Web Services Management for Adaptive Control

Sven Graupner, Tilo Nitzsche
Internet Systems and Storage Laboratory
HP Laboratories Palo Alto
HPL-2004-94
May 20, 2004*

E-mail: sven.graupner@hp.com, tilo.nitzsche@hp.com

Web Services, management, adaptive, control, Utility Data Center, Resource Flexing, Grid, standard

With the increasing use of web services technology in the adaptive enterprise, web services management is becoming critical. Emerging Grid technology is based on web services as well. Problems that need to be addressed are: how can distributed web services be managed and how can a linkage be established between managed web services and management systems such as HP OpenView (OV). The Web Services Management Framework (WSMF) has been developed by HP and has recently merged into the OASIS Web Services Distributed Management (WSDM) standard, which is still under development.

This paper presents a use case how WSMF was used to manage the web services components of the Automatic Flexing Interface (AFI) that has been developed as external control interface for the Utility Data Center based on standard web services. AFI allows external control systems to flex server groups for horizontally-scalable applications. AFI allows control loops through monitoring, assessment of monitored data, and a decision about a corrective action that results in a control instruction which is sent back into the UDC via AFI adjusting the number of servers in a server group.

A key contribution of this work was to demonstrate how WSMF was used to connect web-services components of AFI to OpenView's Service Navigator allowing the visual presentation of control states and allowing operators to perform control operations through Service Navigator's control console. The control system with WSMF management has been demonstrated internally in HP in Nov 2003. AFI is scheduled for the UDC 2.0 product release.

* Internal Accession Date Only
To be published in HP Tech Con '04, 20-23 June 2004, Orlando, FL, USA
Approved for External Publication

Web Services Management for Adaptive Control

Sven Graupner, Tilo Nitzsche

OST&T, HP Labs / SGBU MSO

sven.graupner@hp.com tilo.nitzsche@hp.com

Abstract

With the increasing use of web services technology in the adaptive enterprise, web services management is becoming critical. Emerging Grid technology is based on web services as well. Problems that need to be addressed are: how can distributed web services be managed and how can a linkage be established between managed web services and management systems such as HP OpenView (OV). The Web Services Management Framework (WSMF) has been developed by HP and has recently merged into the OASIS Web Services Distributed Management (WSDM) standard, which is still under development.

This paper presents a use case how WSMF was used to manage the web services components of the Automatic Flexing Interface (AFI) that has been developed as external control interface for the Utility Data Center based on standard web services. AFI allows external control systems to flex server groups for horizontally-scalable applications. AFI allows control loops through monitoring, assessment of monitored data, and a decision about a corrective action that results in a control instruction which is sent back into the UDC via AFI adjusting the number of servers in a server group.

A key contribution of this work was to demonstrate how WSMF was used to connect web-services components of AFI to OpenView's Service Navigator allowing the visual presentation of control states and allowing operators to perform control operations through Service Navigator's control console. The control system with WSMF management has been demonstrated internally in HP in Nov 2003. AFI is scheduled for the UDC 2.0 product release.

Introduction

IT infrastructure must become more adaptive, easier to deploy and to operate in order to support, and not hinder, reaction to change inside and across business organizations. Unifying resources and applications under the notion of services simplifies IT in the data center environment. Crossing technical and organizational boundaries in secure ways is encompassed under Grid's notion of a virtual organization, which may last only for the duration of a joint project or may last over years, and must be established and managed easily and yet securely.

Web services have been used in the enterprise IT domain for some time and are converging to a solid and mature base for creating, operating and managing IT services. The Grid with the Open Grid Services Infrastructure (OGSI) [1], and more recently the Web Services Resource Framework (WSRF) [2], as well as the Open Grid Services Architecture (OGSA) [3] encompass web services as underlying technologies. Standards in web services are increasingly important for managing IT services that are provided as web services.

This paper presents how standard Grid technology has been used in the Utility Data Center (UDC) [4] solution for a specific purpose: to create a control interface – called the Automatic Flexing Interface (AFI) [5] – that allows the creation of automatic control loops for server flexing for horizontally-scalable applications in the enterprise. Horizontally-scalable applications are simultaneously operated on a flexible number of servers of same type such as web servers, application servers or clustered databases. In contrast to similar control systems that operate in clusters [13-15], the control system here operates in the heterogeneous environment of a commercial data center. Joining and releasing servers from its virtualized application environments is inherently more complex and requires securely crossing protection domains.

AFI has been built based on the Open Grid Services Interface (OGSI), the standard prior to the Web Services Resource Framework (WSRF) that has been published in January 2004. AFI will transition to WSRF as soon as implementations of the WSRF standard become available and mature. The Web Services Management Framework (WSMF) [6] has been used to connect AFI's web services components (Interfaces Instances and the Interface Factory) to OpenView's Service Navigator for presentation in its operator console and for control. WSMF was submitted to OASIS. Standards from the OASIS Web Services Distributed Management (WSDM) TC will supercede WSMF [7].

The work is based on previous work on Grid and web services technology used in enterprise data centers [8-12].

