Web Services Management Framework - Overview (WSMF-Overview) Version 2.0

16 July 2003

Authors:

Nicolas Catania, Hewlett-Packard Company
Pankaj Kumar (editor), Hewlett-Packard Company
Bryan Murray, Hewlett-Packard Company
Homayoun Pourhedari, Hewlett-Packard Company
William Vambenepe, Hewlett-Packard Company
Klaus Wurster, Hewlett-Packard Company

Copyright Notice

Abstract

Web Services Management Framework (WSMF) is a logical architecture for the management of resources, including Web services themselves, through Web services. This framework is based on the notion of managed objects and their relationships. A managed object essentially represents a resource and exposes a set of management interfaces through which the underlying resource could be managed. Similarly, relationships among managed objects represent relationships among underlying resources.

To better support the various Web service-managed domains, WSMF is model neutral and is designed to be applied to many different domains with varying management requirements.

This is an overview document of a family of documents specifying basic mechanisms for management using Web services (WSMF-Foundation [WSMF-Foundation]), a Web services-based event subsystem (WS-Events [WS-Events]), and management of Web services (WSMF-WSM [WSMF-WSM]).

Status of this Document

The WSMF-Overview specification is an initial public draft release and is provided for review and evaluation only. Hewlett-Packard Company hopes to solicit your contributions and suggestions in the near future. Hewlett-Packard Company makes no warrantees or representations regarding the specification in any manner whatsoever.

Table of Contents

1. Terminology

The following terms will be used throughout this document with the specified meanings.

event:
An *event* is a change in the state of a resource or request for processing.

interface collection:
An *interface collection* is a group of management interfaces that expose the management capabilities of a type of managed object.

managed object:
A managed object is a management representation of a resource. A managed object implements one or more management interfaces to provide a means to monitor and/or control the underlying resource.

management interface:
A management interface exposes management capabilities of a resource. A management interface is presented as a set of attributes, operations, and notifications to be accessed through a WSDL portType.

model:
A model is a set of objects, properties, and their relationships.

notification:
A notification is a message that is sent to or retrieved by one or more subscribers to inform them that an event has occurred.

resource:

A resource is a component of a deployed environment.

relation:
A relation is a type of association between two managed objects.

relationship:
A relationship specifies two managed objects and the relation to define how two specific objects are associated.

Service:
A managed object that implements the Service management interfaces which represents the management capabilities of a Web service. This Web service may be acting as the provider and/or the consumer of Web service messages.

subscriber:
A subscriber is an entity that is interested in selected notifications from managed objects. These notifications contain information about the state change in a managed object.

Web Services Execution Environment (WSEE):
A managed object that implements the WSEE management interfaces which encapsulates the management capabilities of a Web service execution environment.

2. Introduction

This overview document introduces Web Services Management Framework, a management framework to provide a consistent and secure mechanism based on Web services for managing various types of resources, including Web services themselves. The framework takes advantage of the work being done to define the protocols and behavior around Web services and uses this infrastructure to perform management.

The term Web services is used to describe a new approach to distributed computing in which interactions are carried out through the exchange of XML messages. These messages are formatted and transported as per the binding specified in the Web service description document. This architecture allows Web services to be loosely-coupled and highly extensible. Due to these properties, Web services are rapidly emerging as the ubiquitous distributed programming infrastructure. Enterprises are adopting the Web services technology to address their business integration needs and, inevitably, existing management infrastructures need to be extended to manage these new deployments.

The extensibility and loose-coupling provided by Web services are important to the management solutions as well. Today, resources are managed using a variety of incompatible and often proprietary interfaces and protocols. Management information requirements need to be programmed as specific hooks to address new resources and in many cases these hooks are not updated as new versions of the IT resources become available. WSMF addresses this complex situation by defining a standard extensible interface for extraction of management information that makes use of Web services-based industry standardized protocol and description language. In doing so, it provides the advantages of a platform-independent protocol and will evolve to support more complex IT management issues as new standards are created for security, routing, discovery, federation, and others.

Currently, the mechanisms of WSMF are applied to address the management needs for Web services. In future, it will also be applied to additional management domains.

The WSMF architecture offers a way for current and future Web services specifications to provide management capabilities in a standard way. Hence, we use the word "framework" to collectively represent the management for all of these cooperative specifications.

WSMF is specified in three different specification documents:

- **WSMF-Foundation** defines the base framework for management using Web services.
- **WS-Events** defines the Web services based event notification mechanism. This mechanism is used by WSMF-Foundation.
- **WSMF-Web Services Management** defines the model for management of Web services.

This overview document presents the main ideas of WSMF at a high level. In this high-level description, more attention is paid towards achieving clarity of concepts and basic mechanisms than preserving the rigid boundaries of various WSMF concepts. Formal treatment of the WSMF concepts, abstractions and

mechanisms can be found in the respective specification documents. Information on exposing models defined by existing standards such as CIM, JMX, SNMP MIB etc. will be made available as separate documents.

2.1 Management Solutions

Generally speaking, any management solution must address four necessary management questions:

- What resources have to be managed (managed objects)?
- What are their properties (attributes)?
- How is the management information exchanged (operations and notifications protocol)?
- What are the relationships among the managed objects (model)?

Once this structure is in place, any resources within the domain can be managed.

2.2 Why Manage Using Web Services?

So far, IT management has been addressed functionality-by-functionality (fault management, performance management, etc) resource-by-resource (network management, systems management, database management, etc), or even vendor-by-vendor (plug-ins for Oracle database, plug-ins for Sybase, etc). Standards for management (SNMP, CIM, etc) as well as frameworks (policy-based management systems, event correlation systems, etc) have evolved in a similar manner. With all the standards and frameworks in place, management is still by-and-large a tremendously complex problem. Part of the reason for this failure is that the standards and frameworks were developed either to address a subset of management functionality/resources or were targeted at technologies when they were in their early stages. Another part of the reason is that systems and application developers thought of manageability as a secondary aspect - next to core functionality. Only after trying to put these systems together to meet the needs of a business and IT have we realized that it is time to take a second look at management.

Existing management standards (such as SNMP, CIM) are too low level to allow flexible, coordinated interaction patterns. Such flexibility is required for partners to interface with the standards-based software in novel ways (for instance using adaptation mechanisms like resource reservation, allocation/de-allocation or reboots). Obviously, SNMP and CIM primarily focus on data collection, not on writing rich management applications for the adaptive infrastructure. Although there exist a variety of distributed software platforms that could be useful to implement management applications, we argue that Web services technologies form the likely future platform for management, for the same reason that other distributed applications use Web services glue -- language and platform independence, interoperability through wire level standards, industry momentum and the ability to expose interface and hide implementation.

2.3 Why Manage Web Services?

Web services use a common set of middleware standards across diverse domains for allowing various parties to exchange data. The most prevalent use of Web services today is between businesses or inside enterprises to carry business processes and transactions. Any such environment should be managed in order to provide the security, usability, and reliability needed in today's business environment. Hence, use of tools and standards for security, collaborations, process flows, and management are necessary.

