Case 133
X.509 Authentication and Signature Verification Use Case 134
Attaching Policies 135
Policy Actions 137
How do I make a Copy of a Policy? 137
How do I Delete a Policy? 139

How do I Modify Policy Details? 140
Managing Policy References 142
How do I View Policy References? 142
How do I Manage Policy References? 145

Chapter 9 | Consuming Services 146
Search 146
How do I query, manage, and secure existing web services using Search? 146
Browse 147
What is the Workbench Organizational Hierarchy? 147
What are the Workbench views? [Root, Sub-org, Services, Contracts, Policies, Container, Processes (PM70), Scripts (PM70)] 147
How do I Browse the Workbench? 148
Create a Contract 148
How do I create a Provided Contract? 148
How do I create a Consumed Contract? 151
How do I define a Contract Scope? 157
How do I define a Consumer Identity? 159
How do I add Contract Metadata? 162
How do I attach a QoS Policy to a Contract? 165

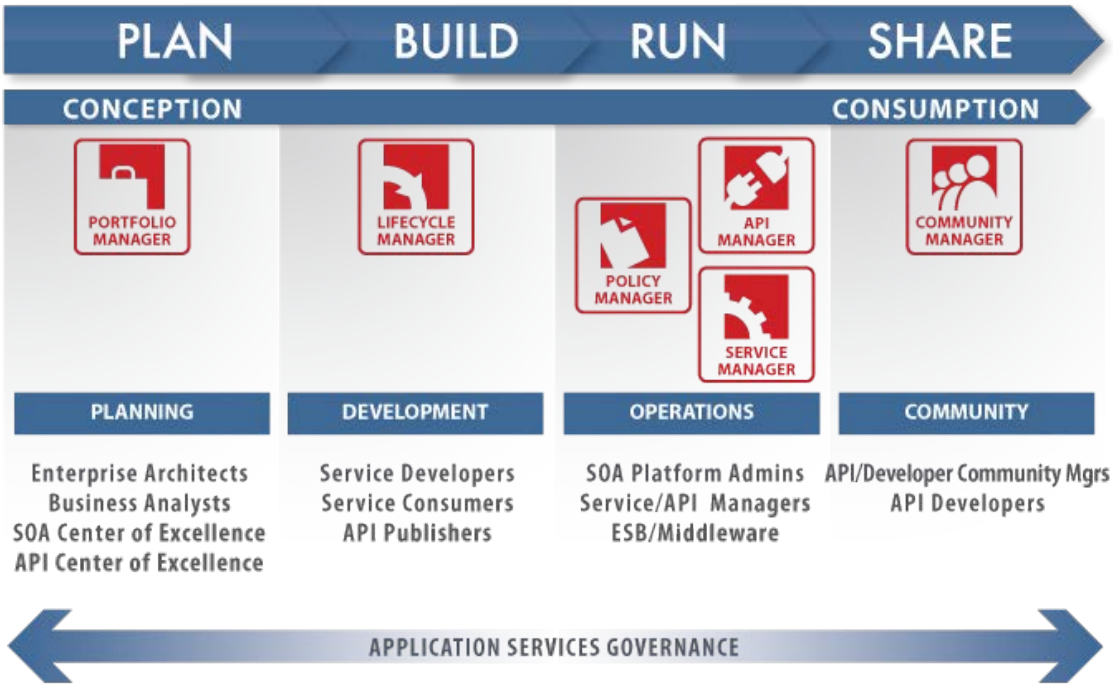
Chapter 10 | Managing Workflow 167
Workflow Control Portlet Functionality 167
Workflow Task Portlet 167
Services Workflow Portlet 168
Contract Workflow Portlet 171

Chapter 11 | Using Delegates 174
Delegate Handler 174
Handler Properties 174
Identity Credentials 175
Using the Gateway Agent 178
Configuring the Gateway Agent 180
Gateway Agent Configuration Properties 180
Property Prefix and Suffix Processing 181
Using Property Files 182
Gateway Agent Configuration Property Details 183
Gateway Agent Processing of User Credentials 186
Using Service Binding Keys 192
Encrypting Gateway Agent Configuration Passwords 193
Using the Gateway Agent with Apache Axis 1.x 194
Using the Gateway Agent without J2EE or JSR109 196
Enabling SOA Client Gateway Agent with Service Delegates 198
Configuring the SOA Client Gateway Agent from a Delegate 199

Advanced Service Delegate Base Class Features 201
Managing ServiceLocator Caching 201
Extending Client Message Handler Processing 201
Using the Gateway Agent Outside of a SOAP Container 202
Installation and Configuration 202
Using the Bare Message Feature 202
Initializing the Bare Message Gateway Handler 203
Processing Bare Message SOAP Requests 203

Chapter 1 | Products Overview

SOA Software builds its Application Services Governance solution around its Community Manager™, Policy Manager™, Portfolio Manager™, Lifecycle Manager™, and Service Manager™ products for API and SOA Developer Engagement, Planning, Development, and Operational Governance.



SOA Software's Community Manager™, Portfolio Manager™, Lifecycle Manager™, Policy Manager™, and Service Manager™ combine to form a comprehensive Application Services Governance Automation solution.

Community Manager™ is a sophisticated developer community product to help enterprises attract, manage, and support the developers that build Apps using their APIs. It provides an extensive set of social capabilities to promote the creation of a community of developers either inside or outside the enterprise, or a combination of both. These developers collaborate to form an innovation engine to drive new channels and improve existing business processes. The network-effect of the social capabilities enable a community commons model for social contribution to the APIs themselves through support and documentation, and to the Apps the developer community creates.

Portfolio Manager™ is an innovative Planning Governance product that helps ensure the alignment of API and SOA Programs with strategic IT investment and business objectives and makes sure that enterprises build the right services at the right time. It helps customers identify candidate services and build an API and SOA roadmap through Modeling, Asset Identification, and a Portfolio Management process. To achieve these goals Portfolio Manager functions as part of a unified Application Services Governance automation suite with seamless integration with Lifecycle Manager™ and Policy Manager™.

Lifecycle Manager™ provides an advanced software development asset (SDA) repository, lifecycle management, and metadata federation solution. It governs leading development platforms, ensuring consistent definition and management of services and other assets across all development environments. Lifecycle Manager supports advanced SDA repository and governance capabilities including the ability to define and manage custom asset and artifact types, asset relationship management, integrated development environment (IDE) integration, and comprehensive asset federation. It integrates seamlessly with Community Manager to publish API definitions and documentation, and to provide workflow for the developer consumption process. It also integrates with Policy Manager where policy decisions are required in the Development Governance process, as well as provisions service consumption agreements made by developers to Policy Manager for further governance. Lifecycle Manager supports application development and architecture teams, providing a comprehensive Development Governance solution.

Service Manager™ automatically implements and enforces policies from Policy Manager. It generates usage, performance and policy compliance metrics that it reports to Policy Manager and into Community Manager so that it can audit that policies are being enforced in a closed-loop process. Service Manager supports API and SOA and enterprise operational management functions, ensuring that services are security, reliable, and meet the performance goals for each consumer.

Policy Manager™ provides a runtime registry/repository and comprehensive Policy Governance solution for APIs and SOA services. Policy Manager includes a built-in policy and service metadata repository supporting its policy governance processes. Policy Manager supports enterprise API and SOA architecture functions, ensuring consistent application of policies throughout enterprise API and SOA programs. Using this solution architects, developers, security administrators, and operations managers can define and govern policies that are applied to services throughout the appropriate stages of their lifecycle.

Chapter 2 | Concepts Overview

This section introduces the core concepts you need to understand in order to properly use the products.

Organizational Structure and IDs

The Workbench "Organization Tree" contains descriptions of businesses or organizations that provide web services that are part of your Policy Manager deployment. The organizational hierarchy is structured with a Root Organization at the top level tier and Sub-Organizations and second level and subsequent tiers. Each Sub-Organization includes "Services," "Contracts," "Policies," and "Containers" Workbench Objects. See Workbench Objects.

The Root Organization is the "Registry" repository where your Policy Manager data is stored and represents an "Organization Tree" node that is not a child of any other Organization. All Organizations are children or other descendants of the Root Organization. You can rename the Root Organization as needed based on your business requirements.

When you add a new "Organization" it is added to the "Organization Tree." You build your organizational hierarchy one organization at a time. Any tier level of the Organization Tree can be a "Parent Organization." Organizations at the same level are peers. You can populate the Organization Tree with additional tier levels (i.e., sub-organizations) and can also designate these sub-organizations as "Parent Organizations" based on your requirements. If you are just starting and have not defined any organizations, the "Parent Organization" will be the "Root" organization.

Note: Organizations added to the Organization Tree are sorted alphabetically.

Organization Tree Functionality

The following list provides an overview of key functionality supported by the Policy Manager "Workbench."

- Organization Creation and Type

  Organizations are created in the Workbench service environment and can be assigned up to four different organization types (i.e., Application, Company, Department, or Project).

- Organization Tiers

  Organizations can include other organizations. There is no limit on the number of tiers that are supported.

- Delegated Management & Administration

  Organizations support delegated management and administration. Organization Administrators can create new services, Containers, and sub-organizations within their organization.

- Permission Assignment

  Read or Modify permissions can be assigned to a User or User Group for an organization.

- Movement of Services Between Organizations

  An administrator with modify privileges can move services between organizations.

- Tree Structure to Represent Organizational Hierarchy

  The Organization hierarchy is represented as a tree that shows entities to the level of Containers. The tree does not show individual operations of a service.

- Derivation of Organizational Hierarchy

  The organization hierarchy as well as the objects (i.e., Containers, Provided and Consumed Services) are obtained as the result of the existing search functionality in the Policy Manager.

- Composition of Organizational Hierarchy

  The organization hierarchy includes the following collections/folders at each node:

  - Services
  - Contracts (Consumed and Provided)
  - Containers
  - Policies

- Extension of Existing Search Functionality

  The existing search functionality supports searching for organizations in addition to the existing services.

- Search Parameters for Services and Organizations

  The search parameters for both services and organizations include the following:

  - Container Name. This includes all services and/or organizations that are managed by or owned by those entities.
  - Organization. This includes name, type, and tree position. Search produces results that include all services belonging to those organizations or the organizations themselves.

- Consumed Services Node

  For each organizational entity, the Organization Tree lists the Services Consumed by that organization. This "Services Consumed" node includes both services and entire organizations. This implies that all the services of the organization can consumed by the consumer. The Organization Tree does not support a drill-down into the details of the consumed organizations.

Workbench Objects

A Workbench Object is an element in an Organization Tree that contains a specific type of "metrics" information pertaining to the web service. Metrics information for each object includes configuration, security (i.e., access control and policy), attribute, and state information.

Objects that display in the "Organization Tree" are first class objects meaning that they can be used in Workbench without restriction and have intrinsic identity. The first class object is a complete entity unto itself, includes all the privileges and properties, and can be passed to functions and returned from them.

Core functionality of the first class object is presented in the Workbench as "Portlets." External attribute information related to a particular function is via tabs.

In Policy Manager, an Organization Tree includes the following object types:

- Root Object

  The "Root Object" represents the top of the "Organization Tree" and is defined during your initial Policy Manager configuration. The "Root Object" name represents the name of the Policy Manager

"Registry." This name can be customized using the "Modify Organization" action. Actions that can be performed at the root level include Add Organization, Add Container, Create Physical Service, Create Virtual Service, Rename Organization, Import Package, and Add Policy. The "System Summary" provides an overview of "state" information for entities defined within your Organizational Hierarchy.

- Organization Object

  The "Organization Object" includes description information about the Organization (i.e., business entity) and includes a Services, Contracts, Containers, and Policies folder. You populate the "Organization Tree" at the root level using the "Add Organization Wizard." Organizations" are structured using a Parent/Child Hierarchy.

- Service Object

  The "Services Object" stores services that have been added to the Organization using the "Create Physical Service," "Create Virtual Service," and "Virtualize Service" wizards. Within the Services Object you can perform service management activities including policy management, access point creation, category management, rule management, and usage data monitoring. You can also transition a service to a new provider by changing the provider organization using the "Select Provider Organization" function.

- Contract Object

  A "Contract" is a document that provisions the expected utilization of services. Each contract is configured with an access control method that represents the method a service uses to enforce a contract. There are two types of access control that can be assigned to a contract. A service can enforce a contract by authorizing an application or Consumer Organization Identities to use a service, OR a service can use a default contract and allow consumer (application and organization) users that do not have a contract explicitly assigned.

  You define contracts at the root level using the "Add Contract Wizard." In order to create a contract you must have at least one Organization defined that is assigned to the contract as either a Provider or Consumer Organization. The contract definition is added the Provided or Consumed Contracts folder of the Organizations selected during the configuration process. Here you can modify the contract, define contract scope, consumer identities, assign QoS policies, and configure contract metadata.

- Policies Object

  The "Policies Object" provides a platform for managing the creation and maintenance of WS-Security and compatibility policies (i.e., Pipeline Policies) that comprise your Policy Manager deployment. The policy framework supports delegated administration which means that policies can be assigned at the Organization and Service Operation levels. Based on your security requirements, you can configure policies as single entities or you can define a policy group (i.e., Aggregate Policy) that includes two or more policies. The "Add Policy Wizard" is used to create a policy. Policy types that support inclusion in an "Aggregate Policy" A subset of policy types can be included in an "Aggregate Policy." Policies can be attached to different governable entities in the Policy Manager "Workbench" including Organizations, Services, Operations, Access Points, Bindings, and Messages.

- Container Object

  The "Container Objects" stores containers that are utilized by services within the current Organization. It provides tools for viewing and modifying the listener configuration of a container, configuring and managing virtual service hosting of containers, and cloning the configuration of a "source" container to a "target" container.

  Containers are initially defined using the Policy Manager "Add Container Wizard." and can then be viewed within its assigned organization in the Containers folder.

  Each Container Cluster has its own tree hierarchy which can be expanded to view Members (i.e., Cluster Nodes). Container Types that are independent (i.e., not a Member of a Cluster) do not include an expandable tree hierarchy.

Figure 1: Container Object with Cluster Hierarchy in Organization Tree

Workbench Object Relationships

Main objects in the Policy Manager are: Organization, Service, Contract, Container, Policy and Identity. All of these objects are associated with different other objects in different ways. For example:

- Service
  - Service belongs to an organization.
  - Service can only be managed by a Container that belongs to the same organization (or ancestor organization) to which service belongs to.
  - Service can use only the policies defined at the organization that the service belongs to or its ancestor organization.
  - Service can only be part of the scope of the contract with the provider organization as the organization the service belongs to or the ancestor organization.
  - Service Identity can only be part of the consumer identities of the contract with the consumer organization as the organization the service belongs to or the ancestor organization.
- Container
  - Container belongs to an organization.
  - Container can only host/manage the services that belong to the same organization as the container or its descendent organization.
  - Container Identity can only be part of the consumer identities of the contract with the consumer organization as the organization the container belongs to or the ancestor organization.
- Organization
  - Organization belongs to its parent organization except the “Root” or “Registry” organization that does not have any parent Organization.
  - Organization contains Services.

  - Organization contains Containers.
  - Organization contains Contracts when it is either the provider or consumer organization of the contract.
  - Organization contains policy definitions.
  - Organization has identities assigned to it.
- Contract
  - Contract has a provider organization.
  - Contract has a consumer organization.
  - Contract has a scope with services. Contract scope contains only the services that are provided by the provider organization sub-tree of the contract.
  - Contract has a set consumer organization identities. Contract's consumer identities contain only the identities that are assigned to consumer organization sub-tree of the contract. Here the word "assigned" means that the identity assigned to any object that is part of the consumer organization sub-tree.
  - Contract has references to QoS Policies. Any contract can only refer to the QoS Policies defined at the provider organization of the contract or its ancestors.
- Policy
  - Policy belongs to an organization. Policy can only be defined at the root or Registry Organization.
  - Policy can only be attached a policy attachment point (service or endpoint or binding or operation or input/output/fault messages or contract) that belongs to the same organization as policy defined at or to a descendent organization.
- Identities
  - Organization can have identities assigned. The purpose of assigning identities to the organization is this. If the organization has consumer applications that are not web services, then Policy Manager only cares about their identity and hence is the need for assigning identities to the organization.
  - Identities are also assigned to Service. Every service has a default identity that service manager creates in local domain. Policy Manager also provides a way to assign a Kerberos identity to the service, in which case this Kerberos identity is the second identity assigned to the service.
  - Container also has an identity. In the Venice release, container identities are internally created and the name of the identity will be same the container key.
  - Any identity (organization identities, service identities or container identity) can only be part of the contract with consumer organization as the organization that identity belongs to (or the organization that the entity with that identity belongs to) or any of its ancestor organizations.

Containers

SOA Software provides a high performance web service management solution that is composed a series of default web services that represent the core product functionality and an intermediary layer referred to as the "Policy Manager" container.

Physical and Virtual web services are deployed inside a "Container." A Container acts as the service "host," executes the web service policy configuration and processes request and response messages

associated with service transaction activity. Container deployment modes (SOA Container—Virtual Services, 5.2 Embedded Management Point—Physical Services, 5.2 Standalone Management Point—Virtual Services) are configured using the Policy Manager and can be deployed in a variety of different scenarios. The act of deployment establishes a relationship between Policy Manager and Containers. The Container then exposes the Policy Manager functionality and web service configuration data to the web services.

When a request is made to a web service, the Container applies the appropriate security policies to the physical and virtual web services, and also captures usage "roll-up" data. You can monitor system activity using the Policy Manager "Dashboard," "Monitoring," or "Alerts" functionality. Maintenance of your service configuration is also performed using Policy Manager.

Note: The utilization of Policy Manager configuration data by one or more Containers is a requirement in order to guarantee optimum performance of the Policy Manager solution. This means that Policy Manager and associated Containers must be running and the Management Console must be launched.

SOA Containers

The SOA Container mediates web service message exchanges between service consumers and service providers. It enforces policies, monitors and reports performance metrics and events, integrates services through virtualization, and provides auditing capabilities. In short, it provides the runtime execution of the Policy Manager capabilities. As a point of reference the following diagram illustrates how the SOA Container fits in to the overall Policy Manager architecture.

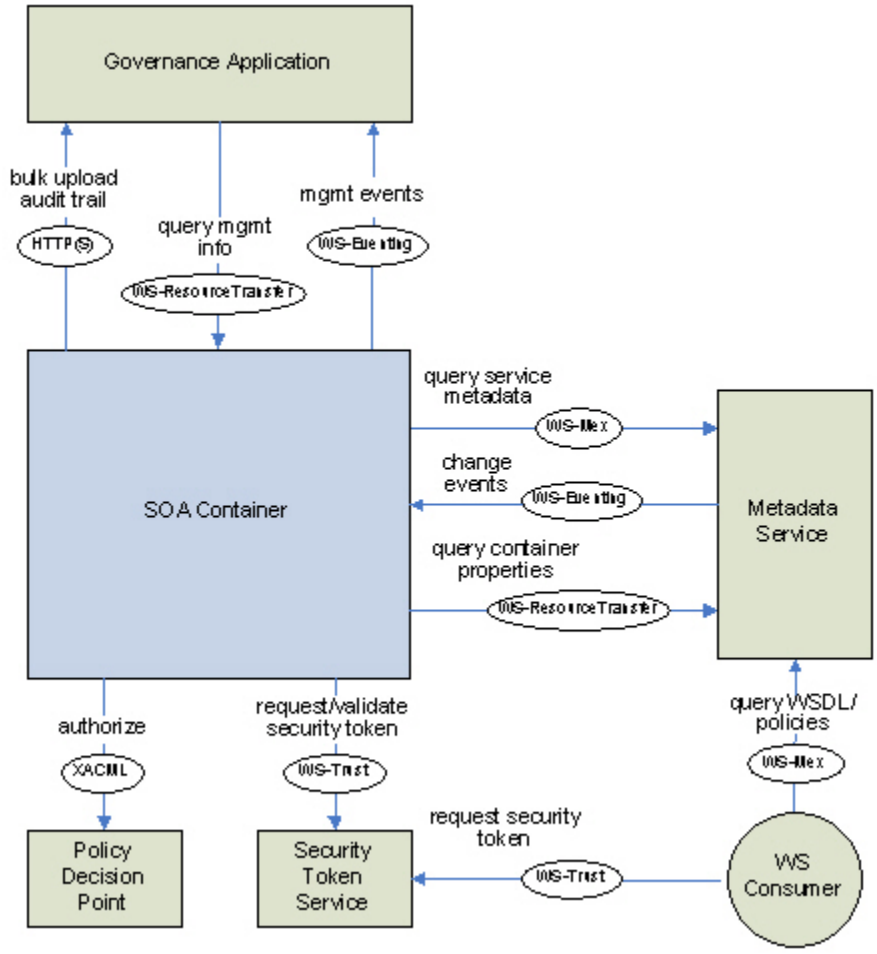


Figure 2: Policy Manager Architecture

The SOA Container component can be deployed on both the consumer and provider side of a message exchange. The SOA Container can be embedded within a consumer or provider process, or it can be deployed as a proxy in a separate process.

Figure 3: SOA Container embedded in Consumers and Providers

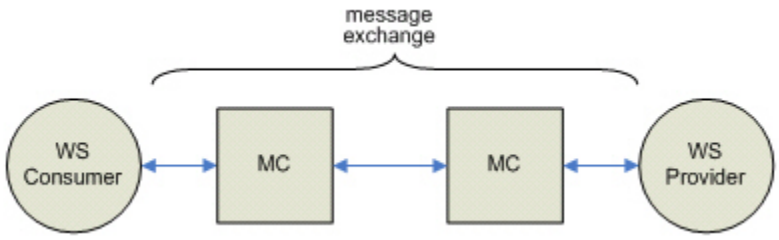


Figure 4: SOA Container as Proxy between Consumers and Providers

In addition to the typical request-response exchange pattern, the SOA Container also supports the one-way exchange pattern. This gives it the ability to easily monitor and mediate events, whether they are delivered in a proprietary way or as defined by well-known specifications such as WS-Eventing and WS-Notification.

Figure 5: SOA Container embedded in Publishers and Subscribers

As an additional deployment option, the SOA Container can be deployed in a J2EE container to provide a tighter integration with a J2EE web service supplier or consumer. All of the flows previously described still apply, just that the link between the SOA Container and the party it's servicing is in-memory and governed by the application server.

With the transitional state of management standards such as WS-Management and WSDM, the SOA Container supports a mix of standards that are closely aligned with the management standards convergence roadmap. The SOA Container provides access to management information identified in the WSDM Management of Web Services (MOWS) specification through a WS-ResourceTransfer compliant interface.

SOA Container Architecture

The SOA Container can be deployed standalone or integrated with a web service container. The primary components and interfaces are the same in both deployments.

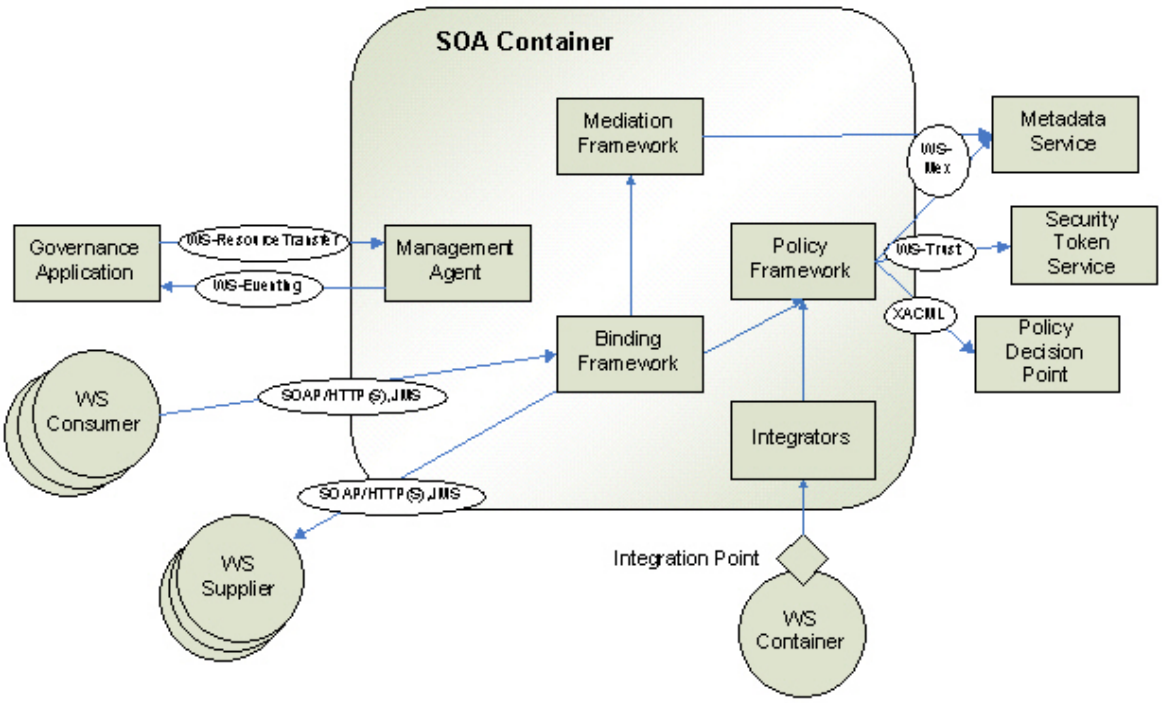


Figure 6: SOA Container Overview

The Policy Framework interprets the effective policy for incoming messages and performs the appropriate actions. The policies are retrieved from the Metadata Service in WSDL documents retrieved using a WS-MetadataExchange interface. The Policy Framework has a modular and extensible design, whereby the different types of policies the SOA Container supports are implemented by independent policy modules, or handlers. Based on the policy expressions found in the effective policy, the Policy Framework will invoke a set of these handlers. Some handlers will have to make use of services external to the SOA Container. For example, an authentication handler will interact with a WS-Trust Security Token Service. An authorization handler will interact with a XACML Policy Decision Point.

The Mediation Framework performs a sequential list of activities, such as transformation, on the input and output messages of a message exchange. The list of activities is defined as metadata for a service that is retrieved from the Metadata Service. The Mediation Framework has a modular and extensible design, whereby the different activities the SOA Container supports are implemented by independent components.

The Management Agent acts as a true “Management Agent” as defined by the WSDM specification. It provides all management capabilities for the SOA Container including metrics gathering and event generation.

The SOA Container can integrate directly with a service container (IIS, WebSphere, WebLogic) and it can act as an independent service container providing its own transport bindings within the Binding Framework This framework is extensible as each transport binding is implemented as an independent module. For example, SOAP and REST are two different bindings available in the SOA Container. New bindings can easily be added. Typically each of the bindings also has a modular and extensible design whereby physical transports can be changed.

The SOA Container integrates with a service container at what is called the Integration point. The Integration Point is an integration mechanism that is provided by the container and may be container specific. For example, the Integration Point for HTTP requests in the WebSphere container is a ServletFilter chain. The Integration Point for JMS requests in the WebSphere container is a JAX-RPC handler chain.

An Integrator connects the SOA Container with the web service container at the Integration Point. Continuing the WebSphere container example from the previous paragraph, the Integrator for HTTP requests is a ServletFilter. The Integrator for JMS requests is a JAX-RPC handler.

The Integration points in the J2EE environment pose deployment challenges because a J2EE container can host multiple web services that are implemented as very independent web applications. The container most likely does not provide a central Integration Point for all web service applications. This requires integrating with each web service in the container, which can be burdensome to customers. To provide easier integration with J2EE web services, the Management Agent will provide service discovery and dynamic integration capabilities.

SOA Container Deployment Modes

There are two primary deployment scenarios for the SOA Container, the Standalone and Embedded deployments.

Standalone Proxy Deployment

In a standalone deployment the SOA Container is deployed as a single Java executable program. This program can be deployed so that it starts when the node it is deployed on is started. In Windows, the program is registered as a Windows service. Since the Integration point in the consumer/supplier interaction is a proxy that relays messages between clients and services using distributed transports, the SOA Container program can be deployed using a proxy node or can be deployed on the same node as the consumer or supplier.

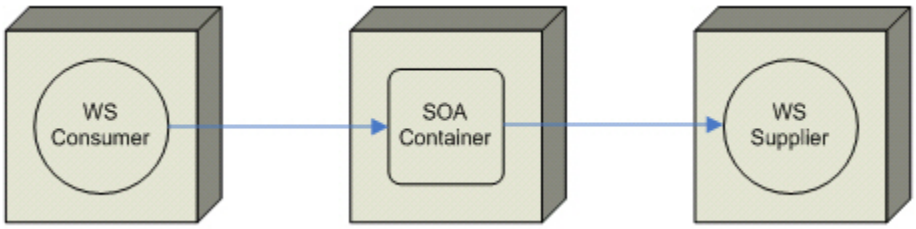


Figure 7: Standalone Proxy Deployment

SOA Container Deployed on Consumer Node

Deploying the SOA Container on a physical proxy is the most un-intrusive deployment option. The SOA Container can be introduced into the network topology without affecting the currently deployed nodes.

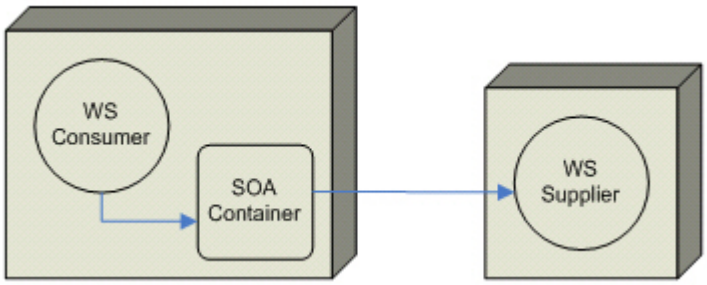


Figure 8: SOA Container Deployed on Consumer Node

SOA Container Deployed on Provider Node

Deploying the SOA Container on the client node may be practical if the customer using the SOA Container is not the service supplier, but the service consumer. The proxy deployment could still be used in this situation with the SOA Container deployed on a separate node in the consumer's domain, but as an alternative to reduce hardware costs, the SOA Container can be deployed directly on the consumer node.

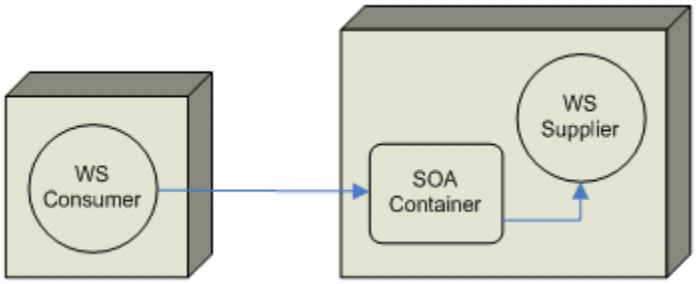


Figure 9: SOA Container Deployed on Provider Node

Provider Side SOA Container in WebSphere

The SOA Container can also be deployed directly on the service supplier node when the SOA Container is deployed by the service supplier. This may reduce hardware costs that the proxy deployment introduces.

In many cases both the service provider and consumer will utilize the capabilities of the SOA Container. The standalone SOA Container can be used by both parties together with a mix of proxy, consumer side, and supplier side deployments (not depicted here).

The embedded SOA Container deployment is more limited. It must be deployed inside the container in which the consumers and/or providers are deployed. As described previously the SOA Container is integrated with the container using an integrator that is specific to the type of container in which it is deployed. The containers currently supported are listed in the Supported Middleware section.

In a J2EE container, the SOA Container is deployed as a web application. The SOA Container does support clustering so that it can be deployed in a clustered J2EE environment. The following diagrams illustrate the deployment of the SOA Container in a WebSphere container. The first diagram shows the SOA Container managing the provider side of a message exchange transported using HTTP. In this scenario a ServletFilter is used as an Integrator.

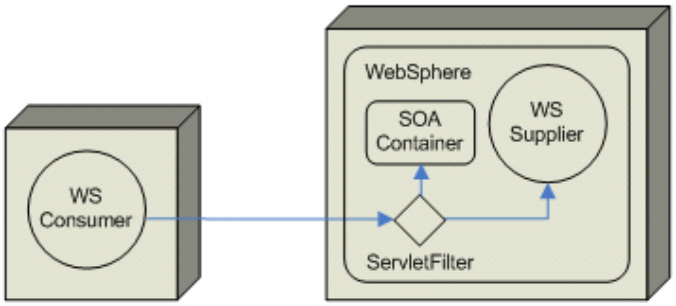


Figure 10: Provider Side SOA Container in WebSphere

Consumer Side SOA Container in WebSphere

The second diagram shows the SOA Container managing the consumer side of a message exchange. Regardless of the transport used, all client side exchanges are mediated using a client side JAX-RPC Handler.

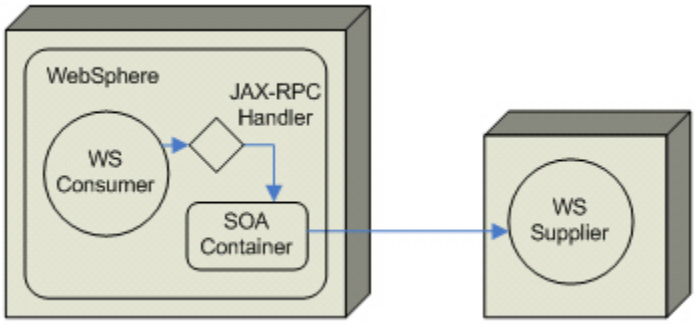


Figure 11: Consumer Side SOA Container in WebSphere

Network Director (ND)

The Network Director feature enables a Container to host Virtual Services. The Virtual Services are defined in the SOA Software Management Console. When the Network Director feature is enabled, the Container's metadata is updated to reflect this support. The metadata must be updated in the SOA Software Management Console so that it is aware that Virtual Services can be hosted on the Container. The feature also enables support for default Bindings and Policies. The default Bindings are SOAP and HTTP (REST and POX). The default Policies are WS-Addressing, WS-Auditing, WS-Security, HTTP Security, Authentication, and Authorization.

Agents (TC Server, WebSphere)

- SOA Software Policy Manager Agent for tc Server

  This feature is the policy enforcement point for the tc Server and provides WS-Policy enforcement to web services deployed to the tc Server. It supports web services deployed to HTTP endpoints.

- SOA Software Service Manager Agent for WebSphere

  This feature is the policy enforcement point for the WebSphere Application and provides WS-Policy enforcement for web services deployed to WebSphere. It can only be deployed in a Container deployed in the WebSphere Application Server.

Policy Manager for IBM WebSphere DataPower (PMDP)

The Policy Manager for IBM WebSphere DataPower is an adaptor that enables DataPower to become a Container for Policy Manager (versions 6.0 and 6.1). The Policy Manager for IBM WebSphere DataPower feature is part of the SOA Software Platform, is installed and configured using the SOA Software Administration Console, and operates in a Policy Manager 6.1 environment.

Delegates

The SOA Software Axis Delegate is an Apache Axis client solution that provides message formatting and transportation based on a web service description stored in the SOA Software Policy Manager product. The solution is implemented by a set of Axis message handlers that work in conjunction with an SOA Software "SOA Container." The SOA Container is an OSGi container that can support multiple features. When used as part of the Delegate solution, the SOA Software Delegate feature must be installed.

The architecture of the Delegate supports two different deployment options; the collocated option in which the SOA Container is deployed in the same Java Runtime Engine (JRE) as the Axis engine, and the remote option in which the SOA Container is deployed in a separate JRE from the Axis Engine. In the collocated deployment, the com.soa.delegate.axis.LocalAxisPivotHandler is installed as an Axis "pivot" handler. It is important that the handler is installed as a “pivot” handler because it will take responsibility for the transportation of the web service messages. In this deployment the transport is actually performed by the SOA Software Delegate feature in the collocated SOA Container. The LocalAxisPivotHandler merely relays the web service messages between the Axis engine and the SOA Container.

The Delegate package also contains specialized components that can be configured to perform specialized tasks such as:

- Capture Single Sign-On (SSO) Authentication Tokens issued by third-party access control products like CA eTrust SiteMinder (SMSESSION cookie) and Oracle Access Manager (ObSSOCookie cookie) for later processing by the Delegate.
    - Capture WS-Security credentials from an incoming web service request so that credential can be propagated into the outbound request in a service-chaining use case.
    - Invoke the Delegate for non-SOAP XML messages.
    - Use of the Policy Manager Delegate outside of a SOAP container in applications that are able to independently generate and process SOAP request and response messages.
    - Use a subset of Delegate features in Microsoft .Net applications.

Services

Access Points

An Access Point represents the technical fingerprint of a web service and provides the technical information needed by applications to bind and interact with the Web service being described. A service must contain at least one access point.

An Access Point is added to your Policy Manager service configuration when you register the service using one of Policy Manager's service creation wizards. You can configure one or more access points for each service based on your protocol requirements. HTTP, HTTPS, and JMS protocols are supported. After configuring an access point you can view operations assigned to an access point, and update Categories assigned to an access point.

Access Points are added to a virtual service by hosting the virtual service in a Container. The access point that is configured during this process can be viewed in on the "Access Points Summary" screen, along with the assigned "Operations" and "Categories." Updates to the access points created during the virtual service hosting process must be performed by modifying the virtual service configuration.

Schemas

When importing WSDL, all referenced schemas that describe the messages used by the services are extracted, indexed and stored in the Metadata Repository. This allows the Workbench to reuse schema between services, allowing users to see how many service references a particular schema has and therefore determine the impact of any change to a particular schema. To preserve the integrity of the metadata repository, each schema should have a unique and sensible namespace to allow for their successful correlation and discovery.

Schema information can be accessed in the "Configure > Registry > Schemas" section of the Management Console. Here you can view and add schemas, and assign/unassign schemas identifiers and categories.

Interfaces

An "Interface" defines the behavior of a service based on a set of defined operations that it implements. The "Interfaces" section provides functionality that allows you to manage the baseline of interfaces that are available for use when modeling a service. Functions are available for viewing details and references, adding, and importing interfaces.

Interface information can be accessed in the "Configure > Registry > Interfaces" section of the Management Console. Here you can add a new schema and model it with existing schemas and interfaces.

Bindings

A binding is reference to an external framework (i.e., interface) that defines how the WSDL user will reach the implementation of services. This reference specifies the protocol and data format to be used in the transmitting message defined by the associated interface.

Each binding technique is specified in the WSDL and points to the server that has access to the actual implementation of your Web service.

Policy Manager provides an extensible WSDL Binding framework that supports out-of-the-box Binding implementations and custom Binding implementations. The Binding Framework includes API's and extensions points in both the SOA Containers and the Policy Manager "Management Console" user interface. The "SOA Container" and "Management Console" portions of the framework are independent of each other however. In some situations, only the user interface of the "Management Console" is extended for a type of binding. This is the case when an "SOA Container" is not used to virtualize services using the binding.

Bindings drive the service management process and must be present and available when defining and managing service configurations. Because of this, Bindings must be added to the Policy Manager "Management Console" as a prerequisite to using the various service management wizards and access point functionality. SOAP 1.1, SOAP 1.2, Plain Old XML (POX), HTTP, and XML binding types are supported. If the specified interface does not contain one of these binding types, a custom binding can be specified by supplying the appropriate XML.

When defining Containers to host services, the listener configuration of the container must match the protocol defined in the binding that will be associated with the service.

The "Configure > Registry > Bindings" section of the "Management Console" provides functionality that allows you to add bindings, view bindings/WSDLs, configure identifiers/categories, import bindings from WSDLs, view service references, and delete bindings.

Policies

The governing of a web service endpoint is driven through the definition of Governance Policies. Governance policies can be associated with different components of a service's definition. Workbench provides a true policy framework for governing services. The policy framework supports delegated administration by allowing policies to be defined within different organizations in the organizational tree.

Governance Policies supported by Policy Manager are organized into the following policy categories. Operational and Compliance policies are further grouped by "Type." The "Type" selection is performed when the policy is created using the "Add Policy" function.

Operational

Operational Policies are used to define a metrics of requirements for addressing the operational implications of services that are shared across enterprise departments. These policies address the operational model for services, capacity monitoring and planning, the handling of policy exceptions and violations, and service execution including the definition and enforcement of runtime policies such as security, access, logging procedures, and service reliability. The operational policy framework supports both a wide variety of out-of-the-box policy implementations as well as custom policy implementations. The framework also supports the grouping of policies for easier administration. Policies are defined in

such a way that they can be attached to different governable entities in the Workbench, such as organizations, services, endpoints, operations, and messages. The allowable attachment points are defined in policy metadata. The following policy categories are supported:

- Aggregate Policy - A collection policies that are gathered together to form a policy group. Policies included in an Aggregate policy are defined to achieve a specific purpose relative to governing Policy Manager objects (i.e., Organizations, Services, and so on). Aggregate Policies can be defined for "Operational" and "Compliance" policy types.
    - Authentication Policy -

Note: A subset of operational policies include "Sample" policies which are pre-configured policies that illustrate common use cases.

Compliance

Compliance Policies allow organizations to create standards that control the quality of the data in the repository. These policies can be used to analyze the service metamodel, WSDL documents, Schema, and transactional data to determine whether they meet corporate standards. There are four "Compliance Policy" types.

A Compliance Policy is added using the "Add Policy" function. After a Compliance Policy instance is added, it is configured with a compliance rule. A compliance rule is a test against the service WSDL, UDDI document message or other aspects of the service metadata model. A combination of Compliance Policies (each representing one compliance rule) can be included in a "Compliance Aggregate Policy" that can be used to analyze the service metamodel, WSDL documents, Schema, and transactional data to determine whether they meet corporate standards.

Multiple Compliance Policies (i.e., rules) can be attached to Policy Manager objects (i.e., organizations, services) using the Policy Attachment functionality and executed as part of the compliance check. Policies (and rules) can also be imported from an XML document using the Import Package function.

Creation and management of Compliance Policies requires assignment of the "System Administrator" role. Compliance Policies are accessible from the Policies -> Compliance tab.

The "Policies > Compliance Polices" section of the "Management Console" includes functionality that allows you to view, add, modify, delete, and manage references for "Compliance Policies."

The following Compliance Polices are supported:

- Aggregate Policy – A collection of policies that are gathered together to form a policy group. Policies included in an Aggregate policy are defined to achieve a specific purpose relative to governing Policy Manager objects (i.e., Organizations, Services, and so on). Aggregate Policies can be defined for "Operational" and "Compliance" policy types.
    - Script Policy - Allows you to define a rule that uses a combination of XQuery and Java to first select the content for analysis and then to perform the analysis.
    - WSI BP 1.1 Policy - Allows you to define a rule that executes an XQuery on the service or message model to determine compliance.

- XQuery Policy - The WS-I Basic Profile (official abbreviation is BP), is a specification from the Web Services Interoperability industry consortium (WS-I), and provides interoperability guidance for core Web Services specifications such as SOAP, WSDL, and UDDI. The profile uses Web Services Description Language (WSDL) to enable the description of services as sets of endpoints operating on messages. This policy type executes an XQuery on the service or message model to determine compliance.

QoS

QoS (Quality of Service) Policies are used to define a metrics of requirements for ensuring service availability, performance, integrity and reliability. Different consumers of the same service might require different Service-Level Agreements (SLAs), including performance, transactional support, and security requirements. The following QoS policy categories are supported:

- Service Level Policies

  A "Service Level Policy" is a Quality of Service (QoS) Policy that defines conditions for measuring and reporting performance of a specific Contract. Each policy is composed of a "Rule" and "Access Interval." Rules represent the conditions you define to measure and report performance of a service contract. When a defined system condition matches a defined rule, an alert is raised. A "Rule" is composed of "Rule Elements" (i.e., Alert Code, Metric, Operator, Value, Units, and Interval) that are configured to meet your service monitoring requirements. An "Access Interval" is composed of one or more "Access Days" (Sunday through Saturday). Each "Access Day" is configured with an "Access Time" that can represent one complete day (All Day) or a date range (Range) that is specified using a 24-hour clock format (HH:MM).

  - Quota Management Policies

  – Bandwidth Quota Policy - Allows you to configure the bandwidth cap (i.e., quota) that a consumer can upload or download at any given time.
  – Script Policy - Allows you to update a policy defined using BeanShell or Jython script languages.
  – Service Level Enforcement Policy - Allows you to enable and configure the error message returned to the consumer when their SLA is violated.
  – Throughput Quota Policy - Allows you to monitor web service throughput performance by specifying a throughput limit (i.e., quota), queue size, and configuring fault and alert notifications.
  – Timeout Policy - Allows you allows you to configure the timeout for each request and specify a custom fault error message that is returned to the client.
  – Concurrency Quota Policy - Allows you to monitor the web service concurrency performance by specifying a concurrency limit (i.e., quota) that represents the maximum number of concurrency connections, and configuring fault and alert notifications.

Contracts

The Workbench "Contracts View" is the starting point for defining, configuring, and managing "Consumed" and "Provided" service contracts. A contract enables the consumption (i.e., use) of a service by a "Provider" or "Consumer" Organization. It is used to monitor service performance levels and access to a service. Contracts are defined in the "Contracts Object" within an Organization.

Contracts are typically modeled to align with your organizations specific Service Level Agreement requirements. Service level agreements are implemented by defining Quality of Service (QoS) policies in Sub-Organization and Service Details sections and are then attached via the "QoS Policy Attachments" portlet in the "Contracts" section of the "Management Console." A "Contract Scope" is also configured that defines the services that the QoS Policies apply to. A contract can be defined at the Sub-Organization Level or within the Contracts folder of a Sub-Organization. When you create a new contract, an initial 1.0 version is assigned. Subsequent updates to the contract will increment this version number (i.e., 2.0, 3.0).

Contracts can be either "offered" or "requested." The "Offer Contract" action is initiated by the "Provider Organization" and offers a contract to potential "Consumer Organizations." The "Request Contract" action is initiated by the "Consumer Organization" and submits a contract usage request to the “Provider Organization” of a service.

Contracts revisions are managed by the Policy Manager "Workflow." When a new contract is defined, the Policy Manager default workflow (`default_contract_workflow.xml`) is involved and manages cycles of the contract. Each subsequent contract revision is tracked using this workflow and a history of transactions is recorded for each revision.

If a contract is written between a "Provider Organization" and "Consumer Organization" you can assign a set of Consumer Identities to authorize use of the contract by the Consumer Organization.

Another key component of a contract definition is called the "Contract Scope." The scope of a contract is defined by configuring the set of services and/or specific operations you would like to be associated with the contract.

Each contract can include one or more QoS Policies. A QoS Policy is composed of Rules and Access Intervals (i.e., Day of Week/Access Time). The Rules define the specific service monitoring criteria for the contract. This monitoring criteria is applied to the list of services/operations defined in the Contract Scope.

Contracts must be approved before they become active within Policy Manager. When a contract definition is complete it enters into an "Approval Workflow" process where a designated list of approvers who have special interest in the contract and its associated services review all the elements of a contract, and submit update recommendations. When a general consensus is agreed upon, the contract moves from a "Review" to "Approved" state and becomes active.

A contract is composed of the following elements:

- Details - A base definition that includes detail information (Name, Description), Duration (Effective and Expiration Date/Time), and Access Level.
  - Consumer Identities - Represents the list of identities (i.e., users) assigned to the Consumer Organization associated with the current contract.
  - Contract Scope - Represents the list of services/operations assigned to the current contract.
  - QoS Policy – Composed of "Rules" that include the conditions you define to measure and report performance of a specific Contract and one or more "Access Intervals" that include one or more "Access Days" (Sunday through Saturday).
  - Metadata - A metadata attachment provides additional technical and reference information pertaining to the current contract.

Provided Contracts

A service is provided when its business functionality is offered to clients via a contract. In Workbench, one organization can provide services to another organization. A service is identified as provided when the "Contract Scope" has an Organization and set of services and operations selected, and it is associated with an approved "Contract" that defines rules and access terms, or it is "Discovered" by Containers that are part of the current organization. The "Offer Contract" action is initiated by the "Provider Organization" and offers a contract to potential "Consumer Organizations."

For example, Company A has a contract with Company B for Company B's services (i.e., the "Contract Scope" includes Company B's products). Company A cannot change the Contract Provider from Company B to Company C (a different company) as long as their contract scope includes services and operations that do not belong to Company C.

Consumed Contracts

A service is consumed when its business functionality is used by clients in different applications or business processes (i.e., Organizations). In Workbench, one organization can consume the services of another organization. A service is identified as consumed when it is associated with an approved "Contract" that defines rules and access terms, or it is "Discovered" by Containers that are part of the current organization. The "Request Contract" action is initiated by the "Consumer Organization" and submits a contract usage request to the "Provider Organization" of a service.

Contract Scope

The "Contract Scope" defines the set of services and operations (i.e., scope) you would like to associate with the current contract. When you initially create your contract using the "Add Contract Wizard" (i.e., "Request Contract" or "Offer Contract") you set the contract scope organization when you select a "Provider Organization." The selected "Provider Organization" displays in the "Contract Scope Portlet" in a checked (i.e., locked) state and includes the list of services and operations that are available to be included in the contract scope.

Using the "Manage Contract Scope" function you then navigate the "Organization Tree" hierarchy and click the checkbox next to the services and operations within the selected "Provider Organization" that you would like monitored by the current contract definition. After you save your selections, you can view the current contract scope selections on the "Contract Scope Portlet." This view is locked and non-editable.

Note: To select a different "Provider Organization" for the current contract, you must unselect the "Provider Organization" and all selected services and operations using the "Manage Contract Scope" function.

Consumer Identities

The "Consumer Identities Portlet" is used to assign users (i.e., identities) to the "Consumer Organization" that has been offered a contract by the "Provider Organization." This type of contract is created by using the "Offer Contract" action in the "Provided Services Contracts" folder.

Consumer identities can be any user "type" defined within the "Management Console" including *Management Console User* and *Web Service End User* defined in the "Security > Users" section, or an *Organization Identity* defined on the "Details" page of a specific Organization.

After defining a contract using the "Offer Contract" action, the Consumer Provider is automatically assigned to the contract and displays in the "Consumer Identities Portlet." The "Organization Name" is locked (i.e., checkbox automatically selected) and an "Identities" folder displays below the Organization Name. You then use the "Manage Consumer Identities" function to select from a list of available identities currently assigned to the Consumer Organization, and you can also add additional Organizations to the "Consumer Identities Portlet."

Contract Metadata

The "Attachments" section of the "Metadata Portlet" is used to add *External Links* (i.e., URL references) and *Document Attachments* that provide additional technical and reference information pertaining to the current contract. A metadata attachment provides additional technical and reference information pertaining to the current contract. This could include external references or actual contract documents, service level agreements, or any other information source that supports the contract. You can "version" attachments using by uploading a new "External Link" or "Document Attachment." The version history can then be viewed for a specific attachment.

Chapter 3 | Capabilities

This section describes the capabilities the products deliver. It helps you understand what the products are designed to do, and lays the groundwork for later descriptions of how you will configure the products in order to deliver the capabilities.

Search

The "Workbench > Search" section of the Management Console provides tools for querying, managing, and securing existing web services. Service management can be applied to *physical*, *virtual*, and *discovered* service types.

Services currently defined in Policy Manager can be queried by performing a Services Search. Services Search supports key service objects associated service configuration elements (e.g., name, category, type), service access (e.g., user account or role), and service activity (e.g., usage, alerts, etc.), and provides a baseline of search criteria for targeting key information in each area.

When you find a service you can Manage PKI Keys, Change Organization, Request Contract, Offer Contract, Export Service, Virtualize Service, Check Compliance, or View WSDL.

Publishing

New services not defined in Policy Manager, or existing services not hosted by a Container (i.e., SOA Container, Legacy Container— Standalone or Embedded) are candidates for management. The initial process of bringing a service under management is performed using the "Create Physical Wizard."

The service management process involves creating a "manageable service" by configuring a set of properties, selecting an organization (i.e., business entity), and associating a "manageability endpoint" (i.e., Container) with the service to complete the deployment process. Based on your business requirements you can configure both activities to create a "managed service" that is deployed, or you can define the service properties and create a "manageable service" and select the Container host via when you are ready to deploy the service.

After completing the management process, you can view service details, modify the policy configuration for the service, configure bindings (i.e., HTTP, HTTPS, and JMS listeners), configure categories for the service, and define rules to manage XML Denial of Service attacks, and monitor service performance.

Workflow

Workflow is part of the overall Governance that an organization needs to have over their SOA. Workflow allows an organization to assign users to a role that provides oversight for the services that are submitted by the development community. Through the approval or rejection of service candidates and provisioning contracts, the organization can manage the quality of the services that are published in the repository. Workflow also provides an auditable record of the submission and approval of a service which is essential for corporate compliance to processes like ISO and CMM.

Services and Contract Workflows that are enabled and configured with "Workflow Definitions" in the "Configure > Workflow" section of Policy Manager, map directly to the "Workflow Tasks" Portlet in the Root or Sub-Organization, and to the Workflow Control Portlets in the Services and Contracts Details screens (i.e., Service Workflow and Contract Workflow).

Mediation

Mediation is the ability to create a virtual service that supports a different set of transports, bindings, policies, or even messages than the physical service(s) that underly it.

The most commonly used example is a RESTful virtual service on top of a SOAP physical service.

Monitoring

The "Monitoring" section of the Workbench "Services Object" provides functionality for managing alerts raised by the current Policy Manager "managed" service, generating and exporting usage data logs, viewing real-time performance metrics charts that provide a graphical presentation of statistical data for the service aggregate or specific operations, generating historical charts using captured usage data to for service and operation usage and response, and viewing and adding dependencies.

Alerts

The "Monitoring > Alerts" tab provides tools for viewing and managing alerts that are raised by the current "managed" service. Alerts are raised when a system condition matches an active Alert Code. Each alert is associated with functionality that occurs in Policy Manager subsystems. Alerts also include a Severity identifier, the Alert Code representing the Policy Manager functional area, and the Container that is managing the web service operations. This information provides administration personnel with key information to effectively review the alert condition and determine the appropriate method of response. An alert response is configured using state filters.

When a new alert is raised, the state filter is set to "Unobserved." During the decision-support process, an alert state is typically changed to "Observed" while the problem is being investigated. When an alert resolution is found, comments are then added to the alert record and the state filter is changed to "Resolved." Alert data can also be exported to an XML file to facilitate additional analysis or reporting requirements. You can view, filter, add comments to, observe, resolve, print, and export alerts.

Logs

The "Logs" section provides options for filtering, viewing, and exporting usage data and associated SOAP Messages for web service operations. This data can be used in the troubleshooting and decision support process for a Policy Manager deployment. The reporting process involves targeting specific Usage Data by defining search criteria filters that are applied against existing logs.

Reporting functionality is user-defined by configuring report-focused policy functionality including configuring Usage Monitoring for the policy configuration, and adding a Record Component to the Request or Response configuration of a Policy for designated web service operations. During a transaction cycle the key information associated with the transaction (i.e., Usage Data or SOAP Messages) is recorded using the Usage Monitoring and Record Component configuration parameters. This data is saved in a database file which can then be queried, viewed, and exported to an XML file using the filter and reporting options in "Services > Monitoring > Logs."

Real-Time Charts

Performance Metrics Charts provide real-time monitoring functionality for tracking service response time, adherence to defined SLA rules, and monitoring of transaction errors. Each service that is "managed" displays a Performance Metrics Chart in the "Monitoring Portlet." If the service is actively receiving requests, real-time performance data displays in the chart. If the service is not receiving requests the chart displays but does not display performance data.

Historical Charts

The "Historical Charts" section provides a series of reporting options for analyzing the usage and response of web services and their operations. Report data can be viewed in a visual Gantt-style chart view, or data line item summary view. Report views can be toggled back and forth depending on presentation requirements. Filtering options are provided for each report option for targeting a specific service or operation, and defining the time increment and date range for gathering report data.

For Pipeline Policies, report data is generated by configuring service operations with a pipeline policy configured with "Usage Monitoring" enabled and/or configured with the "Usage Monitoring Evaluation Point Component" added to the Request, Response, and/or Fault Configuration. The WS-Auditing Policies also provide monitoring options. The following historical charts are supported:

- Service Response Chart - Displays the average response time of a web service including "Time" when the web service was accessed, and "Response Time (in Milliseconds)."
- Service Usage Chart - Displays the number of times a web service was accessed. Each time an Operation is accessed counts as an individual usage.
- Operation Response Chart - Displays the average response time of the web service operations.
- Operation Usage Chart - Displays the number of times a given web service operation was accessed.
- All Operations Usage Chart - Provides an encapsulated view of usage statistics for web operations that are configured with the "Usage" Policy Component.

Dependencies

Root Cause Analysis is a form of diagnostic management that is used for targeting system faults. A "root-cause" typically represents a core set of circumstances (i.e., cause) and their associated relationships (effect) that create a consequence (i.e., system downtime, poor performance, etc.). In web service management the ability to perform root-cause analysis on web services and their dependencies and rapidly restore system performance ensures compliance with defined service-level agreements.

Policy Manager provides a services mapping tool called the "Dependency Map Portlet." This service mapping tool uses a graphical interface to present the hierarchy of the elements that comprise a "managed" web service. These elements can a include web service application (i.e., physical service), one or more virtual services, and Containers. The service hierarchy includes drill-down functionality that allows you to navigate the upstream and downstream dependencies of a service. Each service includes a Performance Metrics Popup that provides a summary of system performance, and a Service Details Popup that displays the "state" of the service, and also includes a set of Actions that pertain to managing the service configuration.

While navigating the service hierarchy you can view key data pertaining to service performance including performance metrics data and alert status. Based on your findings, you can troubleshoot and adjust the service configuration.

Security

The Policy Manager "Workbench" provides object based security functionality that implements a new access control model that enable an organization to define its own access control policy, using the Organization Tree hierarchy to scope and propagate the security policies, supports role-based access control (RBAC) where the Role serves as the fundamental mechanism to define and manage information access, and supports expressive role definition.

Roles are initially defined at the Root Organization level and represent baseline definitions (i.e., templates) that can be modified. Role Definitions are presented in the Role Memberships Portlet that is available at the Root Organization level and in the Security section of each Organization. Within each Organization, the specific users you would like to assign to each Role Membership is configuring using the Manage Role function.

Auditing

The "Auditing" section of the Management Console provides auditing tools for measuring the performance of a Policy Manager deployment. You can monitor system activity based on alert notifications or security policy-related actions, users, and date and time range. This system activity is saved in an "Audit Trail" which is a log of add, modify, and delete operations performed on all objects in the system. Selected "Audit Trail" records can be exported to an XML file and used for reporting purposes. This data can be used to measure system performance and manage the security of web services. Two types of audit trail functionality are offered:

- Alert Audit Trails

  Provides ability to filter, view, and export data that is logged when an alert is "raised" during the operation of your Policy Manager deployment.

  In order to activate the alert audit feature, the "Log Alert" checkbox must be checked in the Alert Code definition. This activity is performed in the "Alerts" section of the Policy Manager.

- Security Audit Trails

  Provides the ability to filter, view and export data that is logged when a security policy related action occurs during the operation of your Policy Manager deployment.

The availability of security audit trail data is based on a combination of standard system actions that are automatically logged and whether you configured specific security operations to audit activity (e.g., security policy component). This is accomplished by enabling the "Log Alert" in an Alert Code (for Alert Audit Trails), and enabling "Generate Audit Data" in the Authentication, Authorization, Signature, and/or Signature Verification sections of the Security Policy Component in a Policy definition (for Security Audit Trails).

Categorization/Classification

The "Configure > Registry > Category Schemes" section of the "Management Console" provides the ability to define your own custom category schemes that align with your specific business requirements. The Policy Manager default installation includes a set of default identifier schemes that include industry specific and custom category schemes. Configuring category schemes is a prerequisite to defining a category hierarchy. This is because category schemes are used to build the category hierarchy. The following key activities can be performed:

Categories Schemes (*also referred to as Category tModels*) are a set of classification codes that represent different aspects of a web service (e.g., products, services, technical specifications). Within UDDI, a categorization tModel is used for structuring category content. This category structure is referred to as an "information taxonomy." Each category scheme contains Key Names and Key Values for each category item. Key Names are typically defined as a physical description of the category item, and Key Values are internal reference numbers for that category item. The use of categorization tModels enhances the search process by providing a broader scope of choices for targeting information.

The "Registry" provides a Category Hierarchy that includes WSDL Entity Type tModels, UDDI Category tModels, Policy Manager Category tModels, and a series of industry standard category schemes referred to as Business Taxonomy tModels. You can navigate the tiers of the Category Hierarchy by clicking the main hyperlink of a specific category scheme, and continuing with this same approach to perform additional drill-downs.

When you register a service with Policy Manager using the "Create Physical Service Wizard" a default set of category schemes are added to the service and access points categories section. Category tModels added to the Registry > Category Schemes section include tModels that comprise a wsdl:service. Category tModels added to the "Workbench > Services > Access Points section include tModels that comprise a wsdl:port. This section can also include tModels associated with the wsdl:binding extensions.

The following category schemes types comprise the category hierarchy:

- WSDL Entity Type tModels
- UDDI Category tModels
- Policy Manager Category tModels
- Business Taxonomy tModels

Chapter 4 | Managing Users and Security

This section describes the way you manage users and user security. It provides deeper functionality descriptions and answers several basic how-to questions.

User Roles and Responsibilities

Workbench Security

The Policy Manager "Workbench" provides object based security functionality that implements a new access control model that enable an organization to define its own access control policy, using the Organization Tree hierarchy to scope and propagate the security policies, supports role-based access control (RBAC) where the Role serves as the fundamental mechanism to define and manage information access, and supports expressive role definition.

Roles are initially defined at the Root Organization level and represent baseline definitions (i.e., templates) that can be modified. Role Definitions are presented in the Role Memberships Portlet that is available at the Root Organization level and in the Security section of each Organization. Within each Organization, the specific users you would like to assign to each Role Membership are configured using the Manage Role function.

Identity Categories

This section provides a description of the different identities that Policy Manager provides.

Consumer Identities

Consumer Identities (or Organization Identity or Application Users) are assigned to the Organization. Therefore, Organization Administrators can fully manage these identities (create, update, view, delete, associate PKI keys). Consumer Identity is used to model any application that is consuming web services, but it itself is not a web service. If the consumer is a web service, user can model the service using Service artifact. To model all consumers that are not services, we use Organization Identities (or consumer applications or application users). Therefore, these and web services represent consumer applications, these are visible along with services in the Contract's Consumer identities portlet for selecting the consumers of the contract. These identities are assigned only contracts. One consumer application identity belongs to one organization only like the service belongs to one organization. Consumer applications can be local domain users or external domain users (like LDAP users)

End User Identities

End User Identities can be called consumers of consumer applications. End Users are users of the consumer application. In a way, these are consumers of consumer applications. For example, if portal application is a consumer application, users of the portal application become End Users for us. End user authorization is done in the Container using Authorization component. End Users authorization rules are defined in the Service Authorization Rules tab of the Organization. End users don't belong to organizations. Same end user can access the services of different organizations and may not even belong

to any of the organizations in Policy Manager. For the most part, these users come from other user repositories like LDAP. If you want to use local domain users, these users have to be created in the System ' Users tab, which means it is a System administrator function. When using Add User in the System users tab, select the Web Service End user option for creating these users. Though users are created by system administrators, every organization administrator has to allow the user to access the services of their organization by creating the service authorization rules. End Users can be Local Domain users or external domain users (like LDAP).

Policy Manager Users

Policy Manager Users (Console Users or Users accessing SM APIs) are users using SOA infrastructure. In general, we don't treat users as someone who belong to the organization. These Users are created in System ' Users tab. When using Add User wizard in System ' Users tab, select Management Console user option for creating these users. Though, user is created in System ' Users tab by System Administrators, permissions to these users are controlled by Organization Security Administrators in Organization ' Security tab. These users can be either Local Domain users or external domain users.

Console users are users using SOA infrastructure. In general, we don't treat users as someone who belong to the organization. These Users are created in System ' Users tab. When using Add User wizard in System ' Users tab, select Management Console user option for creating these users. Though, user is created in System ' Users tab by System Administrators, permissions to these users are controlled by Organization Security Administrators in Organization ' Security tab. These users can be either Local Domain users or external domain users.

Service Identities

Service Identities (Default service identity in the local domain to be able to associate the PKI Keys or Cert for the service) Every service has one identity by default. We use Service Identity to associate the PKI keys to the Service. Username for service identity is derived from Service key with a function that brings the username to size that fits in Username field. We don't display these user identities anywhere in the configuration screens exception on the Service details page to indicate the default username of the service. PKI Keys of the Service are managed from Service Details page using Manage PKI Keys option by anyone that have Service/Modify permission. Default Service identity is in Local Domain.

Container Identity

Container identities are used to store the identity of the container and also to store the inbound/outbound HTTPS certificate with private key. Every container has one identity by default. We use container identity to associate the container seed, PKI keys to the Container. Also, this identity is used to associate the inbound SSL and outbound SSL certificate and private key to Containers. This user identity is not displayed anywhere. To manage the SSL certificate/Private keys for container, Actions are available in the container details page. Container identity is a Local Domain. Container Identity permissions are managed using Container/Modify permission.

Access Control Policies

Access Control Model

Policy Manager "Workbench" implements a security model that supports access control policies and role-based access for "Organization Objects." This security model is referred to as "Workbench

Security." Users and User Groups that "Workbench Security" access policies are assigned to are defined in "Policy Manager Security."

Each organization object has a unique set of attributes that include its relevant security values (e.g., public or private). Supported categories of access control are organized by "Action." Each Action represents a specific type of access control that can be applied to a Workbench Object. Workbench Objects are organized into four Object Types (Organization, Service, Contract, Container, and Policy).

Access control is applied to a Workbench Object by associating it with an Action. This is association is called a "Privilege." When a privilege is attached to an Object Type it is referred to as a "Scoped Privilege." Privileges can allow or deny access and can be constructed into privilege sets referred to as "Permissions." If a user is assigned privileges within an organization, these privileges will be scoped to its sub-organizations. This is referred to as a "Containment Relationship."

