Cloud Service Assurance for VMDC Design Guide

May 2, 2013

Cisco Validated Design

CCDE, CCENT, CCSI, Cisco Eos, Cisco Explorer, Cisco HealthPresence, Cisco IronPort, the Cisco logo, Cisco Nurse Connect, Cisco Pulse, Cisco SensorBase, Cisco StackPower, Cisco StadiumVision, Cisco TelePresence, Cisco TrustSec, Cisco Unified Computing System, Cisco WebEx, DCE, Flip Channels, Flip for Good, Flip Mino, Flipshare (Design), Flip Ultra, Flip Video, Flip Video (Design), Instant Broadband, and Welcome to the Human Network are trademarks; Changing the Way We Work, Live, Play, and Learn, Cisco Capital, Cisco Capital (Design), Cisco:Financed (Stylized), Cisco Store, Flip Gift Card, and One Million Acts of Green are service marks; and Access Registrar, Aironet, AllTouch, AsyncOS, Bringing the Meeting To You, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, CCSP, CCVP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Lumin, Cisco Nexus, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Collaboration Without Limitation, Continuum, EtherFast, EtherSwitch, Event Center, Explorer, Follow Me Browsing, GainMaker, iLYNX, IOS, iPhone, IronPort, the IronPort logo, Laser Link, LightStream, Linksys, MeetingPlace, MeetingPlace Chime Sound, MGX, Networkers, Networking Academy, PCNow, PIX, PowerKEY, PowerPanels, PowerTV, PowerTV (Design), PowerVu, Prisma, ProConnect, ROSA, SenderBase, SMARTnet, Spectrum Expert, StackWise, WebEx, and the WebEx logo are registered trademarks of Cisco and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or website are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1002R)

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED "AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Cloud Service Assurance for VMDC Design Guide
© 2013 Cisco Systems, Inc. All rights reserved.

CONTENTS

Preface

This preface explains the objectives and intended audience of the Cloud Service Assurance for Virtualized Multiservice Data Center (CLSA VMDC) Design Guide (DG) and abridged version of the partner specific Design and Implementation Guide (DIG).

Portions of this document are Copyright © 2006-2013 Zenoss, Inc., all rights reserved. Zenoss and the Zenoss logo are trademarks or registered trademarks of Zenoss, Inc. in the United States and other countries. All other trademarks, logos, and service marks are the property of Zenoss, Inc. or other third parties. Use of these marks is prohibited without the express written consent of Zenoss, Inc. or the third-party owner.

Product screen shots and other similar material in this document are used for illustrative purposes only and show trademarks of EMC Corporation (VMAX), NetApp, Inc. (NetApp FAS6080, FAS3k, FlexPod), VCE (Vblock), and VMware, Inc. (ESXi Hypervisors, Virtual Machines). All other marks and names mentioned herein may be trademarks of their respective companies.

Audience

This guide is intended for, but not limited to, system architects, network/compute/storage design engineers, systems engineers, field consultants, advanced services specialists, and customers who want to understand how to provide service assurance capabilities for VMDC.

This guide assumes that the reader is familiar with the basic concepts, aware of general system requirements, and has knowledge of Enterprise or Service Provider (SP) network and Data Center (DC) architectures and platforms and virtualization technologies.

Document Organization

Table i-1 provides the organization of this document.

Table i-1 Document Organization

Topic	Description
Chapter 1 Introduction	This chapter provides an introduction to CLSA VMDC.
Chapter 2 VMDC System Overview	This chapter provides a brief review of the VMDC infrastructure system and its components.
Chapter 3 CLSA VMDC System Architecture	This chapter provides an overview of the CLSA VMDC system architecture.
Chapter 4 Zenoss Cloud Service Assurance Overview	This chapter discusses the Zenoss Cloud Service Assurance (CSA) architecture and provides an overview of the capabilities of the core assurance platform.

CHAPTER 1

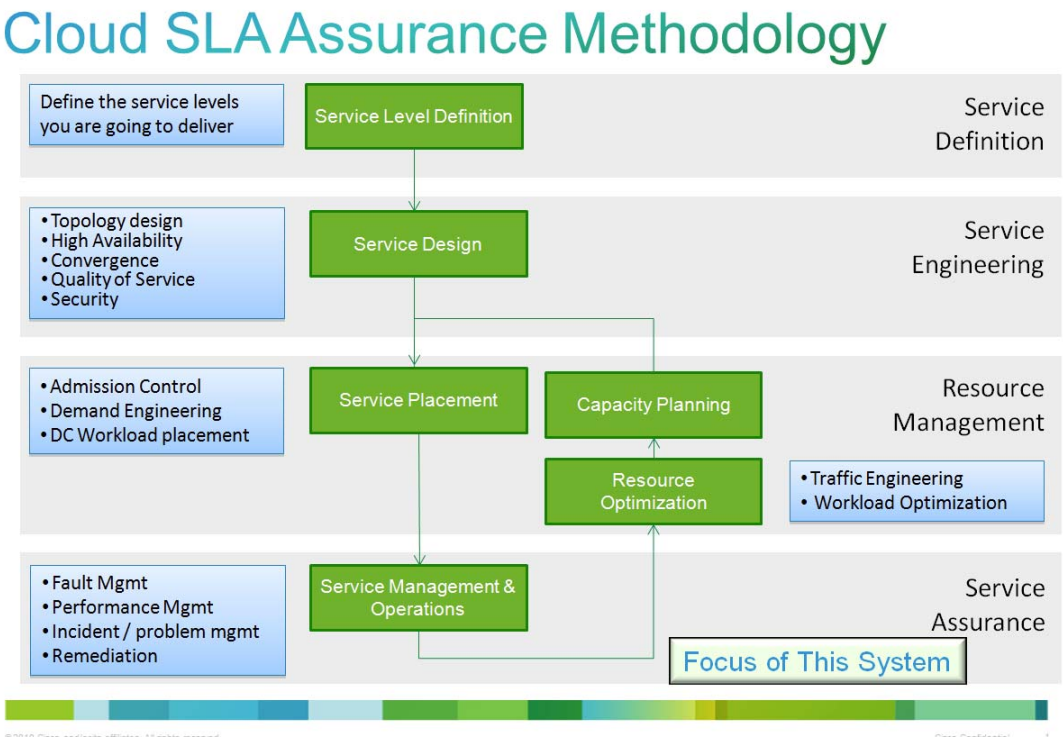
Introduction

In recent years, there has been a race by both traditional Service Providers (SPs) and public cloud providers such as Amazon to capture the cloud services market. SPs have identified the capability to offer Service Level Agreements (SLAs) as their key differentiator in the race for the cloud. In response, SPs are deploying virtual private cloud services accessed by Enterprises (cloud consumers) over the SP's IP/MPLS VPN network infrastructure. In addition, lack of trust had been identified as one of the key barriers for Enterprises to purchase cloud services. To gain end customer trust of cloud services, it is important that a cloud provider offer customers visibility in the performance of their applications hosted in the cloud.

SPs have to take measures both in engineering the service and in operating the service to offer their customers the SLAs necessary to realize the potential of virtual private cloud differentiation. The term "service assurance" is commonly used to refer to performance management and fault management, i.e., monitoring and reporting that the service levels are met and identifying/resolving service impacting faults. More generally, assurance means providing a high level of confidence that a commitment can be met; this encompasses more than just operation and management aspects, but also includes service engineering aspects.

The broader SLA assurance framework with all necessary functions to offer SLAs is illustrated in Figure 1-1. This framework includes service assurance as one of its building blocks, which is the focus of this system and this document. In addition to the virtual private cloud opportunity, service assurance also plays a role in Enterprise private clouds to enable efficient Day 2 operations and gain visibility necessary to optimize resources utilization.

Figure 1-1 Cloud SLA Assurance Methodology



Both Infrastructure as a Service (IaaS) and Software as a Service (SaaS) private and virtual private cloud services can be offered on top of the Virtualized Multiservice Data Center (VMDC) architecture. The Cloud Service Assurance for VMDC (CLSA VMDC) system provides service assurance capabilities for VMDC, as well as private and virtual private cloud IaaS. This system can also be leveraged as a building block of application-based cloud services such as Cisco Hosted Collaboration Solution (HCS), Cisco Virtualization Experience Infrastructure (VXI), and SP TelePresence.

This chapter presents the following topics:

System Purpose

This document describes design guidelines for Cloud Service Assurance for VMDC (CLSA VMDC). This version of the system supports VMDC 3.0, VMDC 2.2, VMDC 2.3, and earlier infrastructure architectures. CLSA VMDC is based on Zenoss Cloud Service Assurance (CSA), which was built from the ground up for cloud technology management. Zenoss CSA is a service impact model-based system that allows for rapid new service introduction, tenant-based service assurance, consolidated monitoring of the VMDC infrastructure, and simple customizations that can be deployed without service down time via plugins called ZenPacks.

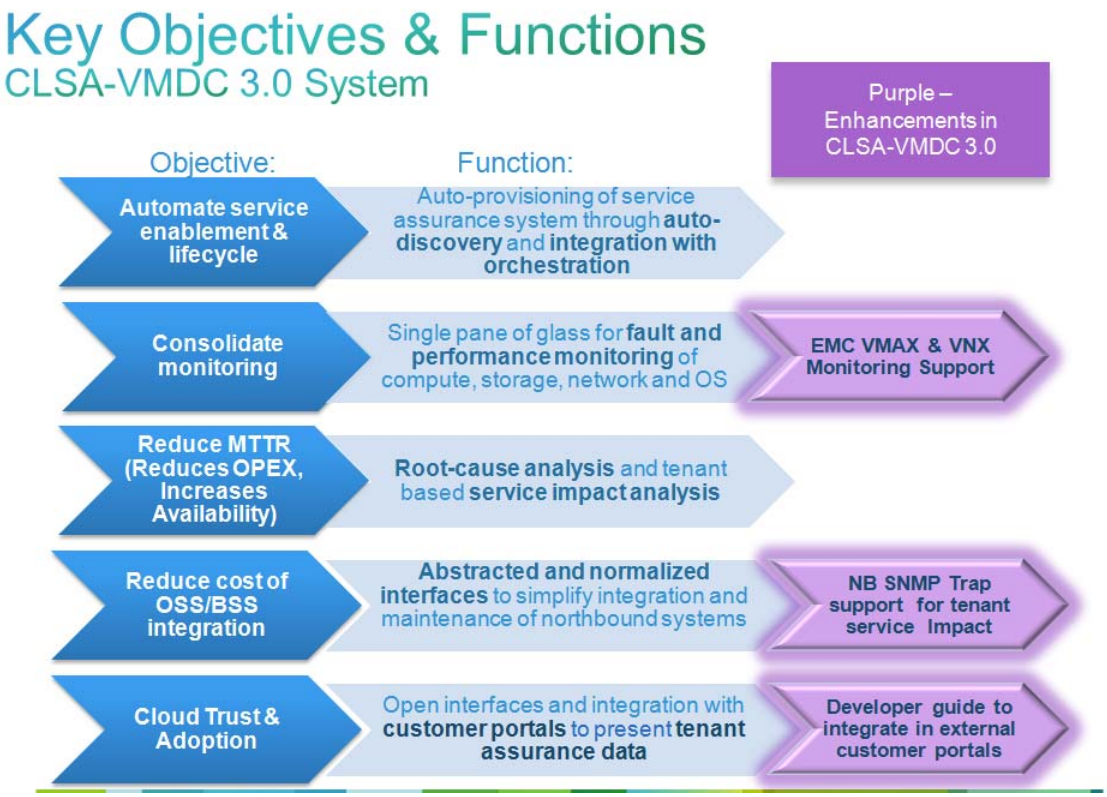
Note While the CLSA VMDC Design and Implementation Guide (DIG) references specific VMDC systems, previous versions of the VMDC system are also supported. The CLSA VMDC system also supports other Data Center (DC) designs, as well as the VCE Vblock and NetApp FlexPod stacks.