Currently there is no standard approach to managing Web services. What management vendors can offer is instrumentation at the SOAP end points and intermediaries (e.g. SOAP servers, UDDI servers, etc.). This will provide information about the Web services as they use these applications. However, this management view is incomplete and lacks critical information on the state of the Web service as it is executing and messages are traveling between various end points. For such information, Web services need to become much more manageable.

2.3.1 Special Considerations for Managing Web Services

Web services have some characteristics that make them especially challenging to manage.

distributed

By their very nature, Web services work together to form a distributed application. This distribution may be across an enterprise or even the public Internet. The challenges here are that there are many view points from which to manage an application. For instance, an IT manager may be interested in managing all of the services in her domain, which includes several machines. The vantage point is all Web services on a given machine. Another example is a process manager may be interested in all of the Web services that work together to drive a process to completion. This may involve Web services throughout an enterprise, or possibly include Web services hosted by other companies. A business manager may be interested in all of the service agreements that her company has regarding Web services hosted by other companies.

extensible

Web services have been designed to be extensible at all levels. The message formatting standard SOAP provides a very flexible mechanism to extend the processing of messages by adding headers to the message. The description standards WSDL and XML-Schema provide a means to define arbitrary data types and Web service interfaces. The management challenges with such an extensible system are discovering exactly what the type of a managed object, or the underlying resource, is and how to communicate with it, for instance discovering all the management capabilities offered by the managed object.

standardized

From the very beginning, Web services have relied on interoperable standards to exchange messages and describe interface capabilities. Many more standards will come into existence over the coming years defining new ways that Web services may be used. There are also organizations, like WS-Interoperability, which are working to reconcile the many Web services standards. What all of this means is that not only are the basic messaging mechanisms standardized, but in many cases the extensions to the basic mechanisms describing advanced capabilities are also standardized. The management challenges for such an extensively standardized environment will be in defining a management standard that does not violate any of the existing standards and provides all the flexibility needed to effectively manage the environment.

discoverable

As Web services have evolved, there is a stronger push to have all Web services discoverable. This discovery may happen at either design time or at run time. Web service endpoints can be discovered at runtime, which provide an interface that the consumer already understands. This discovery mechanism is not limited to use only within an enterprise, but is available as far as the Web service is available, including the Internet. One management challenge with such a powerful discovery mechanism is to make sure that provided services are protected with the right type of security in the right places. Another management challenge is ensuring that there is a way of determining who is making use of the provided services. Yet another more complex management challenge is that all appropriate service agreements are not only discoverable but enforced.

3. Management with WSMF

WSMF provides a management framework that is based on managed objects, event notifications, and relationships.

3.1 Managed Objects

A managed object provides management capabilities by implementing management interfaces. The management interfaces are described using WSDL. Hence, the managed object themselves are Web services. Resources that can be represented by managed objects with WSMF-compliant management interfaces can be managed through WSMF. A manager can use WSMF to manage all the resources of a domain in a uniform way.

When applying management to Web services, WSMF takes into account the special considerations inherent in Web services. For example, a Web service that wants to be managed through WSMF can directly implement WSMF compliant management interfaces and avoid the need for a separate managed object Web service.

The capabilities described by WSMF include a number of management functions, including:

- Discovery of the management WSDL definitions
- Discovery of the relations and event notifications of the managed objects
- Registrations and retrieval of event notifications
- Monitoring, auditing, and controlling various aspects of managed objects by utilizing the supported management operations

Managed objects may be implemented as part of the managed resources themselves or provided in a separate layer. The interface between the managed object and its resource is part of the managed object implementation and is not defined by WSMF. However, there is one management interface that must be implemented by all managed objects -- ManagedObjectIdentity interface. This interface allows a managed object to be uniquely identified. Managed objects may choose from a variety of other management interfaces to support the management capabilities appropriate for the underlying resource. An interface inheritance mechanism has been defined such that a managed object that implements a derived interface must also implement the base interface.

3.2 Interfaces and Interface Collections

Interfaces are descriptions of a set of related management capabilities of managed objects. Managed objects implement interfaces to provide management capabilities for the managed resources that they represent. Management interfaces are grouped into collections of interfaces to describe a group of management capabilities about a specific type of managed object. Interfaces can be extended to define new interfaces with more capabilities. WSMF provides a base interface collection that describes many management capabilities common to all managed objects. Currently, there is one management interface that must be implemented by all managed objects; this interface is called `ManagedObjectIdentity`. Additional management interfaces may be implemented as needed for the particular type of resources represented. Refer to *WSMF-Foundation* for the complete ManagedObject interface collection. A collection of interfaces group has a name that is unique in its context. This group contains the following:

- attributes: The set of properties representing information about a managed object.
- operations: The set of functions that can be provided to support the management of a managed object.
- notification types: The set of events and state changes that can be reported by a managed object.

As an aid to grouping attributes and operations into related sets and establishing a common semantic meaning for the type of the operations, WSMF has defined six categories based on categories defined by OSI. The categories are:

- monitoring
- discovery
- control
- performance
- configuration
- security

The ManagedObject interface collection defines management interfaces for these categories. When defining interfaces for a new type of managed object, the new interface should inherit from the interfaces defined for ManagedObject when applicable. It is recommended that new operations be added to these categories as much as possible. It is allowed to define new interfaces, if necessary. Separating all of the features of a single managed object into multiple management interfaces is useful in many respects:

- They group subsets of management features into bundles that are commonly accepted in the management space and help the ISV and the management client to use a common language to exchange management requests and responses.
- The separation of features allows a managed object to provide management capabilities in a gradual way by supporting one interface, such as ManagedObjectMonitoring, initially and working their way through the other management interfaces over time.
- Management interfaces allow resource owners to selectively decide what management information should be provided to what manager role by exposing selective interfaces. This allows a managed object to expose all management interfaces to an administrator, and a restricted subset to all other users.

Attributes support different access policies -- read-only and read/write. Management interfaces are mapped to WSDL portTypes, operations and access of attributes to WSDL operations and access policies are applied to attributes. For example, a read access policy for an attribute will translate into a `GetXXX` WSDL operation in a portType, where XXX is the name of the attribute. A read/write access policy translates in a pair of operations: `GetXXX` and `SetXXX` for attribute `XXX`.

Interfaces can extend other interfaces to provide further richness in their management capabilities. WSMF enabled managed domains can create their own interfaces (by extension of existing WSMF interfaces or directly) to better expose the management capabilities of their managed object implementations.

3.3 Relationships

Relationships support the model that includes the association between managed objects. Relationships define the behavior and the dependencies between managed objects and consequently the resources that they represent. A relation describes the type of association between two managed objects. Each relationship is a specific instance of a relation and describes a relation and the two managed objects. Relations have an identifier unique to the management scope. WSMF defines some basic relations in the WSMF namespace. Other relations may be defined in other namespaces to describe associations that are important in other management domains.

3.4 Events and Notifications

An event subsystem is an integral part of any management solution; however, currently there is no widely adopted Web services-based standard for such a subsystem. After investigating several event subsystem proposals, WSMF has defined its own Web-services based event subsystem to better meet its requirements. A detailed and formal description of this subsystem can be found in the *WS-Events* specification document.

A brief overview of the event subsystem defined by WS-Events is presented below.