Privileges are configured within a "Role Definition." Role Definitions are defined and maintained using the "Role Administration Wizard." Each Role Definition represents an access control template for your defined Security Model. Within each Role Definition you define the Privileges and Permissions for each Object Type. These Role Definitions are then available within sub-organizations as a "Role Membership." User and User Group assignments are then assigned at each organization level.

The following sections provide additional detail about each access control model element.

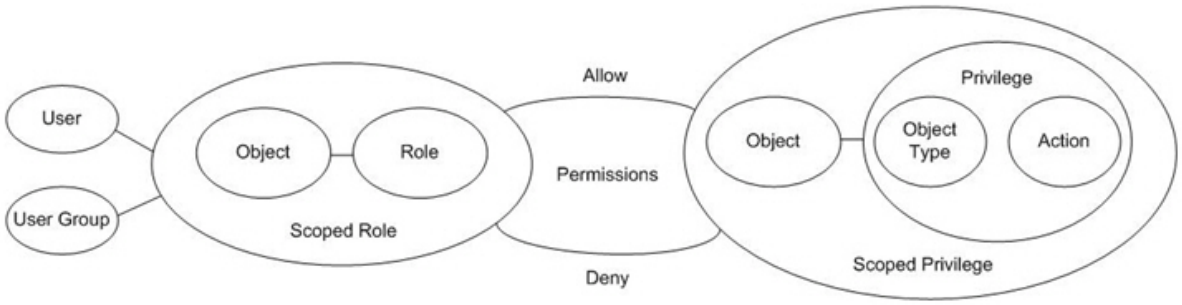


Figure 12: Workbench Access Control Model

Access Control Roles

Roles are initially defined at the Root Organization level and represent baseline definitions (i.e., templates) that can be modified. Role Definitions are presented in the Role Memberships Portlet that is available at the Root Organization level and in the Security section of each Organization. Within each Organization, the specific users you would like to assign to each Role Membership is configuring using the Manage Role function.

The following list provides an overview of access control roles supported by the Policy Manager "Workbench."

- Role

  Role is a set of privileges. The privileges can be either positive or negative. The positive privilege grants the user to perform an action on a resource, while negative privilege denies user to perform an action on a resource.

- Scoped Role

Scoped role is role attached to a resource. All the privileges specified in the role will be scoped to the resource. The scoped privileges contained in the scoped role are classified into two sets: "allowing" privilege set contains only the positive privileges; while "denying" privilege set contains only the negative privileges.

- Users / User Groups – Scope Role Assignment

Users and User Groups can be assigned to scoped roles. This is a many-to-many relationship. Via User/Group – Scoped Role and Scoped Role – Scoped Privilege relationship, we can find all the scoped privileges assigned to any user.

- Organization-level Access Control Policy

The organization-level access control policy is specified by way of defining roles. The organization that defines a role is called the defining scope of the role. A role definition is visible to all the sub-organizations of its defining scope.

- Content of Role Definition

The content of role definition contains a list of entries. Each entry is a triplet <resource predicate, action, allow/deny>, which represents either a positive privilege or a negative privilege. Resource predicate is a Boolean expression with regard to resource types and attributes, it specifies the set of resources that the privilege applies to.

Access Control Privileges

The association between a "Workbench Object" and "Action" is called a Privilege. An "Action" defines a specific type of activity to be performed in "Workbench" (e.g., Add, Read, Monitor, Modify, Delete, Assign Policy, Full Control, etc.).

When defining a Role Definition (i.e., access control policy) at the Root Organization level, you select a specific Workbench "Object Type" and configure the "Actions" that you want to apply to that "Object Type." This Role Definition will then be available as a "Role Membership" within an Organization. Privileges assigned to Roles within a Role Membership are referred to as "Scoped Privileges." This means when a User is assigned to a specific Role within a Role Membership list, the user will automatically have the same privileges scoped to its sub-organizations. This is referred to as a "Containment Relationship."

Privileges support the following "Object Types" and "Actions" actions:

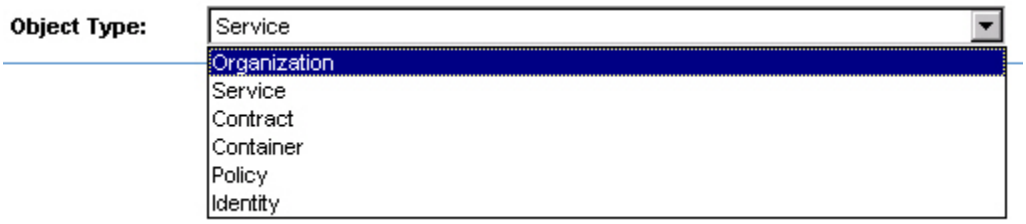
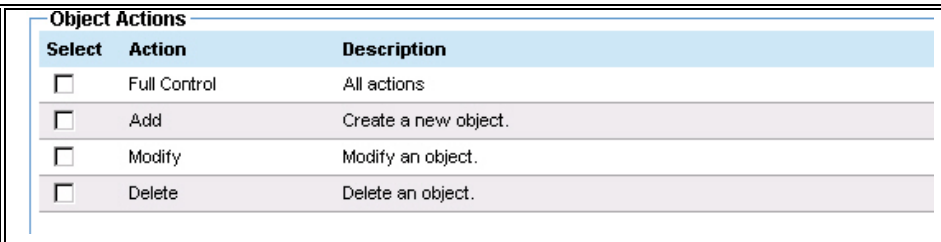
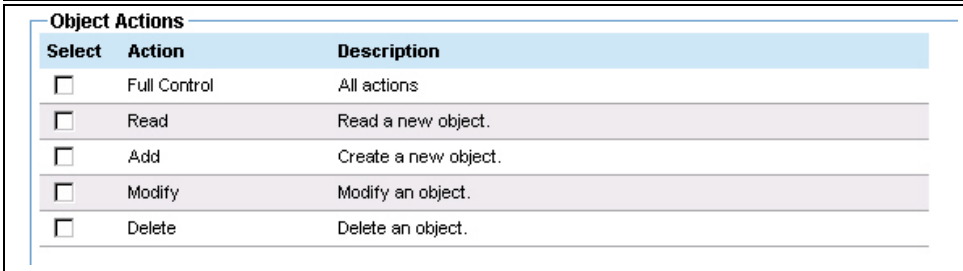


Figure 13: Object Type

Object Type	Supported Privileges

Organization	**Object Actions** **Select** / **Action** / **Description** ☐ Full Control — All actions ☐ Add — Create a new object. ☐ Read — Read an object. ☐ Monitor — Read the monitoring data of services. ☐ Modify — Modify an object. ☐ Delete — Delete an object. ☐ Assign Policy — Assign policy objects to an organization or service.
Service	**Object Actions** **Select** / **Action** / **Description** ☐ Full Control — All actions ☐ Add — Create a new object. ☐ Read — Read an object. ☐ Read as Provider — Read Service with Provider view. ☐ Monitor — Read the monitoring data of services. ☐ Modify — Modify an object. ☐ Delete — Delete an object. ☐ Approve — Approve publishing service objects. ☐ Assign Policy — Assign policy objects to an organization or service.
Contract	**Object Actions** **Select** / **Action** / **Description** ☐ Full Control — All actions ☐ Add — Create a new object. ☐ Read — Read an object. ☐ Modify — Modify an object. ☐ Delete — Delete an object. ☐ Approve Contract — Approve Contract
Container	**Object Actions** **Select** / **Action** / **Description** ☐ Full Control — All actions ☐ Add — Create a new object. ☐ Read — Read an object. ☐ Monitor — Read the monitoring data of services. ☐ Modify — Modify an object. ☐ Delete — Delete an object. ☐ Assign Policy — Assign policy objects to an organization or service. ☐ Host Service — Allows user to Host/Unhost sevice in the container.

Policy	
Identity	

Administrator Role and Privilege

This section provides a summary of the access control model used for "Administrator" roles.

- Access Enforcement and Conflict Resolution Rules:

  The following rules will be used to decide whether a user U has the permission to perform an action P on resource O.

  - If U's allow privilege set does not have privilege [O, P] and any of its containing privilege, then U is not allowed to perform operation P on object O.
  - If U's allow privilege set has the privilege [O, P] or any of the privileges containing [O, P], and U's deny privilege set does not have [O, P] and any of its containing privilege, then U is allowed to perform operation P on object O.
  - When U's allow and deny privilege sets both have the privilege [O, P] or any of its containing privilege, then the access is denied.

  The basic access control model discussed in last section applies to regular objects only. Administrator role is required to create and modify the relationship objects in the access control model. The administrator role contains a special privilege called "security admin" that give it the permission to do the following task:

  - Perform User / Scoped Role assignment
  - Perform Group / Scoped Role assignment
  - Perform User / Scoped Administrator Role assignment for sub-organizations
  - Perform Group / Scoped Administrator Role assignment for sub-organizations
  - Create / Modify/ Delete Role definition (i.e., Perform the Scoped Role – Scoped Privilege assignment)
- Separation of Duty (SoD) Constraint:

  To prevent any potential admin privilege abuse, the following Separation of Duty constraint can be enforced by the system if it is turned on by user:

- The administrator role only has administrator privilege in its allowing privilege set, and cannot have any other privileges.
- The security admin privilege cannot be assigned to any role other than administrator role.
- No user can be assigned to both administrator role and non-administrator roles.

The first two rules ensure that administrator role and non-administrator role cannot have overlapping privileges. The third rule defines administrator and non-administrator roles as conflicting with each other, and user is prohibited to have conflicting roles.

- Access Enforcement Rule for Administration Tasks:

  If a user wants to perform administrator task on an organization O, the following steps will be taken for access enforcement:

  - If the user's allowing privilege set has <O, "administrator"> or any privilege containing it, the action is be permitted.
  - Otherwise, the action is denied.

  Because of the SoD constraint, there are no conflicts when enforcing access to administration resources.

Default Role Definitions

What are the Role Definitions?

A "Role Definition" includes "Object Security Privileges" for "Workbench" Organization Objects (i.e., Organization, Service, Contract, and Containers). Each "Role Definition" is essentially a template that represents the baseline roles that can be utilized throughout the Policy Manager "Workbench" as a "Role Membership."

The Policy Manager default installation includes a set of ten default roles that represent common tasks associated with an SOA Infrastructure. Six of these "Role Definitions" are editable and the privileges and object type can be updated based on your requirements. Four of these "Role Definitions" are non-editable and reserved for system use. You can also add new "Role Definitions." See Default Role Definitions for a complete list of Role Definitions that are part of the Policy Manager default installation.

Each "Role Definition" is replicated to a "Role Memberships Portlet" that displays on the "Details" page of each "Sub-Organization." The "Manage Role" function within the "Role Membership Portlet" is used to assign Users/User Groups to each Role. Assigned User/User Groups have access to the functional areas defined within the "Role Definition." You can assign a base set of User/User Group assignments at the Root Organization level and then customize the assignments at different Organization tiers.

How do I View a Summary of current Roles?

The "Security Summary" screen is the starting point for defining and managing the "Role Definitions" and "Role Memberships" that represent the Object Based Security that will control access to Policy Manager functionality that is available within the Management Console

The following key activities can be performed:

To view the security summary:

1 Enter the following navigation path: **Workbench > Browse**. The "Root Organization Summary" screen displays. Click the "Security" tab. The "Security Summary" screen displays.

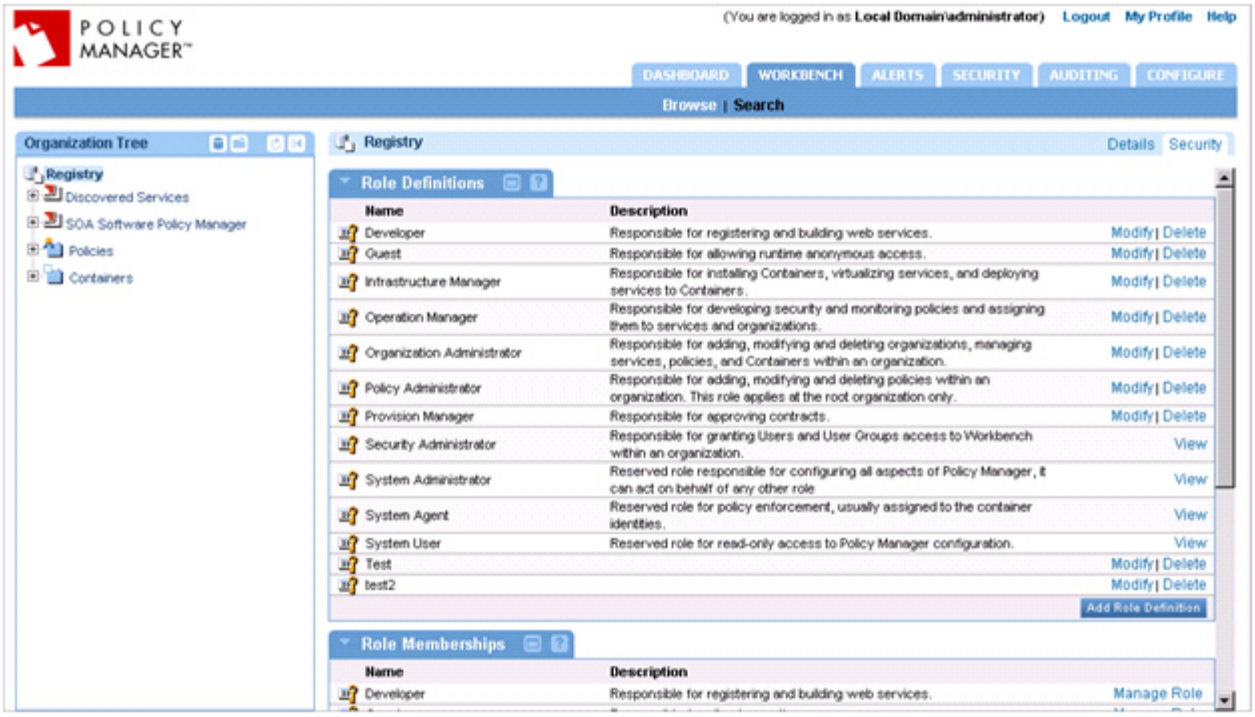


Figure 14: Security Summary (Root Organization Level)

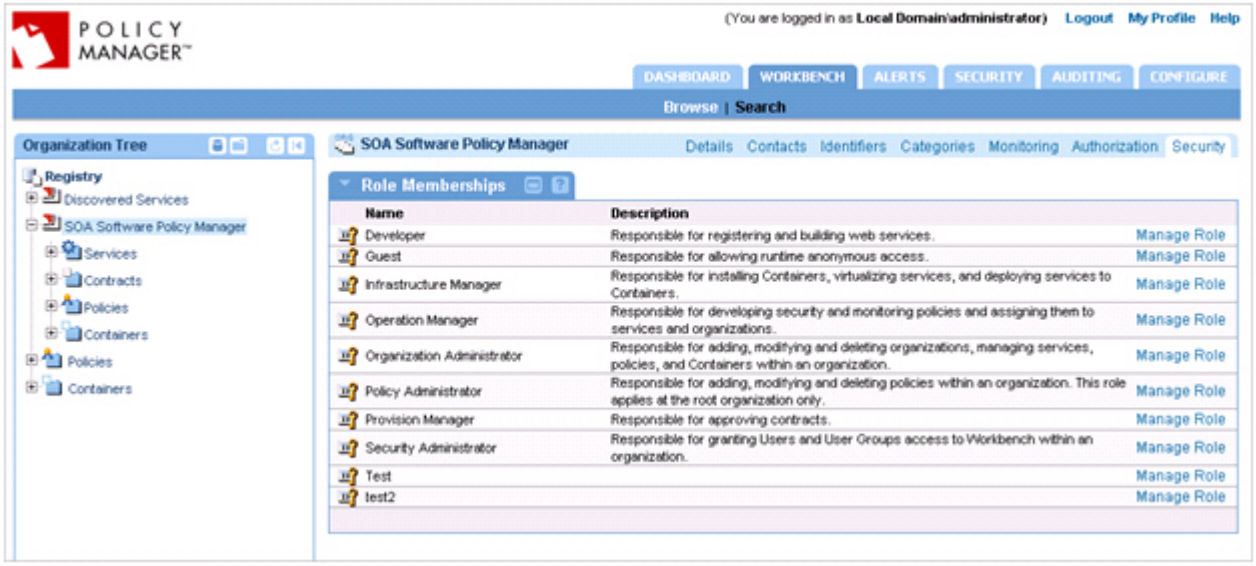


Figure 15: Security Summary (Sub-Organization Level)

Role Definitions Portlet

The "Role Definitions Portlet" is the starting point for defining and managing baseline "Role Definitions" for your Policy Manager deployment. This screen is located at the "Root Organization" on the "Security Summary" screen that is accessible via the "Security Tab."

The "Role Definitions Portlet" provides a listing of all the currently defined "Role Definitions." The screen presentation includes "Name" and "Description" of each Role. Roles that are modifiable include "Modify" and "Delete" hyperlinks. Roles that are non-editable include "View" links.

From the "Role Definitions Portlet," the following key activities can be performed:

- Add Role Definition
- Modify Role Definition
- View Role Definition
- Delete Role Definition

To view role definitions portlet:

1 Enter the following navigation path: **Workbench > Root Organization > Security**. The "Security Summary" screen displays and presents the "Role Definitions" portlet.

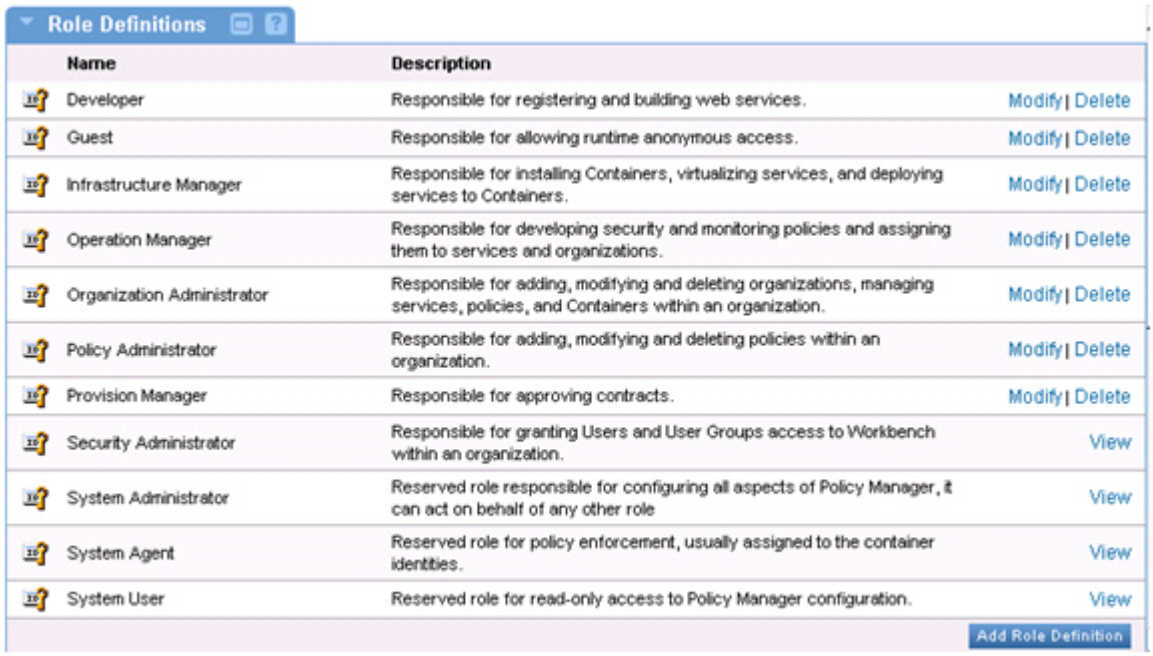


Figure 16: Role Definition Portlet

How do I Add a Role Definition?

The "Role Administration Wizard" is used to define "Role Definitions" that include "Object Security Privileges" for "Workbench" Organization Objects (i.e., Organization, Service, Contract, and Container). Role definitions are defined at the "Root Organization" level and represent the master set of Roles available to "Sub-Organizations" in the "Organization Tree."

Each "Role Definition" is available at the "Sub-Organization" level as a "Role Membership." You can manage access control for a "Sub-Organization" by assigning relevant Users and User Groups to each "Role Membership."

Add Role Definition

- Launch Role Administration Wizard
- Add Role Definition
- Add Privilege
- Select Object Actions

To launch role administration wizard:

1 Enter the following navigation path: **Workbench > Root Organization > Security**. The "Security Summary" screen displays.

To add role definition:

2 In the "Role Definitions Portlet" click "Add Role Definition." The "Role Administration Wizard" launches and displays the "Add Role Definition" screen.

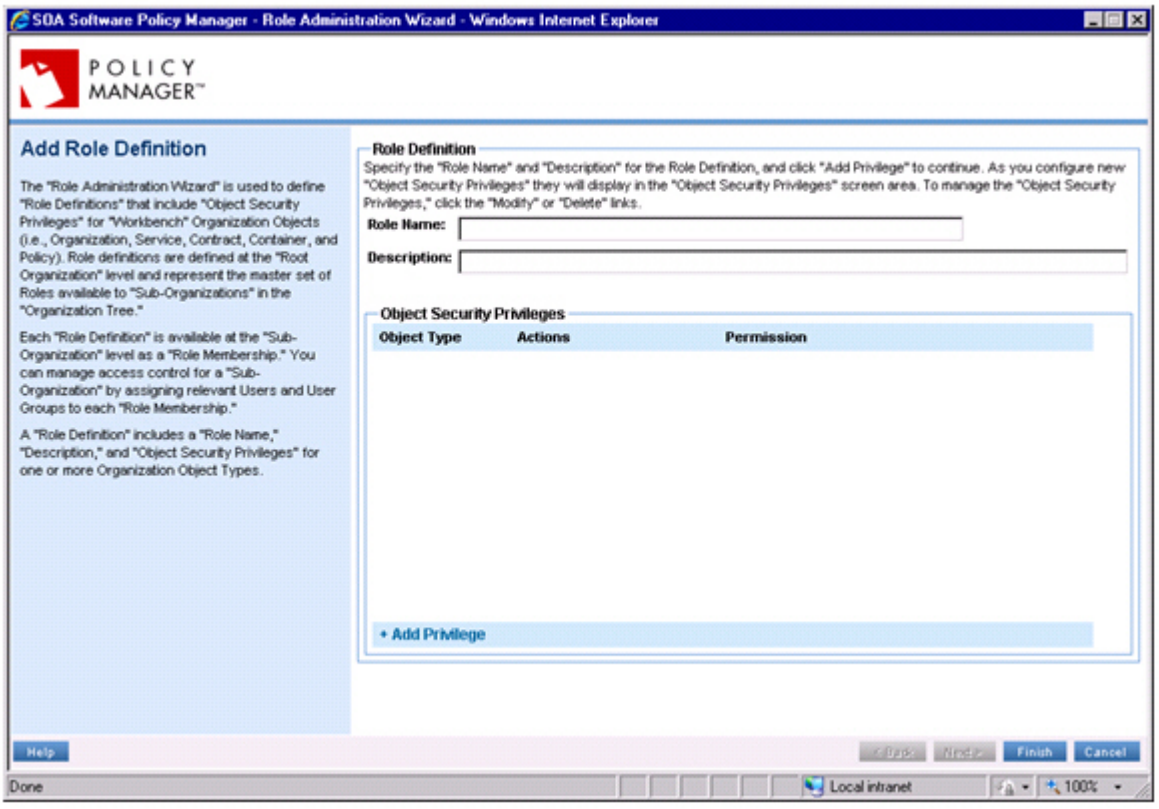


Figure 17: Role Administration Wizard—*Add Role Definition*

The screen is organized as follows:

Role Definition

- Role Name—A text field that allows you to enter the name of the "Role Definition."
- Description—A text field that allows you to enter a description for the "Role Definition."

Object Security Privileges

- The initial view of this screen will be blank. To add privileges, click the "Add Privilege" link. After adding one or more privileges, this screen area displays a breakdown of each privilege definition including Object Type, Permissions, and Actions.

Enter the Role Definition information. Then, to add a privilege, click "Add Privilege."

To add a privilege:

1 The "Select Object Type" screen displays. This screen is used to select the "Organization Object" type that you will be defining Privileges for. Available options include "Organization," "Service," "Contract," "Container," and "Policy."

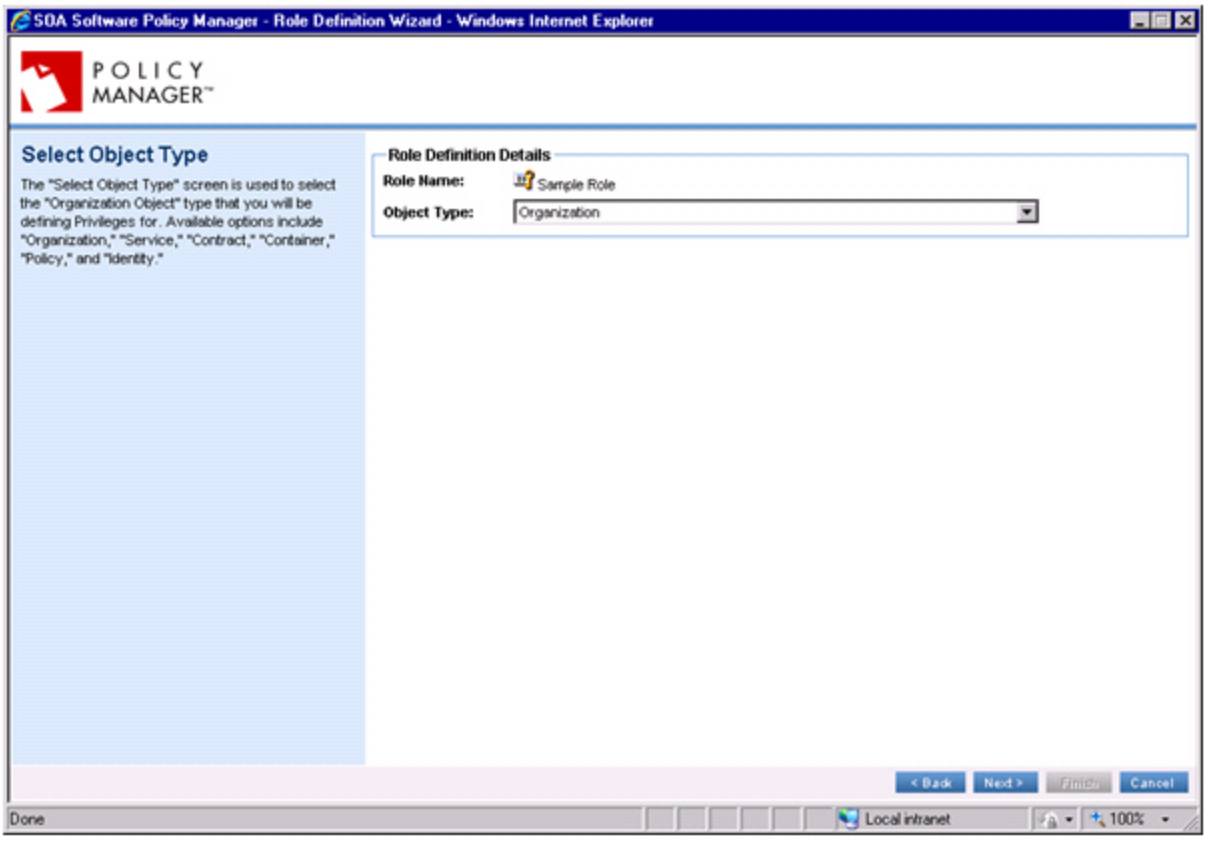


Figure 18: Role Administration Wizard—*Select Object Type*

From the "Object Type" drop-down list box, select the Object Type you would like to assign to the current Role Definition. Click "Next" to continue.

To select object actions:

2 The "Select Object Actions" screen displays. This screen is used to configure access control "Actions" for the current "Role Definition" and the "Permission" (i.e., Allow or Deny) that will be

applied to the selected "Actions." This "Action" and "Permission" combination represents a "Privilege" that is part of the "Role Definition." You can have one or more "Privileges" in a "Role Definition."

You can select specific "Actions" by clicking the checkbox next to each action, or you can select all "Actions" by clicking the "Full Control" checkbox. Selecting this option automatically checks all "Actions" in the list. You can select a "Permission" to apply to the selected "Actions" by clicking the "Allow selected actions" or "Deny selected actions" radio button.

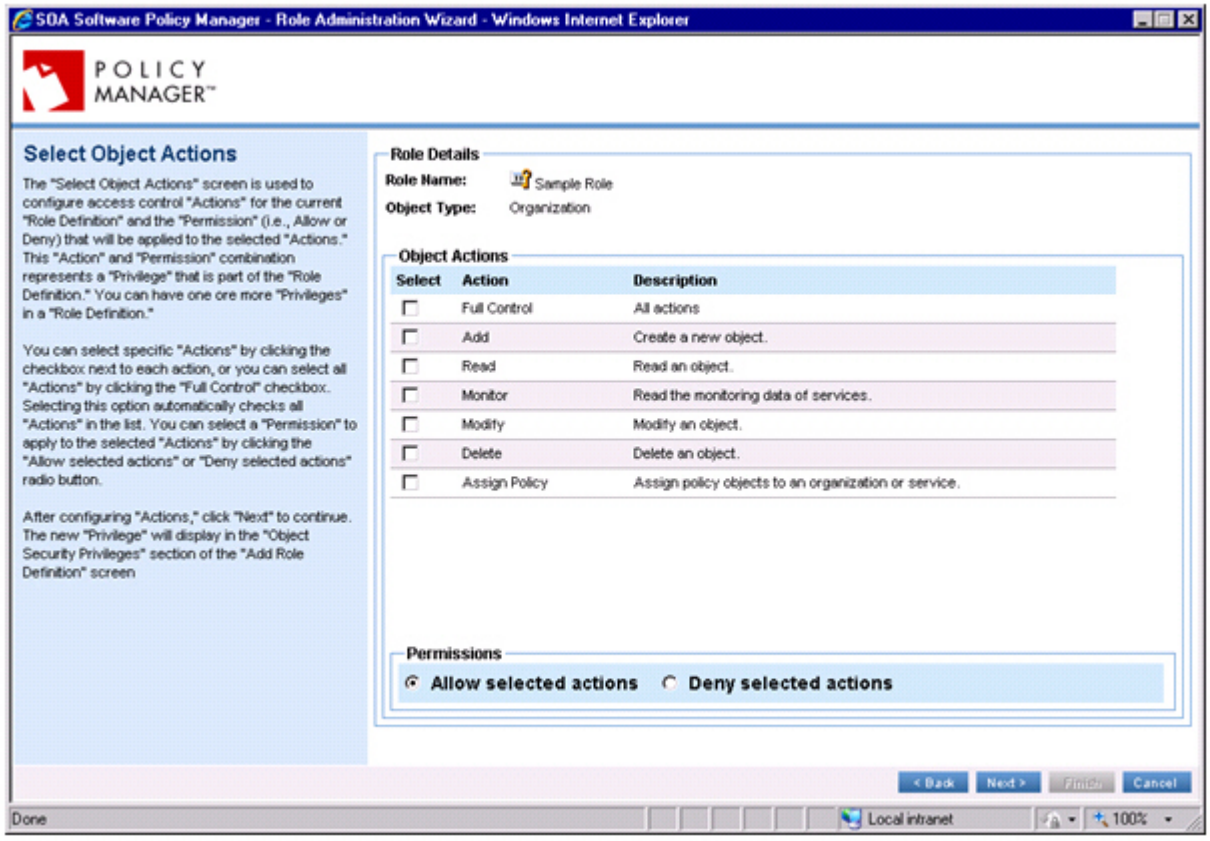


Figure 19: Role Administration Wizard—*Select Object Actions*

After configuring "Actions," click "Next" to continue. The new "Privilege" will display in the "Object Security Privileges" section of the "Add Role Definition" screen.

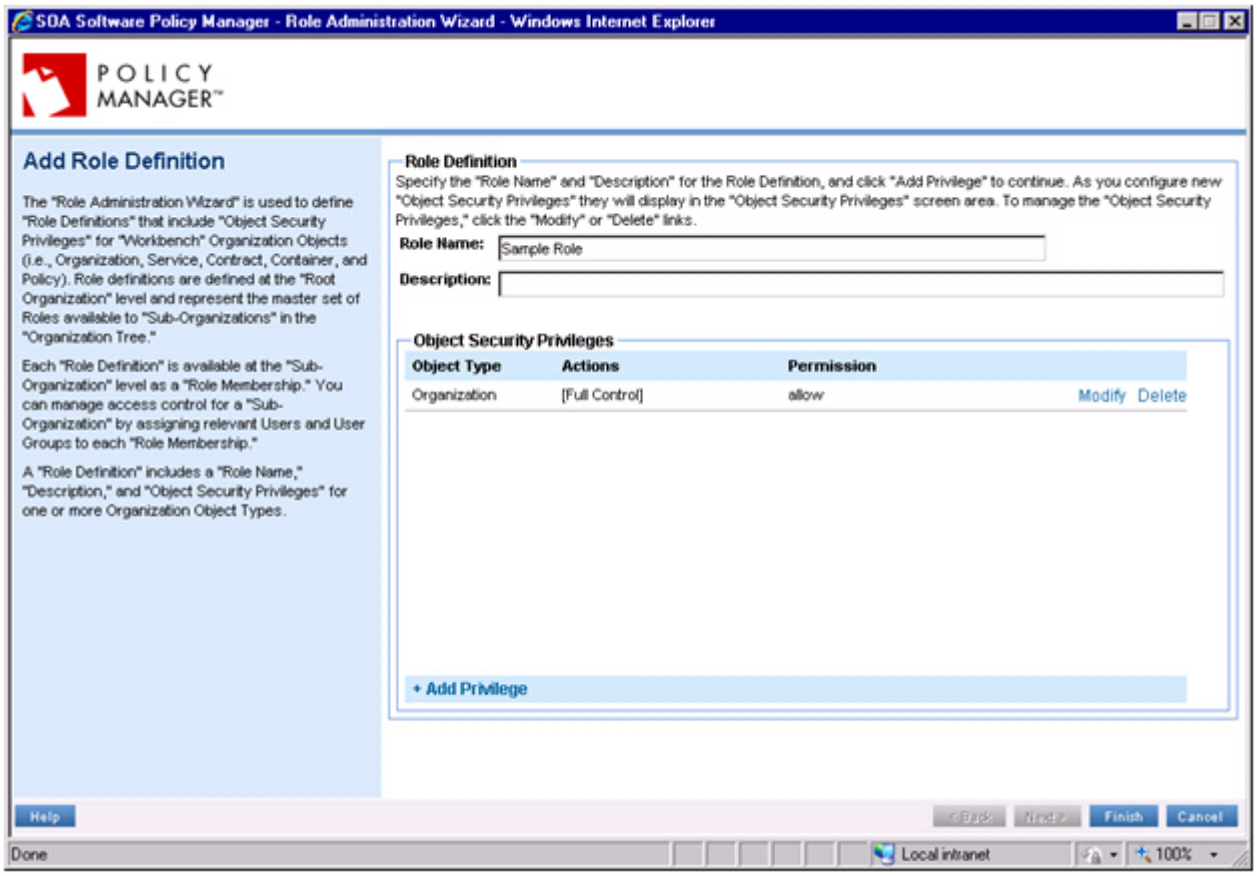


Figure 20: Role Administration Wizard—Select Object Actions (with privilege added to "Object Security Privileges) section

How do I Modify a Role Definition?

The "Modify Role Definition" screen allows you to update "Details" and "Privileges" of your "Role Definition."

Modify Role Definition

- Launch Role Administration Wizard
- Modify Role Definition
- Add/Update Privilege
- Select Object Actions

To launch role administration wizard:

1 Enter the following navigation path: **Workbench > Root Organization > Security**. The ""Security Summary" screen at the Root Organization level displays.

To modify role definition:

2 In the "Role Definitions Portlet" click the "Modify" hyperlink. The "Role Administration Wizard" launches and displays the "Add Role Definition" screen.

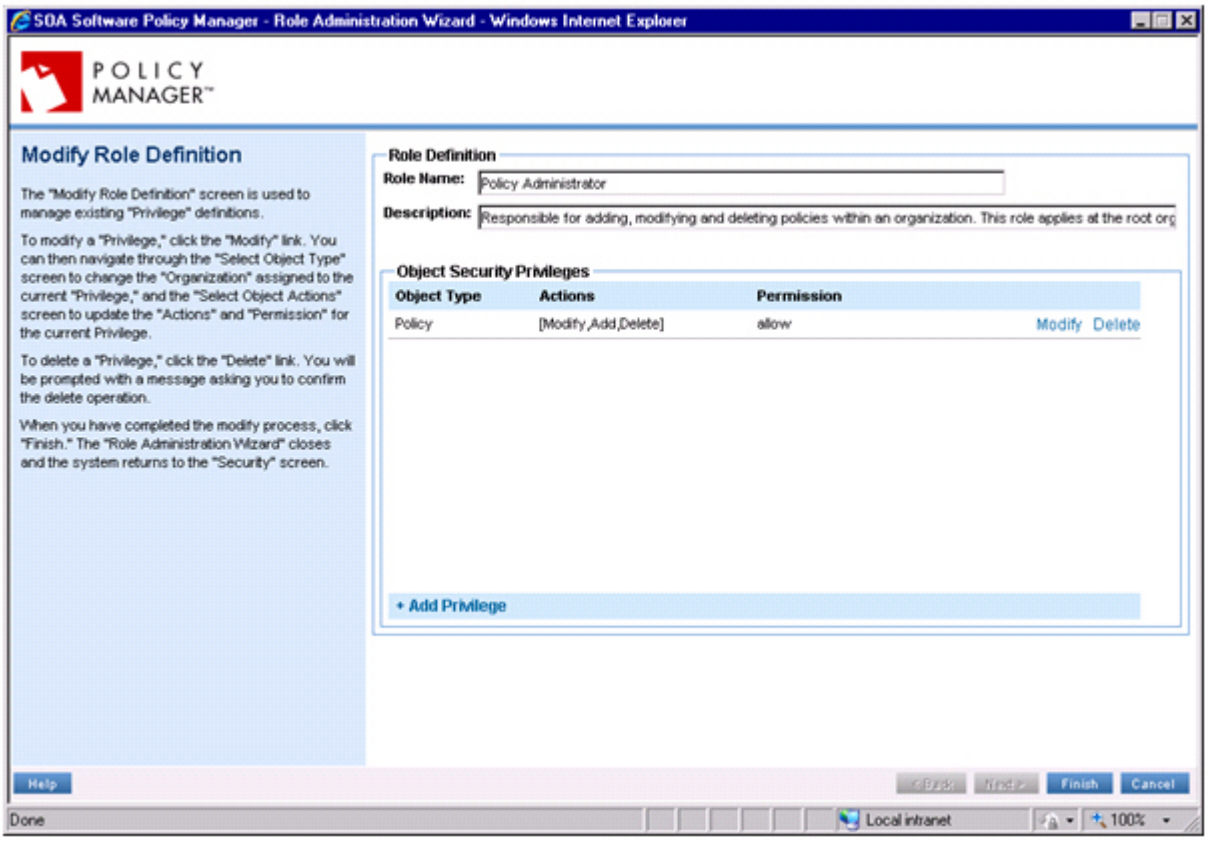


Figure 21: Role Administration Wizard—*Modify Role Definition*

The screen is organized as follows:

Role Definition

- Role Name—A text field that allows you to enter the name of the "Role Definition."
- Description—A text field that allows you to enter a description for the "Role Definition."

Object Security Privileges

- The initial view of this screen will be blank. To add privileges, click the "Add Privilege" link. After adding one or more privileges, this screen area displays a breakdown of each privilege definition including Object Type, Permissions, and Actions.

Update the Role Definition information. To update an existing privilege click the Modify link next to the privilege configuration. To add a new privilege to the current "Role Definition" click "Add Privilege."

To add/update a privilege:

1 The "Select Object Type" screen displays. This screen is used to select the "Object" type that you will be defining Privileges for. Available options include "Organization," "Service," "Contract," "Container," and "Policy."

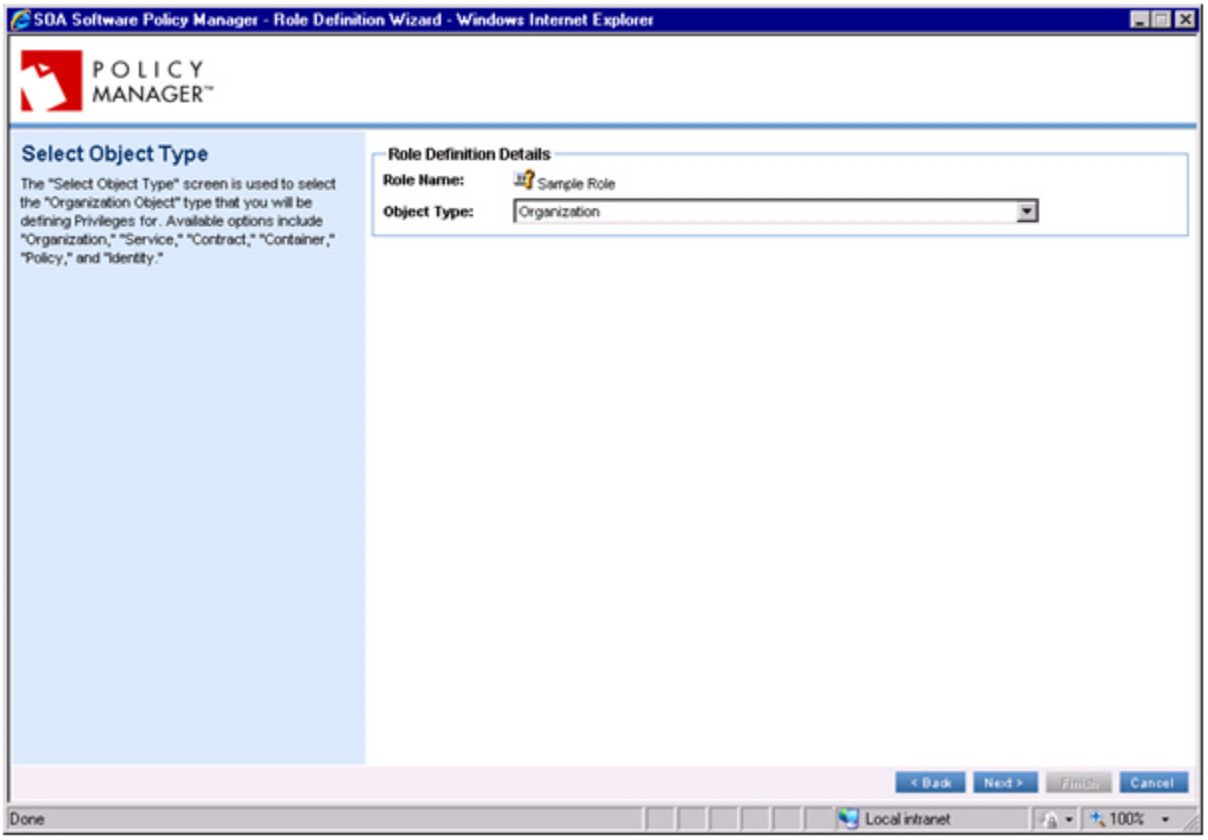


Figure 22: Role Administration Wizard—*Select Object Type*

From the "Object Type" drop-down list box, select the Object Type you would like to assign to the current Role Definition. Click "Next" to continue.

To select object actions:

1 The "Select Object Actions" screen displays. This screen is used to configure access control "Actions" for the current "Role Definition" and the "Permission" (i.e., Allow or Deny) that will be applied to the selected "Actions." This "Action" and "Permission" combination represents a "Privilege" that is part of the "Role Definition." You can have one ore more "Privileges" in a "Role Definition."

You can select specific "Actions" by clicking the checkbox next to each action, or you can select all "Actions" by clicking the "Full Control" checkbox. Selecting this option automatically checks all "Actions" in the list. You can select a "Permission" to apply to the selected "Actions" by clicking the "Allow selected actions" or "Deny selected actions" radio button.

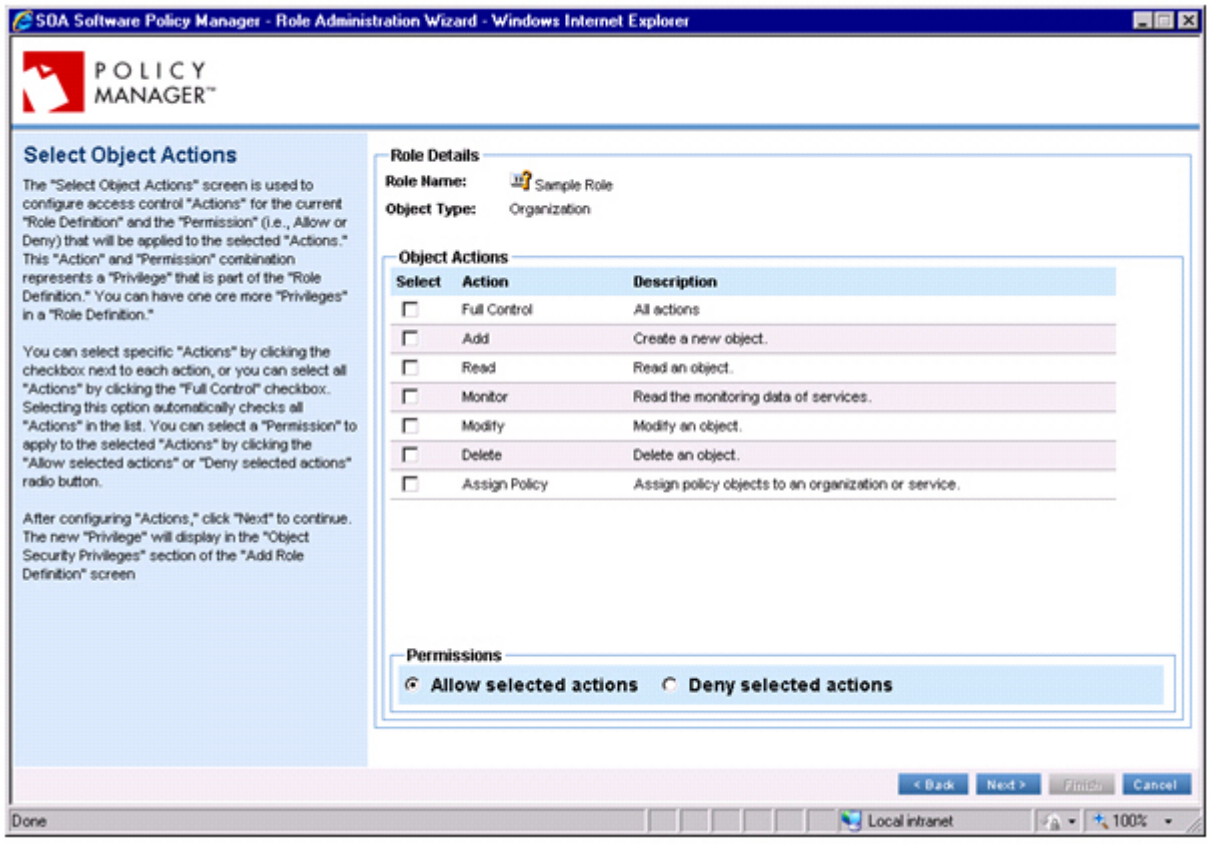


Figure 23: Role Administration Wizard—*Select Object Actions*

After configuring "Actions," click "Next" to continue. The new "Privilege" will display in the "Object Security Privileges" section of the "Modify Role Definition" screen.

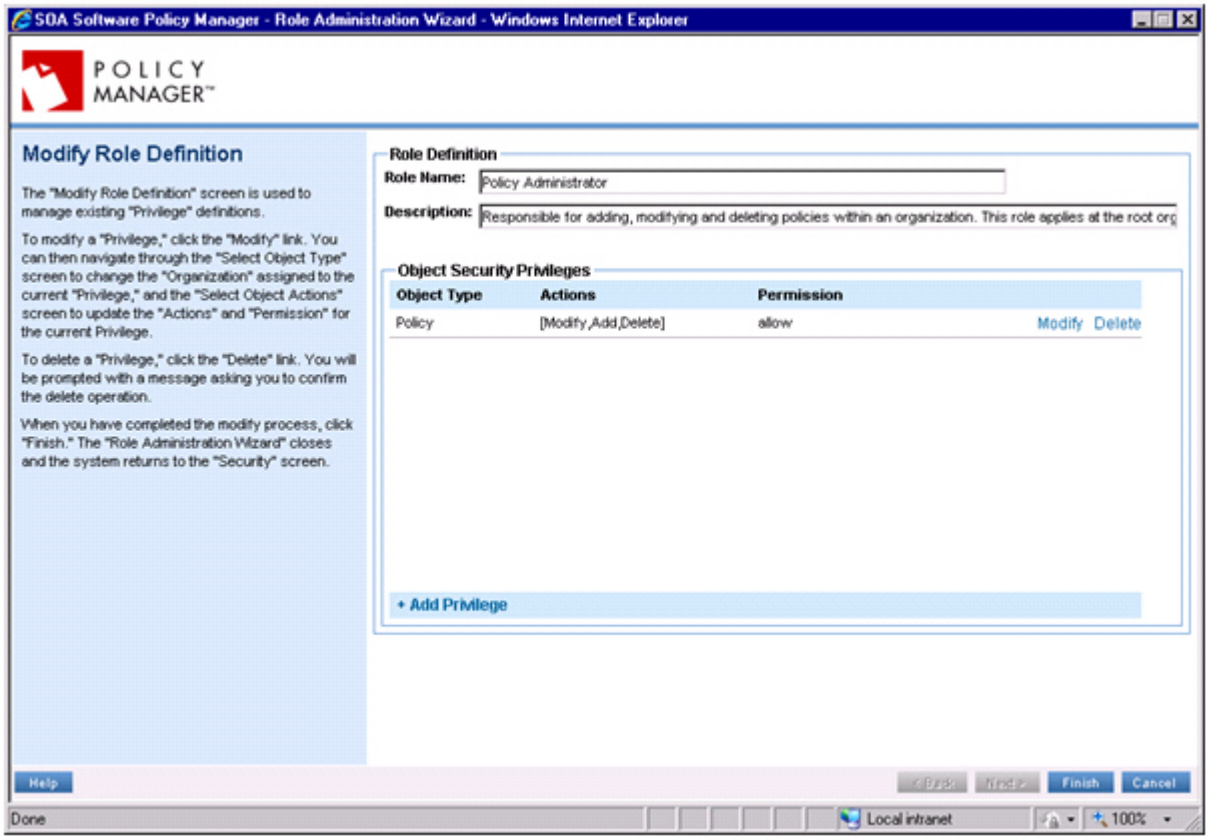


Figure 24: Role Administration Wizard—Select Object Actions (with privilege added to "Object Security Privileges) section.

How do I Delete a Role Definition?

The following procedure illustrates how to delete an Object Security "Role Definition."

To delete an Role Definition:

1. Enter the following navigation path: **Workbench > Root Organization > Security**. The "Security Summary" screen at the Root Organization level displays.
2. In the "Roles Definition Portlet" select the Object Security Role line item you would like to delete. Click "Delete." The following message displays "Are you sure you want to delete this Role Definition?" To delete the Role Definition, click "OK." The system permanently removes the "Role Definition" from the database. To cancel the delete operation, click "Cancel."

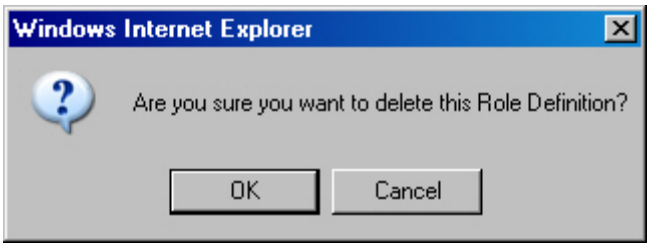


Figure 25: Delete Role Definition

Role Memberships

A "Role Membership" is a replicated instance of a "Role Definition." Each "Role Membership" displays on the "Details" page of each "Sub-Organization." The "Manage Role" function within the "Role Membership Portlet" is used to assign Users/User Groups to each Role. Assigned User/User Groups have access to the functional areas defined within the "Role Definition." You can assign a base set of User/User Group assignments at the Root Organization level and then customize the assignments at different Organization tiers.

How do I View a Summary of current Role Memberships?

The "Security Summary" screen is the starting point for defining and managing the "Role Definitions" and "Role Memberships" that represent the Object Based Security that will control access to Policy Manager functionality that is available within the Management Console

The following key activities can be performed:

To view the security summary:

1 Enter the following navigation path: **Workbench > Browse**. The "Root Organization Summary" screen displays. Click the "Security" tab. The "Security Summary" screen displays.

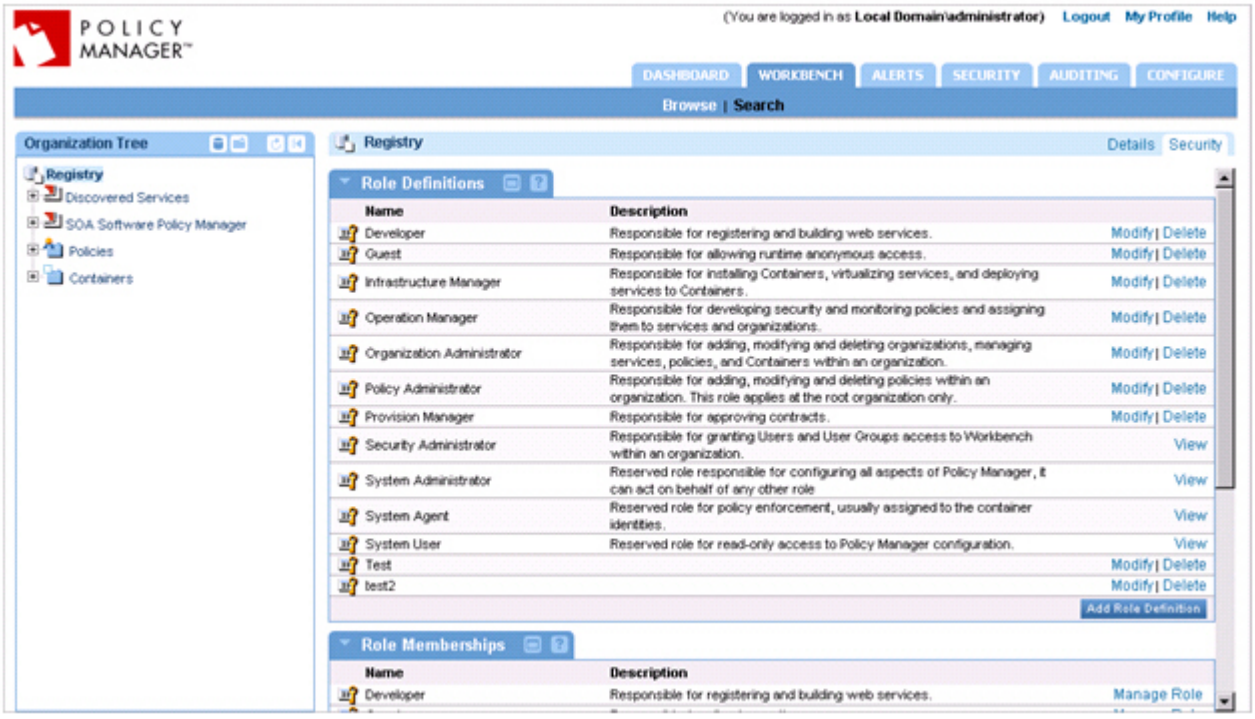


Figure 26: Security Summary (Root Organization Level)

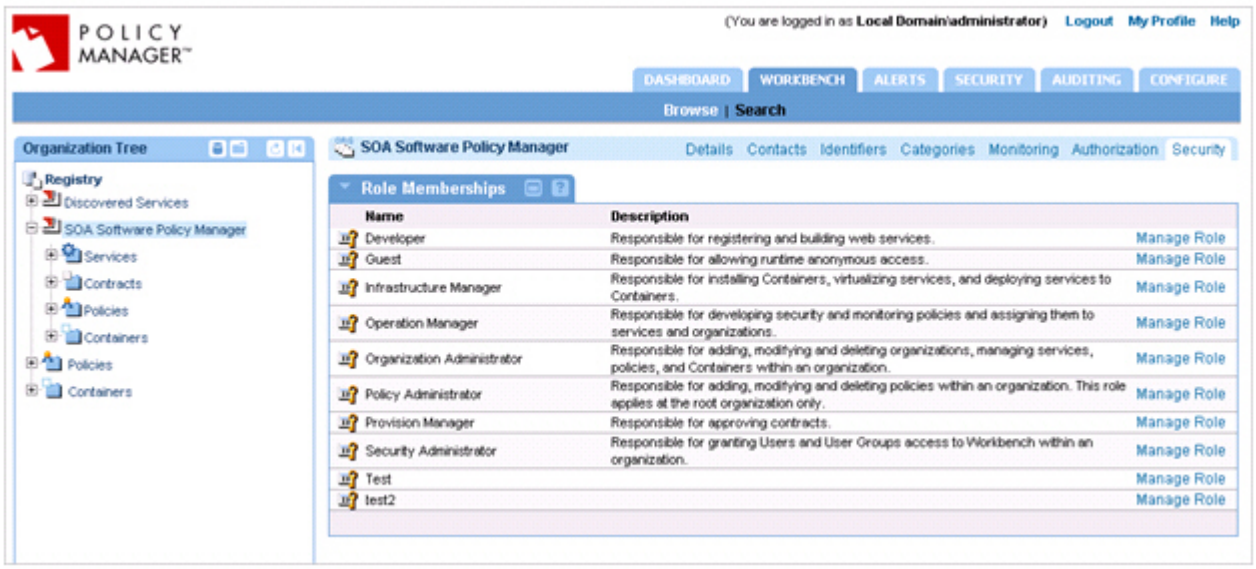


Figure 27: Security Summary (Sub-Organization Level)

How do I Manage a Role?

The "Manage Role" screen allows you to manage user assignments for each organization definition. Users are assigned to an Organization when a Policy Manager deployment is being initially configured, and this function is also used to maintain user assignments.

To manage roles for an organization

1 Enter the following navigation path: **Workbench > Organization > Security**. The "Security Summary" screen displays with the "Role Memberships Portlet."

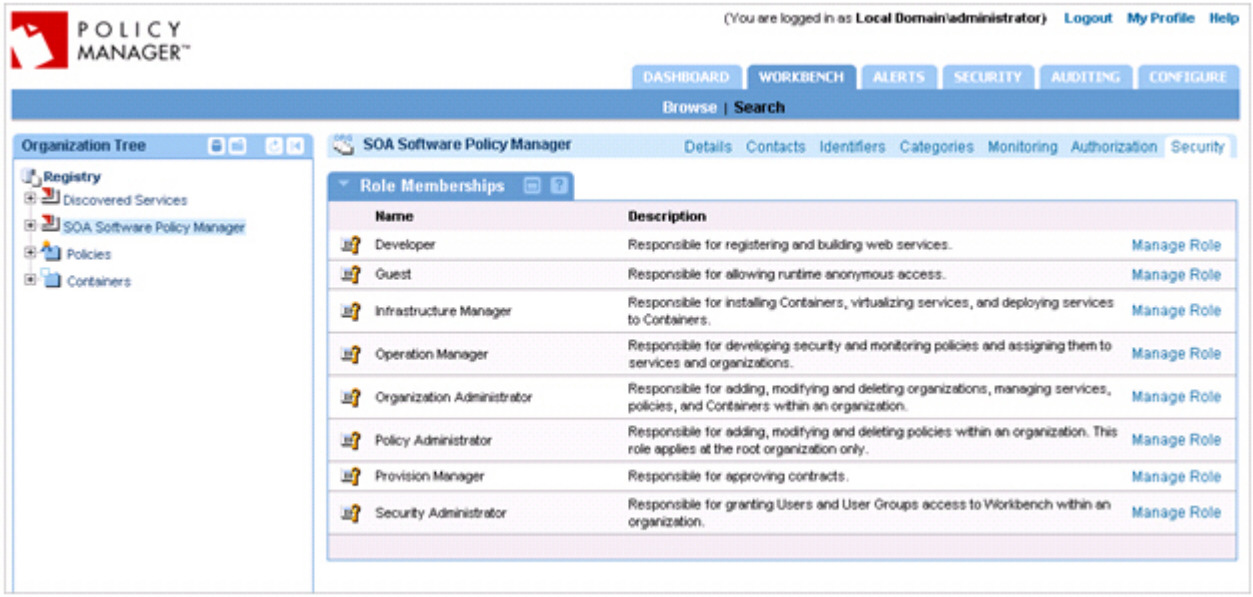


Figure 28: Role Memberships Portlet

2 To manage user assignments for the current Organization, determine the Role line item you would like to manage users for, and click **Manage**. The "Manage Role" screen displays.

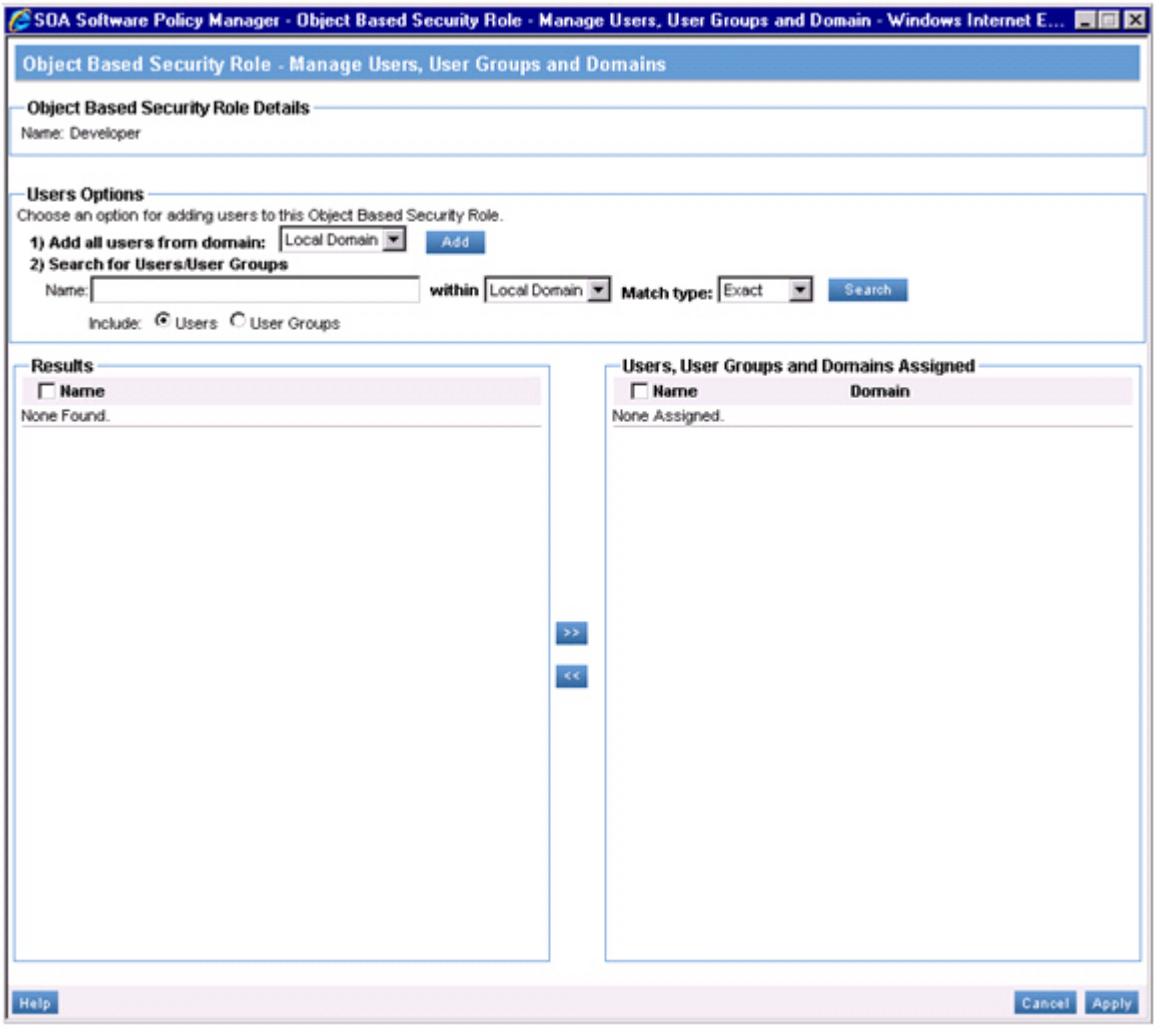


Figure 29: Manage Role (Organizations)

The "Manage Role" screen allows you to search for Policy Manager user accounts and then assign them to the selected Organization. The screen is organized into three sections:

- Object Based Security Role Details—Displays the "Name" of the Role.
- Search—A freeform text box that allows you to specify "Name" search criteria for the users you would like to assign to the Organization, "Domain" drop-down list box, and checkboxes to filter by User/User Group.
- Results—Displays the search results that match your defined search criteria.
- Users, User Groups, and Domains Assigned—Displays the user accounts that are assigned to the current Organization.

3 To search for user accounts to assign to the current Organization:

- In the "Search" section of the "Manage Role" screen enter the "Name" of the user you would like to target. Select the "Domain," and User/User Group filters by clicking the

appropriate checkbox. Click **Search**. Search results that match your search criteria display in the Results screen area.

4 To assign users to the current Organization:

- Select the checkbox next to each username that you would like assigned to the current Organization. To assign the selected users to the user group definition, click  . The selected username copy to the "Assigned" screen area.

5 To remove a username from the "Users, User Groups, and Domains Assigned" screen area:

- Click the checkbox of one or more usernames, and click  to remove it.

6 When you have finished assigning usernames to the Organzation, click **Apply**. The "Manage Role" screen closes.

Creating and Managing Keys and Certificates

Policy Manager provides Certificate Authority (CA) functionality that issues certificates and guarantees the validity of the binding between the certificate owner and its public key. The CA is a trusted authority, and any certificate issued by the CA identifies the owner of the certificate. Therefore the private key that corresponds to the public key in the certificate is deemed to be known only by the specific owner.