Zenoss CSA is a multiservice system that offers real time aggregated dashboards as well as reporting capabilities. The system can be deployed both in centralized and distributed architecture and allows for incremental deployment growth. While it offers rich functionality for IaaS domains, the solution is lightweight and has open interfaces to allow for simple integration into existing Operations Support System (OSS) and ticketing systems with minimal cost. As such, this solution is positioned not as a replacement, but as a complement to existing Manager-of-Manager (MOM) systems (e.g., IBM Netcool), ticketing systems (e.g., BMC Remedy), and so on.

System Objectives

The key business objectives of the CLSA VMDC system and the respective technical functions that realize these benefits are illustrated in Figure 1-2 and discussed throughout this document.

Figure 1-2 Key Objectives and Functions of CLSA VMDC

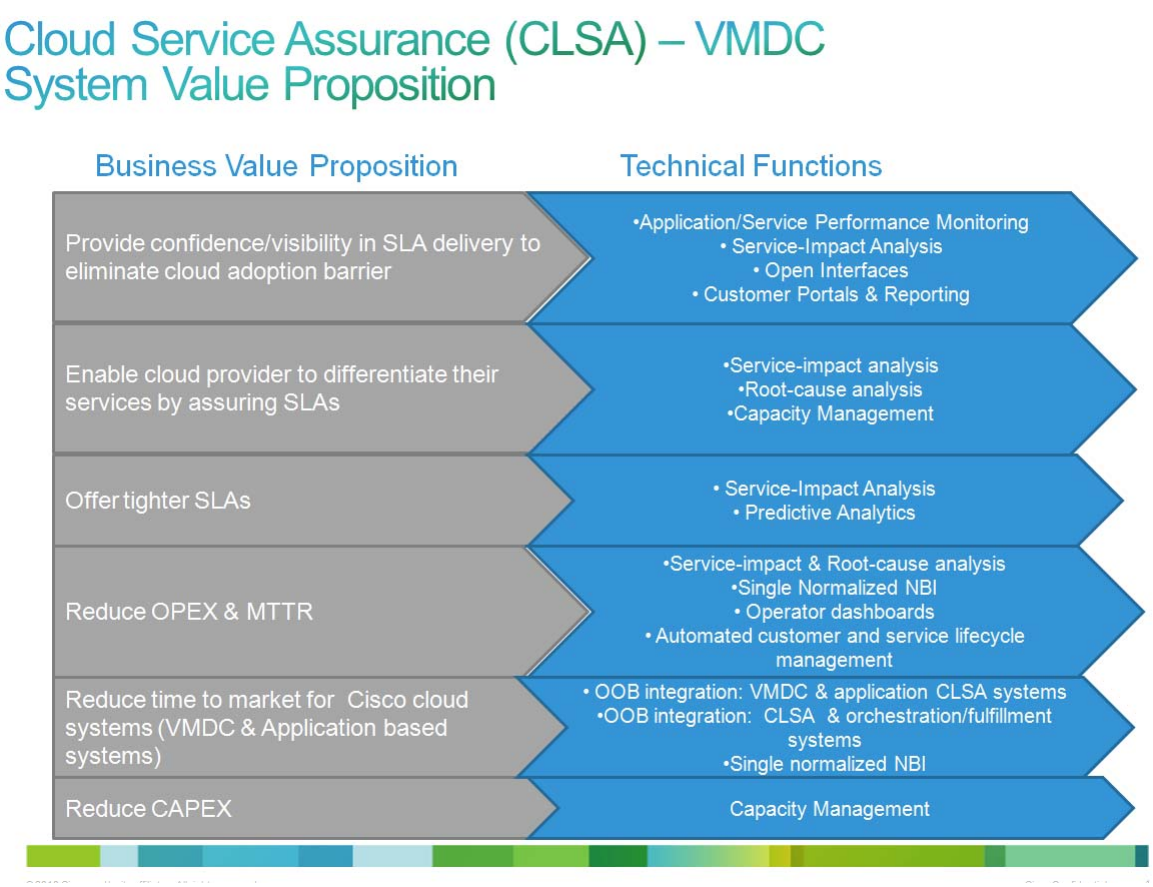


Key Benefits of Cloud Service Assurance, page 1-4 provides more in-depth discussion on the benefits of cloud service assurance.

Key Benefits of Cloud Service Assurance

Figure 1-3 outlines the key business value propositions of cloud service assurance and the technical functions that help realize these value propositions.

Figure 1-3 Key Benefits of Cloud Service Assurance



Cloud service assurance focuses on solving the following four key customer problem statements:

- Automating Service Enablement, page 1-4
- Consolidated Monitoring, page 1-5
- Reducing Mean Time to Repair (MTTR), page 1-6
- Northbound OSS/BSS integration, page 1-7

Automating Service Enablement

As previously noted, assurance services are a key component of the overall cloud service offering. In order to enable and manage the lifecycle of assurance services, a significant amount of manual configuration may be required. In cloud environments that call for self-service and large scale, automatic

enablement of service assurance is required. Automatic enablement of service assurance can be achieved in a couple of different ways. Fundamentally, the following approaches can be taken to automate service enablement and life cycle:

1. Reduce necessary amount of configuration (by using technology that is self learning (e.g., self learning thresholds).
2. Automatic discovery (by assurance system).
3. Programmatic orchestrated provisioning (via integration with orchestration system).

CLSA VMDC utilizes all of the above methods to automate service enablement with specific emphasis on automatic discovery.

The following types of objects are automatically discovered in CLSA VMDC:

- Monitored devices (e.g., UCS, Nexus 7000, MDS 9000, etc.).
- Sub-components of devices and their relationships (e.g., UCS chassis, blades, fabric interconnect, etc.).
- Tenant-based Service Impact Analysis (SIA) models for the compute (e.g., tenant Virtual Machine (VM) mapping to service impacting dedicated and shared vCenter and UCSM managed resources).

Consolidated Monitoring

Due to the large number of components and technologies in many of the SP and IT systems, operations staff are typically segmented and specialized, and they utilize a number of customized tools. This operations staff division of labor results in a monitoring approach that involves observing multiple screens and interaction between a number of organizations when trying to solve even the simplest problems. For example, there are storage operations that are responsible for storage only using their favorite tool, and similarly, there are compute operations with their staff and tools, network operations, and applications operations, and so on. This approach not only increases Mean Time to Repair (MTTR), and thus customer dissatisfaction, but it will also be unmanageable for cloud systems that are extremely dynamic and deployed at extreme scale. While there will always be a need to have specialized staff with focused expertise, there must be some consolidation of monitoring products to provide a single pane of glass that will simplify Tier 1 and 2 operations.

In addition, to fully automate some of operations tasks through value add assurance functions such as Root Cause Analysis (RCA) and SIA, assurance products need to have visibility of all of the components that work together to deliver the service. While segmented visibility will always exist and present challenges in the cloud environment due to business and ownership boundaries, the effort needs to be made to provide as much visibility as possible. More visibility means more value add from the assurance system.

To solve visibility challenges, consolidated monitoring and data collection is one of the fundamental functions of any cloud service assurance system. Consolidated monitoring and data collection needs to be done in the following ways:

- Various domains (applications, compute, storage, network). The cloud assurance system needs to provide a single pane of glass to monitor components from various domains.
- Fault and performance data. The cloud assurance system needs to consolidate fault and performance data and leverage both for all of its higher order functions like RCA and SIA.
- Various data sources, interfaces, and protocols. The cloud assurance system needs to collect data from multiple data sources and protocols and consolidate this data into unified device and service models. Some examples of different data sources and protocols are SNMP, syslog, WS API, Netflow, customer opened tickets, and so on.

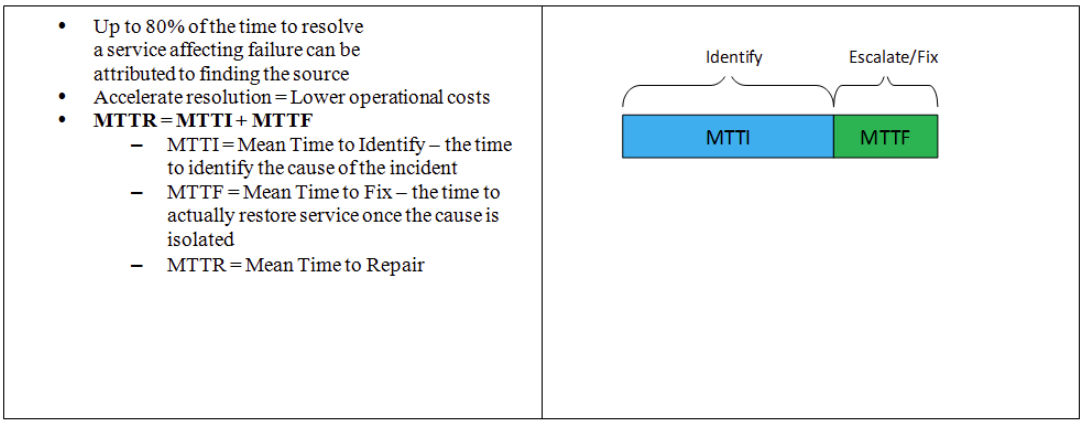
Consolidated monitoring provides the visibility necessary to enable the assurance system to provide more value add, while it can still achieve segmentation of operations through Role-based Access Control (RBAC) and flexible and configurable filtering capabilities.