Server Group Flexing in the Utility Data Center

As shown in Figure 1, a server group consists of a number of servers of same type, running the same version of the operating system and application, both originating from the same disk image. Life-cycle scripts are executed when the operating system is booting for the final configuration (Figure 6). Each server uses its own copy of a master image. The process of creating that copy is in the range of minutes limiting the periodicity of the control loop to 10's of minutes or hours, which is adequate for daily, weekly or monthly workloads patterns.

Reasons for flexing the server number up might be that application load is increasing and servers are approaching saturation levels. Reasons for releasing servers from server groups might be that servers are needed for other applications. Servers operated under a pay-per-use regime provide incentives to free resources when not needed.

Server Flex Control Loop

In a server flex control loop, flex operations are initiated by a control system that continuously oversees the conditions in the farm (based on load, use patterns, failure conditions, etc.) and evaluates conditions in order to derive a conclusion to increase or decrease the number of servers in a server group. Figure 2 shows a control loop for server flexing. The control loop consists of three main stages monitoring, assessment, and adjustment.

Monitoring. – Monitoring is typically based on elementary resource metrics such as CPU utilization, process queue lengths, used network bandwidth, memory utilization and swap rate. These measures are collected in the Utility Data Center by OpenView. If application instrumentation is provided, application-level metrics can be taken into account as well such as transaction rates, response times, numbers of simultaneous sessions, etc.

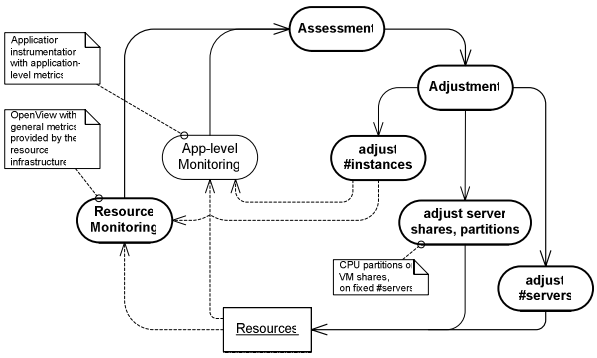

Figure 2: Server flex control loop.

Assessment. – Given measurements as input, the conditions in a server group can be assessed whether the system is considered operating within defined bounds or not. Automating assessment can be simple such as observing thresholds or it can be complex. The behavior of an assessment system is defined as customizable policy.

Adjustment. – As a result of assessment, a decision about a corrective action is derived, which leads to an adjustment in the system. Various means for adjustments exist for horizontally-scalable applications:

1. The number of instances of an application can be increased or decreased.
2. On certain hardware, CPUs or CPU partitions can be added or removed for an application in order to adjust processing capacity. Virtual machines allow adjusting CPU shares between virtual machines.
3. The number of physical machines can be adjusted (which remained constant in the first two cases).

Virtualizing resource infrastructures such as UDC have the capability to dynamically (during the operation of an application) add or remove physical servers from an application environment. It has the capability to configure servers into that environment, or release and unconfigure them from the environment. The outer-most control loop in Figure 2 encompasses the acquisition and release of entire physical servers from the resource infrastructure.

Control System for Automatic Server Group Flexing

The purpose of the control system for server group flexing is to provide horizontally scalable applications the ability to acquire or release physical server resources using the capabilities of the Utility Data Center to configure and unconfigure servers from the virtualized resource environment of a farm.

Design Choices

The following design choices are guided by requirements:

- Split of the control loop into a fixed infrastructure part embedded in the resource infrastructure providing server configuration, basic resource monitoring, and a customizable Control Plug-in containing the assessment and adjustment functions.

- Defining an open-standard's-based interface between the Control Plug-in and the resource infrastructure.
- Selecting OGSI as interface and interconnect technology mainly because of openness, web services standards, and built-in security and event models.
- Simple interaction patterns between the Control Plug-in and the underlying resource infrastructure based on service invocations and events.

The following figure shows the split of the control loop into Control Plug-in and resource infrastructure.

The OGSI interface between Control Plug-in and resource infrastructure primarily has two main functions:

- Request and delivery of basic resource monitoring data (provided by OpenView in the UDC) and delivery of state change events such as a change in the number of servers currently participating in a server group.
- Request to scale a server group up or down to a desired target number of servers.

Control policy is defined by the operator or the application administrator, in the current system in form of simple thresholds that are parameterizable for Control Plug-ins, which are implemented as Java OGSI clients.

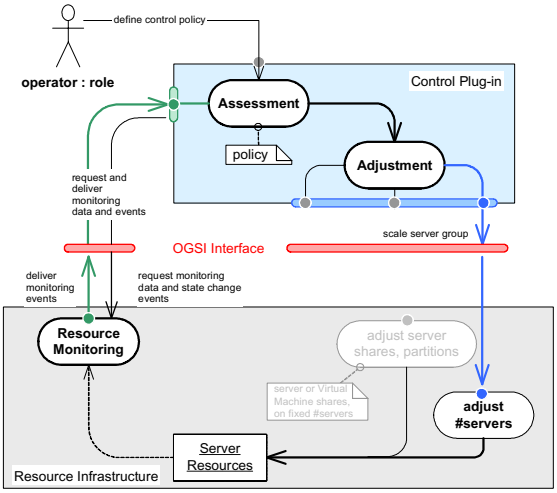

Figure 3: Component split of the control loop.