Two Certificate Authority options are supported. Policy Manager provides a simplified version of Certificate Authority that can issue and renew X.509 certificates, or one can be imported. The Policy Manager Certificate Authority is intended to be used in test environment for verifying features related to Policy Manager. For production environments, importing a formal CA is recommended (e.g., VeriSign) that aligns with security policy requirements.

Certificate/Key Management Options

Generate Options

The "Generate CA Certificate Signing Request" screen is used to generate a "Certificate Signing Request" for the Policy Manager CA. The request can be sent to a third party Certificate Authority to request an X.509 Certificate. This X.509 Certificate can be imported later to make the Policy Manager CA as a subordinate CA to the third party Certificate Authority.

A "Certificate Signing Request (CSR)" is a file that includes encoded information generated by a web server. It is sent from an applicant to Certificate Authority to request a digital certificate. The CSR contains information identifying the applicant and the public key chosen by the applicant. Before creating a CSR, the applicant first generates a key pair, keeping the private key secret. The corresponding private key is not included in the CSR, but is used to digitally sign the entire request.

A CSR file typically has a .CSR extension but can have other extensions based on the source application that generated the file. Policy Manager supports the PEM (which is a BASE64 encoded PKCS10) for Certificate Signing Requests.

To generate a certificate authority CSR:

1 Enter the following navigation path: **Configure > Security > Certificates > Certificate Authority**. The "Certificate Authority Summary" screen displays. Note in this scenario a CA Certificate must already be defined.

2 To import a CA certificate, click "Generate CA CSR." The "Generate CA Certificate Signing Request" screen displays.

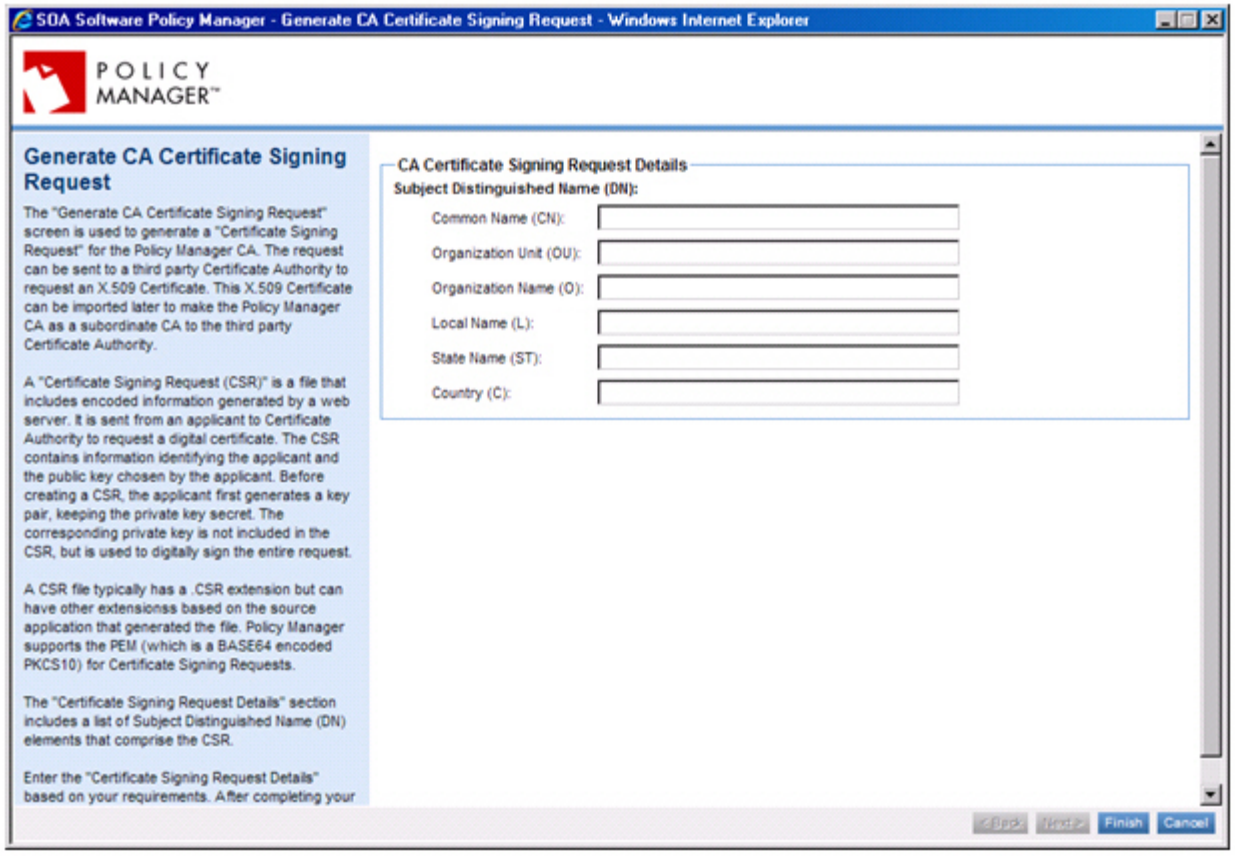


Figure 30: Generate CA Certificate Signing Request

The "Certificate Signing Request Details" section includes a list of Subject Distinguished Name (DN) elements that comprise the CSR.

- Common Name (CN)—A field display that allows you to enter the owner name of the Certificate Authority (e.g., Policy Manager Certificate Authority).
- Organizational Unit (OU)—A field display that allows you to enter the group or department name of the owner of the Certificate Authority.
- Organization Name (O)—A field display that allows you to enter the name of the organization that owns the Certificate Authority.
- Locality Name (L)—A field display that allows you to enter the geographic region of the Organization that owns the Certificate Authority. *This is an optional field.*
- State Name (S)—A field display that allows you to enter the state of the organization that owns the Certificate Authority.

- Country (C)—A field display that allows you to enter the country of the organization that owns the Certificate Authority.
- Enter the "Certificate Signing Request Details" based on your requirements. After completing your entries, click "Finish." The Certificate Signing Request is generated returns to the "Certificate Authority Summary" screen.

Import Options

The "Import CA Certificate" function allows you to replace the existing CA Certificate with an externally imported one, while retaining the current private keys.

To import a CA certificate:

1 Enter the following navigation path: **Configure > Security > Certificates > Certificate Authority**. The "Certificate Authority Summary" screen displays. Note in this scenario a CA Certificate must already be defined.

2 To import a CA certificate, click "Import CA Certificate." The "Import CA Certificate" screen displays.

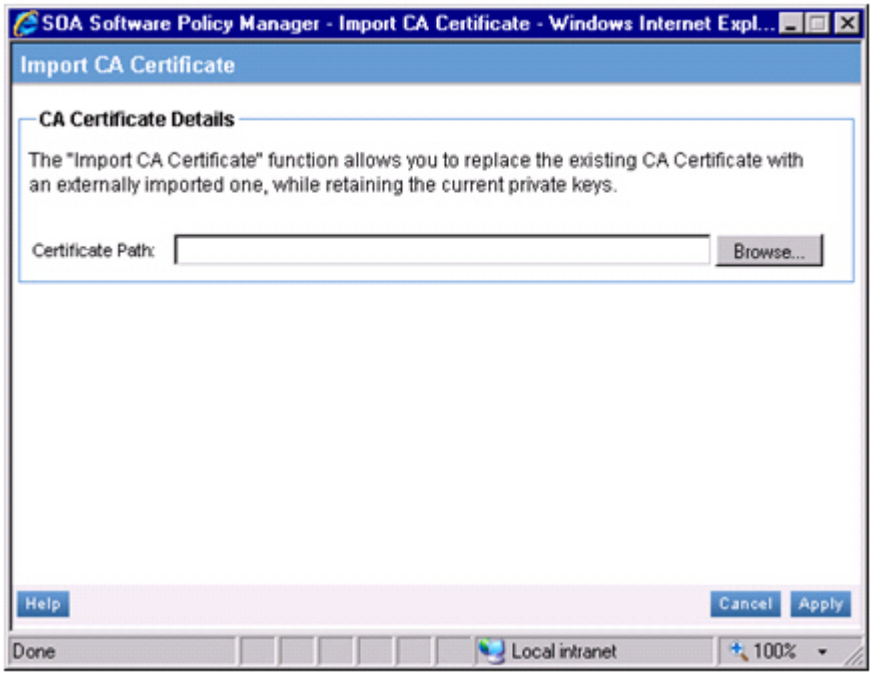


Figure 31: Import CA Certificate

The screen is organized as follows:

CA Certificate Details

- Certificate Path—A text box that allows you to enter the path to the CA Certificate file (manually or by clicking "Browse" and navigating to the directory).

3 Enter the "Certificate Path" manually or by clicking "Browse." After completing your entry, click "Apply." The CA Certificate will be imported and the certificate details will display on "Certificate

Authority Summary" screen. Note that if the specified CA Certificate is not from a trusted source, an error message will be raised.

Export Options

The "Export CA Certificate" function is used if you want to save the CA Certificate to a file to be used by other software applications.

To export a CA certificate:

1. Enter the following navigation path: **Configure > Security > Certificates > Certificate Authority**. The "Certificate Authority Summary" screen displays.
2. To export the current CA Certificate, click "Export CA Certificate." A "File Download" dialog box displays.

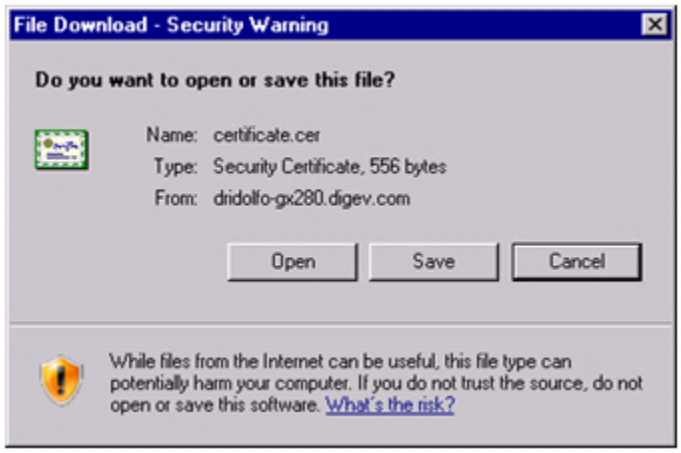


Figure 32: File Download Box—*Export CA Certificate*

3 To view and copy the contents of the Certificate, click "Open." A window displays and presents the contents of the Certificate to be downloaded. Review the contents of the Certificate to determine if you would like to make any adjustments to the prior to performing the download. You can navigate through the three tabs (General, Details, and Certification Path). After reviewing the certificate contents, click "OK." The window closes.

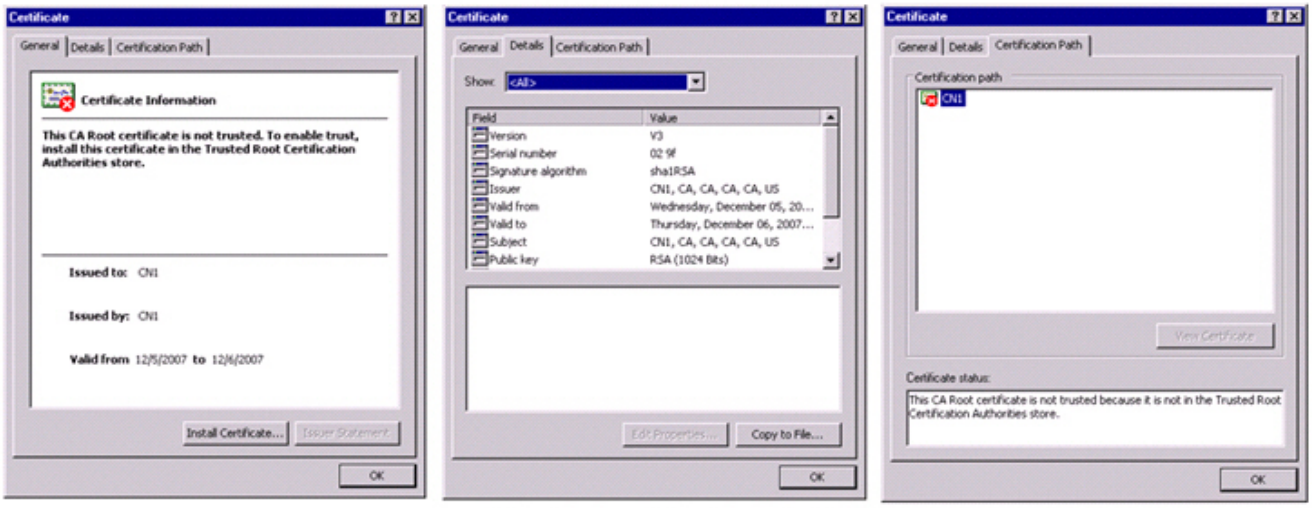


Figure 33: Export CA Certificate—Open Option (General, Details, and Certification Path Tabs)

4 To export the certificate, from the "Certificate Authority Summary," click "Export CA Certificate." A file dialog box displays. Click "Save." The "Save As" screen displays. Navigate to the directory location where you would like to save the CA Certificate. Specify the filename, and click "OK." The CA Certificate is exported and saved using the specified filename and directory location.

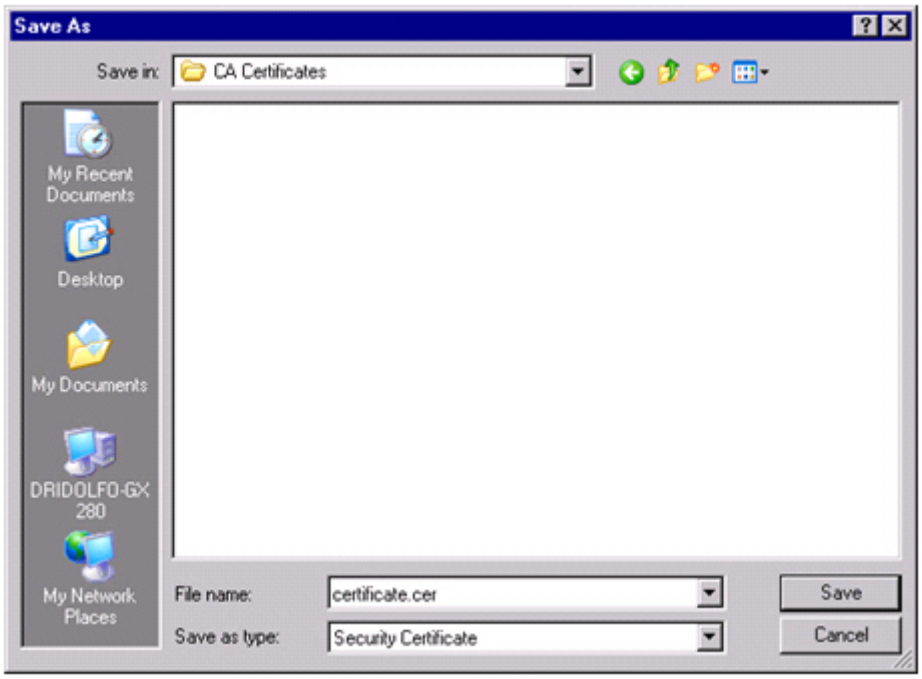


Figure 34: Export CA Certificate—*Save Certificate to File*

Delete Options

The "Certificate Authority Summary" screen provides an option for deleting a CA Certificate. Note that deleting a CA Certificate will invalidate all certificates issued with it.

To delete a CA certificate:

1. Enter the following navigation path: **Configure > Security > Certificates > Certificate Authority**. The "Certificate Authority Summary" screen displays.
2. To delete the CA Certificate click "Delete CA Certificate." The following message displays "Deleting this CA Certificate will invalidate all certificates issued with it. To issue new certificates, you must configure a new CA Certificate which will be used as the trusted authority. Are you sure you want to delete this CA Certificate?" Click "OK" to delete the CA Certificate. The CA Certificate is permanently removed from the system. To cancel the operation, click "Cancel."

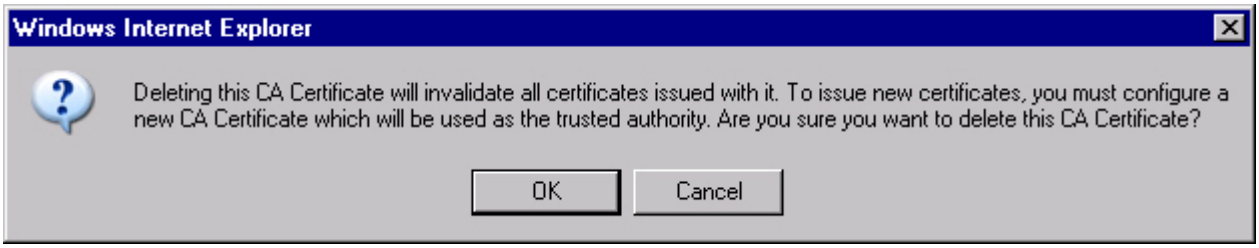


Figure 35: Delete CA Certificate Message

Key Management for Users

Part of the service management process involves assigning a security identity to each service. This is accomplished by generating public and private keys to facilitate service access. A service can be assigned a single key identity and can be deployed in multiple Management Point containers. The complete service identity is then represented by the service key, plus the Management Point listener configuration, and defined service path. Key management functionality is implemented via the Policy Manager Subsystem which provides an auto-generated key capability, and also provides a key import function that allows you to import public keys from a certificate. The Policy Manager Subsystem provides the following key management features for services:

- Import and Generate Option
  - Import Options

Provides ability to import certificate or private key plus certificate by providing a reference to a certificate filename. The Policy Manager Subsystem extracts the public key from the X.509 certificate.

  - Generate Options

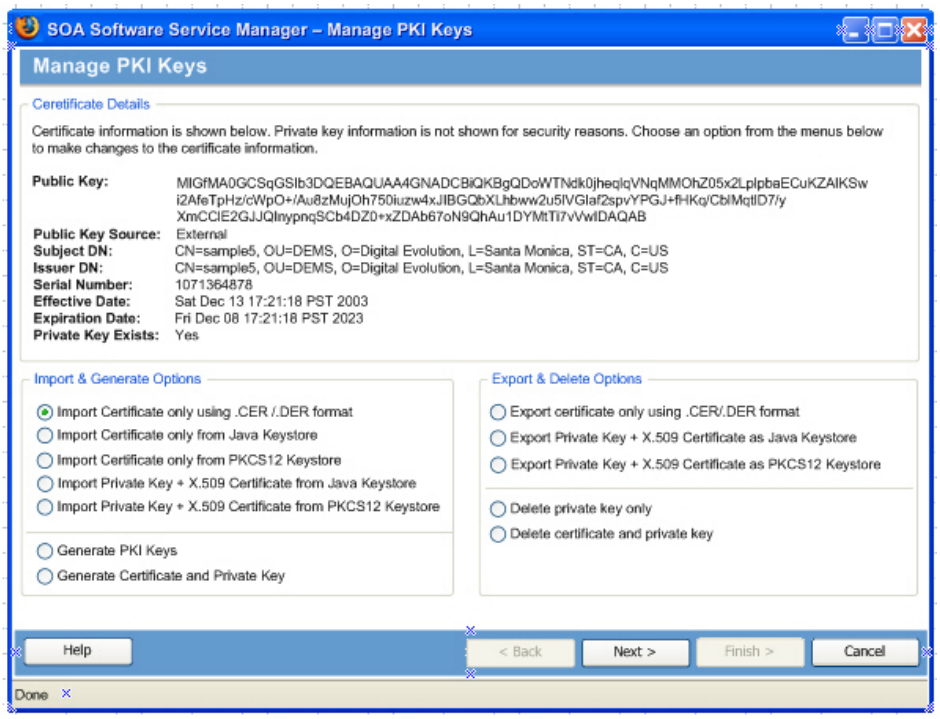
Provides ability to generate public and private keys are created by the Policy Manager Subsystem. Options are provided for generating keys with or without a certificate.

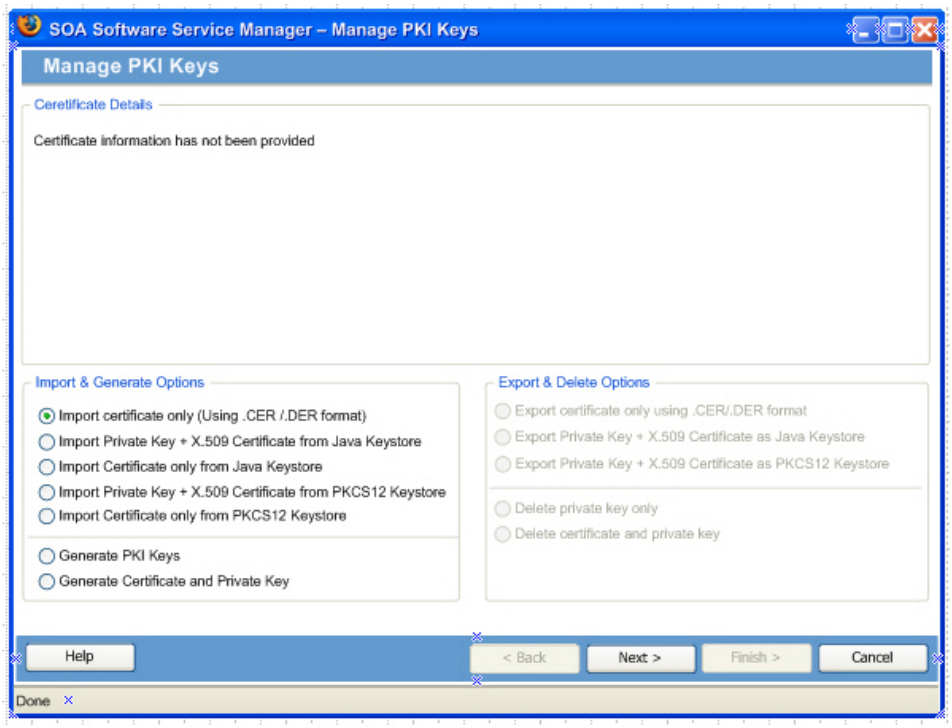
- Export and Delete Options
  - Export Options

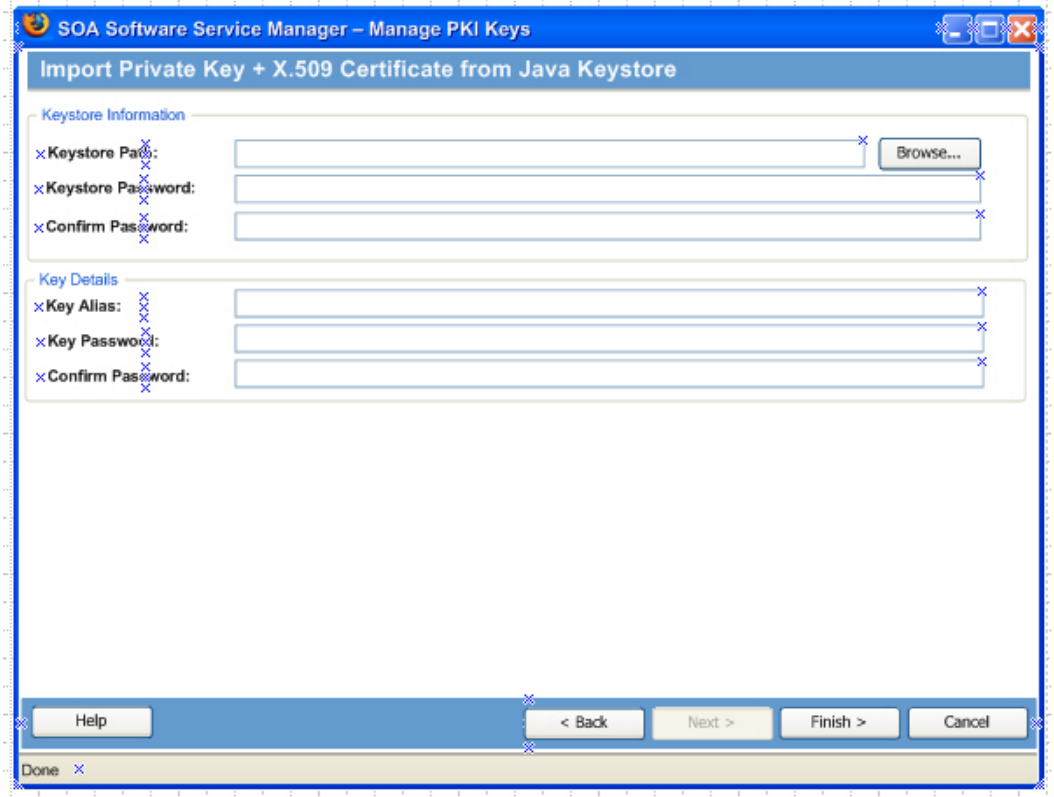
Provides ability to export certificate only or certificate and private key by specifying a directory location.

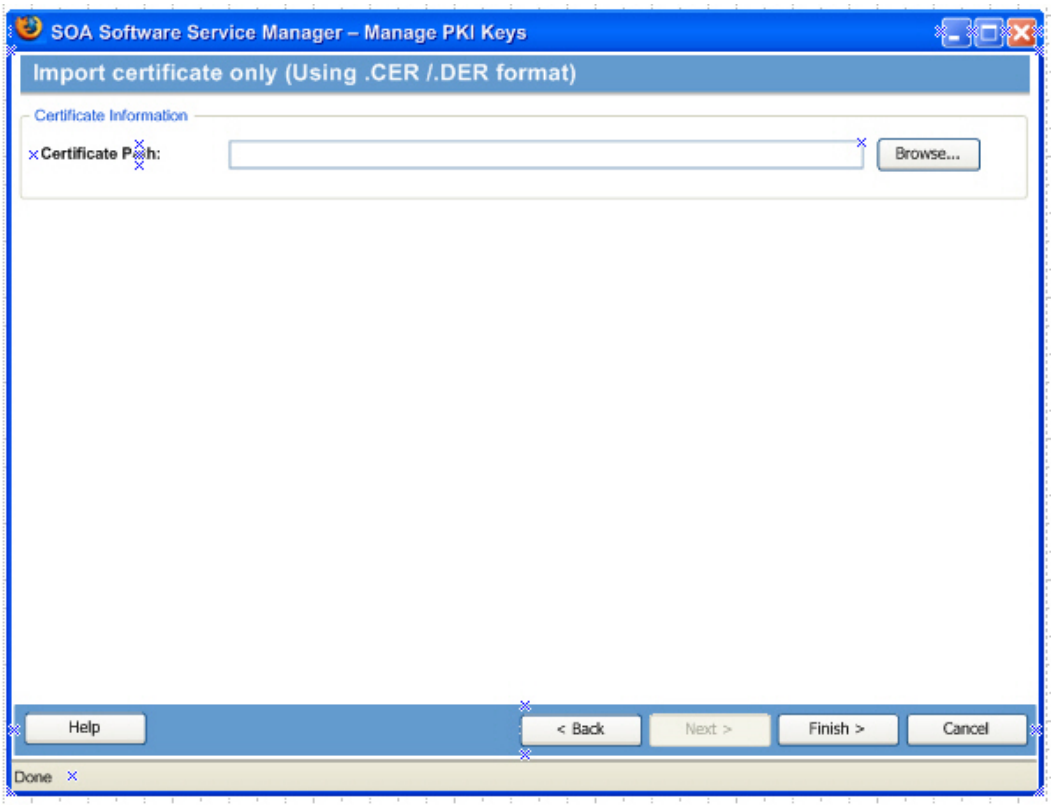
- Delete Options

Provides ability to delete certificate only or certificate and private key by specifying a directory location.






SOA Software Service Manager – Manage PKI Keys
Import Private Key + X.509 Certificate from Java Keystore
Keystore Information
Keystore Path:
Browse...
Keystore Password:
Confirm Password:
Key Details
Key Alias:
Key Password:
Confirm Password:
Help
< Back
Next >
Finish >
Cancel
Done


SOA Software Service Manager – Manage PKI Keys
Import certificate only (Using .CER /.DER format)
Certificate Information
Certificate Path:
Browse...
Help
< Back
Next >
Finish >
Cancel
Done

Key Management for Services

The "Manage PKI Keys (Containers)" screen allows you to configure certificates for the outbound HTTPS configuration of the current *Standalone Management Point container*. Outbound certificates are used if you want the Standalone Management Point container to send messages to the next hop.

For a complete list of supported usage scenarios, see Outbound HTTPS Usage Scenarios.

For a list of prerequisites that must be performed prior to performing key management, see Key Configuration Prerequisites

The following procedures illustrates how to import, export, and delete keys that are assigned to a Subsystem or Standalone Management Point Container using the "Manage PKI Keys (Containers) Wizard" accessible via the "Workbench > Browse > Organization > Containers Summary."

To launch the Manage PKI Keys (Containers) Wizard:

1. Navigate to the "Containers Folder" using the following navigation path: **Workbench > Browse > Organization > Containers**. The "Containers Summary" screen displays.
2. Select the container you would like to modify the outbound HTTPS configuration for, and click the "Container Name." The "Container Details" tab displays and presents a summary of information pertaining to the current container type. Note, three different screen presentations display based on the selected container type (i.e., Subsystem, Standalone Management Point Container, or Embedded Management Point Container).
3. To manage the certificate configuration of the current container, in the "Outbound Configuration Portlet" click **Manage PKI Keys**.

The "Select Certificate Option" screen displays.

Chapter 5 | Auditing Alerts and Security Activities

The "Audit Trails" section of Policy Manager "Auditing" tab provides auditing functionality that allows you to monitor system activity based on alert notifications or security policy-related actions, users, and date and time range. This system activity is saved in an "Audit Trail" which is a log of add, modify, and delete operations performed on all objects in the system. Selected "Audit Trail" records can be exported to an XML file and used for reporting purposes. This data can be used to measure system performance and manage the security of web services. Two types of audit trail functionality are offered:

- Alert Audit Trails

Provides the ability to filter, view, and export data that is logged when an alert is "raised" during the operation of your Policy Manager deployment.

In order to activate the alert audit feature, the "Log Alert" checkbox must be checked in the Alert Code definition. This activity is performed in the "Alerts" section of the Policy Manager.

- Security Audit Trails

Provides the ability to filter, view and export data that is logged when a security policy related action occurs during the operation of your Policy Manager deployment.

The availability of security audit trail data is based on a combination of standard system actions that are automatically logged and whether you configured specific security operations to audit activity (e.g., security policy component). This is accomplished by enabling the "Log Alert" in an Alert Code (for Alert Audit Trails), and enabling "Generate Audit Data" in the Authentication, Authorization, Signature, and/or Signature Verification sections of the Security Policy Component in a Policy definition (for Security Audit Trails).

How do I Perform an Audit Trail Search?

The following procedure illustrates how to perform an Alert Audit Trail search and review search results.

> **Note**: In order for audit trail data to exist, the "Log Alert" checkbox in an Alert Code definition must be checked and active in the system. Refer to About Alerts for more information on alert configuration.

To perform an alert audit trail search:

1 Navigate to the "Monitor" tab using the following navigation path: **Auditing > Alert Audit Trails**. The "Alert Audit Trails Summary" screen displays and presents a list of search criteria options.

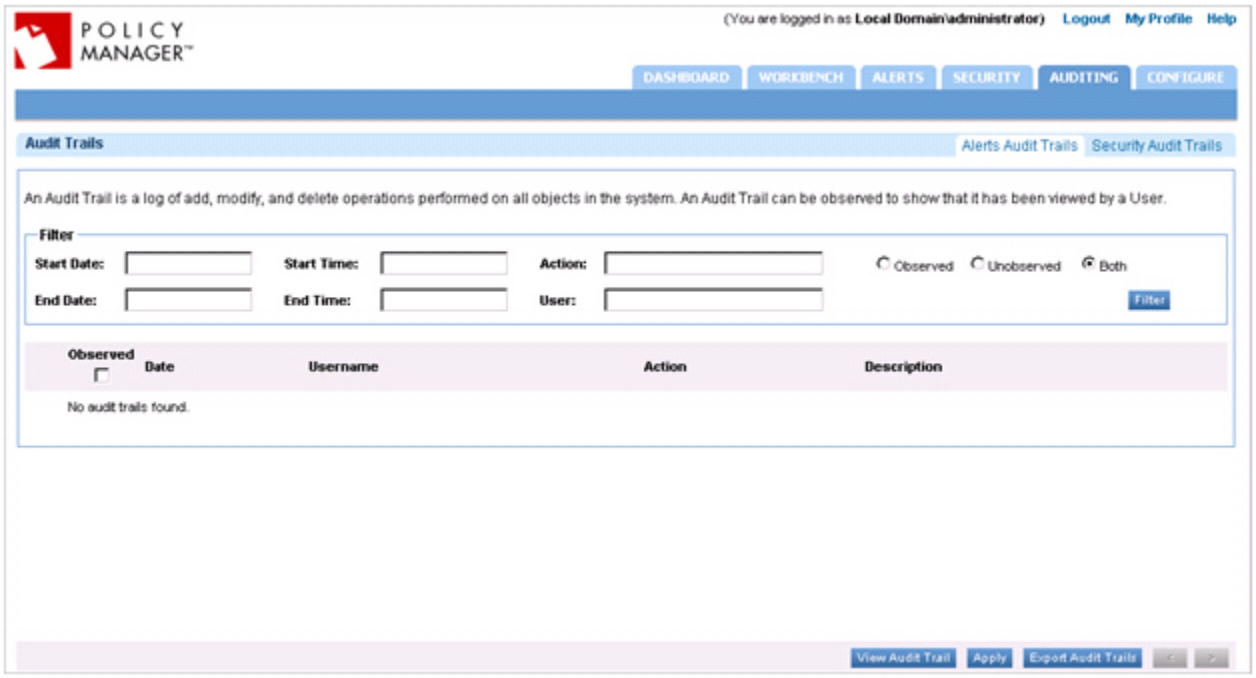


Figure 36: Alerts Audit Trails Summary

Note: The initial search results display presents a listing of all audit trail data associated with the current logged on user.

2 The first step in the audit trail search process is to specify search criteria. Enter the following search criteria. Refer Audit Trail Search Criteria for a complete list if search criteria options and descriptions.

Enter Date Range

The "Start Date" and "End Date" fields invoke a "Calendar Popup" that allows you to navigate Weeks, Months, and Years using arrow keys. To select a specific date, click the date number. The system populates field with the selected date in MM/DD/YYYY format.

Enter Time Range

The "Start Time" and "End time" fields allow you to manually enter a time range using a 24 hour clock (i.e., 20:00:01).

Enter Action

The Action field refers to specific system operations that are invoked relative to add, modify, and delete actions.

Enter User

Enter the username that you would like to audit. Access to this item is based on established privileges.

Audit Status

Select the radio button for the audit status filter you would like to use for current search. Available options include 1) Observed, 2) Unobserved, and 3) Both.

When you are done defining your search criteria, click "Filter." The system displays a listing of matching records in the search results area.

3 The "Audit Trail Search Results" presents a listing of all objects in the system that can be observed for the current logged on user. Each line item is preceded by an Audit Trail icon . System objects can be configured to be observed by clicking the checkbox in the "Observed" column. When a line item is configured to be "Observed" the checkbox displays as checked .

There are two ways to view audit trail data. You can view an audit trail line item by selecting the line item and clicking "View Audit Trail," or you can export the current search results display to an XML file by clicking "Export Audit Trails."

How do I Configure an Audit Trail Status?

The following procedure illustrates how to configure a system object to observe system activity and create a log (i.e., audit trail).

To configure audit trail status:

1 Navigate to the "Users" tab using the following navigation path: **Auditing > Alerts Audit Trails**. The "Alerts Audit Trails Search" screen displays and presents a list of search criteria options.

2 Perform an Alert Audit Trail Search to obtain a list of search results.

3 There are two audit trail status options: Observed or Unobserved.

Note: You can configure an Unobserved system object to be Observed. Once a system object is configured as "Observed" the operation cannot be reversed.

- To configure the audit trail status be observed, select the desired system object line item, and click the "Observed" checkbox.
- To commit the change click "Apply." The Observed checkbox for the selected line displays as .

How do I View Audit Trail Details?

To assist in configuring of your audit data, you can view a profile of a policy object using the "View Audit Trail" feature via the "Alert Audit Trails Summary" screen.

To view alert audit trial details:

1 Enter the following navigation path: **Auditing > Alert Audit Trails**. The "Alert Audit Trails Summary" screen displays and presents a list of search criteria options.

2 Perform an Alert Audit Trail Search to to obtain a list of search results.

3 To view the audit trail of a system object, select the line item of the system object you would like to view the audit trail of, and click "View Audit Trail." The "View Audit Trail" screen displays.

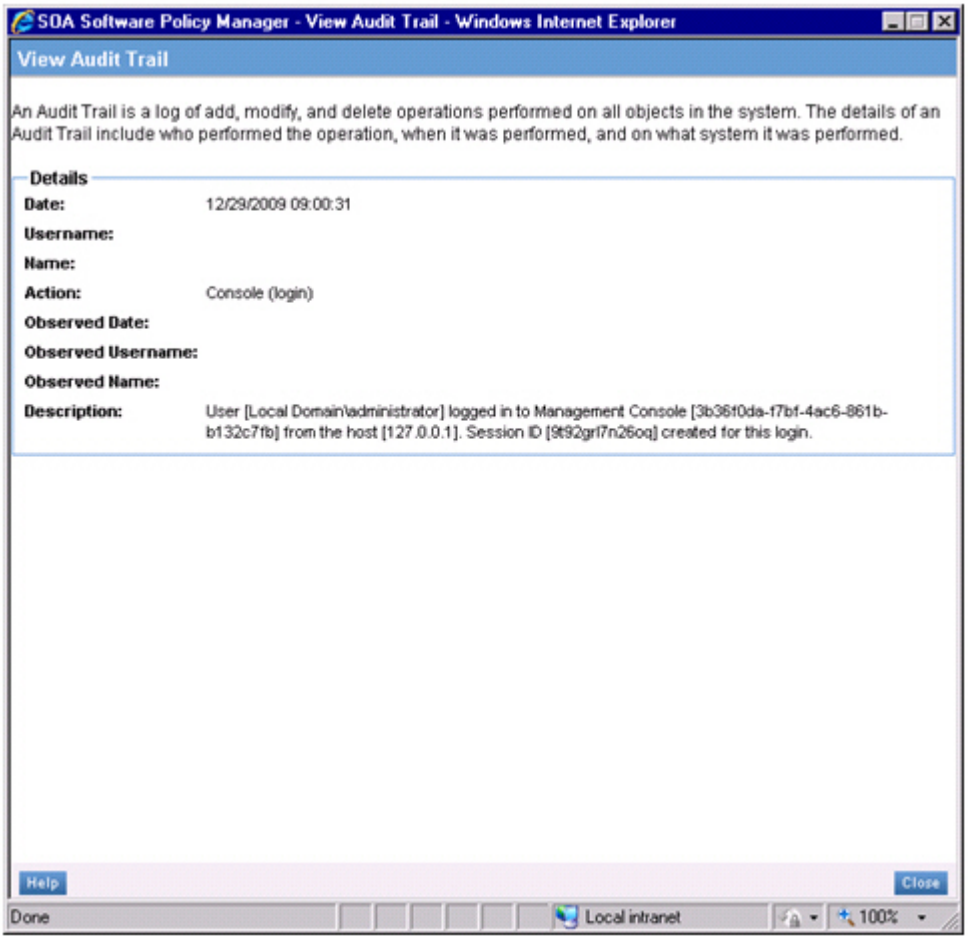


Figure 37: View Audit Trail—*Alerts*

Review the audit trail information as needed. The following table provides descriptions of the policy object information presented on the "Audit Trail Details" screen. See the Audit Trail Field Descriptions for more information on each audit trail field.

4 To exit the "View Audit Trail" screen, click "Close."

How do I Export an Alert Audit Trail?

The Audit Trail functionality provides an additional feature that allows you to export "Audit Trail" data to an XML file. Exported audit trail data can be utilized in third-party statistical tools, and can be archived as a historical record, and so on. The "Audit Trail Export" feature provides export options for managing and archiving audit trail data.

Data to be exported is based on the current filter criteria and resulting audit trail search results. Based on your archiving requirements (i.e., time intervals, etc.) you can customize the Audit Trail search criteria to generate data that aligns with your business needs.

After you have performed an Audit Trail search that contains the focus of data that you want to export, click "Export Audit Trail." The "Audit Trails Export Wizard Welcome" screen displays.

To export alert audit trails:

1. Navigate to the "Monitor" tab using the following navigation path: **Auditing > Alert Audit Trails**. The Alert Audit Trails Summary screen displays.
2. Select the desired search criteria following the guidelines specified in Audit Trail Search Criteria.
3. Verify that the report data meets your requirements. Now you are ready to export the report information.
4. To export the Alert Audit Trail Data, click "Export Audit Trail." The "Audit Trails Export Wizard" screen displays.

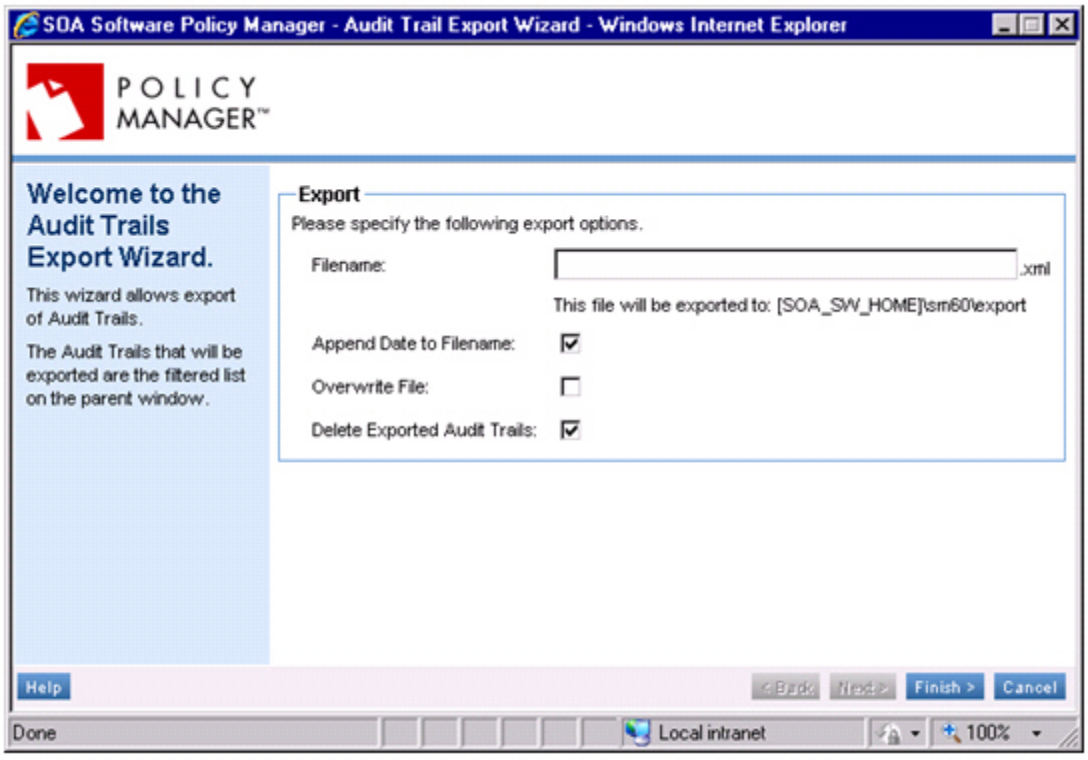


Figure 38: Export Audit Trails—*Alerts*

5. The screen includes an "Export" section that includes the following export customization options are available:
   - Filename

A field display that allows you to enter the name of the .XML file that the Audit Trail data will be saved to. Exported files are always saved to the "export" directory of the current Policy Manager installation.

- Append Date to Filename

A checkbox that allows you to specify if you want to append the system date to the specified .XML filename.

- Overwrite File

A checkbox that allows you to specify if you want to overwrite an existing Audit Trail data file (if applicable).

- Delete Exported Audit Trails

A checkbox that allows you to physically remove gathered Audit Trail data from the Policy Manager system.

6 After entering the file export criteria, click "Finish." The "Export Audit Trail Wizard In Progress" screen displays. This screen displays a profile of export information.

7 When the export is complete, an alert will be sent to the Policy Manager Alerts indicating that the export process has completed, and the "Finish" button will display in-focus.

Click "Finish" to exit the "Audit Trail Export Wizard." The "Completing the Audit Trail Wizard" screen displays and presents a summary of information related to the export including Instance Name, Host Name, File Path, and Alert Code. Click "Close" to exit the "Completing the Audit Trail Wizard" screen.

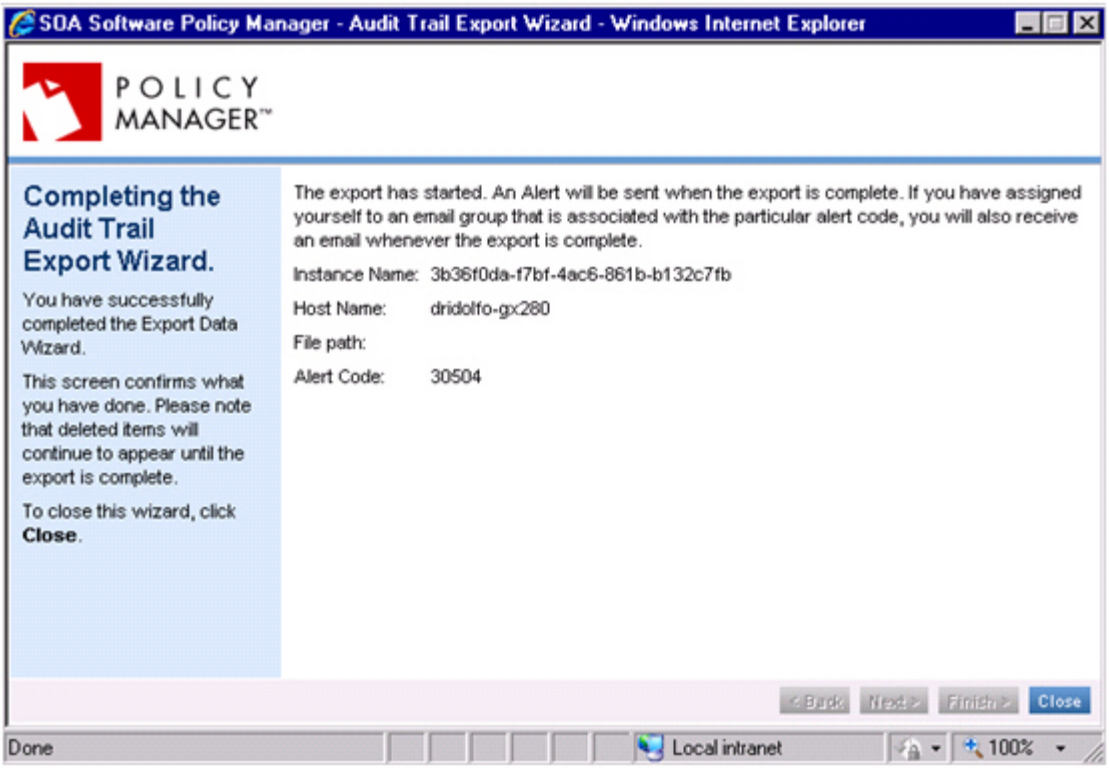


Figure 39: Completing the Audit Trail Wizard—*Alerts*

How do I Schedule an Alert Audit Trail Job?

- Exporting Alert Audit Trails

In addition to the Export Alert Audit Trails option, you can create a scheduled job that invokes a batch script that exports alert audit trail data.

Functionality is equivalent to the "Export" button in the `Monitoring > Audit Trails > Alert Audit Trails` section of the "Management Console" except that it allows you to run the export program as a batch script synchronously. Export results are achieved when the program exits.

Filtering is optional. If any of the filtering criteria is not specified, the following is used:

```
-filterStartDate - today's date
-filterStartTime - 00:00:00
-filterEndDate - today's date
-filterEndTime - 23:59:59
```

Note: In case of error during export, partially exported file doesn't get deleted.

- Usage Options:

Option	Description
-appendDateToFileName	Append Date to Filename
-deleteFilteredData	Delete Exported Data. (Deletes filtered alert audit trails only.)
-filterStartDate (MM-dd-yyyy)	Export data filter start date
-filterStartTime (HH:mm:ss)	Export data filter start time
-filterEndDate (MM-dd-yyyy)	Export data filter end date
-filterEndTime (HH:mm:ss)	Export data filter end time
-outfile <outfile>	This file will be exported to: [SOA_PM_HOME]\pm60\export. This option is required.
-overrideExistingFile	Overwrite Existing File

- Minimum Required Options:

```
-outfile <outfile>
```

- Sample Program Call:

```
exportAlertAuditTrail -deleteFilteredData -outfile alaudit1_ -appendDateToFileName -filterStartDate 01-01-2000 -filterEndDate 04-24-2006
```

Security Audit Trails

How do I Configure a Security Audit Trail Status?

The following procedure illustrates how to configure a system object to observe system activity and create a log (i.e., audit trail).

To configure security audit trail status:

1. Enter the following navigation path: **Auditing > Security Audit Trails**. The "Security Audit Trails Summary" screen displays and presents a list of search criteria options.
2. Perform a Security Audit Trail Search to obtain a list of search results.
3. There are two audit trail status options: Observed or Unobserved.

Note: You can configure an Unobserved system object to be Observed. Once a system object is configured as "Observed" the operation cannot be reversed.

- To configure the audit trail status be observed, select the desired system object line item, and click the "Observed" checkbox.
- To commit the change click "Apply." To commit the change click "Apply." The Observed checkbox for the selected line displays as  .

How do I View Security Audit Trail Details?

To assist in configuring of your audit data, you can view a profile of a policy object using the "View Audit Trail" feature via the "Security Audit Trails Summary" screen. The "View Audit Trails" screen provides the following policy object information:

To view security audit trail details:

1 Enter the following navigation path: **Auditing > Security Audit Trails**. The "Security Audit Trails Summary" screen displays and presents a list of search criteria options.

2 Perform a Security Audit Trail Search to obtain a list of search results.

3 To view the audit trail of a system object, select the line item of the system object you would like to view the audit trail of, and click "View Audit Trail." The "View Audit Trail" screen displays.

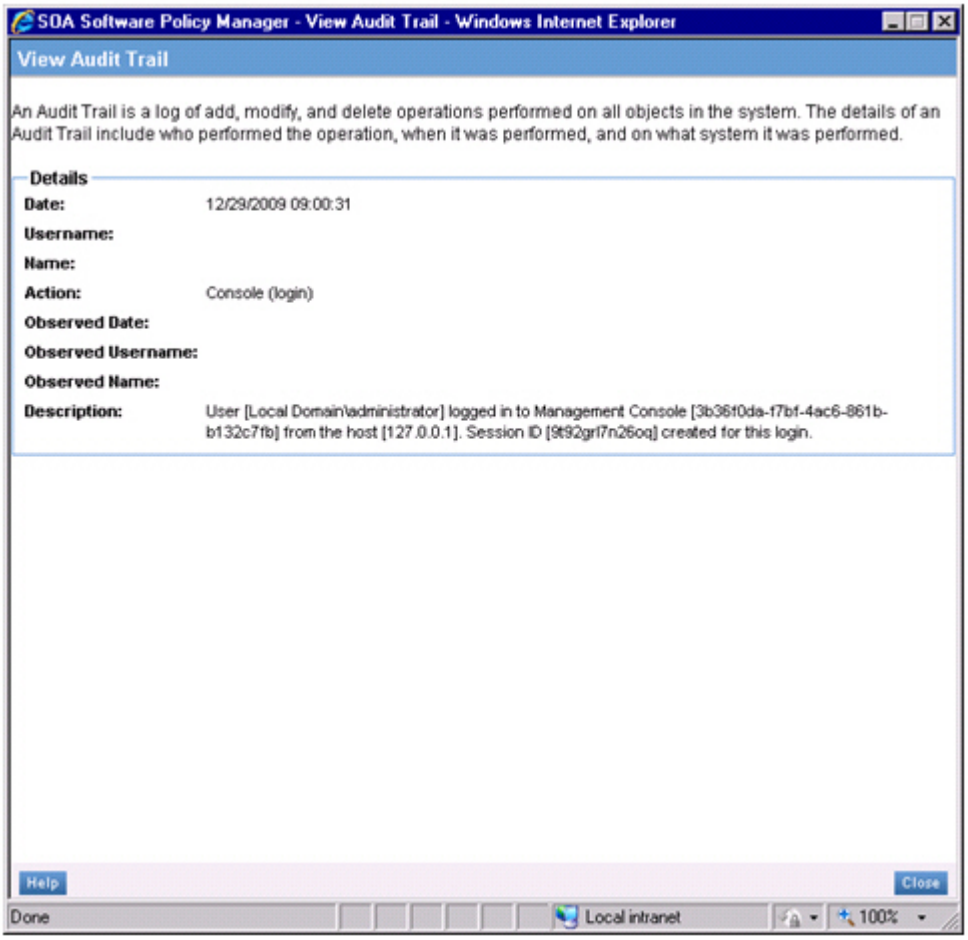


Figure 40: View Audit Trail—*Security*

Review the audit trail information as needed. The following table provides descriptions of the policy object information presented on the "Audit Trail Details" screen. See the Audit Trail Field Descriptions for more information on each audit trail field.

4 To exit the "View Audit Trail" screen, click "Close."

How do I Export a Security Audit Trail?

The Audit Trail functionality provides an additional feature that allows you to export "Audit Trail" data to an XML file. Exported audit trail data can be utilized in third-party statistical tools, and can be archived as a historical record, and so on. The "Audit Trail Export" feature provides export options for managing and archiving audit trail data.

Data to be exported is based on the current filter criteria and resulting audit trail search results. Based on your archiving requirements (i.e., time intervals, etc.) you can customize the Audit Trail search criteria to generate data that aligns with your business needs.

After you have performed an Audit Trail search that contains the focus of data that you want to export, click "Export Audit Trail." The "Audit Trails Export Wizard Welcome" screen displays.

To export security audit trails:

1. Enter the following navigation path: **Auditing > Security Audit Trails**. The "Security Audit Trails Summary" screen displays.
2. Perform a Security Audit Trail Search to obtain a list of search results.
3. Verify that the report data meets your requirements. Now you are ready to export the report information.
4. To export the Security Audit Trail Data, click "Export Audit Trail." The "Audit Trails Export Wizard" screen displays.

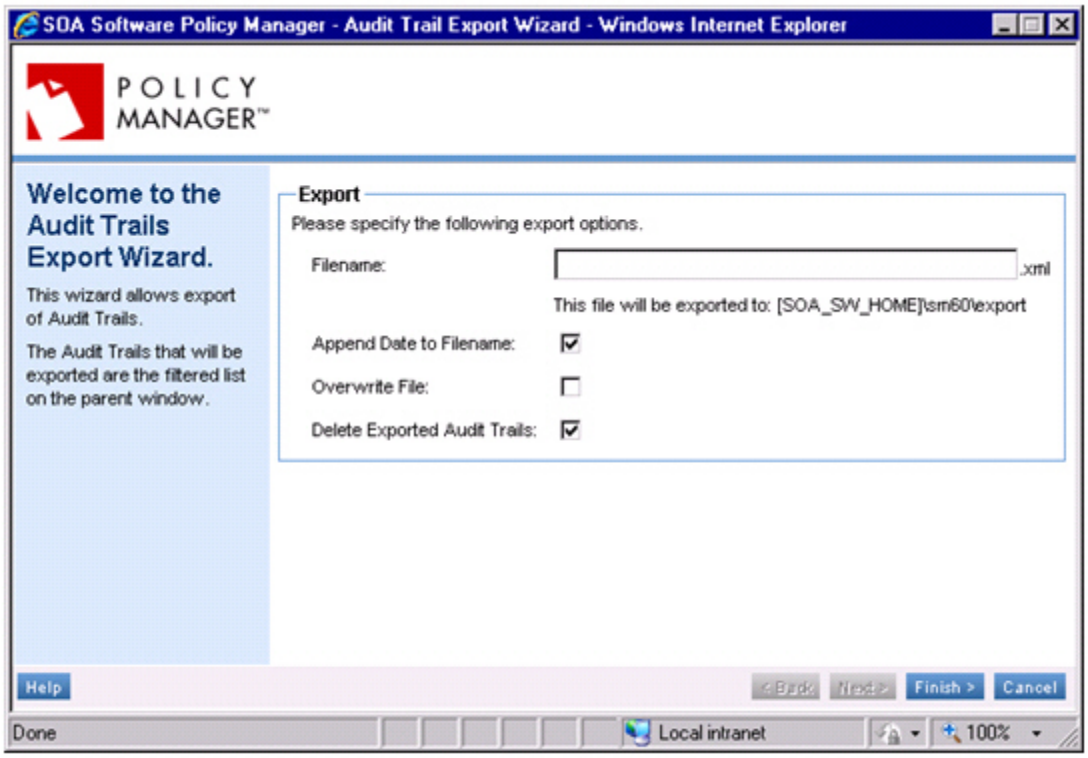


Figure 41: Export Audit Trails—*Security*

5. The screen includes an "Export" section that includes the following export customization options are available:
   - Filename—A field display that allows you to enter the name of the .XML file that the Audit Trail data will be saved to. Exported files are always saved to the "export" directory of the current Policy Manager installation.
   - Append Date to Filename—A checkbox that allows you to specify if you want to append the system date to the specified .XML filename.

- Overwrite File—A checkbox that allows you to specify if you want to overwrite an existing Audit Trail data file (if applicable).
- Delete Exported Audit Trails—A checkbox that allows you to physically remove gathered Audit Trail data from the Policy Manager system.

6 After entering the file export criteria, click "Finish." The "Export Audit Trail Wizard In Progress" screen displays. This screen displays a profile of export information.

7 When the export is complete, an alert will be sent to the Policy Manager Alerts indicating that the export process has completed, and the "Finish" button will display in-focus.

Click "Finish" to exit the "Audit Trail Export Wizard." The "Completing the Audit Trail Wizard" screen displays and presents a summary of information related to the export including Instance Name, Host Name, File Path, and Alert Code. Click "Close" to exit the "Completing the Audit Trail Wizard" screen.

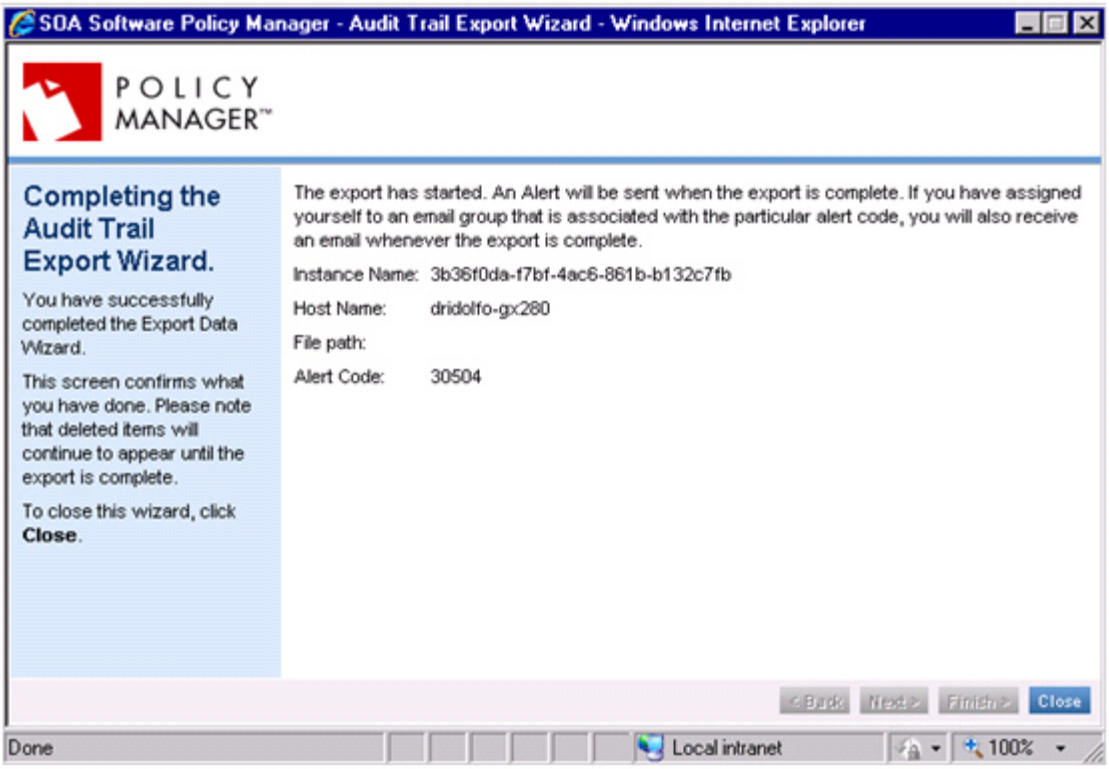


Figure 42: Completing the Audit Trail Wizard—*Security*

How do I Schedule a Security Audit Trail Job?

In addition to the Export Security Audit Trails option, you can create a scheduled job that invokes a batch script that exports security audit trail data.

Functionality is equivalent to the "Export" button in the Monitoring > Audit Trails > Security Audit Trails section of the "Management Console" except that it allows you to run the export program as a batch script synchronously. Export results are achieved when the program exits.

Filtering is optional. If any of the filtering criteria is not specified, the following is used:

-filterStartDate - today's date

```
-filterStartTime - 00:00:00
-filterEndDate - today's date
-filterEndTime - 23:59:59
```

Note: In case of error during export, partially exported file doesn't get deleted.

- Usage Options:

Option	Description
-appendDateToFileName	Append Date to Filename
-deleteFilteredData	Delete Exported Data. (Deletes filtered alert audit trails only.)
-filterStartDate (MM-dd-yyyy)	Export data filter start date
-filterStartTime (HH:mm:ss)	Export data filter start time
-filterEndDate (MM-dd-yyyy)	Export data filter end date
-filterEndTime (HH:mm:ss)	Export data filter end time
-outfile <outfile>	This file will be exported to: [SOA_SW_HOME]\sm31\export. This option is required.
-overrideExistingFile	Overwrite Existing File

- Minimum Required Options:

```
-outfile <outfile>
```

- Sample Program Call:

```
exportSecurityAuditTrail -deleteFilteredData -outfile secaudit1_ -appendDateToFileName -filterStartDate 01-01-2000
-filterEndDate 04-24-2006
```

Chapter 6 | Publishing Services

Policy Manager provides the ability to manage both "physical" and "virtual" service types. The general process involves configuring the applicable Container mode based on the service type to be managed (i.e., physical or virtual), and performing a Policy Manager wizard-based configuration process to bring the service under management. At the end of this process, service types reside in the Policy Manager "Registry." Additional policy configuration or registry updates can be performed as needed.

- Physical Services
- Virtual Services

Physical Services

A "physical" web service contains a WSDL (Web Service Description Language), and endpoint (i.e., port address). The service type can reside on any platform (i.e., application service, mainframe, etc.), and is accessible using the port address. If the service is not managed by a Container, it is referred to as an "unmanaged service."

Physical services that reside inside an application server can be managed using Policy Manager with an embedded deployment (i.e., Agent). The embedded deployment employs a "Discovery Agent" that is application-specific. The agent provides service discovery functionality and deploys an application-specific handler that interacts with Policy Manager to perform various service management functions.

> **Note**: A physical service can be deployed in an "embedded deployment" only. Policy configurations for a physical service run inside the embedded deployment (i.e., container).

Virtual Services

A virtual service is an intermediary service between a consumer and a physical, or terminal, service provider. Virtual services enhance the capabilities of a "physical service" by serving as a proxy or intermediate point for its parent services. You can create multiple "virtual service" instances of a physical service. This allows you to generate multiple versions of the same physical service that can be configured to support a variety of different business needs. You can define custom policy configurations for each virtual service, and perform routing based on your unique requirements (e.g., content, platform, version, etc.) to internal or external consumers.

You can associate Quality of Service (QoS) policies defined in Policy Manager with virtual service operations. Service performance can then be monitored using the "Dashboard" performance metrics charts and "Alert Monitoring" functionality. You can also create a new service that is a "composite" of multiple virtual services that support a group of common business processes.

When you "virtualize" (i.e., create an instance of) a physical web service you create a "parent" of the physical service. Each virtual service has a unique access point that can receive SOAP requests. These requests are then forwarded to one of the physical service "parents" depending on which operation the request is invoking.

Note: A virtual service can be deployed in an "SOA Container" or "5.2 Standalone Management Point" only. Policy configurations for a virtual service run inside these container types.

Virtual services follow a relatively simple pattern in order to provide fairly complex functions. First, a virtual service will leverage WSDL documents and WS-Policies for publishing the interfaces, bindings, security, and message exchange patterns supported to consumers. The virtual service will also leverage the WSDL documents and WS-Policies of provider services that it consumes to adequately exchange messages according to the provider's interface, bindings, security, and message exchange pattern requirements.

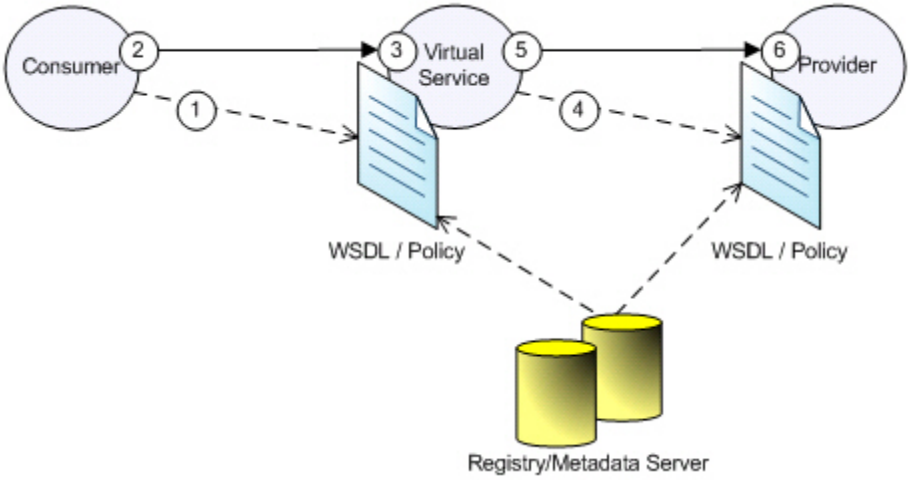


Figure 43: Use of WSDL and Policy

In the "Use of WSDL and Policy" figure the virtual service presents a WSDL document and attached policies to the consumer. The message exchange patterns described in those artifacts are implemented by the virtual service. The policies described in those artifacts are enforced. The virtual service then will act as a consumer to the real provider. It will implement the supported message exchange patterns described in the provider's WSDL document. The virtual service will also implement all policies in the provider's WSDL document to ensure compliance.