Reducing Mean Time to Repair (MTTR)

In high pressure Network Operations Center (NOC) environments, operators handle various types of faults, isolate the issues, troubleshoot the problems, or escalate the problem to experts. To reduce the end-customer impact, it is very important to continuously improve MTTR. In traditional systems, general guidance for MTTR is less than 30 minutes from problem detection to problem resolution. For the cloud system, there is no generally accepted criteria, but expectations are that it will perform at least no worse than traditional systems.

Figure 1-4 illustrates the concept of MTTR.

Figure 1-4 Reducing Mean Time to Repair



The VMDC system consists of multiple technologies and components such as compute, storage, network, and network services components. The VMDC system is integrated to leverage these multiple technologies to create a platform for SPs and Enterprises to offer cloud services. Due to the interdependence of the components in the VMDC system, fault and performance issues in these components impact the services offered. The large number of components and technologies necessary to deliver cloud services increases the challenge of identifying the root cause and normalizing and correlating the faults that are generated by each of the individual components.

System scale plays a key role in creating the need for specific notifications about system failures and a reduced set of faults on the NOC operator dashboard. For example, due to the large size of a VMDC system that serves multiple end-customers, the assurance system can potentially generate thousands of events/faults on the NOC dashboard. If the NOC operator has to look at every fault generated by each domain manager, then the NOC operator may become overwhelmed. This can result in a time-consuming task for the NOC operator, who has to review hundreds of events/faults to identify the actionable events and then escalate those to the experts. This fault isolation time period results in higher mean-time-to-investigate/identify, and hence longer MTTR. This all equates to longer downtimes and unsatisfied end customers.

To reduce the MTTR, it is very important that the NOC operators receive specific notifications identifying the root cause of a failure. To achieve this, CLSA VMDC provides fault processing capabilities across components and domain managers and improves the correlation within the components and domains. CLSA VMDC refers to RCA that spans across multiple domains as X-domain RCA.

Northbound OSS/BSS integration

Almost every SP and many large Enterprises have existing OSS/Business Support Systems (BSS) deployed and operational (e.g., ticketing systems, MoM systems, problem and incident management systems, etc.). The SP staff and processes are generally aligned with the existing OSS/BSS workflows. VMDC is a new solution for SPs, however, SPs expect the VMDC assurance solution to integrate with their existing OSS/BSS.

The individual VMDC system components do offer interfaces to integrate with the OSS systems via SNMP Traps, syslogs, and emails, however, since each device and domain manager is an independent application, the integration interfaces are not consistent, and the number of integration points would be large (on the order of dozens of interfaces for VMDC system). Although the assurance domain manager integration northbound with the SP OSS is a one-time task, it needs ongoing maintenance due to:

- Need for ongoing fine-tuning.
- Changes in the underlying system and interfaces (e.g., API changes on southbound devices and domain managers).
- Deployment of additional instances of domain managers.
- Addition of new components and domain managers in future service assurance enhancements.

In order to ease the integration of the VMDC system in existing OSS/BSS systems, and thus SP adoption of the VMDC system, the number of integration points between VMDC and the SP's OSS/BSS needs to be reduced. The SP needs to be shielded from all maintenance and changes in the underlying VMDC system and interfaces unless the change is introducing significant new functionality to the SP. This can be achieved by providing single normalized interfaces from CLSA VMDC.

CLSA VMDC 2.3 Summary of Changes

CLSA VMDC 2.3 extends the VMDC assurance solution to provide support for several advanced features and to expand coverage of VMDC device discovery and monitoring. The list below identifies the major new features supported in this release.

- ASA 5555
- ACE 4710
- Nexus 7004 with SUP2 and F2 FabricPath Line Cards
- VMware VM to EMC VNX Impact Graphing

Table 1-1 lists the document updates associated with these enhancements for ease of reference.

Table 1-1 CLSA VMDC 2.3 Summary of DIG Updates

Section Title	Section Description
CLSA VMDC 2.3 Summary of Changes, page 1-7	Identifies CLSA VMDC 2.3 DIG updates (this section)
VMDC System Overview, page 2-1	Updated overview of VMDC System to include VMDC 2.3
CLSA VMDC System Architecture, page 3-1	Added VMDC 2.3 architecture and new hardware coverage

CLSA VMDC 3.0 Summary of Changes

CLSA VMDC 3.0 extends the VMDC assurance solution to provide support for several advanced features and to expand coverage of VMDC device discovery and monitoring. The list below identifies the major new features supported in this release:

- Zenoss High Availability Support
- New Zenoss Northbound Service Impact Trap
- New device support for both EMC VMAX and VNX block storage
- Cisco VMDC device families support extended (Nexus, ASA, UCS)
- New Zenoss Sample Tenant Portal

Note CLSA VMDC version numbering is closely tied to VMDC IaaS releases. As new devices are added to the VMDC infrastructure, CLSA VMDC will include new device support for discovery and monitoring in follow-on releases. Subsequent CLSA VMDC releases will also continue to enhance support for SIA and RCA, expanding coverage out-of-the-box for network infrastructure.

Table 1-2 lists the document updates associated with these enhancements for ease of reference.

Table 1-2 CLSA VMDC 3.0 Summary of DIG Updates

Section Title	Section Description
CLSA VMDC 2.3 Summary of Changes, page 1-7	Identifies CLSA VMDC 3.0 DIG updates (this section)
VMDC System Overview, page 2-1	Updated overview of VMDC System to include VMDC 3.0
Zenoss Service Impact SNMP Trap, page 3-21	New section providing details for the Zenoss Service Impact Trap

CHAPTER 2

VMDC System Overview

Cloud Service Assurance for VMDC (CLSA VMDC) is the service assurance system used to monitor Cisco VMDC-based cloud deployments. This chapter provides a brief overview of the VMDC system and its components.

The VMDC system is the Cisco reference architecture for Infrastructure as a Service (IaaS) cloud deployments. This Cisco IaaS cloud architecture is designed around a set of modular Data Center (DC) components consisting of building blocks of resources called pods. A pod, or Point of Delivery, comprises the Cisco Unified Computing System (UCS), SAN and NAS storage arrays, Access (switching) layers, Aggregation (switching and routing) layers connecting into the Data Center Service Node (DSN)-based Services layer, and multiple 10 GE fabric using highly scalable Cisco network switches and routers.

The VMDC system is built around the UCS, Nexus 1000V, Nexus 5000 and Nexus 7000 switches, Multilayer Director Switch (MDS), Aggregation Services Router (ASR) 9000, ASR 1000, Adaptive Security Appliance (ASA) or Adaptive Security Appliance Services Module (ASASM), Catalyst 6500 DSN, Application Control Engine (ACE), Nexus 1000V, Virtual Security Gateway (VSG), VMware vSphere, EMC VMAX/VNX, and NetApp FAS storage arrays. Cloud service orchestration is currently provided by the BMC Cloud Lifecycle Management (CLM) suite, and in the future, by Cisco Intelligent Automation for Cloud (CIAC).

Figure 2-1 provides a synopsis of the functional infrastructure components comprising the VMDC system.

Figure 2-1 VMDC Infrastructure Components

This chapter presents the following topics:

VMDC Modular Components

The VMDC system architecture provides a scalable solution that can address the needs of Enterprise and Service Provider cloud data centers. This architecture enables customers to select the design that best suits their immediate needs while providing a solution that can scale to meet future needs without retooling or redesigning the DC. This scalability is achieved using a hierarchical design with two different modular building blocks, pod and Integrated Compute and Storage (ICS) stack.

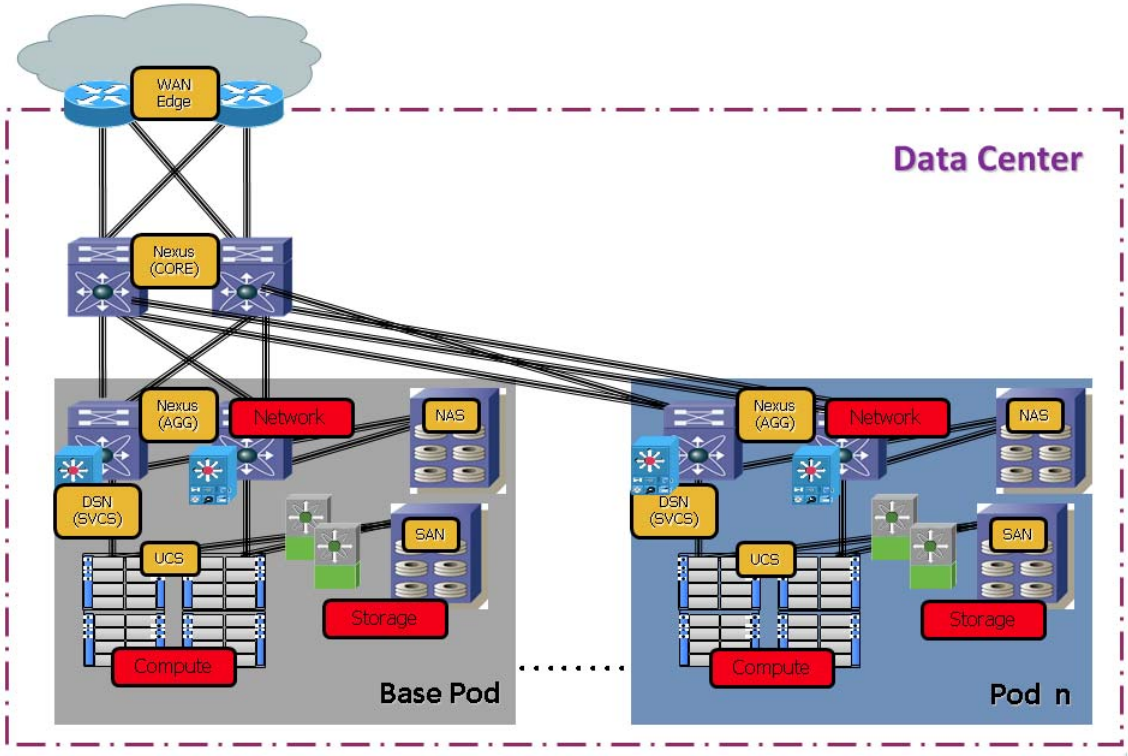
Point of Delivery (Pod)

The modular DC design starts with a basic infrastructure module called a pod, which is a logical repeatable construct with predictable infrastructure characteristics and deterministic functions. A pod identifies a modular unit of DC components and enables customers to add network, compute, and storage resources incrementally. This modular architecture provides a predictable set of resource characteristics (network, compute, and storage resource pools and power and space consumption) per unit that are added repeatedly as needed.