WS-Events defines operations to subscribe for event notifications and the notification syntax and processing rules to inform one or more subscribers that an event has occurred. An event is a state change in a managed resource or processing of a request by the managed resource that can be communicated through a notification. There are two modes to get these notifications: a push mode and a pull mode. In the push mode, when an event occurs, the managed object sends a notification to the subscribers to inform them of a change of state. In the pull mode, the subscriber issues calls to the managed object to request all the notifications that happened since the last pull call.

For efficiency and scalability reasons, WSMF allows bulk subscription and notification operations. In the push mode, the management client can subscribe to more than one notification type in a single

subscribe call. Also, in the pull model, notifications from more than one type can be retrieved through a single call.

It is important to understand the tradeoffs between the push and pull usage modes; the push mode tends to be more network intensive while the pull model may require more computation and memory resources at the managed object. Also, the pull mode makes it possible to apply WSMF to scenarios where the managed object may not be able to establish connection with the subscriber, say, when it is behind a firewall.

3.4.1 Notification Format

Notifications are used in WSMF to capture and deliver the event information from managed objects. Each notification is assigned a unique identifier.

One or more notifications can be packaged within a single message.

3.4.2 Subscription Operations

The operations for the push and pull modes are grouped in two different interfaces. In both modes, a subscribe call is used to register an interest for one or more event types. In the push version, an extra callback URL is needed to identify the end point that will receive the notifications.

For scalability reason, a subscription has an associated expiration time. It is up to the subscriber to ensure that its subscription is renewed before it expires if the subscriber is still interested in the corresponding notifications.

A successful subscription call returns a unique `SubscriptionId`. The `SubscriptionId` acts as a handle and is passed as a parameter for all other notifications related calls. There are operations to terminate any previously registered subscription.

In the pull mode, three calls are provided to retrieve notifications. They can be used to get notifications generated (i) after a particular notification, (ii) after a particular time, or (iii) within a time duration.

3.5 Faults

WSMF supports a simple request response model. A WSMF client using SOAP binding issues requests as SOAP messages and receives SOAP messages in response. If there is an error in the request or with the processing of the request, a SOAP fault message is returned instead of the response. The contents of a SOAP fault is defined in the SOAP specification.

The SOAP fault detail element will contain additional information for faults related to WSMF errors in the form of an Error element in the WSMF namespace. The WSMF Error element includes the error code and a descriptive error message. The WSMF Error may contain multiple error elements to describe multiple failures.

The details of WSMF faults are described in the WSMF-Foundation specification.

3.6 Scalability

In the area of scalability, the current WSMF focus is on optimizing WSMF management information traffic. Future work will address other areas of scalability as well.

As systems become large, there will be many resources that need to be managed. Often, a manager will want to perform the same action on many managed objects. Examples include getting the same attribute from some set of managed objects, or invoking the same operation on many managed objects. In order to provide this capability, WSMF has defined a ManagedObjectCollection interface that may be implemented by any managed object. This interface can be used to manipulate multiple attributes of a managed object in one request. Also, it can be used to execute an operation on multiple managed objects with one request. The object exposing the ManagedObjectCollection interface, also referred to as the collection manager, keeps track of the members of the collection. When a manager wants to perform an action on members in the collection, it will invoke an operation of the

ManagedObjectCollection interface. The collection manager takes the responsibility to make sure the action is performed on each of the specified members of the collection and return the results to the manager. In this way a manager makes a single request and takes action on many managed objects. The collection manager may take action on each of the members of the collection individually, or may do some smart caching so it does not need to contact each individual member for every request, but WSMF does not make any requirements as to how the collection is implemented.

A good example of where the collection interface could be applied is the Web Services management domain. This domain includes a WSEE (Web Services Execution Engine) managed object that represents the container of Web services. By having a WSEE managed object implement the collection interface, it is possible to invoke the same operation on a large number of Web services with a single request message. Another example is a case where a managed object for a Web service provides a Collection interface for all of the conversations with other web services in which this service participates. In this case, all of the related conversations would be members of this collection. A manager could then take action on all of the relevant and current conversations, without having to request the list of conversations first and worry about whether an individual conversation had terminated before the manager was able to take action on it.

WSMF has addressed this type of scalability issues in other areas as well. For instance, in the event interfaces, it is possible to subscribe for many event types in one request-response exchange rather than requiring a manager to subscribe individually for every event type.

3.7 Security Consideration

Security for Web services is a big concern for the industry. Here is a non-exhaustive list of potential threats:

- A malicious program could pose as a management console and harm the managed object through unauthorized use of the WSMF-APIs.
- A person-in-the-middle attacker could delete or alter information sent by the ManagedObject to the manager, thus giving the manager a false view of the state of the managed object. Likewise, the attacker could delete or alter the control information from the manager to the ManagedObject.
- A third party could claim to be a ManagedObject and send false management information to the manager, providing a wrong representation of the ManagedObject.

Since WSMF is built on top of Web services technology, it leverages the Web services security technologies for the security of the framework.

3.7.1 Payload Security

With HTTP binding, HTTPS is the recommended transport protocol when integrity and/or confidentiality of the SOAP payload is required. HTTPS has been around for a while now, is widely accepted and provides a robust way of securing the transport for exchange of request, response and notification messages. This should cover a wide variety of deployment scenarios. However, transport-level security may be inadequate for cases where end-to-end security is needed.

In situations where transport-level security is not adequate, WS-Security, as and when it becomes an OASIS standard, may be used to secure SOAP messages.

3.7.2 Authentication and Authorization

With HTTPS as transport, the SSL certificates can be the basis for the server, and optionally, the client authentication.

With WS-Security, security tokens can be used for authentication.

Finally, even though communication within Web services deployments is largely system-to-system based (machines talking to other machines), the more human-oriented HTTP basic authentication

(username/password) may be used when the security requirements are very light or when cryptography is not suitable for the environment (too heavyweight).

Access control may be applied to the WSMF APIs at the interface or operation level. Specifics of specifying and enforcing the access control rules are implementation dependent and are not dictated by WSMF.

The details of how a manager and a managed object negotiate to agree upon a particular security mechanisms is yet to be worked out. In the meantime, this will happen through out of band means.

3.8 Model Neutrality

Currently, there are many standards and methodologies for modeling the management information. The choice of the modeling methodology is based on the particular domain and the resources that the domain contains. Hence, WSMF will not pick any particular modeling methodology and remain neutral by acting as a normalization layer that allows various domain model information to be extracted. Moreover, once all the modeling information is expressed through WSMF, WSMF acts as a correlator and aggregator of the various management information in a unified end-to-end view of the management domain.

WSMF achieves its neutrality by providing attributes that can map to state and meta-state data (e.g. Status, Type) and operations through which relation information between various managed objects can be discovered (e.g. getRelationships, getServices, getConversations). Since the information from various domains will be extracted and represented in a common way, this information can be correlated by the WSMF manager client to create an aggregated view of the managed objects, their state information and how they are all inter-related.

The model information can be provided to WSMF in two ways; operations that support attribute values that represent static and dynamic values related to states and relations (e.g. getStatus, getType, getRelations), and operations that support the dynamic state and dependencies of managed objects (e.g. notifications).