In some scenarios the virtual service may not be requested to enforce all policies. For example, when performing only a monitoring function, the virtual service may not enforce WS-SecurityPolicy. At the same time, however, the virtual service may be requested to enforce WS-Addressing policies so that asynchronous delivery can be performed. This is done by configuring the virtual service to ignore specific policies while still publishing them in their WSDL to consumers.

The WSDL document of both the virtual service and the provider service provides binding information. In the case of the virtual service, the binding information is used by the consumer to properly encode and transport messages. A virtual service can support multiple bindings, such as SOAP 1.1/HTTP, SOAP 1.2/JMS, or REST. A virtual service can invoke the down-stream provider service using any of the bindings stated in its WSDL document, even if different from what the virtual service publishes. This enables binding mediation.

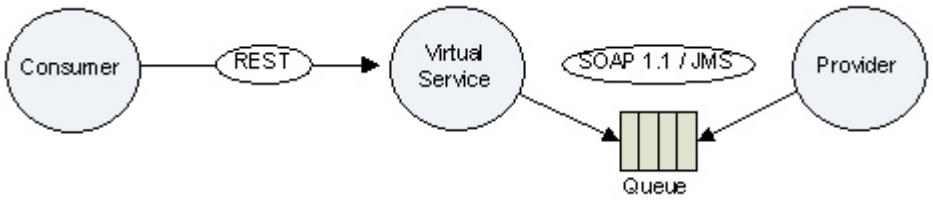


Figure 44: Binding Mediation

In the "Binding Mediation" figure above the virtual service publishes and receives messages using a REST binding. However, the provider only publishes a SOAP 1.1 / JMS binding. The virtual service invokes the provider using JMS. Since the REST call is a synchronous call, the JMS request must be fulfilled before the HTTP timeout of the consumer.

The second key part of the virtual service design pattern is the execution of a business process as the implementation of virtual service operations. The simplest implementation of a virtual service operation is to perform one direct call to the down-stream provider. However, in many mediating scenarios such as those where the virtual service and down-stream provider have different interfaces, there may be more work necessary to take the incoming message to the virtual service and deliver it to the provider.

The implementation of a virtual service operation can include several steps, or activities, in a business process, such as transformation, auditing, and content based routing. The process is defined using a subset of the Business Process Execution Language (BPEL). Although BPEL allows for invoke multiple services from one process, the virtual service implementation of BPEL can only invoke one. However, it can make use of the branching capabilities of BPEL to make decisions as to which service will be invoked, providing content based routing.

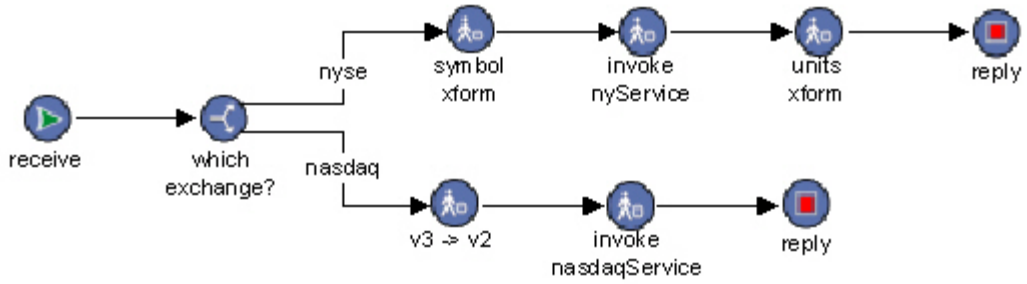


Figure 45: Sample Operation Implementation Process

The "Sample Operation Implementation Process" figure illustrates a sample process for a fictitious virtual service operation that places a trade with one of two down-stream stock market service providers. In the example a decision activity is performed right away to decide which exchange to place the trade with. The decision is based on the content of the trade request. Two branches can be performed, one to place a trade on the NYSE and the other to place a trade on the NASDAQ. When placing a trade on the NYSE, a transformation must be done on the symbol in the trade request before invoking the NYSE service. Then the units in the response must also be transformed before finally returning the response back to the consumer.

Two important implementation details of an operation are not part of the business process, the definition of identity and the treating of headers. Every virtual service has its own identity. It has its own PKI keys and certificate. These can be used when consumers encrypt message content and when the

virtual service inserts tokens and signs message content. In addition, the identity or identities provided in consumer messages can be inserted into downstream service calls if configured.

Binding and transport headers are typically used to aid in the delivery of a message. Some headers are required by service implementations though, such as WS-Addressing headers when sending WS-Transfer messages. A virtual service may be the termination point for some headers while other headers may need to be passed on to the down-stream provider service. Headers that must be preserved and passed to the down-stream stream can be specified in the implementation details of an operation.

Since binding mediation may be occurring between the consumer and the provider, the preserved headers may not actually be sent to the provider, but would most likely not be necessary in this scenario. For example, if the consumer is issuing SOAP 1.1 messages to the virtual service, but the virtual service is communicating with the provider using REST, no SOAP headers will be preserved.

The final key part of the virtual service design pattern is dynamic endpoint resolution. As is typical in BPEL definitions, the endpoints of the down-stream providers that are invoked are not included. BPEL definitions specify the invoked services only by interface. The virtual service BPEL definitions go a step further than interface and actually specify the service to invoke, but not the physical endpoint.

The endpoint of the down-stream service to invoke is determined at runtime. As mentioned previously, the WSDL document of the down-stream service is read by the virtual service. That WSDL document will include the list of endpoints that implement the service. The virtual service will perform a round-robin or random load balancing of the down-stream endpoints listed in the document. If a message cannot be delivered to an endpoint due to communication problems, the message will be delivered to an alternate endpoint for fail-over. The endpoint which could not be communicated with is suspended from the load-balancing algorithm for a short period of time to avoid repetitive failures.

Hosting a Service

The "Host Virtual Service Wizard" allows you to add a virtual service to the current container and configure virtual service access points.

To host a virtual service:

1 Navigate to the "Containers Folder" using the following navigation path: **Workbench > Browse > Organization > Containers**. The "Containers Summary" screen displays.

2 From the Containers Folder in the Organization Tree, or by selecting a line item on the "Containers Summary" select a container that you would like to configure or manage virtual service hosting for. The "Container Details" tab displays. Click the "Hosted Services" tab. The "Hosted Services" screen displays and presents a list of virtual services that are hosted by the current container.

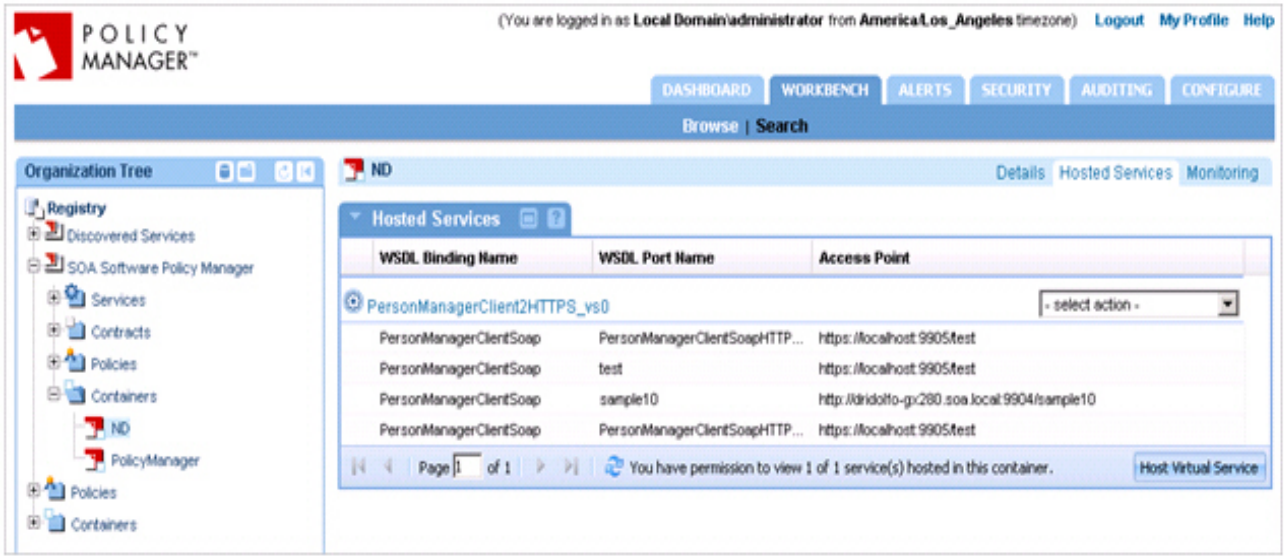


Figure 46: Hosted Services Summary

3 To add a virtual service to the current container, click "Host Virtual Service." The "Host Virtual Service Wizard" launches and displays the "Select Virtual Service" Screen.

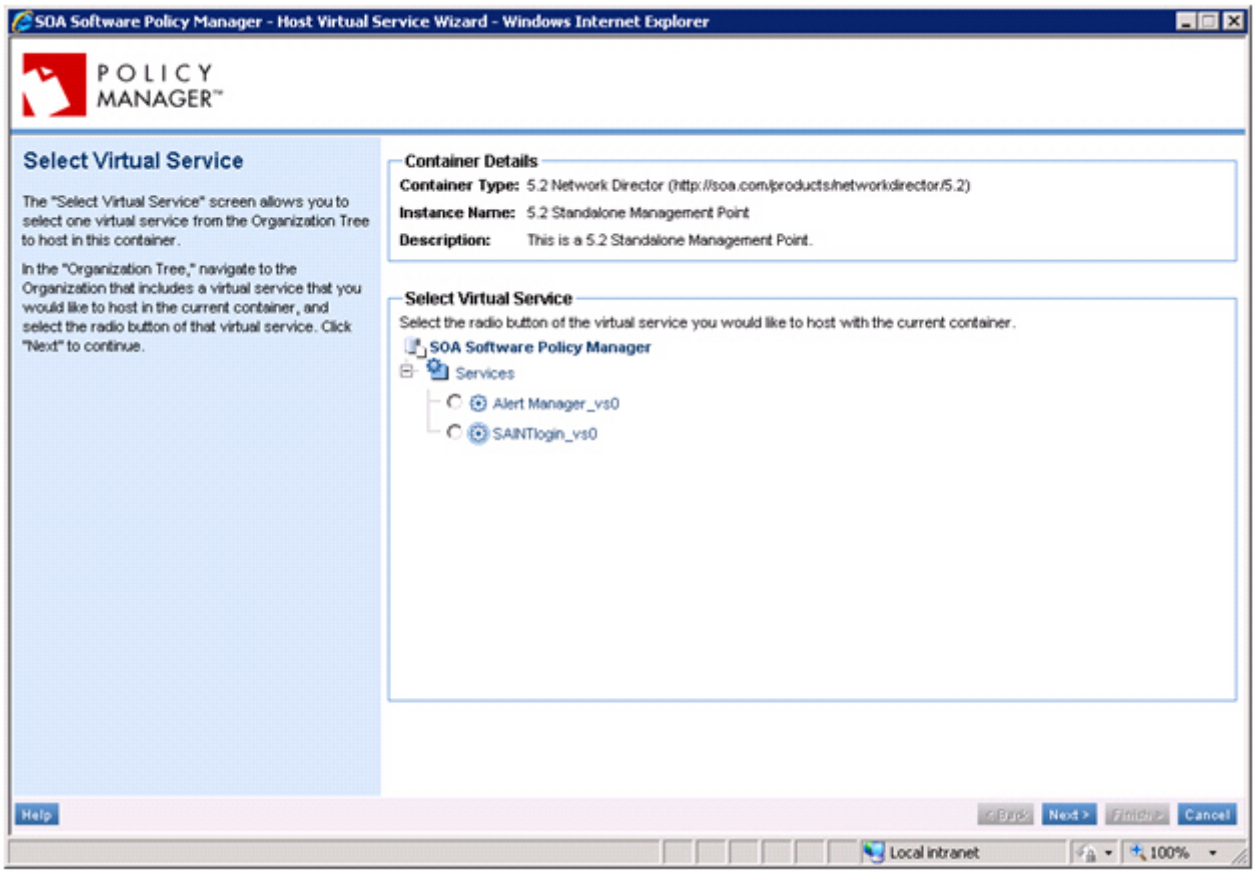


Figure 47: Host Virtual Service Wizard—*Select Virtual Service*

4 The first step in the hosting process is to select a virtual service from the Organization Tree. In the "Select Virtual Service" section navigate through the Organization Tree to an organization that contains a virtual service you would like to host.

5 Select the radio button of the virtual service you would like to host, and click "Next." The "Configure Service Hosting" screen displays. The "Configure Service Hosting" screen allows you to map a binding to a virtual service to the Access Points of the current Container. A virtual service can include one or more bindings. Each binding can include one or more Access Points.

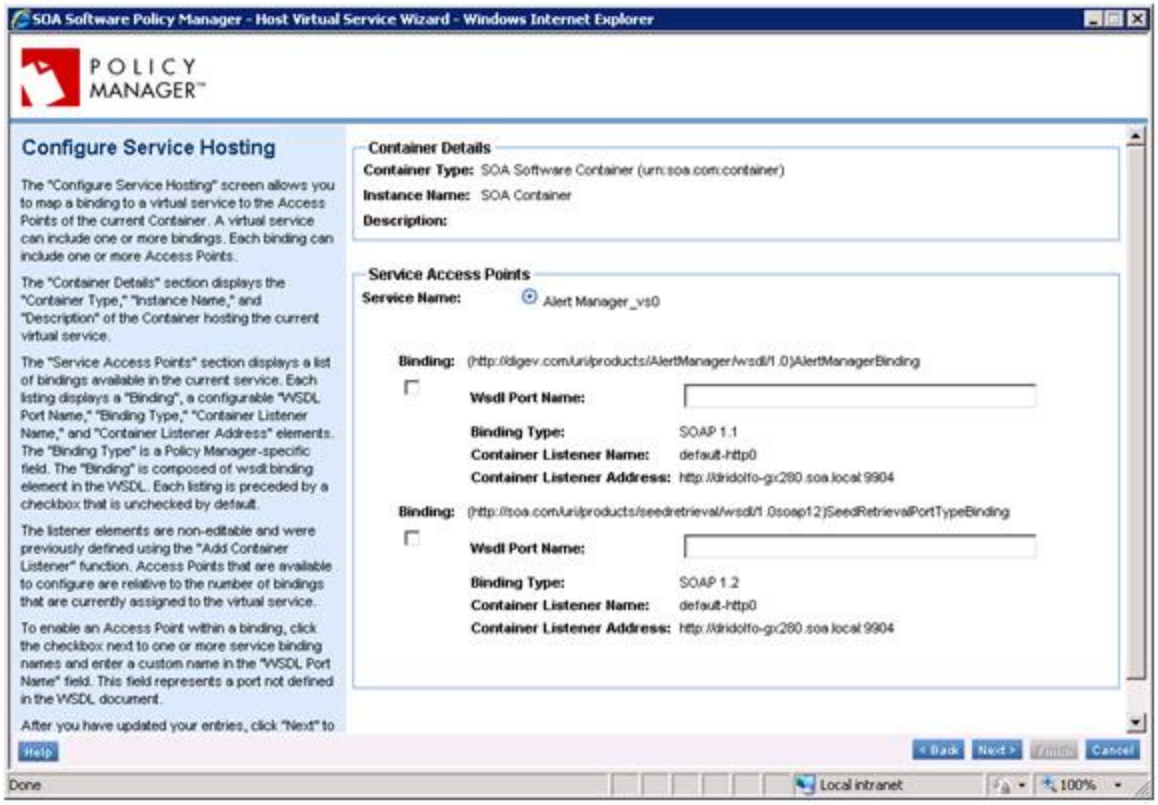


Figure 48: Host Virtual Service Wizard—*Configure Service Hosting*

The screen contains two sections:

- Container Details—Displays the "Container Type," "Instance Name," and "Description" of the Container hosting the current virtual service.
- Service Access Points—Displays a list of bindings available in the current service. Each listing displays a "Binding", a configurable "WSDL Port Name," "Binding Type," "Container Listener Name," and "Container Listener Address" elements. The "Binding Type" is a Policy Manager-specific field. The "Binding" is composed of wsdl:binding element in the WSDL. Each listing is preceded by a checkbox that is unchecked by default.

The listener elements are non-editable and were previously defined using the "Add Container Listener" function. Access Points that are available to configure are relative to the number of bindings that are currently assigned to the virtual service.

To enable an Access Point within a binding, click the checkbox next to one or more service binding names and enter a custom name in the "WSDL Port Name" field. This field represents a port not defined in the WSDL document.

> **Note**: Hosting information updates are added to the service Host Name/IP, Port, and container information to define a URL address for the service.

To specify SOAP 1.1 details:

1 The "Specify SOAP 1.1 Details" screen displays if the "Binding Type" associated with a hosted service is SOAP 1.2. The "Specify SOAP 1.1 Details" screen allows you to configure transport information for the current service Access Point. Transport configuration options are derived from the service binding.

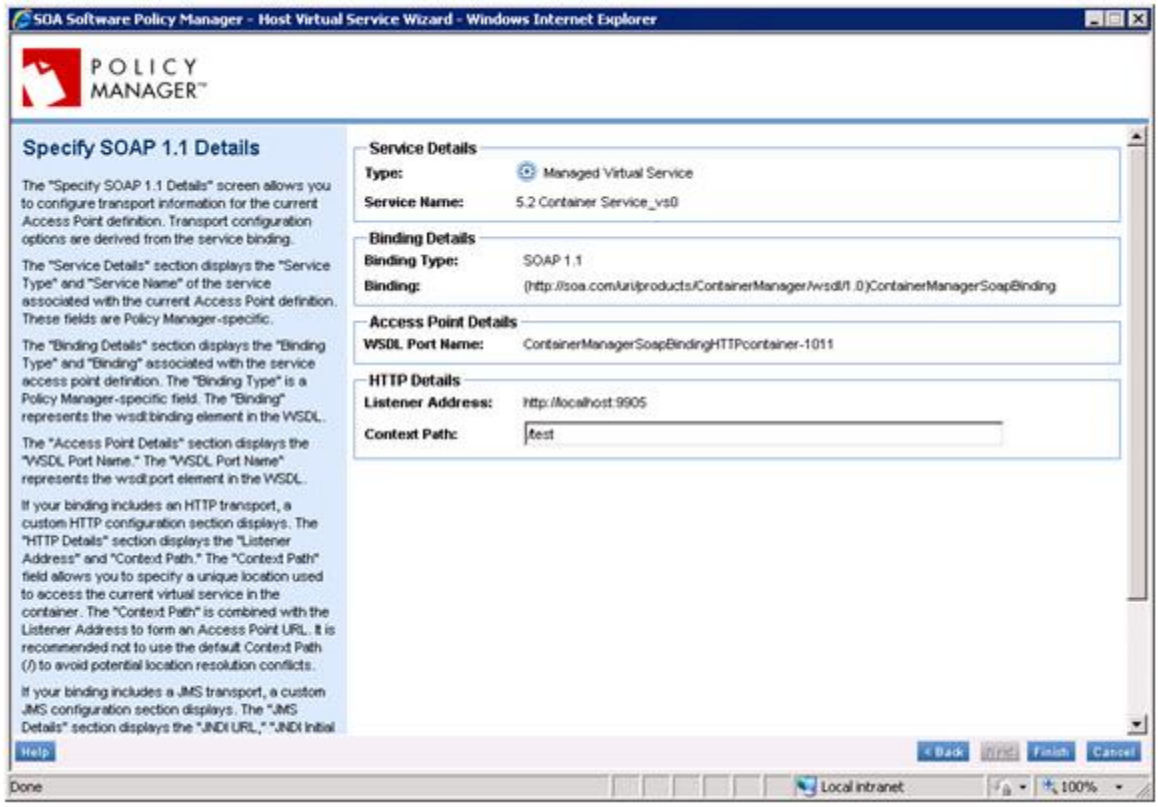


Figure 49: Host Virtual Service Wizard—*Specify SOAP 1.1 Details / HTTP*

Figure 50: Host Virtual Service Wizard—*Specify SOAP 1.1 Details / JMS*

The screen is organized as follows:

- Service Details—Displays the "Service Type" and "Service Name" of the service associated with the current Access Point definition. These fields are Policy Manager-specific.
- Binding Details—This section displays the "Binding Type" and "Binding" associated with the service access point definition. The "Binding Type" is a Policy Manager-specific field. The "Binding" represents the wsdl:binding element in the WSDL.
- Access Point Details—This section displays the "WSDL Port Name" and "Description." The "WSDL Port Name" represents the wsdl:port element in the WSDL. The "Description" represents the wsdl:documentation element in the WSDL. The "Description" field is optional.
- HTTP Details—If your binding includes an HTTP transport, a custom HTTP configuration section displays. The "HTTP Details" section displays the "Listener Address" and "Context Path." The "Context Path" field allows you to specify a unique location used to access the current virtual service in the container. The "Context Path" is combined with the Listener Address to form an Access Point URL. It is recommended not to use the default Context Path (/) to avoid potential location resolution conflicts.
- JMS Details—If your binding includes a JMS transport, a custom JMS configuration section displays. The "JMS Details" section displays the "JNDI URL," "JNDI Initial Context," and "JMS Factory Name" properties defined in your container JMS Listener configuration and a "JMS Queue" field that allows you to enter the name of the JMS Queue where a Queue object reference can be found in the JNDI server.

To specify SOAP 1.2 details:

1 The "Specify SOAP 1.2 Details" screen displays if the "Binding Type" associated with a hosted service is SOAP 1.2. The "Specify SOAP 1.1 Details" screen allows you to configure transport information for the current service Access Point. Transport configuration options are derived from the service binding.

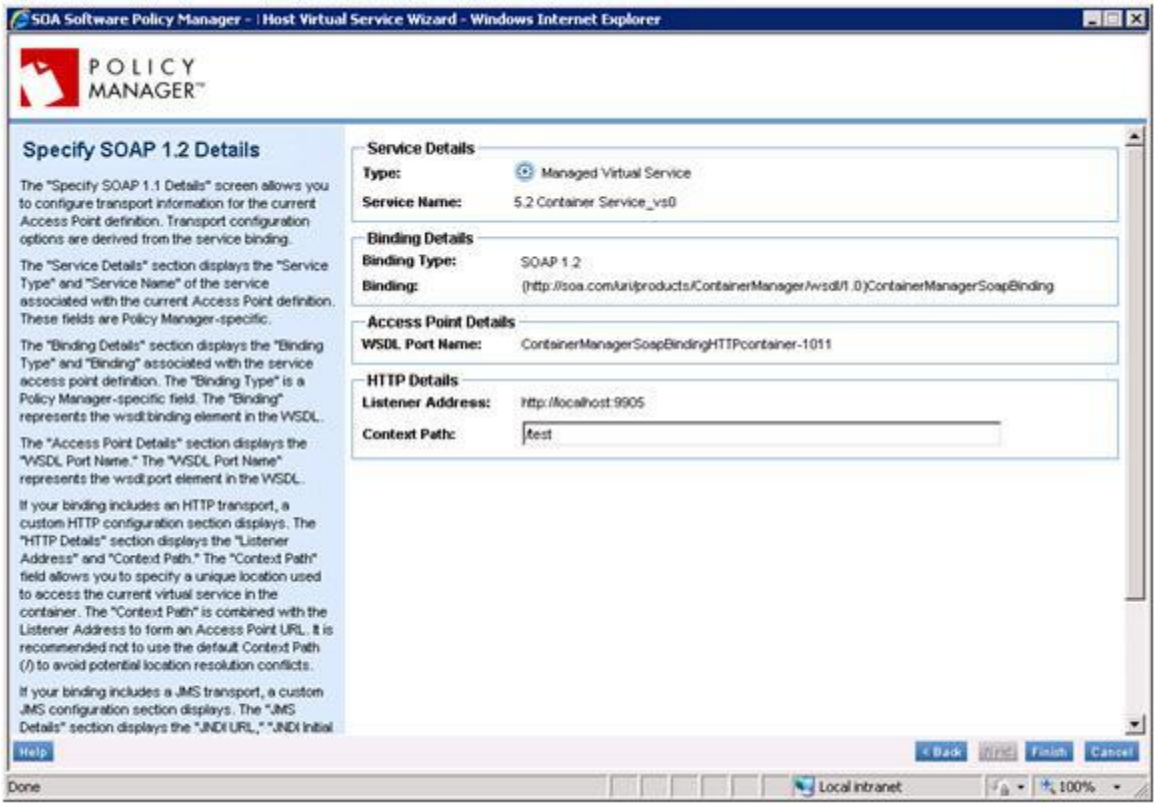


Figure 51: Host Virtual Service Wizard—*Specify SOAP 1.2 Details / HTTP*

Figure 52: Host Virtual Service Wizard—*Specify SOAP 1.2 Details / JMS*

The screen is organized as follows:

- Service Details—Displays the "Service Type" and "Service Name" of the service associated with the current Access Point definition. These fields are Policy Manager-specific.
- Binding Details—This section displays the "Binding Type" and "Binding" associated with the service access point definition. The "Binding Type" is a Policy Manager-specific field. The "Binding" represents the wsdl:binding element in the WSDL.
- Access Point Details—This section displays the "WSDL Port Name" and "Description." The "WSDL Port Name" represents the wsdl:port element in the WSDL. The "Description" represents the wsdl:documentation element in the WSDL. The "Description" field is optional.
- HTTP Details—If your binding includes an HTTP transport, a custom HTTP configuration section displays. The "HTTP Details" section displays the "Listener Address" and "Context Path." The "Context Path" field allows you to specify a unique location used to access the current virtual service in the container. The "Context Path" is combined with the Listener Address to form an Access Point URL. It is recommended not to use the default Context Path (/) to avoid potential location resolution conflicts.
- JMS Details—If your binding includes a JMS transport, a custom JMS configuration section displays. The "JMS Details" section displays the "JNDI URL," "JNDI Initial Context," and "JMS Factory Name" properties defined in your container JMS Listener configuration and a "JMS Queue" field that allows you to enter the name of the JMS Queue where a Queue object reference can be found in the JNDI server.

To specify http details:

1 The "Specify HTTP Details" screen allows you to configure transport information for the current Access Point definition. Transport configuration options are derived from the service binding.

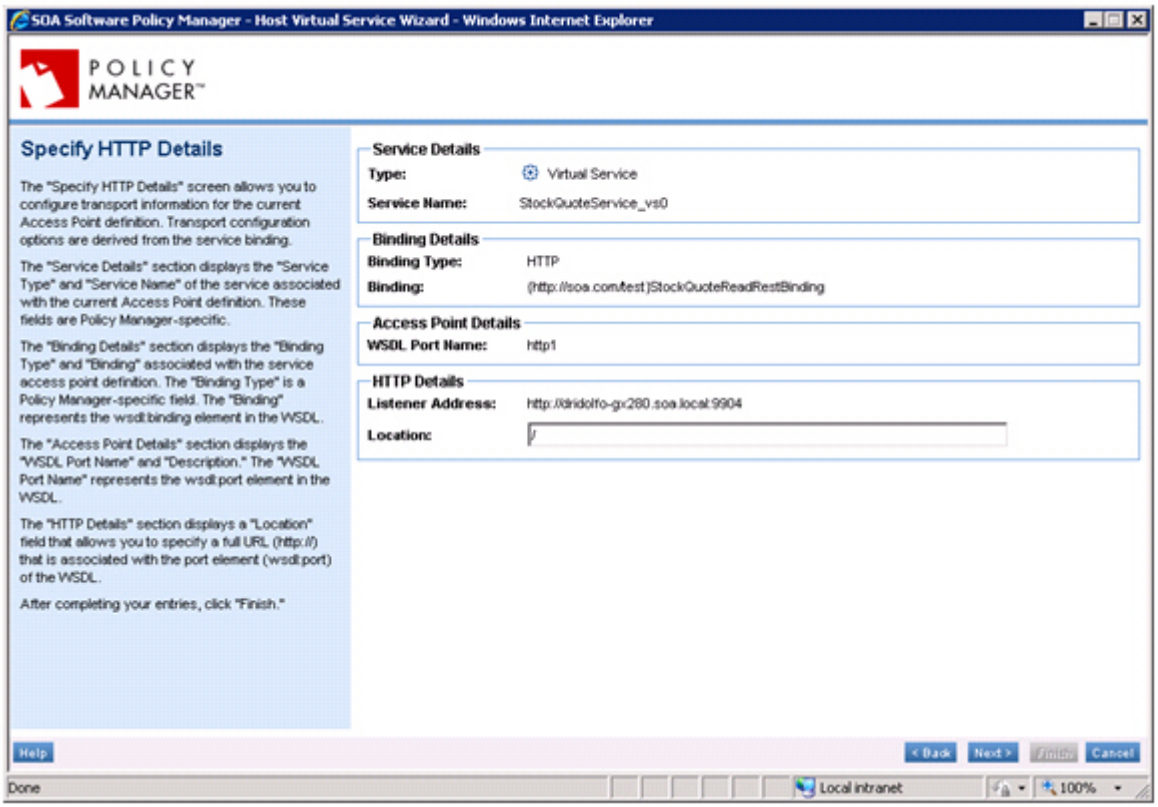


Figure 53: Host Virtual Service Wizard—Specify *HTTP Details*

The screen is organized as follows:

- Service Details—Displays the "Service Type" and "Service Name" of the service associated with the current Access Point definition. These fields are Policy Manager-specific.
- Binding Details—Displays the "Binding Type" and "Binding" associated with the service access point definition. The "Binding Type" is a Policy Manager-specific field. The "Binding" represents the wsdl:binding element in the WSDL.
- Access Point Details—Displays the "WSDL Port Name." The "WSDL Port Name" represents the wsdl:port element in the WSDL.
- HTTP Details—Displays the "Listener Address" and the "Location" field that allows you to specify a full URL (http://) that is associated with the port element (wsdl:port) of the WSDL.

To specify xml details:

1 The "Specify XML Details" screen allows you to configure a custom service access point definition.

The screen is organized as follows:

- Service Details—Displays the "Service Type" and "Service Name" of the service associated with the current Access Point definition.
- Binding Details—Displays the "Binding Type" and "Binding" selected on the "Specify Access Point Details" screen.
- Access Point Details—Displays the "WSDL Port Name" defined on the "Specify Access Point Details" screen.
- XML Details—A text box that allows you to enter content of the XML Access Point extensions. Typical access point elements include address information that is based on the type of binding (e.g., soap12:address, etc.). Note that the wsdl:port element is system generated and should not be specified as part of your access point definition. When an access point definition is saved, the specified XML extensions and content from the "Specify Access Point Details" screen are added to the wsdl:port element.
- Check Syntax—A button allows you to check the XML syntax to verify it is valid, well formed, and complies with the XML 1.0 requirements.
- Check Schema—A button allows you to verify that the XML content is valid according to the XML schemas referenced. The schemas are assumed to be reachable by the Namespace, Schema location, or existence in the "Configure > Registry > Schema" section of the "Management Console."

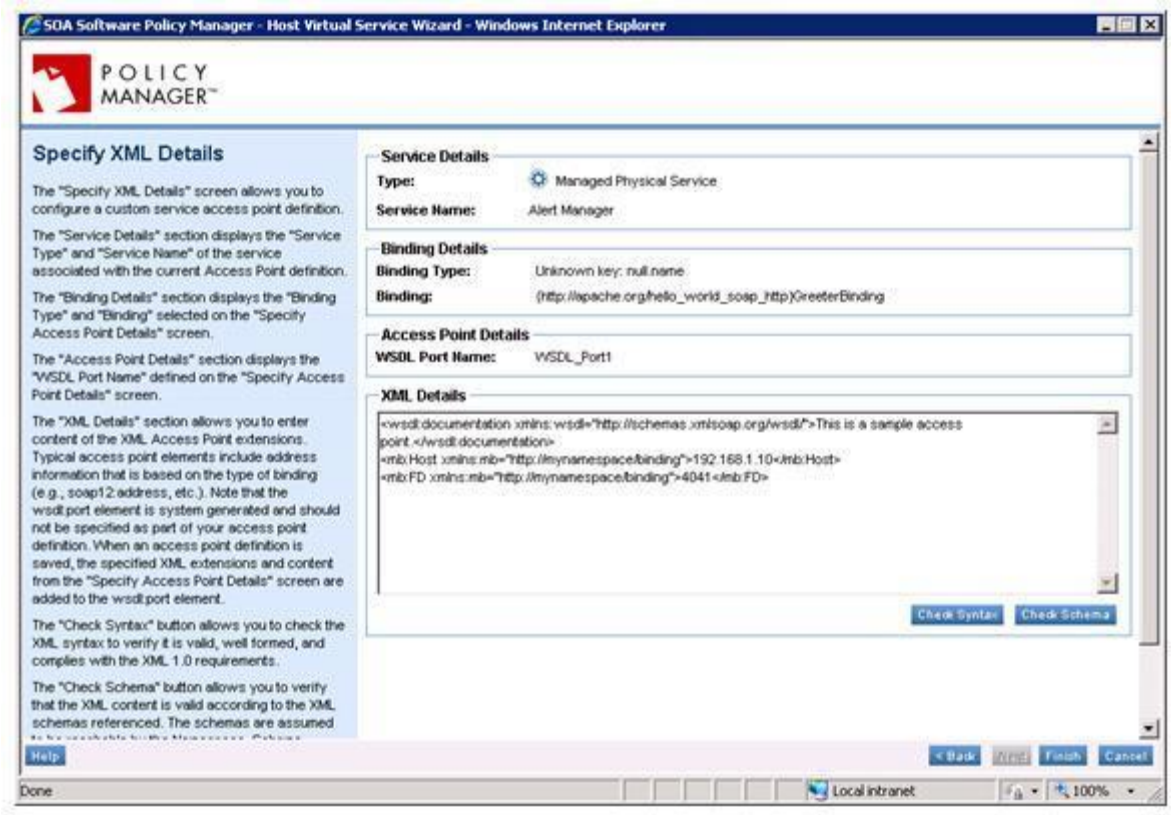


Figure 54: Host Virtual Service Wizard—*Specify XML Details*

To specify pox details:

1 The "Specify POX Details" screen allows you to configure transport information for the current Access Point definition. Transport configuration options are derived from the service binding.

The screen is organized as follows:

- Service Details—Displays the "Service Type" and "Service Name" of the service associated with the current Access Point definition. These fields are Policy Manager-specific.
- Binding Details—Displays the "Binding Type" and "Binding" associated with the service access point definition. The "Binding Type" is a Policy Manager-specific field. The "Binding" represents the wsdl:binding element in the WSDL.
- Access Point Details—Displays the "WSDL Port Name" and "Description." The "WSDL Port Name" represents the wsdl:port element in the WSDL.
- HTTP Details—If your binding includes an HTTP transport a custom HTTP configuration section displays. The "HTTP Details" section allows you to enter Host Name or IP, Port, and Path for the URL you would like assigned to the Access Point. In the "URL" text box enter the Host Name or IP, Port, and Path for the HTTP or HTTPS protocol type. Then, click the radio button of the "Protocol Version" you would like assigned to the Access Point. Options include HTTP 1.1 (default), HTTP 1.0, and Unspecified. The "Unspecified" option uses the endpoint properties protocol version if it is set, or defaults to the HTTP 1.1 protocol version if it is not set.
- JMS Details—If your binding includes a JMS transport, a custom JMS configuration section displays. The "JMS Details" section displays the "JNDI URL," "JNDI Initial Context," and "JMS Factory Name" properties defined in your container JMS Listener configuration and a "JMS Queue" field that allows you to enter the name of the JMS Queue where a Queue object reference can be found in the JNDI server.

The "Add Connection Property" link allows you to add JMS Connection Properties which are user-defined additional properties that allow you to specify the messaging engine that an application will connect to. Only "String" type message properties are supported. Connection Properties are configured by entering a "Property Name" and "Property Value." Entering a Connection Property is optional. Click the "Add Connection Property" link for each Connection Property you would like to add. To remove a Connection Property, click the "Remove" link.

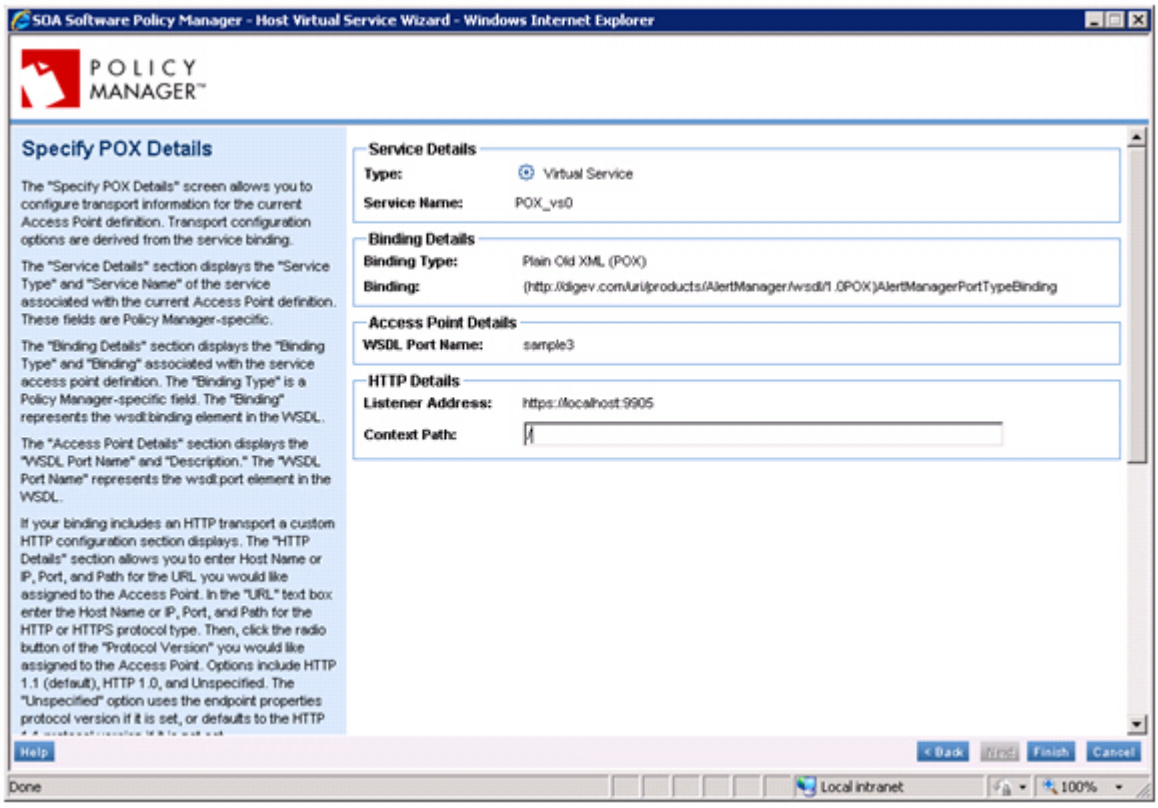


Figure 55: Host Virtual Services Wizard—Specify Pox Details (HTTP Details with HTTPS Listener)

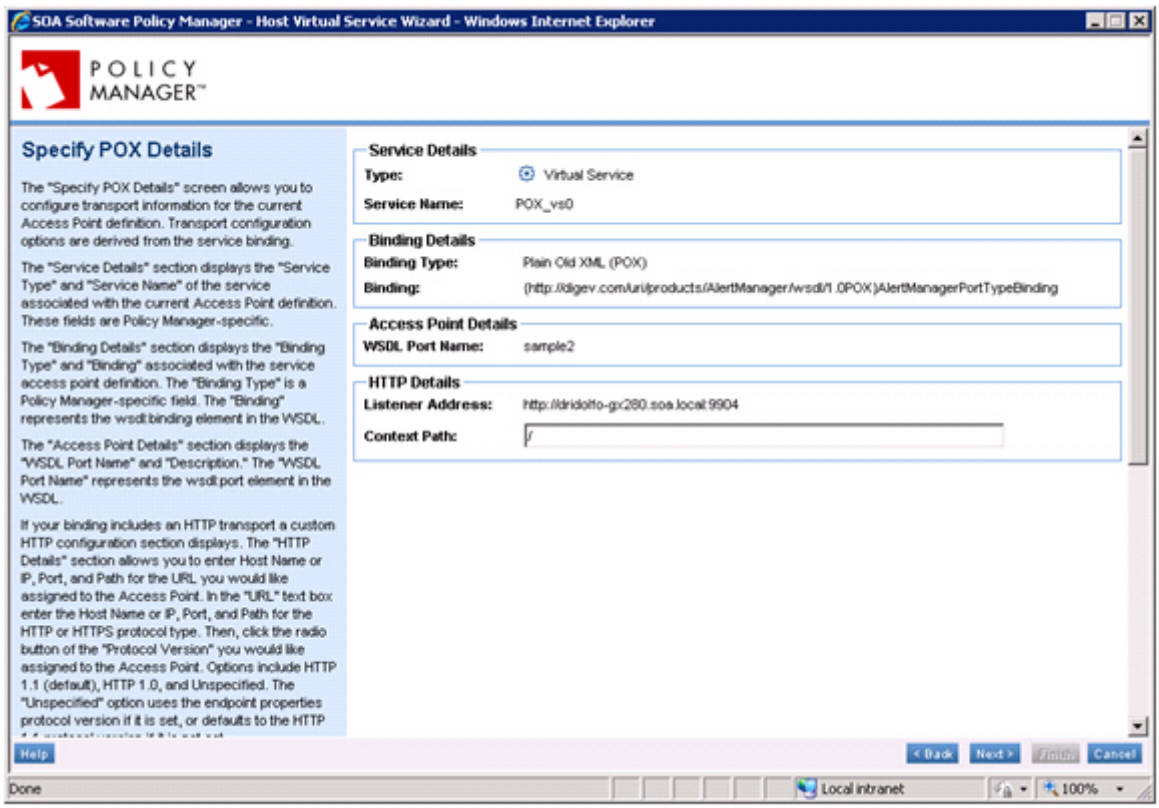


Figure 56: Host Virtual Service Wizard—Specify POX Details (HTTP Details with HTTP Listener)

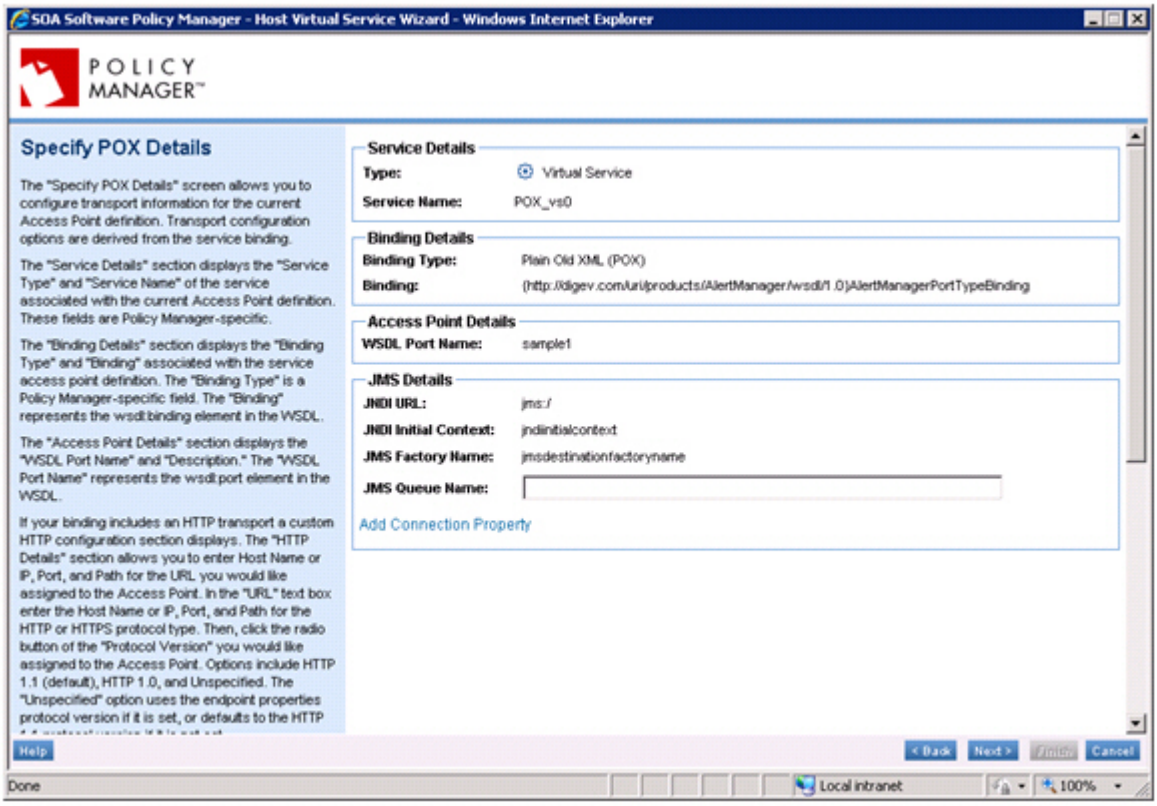


Figure 57: Host Virtual Service Wizard—Specify POX Details (with JMS Details)

2 After you have updated your entries, click "Next" to continue.

3 Enter your hosting information into the applicable fields and click "Finish" to commit your changes. The virtual service and hosting information is added to the virtual service definition and displays on the "Hosted Services" screen.

Attaching Policies

The Policy Attachments Portlet is organized into Compliance, Operational, and QoS sections that list associated policies attached to a service respectively. The "Policy Attachments Portlet" is available in each organization, and in six different elements of a service (i.e., service details, operations, bindings, access points, and contracts). Policies types that apply to each service element vary, but the user interface design is similar in each section. The screen is organized as follows:

- Manage Link

Next to each policy category name is a "Manage" link that launches the "Policy Attachments" pop-up associated with the selected policy. The name of the "Policy Attachments" screen as well as the functionality varies based on the specific section you are in (i.e., organization or service element).

- Attached Policies

Attached policies are displayed in a summary list below the associated policy category name. Each policy line item includes the "Policy Name" and "Description." Policies that are referenced include a (from organization) or (from service) identifier

Managing Schemas

When importing WSDL, all referenced schemas that describe the messages used by the services are extracted, indexed and stored in the Metadata Repository. This allows the Workbench to reuse schema between services, allowing users to see how many service references a particular schema has and therefore determine the impact of any change to a particular schema. To preserve the integrity of the metadata repository, each schema should have a unique and sensible namespace to allow for their successful correlation and discovery.

Schema information can be accessed in the "Configure > Registry > Schemas" section of the Management Console. The following key activities can be performed.

How Do I Add New Schemas?

The "Add Schema Wizard" provides functionality that allows you to import schemas from `.xsd` files into the Policy Manager data repository. These schemas can then be reused to model web services.

Add Binding Wizard

- Launch Add Schema Wizard
- Select Schema Import Option
- View Completion Summary

To launch add schema wizard:

1. Navigate to the "Configure" tab using the following navigation path: **Configure > Registry**. Click the "Schemas" tab. The "Schemas Summary" screen displays.

To select a schema import option:

1 To add a schema, click "Add Schema." The "Add Schema Wizard" launches and displays the "Select Schema Import Option" screen.

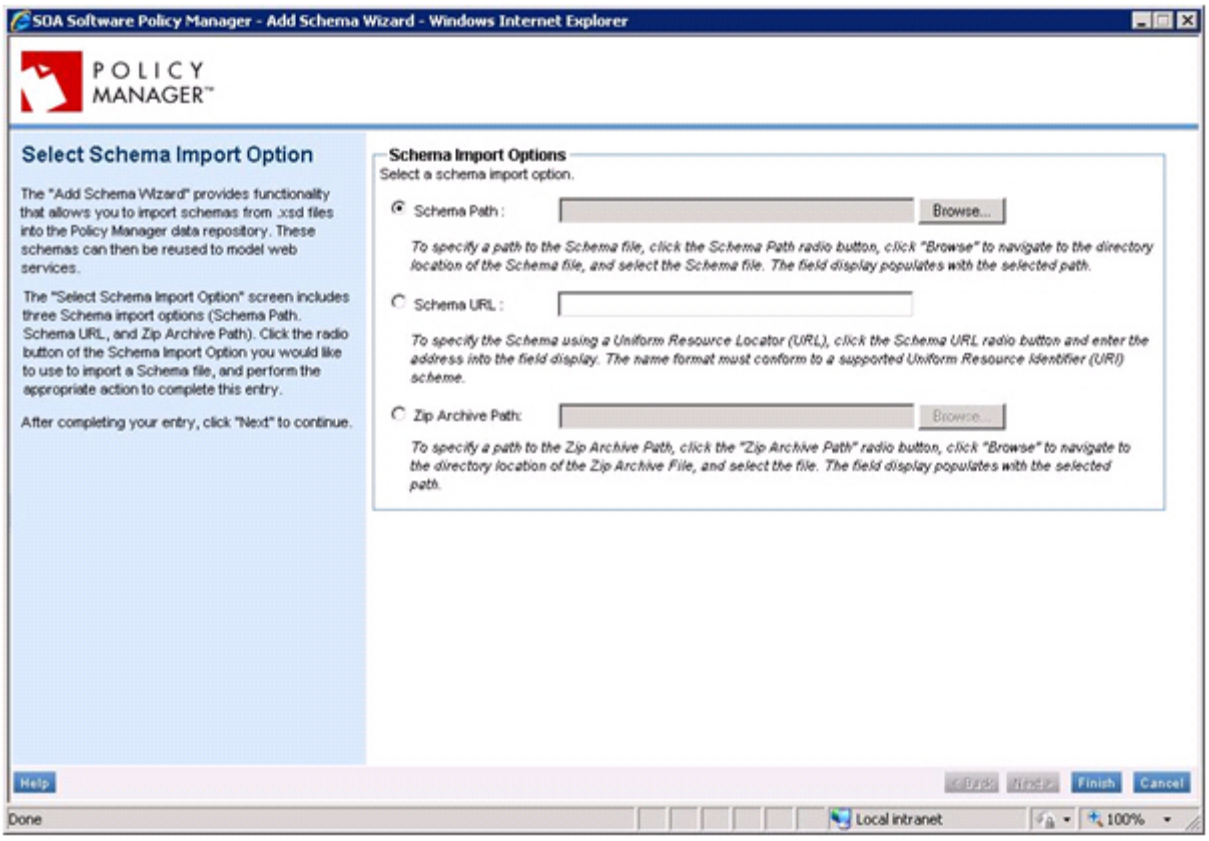


Figure 58: Add Schema Wizard—Select Schema Import Option

Select one of the following schema import options:

- Schema Path

  To specify a path to the Schema file, click the "Schema Path" radio button, click "Browse" to navigate to the directory location of the Schema file, and select the Schema file. The field display populates with the selected path.

- Schema URL

  To specify the Schema using a Uniform Resource Locator (URL), click the "Schema URL" radio button and enter the address into the field display. The name format must conform to a supported Uniform Resource Identifier (URI) scheme.

- Zip Archive Path

  To specify a path to the Zip Archive Path, click the "Zip Archive Path" radio button, click "Browse" to navigate to the directory location of the Zip Archive File, and select the file. The field display populates with the selected path.

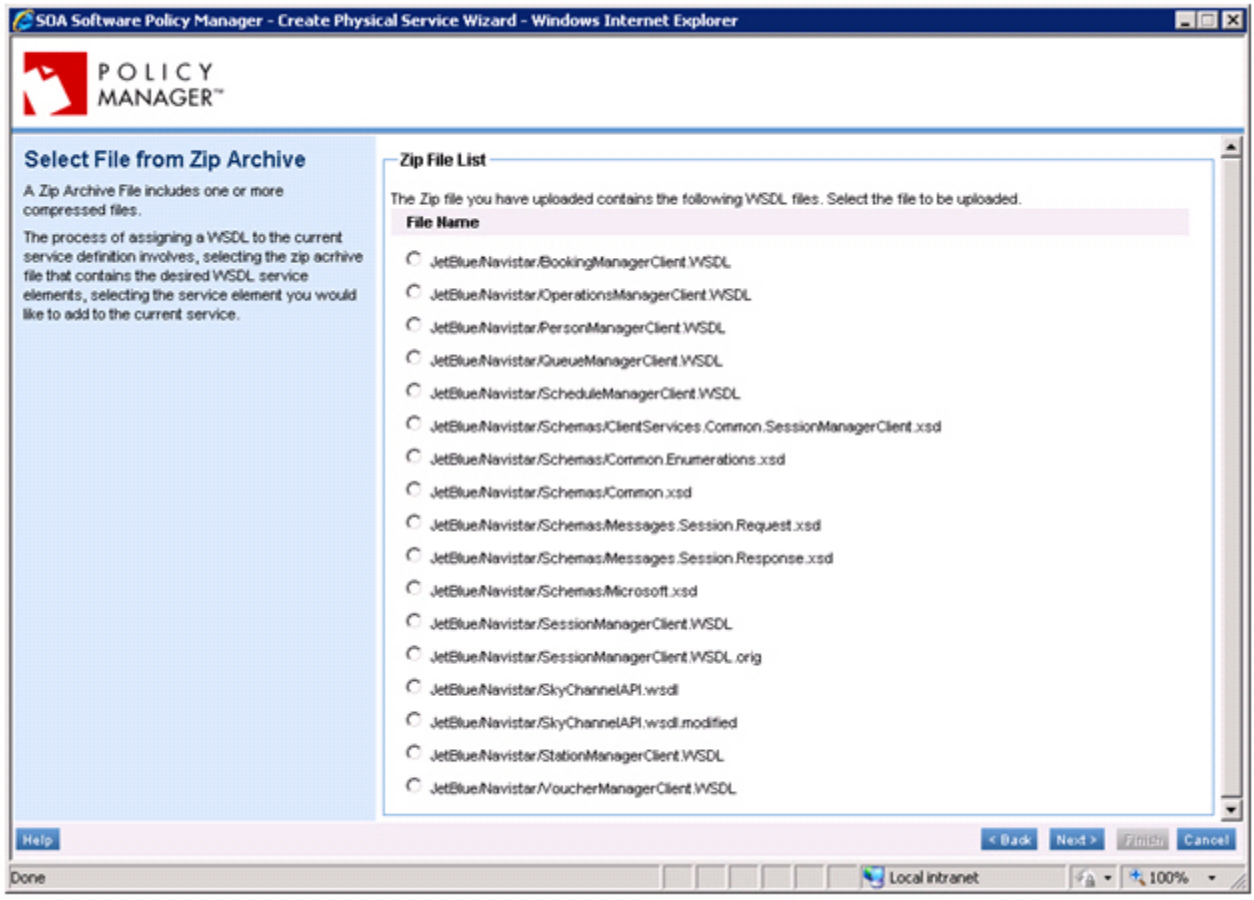


Figure 59: Add Schema Wizard—Select File from Zip Archive

To view completion summary:

1 After selecting and configuring your import option, click "Finish." The "Completion Summary" screen displays. Review the summary information, and then click "Close" to exit.

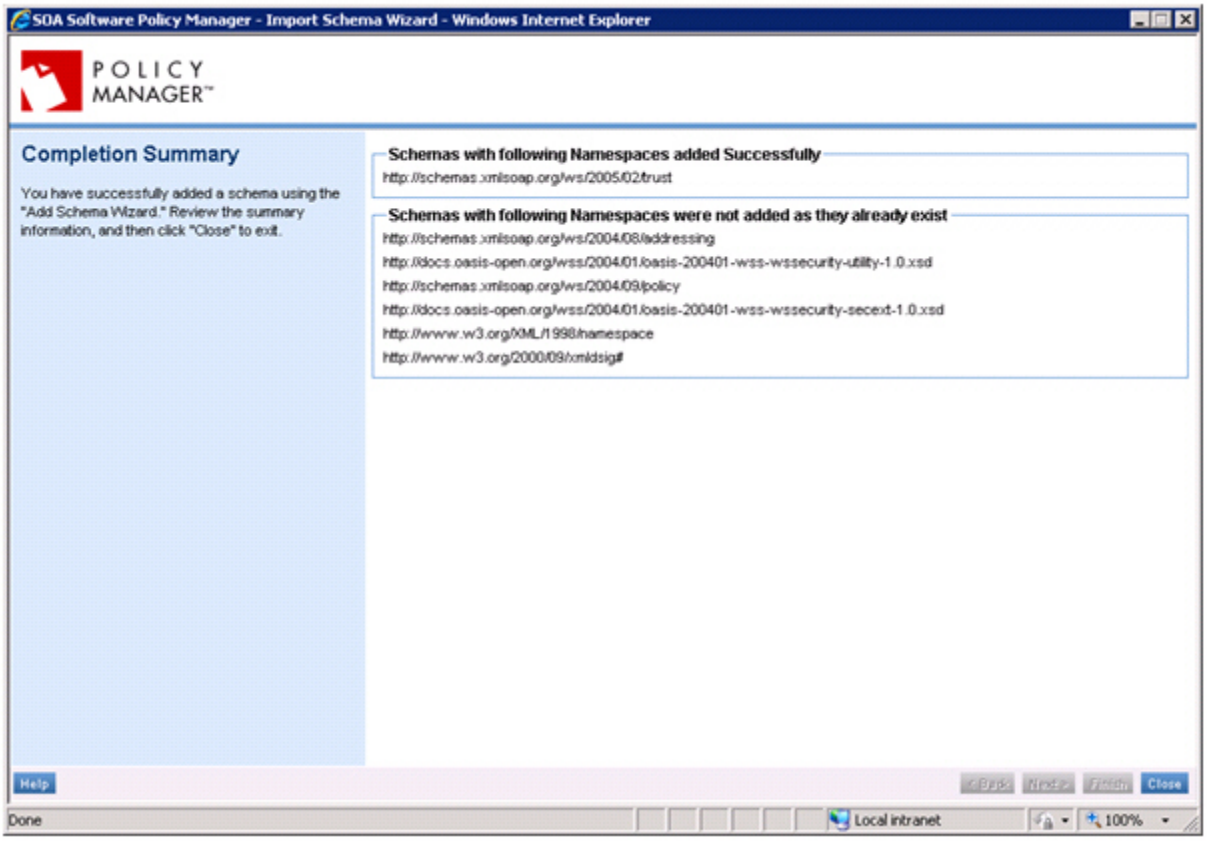


Figure 60: Add Schema Wizard—*Completion Summary*

Managing Interfaces

How do I Import an Interface into Policy Manager?

The "Add Interface Wizard" provides a series of configuration options that allow you to define an "Interface." An "Interface" defines the behavior of a service based on a set of defined operations that it implements. An interface can be imported, or modelled using existing schemas or interfaces. Adding an Interface to the Policy Manager data repository involves defining an Interface QName which is composed of a "Namespace URI" and "Localpart," and selecting the Interface elements and associated Operations you would like to include in the Interface definition.

- Import from WSDL

  This option allows you to import interfaces from a WSDL file into the Policy Manager data repository. These interfaces can then be reused in full or in part to model web services.

- Model using existing schemas

  This option allows you to model a new interface by utilizing schemas and associated operations that are already registered in Policy Manager.

- Model using existing interface

  This option allows you to model a new interface by utilizing interfaces and associated operations that are already registered in Policy Manager.

How do I Model an Interface using Existing Schemas or Interfaces?

The "Add Binding Wizard" provides functionality that allows you to add a new binding to the Policy Manager repository. A binding is reference to an external framework (i.e., interface) that defines how the WSDL user will reach the implementation of services. This reference specifies the protocol and data format to be used in the transmitting message defined by the associated interface. Each binding technique is specified in the WSDL and points to the server that has access to the actual implementation of your Web service.

A variety of different binding configuration options are provided to meet your requirements based on the selected "Binding Type" (HTTP, Plain Old XML (POX), SOAP 1.1, SOAP 1.2, and XML).

Add Binding Wizard

- Launch Add Binding Wizard
- Select Interface
- Specify Binding Details
- Configure HTTP Binding Properties
- Configure POX Binding Properties
- Configure SOAP 1.1 Binding Properties
- Configure SOAP 1.2 Binding Properties
- Configure XML Binding Properties
- Complete Binding Configuration

To launch add binding wizard:

1. Navigate to the "Configure" tab using the following navigation path: **Configure > Registry**. Click the "Bindings" tab. The "Bindings Summary" screen displays.
2. To add a binding, click "Add Binding." The "Add Binding Wizard" launches and displays the "Select Interface" screen.

To select an interface:

3 The "Select Interface" screen provides functionality that allows you to add an interface to the current binding definition. It is organized as follows:

- Search Interfaces—A text box that allows you to search for any part of an "Interface QName" and/or "Localpart." Interface QNames that match your search criteria will display in the "Results" section as part of the "Interface" drop-down list box.
- Results—A drop-down list box that displays the interfaces that represent the results of your Interface QName search. If an interface search is not performed, the default content of the drop-down list box equals all interfaces defined in the Policy Manager data repository.

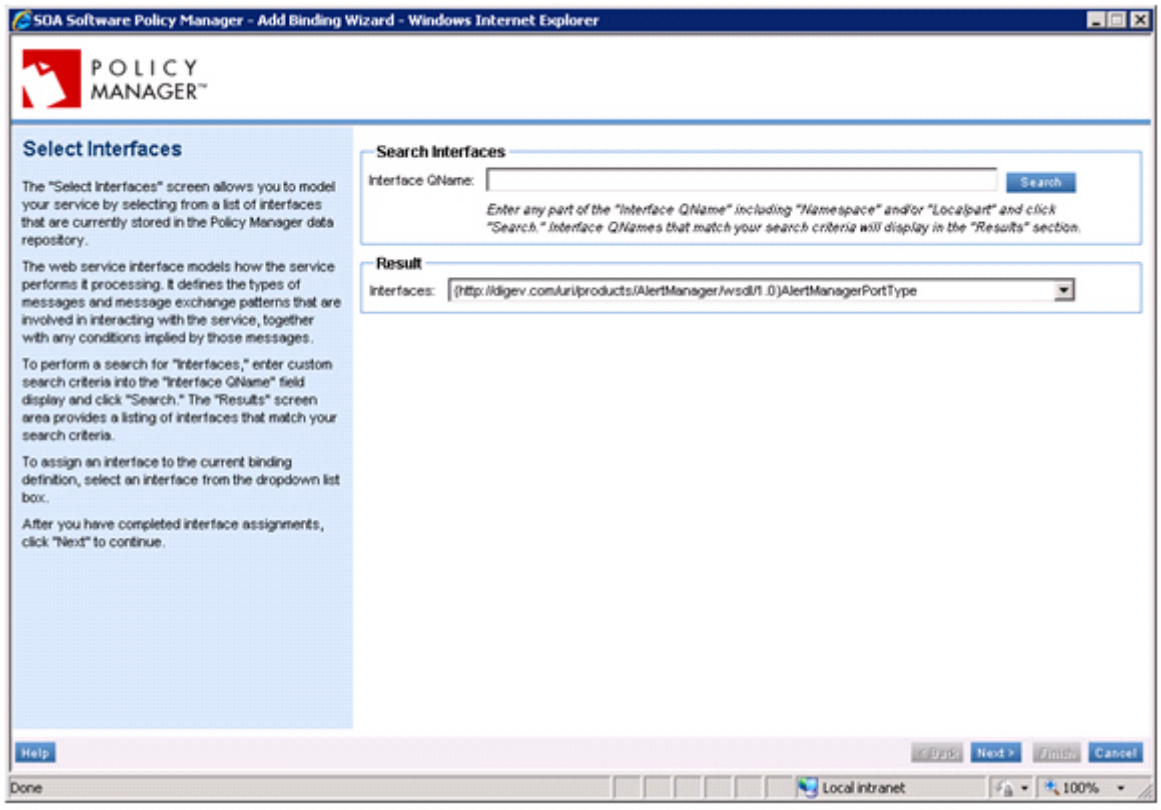


Figure 61: Add Binding Wizard—*Select Interface*

After you configure your "Select Interface" entries, click "Next" to continue. The "Specify Binding Details" screen displays.

To specify binding details:

4 The "Specify Binding Details" screen allows you to add "Namespace URI" and "Localpart" elements to your binding definition. The screen is organized as follows:

Binding Details

Namespace URI—A text box that is initially system-generated and derivative of the elements of the selected interface. You can use the system default binding or enter a custom binding "Namespace URI," and "Localpart," and optional "Description."

Binding Type—A drop-down list box that allows you to select the protocol that will be applied to the Binding Type you will be defining on subsequent binding configuration screen. Supported options include HTTP, SOAP 1.1, and SOAP 1.2.

After you configure your "Binding Details" entries, click "Next" continue. The binding configuration screen for the selected "Binding Type" displays.