In this design, the Aggregation layer switch pair, Services layer nodes, and one or more integrated compute stacks are contained within a pod. The pod connects to the Core layer devices in the DC. To scale a DC, additional pods can be deployed and connected to the Core layer devices.

Figure 2-2 illustrates how pods can be used to scale compute, network, and storage in predictable increments within the DC.

Figure 2-2 VMDC Pods for Scaling the Data Center

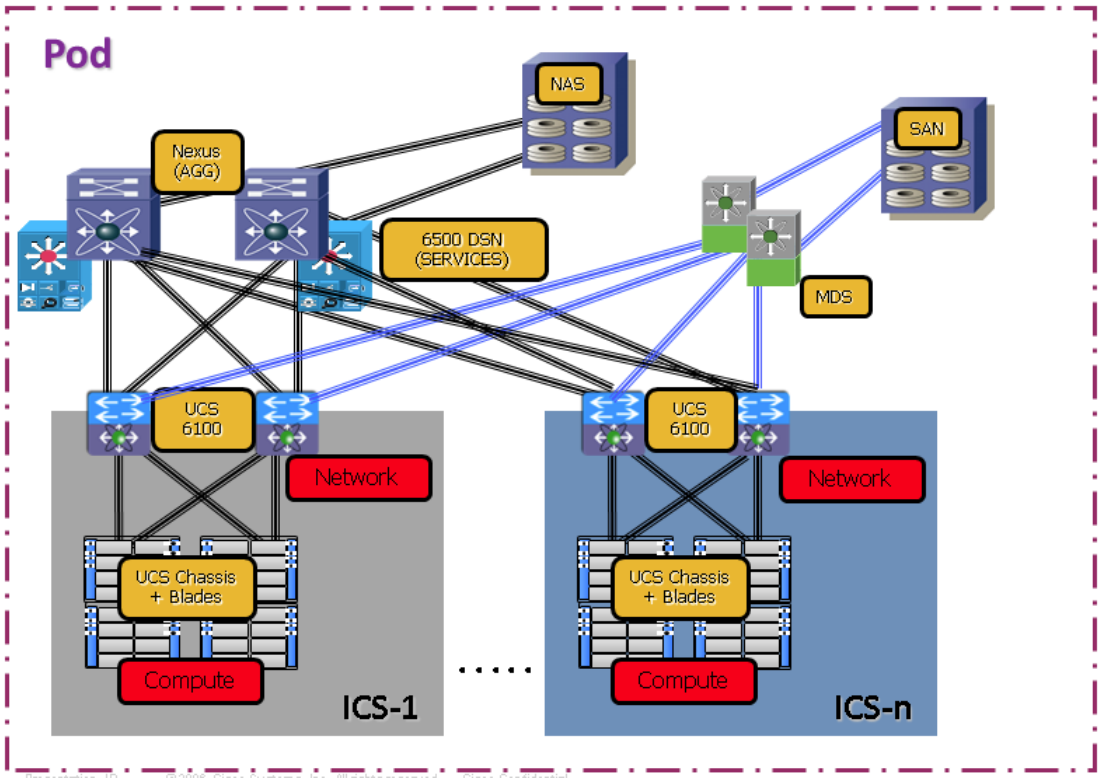


Integrated Compute and Storage (ICS) Stack

The second modular building block used is a generic ICS based on existing models, such as the VCE Vblock or NetApp FlexPod infrastructure packages. The VMDC architecture is not limited to a specific ICS definition, but can be extended to include other compute and storage stacks. An ICS can include network, compute, and storage resources in a repeatable unit. In this document, the Access layer switch pair, storage, and compute resources are contained within an ICS. To scale a pod, providers can add additional integrated compute stacks and can continue to scale in this manner until the resources reach the pod design limit.

Figure 2-3 illustrates how integrated compute stacks can be used to scale the pod.

Figure 2-3 VMDC ICS for Scaling the Data Center



VMDC System Architecture

The VMDC system utilizes a hierarchical network design for High Availability (HA) and scalability. The hierarchical or layered DC design uses redundant switches at each layer of the network topology for device-level failover that creates a highly available transport between end nodes using the network. DC networks often require additional services beyond basic packet forwarding, such as Server Load Balancing (SLB), firewall, and intrusion prevention. These services might be introduced as modules populating a slot of one of the switching nodes in the network or as stand-alone appliance devices. Each service approach also supports the deployment of redundant hardware to preserve High Availability (HA) standards set by the network topology. This layered approach is the basic foundation of the VMDC design to provide scalability, performance, flexibility, resiliency, and service assurance. VLANs and Virtual Routing and Forwarding (VRF) instances are used to provide tenant isolation within the DC architecture, and routing protocols within the VRF instances are utilized to interconnect the different networking and service devices.

The VMDC 2.2, 2.3, and 3.0 releases are the latest released versions of this architecture. This section provides a brief synopsis of the VMDC 2.2, 2.3, and 3.0 systems.

For detailed information on the VMDC 2.2 system architecture, refer to the following documents:

- VMDC 2.2 Design Guide
- VMDC 2.2 Implementation Guide

For detailed information on the VMDC 2.3 system architecture, refer to the following documents:

- VMDC 2.3 Design Guide

- VMDC 2.3 Implementation Guide

For detailed information on the VMDC 3.0 system architecture, refer to the following documents:

- VMDC 3.0 Design Guide
- VMDC 3.0 Implementation Guide

Note Information on previous VMDC system releases can be found at VMDC System Releases.

Note While the CLSA VMDC Design and Implementation Guide (DIG) references the VMDC 2.2, 2.3, and 3.0 systems, previous versions of the VMDC system are also supported. The CLSA VMDC system also supports other DC designs, as well as the VCE Vblock and NetApp FlexPod stacks.

The VMDC 2.2, 2.3, and 3.0 systems utilize a hierarchical multi-tenant DC architecture based on either VRF-Lite or FabricPath to provide secure separation between tenants. Besides scalability, platform, and tenancy model differences, the VMDC 2.2/2.3 and 3.0 systems also differ in the Layer 2 (L2) technologies utilized within the pod to provide redundancy and multi-pathing capabilities.

VMDC 2.2

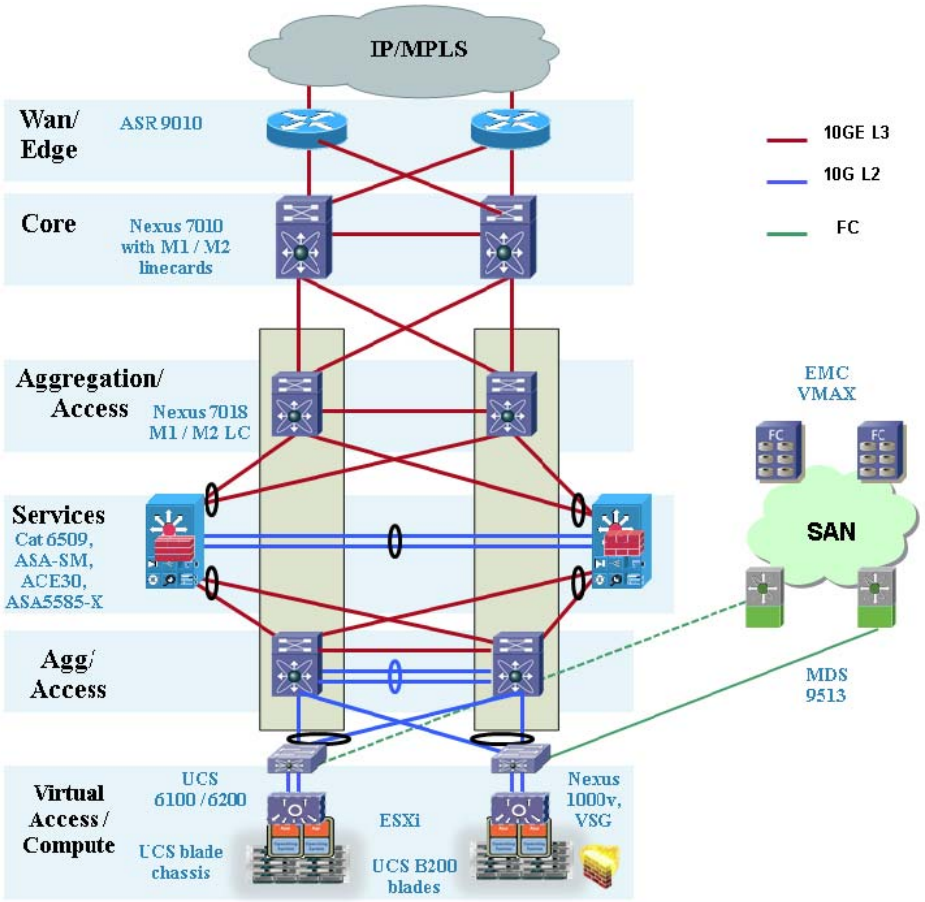
The VMDC 2.2 architecture utilizes a Virtual Port-Channel (vPC) on the Nexus 7000 and Nexus 5000 switches to provide link and chassis redundancy capabilities. Downstream switches (like the UCS 6100/6200 Fabric Interconnect and the Catalyst 6500 DSN) dual connect to a pair of Nexus 7000 aggregation switches, and the individual cross links across the chassis are bundled into a vPC link. The vPC across the chassis protects against any individual link or chassis failures and also provides L2 multi-pathing across the link members to provide higher aggregated bandwidths. In this design, the Nexus 7000 is utilized as the aggregation switch, while the Nexus 5000 and UCS 6100/6200 act as access switches. Only M1 (or M2) line cards are needed on the Nexus 7000 switches in this design.

This multi-layered VMDC 2.2 architecture is comprised of Core, Aggregation, Services, and Access layers. This architecture allows for DC modules to be added as demand and load increases. It also provides the flexibility to create different logical topologies utilizing device virtualization, the insertion of service devices, and traditional Layer 3 (L3) and L2 network configurations. Figure 2-4 provides a logical representation of the VMDC 2.2 architecture, with the Services layer comprised of the Catalyst 6500 DSN, ACE30, and ASASM (or ASA 5585-X).

The VMDC 2.2 architecture forms the basis for the Cisco SP Cloud Smart Solutions Premier Offer for Cloud Ready Infrastructure kit.