In the future, further examples and white papers will be provided on how the management information from specific models like MIB/SNMP, CIM/WBEM, and JMX/JSR77 to WSMF can be achieved.

4. Mapping WSMF Management Interfaces into WSDL

In addition to providing interface definitions, WSMF provides a mapping of those interfaces to a set of WSDL elements. Currently, version 1.1 is the latest version of the WSDL standard and WSMF provides a mapping based on this version. However, WSMF will track important changes (e.g., interface inheritance) in the WSDL specification and incorporate them as necessary. The following WSDL elements are defined by WSMF:

- schemas for WSDL parts
- portTypes
- marker attributes
- namespaces

The specifics can be found in the WSMF-Foundation, WS-Events and WSMF-WSM specifications.

5. Managing Web Services

WSMF describes a framework for managing IT resources. To address the manageability of Web services and also to provide an example of mapping this framework to a specific domain, it includes the managed object definitions for the management of Web services. Hence, it includes applying the core WSMF ideas for the management of Web services. For managing Web services, a minimum set of resources has been identified that should be managed based on how Web services are used. These resources are Web service, Web service execution environment, and conversation. Applying the WSMF

management framework, three interface collections are identified. Additionally, a set of relations to support the model through which the resources interact is created.

The use cases section also provides specific examples of how the Web services management model that is described here is used.

5.1 Web Service Management Interface Collections

The WSMF model for Web services management currently supports the following interface collections:

- Service: A description of the management capabilities of a Web service resource.
- WSEE: A description of the management capabilities of a Web services execution environment.
- Conversation: A conversation resource is a service's view of the state associated with a set of related messages. This interface collection represents this resource's management capabilities.

Refer to *WSMF-WSM specification* for the full WSMF Web service interface collection definitions.

6. End-to-End Management

An end-to-end management view will not only need to deal with Web services but also the applications and infrastructure that support Web services. As stated throughout this document, one of the goals of WSMF is to provide a common channel via which resources can provide management information. However, it is realized that, realistically, WSMF will co-exist with other existing management channels -- WSMF, SNMP, JMX, legacy applications, etc.

7. Use Cases

Two use cases are presented to demonstrate the value of the Web services Management Framework architecture.

7.1 Traditional Management Scenario

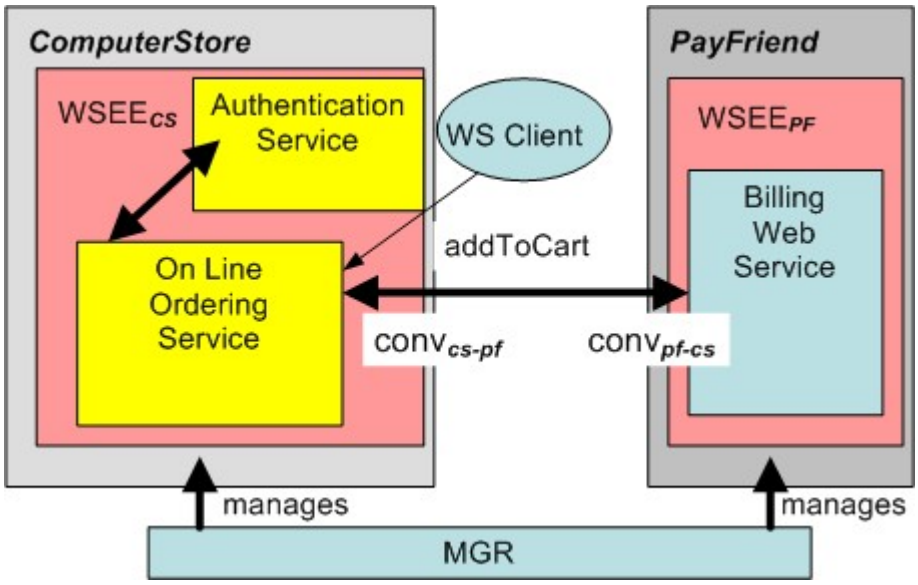
Composition is key to the Web services value proposition. While loosely coupled services enable a cross-domain collaboration, it makes management very difficult because loose coupling is contrary to the notion of a single management domain current management solutions have a unified view of what they manage.

This particular use case illustrates root cause analysis when things go wrong in a highly distributed and loosely-coupled environment. It describes an online shopping service ComputerStore that uses two other services:

- an authentication service operated by the same ComputerStore
- a shopping cart/billing service provided by a 3rd party company PayFriend.

There is a conversations taking place between the OnLine Ordering Service and the Billing Web Service. Notice that the same conversation is labeled differently at each end to reflect the difference in view points.

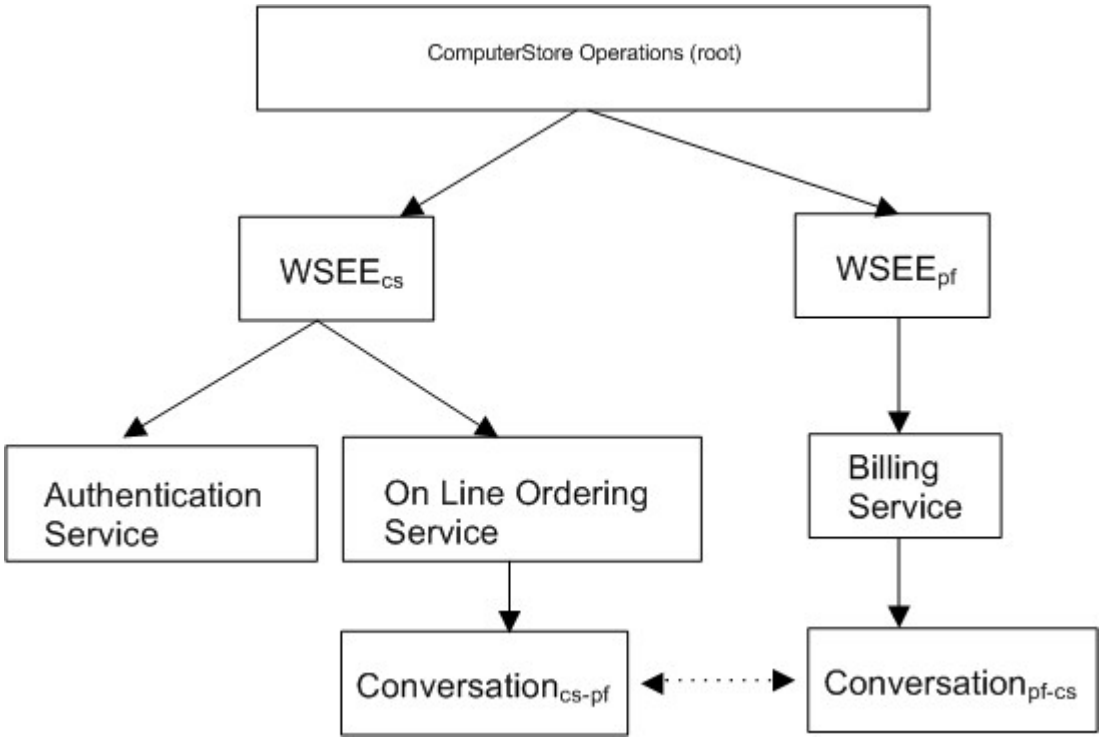
The following picture depicts a scenario where all the sub-services of the composite service belong to the same management domain. There is only one manager to manage the services. This corresponds to an outsourcing situation.