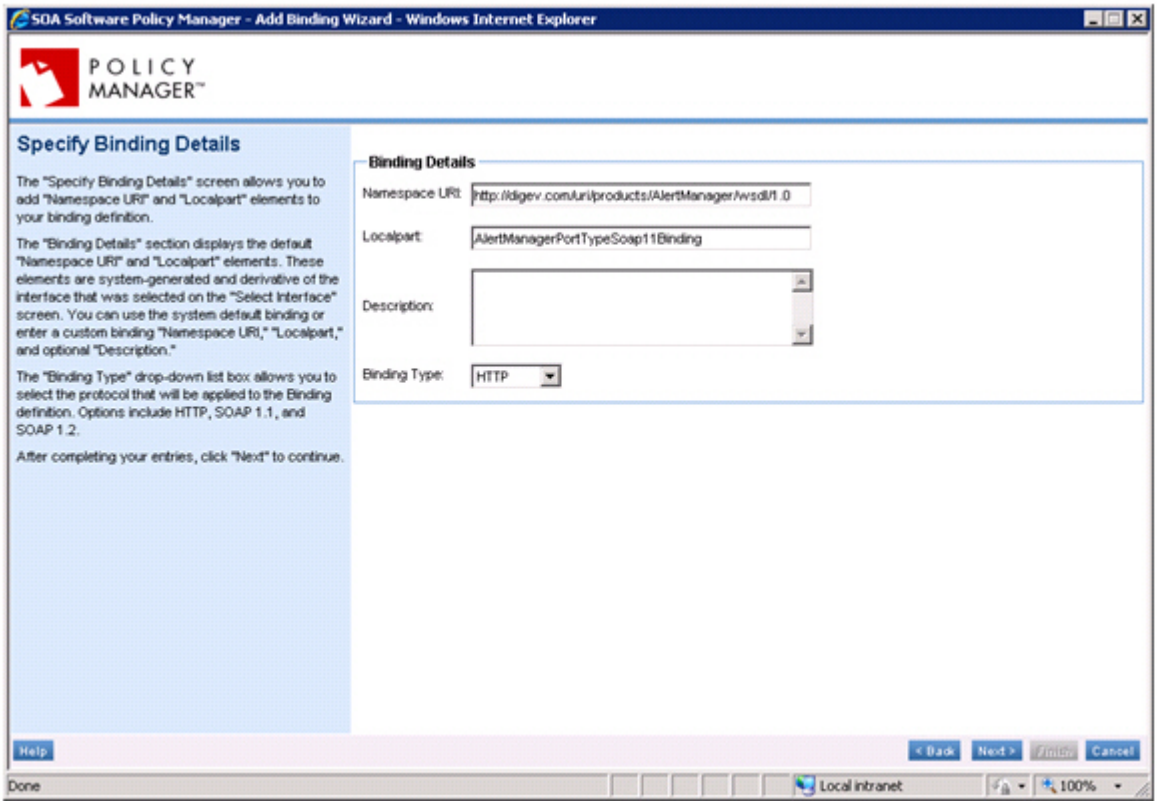


Figure 62: Add Binding Wizard—*Specify Binding Details*

To configure http binding properties:

5 The "HTTP Binding Details" screen provides functionality that allows you to define the HTTP method and URI pattern for your REST operation. The screen is organized as follows:

- Binding Details

Displays the "Interface QName," "Binding QName," and "Binding Type" for the current binding definition.

Binding Properties

Binding Properties include properties that apply to the WSDL operation within an HTTP binding.

- The "Method" is a dropdown list-box that allows you to map to the HTTP method that must be used at runtime when formulating an HTTP request message. Options include GET, POST, PUT, and DELETE.
- The "URI Syntax" is a text field that allows you to specify a location attribute that can be used to map portions of an HTTP request URI to portions of a WSDL input message. The specified syntax

must match the inbound URI pattern. For example, if the HTTP URL looks similar to http://endpoint/context/quotes/xyz where xyz is a variable, then the URI syntax would be /quotes/{var}. The URI can contain multiple variables denoted by {}. This field is optional.

- The "Serialization Type" dropdown list box provides "Pre-defined" or "Custom XML" content types that support different message processing requirements for Input, Output, and Fault messages. The "Pre-defined" content type includes "Form URL Encoded," XML-based," and "JSON" categories. Note that the "Form URL Encoded" category applies to the "Input Serialization" option only. The "Custom XML" content type provides a text box for entering custom XML that represents your message processing requirements.
- Input Serialization Type—This option is used to describe the content type of the Request Message. The default Input Serialization setting is application/x-www-form-urlencoded. A value of application/x-www-form-urlencoded assumes one of two things. 1) If the request message is a GET or DELETE, the query string contains items that are appended to the resulting XML message. 2) If the request message is a PUT or POST, the body contains a URL encoded string whose elements are appended to the resulting XML message. A value of an XML based content type assumes that the body contains the whole XML message.
- Output Serialization Type—This option is used to describe the content type of the response message when it is not a fault. The Container uses the Output Serialization to correctly set the content type of the response sent back to the consumer when the response is not a fault. The default content type is "text/XML."
- Fault Serialization Type—This option is used to describe the content type of the response message when it is a fault. The Container uses the Fault Serialization to correctly set the content type of the response sent back to the consumer when the response is a fault. The default content type is "text/XML."

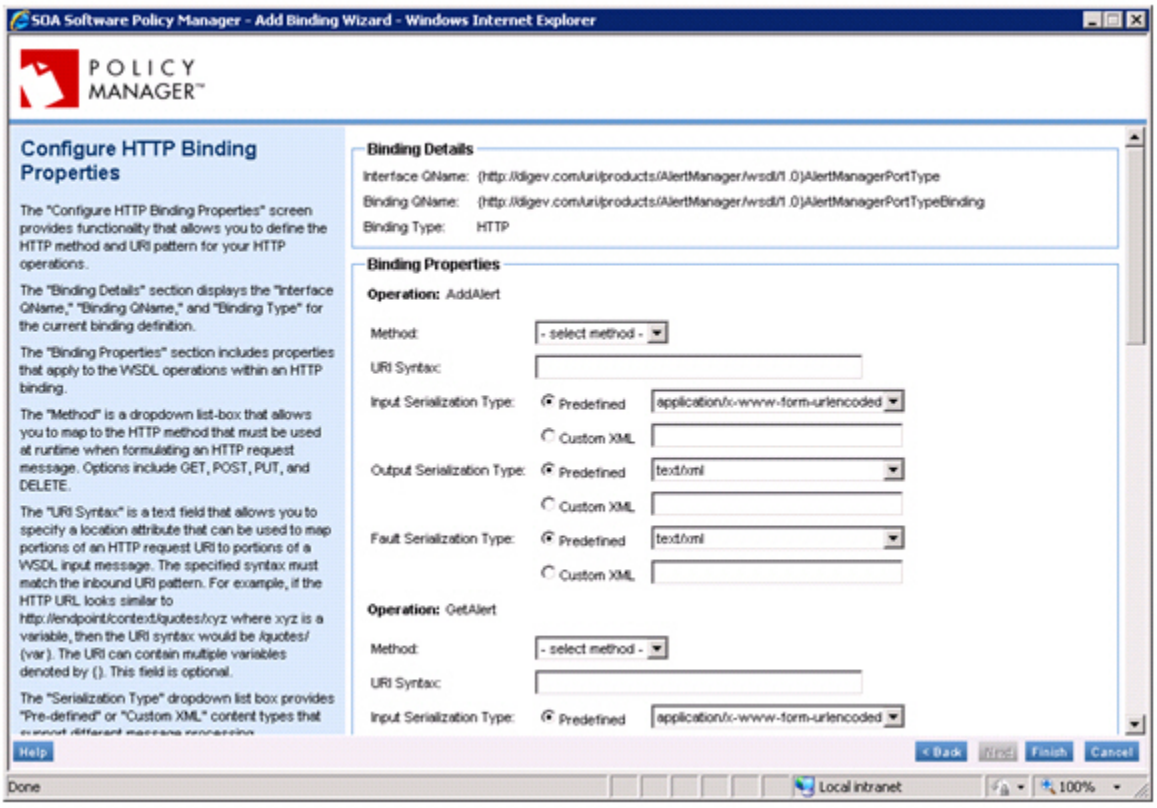


Figure 63: Add Binding Wizard—*HTTP Binding Details*

To configure pox binding properties

1 The "Configure POX Binding Properties" screen provides support for defining a Plain Old XML (POX) binding and provides functionality that allows you to define the Transport URI. The screen is organized as follows:

- The "Binding Details" section displays the "Interface QName," "Binding QName," and "Binding Type" for the current binding definition.
- The "Binding Properties" section allows you to select a binding "Transport" (HTTP, JMS, Other) and "Transport URI" (JMS, Other).
- Each POX binding requires a "Transport URI" which is an address that points to the message transport mechanism used to communicate information required to deliver a message appropriately to an endpoint. The "Transport URI" option displays for "JMS" and "Other" options. JMS supports four "Transport URI" options (WebSphere, WebLogic, WSIF, and Tibco) that are selectable from a drop-down list box. The "Other" transport option displays a "Transport URI" text box that allows you to enter a custom Transport URI (e.g., TCP). Note: The Transport URI for HTTP is a known value and is processed by Policy Manager.

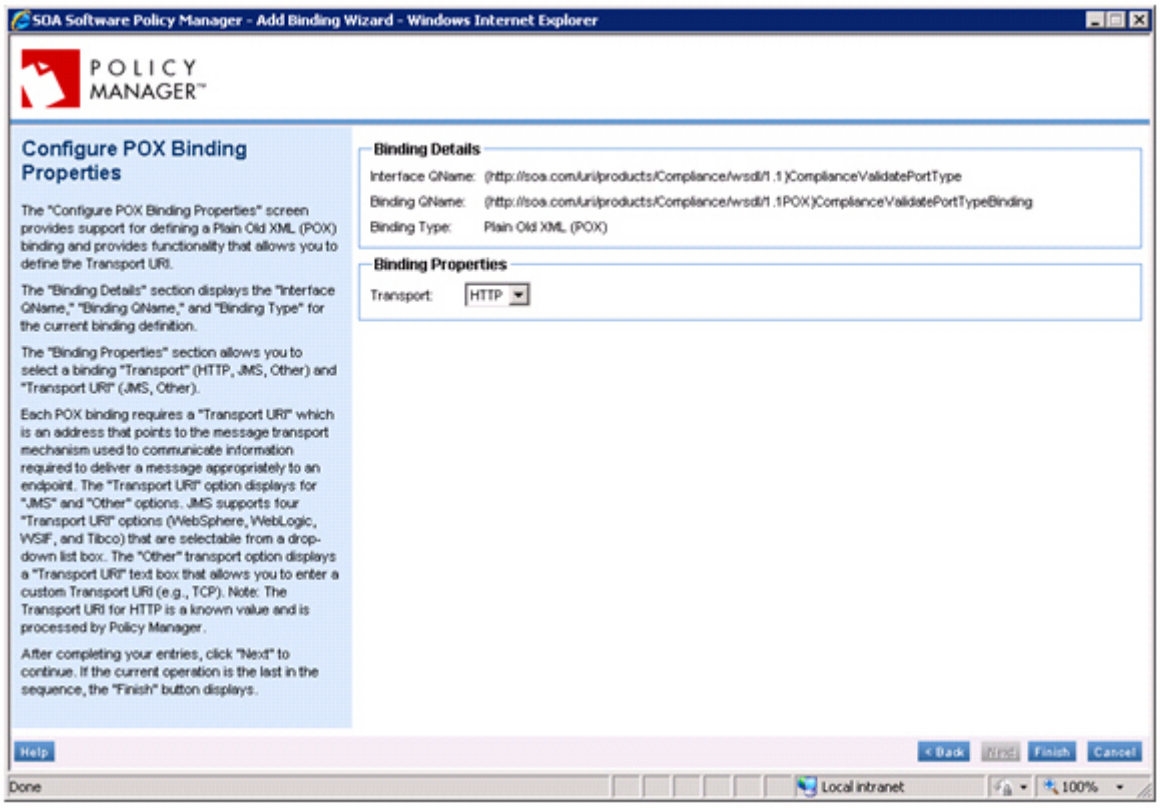


Figure 64: Add Binding Wizard—Configure POX Binding Properties

To configure soap 1.1 binding properties:

1 The "Configure SOAP 1.1 Binding Properties" screen provides options for configuring the properties of the current binding definition including, Style, Transport, Transport URI, and SOAP Action. The screen is organized as follows:

Binding Details

This section provides a summary of the Interface QName, Binding QName, and Binding Type elements of the current binding definition.

Binding Properties

This section allows you to pick a specific binding "Style" (Doc/Literal, RPC Literal, and RPC Encoded), Transport (HTTP/HTTPS), Transport URI (JMS/Other), and SOAP Action.

- SOAP Actions
    - No SOAP Action required for all operations—*A radio button option that is used if you do not want to use a SOAP action to identify the operation. If selected, the system generates an empty string for the SOAP action.*
    - Treat each operation name in interface as a SOAP Action—*A radio button option that is used if you would like to identify the operation of a service by treating each operation name as a SOAP Action.*
    - Configure SOAP Action by adding a prefix value to operation name—*A radio button option that is used if you would like to customize the SOAP Action name for each*

operation with a unique identifier for the service. This prefix generally matches the targetNamespace of the WSDL.

- Namespace

A text field display that is used to specify the Namespace of the operation element defined in the request. This option supports RPC/Literal or "RPC/Encoded" styles only.

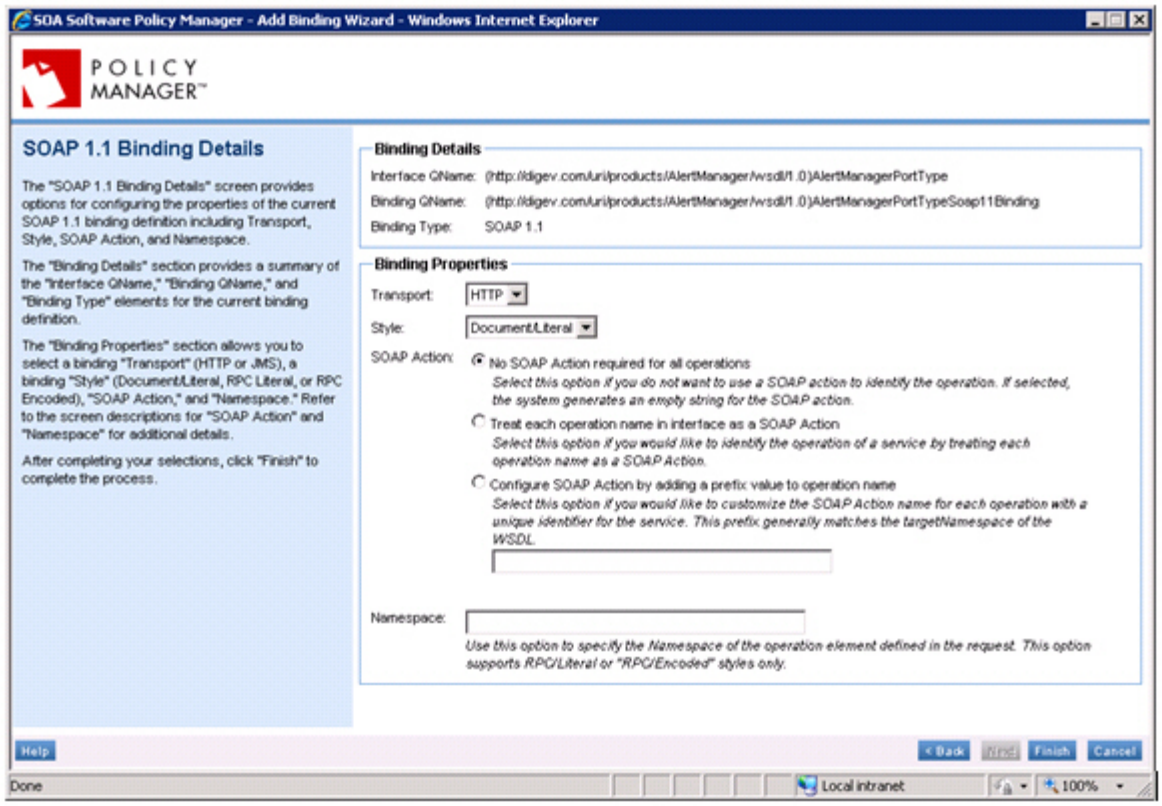


Figure 65: Configure SOAP 1.1 Binding Details

To configure soap 1.2 binding properties:

2 The "Configure SOAP 1.2 Binding Properties" screen provides options for configuring the properties of the current binding definition including, Style, Transport, and SOAP Action. The screen is organized as follows:

Binding Details

This section provides a summary of the Interface QName, Binding QName, and Binding Type elements of the current binding definition.

Binding Properties

This section allows you to pick a specific binding "Style" (Doc/Literal, RPC Literal, and RPC Encoded), Transport (HTTP/HTTPS), Transport URI, and SOAP Action.

- SOAP Actions
    - No SOAP Action required for all operations—*A radio button option that is used if you do not want to use a SOAP action to identify the operation. If selected, the system generates an empty string for the SOAP action.*
    - Treat each operation name in interface as a SOAP Action—*A radio button option that is used if you would like to identify the operation of a service by treating each operation name as a SOAP Action.*
    - Configure SOAP Action by adding a prefix value to operation name—*A radio button option that is used if you would like to customize the SOAP Action name for each operation with a unique identifier for the service. This prefix generally matches the targetNamespace of the WSDL.*
- Namespace

A text field display that is used to specify the Namespace of the operation element defined in the request. This option supports RPC/Literal or "RPC/Encoded" styles only.

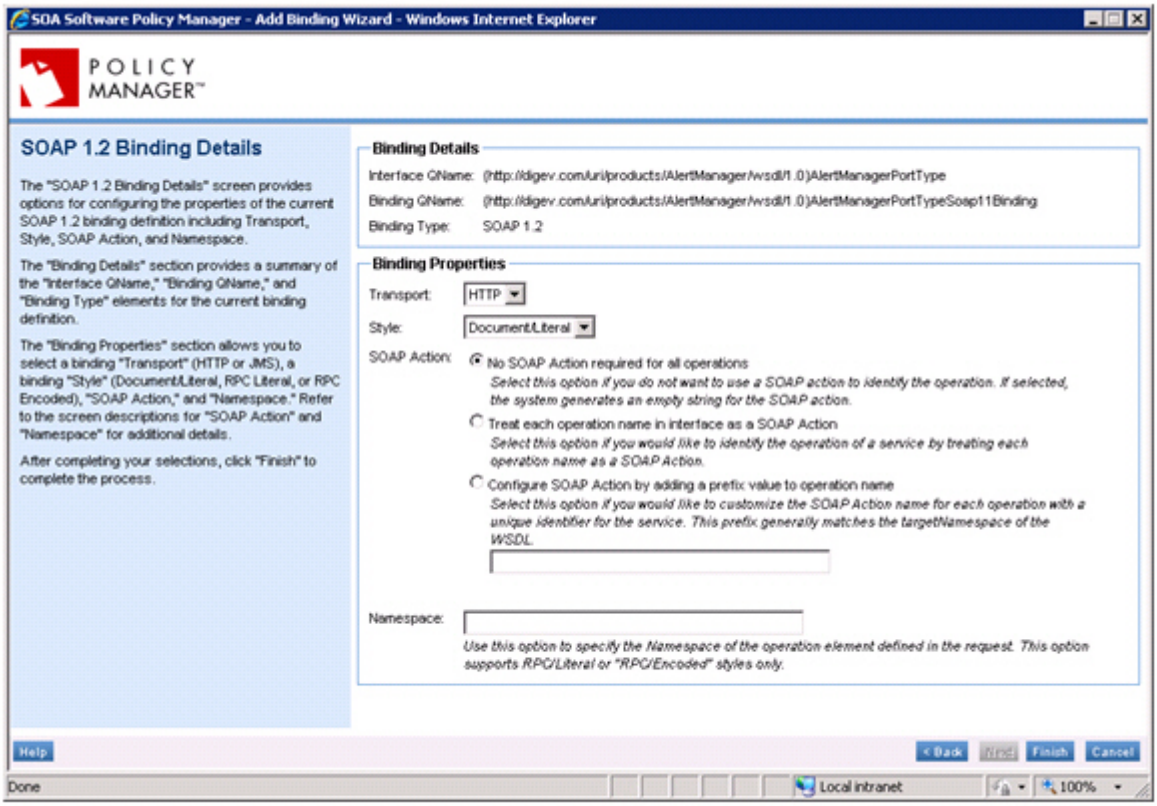


Figure 66: Configure SOAP 1.2 Binding Details

To configure the binding properties:

1 The "Configure XML Binding Properties" screen allows you to configure a custom service binding definition.

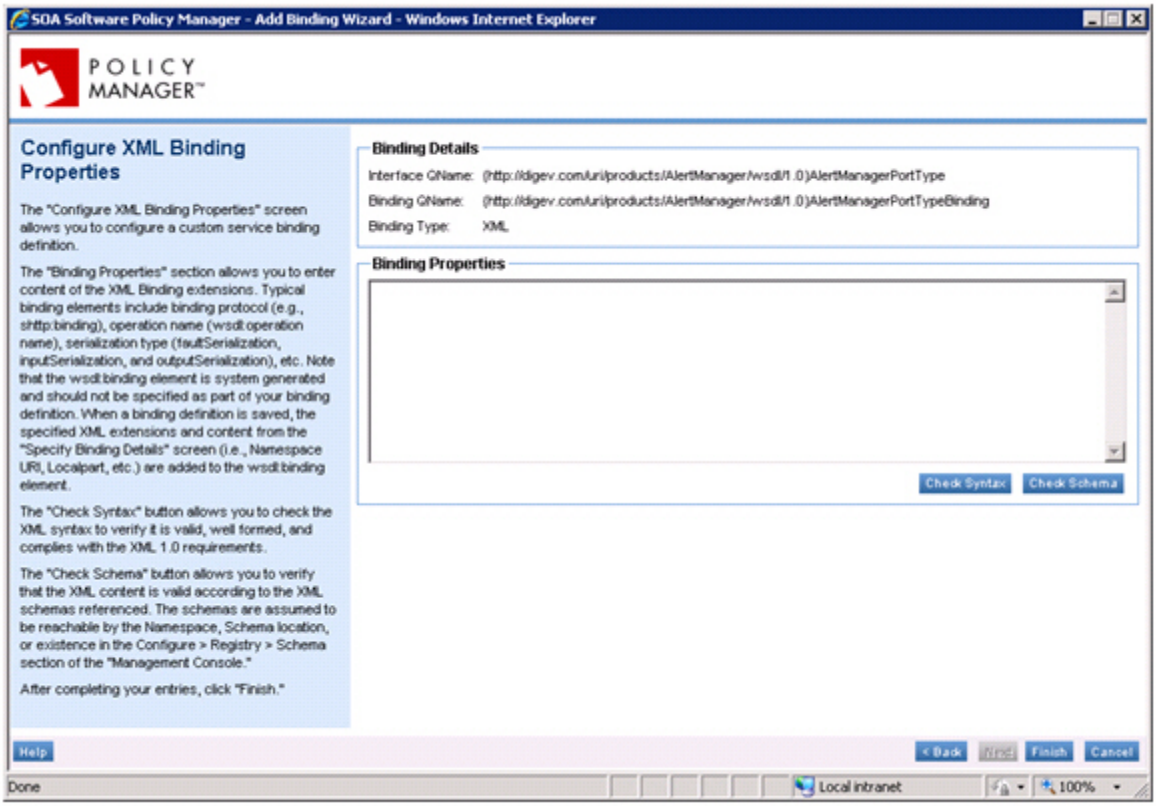


Figure 67: Add Binding Wizard—*Configure XML Binding*

The screen is organized as follows:

- Binding Properties—This section allows you to enter content of the XML Binding extensions. Typical binding elements include binding protocol (e.g., shttp:binding), operation name (wsdl:operation name), serialization type (faultSerialization, inputSerialization, and outputSerialization), etc. Note that the wsdl:binding element is system generated and should not be specified as part of your binding definition. When a binding definition is saved, the specified XML extensions and content from the "Specify Binding Details" screen (i.e., Namespace URI, Localpart, etc.) are added to the wsdl:binding element.
- Check Syntax—A button that allows you to check the XML syntax to verify it is valid, well formed, and complies with the XML 1.0 requirements.
- Check Schema—A button that allows you to verify that the XML content is valid according to the XML schemas referenced. The schemas are assumed to be reachable by the Namespace, Schema location, or existence in the "Configure > Registry > Schema" section of the "Management Console."

To complete the binding configuration:

1 After completing your selections, click "Finish." This wizard exits, and the binding definition displays on the "Bindings Summary" screen.

Chapter 7 | Configuration Migration

What Information Models does the Configuration Migration Process Support? (Organization, Service, Contract, Policy)

Policy Manager provides a web service management solution that replicates web service data from the Policy Manager Organization, Service, Policy, or Contract "Information Models," copies the data to a Package File (i.e., export), and migrates the package to the specified target destination (i.e., import). The Package File can then be imported from one Policy Manager installation to another. The "Configuration Migration" section of the Policy Manager Online Help provides instructions for migrating Organization, Service, Policy, and Contract "Information Models" from source environment to target environment.

Note: Links to the procedures for each migration function are provided below. Please review the concept material and prerequisites before performing migration tasks to ensure the optimum result.

For information about importing services or contracts, service or contract workflow, or importing a package, refer to Importing Policy Manager Data.

What is the General Process for Exporting and Importing Information Models?

The "Export Policy Wizard" provides a method of exporting Policy Manager data to a "Package" file. This "Package" file can then be imported into a different Policy Manager deployment using the "Import Package" function. Data in a "Package" file can include information from one "Information Model" type (Policies). Data to be exported is for the currently selected policy.

To export a policy

1 Navigate to "Policies Summary" using the following navigation path: Workbench > Organization > Policies. The "Policies Summary" screen displays and presents a list of policies associated with the current Organization.

2 In the summary listing, click the "Name" of the policy that you would like to export. The "Policy Details" screen displays.

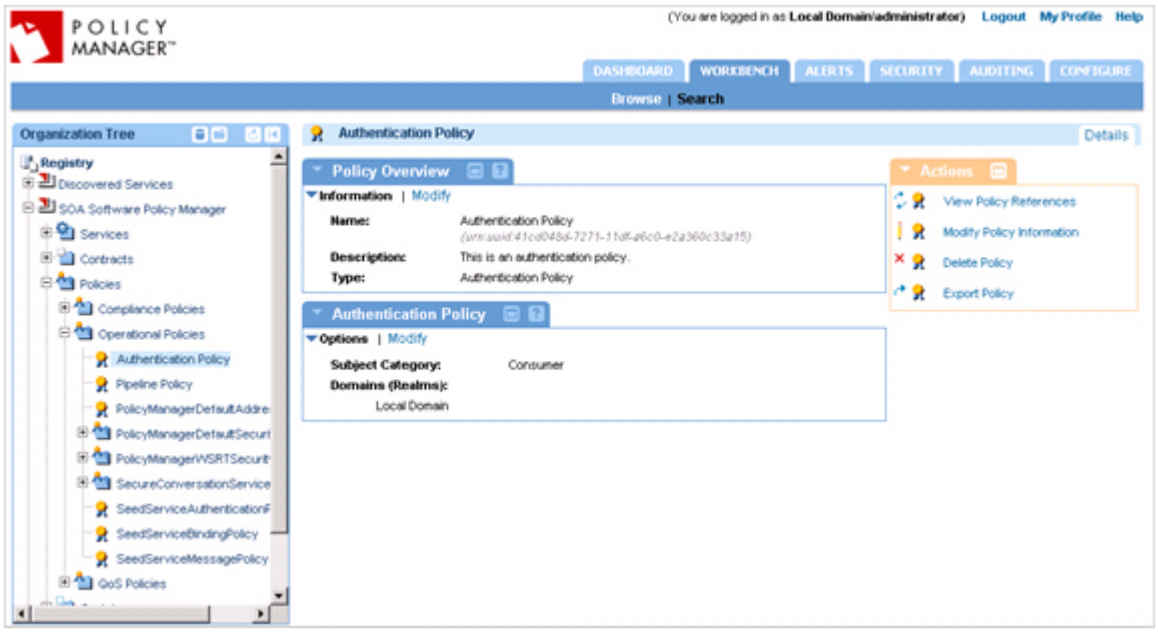


Figure 68: Policy Details—*with Export Policy*

3 To export the current Policy, in the "Actions" portlet click "Export Policy." The "Export Policy Wizard" displays.

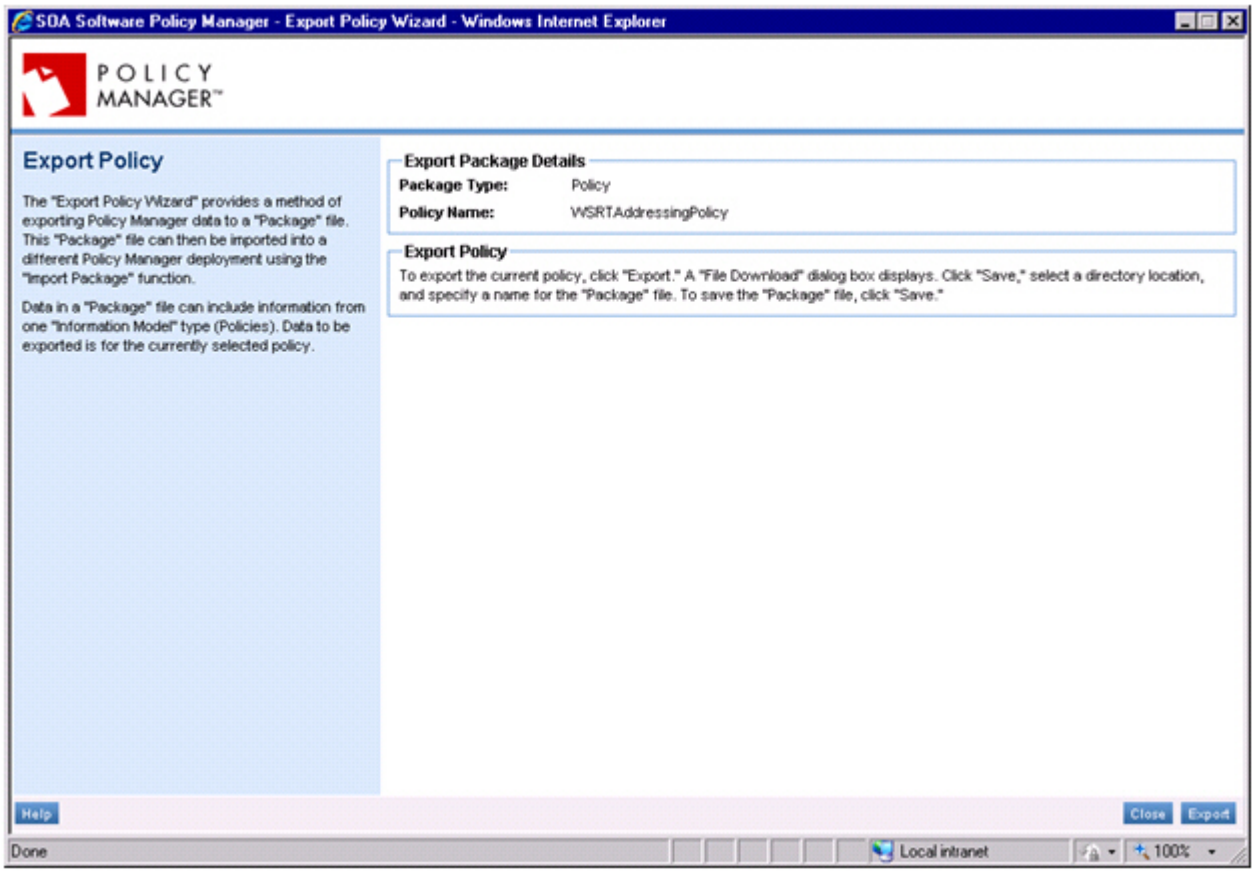


Figure 69: Export Policy Wizard

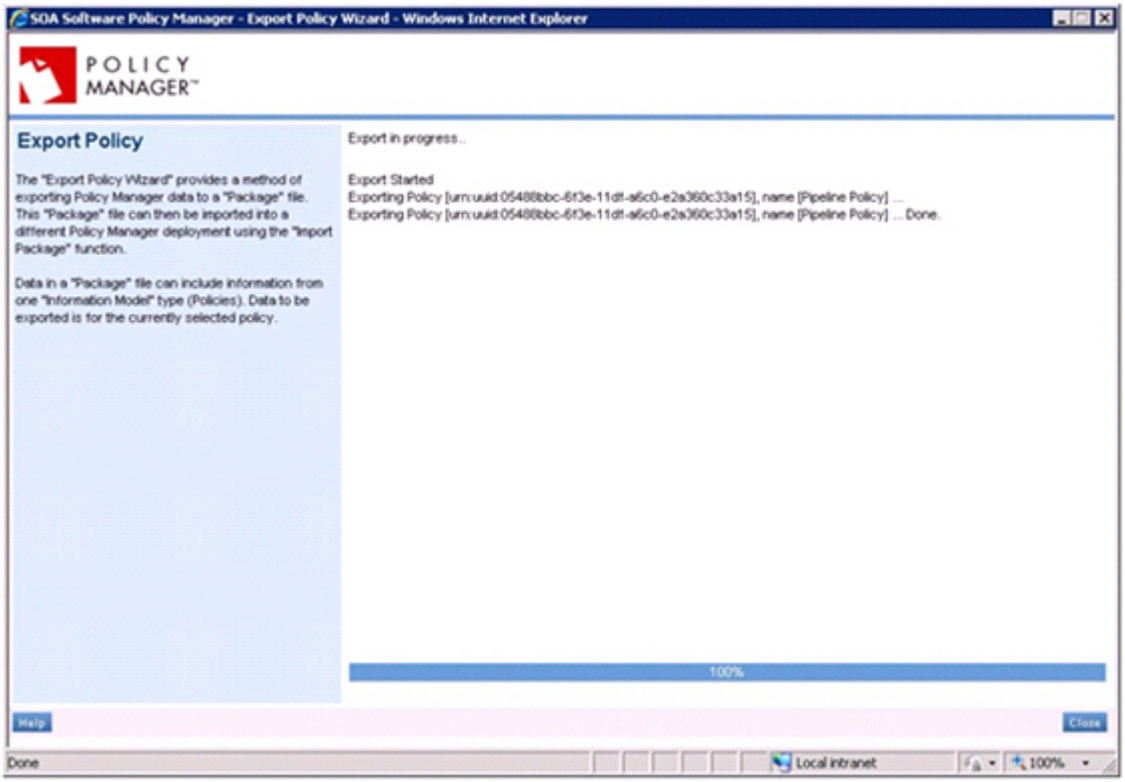
The screen is organized as follows:

Export Package Details

Package Type—Displays the name of the "Information Model" to be exported into a Package File.

Package Name—Displays the name of the current Policy to be exported.

4 To export the current policy, click "Export." The "Export in progress" and a "File Download" dialog box displays.



5 On the "File Dialog," click "Save," select a directory location, and specify a name for the "Package" file. To save the "Package" file, click "Save."

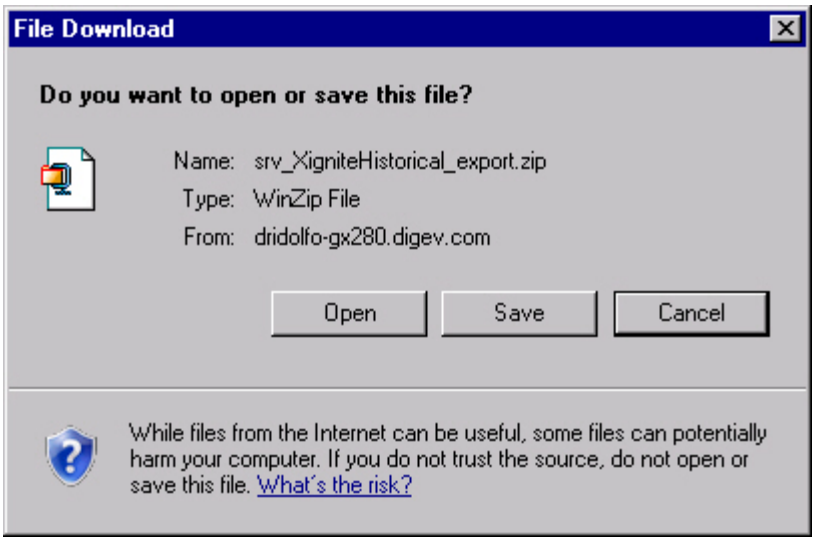


Figure 70: File Download Box—Export Organization Wizard

6 To view and copy the contents of the Package File, click Open. A window displays and presents the contents of the Package (i.e., Zip) File. Review the contents of the Package File to determine if you would like to make any adjustments.

To save the Package File from this view, select "File > Manage Archive > Copy." Select the directory where you would like the specified filename to be copied. Click "OK." The Package File is copied to the specified location.

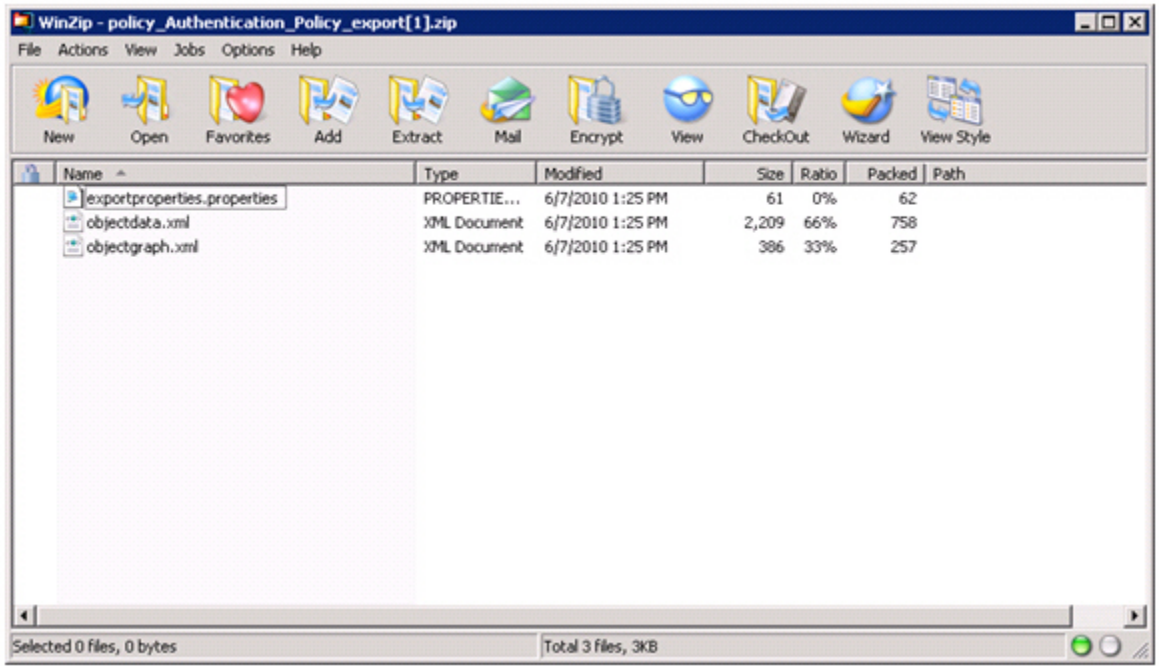


Figure 71: View/Copy Package File—*Export Policy Wizard*

7 To save the Export Package directly from the File Dialog, click "Save." Navigate to the directory location where you would like to save the "Export Package," specify the filename, and click "OK." The Policy data is exported and saved using the specified filename and directory location.

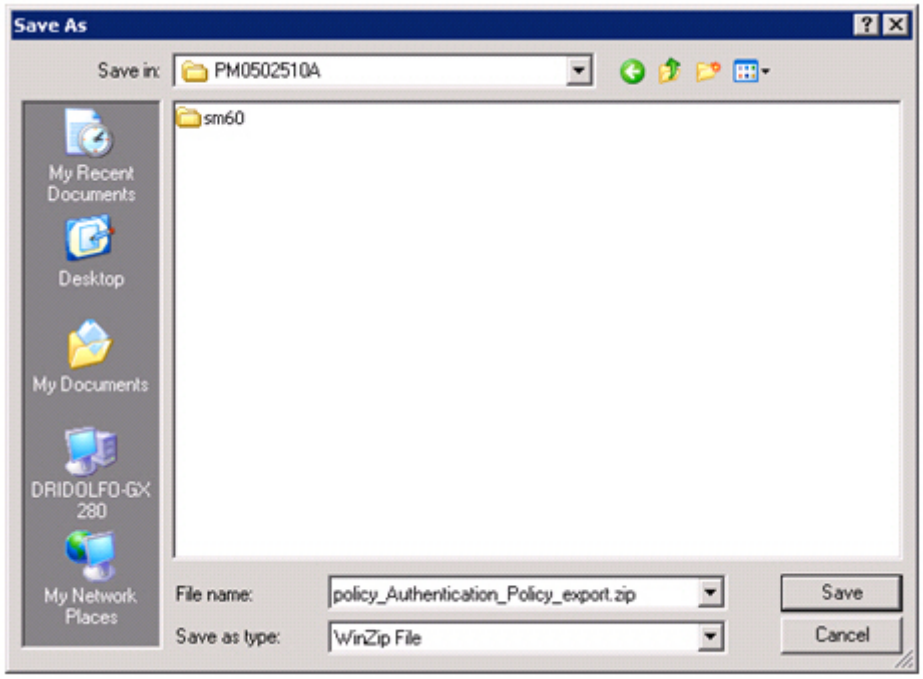


Figure 72: Save Package File—*Export Policy Wizard*

8 Your Policy is now exported to a Package File. Your next step is to complete any Prerequisite steps defined in the "Preface" section of this guide prior to importing the Package File that contains the Service configuration into another Policy Manager deployment.

The "Import Package Wizard" provides a method of importing (i.e., migrating) data into Policy Manager for the four "Information Model" types (Organizations, Services, Policies, and Contracts). Data is imported by specifying a "Package" file generated by the "Export Package Wizard." A "Package" file is a .zip file containing export elements associated with a specific "Information Model" type.

Migration of an "Information Model" is typically used in scenarios where a service needs to transition from one lifecycle stage to another and must be migrated to a different Policy Manager deployment to take advantage of custom configurations and security requirements. In addition, you can migrate a contract which refers to the service and parent organization of the service.

Two import options are provided. You can import a "Package" file that contains the "Information Model" data to be imported, and can optionally import a migration properties file that contains a container mapping configuration.

To import a package

1 Navigate to "Organization Overview" using the following navigation path: Workbench > Organization. The "Organization Details" displays. A package can be imported at the Root Organization level or Sub-Organization level.

Note: That "Reserved" Organizations that are part of the Policy Manager default installation cannot be exported and therefore the "Import Package" function does not display in the "Actions Portlet" for these Organizations.

At the Root Organization level, only Organization Package types can be imported.

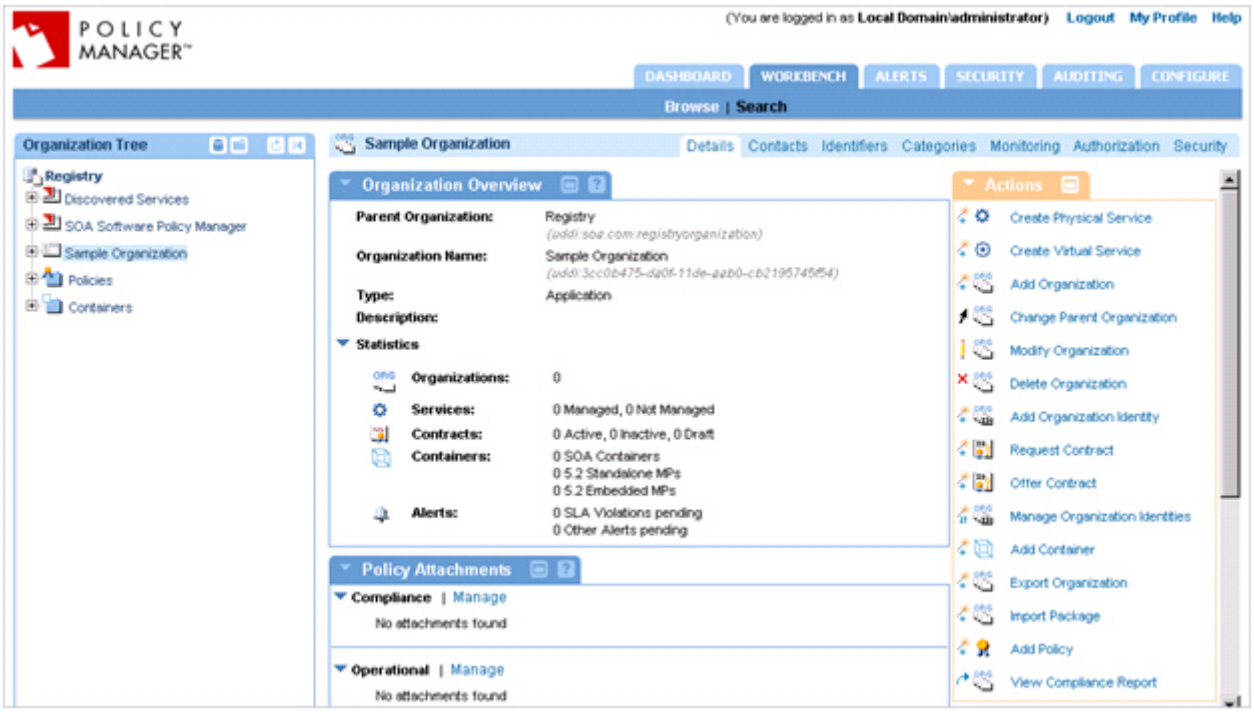


Figure 73: Organization Details—*with Export Organization*

2 To import the current Organization, in the "Actions Portlet" click "Import Package." The "Import Package Wizard" displays.

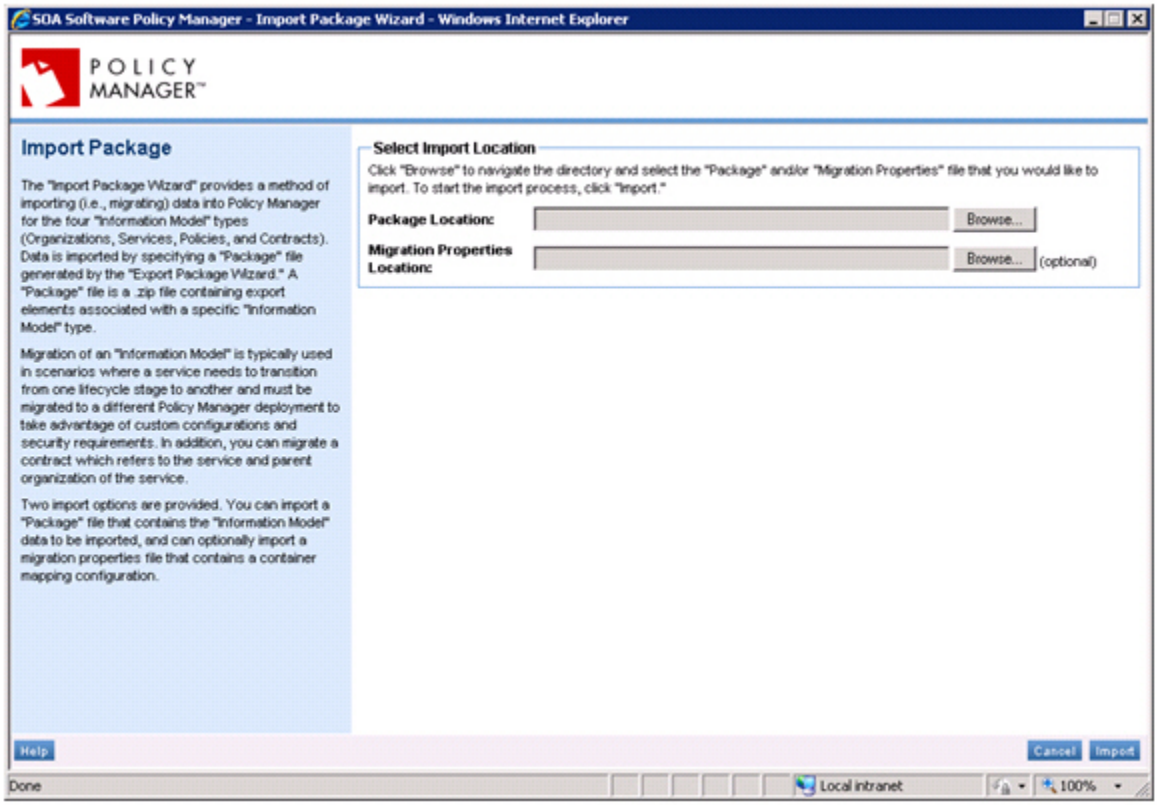


Figure 74: Import Package Wizard

3 The screen is organized as follows:

Select Import Locations

Package Location—A text box that allows you to enter the directory location and filename of the Package File to be import.

Migration Properties Location— A text box that allows you to enter the directory location and filename of the Migration Properties File to be imported.

4 To import a package, click "Browse" next to the "Package Location" field display. The "Choose File" dialog box displays. Navigate to the directory that contains the "Package File" to be imported. Select the "Package File" and click Open. The directory location and "Package File" name populate the "Package Location" field display.

5 To import a migration properties file, click "Browse" next to the "Migration Properties Location" field display. The "Choose File" dialog box displays. Navigate to the directory that contains the "Migration Properties Location" to be imported. Select the "Migration Properties Filename" and click "Open." The directory location and "Migration Properties Filename" name populate the "Migration Properties Location" field display. *Note: Use of the Migration Properties feature is optional*.

6 After completing your entries, click "Import." The "Import Package Wizard" window displays a progress indicator that shows the percentage complete of the import process and associated status messages. Status messages indicate what objects are being imported and the changes done in the environment.

7 After the import process is complete, click "Close" to exit the "Import Package File Wizard."

Services Workflow

How do I Configure Workflow for a Service?

The "Services Workflow Portlet" is a "Control Portlet" that executes the "Workflow Definition Template" assigned in the "Configure > Workflow > Services Workflow Administration" section of the "Management Console.

The "Service Workflow Portlet" is organized into three sections:

Information

State—Represents the "Workflow State" currently in progress. All of the "steps" defined in your "Workflow Definition" are presented in the "Information" section of the "Workflow Portlet" in the "State" field.

```
- <steps>
  - <step id="100" name="Draft">
    - <actions>
      - <action id="101" name="Submit for Approval">
```

Figure. <steps> in a Workflow Definition

Owner—Displays the domain and name of the user that has permission to participate in the current Workflow Task. Workflow access control is configured using the "${caller}" variable. This variable allows you to customize access permissions for Workflow "action." If you do not specify a domain and username and use the default "${caller}" variable, the access is granted for the current logged in user.

Workflow Is Completed—When all of the steps in a Workflow Definition have successfully executed, this message displays.

```
owner="${caller}"
```

Figure. Example of "${caller}" variable in a Workflow Definition.

Workflow Actions

Comments—A text field that allows you to add comment text for the "Action" that is about to be executed. This comment text will be added to the Workflow "History" and associated with the selected "Action."

Actions—Displays the name of the "Action" to be executed. A single "Action" can be displayed or two actions that represent a "choice" can be displayed (e.g., Promote / Reject).

History

Displays a read-only list of key information associated with a specific Workflow "Action."

Date/Time—Displays the date and time the Workflow "Action" was executed.

Status—Displays the Workflow "Step" that the "Action" occurred in.

Action—Displays the Workflow "Action" that was executed.

User—Displays the owner (e.g., ${caller}) who executed the "Action."

Comment—Displays details related to the execution of the "Action."

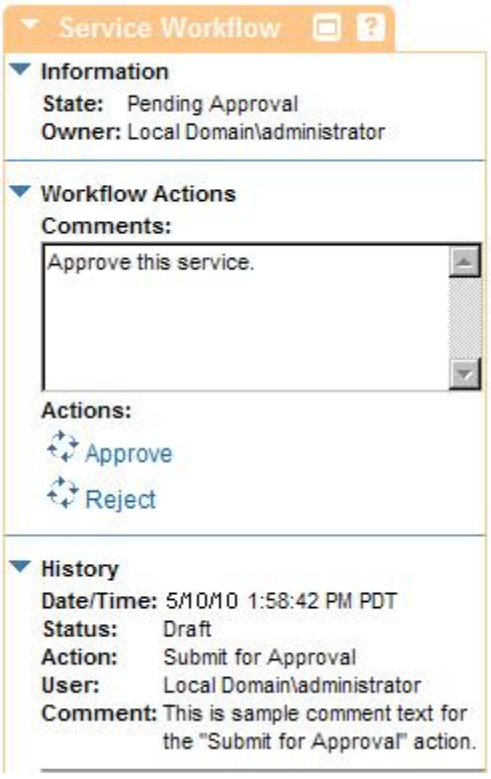


Figure 75: Workflow Control Portlet—*Services Workflow*

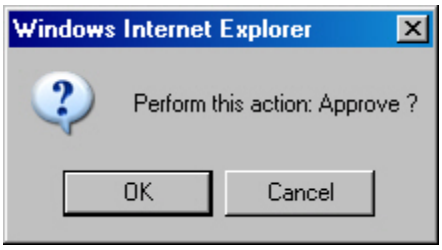


Figure 76: Execute Action Message

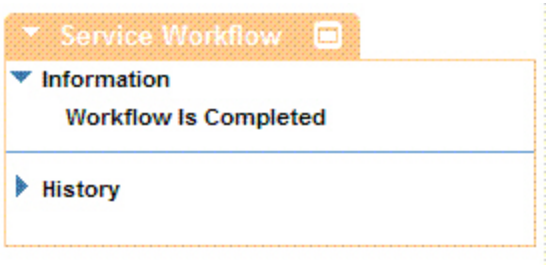


Figure 77: Services Workflow—*Workflow is Completed*

Chapter 8 | Policies

The following table provides a summary of the "Policy Types" associated with each "Policy Category."

Policy Category	Policy Type	Description
Compliance	Compliance Aggregate Policy	An Aggregate Policy is a collection policies that are gathered together to form a policy group. Policies included in an Aggregate policy are defined to achieve a specific purpose relative to governing an Policy Manager objects (i.e., Organizations, Services, etc.). Aggregate Policies can be defined for "Operational" and "Compliance" policy types.
	Compliance Script Policy	A Script Policy allows you to define a rule that uses a combination of XQuery and Java to first select the content for analysis and then to perform the analysis.
	Compliance XQuery Policy	An XQuery Policy allows you to define a rule that executes an XQuery on the service or message model to determine compliance.
	Compliance WSI BP 1.1 Policy	The WSI BP 1.1 Policy is a Policy Manager compliance policy that integrates with the WS-I (Web Service Interoperability) Basic Profile 1.0 for the purpose of providing interoperability support. This WS-I compliant policy has been certified using the WS-I Test Tool. The WS-I Basic Profile (official abbreviation is BP), is a specification from the Web Services Interoperability industry consortium (WS-I), and provides interoperability guidance for core Web Services specifications such as SOAP, WSDL, and UDDI. The profile uses Web Services Description Language (WSDL) to enable the description of services as sets of endpoints operating on messages.
Operational	Aggregate Policy	An Aggregate Policy is a collection policies that are gathered together to form a policy group. Policies included in an Aggregate policy are defined to achieve a specific purpose relative to governing an Policy Manager objects (i.e., Organizations, Services, etc.). Aggregate Policies can be defined for "Operational" and "Compliance" policy types.
	Authentication Policy	An authentication policy provides instructions to an SOA Container for what identity domain, realm, to use when authenticating a caller.
	Authorization Policy	An authorization policy provides instructions to an SOA Container for how to perform an authorization call, such as which domain, or realm, to use.

	Pipeline Policy	Pipeline Policies represent the legacy Service Manager 5.2 policies. If you have upgraded to Policy Manager 6.0 from Service Manager 5.2, the upgrade process migrates your Service Manager 5.2 policies to Policy Manager 6.0 and there you can update your policy configurations then attach them to a Policy Manager objects using the Policy Attachments Portlet.
	WS-Auditing Policies	An auditing policy provides instructions to an SOA Container for when to audit information about messages, what information to audit, and how to report the information. The following auditing policies are supported. Each assertion is represented as a different policy in the user interface, a WS-Auditing Service Policy, WS-Auditing Message Policy, and WS-Auditing SOAP Message Policy. The WS-Auditing Service Policy can be attached to services, bindings, operations, and access points. The WS-Auditing SOAP Binding Policy can be attached to SOAP binding operations. The WS-Auditing Message Policy can be attached to service operations and binding operations. The "WS-Auditing Transaction Tracking Policy" support Transaction Tracking functionality that correlates related web service events within a single activity or transaction.
	WS-Security Asymmetric Binding Policy	The WS-Security Asymmetric Binding Policy is a Security Binding policy that is used when the "client" (Initiator) and "service" (Recipient) both have security tokens.
	WS-Security Message Policy	WS-Security Message Policy specifies which portion of the SOAP message requires signed and/or encrypted.
	HTTP Security Policy	The "HTTP Security Policy" specifies the transport security requirements for web services (SOAP or REST) using HTTP or HTTPS as the underlying transport protocol. Configurable options include HTTP Authentication, SSL Authentication, and Cookie Authentication.
	WS-Security Supporting Tokens Policy	The "WS-Security Supporting Tokens Policy" is used to specify supporting tokens which are additional tokens that can be specified to augment claims provided by the token associated with the "message signature" provided by the Security Binding.
	WS-Security Symmetric Binding Policy	The WS-Security Symmetric Binding Policy is used when only one party needs to generate the security tokens. A symmetric key is established using that security token and further signing and encrypting is done using this. For example, symmetric binding can be used when only the server possesses an X509 Token.
	WS-Security Transport Binding Policy	The WS-Security Transport Binding Policy is used when the message protection is provided by the transport medium. Most common scenario is using HTTPS as the message exchange transport medium. In transport binding assertion, we can define a transport token through which we can constrain messages to be exchanged only through a defined medium. WS-Security policy specification defines a HTTPS token that defines messages be transmitted over HTTPS.

	XML Policy	An XML policy is a policy that is imported into the Workbench that is of a type that has not been registered with the product. In other words, if a policy is imported with a WSDL document that holds an assertion that is not known within the product and therefore does not have an associated user interface, it will be treated as an XML policy. An XML policy will be displayed as raw XML text.
QoS	QOS Policy	QoS (Quality of Service) Policies are used to define a metrics of requirements for ensuring service availability, performance, integrity and reliability. Different consumers of the same service might require different Service-Level Agreements (SLAs), including performance, transactional support, and security requirements.

The following Compliance Policy types are supported:

- Aggregate Policy

  This policy type provides support for building a collection policies defined to achieve a specific purpose relative to governing Policy Manager objects. See Compliance Aggregate Policy Functional Overview for more information.

- Script Policy

  This policy type uses a combination of XQuery and Java to first select the content for analysis and then to perform the analysis. See Compliance Script Policy Functional Overview for more information.

- WSI BP 1.1 Policy

  This policy type provides WS-I Basic Profile 1.0 interoperability support. See Compliance WSI BP 1.1 Script Policy Functional Overview for more information.

- XQuery Policy

  This policy type executes an XQuery on the service or message model to determine compliance. See Compliance XQuery Policy Functional Overview for more information.

Adding Policies (General Process)

The "Add Policy Wizard" is used to add policy definitions that are used to manage web service endpoints. A policy is initialized with a default configuration which you can customize to address the unique requirements of your web service management system. After a policy is configured, you then "attach" it to objects it will be managing including Organizations, Services, Operations, Bindings, and Access Points.

Add Policy Wizard

- Launch Add Policy Wizard
- Select Policy Category
- Specify Policy Details

- View Completion Summary

To launch add policy wizard:

1 Enter the following navigation path: **Workbench > Browse > Organization.** Select the "Policies" folder. The "Policies Summary" screen displays.

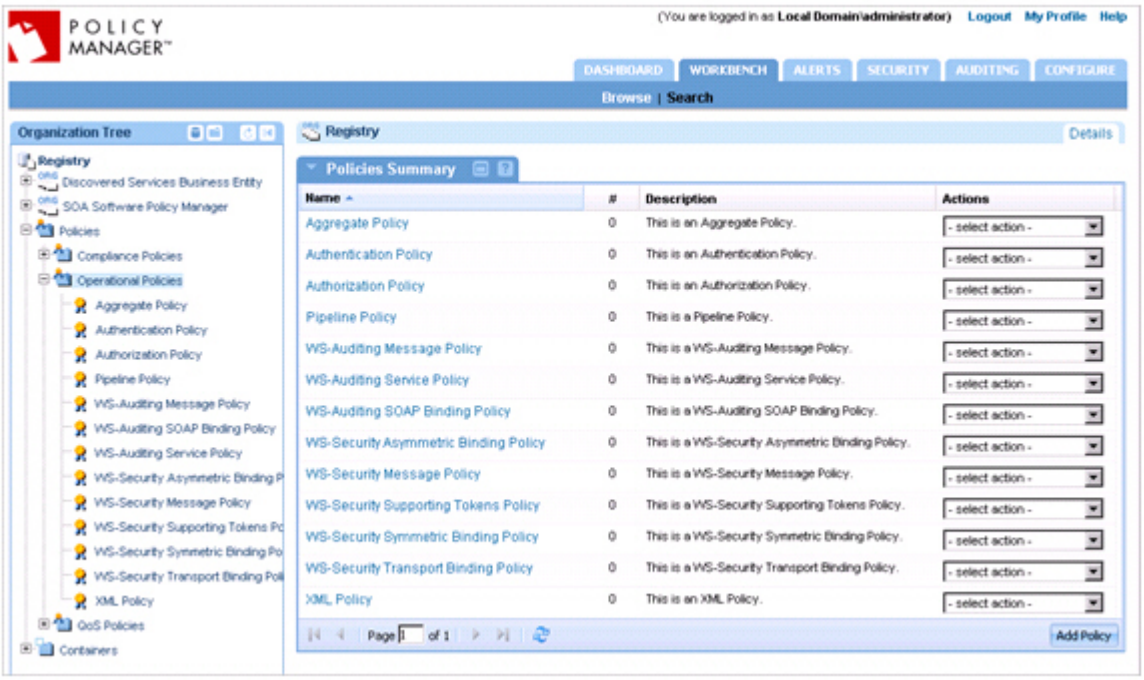


Figure 78: Policies Summary

To select policy category:

2 The "Add Policy Wizard" can be launched from the "Actions" portlet on the Organization Details screen, or by clicking "Add Policy" on the "Policies Summary" screen.

If the "Add Policy Wizard" is launched from an Organization "Actions" portlet, the "Select Policy Category" screen displays. Here you select which Policy Category for the policy to be defined. Available categories include "Compliance Policy," "Operational Policy," and "QoS Policy." Select a policy category and click "Next." The "Specify Policy Details" screen displays.