Design of OGSI Components

The split of the control loop into a part that contains assessment and adjustment(Control Plug-in) and the resource infrastructure delivering monitoring events and providing the capability to actuate server group adjustments leads to the following design of components that are implemented as OGSI services:

- Control Plug-in – implements assessment and making adjustment decisions, contains control policy, and is customizable by application administrators or data center operators.
- Interface Instances – counterpart for one Control Plug-in in the resource infrastructure. It provides port types for actuating flex decisions and delivery of monitoring and state change events.
- Interface Factory – provides a port type for creating Interface Instances. A Control Plug-in initially contacts the Interface Factory in order to create an Interface Instance, which it then uses for all subsequent interactions.

OGSI's event and life cycle mechanisms are used for OGSI components. Each has a number of port types:

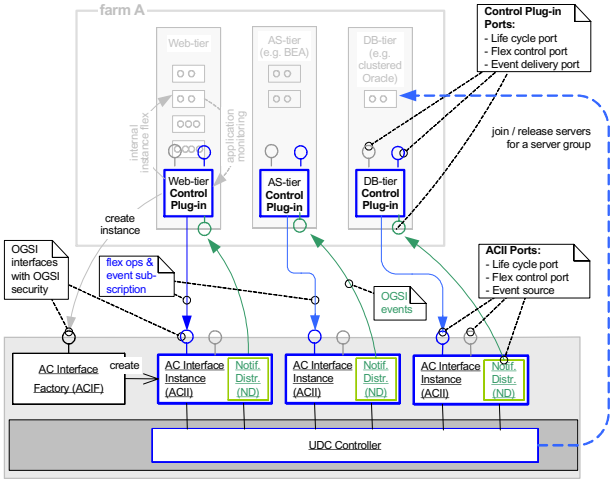

Figure 4: OGSI component design and basic flows.

Control Plug-in:

- Event Delivery Port – primary port for event delivery from the associated Interface Instance or other event sources such as application-level instrumentation.
- Flex Control Port – allows a higher-ordered control systems to connect to and instruct a Control Plug-in.
- Life Cycle Port – for managing the life cycle of the control loop (start, stop, suspend, resume, etc.).

Interface Instance:

- Flex Control Port – primary port for receiving server flex operations from the associated Control Plug-in.
- Life Cycle Port – port for life cycle operations on Interface Instances.

- Event Distribution Port – port where Control Plug-ins subscribe for monitoring and state change events.

Interface Factory:

- Factory Port – initial contact point for a Control Plug-in that allows the creation of one Interface Instance after its authenticity and authorization has been validated.

Securely Crossing Protection Domains

Since the UDC Controller is performing critical resource management functions in the Utility Data Center, it is located in a protected domain that cannot be reached from applications outside. Since Control Plug-ins are located outside the UDC Controller in order to meet the customization and integration requirements, the interaction between Control Plug-ins and Interface Instances must cross the protection domain boundary in a secure way (in analogy to system calls in operating systems). OGSI offers a built-in security model that is used for securely crossing protection domains between the Control Plug-in residing outside and the Interface Instance residing inside the protected UDC Controller domain. OGSI Security is applied at two stages:

1. Control Plug-ins are authenticated. A Control Plug-in's certificate is validated by the Interface Factory at initial contact. Only Control Plug-ins with registered certificates can create Interface Instances and are able to subsequently interact with the UDC Controller.
2. Establishing an encrypted communication channel between the Control Plug-in and the Interface Instance ensuring for both sides that the exchanged information is authentic and unaltered.

Server Flex Control Cycle

Unassigned servers are maintained in UDC's resource pool. When the Control Plug-in initiates scaling a server group up or down, selected server resources transition through the stages shown in Figure 5.

Flex up. – A server (or a number of servers) matching the type of the server group is selected from the resource pool and set into the join state. During that stage, the farm resource environment is reconfigured to include the new server(s). This means that the VLAN is reconfigured, virtual IP addresses are assigned to the server, DNS is reconfigured, and the logical disks from the storage array with copies of the server group images are created and mapped onto the joining servers. After that, the joining servers reboot and launch applications from those disks. The join state is finished when new server(s) have booted and applications are launched. Launching applications is initiated by start-up scripts configured on images.

The following init state allows executing further life cycle operations on joined servers (shown as ALCD – Application Life Cycle Daemon in the interaction diagram in Figure 6). This allows reconfiguration in the software environment, for instance, to start directing workload to joined servers. At the end of the init state, the new servers are fully operational as part of the server group.

Flex down. – The flex down cycle is also initiated by the Control Plug-in by selecting a number of servers to be released. In reverse order to flex-up, life cycle scripts on servers are executed first during withdraw, unconfiguring applications of affected servers from their environment. A load balancer may need to be reconfigured to not send further workload to affected servers, active sessions have to be finished, etc. At the end of the withdraw state, the local application instances are no longer part of the application and can finally be terminated.