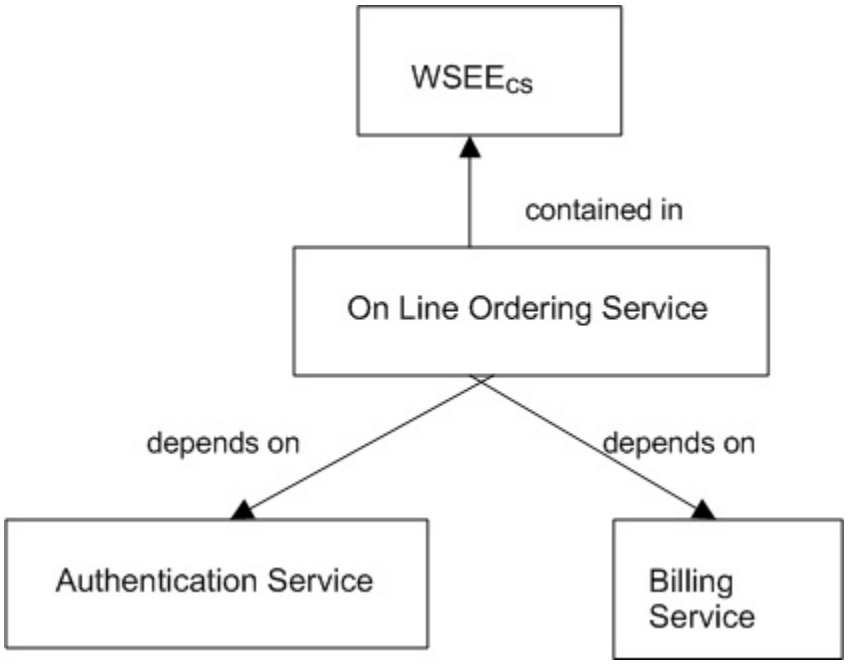
7.1.1 Deployment Discovery

The manager could have the managed objects management interfaces (WSDLs) listed in a configuration file or it could discover these with a UDDI lookup. Once a managed object has been discovered, the manager invokes the GetRelationships method to discover other related managed objects. The method GetContainer is available to retrieve the execution environment for the Web service.

Once the relationships between the managed objects have been discovered, they could be depicted in a traditional bottom-up view as depicted in the following image:



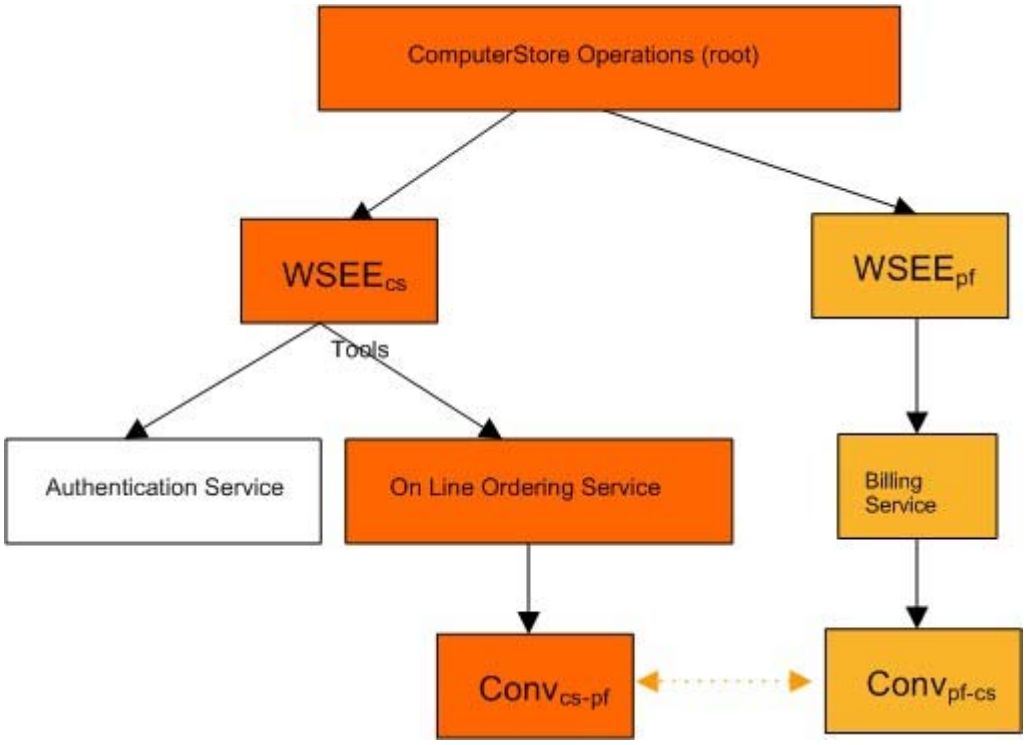
Another more service centric view is depicted in the following image:



7.1.2 Problem Detection

Once the topology of the managed system has been discovered the manager uses the Subscribe call on the managed objects to get failure notifications.

Assume that the manager receives a failure notification from convcs-pf with status change to critical. This will subsequently change the status values of all other objects that can be affected, using a simple status propagation rule that changes the status of the parent node to the highest status value of its children. This object's status will immediately be reflected and update the status of all the other objects in my view that are related to this object. This is how a problem in convcs-pf is propagated to the Root Object.