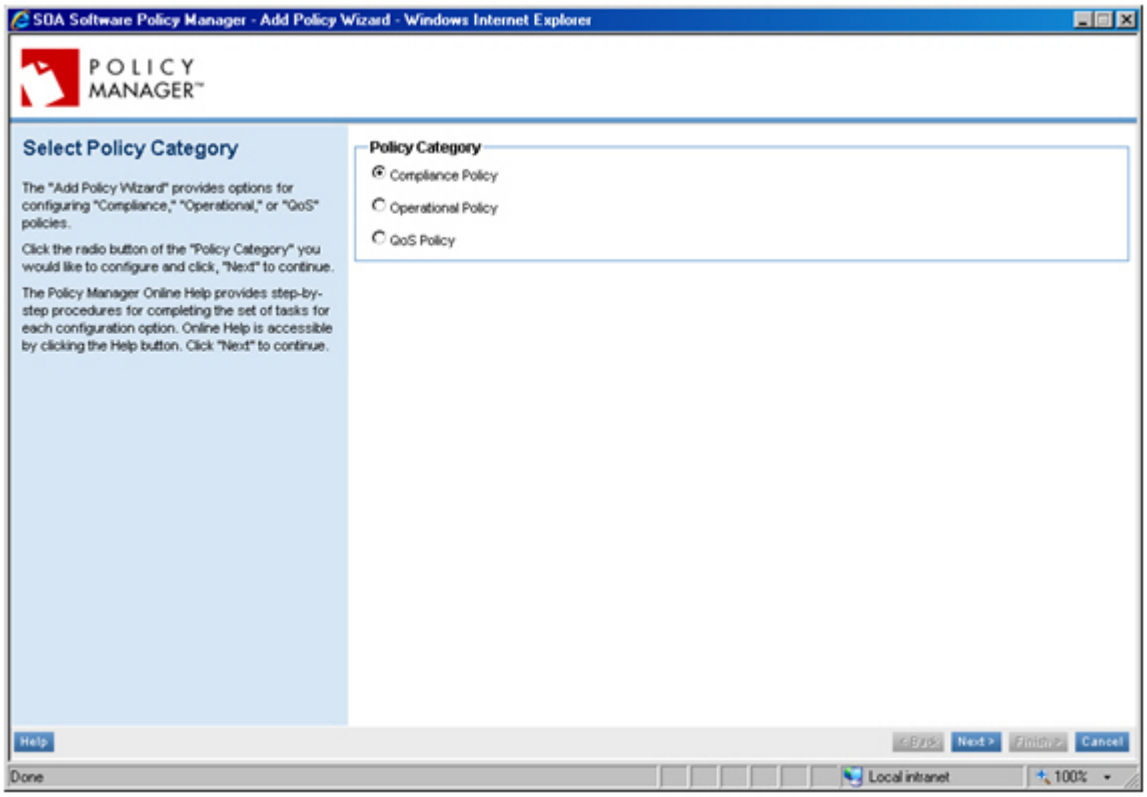


Figure 79: Select Policy Category—*Compliance Policy*

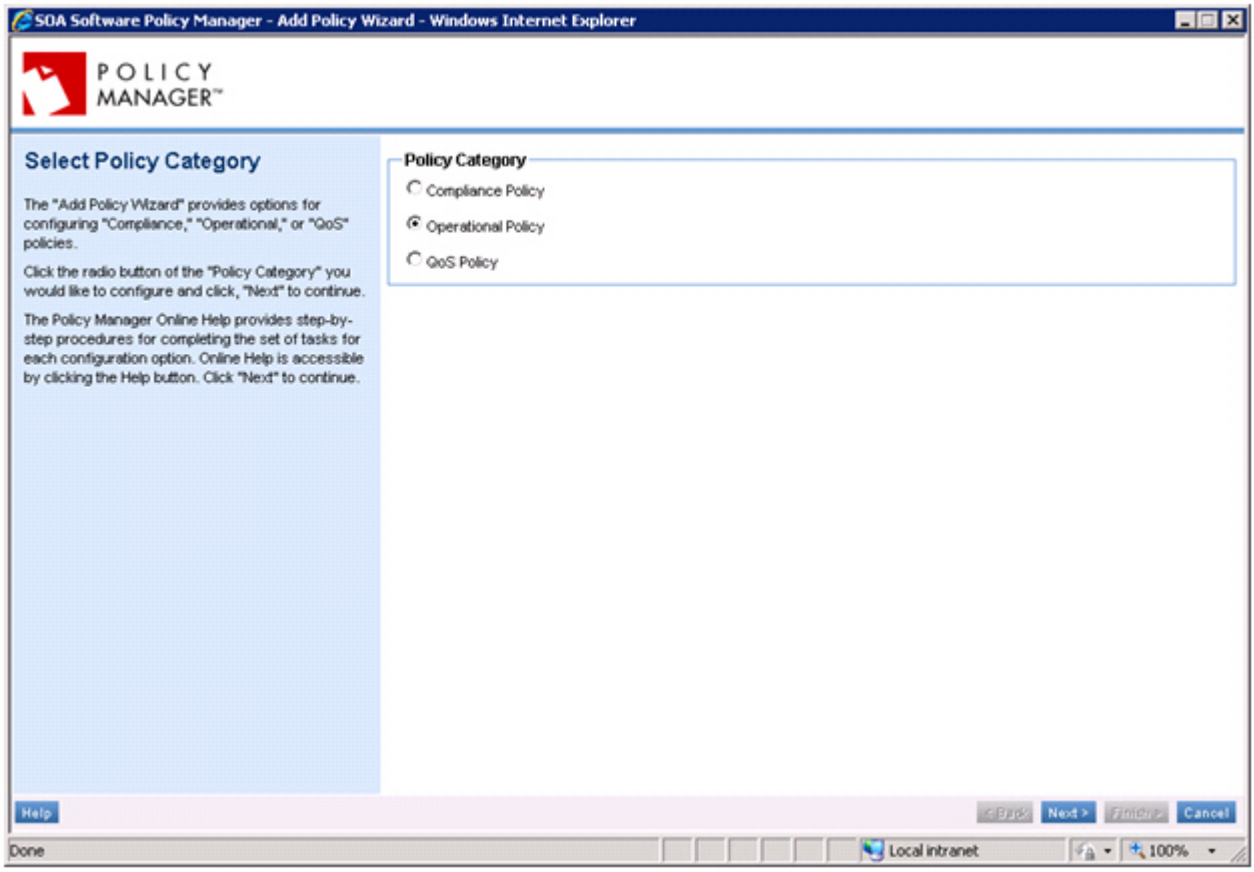


Figure 80: Select Policy Category—*Operational Policy*

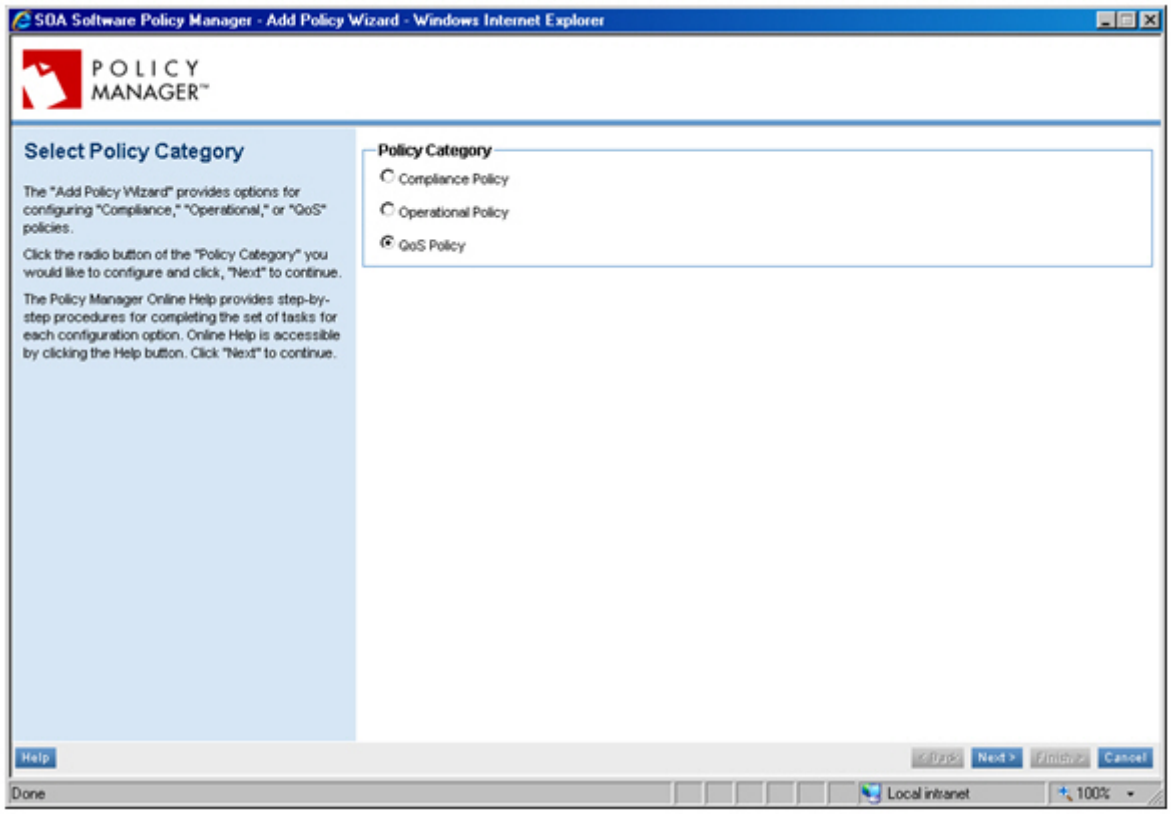


Figure 81: Select Policy Category—*QoS Policy*

To specify policy details:

3 The "Policy Details" section is used to define the "Policy Name," "Policy Key," "Description," and policy "Type" (if applicable). The policy "Type" drop-down list box displays a list of policy types that can be defined for the current policy category.

The screen is organized into the following sections:

Policy Details

- Category—Displays the policy category associated with the policy being defined.
- Policy Name—A field display that allows you to enter a policy name.
- Policy Key—A field display that allows you to enter a policy key. Note that if you do not provide a field value, a system value will automatically be assigned.
- Description—A field display that allows you to enter a policy description.
- Type—A drop-down list box that allows you to select a policy type. *This option applies to "Compliance" and "Operational" Policy Categories only. See* Policy Types.

Compliance Policy

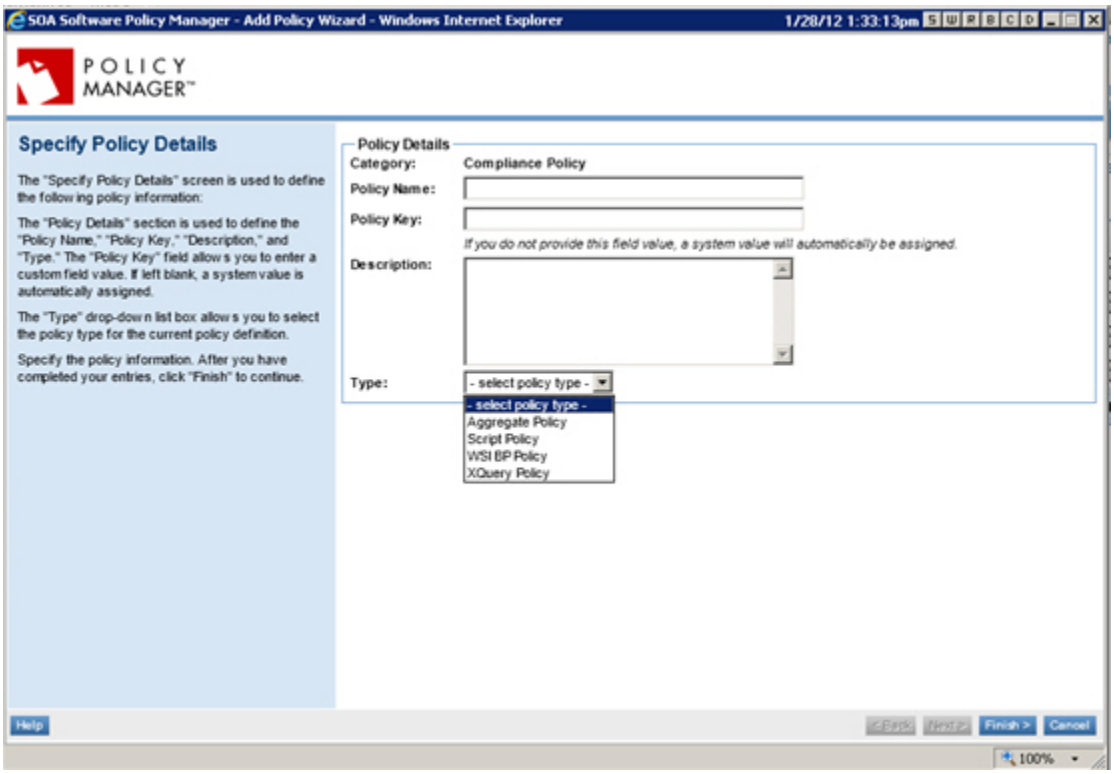


Figure 82: Add Policy Wizard—Specify Policy Details (Compliance)

Operational Policy

The Operational Policy Type includes a "Select Policy Creation Option" screen *and* a "Specify Policy Details" screen as follows:

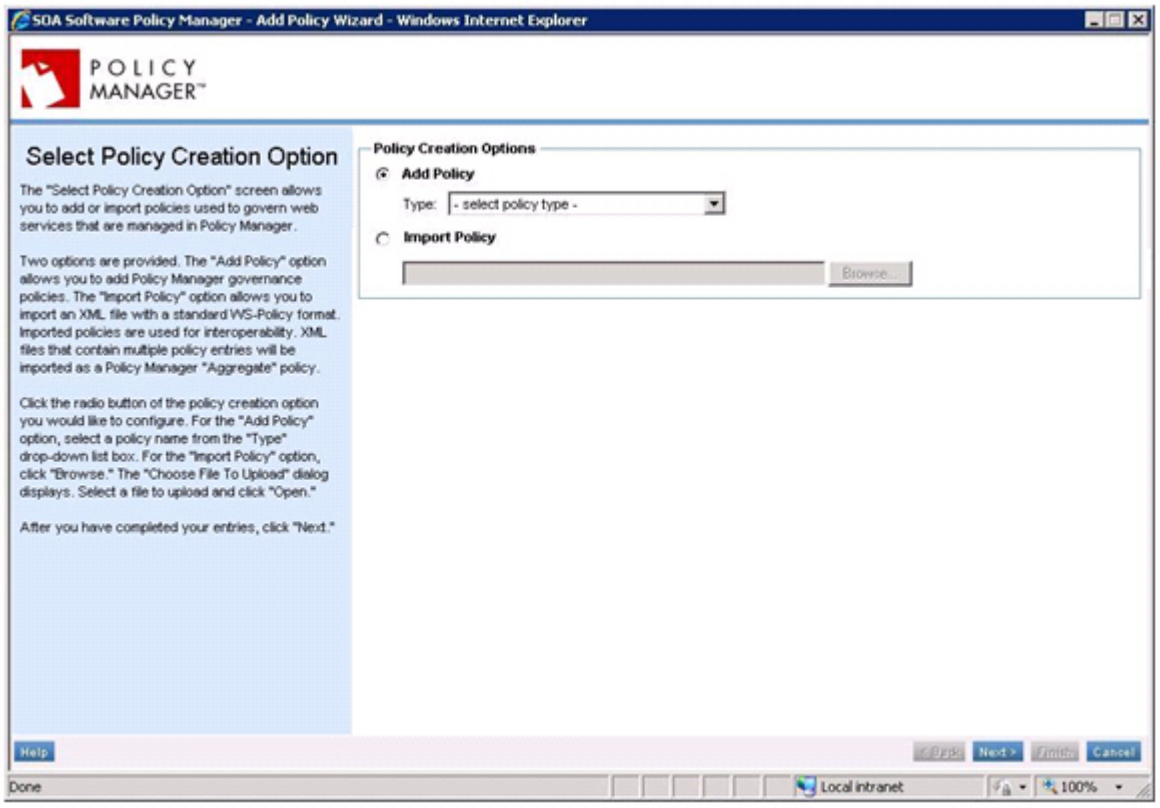


Figure 83: Add Policy Wizard—Select Policy Creation Option (Operational)

Figure 84: Policy Dropdown—Policy Drop-down (Operational)

The "Select Policy Creation Option" screen allows you to add or import policies used to govern web services that are managed in Policy Manager.

Two options are provided. The "Add Policy" option allows you to add Policy Manager governance policies. The "Import Policy" option allows you to import an XML file with a standard WS-Policy format.

Imported policies are used for interoperability. XML files that contain multiple policy entries will be imported as a Policy Manager "Aggregate" policy.

Click the radio button of the policy creation option you would like to configure. For the "Add Policy" option, select a policy name from the "Type" drop-down list box. For the "Import Policy" option, click "Browse." The "Choose File To Upload" dialog displays. Select a file to upload and click "Open."

After you have completed your entries, click "Finish." The "Specify Policy Details" for an Operational policy displays.

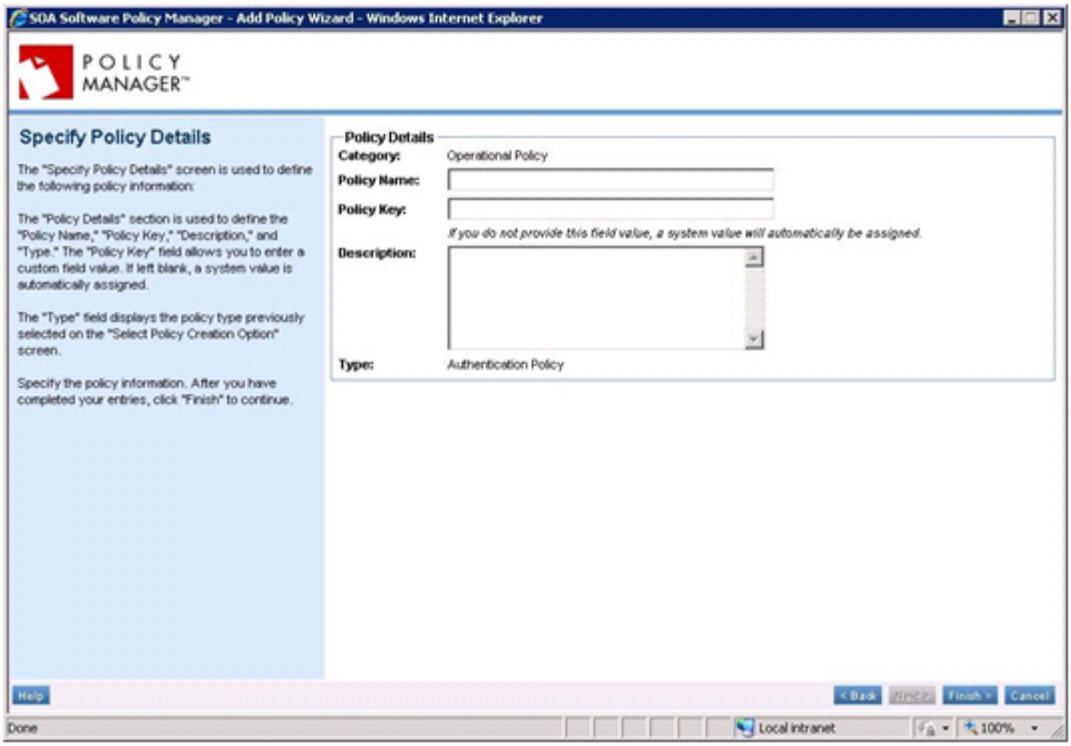


Figure 85: Add Policy Wizard—Specify Policy Details (Operational)

QoS Policy

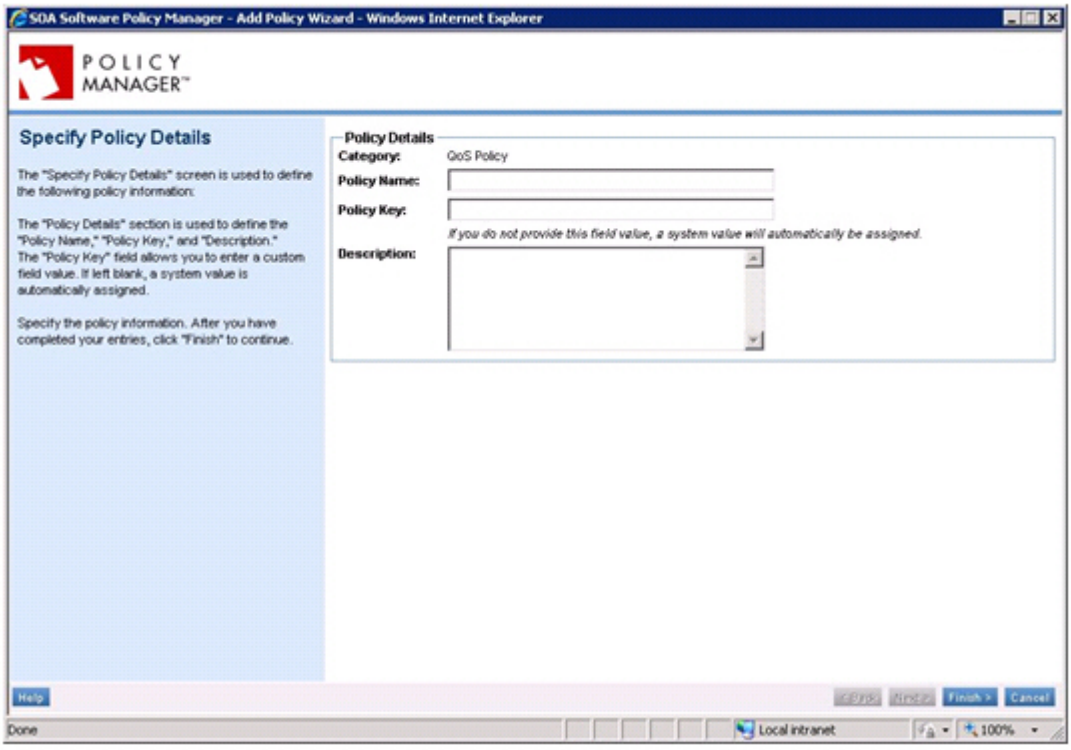


Figure 86: Add Policy Wizard—Specify Policy Details (QoS)

1 Enter the policy "Policy Name," "Policy Key," and "Description." If you have selected the "Compliance" or "Operational" policy category, select a policy "Type" from the drop-down list box. After completing your entries, click "Finish." The "Completion Summary" displays.

To view completion summary:

2 Review the summary of policy information and click "Close." The "Add Policy Wizard" closes, the policy is created, and the "Policy Details" screen displays. Here you can perform your policy configuration activities.

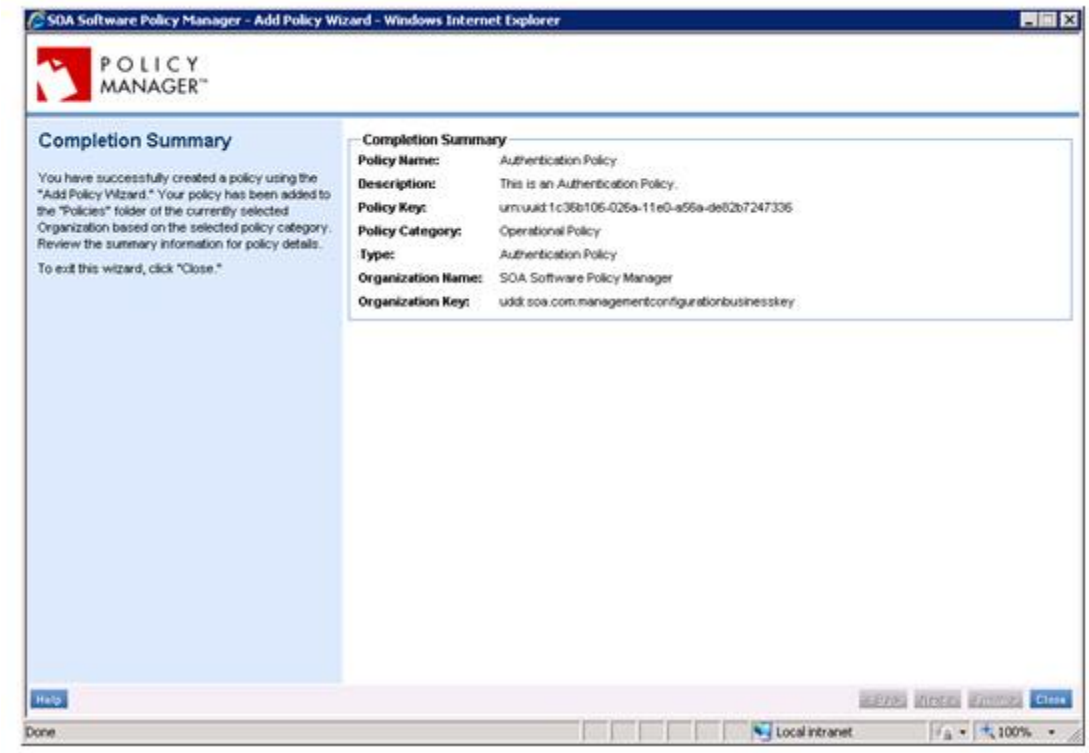


Figure 87: Add Policy Wizard—*Completion Summary*

Figure 88: Policy Details—*with Sample Policy*

Policy Use Cases

This section describes common security use cases that can be implemented using the Policy Manager 6.0 Policy Framework.

Each use case is constructed by first defining an "Aggregate Policy" and adding a series of policies that meet all the requirements of the defined use case. Each policy that is part of the Aggregate can be modified at the time it is initially defined, or after all policies that comprise the use case have been added to the Aggregate.

Use cases scenarios are presented in a table format that includes Use Case Name, a list of Policy Components that comprise the use case and recommended configuration settings to achieve the use case objective. Configuration settings listed in the table represent adjustments that should be made to the default policy configuration.

WS-Security Username/Password Use Case

The WS-Security Username/Password Use Case client authentication is performed with a Username Token which appears at the SOAP layer as a signed supporting token that is always sent from the client to the service. The user is authenticated using username/password pass in the WS-Security header.

This use case is composed of the Authentication Policy, WS-Security Transport Binding Policy, and WS-Security Supporting Tokens Policy Authentication Policy.

WS-Security Username/Password Use Case

Use Case Name	**Policy Components and Configuration**
WS-Security Username/Password	**Authentication Policy** *Use default configuration with the following changes:* 1) Subject Category = End-User 2) Domain (Realms) = Local Domain
	WS-Security Transport Binding Policy *Use default configuration.*
	WS-Security Supporting Tokens Policy *Use default configuration with the following changes:* On "Specify Supporting Token Options" screen: 1) Click "Add Token Choice" 2) Select the Token Choice 3) Click "Add Token" On "Add Supporting Token" screen: 1) Token Type = Username 2) Token Inclusion – Always to Recipient 3) Subject Category - End-User On "Specify Username Token Options" screen: Version = UsernameToken profile 1.0

X-509 Authentication Use Case

The X-509 Authentication Use Case is performed using an X.509 certificate. The client is authenticated using an X.509 certificate.

This use case is composed of the Authentication Policy, WS-Security Transport Policy, and WS-Security Supporting Tokens Policy.

X.509 Authentication Client Authentication Use Case

Use Case Name	Policy Components and Configuration
X-509 Authentication Client Authentication	**WS-Security Supporting Tokens Policy** *Use default configuration with the following changes:* On "Specify Supporting Token Options" screen: 1) Click "Add Token Choice" 2) Select the Token Choice 3) Click "Add Token" On "Add Supporting Token" screen: 1) Token Type = X.509 2) Token Inclusion – Always to Recipient 3) Subject Category - End-User On "Specify Username Token Options" screen: Version = X.509 v3 Token Profile 1.0
	Authentication Policy Subject Category = End-User Select a domain.
	WS-Security Transport Binding Policy *Use default configuration.*

Basic Authentication Using WS-Security Policy Use Case

The Basic Authentication Use Case supports the security requirement that the client is authenticated with a Username and Password by adding the credential to the HTTP Basic Authentication Header.

This use case is composed of the WS-Security Transport Binding Policy and Authentication Policy.

Basic Authentication Using WS-Security Policy Use Case

Use Case Name	Policy Components and Configuration
Basic Authentication	**WS-Security Transport Binding Policy** *Use default configuration with the following changes:* WS-Security Policy Version: 1.2 Include Timestamp: unchecked HTTPSTOKEN: Require HTTP Authentication: Checked Authentication Option: HTTP Basic Authentication Certificate Subject Category: End User **Authentication Policy** Subject Category = End-User Select a domain.

Basic Authentication Using HTTP Security Policy Use Case

The Basic Authentication Use Case supports the security requirement that the client is authenticated with a Username and Password by adding the credential into the HTTP Basic Authentication Header.

This use case is composed of the HTTP Security Policy and Authentication Policy.

Basic Authentication Using HTTP Security Policy

Use Case Name	Policy Components and Configuration
Basic Authentication Using HTTP Security Policy	**HTTP-Security Policy** *Default configuration with the following changes:* Require Authentication Scheme: Basic Authentication Subject Category: End-User
	Authentication Policy Subject Category: End-User Select a domain.

X.509 Authentication and Signature Verification Use Case

The X-509 Authentication and Signature Verification Use Case client authentication is performed using an X.509 Certificate and verified Signature.

This use case is composed of the Authentication Policy, WS-Security Message Policy, and WS-Security Asymmetric Policy.

Basic Authentication Using HTTP Security Policy

Use Case Name	Policy Components and Configuration
X509 Authentication and Signature Verification	**WS-Security Message Policy** *Use default configuration with the following changes:* Encryption: Encrypted Parts: unchecked
	Asymmetric Binding Policy *Use default configuration with the following changes:* Initiator Token: Subject Category = End-User
	Authentication Policy (Optional) Use default configuration.

Attaching Policies

The "Manage Operational Policy Attachments (Operations)" screen provides a method of attaching policies to the current service operation. Policies that are candidates for attachment include those defined in the current Organization, or in higher tier levels.

To manage operational policy attachments (operations):

1. Enter the following navigation path: **Workbench > Browse > Organization > Services**. The "Details" tab displays. Click the "Operations" tab. The "Operations Summary" screen displays.
2. Select an operation that you would like to manage policy attachments for and click the operation "Name" in the summary listing or select the "View Operation Details" action from the "Actions" drop-down list box. The "Operations Details" screen displays.
3. Page down to view the "Operational Policy Attachments" portlet.

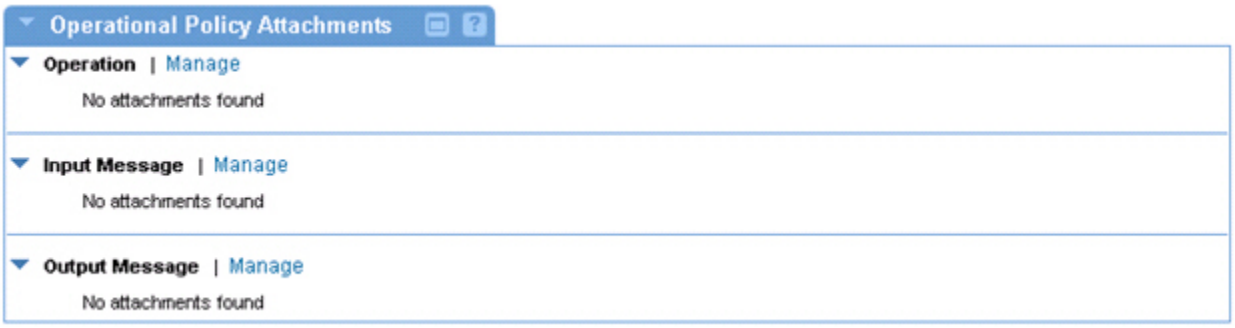


Figure 89: Operational Policy Attachments Portlet (Operations)

4 To configure operational policy attachments for the current operation, click "Manage" in the "Operation" section of the "Operational Policy Attachments" portlet.

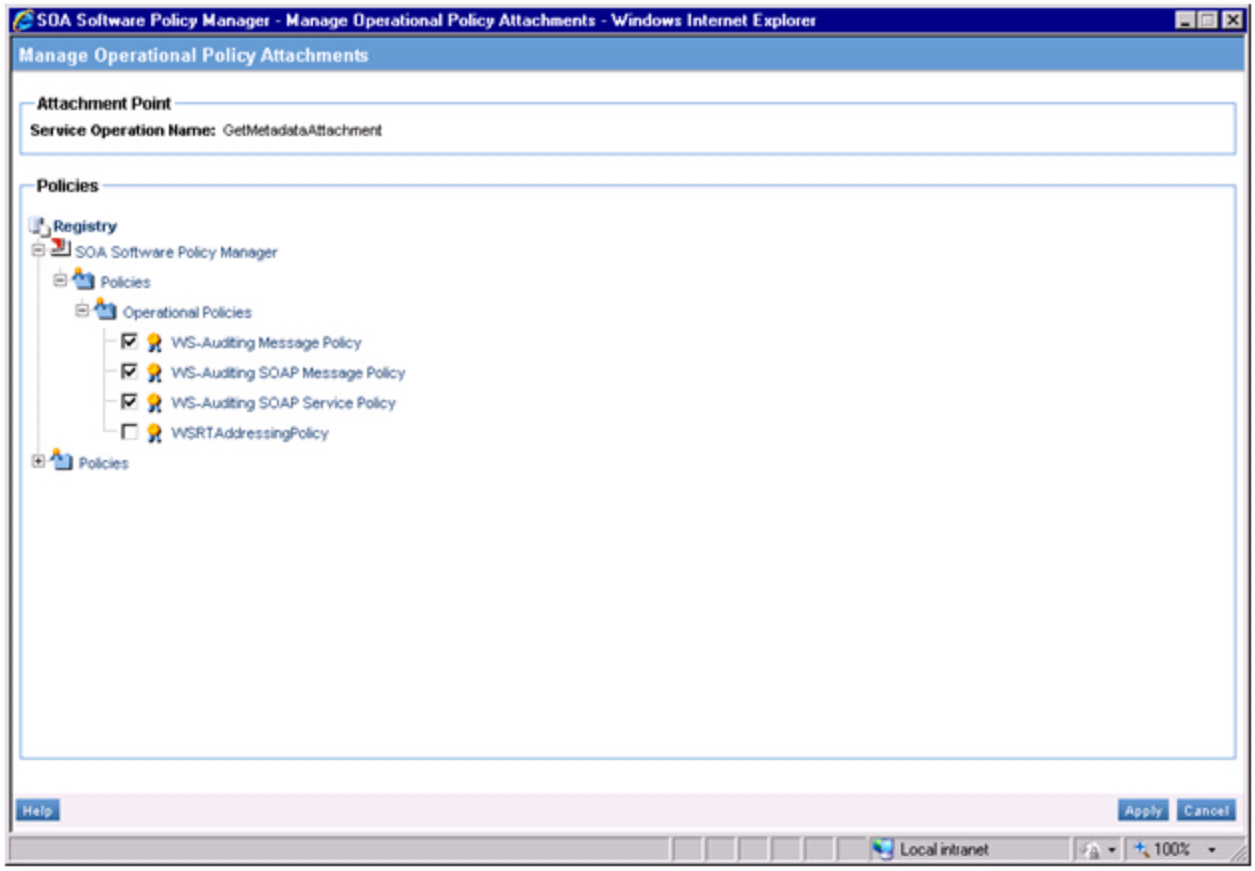


Figure 90: Manage Operational Policy Attachments (Operations)

5 To attach policies to the current service, navigate to the "Operational Policies" folder that contains Operational Policy definitions you would like to add to the current operation and click the checkbox next to each policy you would like to attach. After completing your selections, click "Apply." The selected policies display in the "Operations" section of the "Operational Policy Attachments" portlet.

Figure 91: Policy Attachments Portlet (Operations)—*with Operational Policy Attachments*

Policy Actions

How do I make a Copy of a Policy?

The "Copy Policy" function allows you to replicate a policy configuration and assign a new "Policy Name" and "Policy Key." Using this function you can configure elements that represent core functionality of a policy, replicate them to a new policy, and then perform additional customization on the copy. The "Copy Policy" function displays in the "Actions" portlet of the "Policy Details" screen for each policy.

To copy a policy:

1 Enter the following navigation path: **Workbench > Browse > Organization.** Select the "Policies" folder. The "Policies Summary" screen displays.

2 To navigate to the "Policy Overview" portlet, click the "Name" of the policy, double-click the policy line item, or select "View Policy Details" from the "Actions" drop-down list box. The "Policy Details" screen displays.

Figure 92: Policies Summary

3 To make a copy of the current policy, click "Copy Policy" in the "Actions" portlet. The "Copy Policy" screen displays.

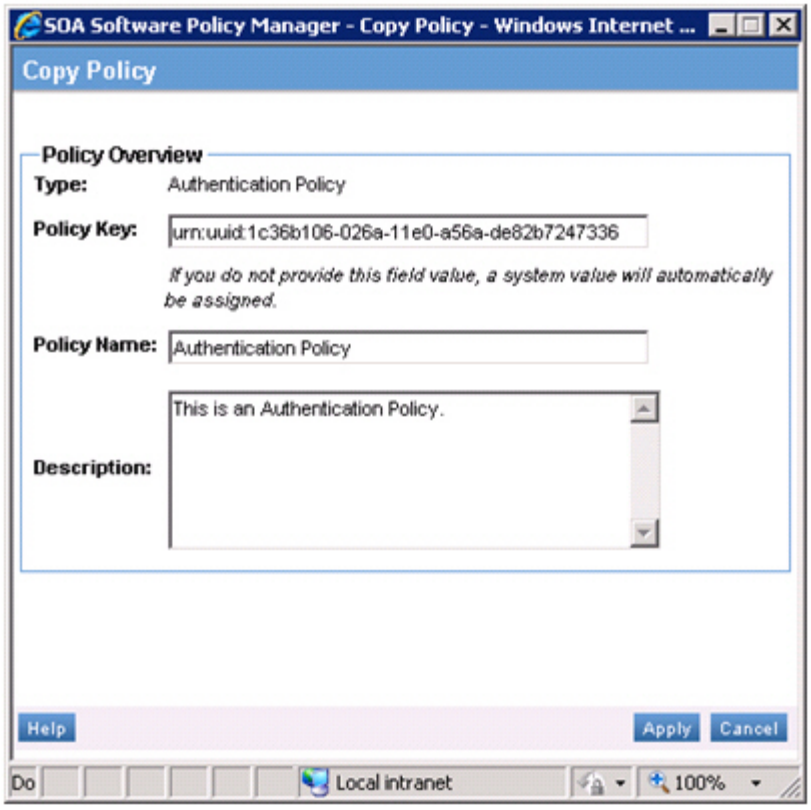


Figure 93: Copy Policy

4 Update the "Policy Key," "Policy Name" and "Description." A unique "Policy Key" is required. If you do update the "Policy Key" and attempt to save the policy, the following error displays:

Policy modification failed. Policy cannot be added. Policy with [urn:uuid:1c36b106-026a-11e0-a56a-de82b7247336] already exists.

A unique "Policy Name" is not a required entry, but is recommended to effectively differentiate the "Policy Name" on various screens throughout the "Management Console" user interface (e.g., Policies Summary, Policy Attachment Portlet, etc.)

5 After completing your entries, click "Apply." The current policy is replicated using the updated information and displays in the "Policies Folder" of the current Organization.

6 To move the policy to a new Organization, see "Change Organization (Policy)."

How do I Delete a Policy?

The "Policies Summary" screen provides an option for deleting a policy. In order to delete a policy, it cannot be "Referenced" (i.e., attached to one or more objects). The number of policy references are indicated in the "#" column and can be viewed by selecting the "View Policy References" from the "Actions" drop-down list box.

To delete a policy:

1 Enter the following navigation path: **Workbench > Browse > Organization.** Select the "Policies" folder. The "Policies Summary" screen displays.

2 Determine the policy you would like to delete, then from "Actions" drop-down list box of the policy line item select "Delete Policy."

- If a policy contains zero references, the following message displays: "Are you sure you want to delete selected policy?" displays. To delete the policy, click "OK." The selected policy is deleted from the "Policies Folder" of the organization. To exit without deleting the policy, click "Cancel."

Figure 94: Delete Policy Message—for policy with zero references

- If a policy contains one or more references, the following message displays "This policy has "#" references(s) and cannot be deleted. Update your policy attachments prior to deleting a policy." To delete references, first review the references associated with the current policy by selecting "View Policy References" from the "Actions" drop-down list box. Then delete references by navigating to the relevant "Policy Attachment" screens and removing the policy references.

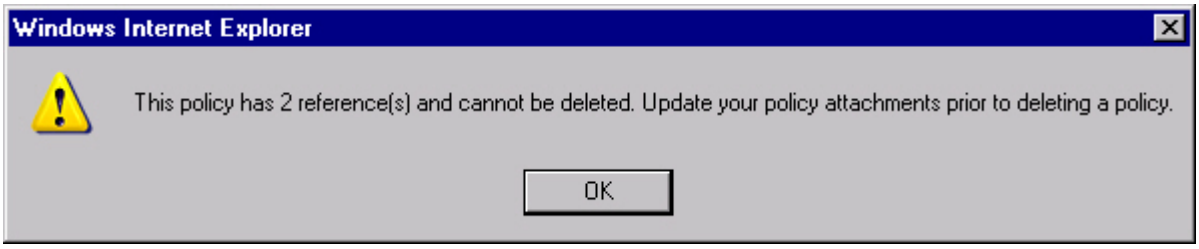


Figure 95: Delete Policy Message—for policy with one or more references

3 If your policy contains zero references and qualifies for deletion, click "OK." The policy will be deleted and will no longer display on the "Policies Summary" screen.

How do I Modify Policy Details?

The "Modify Policy Information" function allows you to change the "Policy Name" and "Description" of a service.

To modify policy information:

1 Enter the following navigation path: **Workbench > Browse > Organization.** Select the "Policies" folder. The "Policies Summary" screen displays.

2 The "Modify Policy Information" can be launched using two different methods.

- From the "Actions" portlet" on the "Policies Summary" screen select "Modify Policy Information."
- In the "Name" column of the "Policies Summary," click the name of the policy you would like to administer. The "Policy Overview" for the selected policy displays. On the "Policy Overview" portlet, click the "Modify" link next to "Information" section.

Figure 96: Policies Overview—*Information*

3 Update the "Policy Name" and "Description" as needed and click "Apply" to commit your changes. The policy updates are reflected in the Organization Tree and associated screen areas.

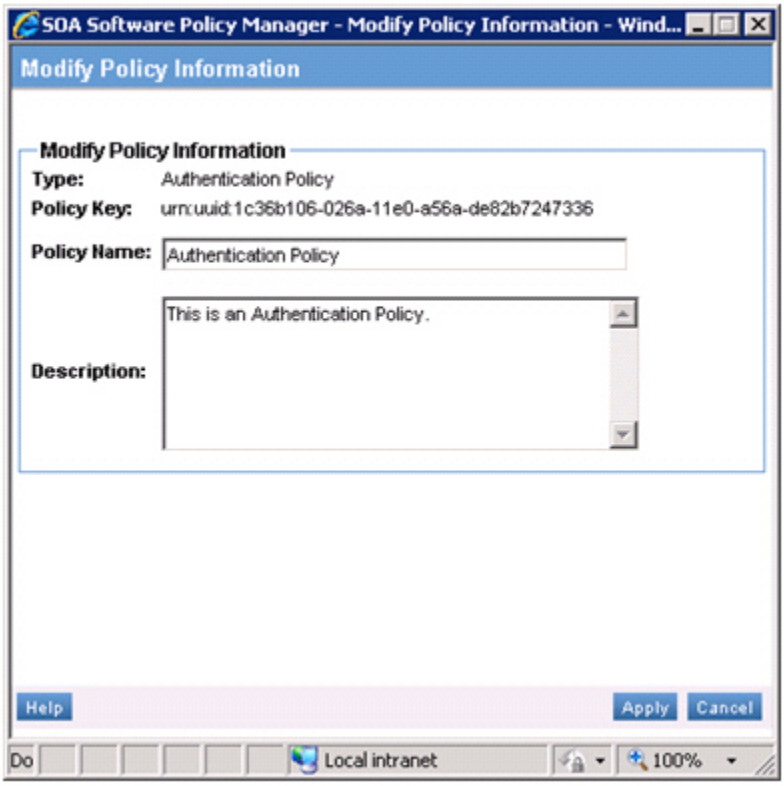


Figure 97: Modify Policy Information

Managing Policy References

How do I View Policy References?

The "View Policy References" screen displays a list of subjects that the current policy is "directly" attached to. The screen is accessible by selecting the "View Policy References" action on the "Policies Summary" screen for a selected policy.

To view policy references:

1 Enter the following navigation path: **Workbench > Browse > Organization.** Select the "Policies" folder. The "Policies Summary" screen displays.

2 In the summary listing, click the "Name" field of the policy you would like to view references for. The "Policy Details" screen displays.

Figure 98: Policy Details—View Policy Reference Action

3 In the "Actions" portlet, click "View Policy References." The "View Policy References" screen displays.

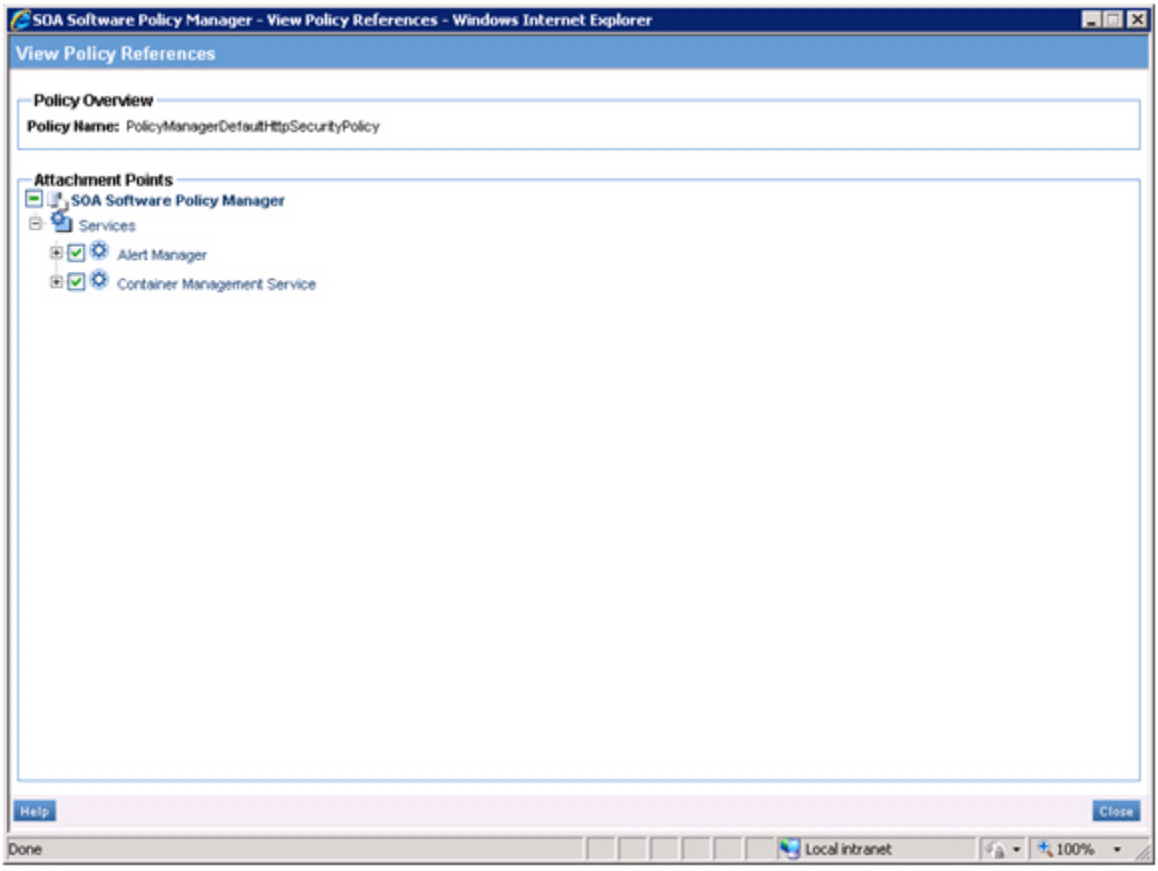


Figure 99: View Policy References

The screen is organized as follows:

- Policy Overview

Read only text label that displays "Policy Name" associated with the reference.

- Attachment Points

A tree hierarchy that containers all attached subjects of the policy. Only subjects or parent nodes of the subjects to enable navigation are in the tree. Children of attached subjects are also listed in the tree and are shown as selected implicitly (green box). The tree is expanded to each attached subject when the page is displayed. The nodes of the tree are Organizations, Services, Service Port Types, Service Port Type)Operations, Service Port Type Operation Messages, Service Bindings, Service Binding Operations, Service Binding Operation Messages, and Service Access Points (nested within a Service Binding). The nodes can be expanded and collapsed, but the check boxes are read-only.

4 After you have completed reviewing policy references, click "Close."

How do I Manage Policy References?

The "Manage Policy Attachments for Organization" screen is used to attach policies to the current organization. Each "Policy Category" (Compliance, Operational, QoS) includes a separate "Manage Policy Attachments for Organization" screen. This screen is accessed by clicking "Manage" next to each Policy Category type on the "Policies Attachment" portlet.

Attaching a policy is accomplished by clicking "Attach," navigating the "Organization Tree" and selecting one or more policies. The "Organization Tree" is filtered to display policies for the Policy Category that is the focus of the screen.

If services contained in the current Organization are configured with assigned categories (i.e., categorization tModel's) you can assign these same categories to the Organization. This task is accomplished using the "Filtering Categories" section. If the assigned categories match those assigned to services that are part of the Organization, the policy is automatically referenced in the Policy Attachment Portlets of the service and its elements. Category filtering is performed by navigating the Category Tree and clicking the checkbox of the desired category. About Service Categories for details on service category assignment.

> **Note**: Defining policies for each Policy Category is a prerequisite to using this feature. Also, to take full advantage of the category assignment functionality, it is recommended that service category assignment be performed prior to attaching policies to realize the full benefits of the service referencing.

Chapter 9 | Consuming Services

Note: Service consumption will typically be driven out of either Lifecycle Manager or Community Manager. In Policy Manager the process is much more manual.

Search

Search provides tools for querying, managing, and securing existing web services. Service management can be applied to *physical*, *virtual*, and *discovered* service types.

How do I query, manage, and secure existing web services using Search?

The "Workbench > Search" section of the Management Console provides tools for querying, managing, and securing existing web services. Service management can be applied to *physical*, *virtual*, and *discovered* service types.

Services currently defined in Policy Manager can be queried by performing a Services Search. Services Search supports key service objects associated service configuration elements (e.g., name, category, type), service access (e.g., user account or role), and service activity (e.g., usage, alerts, etc.), and provides a baseline of search criteria for targeting key information in each area.

New services not defined in Policy Manager, or existing services not hosted by a Container (i.e., SOA Container, Legacy Container— Standalone or Embedded) are candidates for management. The initial process of bringing a service under management is performed using the "Create Physical Wizard."

Note: It is recommended that you first populate your "Organization Tree" with business entities that will part of your Policy Manager deployment. This activity is performed in "Workbench > Browse" section of the Management Console using the "Add Organization Wizard." This wizard is available in the "Actions Portlet" of the Root Organization (i.e., Registry). After this task is complete, the Organization Tree that is part of the "Select Organization" screen in each service management wizard will be populated with your defined organizations.

The service management process involves creating a "manageable service" by configuring a set of properties, selecting an organization (i.e., business entity), and associating a "manageability endpoint" (i.e., Management Point) with the service to complete the deployment process. Based on your business requirements you can configure both activities to create a "managed service" that is deployed, or you can define the service properties and create a "manageable service" and select the Management Point host via the Management Points tab when you are ready to deploy the service.

After completing the management process, you can view service details, modify the policy configuration for the service, configure bindings (i.e., HTTP, HTTPS, and JMS listeners), configure categories for the service, and define rules to manage XML Denial of Service attacks, and monitor service performance.

The "Workbench > Search" navigation is the starting point for beginning service management activities. After performing your search, double click on the name of the service you would like to manage. The " Workbench > Organization > Services" screen displays. Within Workbench > Search, the following key activities can be performed: Perform Services Search.

Browse

Policy Manager provides functionality that allows you to define Organizations and structure them in what is called an "Organizational Hierarchy." The "Organizational Hierarchy" is a tree structure that contains a series of nodes that represent different "states" or areas of functionality associated with the current web service. When you add a new "Organization" to Policy Manager it is added to the "Organizational Hierarchy." You build your Organizational Hierarchy one organization at a time. Any tier level of the Organizational Hierarchy can be a "Parent Organization." Organizations at the same level are peers. You can populate the Organization Tree with additional tier levels (i.e., Sub-Organizations) and can also designate these sub-organizations as "Parent Organizations" based on your requirements. You can also assign a "Type" to an organization to uniquely identity it based on its relationship in the Organization Tree. If you are just starting and have not defined any organizations, the "Parent Organization" will be the "Root" organization.

By structuring your organizations in a Parent/Child hierarchy you achieve the ability to measure and report consumption of services at a specific organization level (i.e., Parent or Sub-Organization) and by groups of departments within an organization. You can also evaluate service consumption based on usage by a specific user or application.

When an Organization is assigned to a service, it acts as a container for the Service and its associated nodes.

What is the Workbench Organizational Hierarchy?

The "Workbench" section of the Management Console provides tools for publishing web services and performing registry and service management activities. The following key activities can be performed:

What are the Workbench views? [Root, Sub-org, Services, Contracts, Policies, Container, Processes (PM70), Scripts (PM70)]

Defining your service management solution involves defining the necessary use cases to meet your customer requirements, deploying the solution to a development environment, and the building a process around the Policy Manager product. A plan is then developed that describes how to implement this solution in an SOA Management Platform. This document is the input to Policy Manager.

Policy Manager provides a variety of different wizard utilities that automate the configuration and deployment processes. To optimize the use of these wizards, a series of prerequisite configuration steps are required.

- Define Organizations

  The first step in building your SOA Infrastructure is to define your Organizations and assign an Organization Type (Application, Company, Department, or Project).

- Define Containers

  Second, you define the Containers that will act as the service "host," and will execute the web service policy configuration and process request and response messages associated with service transaction activity.

- Define Services

  Third, you create your web service configuration using the "Create Service Wizard." With your first two prerequisites completed, both the Organization and Containers will be selectable when you run the wizard.

- Define Policies

  Define security policies that reflect the business requirements you would like to apply to Policy Manager "Objects" (i.e., Organizations, Services, Operations, Bindings, Access Points, Contracts). After defining policies, you then assign policies to "Objects" using the "Policy Attachments Portlet" available in each "Object" section of the "Management Console."

- Define Contracts

  Contracts can be defined after you have created your Services as they are applied to services that registered in the Policy Manager data repository.

How do I Browse the Workbench?

Browsing provides tools for publishing business entity and business services information. Information is grouped by Organization in an "Organization Tree." Each organization includes Services, Contracts, and Containers that pertain to that organization.

Create a Contract

How do I create a Provided Contract?

The "Activate Contract" action deploys the current contract version and the contract begins to monitor service activity.

When a "Contract State" displays as "Activated," this means that it has *either* just completed Workflow and the "Contract Workflow Portlet" displays the "Workflow Is Completed" message, or the contract has been deactivated using the "Deactivate Contract" action in the "Actions Portlet."

The "Activate Contract" action in the "Actions Portlet" is used to activate a contract that displays a "Contract State" as "Deactivated."

> **Note**: A contract cannot be activated when Contract Workflow is in progress. The "Activate Contract" action in the "Actions Portlet" is only visible when the "Contract Workflow Portlet" displays the "Workflow Is Completed" message.

To activate a contract:

1 Enter the following navigation path: **Workbench > Browse**. The "Root Organization Summary" screen displays.

2 In the "Organization Tree," double click the name of the organization that includes contracts you would like to view. To view contracts associated with the current organization, click the "Contracts Folder." The "Contracts Summary" screen displays.

3 To view the "Provided Services Summary" and a listing associated contracts in the "Organization Tree," click the "Provided Contracts" folder. The "Provided Services Contracts" screen displays and the folder hierarchy expands and displays a selectable list of contracts.

4 To view the "Consumed Services Summary" and a listing of associated contracts in the "Organization Tree," click the "Consumed Contracts" folder. The "Consumed Services Contracts" screen displays and the folder hierarchy expands and displays a selectable list of contracts.

5 Select a contract in either the "Provided Contracts" or "Consumed Contracts" folder. The "Contract Details" screen displays for the selected contract.

6 Complete the "Contract Workflow" so that the status message in the "Contract Workflow Portlet" displays "Workflow Is Completed." The "Contract State" in the "Contract Overview Portlet" will display as "Activated."

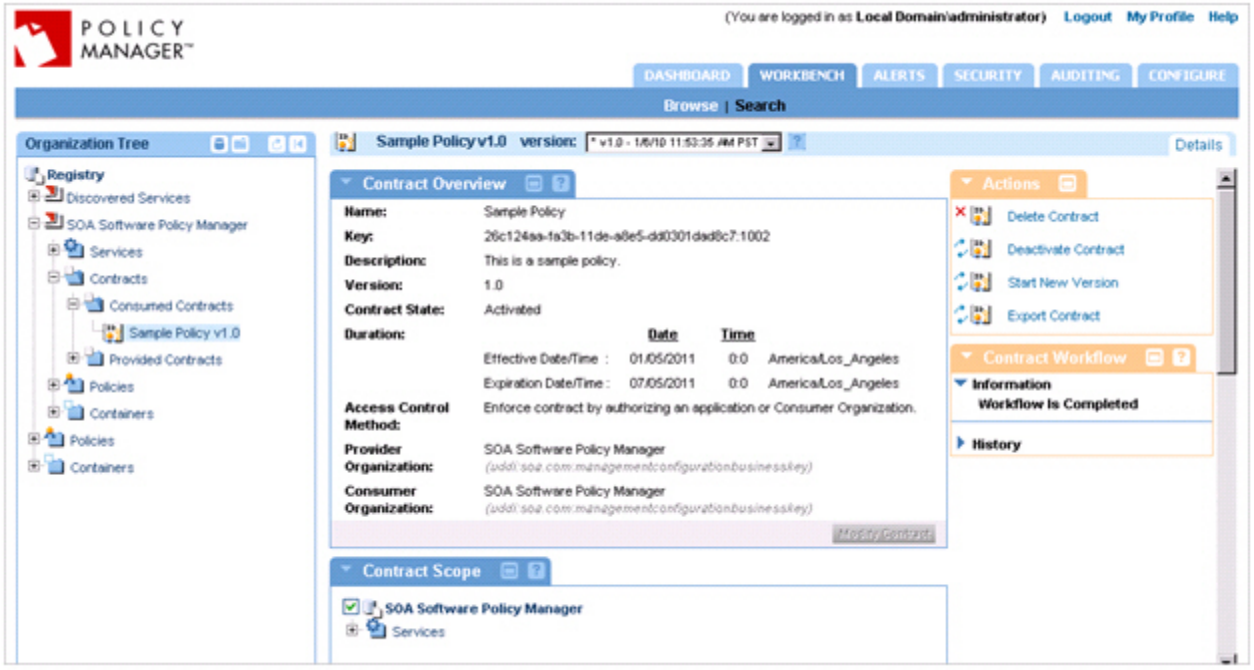


Figure 100: Contract Overview—with Contract State "Activated"

7 To deactivate the contract, navigate to the "Actions Portlet" and click "Deactivate Contract." The following message displays "Do you wish to deactivate this contract?" To deactivate the Contract, click "OK."

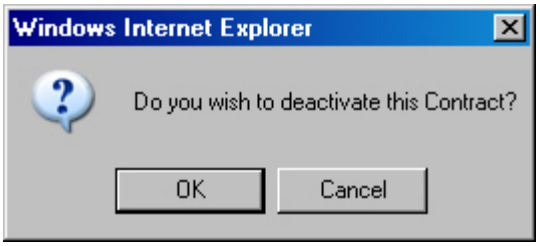


Figure 101: Deactivate Contract Message

8 The "Contract State" in the "Contract Overview Portlet" changes to "Deactivated.

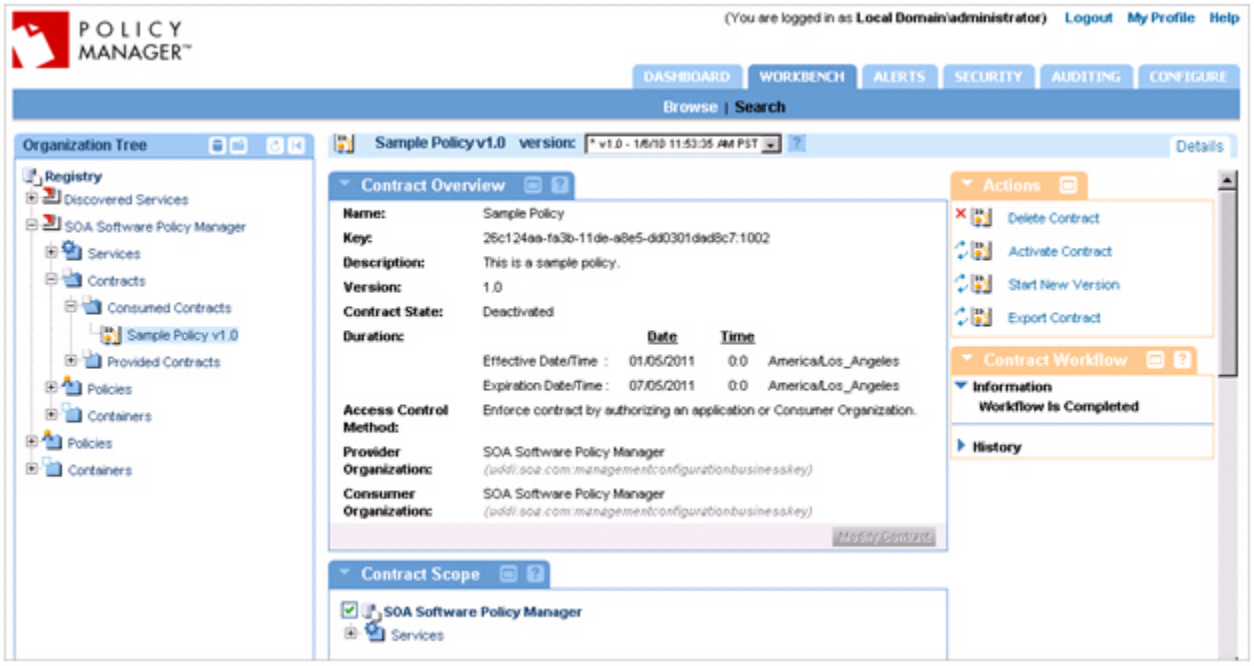


Figure 102: Contract Overview—with Contract State "Deactivated"

9 To activate the contract, navigate to the "Actions" portlet and click "Activate Contract." The following message displays "Do you wish to activate this contract?" To activate the Contract, click "OK."

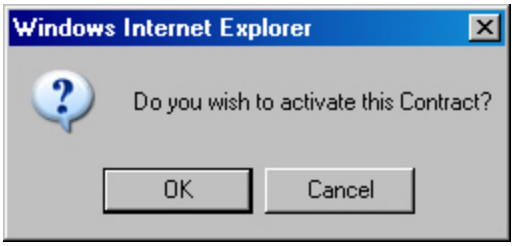


Figure 103: Activate Contract Message

How do I create a Consumed Contract?

A "Contract" is a document that provisions the expected utilization of services. Each contract is configured with an access control method that represents the method a service uses to enforce a contract. The "Request Contract" action (which launches the "Add Contract Wizard") is used to create a contract definition that includes contract details, access control method, and applicable organization assignments.

There are two types of access control that can be assigned to a contract. A service can enforce a contract by authorizing an application or Consumer Organization Identities to use a service, or a service can use a default contract and allow consumer (application and organization) users that do not have a contract explicitly assigned.

The "Request Contract" action is initiated by the "Consumer Organization" and submits a contract usage request to the "Provider Organization" of a service.

> **Note**: The "Request Contract" action utilizes the "Enforce Contract by Authorizing an application or Consumer Organization" Access Control method ONLY. The "Allow consumer application or organization users that do not have a contract explicitly assigned" access control method is NOT supported and is grayed out.

Request Contract

- Launch Add Contract Wizard (via Request Contract)
- Specify Contract Details
- Select Provider Organization
- Select Consumer Organization
- View Completion Summary

To launch add contract wizard:

1 Enter the following navigation path: **Workbench > Browse**. The "Root Organization Summary" screen displays.

2 In the "Organization Tree," double click the name of the organization that includes contracts you would like to view. To view contracts associated with the current organization, click the "Contracts Folder." The "Contracts Summary" screen displays.

3 To view the "Consumed Services Summary" and a listing associated contracts in the "Organization Tree," click the "Consumed Contracts" folder. The "Consumed Services Contracts" screen displays and the folder hierarchy expands and displays a selectable list of contracts.

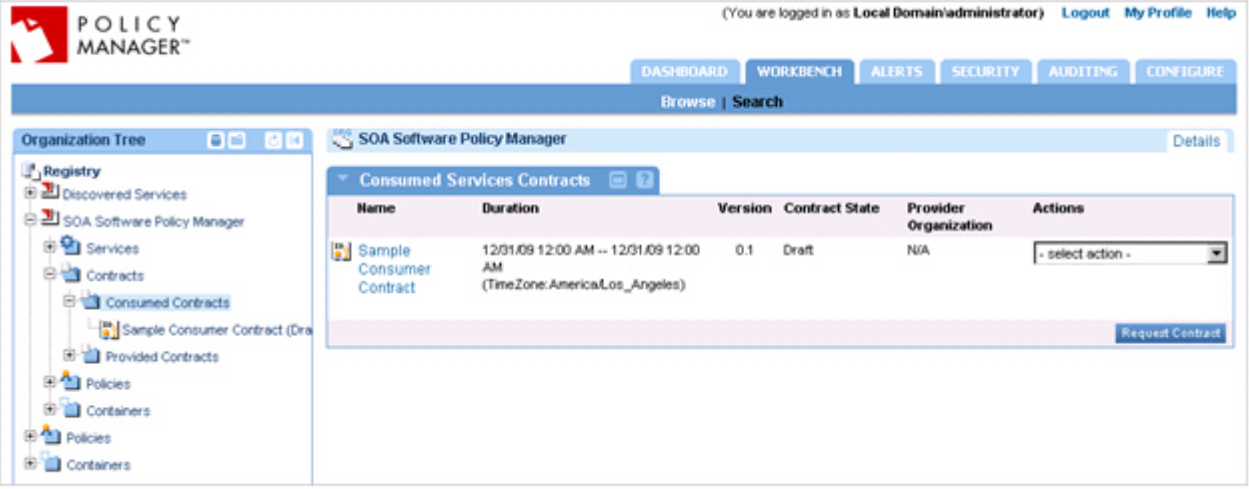


Figure 104: Consumed Services Contracts

4. To submit a contract usage request to the “Provider Organization” of a service, click "Request Contract." The "Add Contract Wizard" launches and displays the "Specify Contract Details" screen.

To specify contract details:

5. The "Specify Contract Details" screen allows you to configure the Name, Description, and Duration (Effective/Expiration Date and Time) of a contract, and select the Access Control Method that will be applied to the contract. The screen is organized into two sections:

Contract Details

Contract Name—A text field that allows you to enter the name of a contract. A Contract Name can consist of multiple words and supports the complete ASCII character set.

Contract Description—A text field that allows you to a description for the contract.

Effective Date/Time—A series of date and time fields that allow you to configure an Effective Date and Time for the current contract.

- Day—A text field that allows you to enter the day within the selected month for the Effective Date. *This field can also be populated automatically using the calendar pop-up.*
- Month—A drop-down list box that allows you to select the month for the Effective Date. *This field can also be populated automatically using the calendar pop-up.*
- Year—A calendar pop-up that allows you to select the Month, Day, and Year for the Effective Date. *The Day and Month can also be populated automatically using the calendar pop-up.*
- Hour—A text field that allows you to enter the Hour (using a 24-hour clock) for the Effective Time.
- Minute—A text field that allows you to enter the Minute for the Effective Time.

Expiration Date/Time—A contract expiration date can be configured with a set expiration date and time, or you can configure it to never expire.

Expired Contract

A series of date and time fields that allow you to configure an Expiration Date and Time for the current contract.

- Day—A text field that allows you to enter the day within the selected month for the Expiration Date. *This field can also be populated automatically using the calendar pop-up.*
- Month—A drop-down list box that allows you to select the month for the Expiration Date. *This field can also be populated automatically using the calendar pop-up.*
- Year—A calendar pop-up that allows you to select the Month, Day, and Year for the Expiration Date. *The Day and Month can also be populated automatically using the calendar pop-up.*
- Hour—A text field that allows you to enter the Hour (using a 24-hour clock) for the Expiration Time.
- Minute—A text field that allows you to enter the Minute for the Expiration Time.

Time Zone—A drop-down list box that allows you to select the Time Zone that applies to both the Effective and Expiration Date.

Never Expires

A radio button that configures the contract to no specific date or time to expire.

Access Control

A radio button that allows you to select one of the following access control methods to apply to the current contract definition:

- Enforce contract by authorizing an application or Consumer Organization.

  Note: The "Request Offer" action does not support the "Allow consumer (application or organization) users that do not have a contract explicitly assigned" access control method and this option is grayed out.