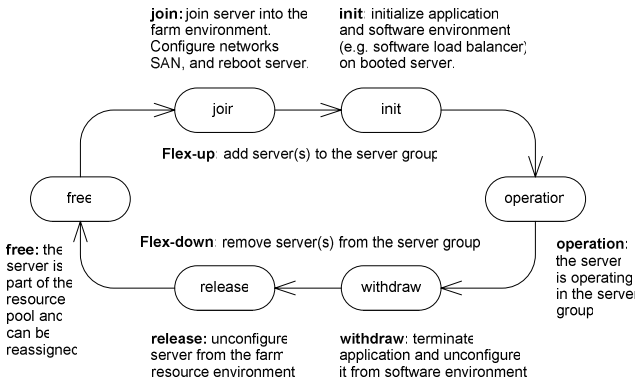

Figure 5: State diagram for server flex operations.

During the release state, the servers to be released are unconfigured from the farm's resource environment including the VLAN. Disks (logical volumes) are detached by reprogramming the SAN and cleared. No state of application instances is kept for server groups.

At the end of the release state, all state associated with affected servers has been removed from the software and hardware environment including the servers themselves. Servers now transition to the free state and return to the server pool for new assignment.

Server Flex Protocol

The server flex protocol encompasses two parts. Monitoring and state change events must be delivered to the Control Plug-in and eventually other subscribers. And flex instructions (scale server group requests) must be delivered in opposite direction.

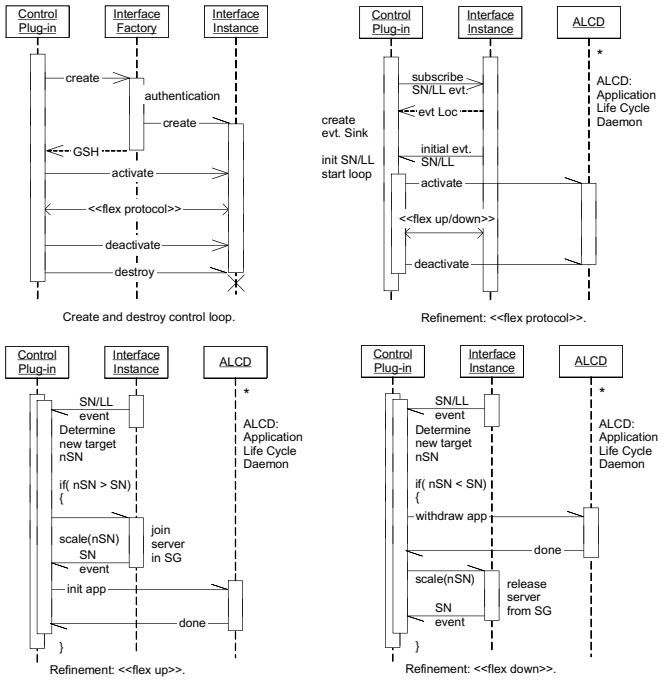

Figure 6: Server flex interaction diagrams.

Event delivery. – Event delivery is based on OGSI events, which are based on changes in Service Data Elements (SDE) [1]. SDEs are maintained in Interface Instances. Two SDEs have been defined:

SDE SN – the current number of servers (SN – Server Number) in a server group. This SDE triggers events informing the associated Control Plug-in and other subscribers when either the farm administrator has reconfigured the server group through a console, or the server number has changed in effect of a previous flex operation. If the SN reported in the SDE equals to the target SN previously issued as flex request, the Control Plug-in concludes that the previous operation has succeeded. SN change events are trigged at state changes init operation and release free (Figure 5).

SDE LL – represents the current Load Level (LL) in a server group. This is an aggregate number taking various OpenView base measurements into account: CPU load, memory usage, IO and SWAP activity. Aggregation is based on a model that for any server i in a server group LLi = max(CPU, MEM, IO, SWAP), with each of the base measurements normalized to a scale of 0-100. The load level LL of the server group then is then LL= Σ LLi. This simple model assumes that load between servers in the server group is evenly distributed. The rate of LL SDE change events is controlled by OGSI's event flow rate control mechanism with min_interval=30sec and max_interval=300sec.

Based on SN and LL change events, which are also time stamped at the source, the Control Plug-in can obtain a load and utilization profile over time and assess changes in the server group that may lead to flex decisions. Policy parameters in the Control Plug-in must be tuned to avoid thrashing effects.

Flex request. – Once the Control Plug-in has come to the conclusion to flex a server group, it issues a scale server request and waits for its completion indicated by a SN change event. Since execution of flex operations is in the range of 10+x minutes, flex requests are asynchronous. The Control Plug-in observes the time of a flex operation and can take action when this time passes without receiving a server number change event. Flex requests are issued with the total target number of servers rather then incremental changes in order to make flex requests idempotent. Failure or incompletion within the expected time frame simply may cause the Control Plug-in to repeat the request leading to the same result.

Figure 6 shows the server flex protocol including event delivery and server scale requests.

WSMF – The Web Services Management Framework

Web Services Management has two different aspects. The first is Management Using Web Services (MUWS), allowing the management of any kind of resource via Web Services. The second is Management of Web Services (MOWS). Both MUWS and MOWS are addressed by the Web Services Management Framework [6].