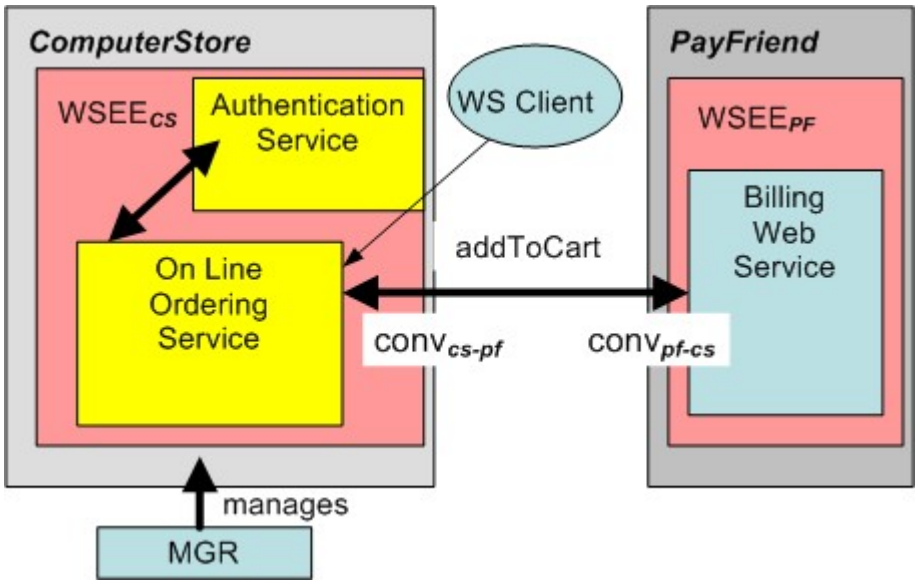


7.1.3 Problem Resolution

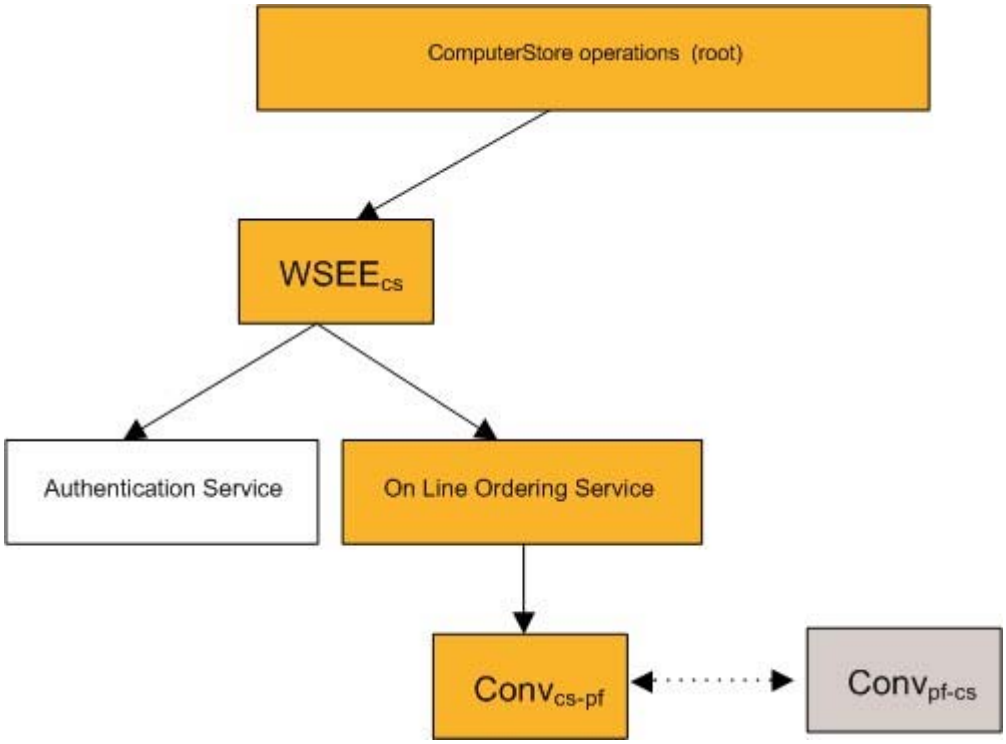
The manager starts from the Root object and asks for the root cause of the problem. It finds that notifications are associated with the convcs-pf object. After inspecting the notifications, their status values, and their time stamp, one determines which notification caused the status change and determines what data are provided. The operator runs the corrective actions and solves the issues. A notification is received from convcs-pf with status back to normal. All the other objects in the tree return to normal.

7.1.4 A Different Twist

However, this cannot be the general case and we have to assume that the ComputerStore manager may not be able to manage PayFriend because it is in a different management domain. In this situation the manager has only partial view and control of the composite service.



In this scenario, the topology of the managed objects looks like this:



7.1.5 Problem Resolution -- The Manager Does Not Cover PayFriend

Now assume that WSEEpf goes down. The problem shown at the Root indicates that convcs-pf is the root cause of the problem. The manager drills down to convcs-pf and find notifications regarding convcs-pf. After inspecting the notifications, their status values, and their time stamp, the manager determines which notification caused the status change. In our case convpf-cs is not responding. Since it is out of the management domain the corrective actions are limited (e.g. wait till it comes back, send message to PayFriend mgr or kill convcs-pf and re-initiate conversation with similar service).

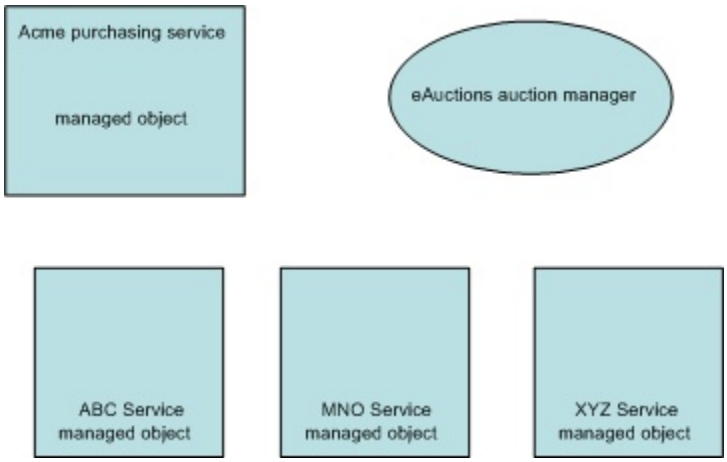
7.2 Management As a Service

Actors: eAuctions auction monitoring manager, ABC Company, MNO Corp., XYZ Inc. Web services, Acme Airlines purchasing agent.

Goal: To monitor the progress of a distributed business process

Eager to move more into the B2B market, eAuctions, an online auctions company, offers a new management service that can monitor the progress of a distributed quote process. To do so, eAuctions has defined a process to follow to send a RFQ and get bids from vendors using BPEL. The process is executed in a distributed environment without a central controller; eAuctions runs the monitoring manager while all the business logic and operational work happens through various Web services involved in the process, not on an eAuctions-hosted server. In other words, if eAuctions stepped out of the picture, the process would not be monitored but nothing would prevent it from still being executed.

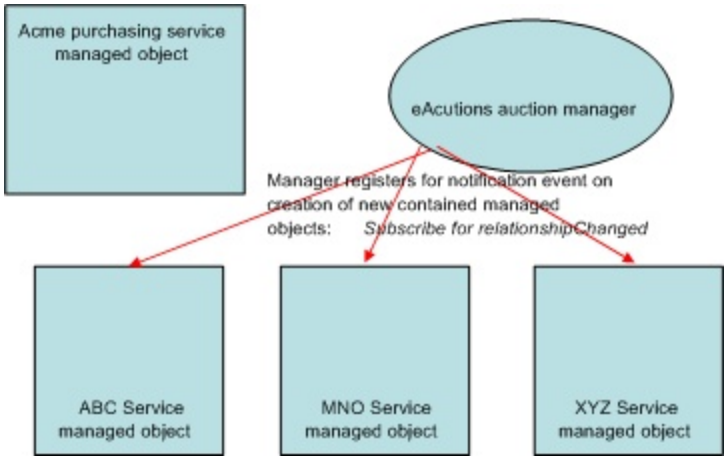
eAuctions has an agreement with ABC Company, MNO Corp. and XYZ Inc. in which eAuctions defines a BPEL process for anyone to submit a quote to these three vendors. The process then describes the flow of messages for the bidding, until the process terminates. eAuctions has agreed to monitor the process.