Figure 2-4 provides a logical representation of the VMDC 2.2 architecture, with the Services layer comprised of the Catalyst 6500 DSN, ACE30, and ASASM (or ASA 5585-X).

Figure 2-4 VMDC 2.2 System Architecture



VMDC 2.3

The VMDC 2.3 architecture is based on VMDC 2.2 and also utilizes a Virtual Port-Channel (vPC) on the Nexus 7000 and Nexus 5000 switches to provide link and chassis redundancy capabilities. Downstream switches (Nexus 5500 or UCS 6100/6200 Fabric Interconnect) dual connect to a pair of Nexus 7000 aggregation switches, and the individual cross links across the chassis are bundled into a vPC link. The vPC across the chassis protects against any individual link or chassis failures and also provides L2 multi-pathing across the link members to provide higher aggregated bandwidths. In this design, the Nexus 7000 is utilized as the aggregation switch, while the Nexus 5000 and UCS 6100/6200 act as access switches. While the VMDC 2.3 system has been validated with F2 line cards on the Nexus 7004, any Nexus 7000 platform and any of M1, M2, F2 or F2e line cards can be used in this design.

This multi-layered VMDC 2.3 architecture is comprised of Aggregation (Nexus 7004) and Access (Nexus 5548 and UCS 6248) layers, along with services appliances (ACE 4710 and ASA 5500 appliances). VMDC 2.3 is an optimized version of VMDC 2.2 to meet a reduced cost and footprint, yet higher tenancy scale VMDC requirements from customers. In order to reduce the cost of the overall solution, and to increase tenant scalability (by reducing BGP and HSRP consumption on the Nexus 7000 Agg) the following changes have been made, from the VMDC 2.2 to VMDC 2.3 architectures:

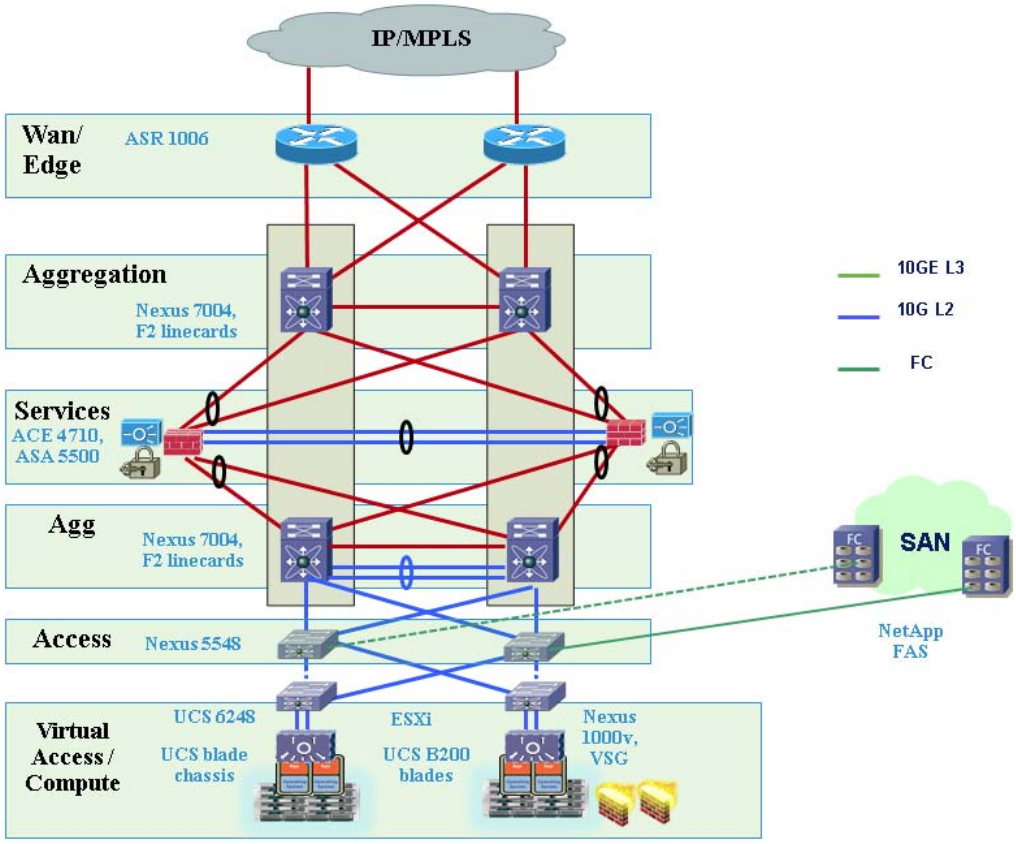
- Utilize ASR 1000 as WAN/PE layer, instead of ASR 9000
- Eliminate Nexus 7000 DC Core layer

- Utilize Nexus 7004 as the Aggregation device Use cheaper F2 line cards on the N7004 (instead of M2 or M1) Eliminate Catalyst 6500 DSN and utilize ASA 5500 and ACE 4710 service appliances connecting directly to Nexus 7004 Agg
- Optimized tenancy models
- New Copper network container
- ACE used in one-arm SLB mode (instead of two-arm mode)

The VMDC 2.3 architecture forms the basis for the Cisco SP Cloud Smart Solutions Standard Offer for Cloud Ready Infrastructure kit.

Figure 2-5 provides a logical representation of the VMDC 2.3 architecture, with the Services layer comprised of the ACE 4710, ASA 5585-X, and ASA 5555.

Figure 2-5 VMDC 2.3 System Architecture



VMDC 3.0

The VMDC 3.0 design introduces FabricPath into the VMDC system architecture. Instead of using a vPC, the VMDC 2.0 architecture utilizes FabricPath on the Nexus 7000 and Nexus 5000 switches to provide link and chassis redundancy. FabricPath uses Intermediate System to Intermediate System (IS-IS) as the underlying control plane for MAC learning, and also provides much higher link capacity utilization through 16x equal cost multi-pathing (ECMP). FabricPath provides a larger, flatter L2 domain, with the capability for "Any VLAN Anywhere" across the DC. FabricPath can be used to extend the server VLANs within the pod, or across pods in the DC. In this design, the Nexus 5000 (and/or Nexus 7000) switches are used as FabricPath Leaf (Access) nodes, while Nexus 7000 switches are used as

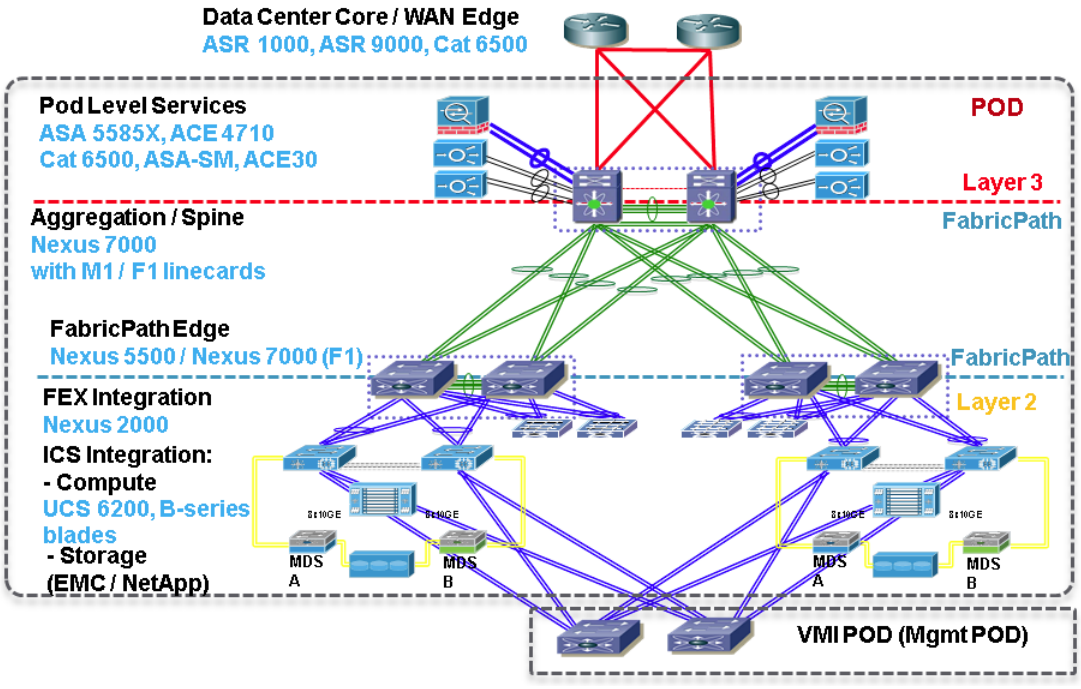
FabricPath Spine (Aggregation) nodes in the FabricPath domain. F1 (or F2) line cards are used on the Nexus 7000 switches for FabricPath downstream L2 connectivity, while M1 (or M2) line cards are utilized on the Nexus 7000 for upstream L3 connectivity.

Cisco FabricPath provides the following benefits to the VMDC 3.0 solution:

- Replaces Spanning Tree with a mature link state protocol (IS-IS)
- Single control protocol used for unicast/multicast forwarding, and VLAN pruning
- Expansion of the L2 domain—Any VLAN Anywhere (within pod and across pods)
- Improved link capacity usage through 16-way ECMP
- Improved convergence time
- Easy expansion—add additional access or spine nodes in plug-n-play manner

Figure 2-6 provides a logical representation of the VMDC 3.0 typical DC architecture with FabricPath utilized within the pod, and the Services layer comprised of the ACE 4710 and ASA 5585 appliances (or Catalyst 6500 DSN, ACE30, and ASASM).