The Web Services Management Framework (WSMF) is a logical architecture for managing computing resources, including Web services themselves, through Web services. This framework is based on the concept of managed objects and their relationships. In this view, a managed object represents a managed resource and exposes a set of management interfaces. Through the management interfaces, the underlying resource can be managed and controlled. To better support important IT domains such as EAI, ERP, or SCM, WSMF is model-neutral and is

designed to be applied to many different domains with varying management requirements. It also is platform-neutral, allowing for consistent management of J2EE, .NET, and other platforms. WSMF defines the functions, interactions, and formats through which web services can become managed in a unified manner. WSMF covers various areas of management such as service discovery, relationships, performance, and control.

Standards for management (SNMP, CIM, etc) have been established. But with all the standards and frameworks in place, management is still a 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 or were targeted at technologies when they were in their early stages. The existing management standards (such as SNMP, CIM) are of too low level to allow flexible, coordinated interaction patterns. SNMP and CIM primarily focus on data collection, not on writing rich management applications for the adaptive infrastructure. For the same reason that other distributed applications use Web services for language and platform independence, interoperability through wire level standards, industry momentum and the ability to expose interface and hide implementation, management systems will adopt web services technology.

WSMF-Foundations and WSMF-WSM

The WSMF-WSM (Web Services Management) defines the basic management capabilities an underlying managed web service must provide. WSMF-WSM extends WSMF-Foundation, which defines manageability of any resource (including non-web services) through WSMF. Managed resources that are not web services are accompanied by a management service that implements the WSMF Foundation port types and is linked with the managed resources. The WSMF-Foundation interface basically only provides discoverability and relationships of the resource associated with the management service.

Figure 7 shows the two kinds of implementations. The left-hand side shows a non-web service being associated with a management service (web service) that implements the WSMF-Foundation interface. The right-hand side shows a managed web service that implements the WSMF-WSM interface for managed web services that in turn implements the WSMF-Foundation interface.

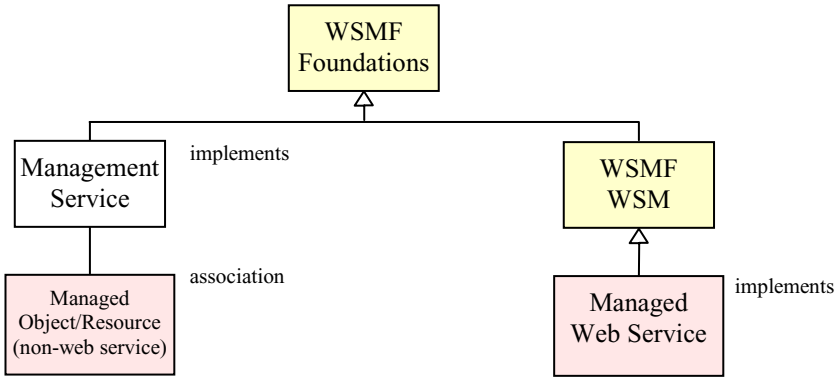

Figure 7: Managing web services and non-web services through WSMF.

The WSMF2.0 definition is structured in sections of:

- events,
- foundation,
- wsm/service,
- wsm/wsexecutionenvironment,
- wsm/conversation.

WSMF-Foundation operations are (classified by port types):

ManagedObjectIdentityInterface	
	- GetManagedObjectReference
	- GetManagementWsdlUrl
ManagedObjectConfigurationInterface	
	- GetName
	- GetType
	- GetDescription
	- GetOwner
	- GetVendor
	- GetVersion
	- GetManagedObjectVersion
	- GetCreatedOn
	- GetHostName
	- GetManagedObjectHostName

ManagedObjectMonitoringInterface	
	- GetState
	- GetSupportedStates
ManagedObjectDiscoveryInterface	
	- GetRelationships
	- GetSupportedRelations
ManagedObjectControlInterface	
	- SetState
	- SetName
ManagedObjectCollectionInterface	
	- Invoke
	- getMembers

Table 1: WSMF-Foundation operations (classified by port types).

Additional interfaces of managed web services are defined by WSMF-WSM (Web Services Management). The WSMF-WSM interface inherits interfaces from WSMF-Foundations.

WSMF-WSM operations are (classified by port types):

ServiceConfigurationInterface	
	- GetOperationalWsdlUrl
ServiceMonitoringInterface	
	- GetConversations
	- GetConversationsById
	- GetLastMessageReceived
	- GetLastMessageSent
	- GetLastFaultMessageReceived
	- GetLastFaultMessageSent
ServiceDiscoveryInterface	
	- GetContainer
ServiceControllInterface	
	- Start
	- Stop
ServicePerformanceInterface	
	- GetTotalMessagesReceivedCountInput
	- GetTotalMessagesSentCount
	- GetSuccessMessagesReceivedCount
	- GetSuccessMessagesSentCount
	- GetFaultMessagesReceivedCount
	- GetFaultMessagesSentCount
	- GetTotalMaximumResponseTime
	- GetTotalMinimumResponseTime