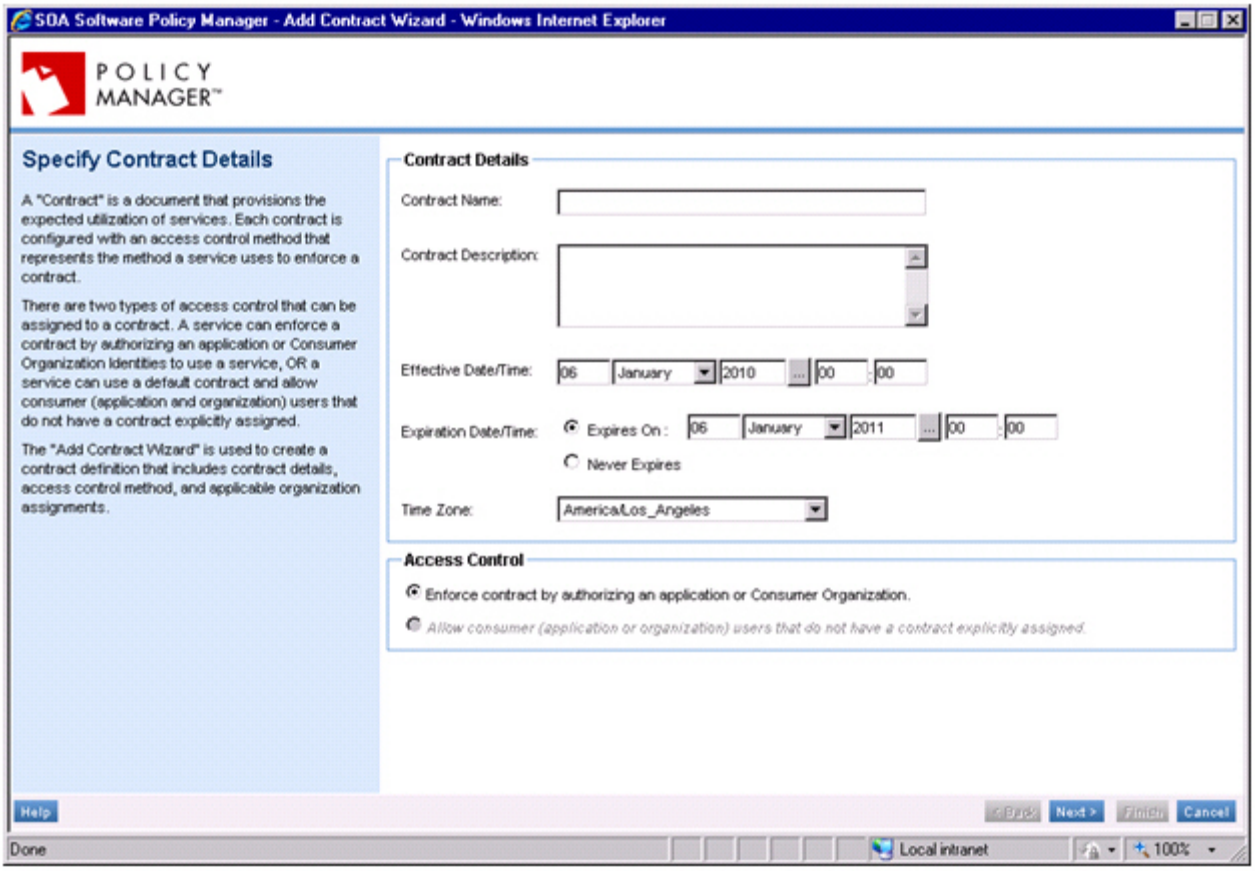


Figure 105: Request Contract (Add Contract Wizard)—*Specify Contract Details*

Enter your selections and click "Next" to continue. The "Select Provider Organization" screen displays.

To select provider organization:

1 The "Select Provider Organization" screen allows you to choose an organization from the "Organization Tree" that is assigned to the new contract as the "Provider Organization." Assigning a "Provider Organization" is optional for the "Request Contract" action.

If the organization you would like to assign to this contract is not available, you can exit this wizard and populate your "Organization Tree" using the "Add Organization Wizard." Note that the "Root Organization" (i.e., Registry) cannot be assigned to a contract.

> **Note**: If you select "Request Contract" and launch the "Add Container Wizard" and are configuring a contract for a specific service (in the Services Folder), the "Select Provider Organization" screen will be omitted, as the service is already a part of the Provider Organization.

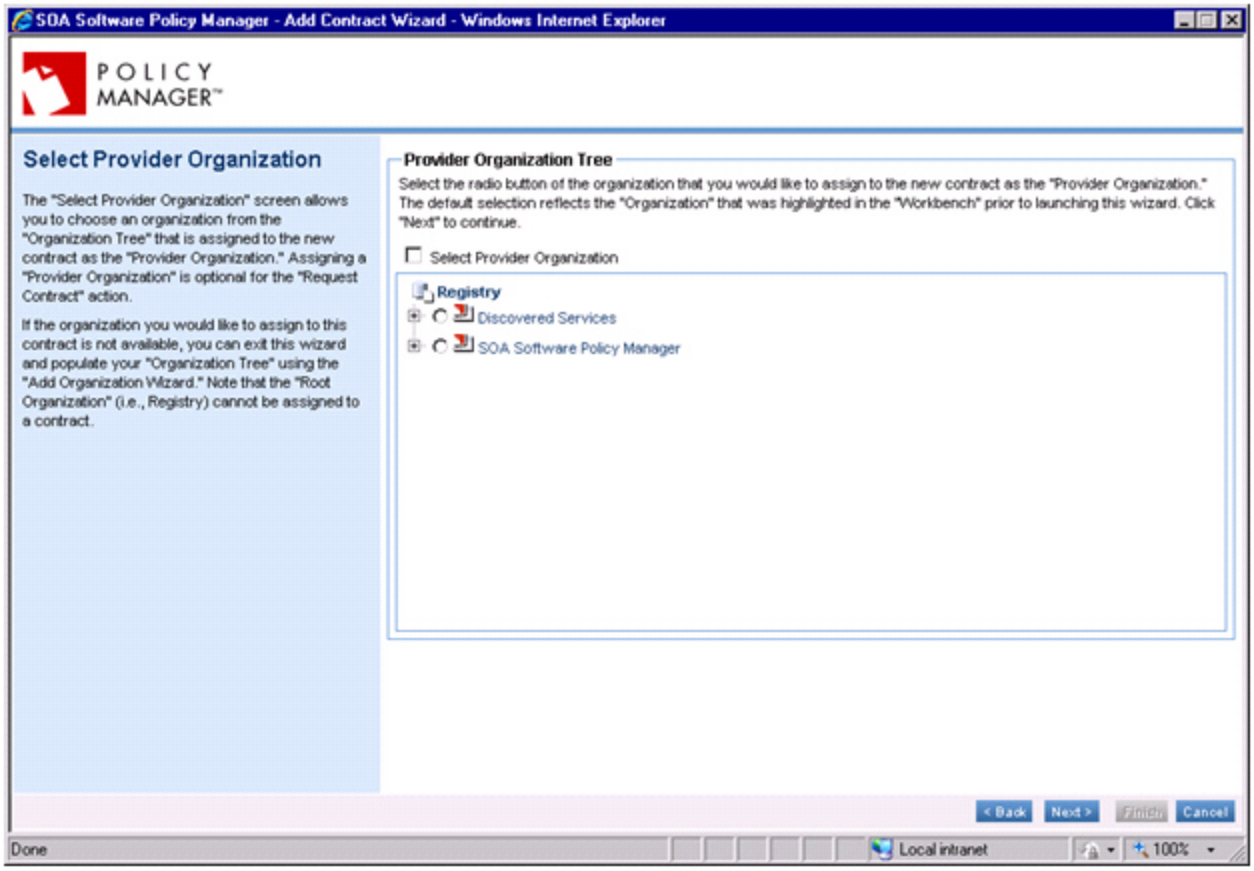


Figure 106: Request Contract (Add Contract Wizard)—*Select Provider Organization*

The "Select Provider Organization" screen allows you to choose an organization from the "Organization Tree" that is assigned to the new contract as the "Provider Organization." Assigning a "Provider Organization" is optional for the "Request Contract" action.

To select consumer organization:

1 The "Select Consumer Organization" screen allows you to choose an organization from the "Organization Tree" that is assigned to the new contract as the "Consumer Organization." Assigning a "Consumer Organization" is required for the "Request Contract" action.

If the organization you would like to assign to this contract is not available, you can exit this wizard and populate your "Organization Tree" using the "Add Organization Wizard." Note that the "Root Organization" (i.e., Registry) cannot be assigned to a contract.

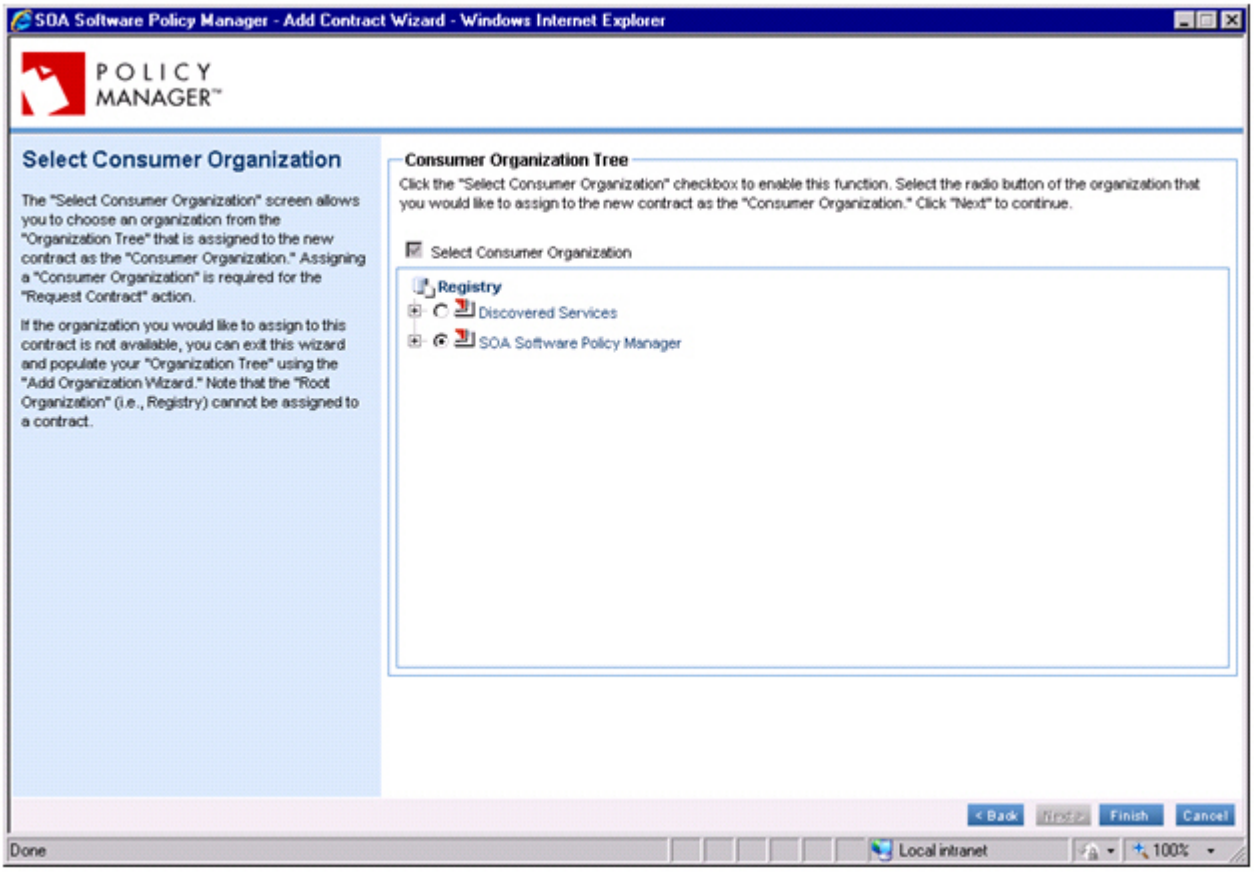


Figure 107: Request Contract (Add Contract Wizard)—*Select Consumer Organization*

Select the radio button of the organization that you would like to assign to the new contract as the "Consumer Organization." Click "Next" to continue. The "Completion Summary" screen displays.

To view completion summary:

1 The "Completion Summary" screen displays key information pertaining to the new contract definition. After reviewing the summary information, click "Close."

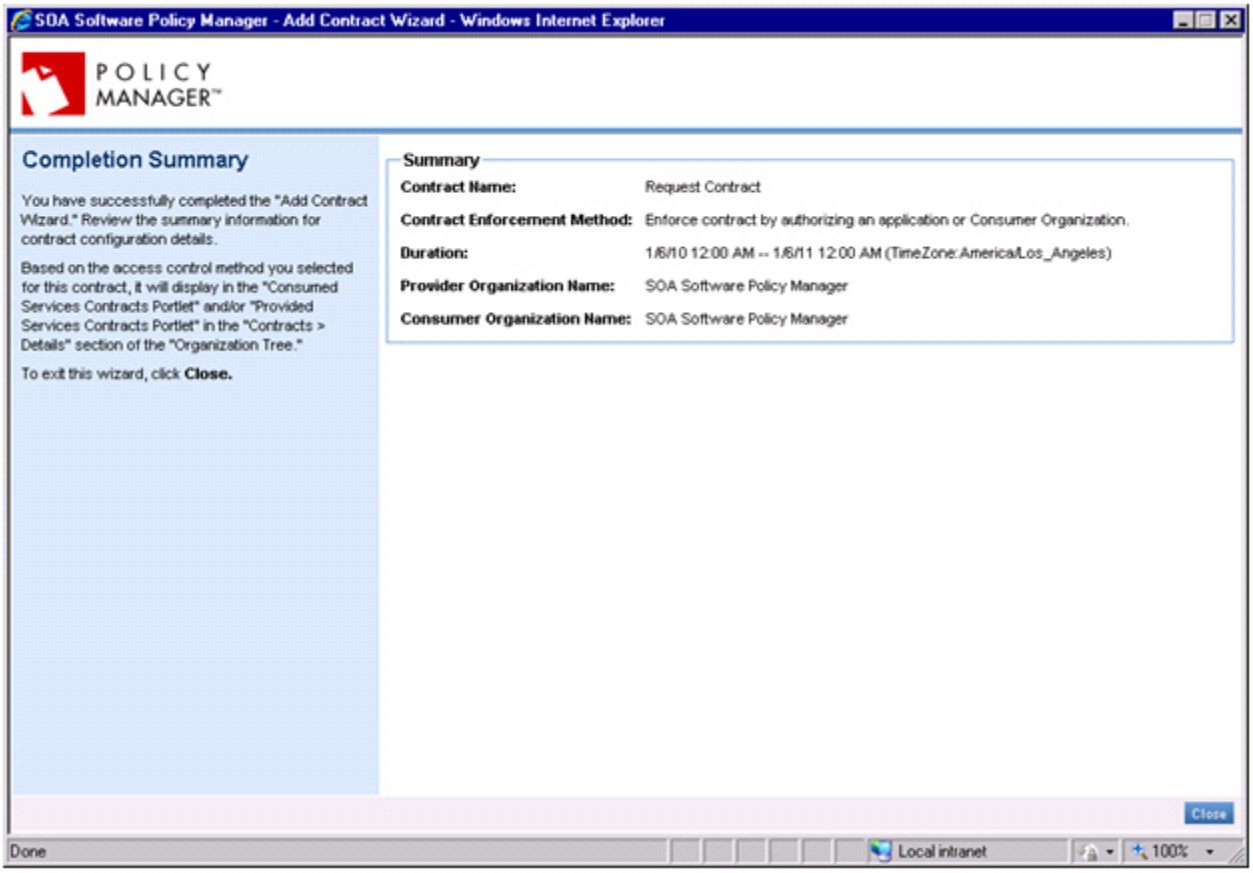


Figure 108: Request Contract (Add Contract Wizard)—*Summary*

How do I define a Contract Scope?

The following table illustrates the available Contract Scope states:

Contract Scope Icon

Org Tree / Icon Presentation	Description

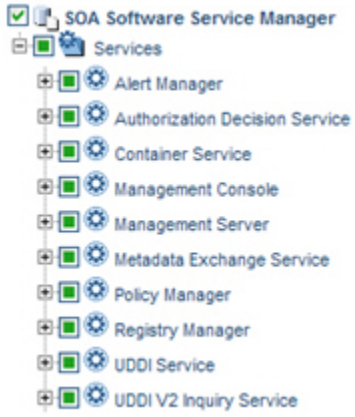


Figure. Full Contract Scope (Default)

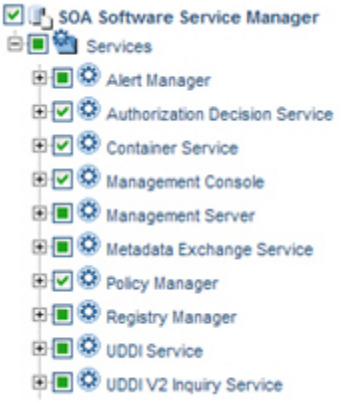


Figure. Full Contract Scope (Intermediate)

Full Contract Scope

This Contract Scope state is the default state when you create a contract and select a "Provider Organization." You can also configure a "Full Contract Scope" if you have customized your existing scope and would like to switch to this state.

The Organization Name includes a checkbox and all services within the organization are selected.

If you uncheck the Organization Name, all services within the Organization become unselected.

Full Contract Scope includes two modes:

- Default Mode—This is the default state "Full Contract Scope" that displays when you initial create a contract. The Organization node is selected in addition to all of the Services using a box selection icon.
- Intermediate Mode—This is an "Intermediate" Contract Scope" that displays when you customize your scope with specific selections and then switch back to "Full Contract Scope." Service selections that were part of your custom configuration will display with checkboxes, and the remainder of Services will display using the box selection icon. This mode allows you to easily identify those Services that were originally part of your custom configuration.

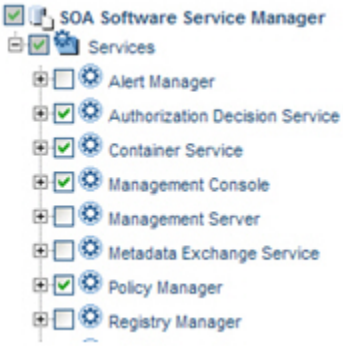


Figure. Partial Contract Scope

Partial Contract Scope

A "Partial Contract Scope" state can be configured by selecting one or more services in the Services Folder. In this state the Services Folder, and Organization Folder are marked automatically with a checkbox icon. This indicates a partial selection, versus the box selection icon which indicates a full selection.

To remove a "Partial Contract Scope," you must unselect all of the selected services within the Services Folder, prior to unselecting the Services Folder and Organization Folder checkboxes.

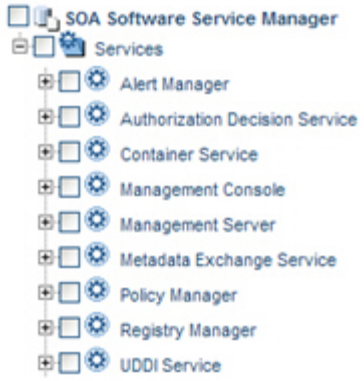


Figure. No Contract Scope

No Contract Scope

Disabling contract scope functionality is performed by unselecting all selected services within the Services Folder, and unchecking the Service Folder and Organization Folder checkboxes.

How do I define a Consumer Identity?

The "Consumer Identities Portlet" includes a read-only "Organization Tree" that displays the selected "Consumer Organization," additionally assigned Organizations, and identities (i.e., user accounts) that have access to the scope of services defined in the current contract. You can navigate the tree hierarchy to view the Organizations and Identity assignments.

- View Consumer Identities
- Manage Consumer Identities

To view consumer identities:

1. Enter the following navigation path: **Workbench > Browse**. The "Root Organization Summary" screen displays.
2. In the Organization Tree, double click the name of the organization that includes contracts you would like to view. To view contracts associated with the current organization, click the "Contracts Folder." The "Contracts Summary" screen displays.
3. To view the "Provided Services Summary" and a listing associated contracts in the "Organization Tree," click the "Provided Contracts" folder. The "Provided Services Contracts" screen displays and the folder hierarchy expands and displays a selectable list of contracts.
4. To view the "Consumed Services Summary" and a listing of associated contracts in the "Organization Tree," click the "Consumed Contracts" folder. The "Consumed Services Contracts" screen displays and the folder hierarchy expands and displays a selectable list of contracts.
5. Select a contract in either the "Provided Contracts" or "Consumed Contracts" folder. The "Contract Details" screen displays for the selected contract.
6. Page down to view the "Consumer Identities Portlet." To view the Organizations and Identities assigned to the current contract, expand the hierarchy tree.

Note: The Consumer Identities Portlet supports the "Enforce Contracts by authorizing an Application or Consumer Organization" access control method ONLY.

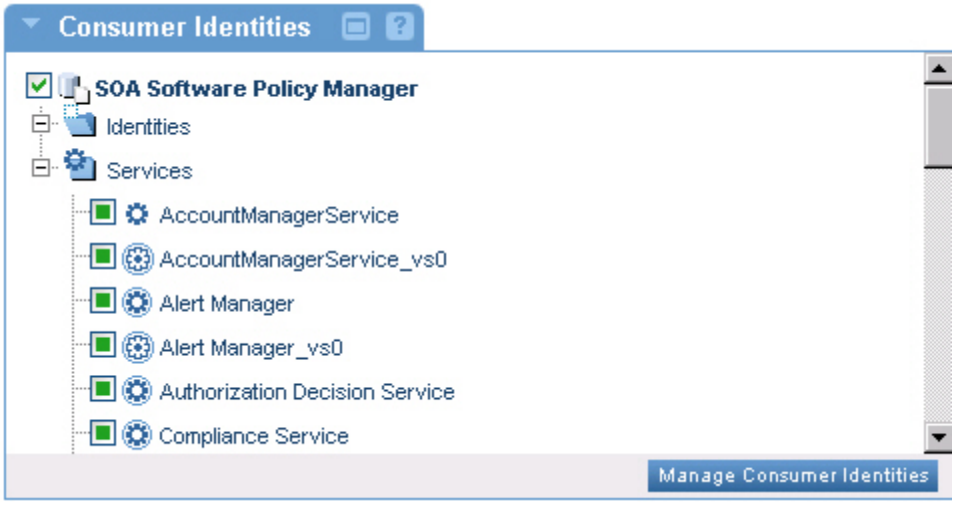


Figure 109: Consumer Identities Portlet—using the "Enforce Contracts by authorizing an Application or Consumer Organization" access control method

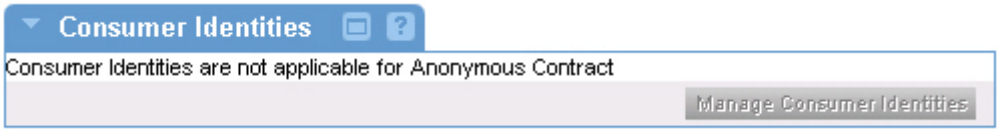


Figure. Consumer Identities Portlet—using the "Allow consumer (application or organization) users that do not have a contract explicitly assigned" is the Access Control Method selected for the current contract" access control method

To manage consumer identities:

1 Enter the following navigation path: Workbench > Browse. The "Root Organization Summary" screen displays.

2 In the "Organization Tree," double click the name of the organization that includes contracts you would like to view. To view contracts associated with the current organization, click the "Contracts Folder." The "Contracts Summary" screen displays.

3 To view the "Provided Services Summary" and a listing associated contracts in the "Organization Tree," click the "Provided Contracts" folder. The "Provided Services Contracts" screen displays and the folder hierarchy expands and displays a selectable list of contracts.

4 To view the "Consumed Services Summary" and a listing of associated contracts in the "Organization Tree," click the "Consumed Contracts" folder. The "Consumed Services Contracts" screen displays and the folder hierarchy expands and displays a selectable list of contracts.

5 Select a contract in either the "Provided Contracts" or "Consumed Contracts" folder. The "Contract Details" screen displays for the selected contract.

6 Page down to view the "Consumer Identities Portlet." To view the Organizations and associated Identities that have access to the web services defined in the scope of the current contract, expand the hierarchy tree.

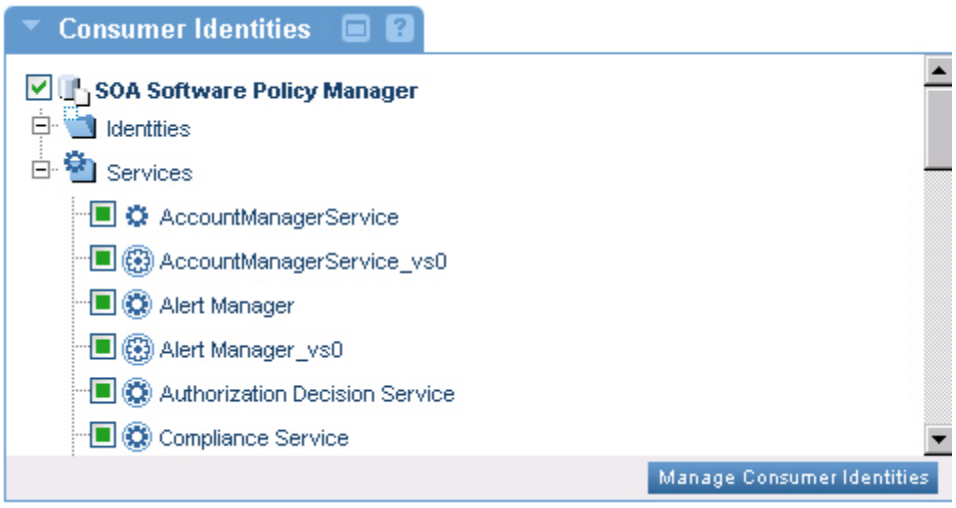


Figure 110: Contract Scope Portlet

7 To update the consumer identities configuration, click "Manage Consumer Identities." The "Manage Consumer Identities" screen displays.

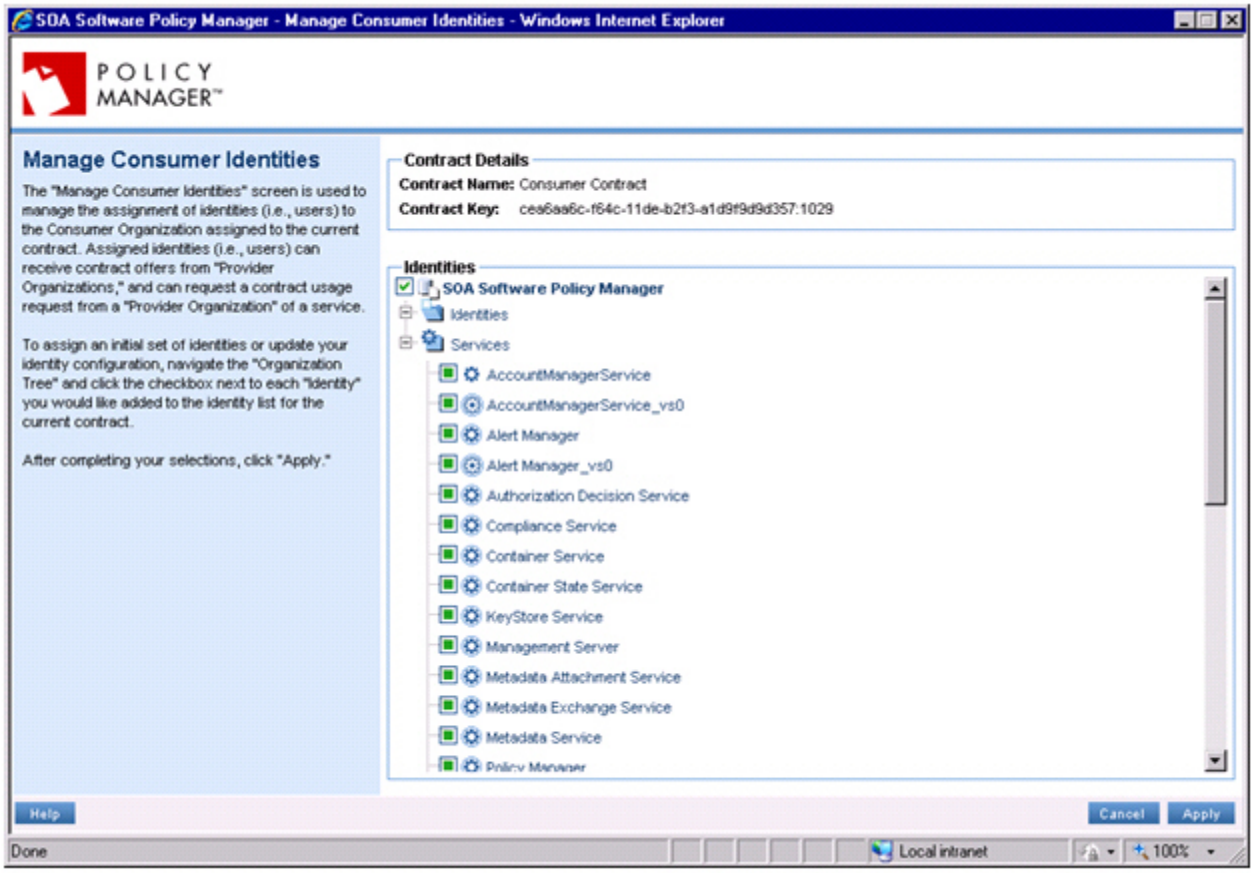


Figure 111: Manage Consumer Identities

8 Navigate the hierarchy tree and configure the Organizations and associated Identities by checking and unchecking the relevant Organization and Identities. After you have completed your selections, click "Apply." The "Manage Consumer Identities" screen closes and the "Consumer Identities Portlet" updates with the new selections.

How do I add Contract Metadata?

The "Add Metadata Attachment (Services)" screen allows you to add two different attachment types.

- External Link

  You can add an External Link by specifying a URL address. This attachment type is represented by the icon.

- Document Attachment

  You can add a Document Attachment. Document attachments can be any artifact that provides key information about the service (i.e., design documents, UML, test messages, sample code, etc.) This attachment type is represented by the icon.

To add a metadata attachment (contract):

1. To add a metadata attachment (contract), enter the following navigation path: **Workbench >** Organization > Contracts. The "Contracts Summary" screen displays and presents a list of contracts associated with the current organization.
2. In the "Organization Tree," select a contract in either the "Provided Contracts" or "Consumed Contracts" folder that you would like to add attachments to. The "Contract Details" screen displays.
3. Page down to the "Metadata Portlet." The "Attachments" section of the "Metadata Portlet" is used for adding and managing "External Reference" and "Document" attachments for the current contract.

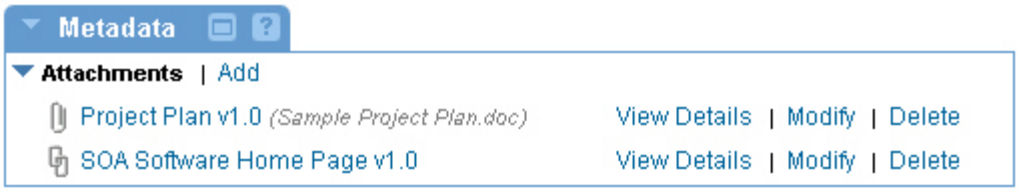


Figure 112: Metadata Portlet

4. To add an attachment to the current contract. Click "Add." The "Add Metadata Attachment (Contract)" screen displays.

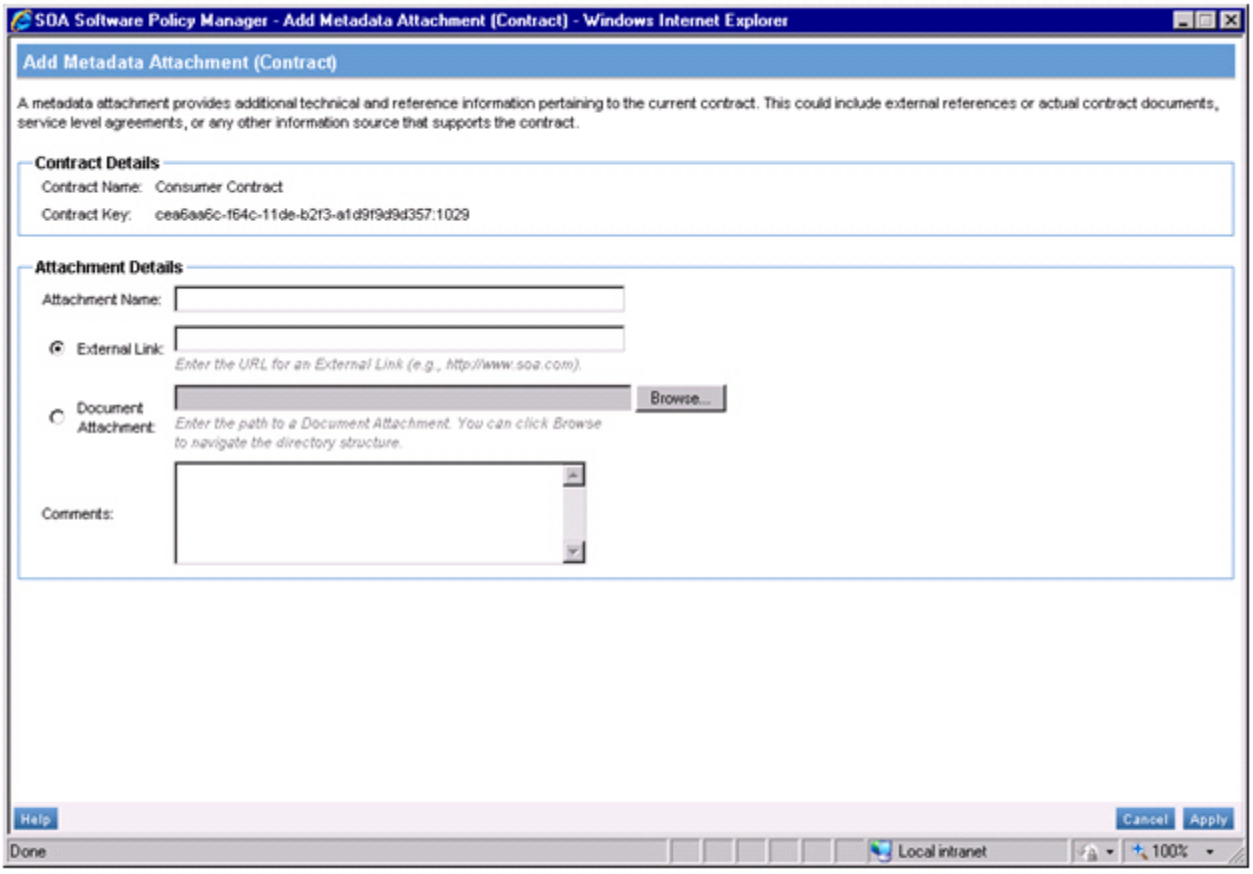


Figure 113: Metadata Portlet—Add Metadata Attachment (Contract)

5 A document attachment is composed of three elements.

- Attachment Name

The "Attachment Name" is a user-defined name that represents the display name that will be presented on the "Metadata Portlet" screen. This "Attachment Name" is linked to the actual "Reference" or "Document" attachment that you configure. Each "Attachment Name" is followed by the name of the actual attachment (in *italics*).

- Attachment Type

You can add two different attachment types:

  - External Link

A radio button / text box combination that allows you to add an External Link by specifying a URL address. This attachment type is represented by the icon.

  - Document Attachment

A radio button / text box combination that allows you to add a Document Attachment by browsing the directory and selecting a document. Document attachments can be any artifact that provides key information about the service (i.e., design documents, UML, test messages, sample code, etc.)

This attachment type is represented by the icon.

- Comments

A text box that allows you to specify details pertaining to the document attachment. Note that the "comment" information will display in the "View Metadata Attachment" screen and should be unique to the specific document version.

6 To add a document attachment perform the following steps:

- External Link

To add an "External Link" attachment, enter the "Attachment Name," click the "External Link" radio button, and enter the URL for the link into the text box. Enter any description information about the attachment in the "Comments" section. To add the attachment to the current service, click "Apply." The attachment is added and displays in the "Attachment" section of the "Metadata Portlet.

- Document Attachment

To add an "Document" attachment, enter the "Attachment Name," click the "Document Attachment" radio button, click Browse to navigate the directory structure and select a file you would like to add as a document attachment. Enter any description information about the attachment in the "Comments" section. To add the attachment to the current service, click "Apply." The attachment is added and displays in the "Attachment" section of the "Metadata Portlet.

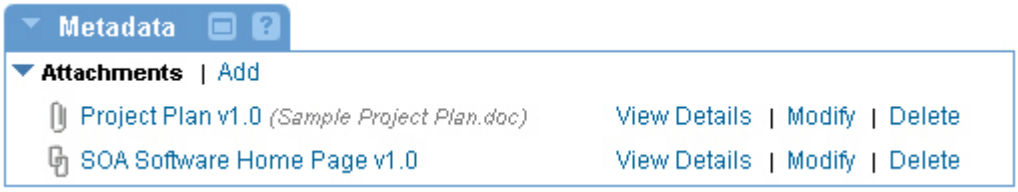


Figure 114: Metadata Portlet—Attachments (with External Link and Document Attachment)

How do I attach a QoS Policy to a Contract?

The "Manage QoS Policy Attachments (Contracts)" screen allows you to attach policies to the current contract. For more information about Quality of Service (QoS) policies, refer to About QoS Policies.

Note: This function is available when Contract Workflow is in progress. When Workflow is completed the "Manage" button in the "QoS Policy Attachments" portlet is disabled.

To manage qos policy attachments:

1. Enter the following navigation path: Workbench > Browse. The "Root Organization Summary" screen displays. Select an "Organization" in the Organization Tree. The "Sub-Organization Details" screen displays.
2. Select the "Contracts" folder. The "Contracts Summary" screen displays. Double click the name of the contract you would like to manage policies for. The "Contract Details" screen displays.
3. Page down to view the "QoS Policy Attachments (Contracts)" portlet.

Figure 115: QOS Policy Attachments (Contracts)—*without QoS Policy Attachments*

4 To configure QoS policy attachments for the current Contract, click "Manage."

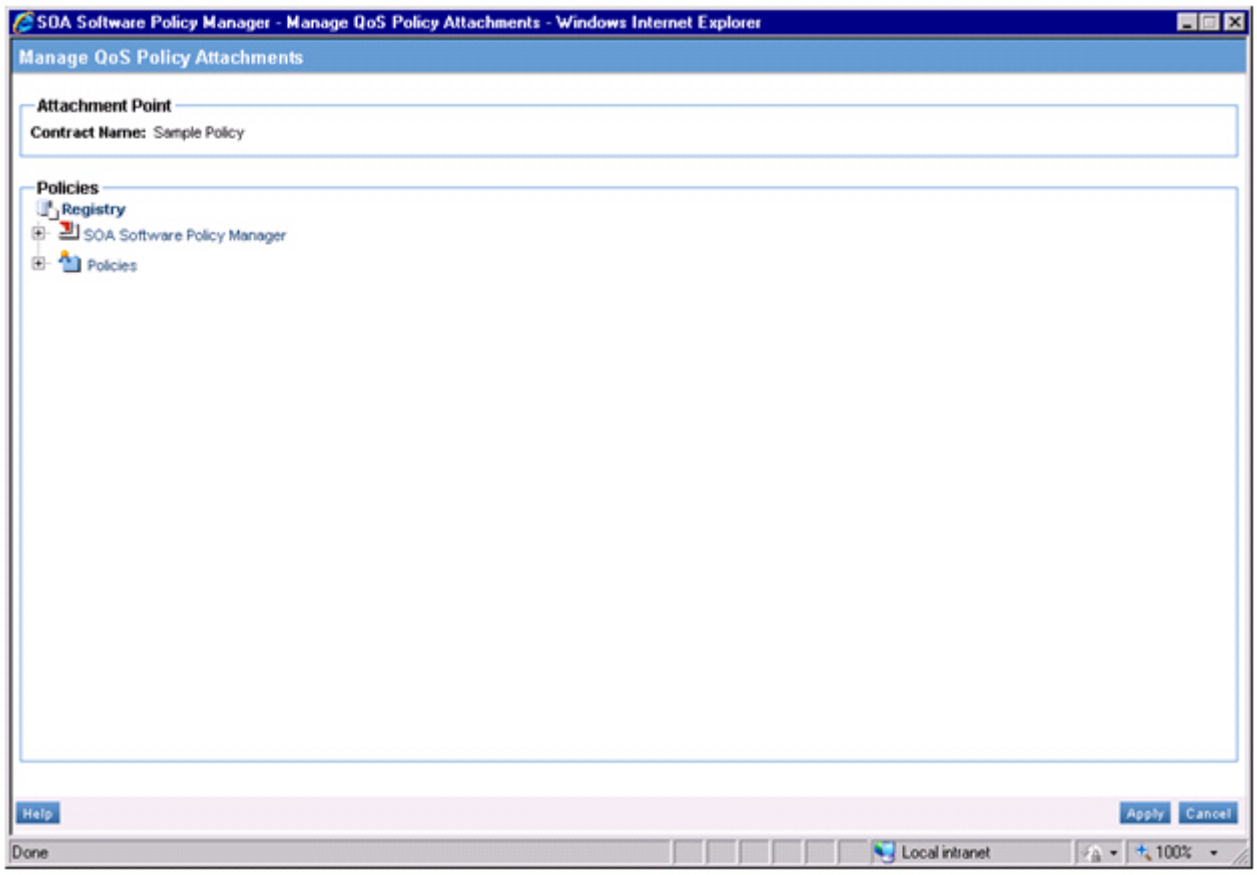


Figure 116: Manage QoS Policy Attachments (Contract Details)

5 To attach policies to the current contract, navigate to a "QoS Policies" folder that contains QoS Policy definitions you would like to add to the current contract and click the checkbox next to each policy you would like to attach. After completing your selections, click "Apply." The selected policies display in the "QoS Policy Attachments (Contracts)" portlet.

Figure 117: QoS Policy Attachments (Contracts)—*with QoS Policy Attachments*

Chapter 10 | Managing Workflow

Managing workflows involves two processes:

- Global Workflow Task Portlet

  In the "Workflow Tasks Portlet" located in the "Root Organization" you can globally manage urgent *service* and *contract* "Workflow Tasks" in the current organization or within the complete Organization Sub-Tree.

- Workflow Control Portlet

  In the "Workflow Control Portlet" you can manage the state of a particular service or contract.

Workflow Control Portlet Functionality

The "Workflow Control Porlet" is associated with each Workbench Object (*service* or *contract*). In the Policy Manager "Workbench" these portlets are referred to as "Service Workflow Portlet," and "Contract Workflow Portlet." They provide the following functionality:

- Lists all the available Workflow actions for a service or contract object.
- View or Modify options are permitted based on defined permissions in the "Workflow Definition"
- Provides "Comment" feature for commenting details of a specific action. Comment is posted to the "History" section of the "Workflow Control Portlet" after an action is submitted.

Workflow Task Portlet

The "Workflow Task Portlet" is located in a "Workbench" Organization Object and displays the Workbench actions that have executed and that are marked as needing immediate attention in the "Workflow Definition." In the "Root Organization" or Sub-Organizations this portlet is referred to as the "Workflow Tasks Portlet.

This portlet provides an overview of workflow activities associated with services and provisioned contracts at the Root Organization level and Sub-Organizations. This portlet can be used by administrators who are responsible for overseeing and managing the quality of services published in the repository (i.e., Registry). After assessing the general state of Workflow Tasks at the Root Organization level, filtered tasks within each Organization can be managed in the associated Services and Contracts Folders.

To view the special Workflow Tasks, you must have the "Include Subtree" checkbox selected in the "Workflow Tasks Portlet" located in the Root Organization. You can review the posted service by selecting it in the "Organization Tree." To approve the pending "State" click the service "Name."

Note: The default setting for the "Include Subtree" option is unchecked. If your Policy Manager deployment includes a high volume of services, and your "Workflow Definition" includes one or more "Immediate Attention" entries, the "Include Subtree" option located in the "Root

Organization" could introduce performance problems as all actions from every service will post to this portlet. See Special Workflow State and Workflow Task List for more information on how these actions are configured in the "Workflow Definition."

```
<step id="200" name="Pending Approval!!!">

<step id="1400" name="Pending Deprecation!!!">
```

Figure 118: Immediate Attention Workflow Entries (Examples)

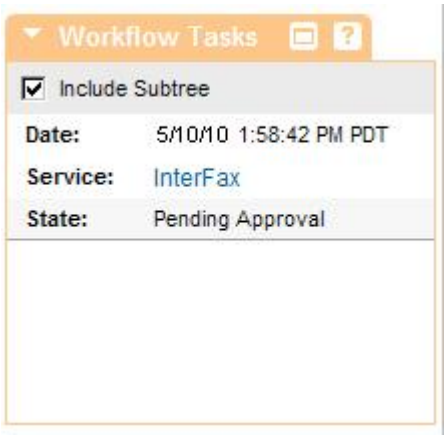


Figure 119: Workflow Tasks Portlet—*for Organization Objects*

Services Workflow Portlet

The "Services Workflow Portlet" is a "Control Portlet" that executes the "Workflow Definition Template" assigned in the "Configure > Workflow > Services Workflow Administration" section of the "Management Console.

The "Service Workflow Portlet" is organized into three sections:

Information

State—Represents the "Workflow State" currently in progress. All of the "steps" defined in your "Workflow Definition" are presented in the "Information" section of the "Workflow Portlet" in the "State" field.

```
- <steps>
  - <step id="100" name="Draft">
    - <actions>
      - <action id="101" name="Submit for Approval">
```

Figure 120: <steps> in a Workflow Definition

Owner—Displays the domain and name of the user that has permission to participate in the current Workflow Task. Workflow access control is configured using the "${caller}" variable. This variable allows you to customize access permissions for Workflow "action." If you do not specify a domain and username and use the default "${caller}" variable, the access is granted for the current logged in user.

Workflow Is Completed—When all of the steps in a Workflow Definition have successfully executed, this message displays.

```
owner="${caller}"
```

Figure 121: Example of "${caller}" variable in a Workflow Definition.

Workflow Actions

Comments—A text field that allows you to add comment text for the "Action" that is about to be executed. This comment text will be added to the Workflow "History" and associated with the selected "Action."

Actions—Displays the name of the "Action" to be executed. A single "Action" can be displayed or two actions that represent a "choice" can be displayed (e.g., Promote / Reject).

History

Displays a read-only list of key information associated with a specific Workflow "Action."

Date/Time—Displays the date and time the Workflow "Action" was executed.

Status—Displays the Workflow "Step" that the "Action" occurred in.

Action—Displays the Workflow "Action" that was executed.

User—Displays the owner (e.g., ${caller}) who executed the "Action."

Comment—Displays details related to the execution of the "Action."

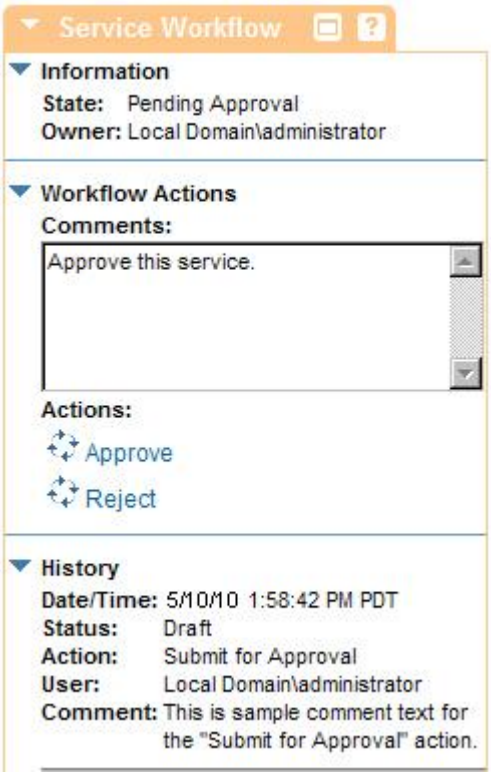


Figure 122: Workflow Control Portlet—*Services Workflow*

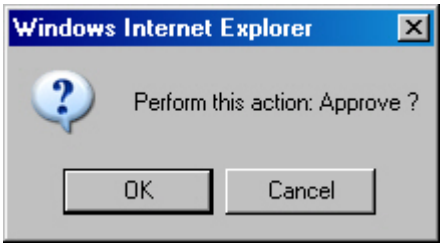


Figure 123: Execute Action Message

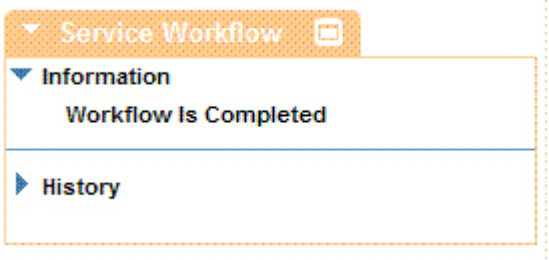


Figure 124: Services Workflow—*Workflow is Completed*

Contract Workflow Portlet

The "Contract Workflow Portlet" is a "Control Portlet" that executes the "Workflow Definition Template" assigned in the "Configure > Workflow > Contracts Workflow Administration" section of the "Management Console.

The "Contract Workflow Portlet" is organized into three sections:

Information

State—Represents the "Workflow State" currently in progress. All of the "steps" defined in your "Workflow Definition" are presented in the "Information" section of the "Workflow Portlet" in the "State" field.

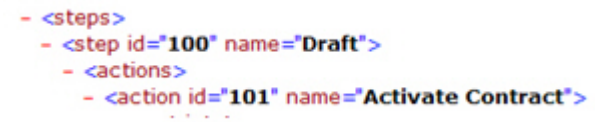


Figure 125: <steps> in a Workflow Definition

Owner—Displays the domain and name of the user that has permission to participate in the current Workflow Task. Workflow access control is configured using the "${caller}" variable. This variable allows you to customize access permissions for Workflow "action." If you do not specify a domain and username and use the default "${caller}" variable, the access is granted for the current logged in user.

Workflow Is Completed—When all of the steps in a Workflow Definition have successfully executed, this message displays.

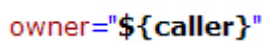


Figure 126: Example of "${caller}" variable in a Workflow Definition.

Workflow Actions

Comments—A text field that allows you to add comment text for the "Action" that is about to be executed. This comment text will be added to the Workflow "History" and associated with the selected "Action."

Actions—Displays the name of the "Action" to be executed. A single "Action" can be displayed or two actions that represent a "choice" can be displayed (e.g., Promote / Reject).

History

Displays a read-only list of key information associated with a specific Workflow "Action."

Date/Time—Displays the date and time the Workflow "Action" was executed.

Status—Displays the Workflow "Step" that the "Action" occurred in.

Action—Displays the Workflow "Action" that was executed.

User—Displays the owner (e.g., ${caller}) who executed the "Action."

Comment—Displays details related to the execution of the "Action."

```
-<workflow>
-<steps>
-<step id="100" name="Draft">      (Represented as "State" in "Workflow Portlet")
   -<actions>
     +<action id="101" name="Activate Contract">   (Represented as "Action" in Workflow Portlet)
+<step id="200" name="Active">

     (When the last target step (i.e., step id) in a Workflow Definition executes, the "Information" section of the "Workflow
     Portlet" displays "Workflow Is Completed."

     At this point Service Manager "Workflow" assumes processing and you can manage the contract based on your
     requirements.

     </steps>
     </workflow>
```

Figure 127: Annotated section of Workflow Definition

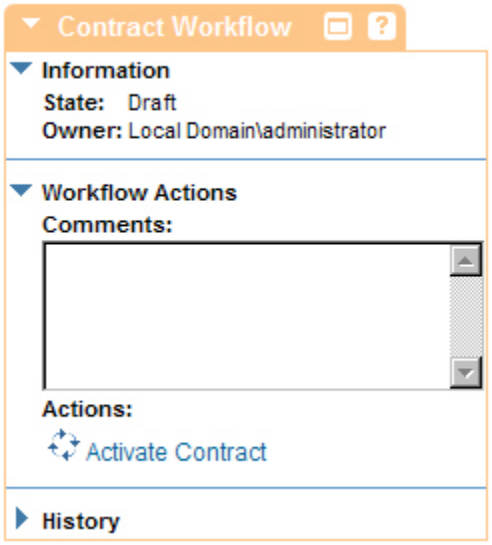


Figure 128: Workflow Control Portlet—*Contracts Workflow*

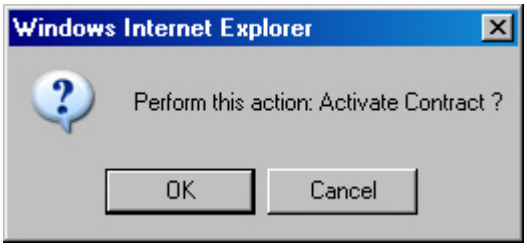


Figure 129: Execute Action Message

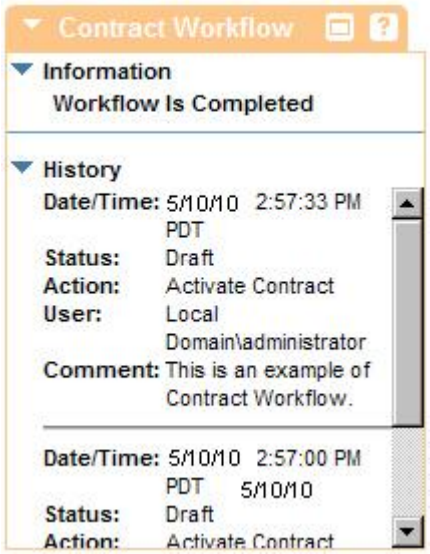


Figure 130: Contracts Workflow—*Workflow is Completed*

Chapter 11 | Using Delegates

Delegate Handler

To deploy a Delegate Handler a handler entry is made in the client-config.wsdd file of the Axis client. The handler is then referenced as the pivot handler. For example:

```
01) <deployment ...>
02)   ...
03)   <handler name="soapivot" type=""java:com.soa.delegate.axis.LocalAxisPivotHandler>
04)     ...
05)   </handler>
06)   <transport name="http" pivot="soapivot"/>
07)   ...
08) </gwap:Report>
```

Lines 03 – 05 hold the LocalAxisPivotHandler declaration and configuration parameters. Line 06 defines the LocalAxisPivotHandler as the Axis "pivot" handler.

Handler Properties

The configuration parameters of the Handler are described using name/value pair elements within the handler element. Some of the parameters describe the scope of the handler and are described in the following table. These configuration parameters apply to both the LocalAxisPivotHandler and the RemoteAxisPivotHandler.

Parameter Name	Parameter Description
service_ns	Namespace of the service the handler is processing messages for. This should be the same namespace as the service element in the WSDL document describing the service in Policy Manager. This parameter (in conjunction with the service_name parameter) or the binding_identifier parameter is required.
service_name	Unqualified name (local part of the qualified name) of the service the handler is processing messages for. This should be the same local part as the service element in the WSDL document describing the service in Policy Manager. This parameter (in conjunction with the service_ns parameter) or the binding_identifier parameter is required.

binding_identifier	Unique identifier of the service to be used as an alternative to the service_ns and service_name parameters. See the Policy Manager Online Help for more information on binding identifiers. This parameter or the service_ns and service_name parameters are required.
interface_ns	Namespace of the interface the handler is processing messages for. This should be the same namespace as the portType element in the WSDL document describing the service in Policy Manager. This parameter is optional.
interface_name	Unqualified name (local part of the qualified name) of the interface the handler is processing messages for. This should be the same local part as the portType element in the WSDL document describing the service in Policy Manager. This parameter is optional.

In the remote deployment, the com.soa.delegate.axis.RemoteAxisPivotHandler is installed as an Axis "pivot" handler. The RemoteAxisPivotHandler will relay messages between the Axis engine and the remote SOA Container which will perform the actual message transportation to the service implementation. If the remote SOA Container is a standalone container, the SOA Software Delegate Access Point feature is installed to perform the message processing. If the remote SOA Container is a J2EE container, the SOA Software Delegate feature is installed to perform the message processing as it can act as an access point as well as a collocated delegate. The deployment of a remote access point is ideal for Axis clients that are not deployed in a J2EE container as the SOA Container required to run the Delegate feature requires a J2EE container or its own container to function.

In addition to the parameters described for the LocalAxisPivotHandler, the RemoteAxisPivotHandler has an additional parameter named proxy.address. This parameter holds a URL to the remote Delegate container.

The following is an example of the parameter usage.

```
01) <handler ...>
02)   <parameter name="service_ns" value="http://soapinterop.org/"/>
03)   <parameter name="service_name" value="echo"/>
04)   <parameter name="interface_ns" value="http://soapinterop.org/"/>
05)   <parameter name="interface_name" value="InteropTestPortType"/>
06)   <parameter name="proxy.address" value="http://host1:9905/soapdelegate"/>
07)   ...
08) </handler>
```

Identity Credentials

The Delegate supports multiple forms of identity credentials that can be provided by the client to be used in messages. The format of the credentials included in the messages is dictated by the policies of the endpoint being communicated with.

Multiple sets of credentials may be required for communications with an endpoint. For example, the X.509 certificate for the client application may be required for HTTPS communication and a username token for the logged in user may be required in the SOAP message. These credentials are distinguished by a customizable categorization scheme described using URI's. Two standard categories that also serve as examples are urn:org:federatedgovernance:security:subject-category:consumer and urn:org:federatedgovernance:security:subject-category:enduser.

The Axis Delegate integrates with the client to obtain credentials through JAAS Callback Handlers. Some out-of-the-box Callback Handlers are provided with the Axis Delegate, however Callback Handlers developed by third parties can also be used. The Delegate instantiates the Callback Handlers using Callback Handler Factories. Each credential that will be provided to the AxisPivotHandler will be associated with a Callback Handler Factory. The factory creates the Callback Handler that will provide the credentials to the Delegate. Each Callback Handler Factory may have its own set of parameters. For example, a factory that creates a Callback Handler to retrieve X.509 credentials from a Java KeyStore will have parameters for the alias to retrieve information for and the password to the keystore.

The configuration of Callback Handler Factories and their association with credential categories is described using additional parameters in the handler element. First, credential categories are listed in a parameter. These categories are identified with symbolic names, not URI's. For example, instead of listing "“urn:org:federatedgovernance:security:subject-category:enduser," simply "user" may be listed. All authentication credentials parameters are prefixed with "auth." The credential categories are listed using a parameter named "auth.categories." The categories are listed in the parameter value delimited by commas.

For each category a set of parameters will be supplied. The parameters for each category listed in "auth.categories" are prefixed with auth.<category name>. For example, for the "user" category, all parameters are prefixed with "auth.user." All credential categories must have at a minimum the "uri" and "factory" parameters. The "uri" parameter specifies the literal category name as a URI, such as "urn:org:federatedgovernance:security:subject-category:enduser." The "factory" parameter specifies the Callback Handler Factory class that will be used to obtain the credentials. Any number of additional parameters specific to each factory can also be listed. These additional parameters must be prefixed with the same prefix as the factory parameter.

Three Callback Handler Factories are provided with the Axis Delegate, com.soa.delegate.auth.callback.handler.X509KeyStoreCBHandlerFactory, com.soa.delegate.auth.callback.handler.UserPropertiesCBHandlerFactory, and urn com.soa.delegate.auth.callback.handler.AgentSubjectPropertiesCBHandlerFactory. The UserPropertiesCBHandlerFactory retrieves a username and password from properties, or parameters, named "username" and "password" respectively. The following is an example of a UserPropertiesCBHandlerFactory configuration.

```
01) <handler ...>
02)   ...
03)   <parameter name="auth.categories" value="user"/>
04)   <parameter name="auth.user.uri" value="urn:org:federatedgovernance:security:subject-category:enduser"/>
05)   <parameter name="auth.user.factory"
value="com.soa.delegate.auth.callback.handler.UserPropertiesCBHandlerFactory"/>
06)   <parameter name="auth.user.username" value="jane"/>
07)   <parameter name="auth.user.password" value="doe"/>
08)   ...
```

```
09) </handler>
```

The X509KeyStoreCBHandlerFactory retrieves an X.509 private key and certificate from a Java keystore. It takes the following parameters:

- keystore.path – Path to the keystore to retrieve the information from.
- keystore.alias – Alias of the private key certificate pair.
- keystore.password – Password required to open the keystore.

The following is an example of an X509KeyStoreCBHandlerFactory configuration.

```
01) <handler ...>
02)   ...
03)   <parameter name="auth.categories" value="app"/>
04)   <parameter name="auth.app.uri" value="urn:org:federatedgovernance:security:subject-category:consumer"/>
05)   <parameter name="auth.app.factory"
value="com.soa.delegate.auth.callback.handler.X509KeyStoreCBHandlerFactory"/>
06)   <parameter name="auth.app.keystore.path" value=".keystore"/>
07)   <parameter name="auth.app.keystore.alias" value="jdoe"/>
08)   <parameter name="auth.app.keystore.password" value="password"/>
09)   ...
10) </handler>
```

The AgentSubjectPropertiesCBHandlerFactory retrieves credentials from a JAAS Subject that was created from an Agent as a result of authenticating credentials in an inbound message. The AgentSubjectPropertiesCBHandlerFactory has only one parameter named "category." This parameter lists the literal credential category to pull from the JAAS Subject. The following example illustrates using the inbound consumer credentials for the outbound enduser.

```
01)  <handler ...>
02)   ...
03)   <parameter name="auth.categories" value="agent"/>
04)   <parameter name="auth.agent.uri" value="urn:org:federatedgovernance:security:subject-category:enduser"/>
05)   <parameter name="auth.agent.factory"
value="com.soa.delegate.auth.callback.handler.AgentSubjectPropertiesCBHandlerFactory”/>
06)   <parameter name="auth.agent.category" value=":org:federatedgovernance:security:subject-
category:consumer"/>
07)   ...
08) </handler>
```

In the above example, line 04 states that the credentials retrieved from this Callback Handler will be used for the enduser credentials. Line 06 states the consumer credentials in the Subject created by the Agent should be retrieved for this purpose.

In the Axis architecture there is only one "pivot" handler for all services consumed by the client. As described previously, the Delegate handlers require parameters that provide the context for the service being consumed. Specifying these parameters in the "pivot" handler will restrict the client to consuming only the one service. If the client consumes more than one service, the SOA pivot handlers must be used in conjunction with the com.soa.delegate.axis.AxisServiceHandler. The sole purpose of the AxisServiceHandler is to provide the consumed service context to the pivot handler so that the pivot handler does not have to be tied to just one consumed service. When deployed the AxisServiceHandler will be configured with the following parameters already described above and the pivot handler

(LocalAxisPivotHandler or RemoteAxisPivotHandler) will not: service_ns, service_name, binding_identifier, interface_ns, interface_name, and all authentication parameters. If preferred, the authencation parameters can be defined for the pivot handler even when the service handler is present and configured with service context parameters.

The AxisServiceHandler is configured in the Axis client wsdd file within the “service” element. The following is an example of the configuration.

```
01) <deployment ...>
02)   ...
03)   <handler name="soapivot" type="java:com.soa.delegate.axis.LocalAxisPivotHandler>
04)     ...
05)   </handler>
06)   <transport name="http" pivot="soapivot"/>
07)   <service name="..." provider="...">
08)     ...
09)     <requestFlow>
10)       <handler name="soaservice" type="java:com.soa.delegate.axis.AxisServiceHandler>
11)         <parameter name="binding_identifier" value="myservice"/>
12)       </handler>
13)     </requestFlow>
14)     ...
15)   </service>
16)   ...
17) </gwap:Report>
```

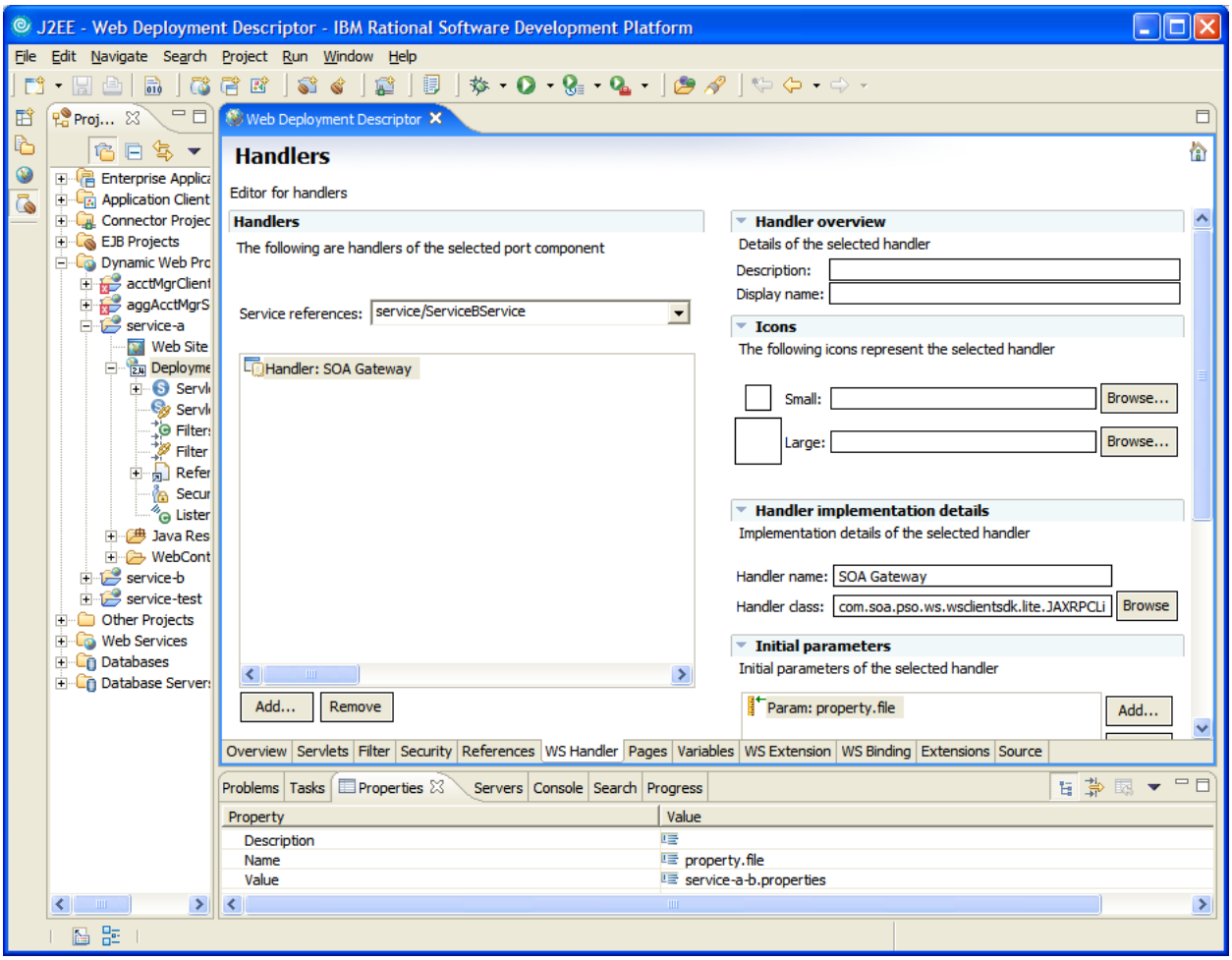
Lines 03 – 05 hold the LocalAxisPivotHandler declaration and configuration parameters. These parameters will not include any service context. Lines 07 – 15 hold a service declaration. Lines 10 – 12 hold the AxisServiceHandler declaration and configuration parameters. Line 11 specifies that the service is identified in Policy Manager with a binding identifier of “myservice."