Figure 2-6 VMDC 3.0 System Architecture



CHAPTER 3

CLSA VMDC System Architecture

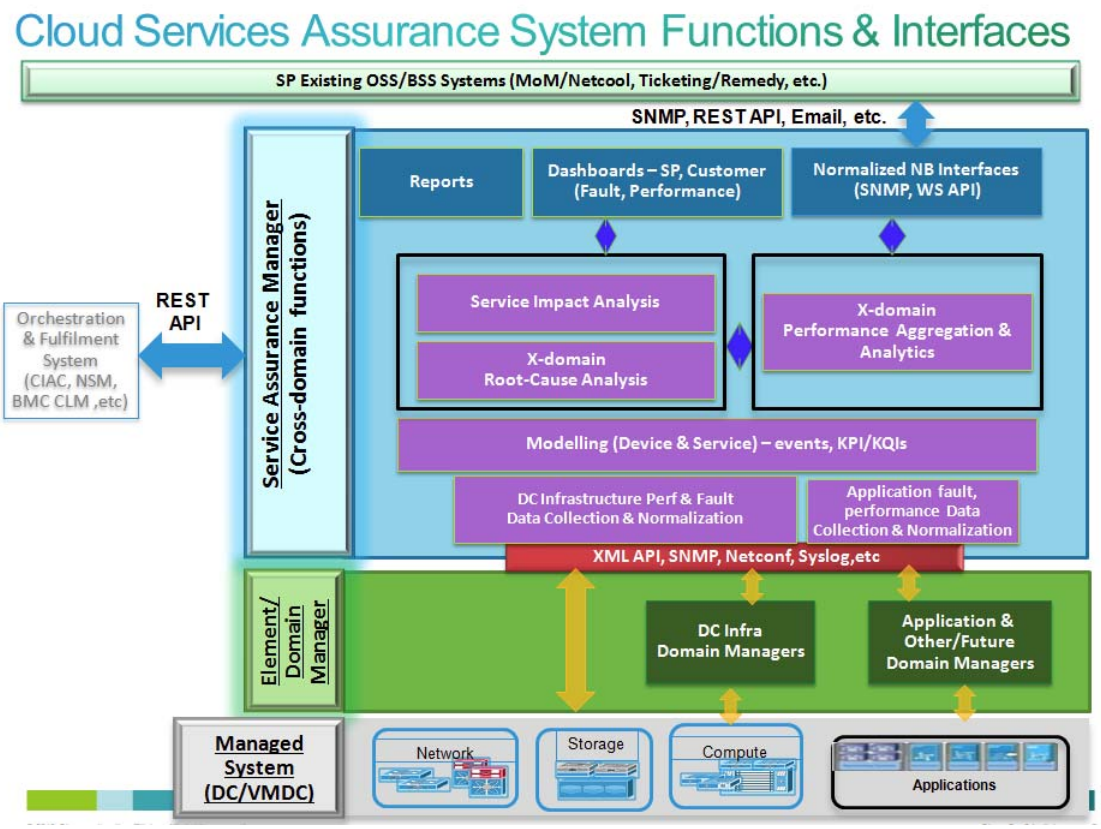
This chapter provides an overview of the Cloud Service Assurance for VMDC (CLSA VMDC) system architecture.

- Functional View, page 3-1 and Component View, page 3-3 provide the functional and component views of the CLSA VMDC system architecture.
- System Components, page 3-4 defines the components and interfaces used to deliver the system functions.
- Monitored Components and Services, page 3-5 lists the VMDC devices that are monitored by CLSA VMDC.
- Key Functions, page 3-6 defines the functions of the new architecture.

Functional View

Figure 3-1 illustrates the functional framework for CLSA VMDC. This functionality is delivered with one or more of the integrated products/components. In CLSA VMDC, only a subset of this functionality is available. This section defines the functional layers of this architecture and identifies the layers that are available in CLSA VMDC.

Figure 3-1 Functional View of CLSA VMDC Architecture



The **Managed Device Layer** consists of Data Center (DC) infrastructure including compute, storage, and network components with instrumentation for inventory, fault, and performance data collection. The instrumentation used in this system includes Simple Network Management Protocol (SNMP), syslog, XML Application Programming Interface (API), NETCONF, vSphere API, and so on. Details of interfaces used per VMDC component are included in Monitored Components and Services, page 3-5.

The **Domain/Element Management Layer** includes the UCS Manager (UCSM) and vCenter. They provide intra-domain inventory, fault, and performance monitoring for UCS and VMware hosts and VMs. These domain managers offer northbound interfaces APIs as well as SNMP and syslog interfaces. CLSA VMDC utilizes UCS XML API and vSphere API interfaces. CLSA VMDC 3.0 also introduces the Storage Management Initiative Specification (SMI-S) Provider domain manager to incorporate EMC VMAX and VNX inventory, fault, and performance monitoring.

The **Service Assurance Manager (SAM) Layer** provides all inter-domain functions and a single pane of glass to monitor all VMDC domains including compute, storage, and network. The high-level functions of each of the SAM layers are as follows:

- **Data Collection Layer**. This layer leverages domain managers, third-party tools, and so on to obtain performance, availability, and event data for the end-to-end multi-domain system via a range of open protocols such as SNMP, SSL, WMI, and so on. This layer is responsible for normalizing this data into a consistent format and persisting data. Collected data includes inventory, fault, and performance type of information.
- **Modeling Layer**. This layer performs discovery, classification, and modeling to determine component dependencies and service dependency graphs. Both performance and fault data should be included in device and service models.

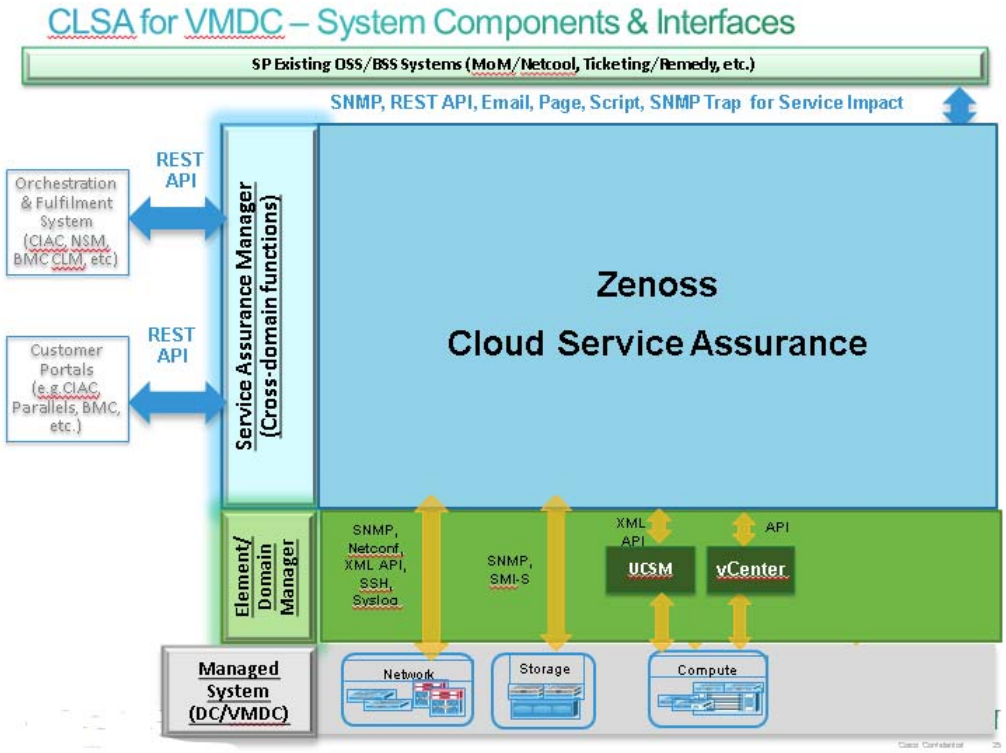
- **Service Model-based Technology**. CLSA VMDC uses service model-based technology which is described in more detail in Root Cause Analysis and Service Impact Analysis, page 3-14 and Zenoss Cloud Service Assurance Overview, page 4-1CLSA VMDC.
- **Root Cause Analysis (RCA)**. Leverages the dependency graph or analytics algorithms to determine which events are the probable root cause of the problem and which ones are just consequences that create noise. Therefore, RCA reduces Mean Time to Repair (MTTR). There are a number of different approaches to RCA, but most of them can be classified in one of the following technologies:
  1. Event correlation rules-based
  2. Topology and service model-based
  3. Analytics based
- **Service-Impact Analysis (SIA)**. Leverages the dependency graph or analytics algorithms and collects fault and performance data to do the following:
  - Answer who is impacted by the failures
  - Prioritize urgency of failure tickets based on business relevance
  - Determine whether redundancy protected the service
  - Identify failure impacted customers/tenants
  - Prevent future failures by identifying potential service impacting technical risks before they impact service
  - Provide data for SLA measurements and reporting
- **Performance Aggregation Layer**. This layer aggregates performance data from multiple domains (e.g, storage, network, compute for VMDC), normalizes it in the same format and units, provides threshold crossing alerts to the fault management part of the SAM, trends the data over time, and in some cases, performs additional analysis of the data.
- **Presentation Layer**. This layer provides a single view to do both fault and performance monitoring for the entire system. Presentation is done both via dashboards and reports. CLSA VMDC includes SP dashboards for both fault and performance.
- **Northbound Interface**. The Northbound Interface (NBI) is a special form of the presentation layer where normalized and enriched data is presented to northbound OSS/BSS systems via open interfaces such as WS API, SNMP, and email.

Component View

This section defines the components used to deliver those functions, as well as their interfaces. The key component of the architecture for CLSA VMDC is Zenoss Cloud Service Assurance (CSA), which plays the role of the SAM. In addition, several domain managers are utilized - UCS Manager (UCSM) for UCS hardware monitoring, VMware vCenter for monitoring the virtualized infrastructure, and SMI-S Provider for EMC VMAX and VNX monitoring.

Figure 3-2 illustrates the components and interfaces used to deliver the functional layers of the CLSA VMDC architecture.

Figure 3-2 Component View of CLSA VMDC Architecture



Key system interfaces include:

- Southbound interface instrumentation to collect data from managed system devices.
- Northbound interface to integrate with OSS/BSS systems such Manager-of-Managers (MoM) (e.g., IBM Netcool), ticketing systems (e.g., Remedy) and so on. The interfaces available from CLSA VMDC are SNMP, JSON API, email, page, commands, and Advanced Message Queuing Protocol (AMQP).
- CLSA VMDC offers the JSON API interface for integration with orchestration and fulfillment systems.

System Components

Table 3-1 lists the Cisco and third-party components used in CLSA VMDC.

Table 3-1 Cisco and Third-Party Components Used in CLSA VMDC

Vendor	Model	Description
Zenoss	Resource Manager 4.2.3	Zenoss CSA software module that performs resource discovery, monitoring, and modeling.
Zenoss	Impact 4.2.3	Zenoss CSA software module that performs service impact discovery and analysis.
Zenoss	Analytics 4.2.3	Zenoss CSA software module that performs long term data trending, processing, and reporting.

Table 3-1 Cisco and Third-Party Components Used in CLSA VMDC (continued)

Vendor	Model	Description
vCenter	vCenter 5.0	Domain manager for VMware based virtualization
Cisco	UCSM 2.0	Domain manager for UCS platform
EMC	SMI-S Provider 4.5.0.1	Domain manager for EMC VMAX and VNX platforms

Note The Zenoss software modules are packaged together as Zenoss CSA 4.2.3.