	- GetTotalCumulativeResponseTime
	- GetSuccessMaximumResponseTime
	- GetSuccessMinimumResponseTime
	- GetSuccessCumulativeResponseTime
	- GetFaultMaximumResponseTime
	- GetFaultMinimumResponseTime
	- GetFaultCumulativeResponseTime
ServiceOperationPerformanceInterface	
	- GetTotalMessagesReceivedCountForOperation
	- GetTotalMessagesSentCountForOperation
	- GetSuccessMessagesReceivedCountForOperation
	- GetSuccessMessagesSentCountForOperation
	- GetFaultMessagesReceivedCountForOperation
	- GetFaultMessagesSentCountForOperation
	- GetTotalMaximumResponseTimeForOperation
	- GetTotalMinimumResponseTimeForOperation
	- GetTotalCumulativeResponseTimeForOperation
	- GetSuccessMaximumResponseTimeForOperation
	- GetSuccessMinimumResponseTimeForOperation
	- GetSuccessCumulativeResponseTimeForOperation
	- GetFaultMaximumResponseTimeForOperation
	- GetFaultMinimumResponseTimeForOperation
	- GetFaultCumulativeResponseTimeForOperation

Table 2: WSMF-WSM operations (classified by port types).

The Future of WSMF

WSMF was submitted to the OASIS WSDM Technical Committee (WSDM TC). The WSDM TC is chartered with addressing both MUWS and MOWS. The standards created by the WSDM TC will supercede WSMF.

As of today, drafts of the initial versions (version 0.5) of the WSDM MUWS and MOWS specifications have been released. These specifications are built on top of a subset of the Web Services Resource Framework (WSRF)

suite of specifications [2]. The WSRF standards used are WS-ResourceProperties and WS-Addressing.

The current WSDM specifications provide significantly less functionality than WSMF. Neither managed resource discovery nor events are supported, which as described below, are being used in the WSMF management interface for AFI. Those functions are assumed from the underlying infrastructure such as WSRF. The WSDM TC plans to address those areas in version 1.0 of the specifications. Additional WSRF specifications will very likely form the basis.

Scope of WSMF Management for the Automatic Control System

WSMF manageability has been included in the Automatic Flexing Interface in the following way:

- AC Interface Factory (ACIF) and AC Interface Instances (ACII) will, as OGSI services, provide interfaces to become discoverable through WSMF, including service relationships.
- WSMF service discovery will allow a WSMF proxy process to obtain the information about currently existing service instances (ACIF, ACII's) representing active control loops.
- This information will be used to provide input to a console tool, OV Service Navigator, which displays status information of control loops represented by those services. ACIF and ACII will be shown in the Service Navigator console as they enter or leave the AC system.

Service discovery and relationships are two fundamental aspects of a WSMF-managed system.

WSMF Service Model for AC Case

WSMF assumes a service model for a service environment to be managed. The simple service model that applies to the AC case consists of the ACIF factory service and ACII instances at two stages: farm level and server group level. There are a WSMF managed objects (MO's) for the ACIF factory service, the ACII farm level instances and the ACII server group level instances. WSMF relationships allow the discovery of ACII farm level instance MO's starting from the ACIF MO's and discovery of ACII server group level MO's from ACII farm level MO's. These MO's form a hierarchy as shown in Figure 8.

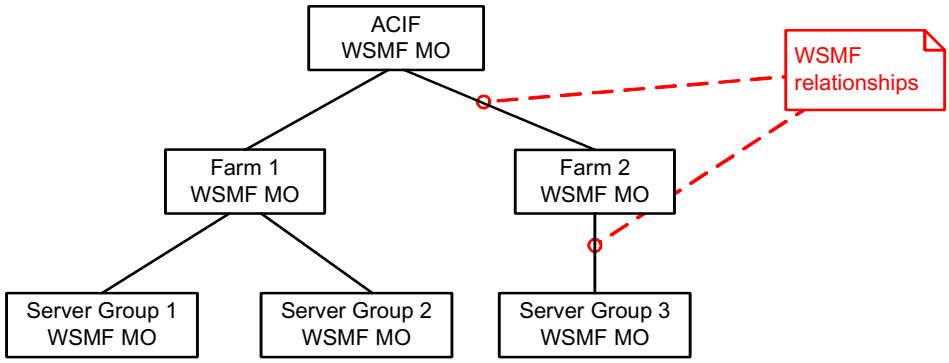

Figure 8: WSMF Managed Object hierarchy

The management application (OV Service Navigator in this case) will be provided with the ACIF MO as the starting point. It can then automatically discover all other MO's using the WSMF relationships between them.

OGSI / WSMF Integration

The interfaces defined by WSMF also define the message ("wire") formats. OGSI also defines a message format and its own event model. Both are not compatible. In order to make them interoperate, a so-called WSMF proxy was introduced which translated between the two formats.

The WSMF proxy uses the OGSI interface provided by the ACIF and ACII's to communicate with the AFI infrastructure. The ACIF location is known to the WSMF proxy. Starting from here, the proxy discovers the ACII instances using the OGSI mechanism to enumerate the AC Interface Instances (via a Service Data Element).

It now builds a corresponding hierarchy of WSMF managed objects as shown in Figure 8. These WSMF MO's act as proxy objects for the ACIF and ACII's, effectively performing a protocol translation from OGSI to WSMF.