Using the Gateway Agent

Once you have created your WebSphere web service consumer application, it is very simple to enable it to take advantage of the features of the Client Gateway Agent. Just follow these steps:

Copy *`[agent-zip-dir]/soa-framework-52-client-was60.jar`* to WEB-INF/lib in your WAR file or to your EAR file depending on where you are invoking web services in your application.

Modify the Web Service Handler configuration of your web application. Although you could do this by directly editing the XML files, we recommend that you do this using the Deployment Descriptor Editor included in Rational Application Developer. Here is a screen shot of that configuration screen for a Web Application. Similar functions are available for the EJB deployment descriptors.



The critical aspect of this configuration is the *Handler class* field in the **Handler implementation details** area. This must be set to:

```
com.soa.pso.ws.wsclientsdk.lite.JAXRPCLiteGatewayHandler
```

This message handler enables the Gateway Agent. The processing performed in the Gateway Agent is specified either directly or indirectly using the **Initial parameters** section of the handler configuration. In the example above, this configuration is pushed out to an external property file that is loaded via the application's class path. You can see this in the *Properties* view where the `property.file` property is set to the name of the Gateway Agent configuration property file that is in the class path.

More information on the initialization properties that can be specified to control the actions of the Gateway Agent can be found in the next section.

When a web service client application has been configured to enable the Client Gateway, it is still possible to run that application in environments that do not have the full Gateway Agent installed. This might be desirable, for instance, in the initial development environment where the client application is still being debugged in Rational Application Developer. The RAD integrated test environment is not really intended to have other complex applications installed.

If the Gateway Agent "shared" JAR (installation step #1) and the EAR file (step #4) are not detected, then all Gateway Agent processing is bypassed and the native SOAP container processing is used. The

Gateway Agent processing can also be forcibly disabled by specifying the following Java system property:

```
-Dsoa.gateway.enable=false
```

When the Gateway Agent is disabled, the WSDL file that was used to create the client stubs is used to determine the URL of the web service that is called.

Configuring the Gateway Agent

The SOA Web Service Client Gateway brings the following benefits to Web Service consumer applications without requiring any changes to the application's code:

1. The Web Service end-point and transport protocol are dynamically discovered and bound based on a service binding key. When the Web Service is being managed by the Service Manager, communication from the consumer is automatically diverted to the desired Management Point(s).
2. Communication between Service Consumers and Management Points can be clustered for availability and scalability without the need of any special network devices.
3. Management policy for a Web Service can be dynamically applied for the Service Consumer as needs change without making any changes to the consuming application or its environment. For instance, if the management policy indicates that a Web Service response is to have critical information encrypted, this dynamic processing would automatically decrypt the information before returning it to the Workshop application.

The capabilities of the Gateway Agent are not limited to WebSphere 6.0. Other distributions can be delivered in any environment that supports JAX/RPC 1.0 or higher. Examples of other platforms in which this component could be used are BEA WebLogic, Oracle OC4J, JBoss, and all J2EE 1.4 compliant platforms. All of these platforms allow message handlers to be configured through deployment descriptors eliminating any need to change the consumer application. (Some platforms such as WebLogic 8.1 can also be used but a small amount of application programming must be done to enable the message handler since these platforms do not support configuring client0side message handlers using deployment descriptors.)

The Client Gateway Agent is configured using property values that are evaluated when the handler is initialized. These properties can come from a number of sources including Java system properties and the J2EE Application Environment. For complete information on using these configuration properties, see the next section on configuring the message handler.

The Client Gateway Agent also contains an interface that allows applications to augment and substitute for the behavior contained in the message handler. Methods of this interface correspond to each of the main processing steps in the life cycle of the handler. To take advantage of this feature, an application would create a Java class that implements the necessary parts of the extension interface and specify the name of this new Java class in one of the handler configuration properties.

Gateway Agent Configuration Properties

Several sources of information can be used to configure the Gateway Agent message handler. These are:

1. Properties that were set by the application during execution before the Web Service is invoked. (These are called Thread Context properties.) The Gateway Agent provides an API that allows

the application to set any of the initialization properties from anywhere in the thread of execution leading up to the Web Service invocation. The following example shows how to use this feature.

```
import com.soa.pso.ws.gateway.GatewayAgentProperties;
. . .
GatewayAgentProperties.setProperty("user.credential.type", "static");
GatewayAgentProperties.setProperty("user.username", "enduser1");
GatewayAgentProperties.setProperty("user.password",
"{ENCR}oGN3wDkNmGPoYKE+pRevRg==");
```

These properties are all cleared after each Web Service request.

2 Initialization properties included in the JAX/RPC handler definition. (See the previous section on enabling the Gateway Agent for details on specifying initialization properties.) These properties can be extended by one or more property files as discussed in a later section.

3 J2EE Application Container Environment entries. These values are specified in <*env-entry*> elements in either the *web.xml* or *ejb-jar.xml* deployment descriptors depending on what container is issuing the Web Service call.

4 Java system properties. These are specified on the Java command line using *–Dproperty=value* arguments.

During the initialization of the Client Gateway Agent, configuration properties are examined from these sources in the order shown: first, Thread Context properties, then the JAX/RPC initialization properties, then the J2EE environment, and, finally, Java system properties.

The actual Gateway Agent configuration property names depend on the source of the property according to the following rules:

Property Source	Property Name Structure
Thread Context property	*property-name*
JAX/RPC initialization property	*property-name*
J2EE container environment	soa.gateway/*property-name*
Java system property	soa.gateway.*property-name*

Some Gateway Agent properties end up pointing to other properties that contain the actual configuration value. An example is the *service.binding.key.property* whose value is the name of another property that contains the actual service binding key. In this case, the *service.binding.key.property* property would be subject to the above prefixing rules when it is looked up but the actual named property containing the real value would not. The value specified for the referencing property is taken literally when looking up the referenced property containing the "real" configuration value.

Property Prefix and Suffix Processing

The *property-name* in the table above can be augmented with a constant prefix and/or suffix. This might be useful in a situation where a file contains properties for a number of different environments and the specific environment is selected by specifying a suffix to be applied to all other property names. The specified prefix and suffix values are case sensitive (as are all property names) and will be separated from the *property*-name by a period. The order of evaluation for prefixing and suffixing is:

prefix.property-name**.suffix**
prefix.property-name
property-name**.suffix**
property-name

These additions to the property names occur before the source of the property and, thus, the *soa.gateway* qualifier is added after any prefix processing is done. For instance, with `property.prefix=PROD`, a fully-qualified system property on the Java command line might be:

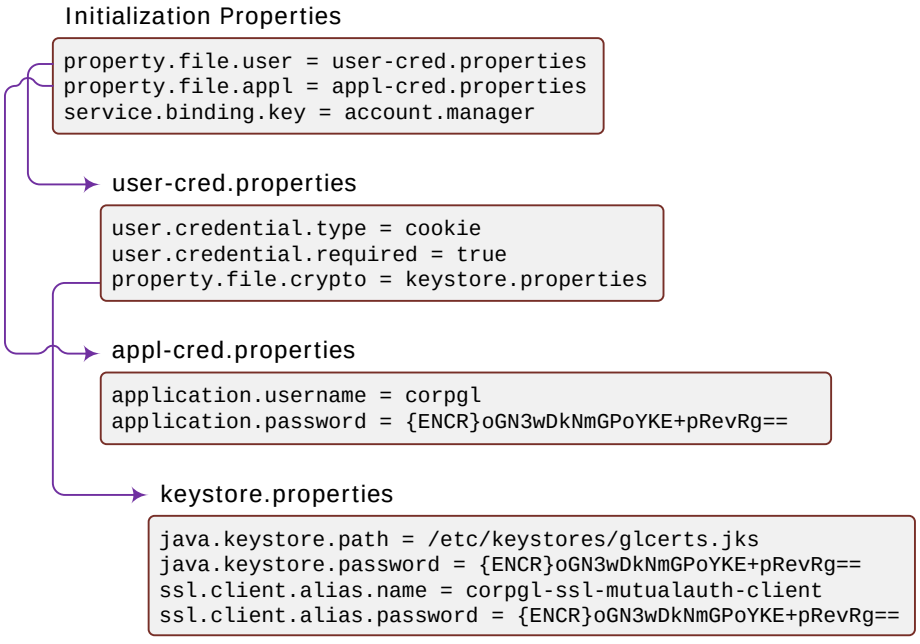
```
-Dsoa.gateway.PROD.application.username=genledger
```

As with the source name space prefixing (*soa.gateway*), prefixing and suffixing is disabled for Gateway Agent properties that reference other properties. In this case, the *referencing* property name would be subject to the above prefix/suffix rules when it is looked up but the actual named property containing the real value would not. The value specified for the referencing property is taken literally when looking up the referenced property containing the "real" configuration value.

One final rule regarding prefixing and suffixing is that the properties defining prefixes and/or suffixes are never, themselves, subjected to these prefixing and suffixing rules. They are, however, still subject to the property source name space prefix (*soa.gateway*) rules.

Using Property Files

The initialization parameters used to define the Client Gateway message handler can be extended using one or more external property files. These are conventional Java property files and must all be accessible on the active Java class path. Here is a diagram that shows an example of the logical structure of some initialization parameters and a couple of property files



The properties in this example that start with *property.file* are used to include the contents of external property files in the set of configuration properties used by the Gateway Agent. In this example, the Message Handler initialization properties referenced two property files which got loaded from the Java class path in the order specified. One of those files references a third file would not get loaded until after the first two.

All of the properties from these files, along with the original initialization parameters are combined together to configure the Gateway Agent processing. If there are duplicate properties in these sources, then the ***first*** property specified is the one that will be used and subsequent entries in other files will be ignored. However, if a property is duplicated within a property file, then the ***last*** one is the one that will be used because of standard Java property processing.

Gateway Agent Configuration Property Details

The details of each configuration property for the SOA Web Service Client Gateway Agent are discussed in the following table

<table>
<tr><th>Property</th><th>Description</th><th>Default Value</th></tr>
<tr><td>property.file[.xxx]</td><td>Specifies the file id of a standard Java property file that contains more Gateway Agent configuration properties from the list below. This property file must be found in the application's class path and the files cannot be chained or cascaded.</td><td>None.</td></tr>
<tr><td>property.prefix</td><td>Specifies a constant prefix value (with any punctuation) to be applied to other property names.</td><td rowspan="2">None</td></tr>
<tr><td>property.prefix.property</td><td>Exactly specifies another property (usually a Java system property) that indirectly specifies the constant prefix value to be used.</td></tr>
<tr><td>property.suffix</td><td>Specifies a constant suffix value (with any punctuation) to be applied to other property names.</td><td rowspan="2">None</td></tr>
<tr><td>property.suffix.property</td><td>Exactly specifies another property (usually a Java system property) that indirectly specifies the constant suffix value to be used.</td></tr>
<tr><td>bootstrap.url</td><td>A comma delimited list of URLs to the Registry Manager used to initially establish the connection from the Client Gateway to the Service Manager subsystems. For the Gateway Agent Lite solution, this property is contained in the soa-environment.properties file and does not need to be specified in the client application.
As of Workbench v5.2 Update 1.0, this is a list of URLs to the Metadata Exchange (WS-MEX) Service; e.g. http://host:9904/wsmex</td><td rowspan="2">None. One of these two forms must be specified</td></tr>
<tr><td>bootstrap.url.count
bootstrap.url.n</td><td>Alternative way of specifying any number of initial URLs for the Registry Manager or, ***as of 5.2 Update 1.0***, the WS-MEX service.</td></tr>
<tr><td>application.domain</td><td>The optional Identity System domain name for the application user.</td><td>None</td></tr>
<tr><td>application.username
application.password</td><td>The userid and password credentials for the Service Manager user that the Gateway will use to connect to the Service Manager. See a later section about encrypting passwords in properties
This credential is also passed in the SOAP message when the Security Component is configured to authenticate a "Consumer Identity"</td><td>None, must be specified</td></tr>
<tr><td>user.credential.type</td><td>Choose one of the following values for the type of user credential to use in the Web Service call:
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>none</td><td>do not use any client credential</td></tr>
<tr><td>static</td><td>use specified userid and</td></tr>
</table>
</td><td>Auto</td></tr>
</table>

<table>
<tr><th>Property</th><th>Description</th><th>Default Value</th></tr>
<tr><td></td><td>
<table>
<tr><td></td><td>password</td></tr>
<tr><td>x509</td><td>use an X.509 public-key certificate that is specified by the user.cert.* and java.keystore.* properties discussed below</td></tr>
<tr><td>cookie</td><td>use SSO session token that was captured by one of the provided servlet filters</td></tr>
<tr><td>siteminder-asa</td><td>use information in the JAAS Principal built by the SiteMinder Identity Asserter</td></tr>
<tr><td>context</td><td>uses the credential from an incoming SOAP request that has been captured by either the collectSoaCredential Management Policy component or message handler. See a later section on this specialized function for service chaining</td></tr>
<tr><td>auto</td><td>find user credentials in the container using a specified algorithm</td></tr>
</table>
A later section discusses the use of credentials in more detail.
</td><td></td></tr>
<tr><td>user.credential.required</td><td>Specifies whether or not credential is required. This property only applies for credential types of “cookie”, “siteminder-asa”, “context”, or “auto”. When set to true, the caller will be returned a SOAP Fault if credentials cannot be found. If set to false, the request is processed without user credentials.</td><td>True</td></tr>
<tr><td>user.domain</td><td>The optional Identity System domain name for the end user. This can be specified for any type of credential and controls the domain that is used for authenticating the end user</td><td>None</td></tr>
<tr><td>user.username
user.password</td><td>The userid and password credentials for the “user” that is calling the Web Service. These properties are only used if the credential type (above) is set to “static”. See a later section about encrypting passwords in properties</td><td>None, only required for “static” credential type</td></tr>
<tr><td>service.binding.key</td><td>Specifies the value that will be used to locate the Web Service in the UDDI Registry. Creating and using service binding keys is discussed in more detail in a later section</td><td>None, see below for usage notes</td></tr>
<tr><td>service.binding.key.property</td><td>Exactly specifies another property to be examined to determine the service binding key. This allows for the indirect specification of the binding key of a service. If specified, this property name is used as-is to locate the actual binding key.</td><td>None, see below for usage notes</td></tr>
<tr><td>service.binding.url</td><td>Specifies the URL to which Web Service requests will be sent. The UDDI Registry is not accessed to determine service properties and the SOA Client Gateway’s dynamic configuration processing is not invoked.</td><td>None, see below for usage notes</td></tr>
</table>

<table>
<tr><th>Property</th><th>Description</th><th>Default Value</th></tr>
<tr><td>service.binding.url.property</td><td>Exactly specifies another property to be examined to determine the service binding URL. This allows for the indirect specification of the service URL. If specified, this property name is used as-is to locate the actual service URL.</td><td>None, see below for usage notes</td></tr>
<tr><td>transport.type</td><td>Choose one of the following values for the type of transport to use in the Web Service call:<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>any</td><td>no transport preference given</td></tr><tr><td>http</td><td>only allow the use of HTTP</td></tr><tr><td>https</td><td>only allow the use of HTTPS</td></tr><tr><td>jms</td><td>only use JMS transport</td></tr></table></td><td>Any</td></tr>
<tr><td>java.keystore.path</td><td>Specifies the full file system path and name of the Java Key Store file that will be used by the Gateway Agent.</td><td>None. Required for SSL mutual-auth is needed</td></tr>
<tr><td>java.keystore.password</td><td>The password for accessing the Java Key Store specified above. See a later section about encrypting passwords in properties</td><td>None. Required to access the JKS file</td></tr>
<tr><td>ssl.client.alias.name</td><td>The “alias” name within the Java Key Store (java.keystore.path property) of the key pair that is to be used for the client-side of SSL mutual-auth connection to the web service (or MP)</td><td>None. Required if SSL mutual-auth is needed</td></tr>
<tr><td>ssl.client.alias.password</td><td>The password of the “alias” specified in the ssl.client.alias.name property above. See a later section about encrypting passwords in properties</td><td>None. Required if SSL mutual-auth is needed</td></tr>
<tr><td>user.cert.alias.name
user.cert.alias.password</td><td>The “alias” name and password within the Java Key Store (java.keystore.path property) of the X.509 certificate to be used as the end-user credential. This property only used if the credential.type property is set to “x509”</td><td rowspan="2">None. One of these is required if credential.type is set to x509</td></tr>
<tr><td>user.cert.path</td><td>The full path and file name of a .cer X.509 certificate file that is to be used as the end-user credential. This property only used if the credential.type property is set to “x509”</td></tr>
<tr><td>extension.class</td><td>Specifies the fully-qualified name of a Java class that implements the IGatewayHandlerExtension interface. This class is used to extend the processing of the Client Gateway Agent in many diverse ways. See the Appendix on extending the Gateway Agent for more details.</td><td>No extension class used.</td></tr>
</table>

Once you have examined the collection of Gateway Agent properties in the tables above, you will see that there is a hierarchy of properties relative to a given application, the web services it invokes, and the environment the application in which the application is executing. For instance, consider the following points:

- Some of the configuration properties apply to the environment in which the consumer application has been deployed within. Specifically, the *`bootstrap.url`* property is always specific to an environment. It is for this reason that the "Gateway Agent Lite" implementation suggests including that property in the separate *`soa-envirnoment.properties`* file that is loaded by the Gateway Agent EAR file and does not need to be included in the properties presented by the actual consumer application.

Many other properties are related to the consumer application rather than the environment or the service being invoked. By far, the majority of the properties fall into this category. Here are some examples:

application.username and application.password
java,keystore.path and java.keystore.password
ssl.client.alias.name and ssl.client.password

We recommend that these properties be externalized from the deployed application since, although they apply to the application, they need to change based on the deployed environment. Therefore, these properties should be contained in a property file that is referenced from the *`property.file`* initialization property.

- Several of the properties are unique to the actual service being invoked. Most applications will need to invoke more than one service so these properties need to be specified at the service-client level. The most notable of these properties is the *`service.binding.key`* property. Since these properties need to be bound to the consumer invocation of an individual service, they should be included in either of these two options:
  - Include in the **Initial parameters** when the handler was initially enabled. This is specific to the service invoked and, it external process allows, the binding key and other service-specific properties should be established at this point.
  - Specify a unique *`property.file`* property for each invoked service in the **Initial parameters** when the Gateway Agent handler is defined. Although this requires the duplication of Gateway Agent configuration properties that belong at the application level, it probably offers the most flexibility.

Gateway Agent Processing of User Credentials

The Service Manager Client Gateway agent has several different options for collecting and processing user credentials to be used with the Web Service call. These options are provided to address a number of customer use cases. Options exist to specify user credentials for the typical application-to-application (A2A) use case. Several other options are provided to address single sign-on (SSO) use cases in a seamless integration with SiteMinder in the J2EE application container.

Static user credentials: When application needs do not require single sign-on, then the application can provide user credentials that consist of a fixed user name and password. Specify the following Gateway Agent configuration properties to enable this option:

```
user.credential.type = static
user.username = userid
user.password = password
```

X.509 Certificate credentials: The Client Gateway can use an X.509 Certificate as the web service end-user credential. This feature is enabled by specifying the following Gateway Agent configuration property

```
user.credential.type = x509
```

The Client Gateway can load the X.509 certificate from a certificate file (.CER) or from a Java Key Store. To load the certificate from a loose file, specify the following configuration property:

```
user.cert.path = full path and name of certificate file
```

To have the Client Gateway load the X.509 certificate from an entry in a Java Key Store file, you must include the following configuration properties:

```
java.keystore.path = full path and name of JKS file
java.keystore.password = password
user.cert.alias.name = name of JKS entry to use
user.cert.alias.password = password
```

SSO session token cookie: The Gateway Agent has the ability to use the value of the SSO session token that is generated by third-party authentication agents such as SiteMinder and OAM CoreId as the source of authenticated user identity when making the Web Service call. This allows the user's identity to be authenticated by the third-party Agent protecting the web application that invokes the Web Service. This identity is then passed on to the Web Service. There are a couple of steps to take to enable the Gateway Agent to process SSO token cookies. The first is to specify this type of credential processing with the following Gateway Agent configuration property

```
user.credential.type = cookie
```

The other requirement for this type of credential processing is that the SOA Identity Manager (part of the Client Gateway Agent package from SOA Software Professional Services) must be enabled when the "cookie" credential type is selected. This is implemented as a J2EE Servlet Filter and is responsible for collecting the HTTP cookie from an incoming servlet request and making it available to any other classes running on the same thread in the container. The following XML elements must be added to the `web.xml` deployment descriptor to activate this function:

For SiteMinder (SMSESSION cookie):

```
<filter>
   <filter-name>SMCookieFilter</filter-name>
   <filter-class>com.soa.pso.fw.identity.web.netegrity.SMHttpFilter</filter-class>
</filter>
<filter-mapping>
   <filter-name>SMCookieFilter</filter-name>
   <url-pattern>/*</url-pattern>
</filter-mapping>
```

For OAM CoreId (ObSSOCookie cookie):

```
<filter>
   <filter-name>OBCookieFilter</filter-name>
   <filter-class>com.soa.pso.fw.identity.web.oam.OBHttpFilter</filter-class>
</filter>
<filter-mapping>
   <filter-name>OBCookieFilter</filter-name>
```

```
   <url-pattern>/*</url-pattern>
</filter-mapping>
```

For Sun Access Manager [Open SSO[(iPlanetDirectoryPro cookie):

```
<filter>
   <filter-name>AMCookieFilter</filter-name>
   <filter-class>com.soa.pso.fw.identity.web.opensso.AMHttpFilter</filter-class>
</filter>
<filter-mapping>
   <filter-name>AMCookieFilter</filter-name>
   <url-pattern>/*</url-pattern>
</filter-mapping>
```

For other token providers or cookie names:

```
<filter>
   <filter-name>SSOCookieFilter</filter-name>
   <filter-class>com.soa.pso.fw.identity.web.HttpFilter</filter-class>
   <init-param>
      <param-name>cookie.name</param-name>
      <param-value>your-cookie-name</param-value>
   </init-param>
</filter>
<filter-mapping>
   <filter-name>SSOCookieFilter</filter-name>
   <url-pattern>/*</url-pattern>
</filter-mapping>
```

The Identity Manager depends on standard Java *ThreadLocal* class for proper operation. This means that the technique will only work when the Gateway Agent JAX/RPC message handler is executed on the same Java thread that received the original HTTP servlet post. There have not been any incidents of this constraint causing any problems but Java threading is a complex subject and issues may arise in the future. There, however, is a known issue with the BEA JRocket Java Virtual Machine. Certain versions do not appear to process threads (and the *InheritableThreadLocal* class in particular) in the same manner as Sun Java. Because of this, the Identity Manager cannot be used with BEA JRocket.

SiteMinder JAAS Principal: The Gateway Agent can extract the user identity and session token from the JAAS Principal that was created by a component of the SiteMinder Application Server Agent for WebLogic 8.1. This type of credential processing is, again, provided for Single Sign-On (SSO) use cases. In some cases, it may be preferable to the previously-discussed SiteMinder session cookie processing. This is due to the Java thread management issues that were mentioned. On the other hand, this option may not be available when the Gateway Agent is to be used on a platform other than WebLogic version 8.1. To enable SiteMinder JAAS Principal credential processing in the Gateway Agent, specify the following configuration property:

user.credential.type = siteminder-asa

The *SiteMinder Identity Asserter* from the SiteMinder ASA for WebLogic 8.1 (not to be confused with an earlier Netegrity offering called, simply, the WebLogic Identity Asserter) must be configured and enabled for the application. Please refer to the Netegrity documentation for information on configuring the *SiteMinder Identity Asserter for WebLogic 8.1*. In order to activate the Identity Asserter processing for a given application, the following lines must be placed in the *web.xml* deployment descriptor.

```
<login-config>
   <auth-method>CLIENT-CERT</auth-method>
```

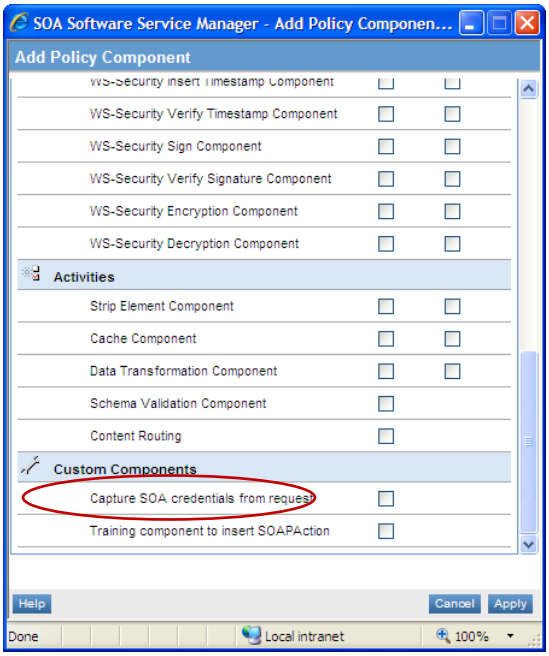
```
</login-config>
```

Credential From SOAP Request: The Service Manager Client Gateway Agent has a feature that can be used when one web service calls another web service. The Gateway Agent has components that can be used to collect and save the credential from the SOAP request sent to the first web service. Then, when the first web service calls the second web service, the Gateway Agent can be configured to look for and use this collected credential using the following Gateway Agent configuration property

```
user.credential.type = context
```

The other requirement when using this credential propagation feature is that you must configure the mechanism that captures the credential from the first (calling) web service's incoming SOAP request. The Gateway Agent offers two different ways of doing this depending on the environment in which the web service has been deployed. These two methods are:

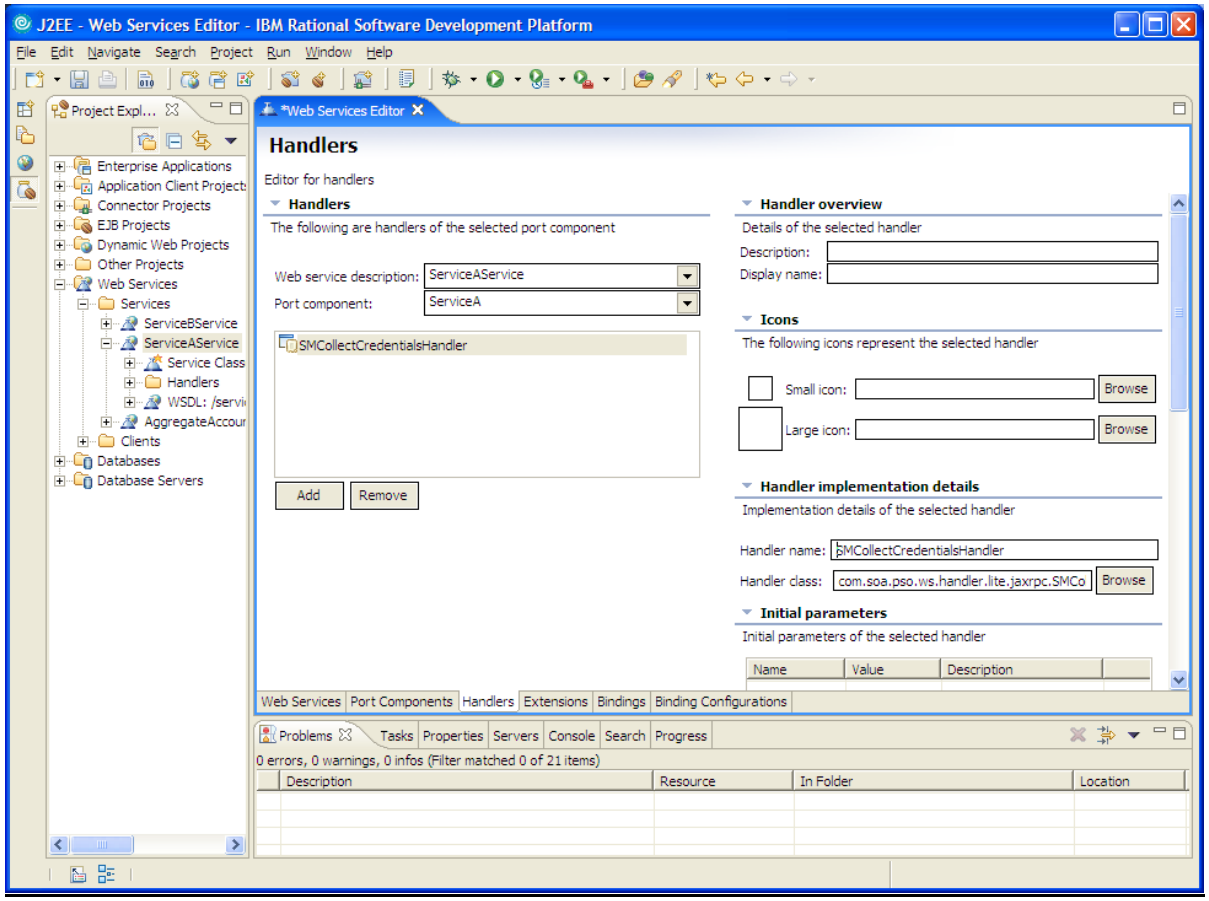
1. If the web service is being hosted by an Embedded Management Point, then the Gateway Agent provides a custom management policy component that can be included in the request policy of the first web service. When this custom component has been installed (see page), it will show up in the *Custom Components* section of the Add Policy Component screen as shown here:



   This custom component would, typically, be configured in a management policy that would include the following components in the Request policy:

   - The *Security Component* that would Authenticate using, for instance, a WS-Security SAML token.
   - The custom *Capture SOA credentials from request* component that would collect the SAML Assertion from the WS-Security header and save it for later use by the Gateway Agent.
   - The *Strip Component* that will, then remove the WS-Security header from the message because it contains the *mustUnderstand="1"* attribute which will likely cause WebSphere to reject the message.

2 If the web service is not managed by the Embedded Management Point or the custom component has not been installed, then the Gateway Agent provides a JAX/RPC message handler that can be attached to the web service to capture the credential from the incoming SOAP request. This message handler needs to be configured using the Rational Application Developer's *Web Service Editor*. The following screen shows this handler being configured.



The fully-qualified name of the class that must be specified in the **Handler implementation details** section for this message handler is:

`com.soa.pso.ws.handler.lite.jaxrpc.SMCollectCredentialsHandler`

Note that when you are using the JAX/RPC message handler to collect credentials from the incoming message that the management policy for that web service will, likely, need to be configured to remove the *mustUnderstand* attribute from any WS-Security headers. You cannot remove these headers because the message handler will be extracting the credentials from those headers. The *Strip Component* cannot be used to remove attributes. Instead, you must use the *Transform Component* to remove the attribute using XSL. Here is an example transform that will accomplish this:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

    <xsl:template match="*|@*|node()|text()">
        <xsl:copy>
```

```
            <xsl:apply-templates select="*|@*|node()|text()"/>
        </xsl:copy>
    </xsl:template>
    <xsl:template match="@soap:mustUnderstand">
    </xsl:template>
</xsl:stylesheet>
```

Beginning with Service Manager v5.1, there are a number of common situations where the SOAP request will need to contain two or more credentials. The credential collection process has a couple of configuration properties that can be used to control which credential in the message is to be saved for propagation into any outgoing SOAP requests. These properties are specified as either:

1 Policy component configuration properties when the custom policy component is configured for the service

2 Initialization parameters when the JAX/RPC message handler is configured.

The following properties can be used to specify which credential is to be retrieved from the message. If no properties are specified, the first credential found in the message will be used.

<table>
<tr><th>Property</th><th>Description</th></tr>
<tr><td>credential.type</td><td>Set to one of the following values:
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td colspan="2">These values refer to SOA Service Manager proprietary credential headers</td></tr>
<tr><td>soa</td><td>Any SOA-proprietary credential header</td></tr>
<tr><td>soa.xpath</td><td>An SOA proprietary credential header at a specific XPath. You must also specify the credential.xpath property</td></tr>
<tr><td>soa.username</td><td>A user name and password header</td></tr>
<tr><td>soa.x509</td><td>An X.509 credential header</td></tr>
<tr><td>soa.saml</td><td>A SAML assertion header</td></tr>
<tr><td>soa.cookie</td><td>An SSO token header</td></tr>
<tr><td>soa.siteminder</td><td>A SiteMinder session token header</td></tr>
<tr><td colspan="2">These values refer to WS-Security credential headers</td></tr>
<tr><td>wss</td><td>Any WS-Security header</td></tr>
<tr><td>wss.username</td><td>A WS-Security Username token header</td></tr>
<tr><td>wss.x509</td><td>A WS-Security BinarySecurity token header</td></tr>
<tr><td>wss.saml</td><td>A WS-Security SAML token header</td></tr>
</table></td></tr>
<tr><td>credential.xpath</td><td>Specifys the optional XPath that contains the Service Manager proprietary credential</td></tr>
<tr><td>credential.actor</td><td>Specifies the actor to be used to select the WS-Security credential header.</td></tr>
</table>

Automatic User Credential Collection: The Service Manager Client Gateway Agent has an automatic mode of credential processing that will attempt to locate user credentials or authentication tokens from

a number of sources within the J2EE application container. To invoke automatic user credential processing, specify the following Gateway Agent configuration property

```
user.credential.type = auto
```

When the credential type of "auto" is specified, the Client Gateway Agent looks in the J2EE application container for user credentials in the following sequence:

1. An explicitly specified userid and password from the JAX/RPC message context, associated Call object, or the Stub object. All of these tests apply to explicitly specifying a userid and password using JAX/RPC dynamic invocation of services.
2. The userid and password that was specified when the Service object was created. This test applies to applications specifying explicit credentials in the JAX/RPC static client model.
3. A SSO session token that has been stored into the Stub or Call object by the application. This is a JAX/RPC extension that is provided as part of the Identity Manager function. (This is also a function that would be used by applications taking advantage of the JAX/RPC dynamic invocation features.)
4. SiteMinder identity information that has been placed into the JAAS Principal by the SiteMinder Identity Asserter component of the SiteMinder Application Server Agent. This information reflects the originally logged in browser user when the ASA's Identity Asserter is properly configured for the application.
5. The Identity Manager is interrogated to obtain the contents of the SSO session cookie that was posted to the servlet that is making the Web Service call. See the section above related to *credential.type=cookie* for details about configuring this option.
6. A credential that was captured from the incoming SOAP request. This would have been saved in the thread context by either the *Capture CredentialsComponent* custom management policy component or the *SMCollectCredentialsHandler* JAX/RPC message handler.
7. If all of these sources fail, then there are no user credentials available in the environment. This condition will cause the Web Service call to immediately fail if the *user.credential.required* property has been set to true.

Using Service Binding Keys

One of the advantages of using the Service Manager Client Gateway Agent is that the address of the Web Service endpoint does not need to be hard-coded in the consumer application. Using the Service Manager Registry Manager, the Client Gateway can provide dynamic binding by which the endpoint address is found at the runtime instead of at the build time or design time.

There are two forms of dynamic binding provided by Client Gateway:

- **Identifier Binding:** As part of identifier binding, the client will be providing an identifier to bind to the endpoint. The Web Service Gateway searches for this identifier in the reserved Web Service Gateway look-up category scheme to find the service key of the endpoint to bind to. From this service key, the Web Service Gateway finds the address of the endpoint by using Service Manager Registry Manager. To set up for Identifier binding, in the Service Manager Management Console add a *Binding Identifier* to the web service in the Service Details section of the Service Manager Management Console. When the identifier is added, Service Manager creates a category in the reserved category scheme in UDDI.

An advantage of identifier binding is that the identifier is a key that can remain consistent from development, test, through the quality assurance environment, and into production. Conformance with a "build once, deploy anywhere" deployment standard can be realized when this type of value is used.

- **Service Key Binding:** As part of service key binding, Client Gateway looks up in Service Manager Registry Manager for the service with the received service key to find the address of the endpoint. This is the actual "UDDI service key" of the virtual (standalone MPs) or physical (for embedded MPs) service that is being managed by the Service Manager. Typical values are unique keys like:

  uddi:93dc76cf-2563-11da-bc6b-ea3fc0729212

One of these two forms can be specified in the service.binding.key configuration property for the Gateway Agent. It is also possible to have the service binding key indirectly specified by a different configuration property by using the service.binding.key.property configuration property.

The Client Gateway Agent has an additional way of processing service bindings that automatically determines a binding identifier based on the content of the SOAP message being sent. To invoke this automatic processing, simply code the binding key configuration property as:

service.binding.key = default

When this is specified, the Client Gateway Agent uses the XML Namespace of the Web Service operation being called as the service binding key. If the operation has no namespace, then the WSDL bind address of the service is used as the service binding key.

If neither the service.binding.key nor the service.binding.url properties are specified, then the SOA Service Manager Client Gateway is not invoked and the message handler performs no processing of either the request or response message. This configuration could be considered to make the earlier stages of developing web service consumers a little easier because the Service Manager environment would not be needed.

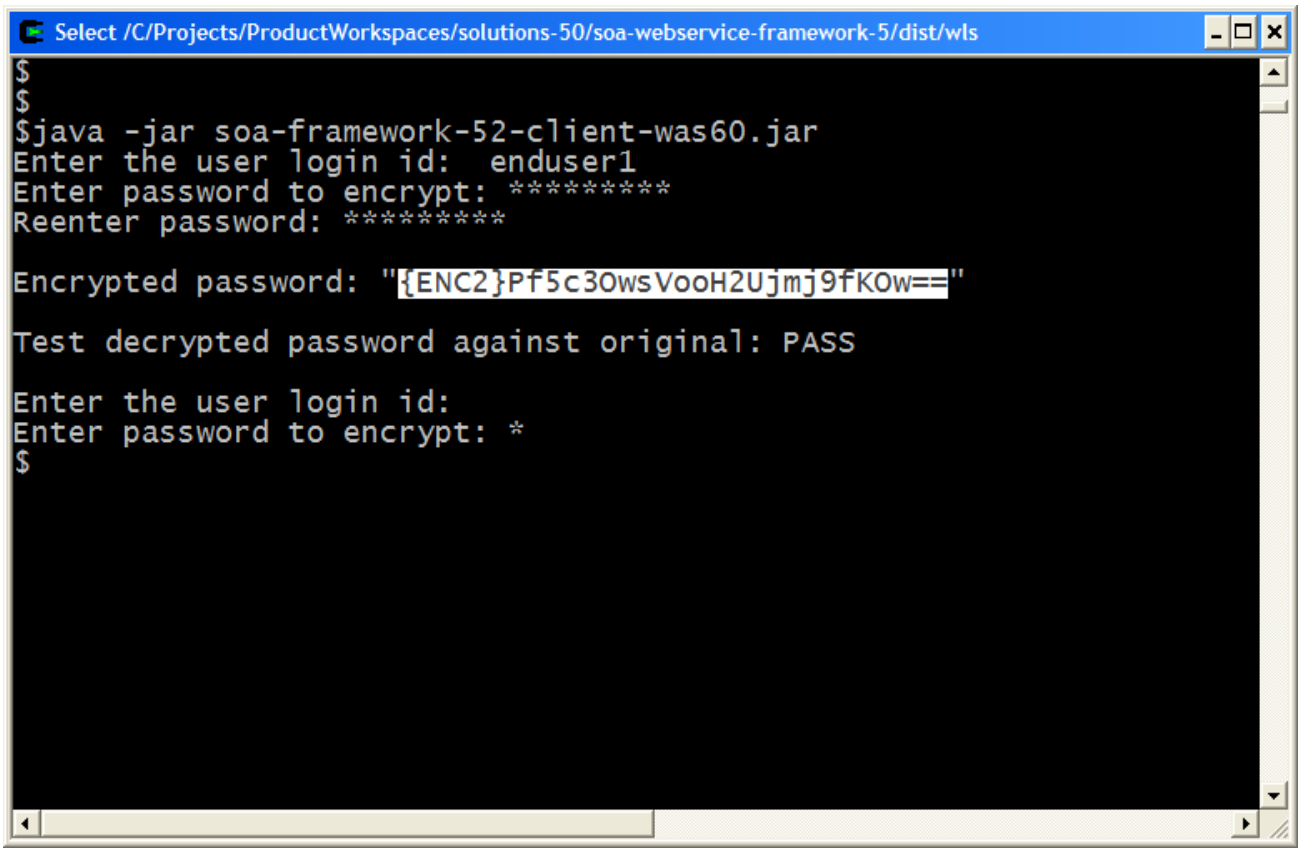
Encrypting Gateway Agent Configuration Passwords

The SOA Client Gateway Agent comes with a feature that allows you to encrypt the passwords that are placed in the Gateway Agent configuration. This consists of a command-line tool that will accept a password and print out the encrypted value. You then cut and paste this value into the Gateway Agent configuration. At run-time, the Gateway Agent recognizes these encrypted passwords and preserves the encrypted values until just before they are actually used. The Gateway Agent then decrypts the value, uses the password, and then destroys the plain-text value.

The command-line tool to generate encrypted passwords is a Java program that is included in the Gateway Agent's client JAR file: `[agent-zip-dir]/soa-framework-52-client-was60.jar`. To use this tool, navigate to the `[agent-zip-dir]` directory and issue the following command:

```
java –jar soa-framework-52-client-was60.jar
```

Here is an example:



The Encrypted password value inside the quotes should be cut and pasted into the Gateway Agent configuration. Here is what a completed configuration property might look like:

```
application.password={ENC2}Pf5c3OwsVooH2Ujmj9fKOw==
```

Using the Gateway Agent with Apache Axis 1.x

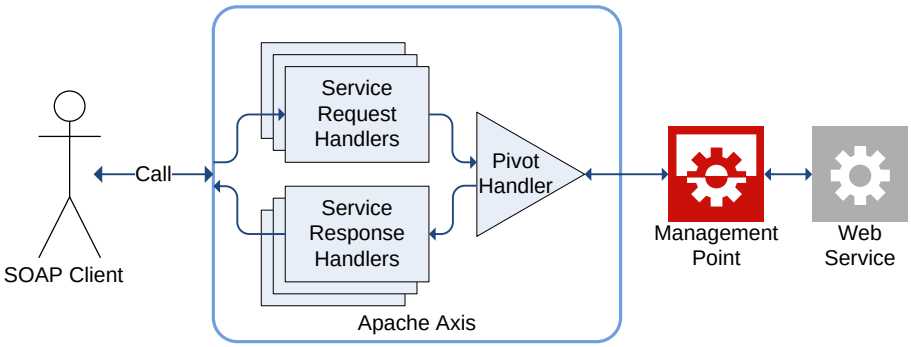
In addition to WebSphere's native JAX/RPC support, you can use Apache Axis 1.x package to call web services. Some people might prefer to use this SOAP "container" because of improved portability or to leverage existing processes or knowledge. All of the features of the SOA Web Service Client Gateway Agent are still available if you choose to implement your web service consumer using Apache Axis 1.x. (Axis 2.x is very different and it not yet supported.)

The primary difference between using the Gateway Agent with Axis and WebSphere's JAX/RPC support is the manner of declaring the Message Handlers and the Java class of the handlers. In Axis, the client-side handlers are declared in the Web Service Deployment Descriptor (`client-config.wsdd`). Here is an example of this file showing how to define the message handlers needed to use the Gateway Agent with Axis 1.x.

```
<?xml version="1.0" encoding="UTF-8"?>
<deployment xmlns="http://xml.apache.org/axis/wsdd/"
```

```
            xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
   <handler name="SOADelegatePivot"
            type="java:com.soa.pso.ws.wsclientsdk.lite.AxisLiteGatewayPivot">
      <parameter name="delegate-handler"
                 value="org.apache.axis.transport.http.HTTPSender"/>
      <parameter name="property.file" value="acct-mgr-sample.properties" />
   </handler>
   <service name="AccountManager">
      <requestFlow>
         <handler
type="java:com.soa.pso.ws.handler.wsclientsdk.axis.AxisClientGatewayHandler">
            <parameter name="service.binding.key" value="account.manager.service" />
         </handler>
      </requestFlow>
   </service>
   <transport name="http" pivot="SOADelegatePivot"/>
</deployment>
```

Notice in this example that there are two client-side message handlers specified. This is in order to deal with the fact that Axis has only one definition for the "pivot" message handler and applications that call more than one web service need to be able to configure the Gateway Agent with different properties for each service. The structure of an Axis client application and its message handlers is shown here.



This diagram shows that there can be one or more message handlers defined to process SOAP requests and responses for each service the application calls. To configure the SOA Client Gateway Agent, you should define the `<request-flow>` for each service the application calls. Let's look at that segment of the `client-config.wsdd` again.

```
<service name="AccountManager">
   <requestFlow>
      <handler
type="java:com.soa.pso.ws.handler.wsclientsdk.axis.AxisClientGatewayHandler">
         <parameter name="service.binding.key" value="account.manager.service" />
      </handler>
   </requestFlow>
</service>
```

Axis provides several ways to define the `<service>` element in the `client-config.wsdd` file but the simplest is to set the *name* attribute to match the `<wsdl:port>` name in the WSDL file of the service.

The request flow handler should be set to the *AxisClientGatewayHandler* for the SOA Client Gateway Agent. This handler just serves as a place to specify Gateway Agent configuration properties that are

unique to that particular service such as the `service.binding.key` property as shown in this example. This handler does not perform any processing of the actual SOAP messages.

All of the Client Gateway Agent processing is performed in the Axis "pivot" handler. Let's look again at the definition of this handler in `client-config.wsdd`:

```
<handler name="SOADelegatePivot"
         type="java:com.soa.pso.ws.wsclientsdk.lite.AxisLiteGatewayPivot">
   <parameter name="delegate-handler"
              value="org.apache.axis.transport.http.HTTPSender"/>
   <parameter name="property.file" value="acct-mgr-sample.properties" />
</handler>
    ...
<service name="...">
    ...
</service>
    ...
<transport name="http" pivot="SOADelegatePivot"/>
```

The SOA Client Gateway Agent package supplies two Axis transport "pivot" handlers that you can use depending on your configuration:

1. If you are using Axis with an application running inside a WebSphere application server and wish to use the "lite" configuration then specify the pivot handler as:
   `com.soa.pso.ws.wsclientsdk.lite.AxisLiteGatewayPivot`
2. If you are using Axis with a Java batch application or an application that cannot use the "lite" configuration then specify the pivot handler as:
   `com.soa.pso.ws.handler.wsclientsdk.axis.SMClientGatewayInvoker`

Remember that the pivot handler definition is shared by all services the application might call so the configuration parameters specified will be applied to every service call in your application. Parameters such as `application.username` should be considered. The special parameter shown above (`delegate-handler`) should be specified as shown and allows your application to run even if the Gateway Agent is not fully installed in your environment.

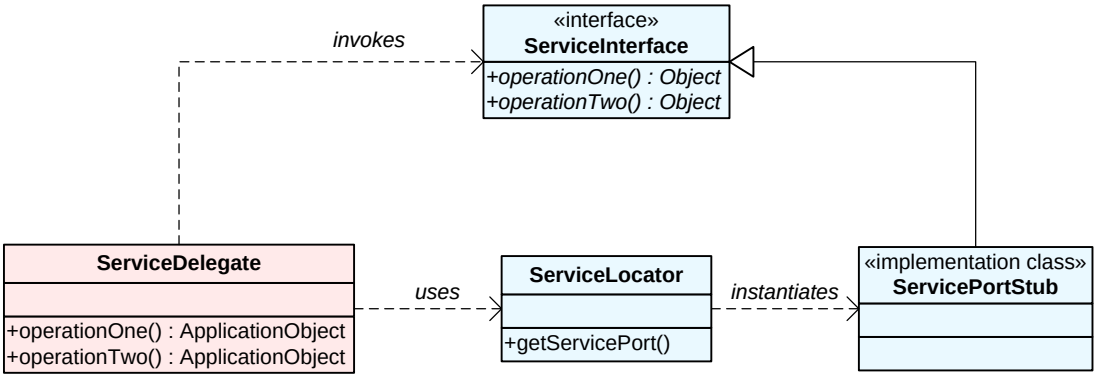
Using the Gateway Agent without J2EE or JSR109

Features are included in J2EE 1.4+ and JSR109 that allow consumer applications to configure Web Services (specifically, the chain of Message Handlers and their configuration properties) by using deployment descriptors that are part of that application but are separate from the application's Java code. These features are used to include the SOA Client Gateway Agent in an application non-intrusively without requiring changes to any of the application's Java code. However, there are some application deployment scenarios that do not allow the J2EE or JSR109 features to be used. A couple of examples are:

- JAX/RPC web service calls being made from a JAR file that is registered as a WebSphere Shared Library.
- A Standalone Java `main()` program
- Earlier, pre-JSR109, JAX/RPC implementations that did not include any client-side deployment descriptors (e.g. WebLogic v8.1)

In these situations, there is no alternative but to make modifications to the Java application code in order to include the SOA Client Gateway Agent in the application. This is because the Gateway Agent is still implemented as a JAX/RPC Message Handler and, without the deployment descriptors, the application must make use of the JAX/RPC `HandlerRegistry` functionality to activate message handlers.

To assist in dealing with these situations, the SOA Client Gateway Agent package provides a feature that eliminates most of the burden of these application changes. It is implemented with the assumption that most web service consumer applications are following the SOA-suggested best practice of always using the "delegate" and "service locator" design patterns to create a "delegate" class in front of the actual, generated web service stub. The delegate would use the generated "service locator" to access the concrete implementation of the "service interface". This design is typical of all JAX/RPC client-side tooling and is shown in the following UML diagram.



The **ServiceDelegate** class would be coded as part of the application logic. Typically, a *delegate* has the purpose of decoupling the application from the implementation details of calling web services. Ideally, as a best practice, the *delegate* should address two issues:

- Handle mapping between the data and objects used in the application and the data formats actually needed by the web service (e.g. mediate between `Date` and `String` or between `Currency` and `Float`).
- Present exceptions in a manner that makes sense to the application. For instance, JAX/RPC web service methods all return `RemoteException`. This is not something that most application business logic should expect to process. The *delegate* should trap that exception, perform whatever logging is necessary and return an `ApplicationException` that indicates that there was a back-end system failure while processing the web service call.

The *ServiceDelegate* would use the **ServiceLocator** (generated by the JAX/RPC tooling that creates the client-side stubs from the WSDL) to create an instance of the generated implementation of the **ServiceInterface** which is also generated from the WSDL. The *ServiceDelegate* would then invoke the web service operations from the *ServiceInterface*.

Enabling SOA Client Gateway Agent with Service Delegates

The SOA Client Gateway Agent provides support for the *ServiceDelegate* model present above by including a class that can be used as the base class for the delegates implemented by an application. The following sample code shows the changes that would need to be made to the delegate classes of an application.

```
. . .
import com.soa.pso.sample.services.AccountManager;
import com.soa.pso.ws.delegate.JAXRPCBaseDelegate;
import com.soa.pso.ws.delegate.SoaConfigurationBean;
. . .
public class AccountManagerDelegate extends JAXRPCBaseDelegate {
. . .
    public AccountManagerDelegate () {
        . . .;
        super.setServiceInterface(AccountManager.class);

        SoaConfigurationBean configBean = new SoaConfigurationBean();
        configBean.setBootstrapURL("http://localhost:39904/wsmex");
        configBean.setApplicationCredential("administrator", "password");
        configBean.setEndUserCredential("enduser1", "password");
        configBean.setServiceBindingKey("account.manager.service");
        super.setConfiguration(configBean);
    }
    . . .
    public String listAccounts () throws ApplicationException {
        try {
            AccountManager stub = (AccountManager)super.getService();
            String result = stub.listAccounts();
            return result;
        } catch (Exception e) {
            throw new ApplicationException("Error in Account Manager",
e);
        }
    }
    . . .
}
```

1 2 3 4 5

1 The two classes shown must be imported.

- *JAXRPCBaseDelegate* is the class provided by the Client Gateway Agent that contains all of the processing needed to find the classes that were generated by the JAX/RPC tooling and to enable the “proper” Message Handler to enable Gateway Agent processing. This will be the base class of the application’s delegate.
- *SoaConfigurationBean* is a Java Bean provided by the Gateway Agent to hold all of the configuration information for the Gateway Agent.

2 All delegates in an application should extend *JAXRPCBaseDelegate*. This will insure that the SOA Client gateway Agent is properly enabled for all web service calls made from the application.

3 The *JAXRPCBaseDelegate* super-class needs to be told what the *ServiceInterface* class (refer to the UML diagram above) is being referenced from this delegate. The super-class uses this to locate the other classes that were generated from the WSDL.

4 An instance of the *SoaConfigurationBean* is created, populated with the desired Gateway Agent configuration information, and handed off to the super-class to be used in setting up the Gateway Agent message handler. It is up to the application and the delegate as to how this configuration information is determined. The super-class does not place any restrictions or requirements on the source or location of any configuration data.

5 Finally, each place the application's delegate would call the *ServiceLocator* to obtainthe "stub" that implements the *ServiceInterface* is replaced with a call to the super-class *getService()* method.

This has illustrated the four required steps a delegate must implement to have the Client Gateway Agent included in its configuration:

1 Extend the *JAXRPCBaseDelegate* base class.

2 Set the *ServiceInterface* class in the super-class.

3 Build a *SoaConfigurationBean* and pass it to the super-class.

4 Call the *getService()* method of the super-class to obtain the *ServiceInterface* implementation stub.

This same processing can be used dto activate the Client Gateway Agent regardless of the execution environment of the application. The *JAXRPCBaseDelegate* super-class dynamically determines the nature of the environment and configures the appropriate client-side message handler. The following environments are supported:

- Inside an application server container with the Gateway Agent Lite available.
- Running in a J2EE Client Application container with remote EJB access to the Gateway Agent Lite.
- Running as a standalone Java program. Note that this environment requires considerably more setup effort. More details are presented later.

Configuring the SOA Client Gateway Agent from a Delegate

The delegate uses an instance of the *SoaConfigurationBean* class to store all of the initial configuration information to be passed to the Gateway Agent. This class has methods to configure any of the features of the Gateway Agent. Here are the methods available to manipulate the *SoaConfigurationBean*.

Category / Method		**Notes**
Bootstrap URLs		
	setBootstrapURLs(String[] urls)	Specify an array of WS/MEX bootstrap URLs
	setBootstrapURL(String url)	Specify the list of WS/MEX bootstrap URLs one at a time.
	addBootstrapURL(String url)	

Category / Method		Notes
Application Credentials		
	setApplicationCredential(String domain, String username, String password) setApplicationCredential(String username, String password)	Specify the application's username, password, and optional "domain" name
End-User Credentials		
	setEndUserCredential(String domain, String username, String password) setEndUserCredential(String username, String password)	Specify the end-user's username, password, and optional "domain" name
	setEndUserCertificate(String domain, String alias, String password) setEndUserCertificate(String alias, String password)	Specify the end-user's certificate or key-pair alias in the Java Key Store. The *setJavaKeyStore()* method must also be called.
	setEndUserCertFile(String domain, String filepath) setEndUserCertFile(String filePath)	Specify the end-user's X.509 certificate that is stored in a .CER file
	setEndUserCredentialFromCookie(String domain) setEndUserCredentialFromCookie()	The end-user's credential will be taken from an SSO Token stored in a HTTP cookie. Please see page 187 for details on capturing the SSO cookie.
	setEndUserCredentialFromContext()	The end-user's credential will be taken from the execution context where a credential was captured from an incoming SOAP request. Please see page 189 for more information on setting this up.
	setEndUserCredentialAuto()	Specifies automatic end-user credential processing mode. This is the default setting. Please see page 191 for details.
	setEndUserCredentialOff()	Turns off any end-user credential processing.
	endUserCredentialIsRequired(boolean flag)	Controls whether the end-user credential is required. The defrault setting is *true*.
Service Binding		
	setServiceBindingKey(String binding)	Specify the "service binding key" that is used to locate dynamic configuration information. Please see page 192 for more information on binding keys.
	setServiceURL(String url)	Specifies a static URL endpoint for the service. Note that this disables all dynamic configuration processing.
	setServiceTransport(int type)	Specified the preferred type of transport to be used when invoking the web service. The following values can be specified: SoaConfigurationBean.TRANSPORT_ANY SoaConfigurationBean.TRANSPORT_HTTP SoaConfigurationBean.TRANSPORT_HTTPS SoaConfigurationBean.TRANSPORT_JMS
Java Keystore		
	setJavaKeyStore(String filepath, String password)	Specify the file path and password for the

Category / Method		Notes
		Java Key Store that will be used to obtain either end-user or SSL mutual authentication key-pairs.
SSL Mutual Authentication		
	setSSLMutualAuthKeyPair(String alias, String password)	Specify the Java Key Store key-pair that should be used for the client side of an SSL Mutual Authentication link. The *setJavaKeyStore()* method must also be called.

Any of the passwords passed into these methods can be an string that has been encrypted using the process documented on page 193.

An instance of the *SoaConfigurationBean* class must be created and passed into the *JAXRPCBaseDelegate* super-class prior to the first call made to the *getService()* method. The configuration bean is not used after the Gateway Agent message handlers have been set up. If you need to dynamically specify some of the Gateway Agent properties with each web service call (end-user credentials for instance), then you can use the following static method:

```
SoaConfigurationBean.setProperty(String name, String value)
```

The valid property names and meanings can be found in the table starting on page 183. Any properties set using this method will be cleared on the next web service call made.

Advanced Service Delegate Base Class Features

The previous sections have covered the basic information needed to use the *JAXRPCBaseDelegate* in the most common use cases. The Gateway Agent feature has a few more capabilities that are useful in specialized situations.

Managing ServiceLocator Caching

JAXRPCBaseDelegate dynamically locates the *ServiceLocator* class that was generated from the WSDL by the JAX/RPC client-side application build tooling. This can be a relatively expensive task in terms of processing overhead. Because of this, the base class has the capability to cache the *ServiceLocator* that was found to process requests for the given *ServiceInterface*. The default setting is for this cache to be enabled and there is a method available to disable the *ServiceLocator* cache is needed:

```
AbstractDelegate.enableServiceLocatorCache(false);
```

Although SOA Software has not yet encountered a JAX/RPC implementation where this is needed (e.g. thread safety issues in the *ServiceLocator* class), this method can be called during delegate initialization to disable all caching of any *ServiceLocator* – not just the one associated with the delegate making the cache-disabling call.

Extending Client Message Handler Processing

JAXRPCBaseDelegate also provides a “hooking” mechanism that can be used to include your own processing the chain of message handlers that will be called for each web service call from the *delegate*. The delegate base class processing insures that the Gateway Agent’s message handler is configured in

the proper sequence along with the "hooks" specified by the application. These hooks are specified using a method included in the delegate base class. This method is:

```
public void addHook (Class hookClass, Map initParms)
```

This method is called during initialization once for each processing "hook" the application needs to have configured. The hooks are invoked in the order they are specified. The arguments to this method are:

Class hookClass is the Java class of the new hook. An instance of this class will be created by the JAX/RPC implementation when needed. This class must directly or indirectly implement *javax.xml.rpc.handler.Handler*. If it does not, it will not be called.

Map initParms is an optional collection of name/value pairs that will be passed into the *init()* method of the hook. This allows the application's delegate to pass configuration or processing options into instances of the hook class when it is initialized

You can also specify the List of hooks that are to be included using the following short-cut method:

```
public void setHooks (List hooks)
```

where:

hooks is a *java.util.List* specifying the hook(s) to be processed. Each entry in the list must be an instance of com*.soa.pso.ws.delegate.HookInfo*. This is a holder class for the hook *Class* and *Map* of initialization parameters to configure the hook. You can create a *HookInfo* object with the following constructor:

```
public HookInfo (Class hookClass, Map initParms)
```

There is also the option of using a "noarg" constructor and the following "setters"

```
public void setHookClass (Class hookClass)
public void setInitParms(Map initParms)
```

Using the Gateway Agent Outside of a SOAP Container

The features of the SOA Client Gateway Agent can also be used for processing web service messages outside the SOAP environment provided by WebSphere or Apache Axis. This capability is referred to as the "Bare Message Feature" and is useful for applications that are generating and processing their own SOAP messages using their own framework similar to JBI and JAX/B. This capability can also be deployed in a non-WebSphere but J2EE-like environment such as SAP NetWeaver or Tibco BusinessWorks.

Installation and Configuration

The Bare Message Feature is included with the basic SOA Client Gateway Agent package. There is no special installation or configuration that is necessary to make its functions available to the client applications. In WebSphere Application Servers, the normal Gateway Agent installation steps presented earlier should be followed. The Gateway Agent can also be deployed in a number of non-WebSphere environments such as Apache Tomcat, Tibco BusinessWorks, and SAP NetWeaver. Please contact SOA Software for special instructions if you are deploying this feature in one of these environments.

Using the Bare Message Feature

The Bare Message Feature of the Client Gateway Agent can be incorporated in your web service client application by adding just a few lines of code to have your soap message processed by the Gateway Agent. Here is a rough outline of the code changes needed:

```
/* ***********************
 * * Initialization code
 * ***********************/
   Properties initParms = new Properties()
   initParms.put(“property.file”, “service-a.properties”);

   BareMsgLiteGatewayHandler msgHandler = new BareMsgLiteGatewayHandler();
   msgHandler.init(initParms);

/* ************************************
 * * To send a request message to the
 * * web service and receive the response
 * ************************************/
   String requestMsg = ...    // construct the message;

   Properties reqProps = new Properties();
   reqProps.put(BareMsgProperties.OPERATION_SOAP_ACTION,
"urn:action:servicea:lookup");

   String responseMsg = msghandler.handleRequest(reqProps, requestMsg);
```

As you can see in this example, the Bare Message feature consists of one Java class, *BareMsgLiteGatewayHandle*r with two methods: *init()* and *handleRequest()*. Details for these are given below.

Initializing the Bare Message Gateway Handler

At least one instance of the *BareMsgLiteGatewayHandler* class needs to be created and initialized for each web service used in the application. This handler operates in a manner similar to the JAX/RPC and Axis Message Handlers that are used to perform Gateway Agent processing inside those SOAP containers. Once an instance of the Handler class is created and initialized, it can be used to process many web service requests. In the case of the *BareMsgLiteGatewayHandler*, each instance is thread-safe and can be shared among many threads but cannot be shared between different class loading contexts.

Once created, the *BareMsgLiteGatewayHandler* instance needs to be initialized. This is done once by calling the *init()* method. This method takes a single parameter which is a *Map* of properties used to configure that instance of the handler. Any and all of the Gateway Agent configuration properties discussed in section 3.2 can be used to configure the processing that will be done by the Bare Message handler.

Processing Bare Message SOAP Requests

Once your application has constructed the SOAP web service request message, you can invoke one of the handleRequest() methods on the initialized BareMsgLiteGatewayHandler object. There are three different versions of this method you can use depending on the form your SOAP request message takes:

1. `Document handleRequest(Map props, Document req)` processes a SOAP request that is an XML Document and returns the SOAP response as another XML Document.
2. `String handleRequest(Map props, String req)` processes a SOAP request that is a String and returns the SOAP response as another String.
3. `byte[] handleRequest(Map props, byte[]req)` processes a SOAP request that is a byte-array and returns the SOAP response as another byte array.

Each of these methods will accept your SOAP request, send it to the web service with processing based on configured management policy, and return the SOAP response in the same form as the request.

Although the processing of the message is controlled by the management policy, there are a few parameters that are sometimes needed that, for the JAX/RPC and Axis handlers are obtained from the SOAP message processing context. Since the Bare Message feature does not have a SOAP container like the other handlers, these must be supplied along with the SOAP request as a *Map* of name/value pairs. This is the first argument to each of the three *handleRequest()* method calls. These properties are supplied with each request instead of the *init()* method because they can possible change from one request to another. This way, one instance of the *BareMsgLiteGatewayHandler* can be used to invoke multiple operations on a single service.

The values that the request processing properties passed to the *handleRequest()* method are in the following table:

Property	Description
operation.namespace operation.localname	Specifies the QName (namespace and local name) of the WSDL binding operation being called
operation.soap.action	The SOAPAction to be applied to the web service call.
context.username context.password	Specifies the optional username and password of the “message context” credential that will be used with credential.type=auto
context.token context.token.name	Specifies an optional SSO token associated with the “message context” that can be located and processed with the credential.type=auto option. This can be specified on one of two ways: context.token=”token-name=token-value” context.token.name=token-name context.token=token-value
context.transport.headers	Specifies an optional *Map* of name/value pairs that will be passed as transport headers (e.g. HTTP headers) along with the SOAP request message. Requirements for a specific Context-type header is an example of when this property would be useful.