We assume that someone (WSMF, W3C, other) has extended WSMF to define a BPELProcess managed object. This extension might live in the BPEL namespace or some other namespace (other than the WSMF namespace). This managed object is an extension on the Conversation managed object defined by WSMF. The different services involved in the business process have agreed to implement WSMF management interfaces and are aware of the BPELProcess managed object. For this use case, we assume that the BPELProcess managed object includes the following elements (plus many others that are not used in this example):

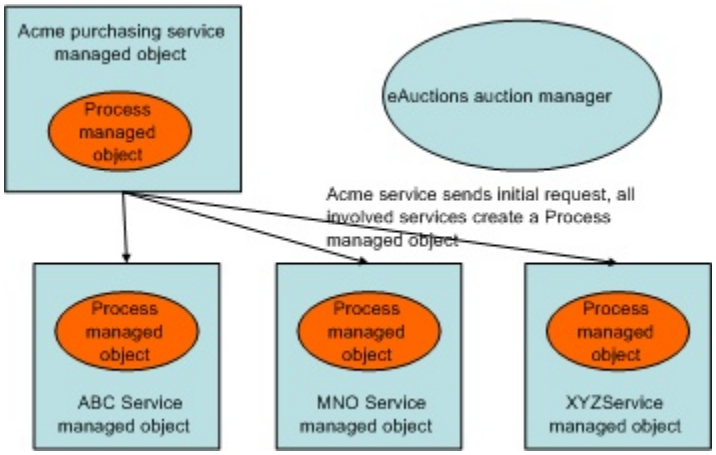
- `GetGlobalProcessID` operation (returns the global process ID for a BPEL process instance, the global process ID is the URI contained in the Context/Identifier element defined by WS-Coordination)
- `GetBPELProcess4ID` operation (returns the BPELProcess managed object for a specific global process ID)
- `ProcessStepCompleted` notification (whenever a step in the BPEL process is completed)

In addition, the use case also use operations and notification defined in the Conversation and ManagedObject interfaces as the BPELProcess interface extends them.When the agreement between eAuctions, ABC, MNO, and XYZ was made, the three vendors each provided eAuctions with a URL pointing to the management WSDL for the service that will represent them in the transaction. Using this representative URL makes discovery of the management WSDL very simple; it requires the management technology to simply dereference the URL. Having retrieved these management WSDL, the eAuctions manager finds that the services all expose a registration service that allows eAuctions to register for notification when a new BPELProcess managed object is created in the service. eAuctions registers with notification for all three services. This is done by registering for the relationshipChanged notification on each of the services (relationshipChanged is a notification in the ManagedObject interface).

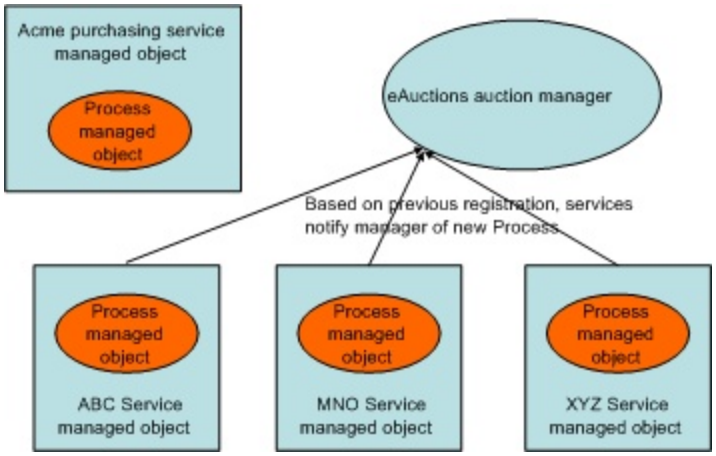


The Acme purchasing service knows about this available process and decides to use it to buy a set of Unix servers at a competitive price from one of the three vendors. The Acme purchasing agent knows the BPEL document that describes the process and also exposes some WSMF management interfaces. It also knows about a management service that monitors this BPEL process but it does not necessarily know the identity of the management service.

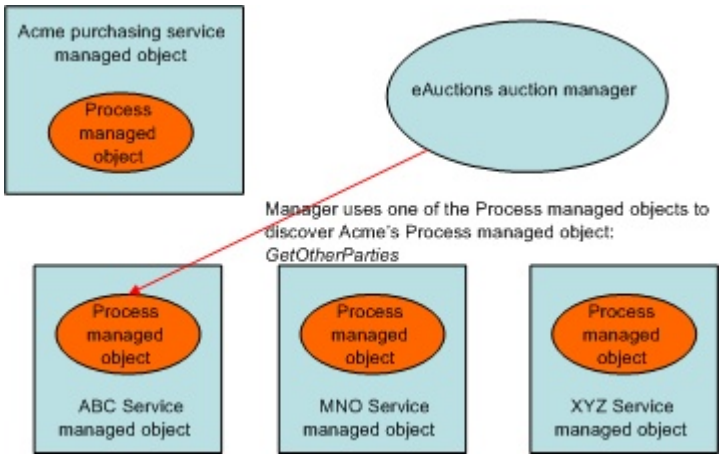
Based on what the BPEL document says, the Acme service sends an RFQ document to the three vendors. Upon receiving this message, the ABC service creates a new BPELProcess managed object to represent its view of this process. Because the eAuctions manager has registered for notification on creation of such a new managed object, the ABC service also sends a notification to the eAuctions manager. The notification includes a link to the management interface of the BPELProcess management object so that the eAuctions manager can invoke the WSMF operations it exposes.



The XYZ and MNO services do the same thing as the ABC service. The eAuctions manager thus receives three such notifications. When such a notification arrives, the eAuctions manager retrieves the management WSDL for the BPELProcess managed object using the information in the notification. Having done this, it calls the getGlobalProcessID operation on this managed object. This call will return the same global ID for the ABC, MNO and XYZ BPELProcess managed objects, thus allowing the eAuctions manager to logically group them together as representing the same process instance. This stated ID is provided by the use of the WS-Coordination Context on the wire, as specified by BPEL. More specifically, it is the "Identifier" element (a URI) of the Context type defined by WS-Coordination.



The eAuctions manager invokes the getOtherParties operation on the Conversation managed object for one of the three vendors (remember that BPELProcess extends ManagedObject, therefore the BPELProcess managed object we are using also provide the management capabilities described in the Conversation interface). This returns the three vendors (assuming that the vendor to which this request is sent knows about the other vendors, it doesn't have to be the case, based on how the BPEL process document is constructed) plus the Acme service (whether or not they know about the other vendors, each vendor will know about the Acme service as it exchanges message with Acme). This allows the eAuctions manager to get hold of the Acme service. Using this information, the eAuctions manager retrieves the management WSDL for the Acme service. It then invokes the getBPELProcess4ID operation, passing the global process ID provided by the vendors. This returns to the eAuctions manager a link to the BPELProcess managed object for the Acme service.