The translation primarily delivered OGSI events from AFI to OV Service Navigator in order to update service relationships and also Service Navigator's console screen. Control instructions such as start and suspend a control loop could be entered by an operator in the console and needed to be translated from resulting WSMF control operations into OGSI invocations performed on AFI service instances.

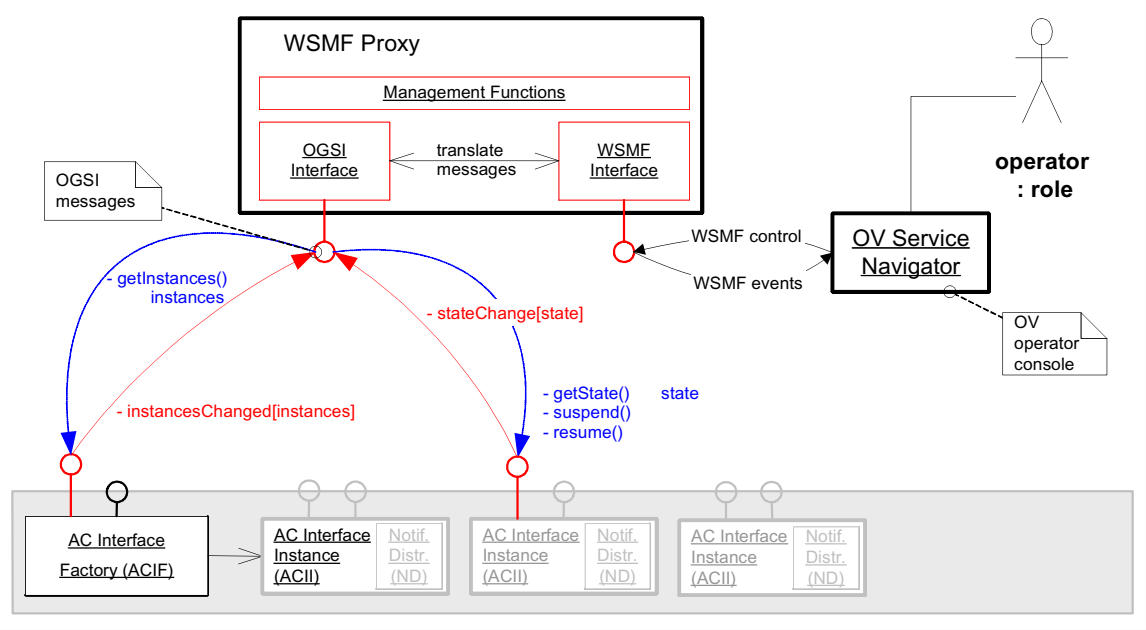

Figure 9: OGSI / WSMF Integration for the Automatic Flexing Interface.

The proxy utilizes the OGSI event mechanism in order to be notified of any changes in the AFI state. Whenever an AC Interface Instance is created or destroyed, the AC Interface Factory will generate an OGSI event. When an AC Interface Instance is created, a new WSMF proxy managed object for that instance is created. A WSMF relationship change event will be triggered. This event contains the new relationship between ACIF and/or ACII(s). When an AC Interface Instance is destroyed, the corresponding WSMF managed object is destroyed. A WSMF relationship change event will be triggered. This event contains the deleted relationship.

A generic WSMF client that existed for OpenView Service Navigator allowed displaying the WSMF managed object hierarchy provided by the WSMF proxy as shown in Figure 10 (left figure). This view is automatically discovered starting from the root managed object for the ACIF by following the WSMF relationships. It is automatically updated as triggered by WSMF events for WSMF relationship changes.