Monitored Components and Services

Table 3-2 lists the VMDC devices that are monitored by the CLSA VMDC system out-of-the-box and the instrumentation (interfaces) utilized by Zenoss CSA to collect data.

Table 3-2 VMDC Components Monitored by CLSA VMDC

Managed Component	Interfaces Utilized in CLSA VMDC 3.0
Compute Components	
UCS 5108; B-series blades	ICMP, UCSM XML API
UCS 6100, 6200	ICMP, UCSM XML API
VMware ESX and ESXi Hypervisors	ICMP, vSphere API
VMware Virtual Machines	ICMP, vSphere API
Storage Components	
MDS 9000	ICMP, SNMP
EMC VMAX [1]	ICMP, SMI-S API
EMC VNX [1]	ICMP, SMI-S API
FAS6080, FAS3000	ICMP, SNMP, SSH
Network Components	
UCS 6100, 6200	ICMP, UCSM XML API
Nexus 7000 (e.g., 7018, 7010, 7009, 7004 including M1 and F1/F2 cards) [3,4]	ICMP, NETCONF, SNMP
Nexus 5000 (e.g., 5548, 5596, and 5020)	ICMP, NETCONF, SNMP
Nexus 3000 [1]	ICMP, NETCONF, SNMP
Nexus 2000 (e.g., 2248 and 2232)	ICMP, NETCONF, SNMP
Nexus 1000V / Nexus 1010 [1]	ICMP, NETCONF, SNMP
ASR 9000	ICMP, SNMP, SSH
ASR 1000	ICMP, SNMP
Network Services Components	
Catalyst 6500 VSS	ICMP, SNMP, SSH

Table 3-2 VMDC Components Monitored by CLSA VMDC (continued)

Managed Component	Interfaces Utilized in CLSA VMDC 3.0
ACE (e.g., ACE20, ACE30, ACE4710 2)	ICMP, SNMP, ACE XML API
FWSM	ICMP, SNMP
ASASM [1]	ICMP, SNMP
ASA 5555 [2]	ICMP, SNMP
ASA 5580-40	ICMP, SNMP
ASA 5585-40	ICMP, SNMP
Virtual Security Gateway	ICMP, SNMP, NETCONF, SSH

1 Denotes new enhancement for CLSA VMDC 3.0.

2 Denotes new enhancement for CLSA VMDC 2.3.

3 FabricPath F1 cards are added to Nexus 7000 devices for CLSA VMDC 3.0.

4 FabricPath F2 cards are added to Nexus 7000 devices for CLSA VMDC 2.3.

For detailed information on software releases, please refer to the following documents:

- VMDC 2.2 Implementation Guide
- VMDC 2.3 Implementation Guide
- VMDC 3.0 Implementation Guide

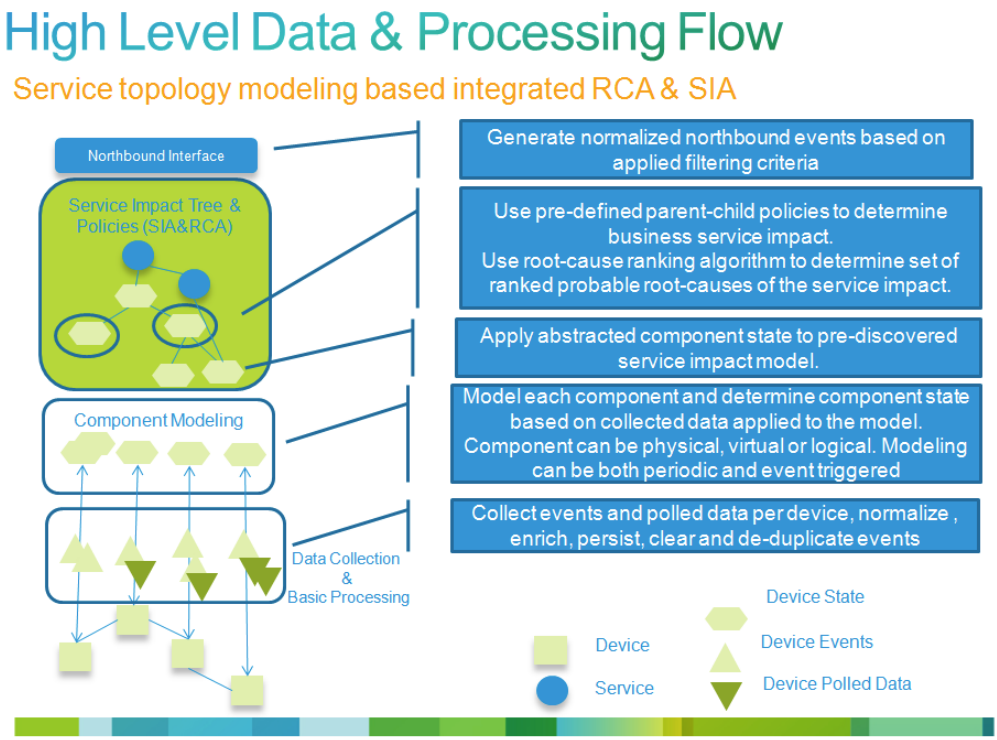
Note Information on previous VMDC system releases can be found at VMDC System Releases.

Key Functions

This section describes the key functions of CLSA VMDC.

In the overall lifecycle of assurance services, the first task that has to be completed is enablement of service assurance services. Automatic Enablement of Service Assurance, page 3-7 provides details about enabling service assurance, including provisioning and automatic discovery. Once assurance services are enabled, they can be used for Day 2 operations. Figure 3-3 illustrates and explains the high-level, end-to-end data flow through the fault and problem management part of CLSA VMDC.

Figure 3-3 End-to-End Fault and Problem Management Data and Processing Flow



The following sections discuss each of the stages and functions in this sample data flow:

- Fault Performance, Configuration Data Collection, and Device Modeling, page 3-10
- Event Processing, page 3-13
- Root Cause Analysis and Service Impact Analysis, page 3-14
- Northbound Interface, page 3-19

This section also discusses the following additional functions related to the overall platform and its use:

- Performance Management, page 3-29
- Dashboards, page 3-30
- Reporting, page 3-35
- Multiservices, page 3-37

Automatic Enablement of Service Assurance

Automatic enablement of service assurance can be achieved in a couple of different ways. Fundamentally, the following are approaches that can be taken to automate service enablement and life cycle:

1. Reduce necessary amount of configuration (by using technology that is self learning (e.g., self learning thresholds))
2. Automatic discovery (by assurance system)
3. Programmatic orchestrated provisioning (via integration with orchestration system)

CLSA VMDC focuses on automatic discovery. CLSA VMDC also provide APIs for programmatic orchestrated provisioning, but they are not integrated or validated with any particular orchestration system. Automatic discovery and APIs are discussed in the following sections.

- Automatic Discovery, page 3-8
- Zenoss APIs for Programmatic Provisioning, page 3-9

Automatic Discovery

The following types of objects are automatically discovered in CLSA VMDC:

- Monitored devices (e.g., UCS, Nexus 7000, MDS 9000, etc.)
- Sub-components of devices and their relationships (e.g., UCS chassis, blades, fabric interconnect, etc.)
- Tenant-based Service Impact Analysis (SIA) model for the compute (e.g., tenant Virtual Machine (VM) mapping to service impacting resources, both dedicated and shared vCenter and UCSM managed resources). The exception is tenant name and its link to the service, which cannot be discovered, but relies on orchestrated provisioning. In this release, tenant name and mapping to the VM are provisioned manually, but the API is provided.

Figure 3-4 and Figure 3-5 illustrate examples of automatic enablement of service assurance.