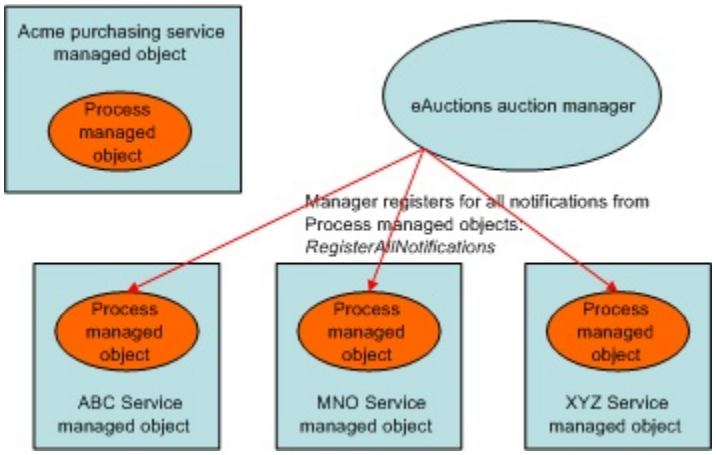


In the case where the vendors don't know of one another existence (depending on how the BPEL Process is designed), the eAuctions manager can now call getOtherParties on the BPELProcess managed object for the Acme service. This will return all three other services. It can then call getBPELProcess4ID on each to retrieve all the BPELProcess managed objects involved in this process instance.

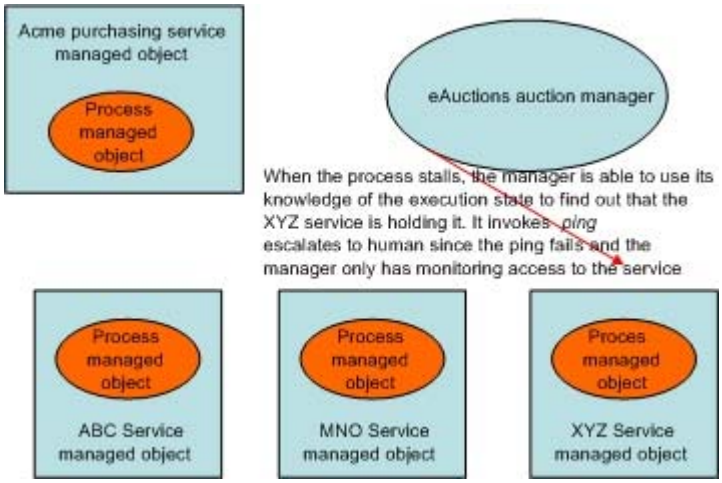
At this point, discovery is completed and the eAuctions manager has gotten hold of all 4 services managed objects and all 4 BPELProcess managed objects.

The eAuctions manager then uses the management interface on each of these BPELProcess managed objects to register for notification every time a new step is completed in the process. This way, the eAuctions manager can monitor the progress of the process.

At some point, the process is stalled because the XYZ service is not sending a message that is expected.



Having seen the process stall for a certain period of time, the eAuctions manager uses its knowledge of what step the process is at (because the manager has been kept updated by all the agents) to find out that the process is stalled because of the XYZ service. Using the management interface for the XYZ service, the eAuctions manager sends a getStatus message to the XYZ service, which does not reply. This is reported to a human operator at eAuctions who picks up the phone and calls a XYZ contact person to solve the problem. After getting this phone call, XYZ fixes the technical problem and the transaction can complete.



In this use case, WSMF was used to:

- Finding the management operations offered by a managed object
- Register for notification upon creation of a new managed object
- Send notification of the creation of a new managed object
- Retrieve the global process ID of a BPELProcess managed object
- Get a list of services involved in a BPEL process
- Register for notification on completion of a process step
- Ping a service

8. References

[XML 1.0]
Extensible Markup Language (XML) 1.0 (Second Edition), T. Bray, J. Paoli, C. M. Sperberg-McQueen, and E. Maler, Editors. World Wide Web Consortium, 10 February 1998, revised 6

October 2000. This version of the XML 1.0 Recommendation is http://www.w3.org/TR/2000/REC-xml-20001006. The latest version of XML 1.0 is available at http://www.w3.org/TR/REC-xml. (See http://www.w3.org/TR/2000/REC-xml-20001006.)

[XML Namespace]
Namespaces in XML, T. Bray, D. Hollander, and A. Layman, Editors. World Wide Web Consortium, 14 January 1999. This version of the XML Information Set Recommendation is http://www.w3.org/TR/1999/REC-xml-names-19990114. The latest version of Namespaces in XML is available at http://www.w3.org/TR/REC-xml-names. (See http://www.w3.org/TR/1999/REC-xml-names-19990114.)

[XML Schema: Structures]
XML Schema Part 1: Structures, H. Thompson, D. Beech, M. Maloney, and N. Mendelsohn, Editors. World Wide Web Consortium, 2 May 2001. This version of the XML Schema Part 1 Recommendation is http://www.w3.org/TR/2001/REC-xmlschema-1-20010502. The latest version of XML Schema Part 1 is available at http://www.w3.org/TR/xmlschema-1. (See http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/.)

[XML Schema: Datatypes]
XML Schema Part 2: Datatypes, P. Byron and A. Malhotra, Editors. World Wide Web Consortium, 2 May 2001. This version of the XML Schema Part 2 Recommendation is http://www.w3.org/TR/2001/REC-xmlschema-2-20010502. The latest version of XML Schema Part 2 is available at http://www.w3.org/TR/xmlschema-2. (See http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/.)

[SOAP 1.1]
Simple Object Access Protocol (SOAP) 1.1, D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. Frystyk Nielsen, S. Thatte, D. Winer, Editors. World Wide Web Consortium, 8 May 2000. This version of the Simple Object Access Protocol 1.1 Note is http://www.w3.org/TR/2000/NOTE-SOAP-20000508. The latest version of Simple Object Access Protocol 1.1 is available at http://www.w3.org/TR/SOAP. (See http://www.w3.org/TR/2000/NOTE-SOAP-20000508/.)

[SOAP 1.2]
SOAP Version 1.2 Part 1: Messaging Framework, M. Gudgin, M. Hadley, N. Mendelsohn, J-J. Moreau, H. Frystyk Nielsen, Editors. World Wide Web Consortium, 19 December 2002. This version of the SOAP Version 1.2 Part 1 Specification is http://www.w3.org/TR/2002/CR-soap12-part1-20021219/. The latest version of SOAP Version 1.2 Part 1 is available at http://www.w3.org/TR/soap12-part1/. (See http://www.w3.org/TR/2002/CR-soap12-part1-20021219/.)

[WSDL]
Web Services Description Language (WSDL) 1.1 , E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, Authors. World Wide Web Consortium, 15 March 2002. This version of the Web Services Description Language Note is http://www.w3.org/TR/2001/NOTE-wsdl-20010315. The latest version of Web Services Description Language is available at http://www.w3.org/TR/wsdl. (See http://www.w3.org/TR/2001/NOTE-wsdl-20010315.)

[WSDL 1.2]
Web Services Description Language (WSDL) Version 1.2 , R, Chinnici, M. Gudgin, J. Moreau, and S. Weerawarana, Editors. World Wide Web Consortium. The latest version of Web Services Description Language, Version 1.2 is available at http://www.w3.org/TR/wsdl12. (See http://www.w3.org/TR/2003/WD-wsdl12-20030303.)

[WSMF-Foundation]
WSMF-Foundation, B. Murray, Editor. Web Services Management Framework, Hewlett-Packard.

[WS-Events]
Web Services Events, N. Catania, Editor. Web Services Management Framework, Hewlett-Packard.

[WSMF-WSM]
WSMF: Web Services Model, W. Vambenepe, Editor. Web Services Management Framework, Hewlett-Packard.