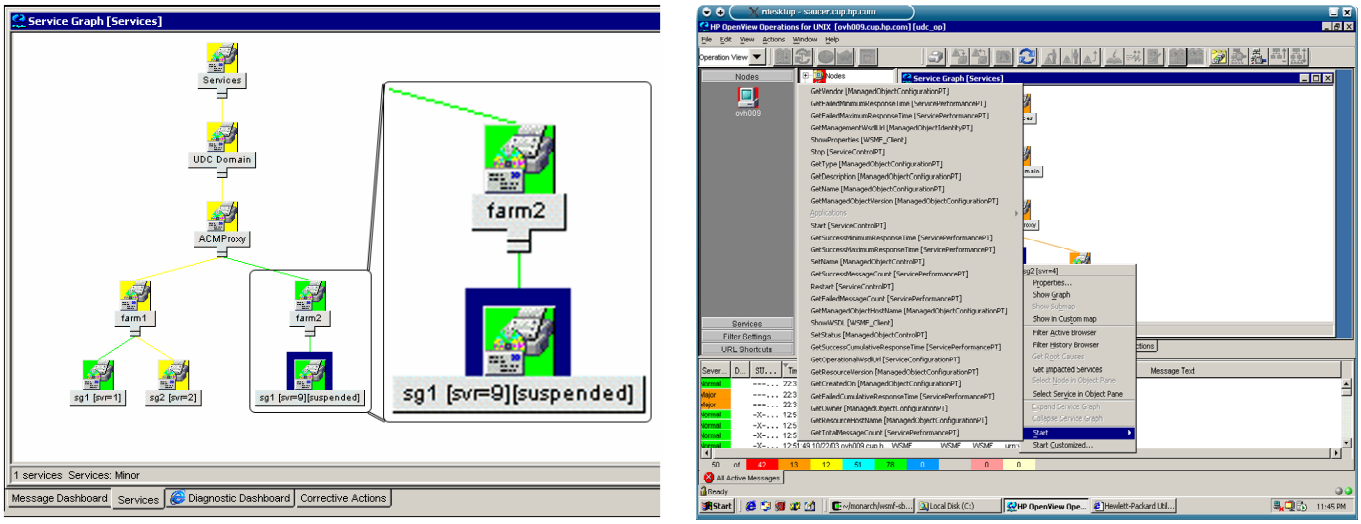

Figure 10: OV Service Navigator screens showing AC Interface Instances (left) and control operations (right).

The WSMF proxy defines a number of custom WSMF events. These events include information about the current load level of a server group, the number of servers in a server group and the operational state of the control plug-in connected to an AC Interface Instance. The WSMF client for OV Service Navigator can subscribe to these events and automatically convert them into OpenView events for OV Service Navigator.

OV Operations templates [16] have been defined that utilize these OpenView events. The current load level of a server group is shown as the color of the corresponding icon in the Service Navigator user interface (Figure 10, left figure). There are three different load levels: low (green), medium (yellow) and high (orange) that. The current number of servers in a server group and the current operational state of the control plug-in connected to an ACII are displayed in brackets after the server group name, also shown in Figure 10.

The WSMF proxy MO's expose a number of command actions. Those actions are automatically discovered by OV Service Navigator and can be triggered in the Service Navigator console by an operator. They are propagated back as WSMF invocations to the WSMF proxy, which translates them into OSGI invocations for associated AFI/ OGSI services. It is possible to start and suspend the operation of an AFI control plug-in. Once the control plug-in has performed its operational state change, an OGSI event is triggered that in turn triggers the corresponding WSMF proxy MO to send a WSMF event, which will trigger an automatic update in Service Navigator view.

Further Steps for Web Service Manageability in the Automatic Flexing Interface:

- Upgrade OGSI to the new Web Services Resource Framework (WSRF) standard released in January 2004.
- Incorporate the new WSDM standards for management when it has become mature and implementations are available.
- Manage life-cycle of control loops through OV Service Navigator.

Summary

This paper presented how UDC's Automatic Flexing Interface was built based on web services standards and a management link was established to OV Service Navigator using WSMF, which will be superceded by the OASIS WSDM web services management standards.

The Automatic Flexing Interface has been built using Globus Toolkit 3 (based on OGSI) and is part of the UDC 2.0 release. AFI will be upgraded to Globus Toolkit 4 (based on WSRF), which is expected by the end of 2004.

References

[1] Global Grid Forum: Open Grid Services Infrastructure v1.0, April 2003.

[2] The Globus Alliance, The WS-Resource Framework WSRF, http://www.globus.org/wsrf.

[3] The Globus Alliance, Towards Open Grid Services Architecture (OGSA), http://www.globus.org/ogsa.

[4] Hewlett-Packard: Utility Data Center, http://www.hp.com/solutions1/infrastructure/solutions/utilitydata, 2003.

[5] Hewlett-Packard: HP Adaptive Control Specification v1.1, October 3, 2003.

[6] Hewlett-Packard: Web Services Management Framework (WSMF) v2.0, Specification, 2003.

[7] OASIS, Web Services Distributed Management (WSDM), http://www.oasis-open.org.

[8] Hewlett-Packard: Enterprise Grid, http://www.hp.com/techservers/grid/index.html, 2003.

[9] Sahai, A., Graupner, S., Machiraju, V., van Moorsel, A.: Specifying and Monitoring Guarantees in Commercial Grids through SLA, Proc. of CCGrid 2003, May 2003.

[10] Graupner, S., Pruyne, J., Singhal, S.: Making the Utility Data Center A Power Station for the Enterprise Grid, HP Labs Technical Report HPL-2003-53, http://www.hpl.hp.com/techreports/2003, March 2003.

[11] Andrzejak, A., Kotov, V., Graupner, S., Trinks, H.: Service-Centric Organization of Globally Distributed Computing, IEEE Internet Computing Journal, Special Issue on Grid Computing, July/August 2003.

[12] Graupner, S., Machiraju, V., Sahai, A., van Moorsel, A.: Management += Grid, DSOM'2003, October 2003.

[13] IBM: xCAT, Extreme Cluster Administration Toolkit, http://www.alphaworks.ibm.com/tech/xCAT, 2003.

[14] Ahmed, K., Priddy, K., Mashayekhi, V., Fang, Y.-C.: The Cluster as Server, Platform Computing, Inc., 2003.

[15] Sun Microsystems: Sun Grid Engine, http://wwws.sun.com/software/gridware, 2003.

[16] Hewlett-Packard: OpenView Operations Manuals, http://ovweb.external.hp.com/lpe/doc_serv, 2003.