Figure 3-4 Real-time Automatic Discovery of Device Components - Cisco UCS

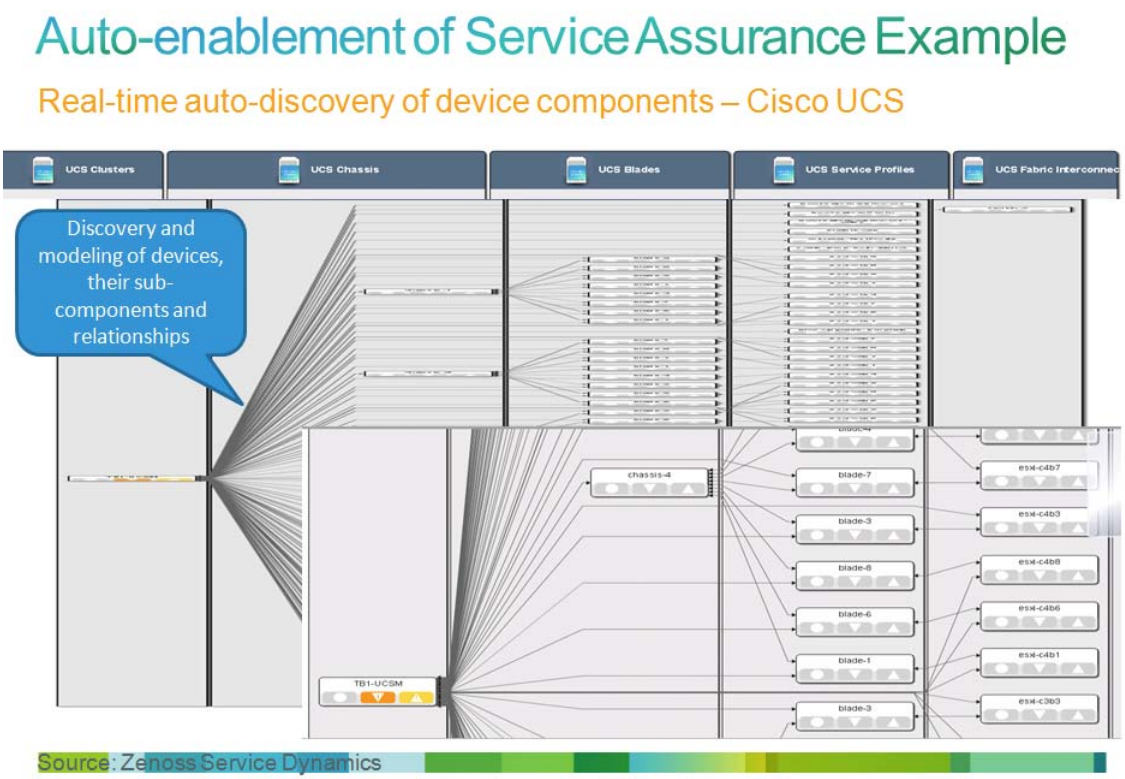
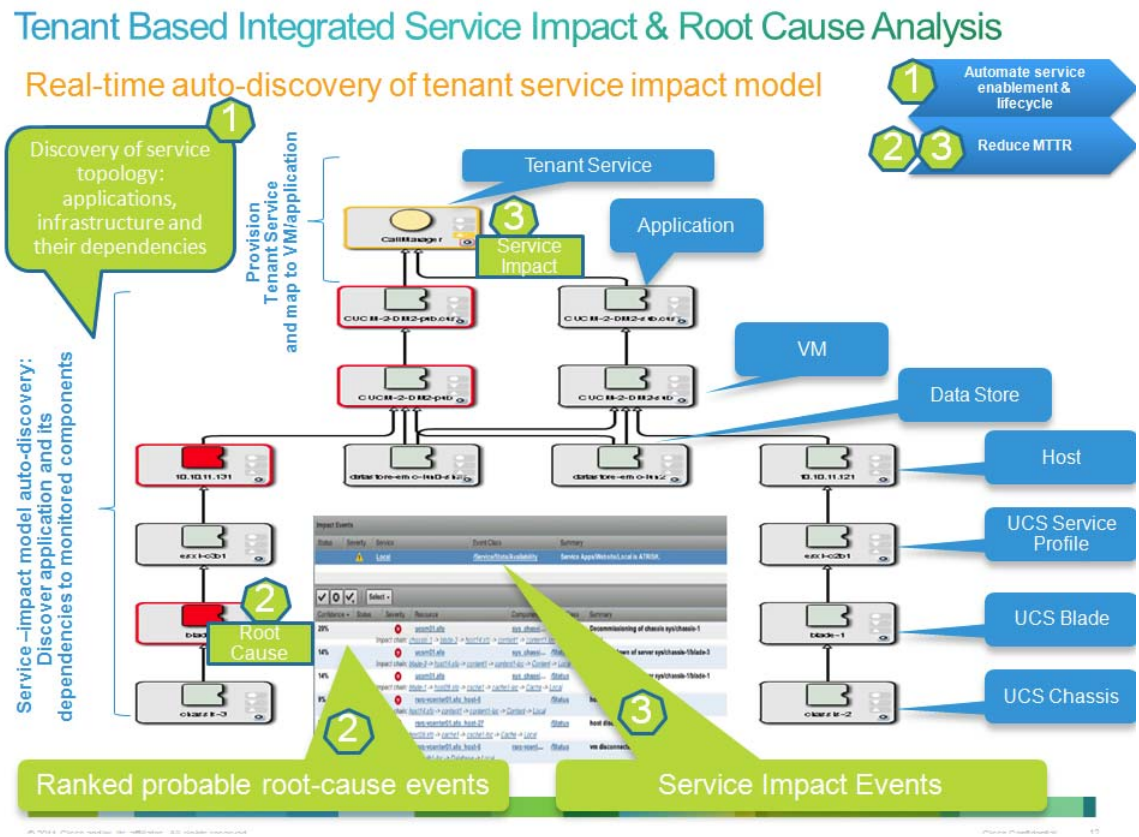


Figure 3-5 Real-time Automatic Discovery of Tenant Service Impact Model



Zenoss APIs for Programmatic Provisioning

CLSA VMDC offers APIs to programmatically provision the following components in the service impact tree:

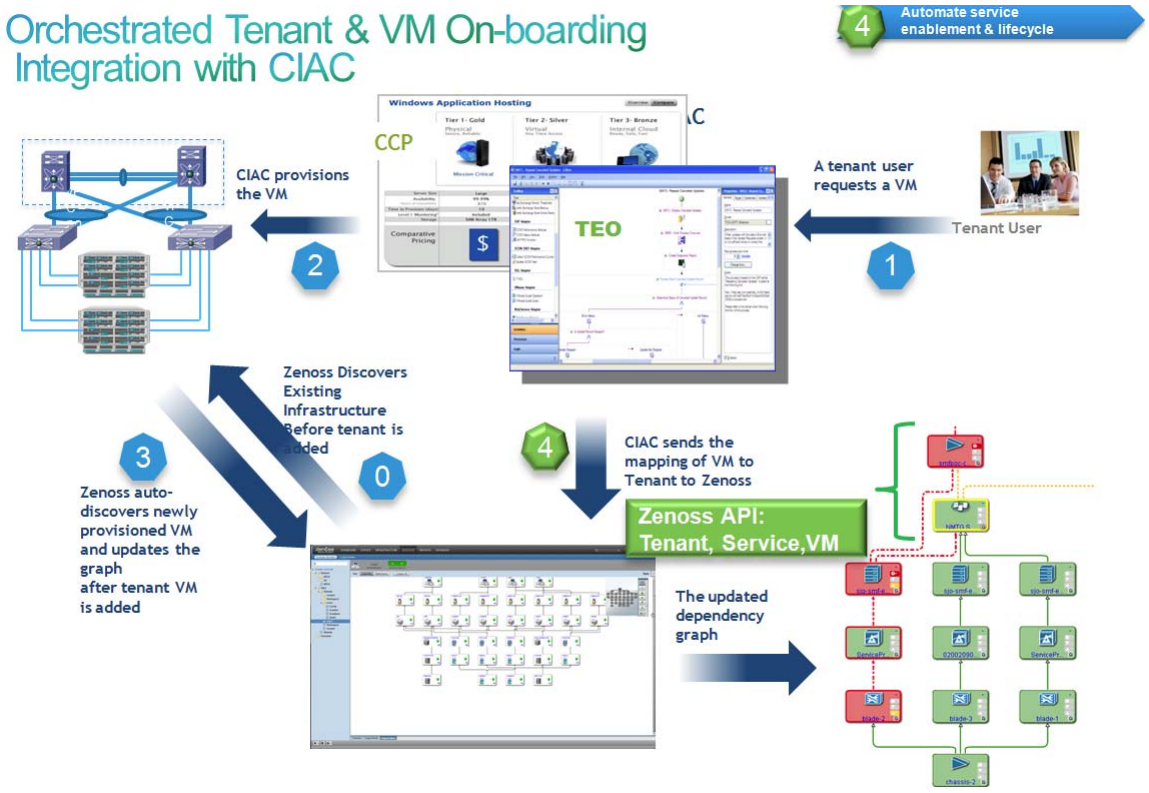
- Tenant Name
- Tenant ID
- Service Name
- Service ID
- VM Name
- VM ID

This enables automatic onboarding of the tenant and tenant compute service, which maps them to the already automatically discovered VM and its relationships to shared hardware.

Note Proof of Concept (PoC) of this functionality integrated with the Cisco Intelligent Automation for Cloud (CIAC) orchestration stack has been performed by Cisco Advanced Services; however, it was not validated as part of the CLSA VMDC system. If this functionality is desired in the field before it is included as part of the Systems Development Unit (SDU) system release, then Cisco Advanced Services can perform integration with the desired orchestration stack using the provided API.

Figure 3-6 illustrates the high-level workflow that provisions the tenant and tenant service and then maps the workflow to the automatically discovered VM and the rest of the automatically discovered infrastructure.

Figure 3-6 Zenoss Tenant Provisioning Using CIAC Orchestration



Fault Performance, Configuration Data Collection, and Device Modeling

Consolidated monitoring and data collection at the SAM layer is one of the fundamental functions of CLSA VMDC. Consolidated monitoring and data collection is characterized by the following attributes:

- Various domains (applications, compute, storage, network). The cloud assurance system needs to provide a single pane of glass to monitor components from various domains.
- Fault and performance data. The cloud assurance system needs to consolidate fault and performance data and leverage both for all of its higher order functions like RCA and SIA.
- Various data sources, interfaces, and protocols. The cloud assurance system needs to collect data from multiple data sources and protocols and consolidate this data in unified device and service models. Some examples of different data sources and protocols are SNMP, syslog, WS API, Netflow, customer opened tickets, and so on.

Zenoss Data Collection

Zenoss CSA offers consolidated monitoring for VMDC, including consolidation of domains (i.e., support for OS, compute, storage, and network), consolidation of performance and fault data (i.e., takes into consideration both polled performance data, asynchronous events it receives, as well as synthetic

events it generates for both performance and availability), and consolidation of data sources (i.e., device monitoring models utilize multiple data sources such as SNMP, syslog, API, and consolidate it within unified device model).

Zenoss CSA uses an agentless data collection approach, which is critical for the type of scale expected in cloud systems. Instead of installing an agent on monitored devices, Zenoss supports a rich set of protocols to enable data collection. A list of protocols used for data collection from VMDC devices is included in Monitored Components and Services, page 3-5. The following is a more comprehensive list of data collection interfaces that the Zenoss CSA platform supports:

Event input:

- SNMP
- Syslog
- XML Remote Procedure Call (RPC)
- JavaScript Object Notation (JSON)/API
- AMQP
- Windows Event Log

Easily configurable protocol usage:

- Secure Shell (SSH)
- Java Management Extensions (JMX)
- Windows Management Instrumentation (WMI)
- Perfmon
- Any script that returns data in a known format (such as Nagios)

Other collection mechanisms (model/performance/event data):

- Internet Control Message Protocol (ICMP)
- Telnet
- JMX
- Hypertext Transfer Protocol (HTTP) - Web Transactions
- Oracle
- Structured Query Language (SQL) Server
- MySQL
- Apache (mod_status)
- memcache
- Splunk Queries
- Simple Mail Transfer Protocol (SMTP)
- SMI-S Provider
- Post Office Protocol (POP)
- UCSM XML API
- vSphere Simple Object Access Protocol (SOAP) API
- vCloud Director
- Amazon EC2 and CloudWatch

- Cisco CallManager (AXL)
- Domain Name System (DNS)
- Lightweight Directory Access Protocol (LDAP)
- Network Time Protocol (NTP)
- File Transfer Protocol (FTP)
- Internet Relay Chat (IRC)
- Extensible Messaging and Presence Protocol (XMPP)
- Remote Procedure Call (RPC)
- Network News Transfer Protocol (NNTP)

Zenoss Device Modeling

Device modeling in Zenoss goes beyond traditional device discovery; it also uses standard Management Information Bases (MIBs) to discover interesting aspects of the device and automatically defines models for that device type. Once modeled, these learned attributes can be inherited as part of the model when a new device of the same type is discovered again. The information below describes various attributes of the Zenoss device modeling process.

Initial Zenoss Model (plugins):

- Interfaces to access device and objects of interest (KPI statistics, events, thresholds, etc.) are statically defined
- Models are assigned to a device class

Device Modeling:

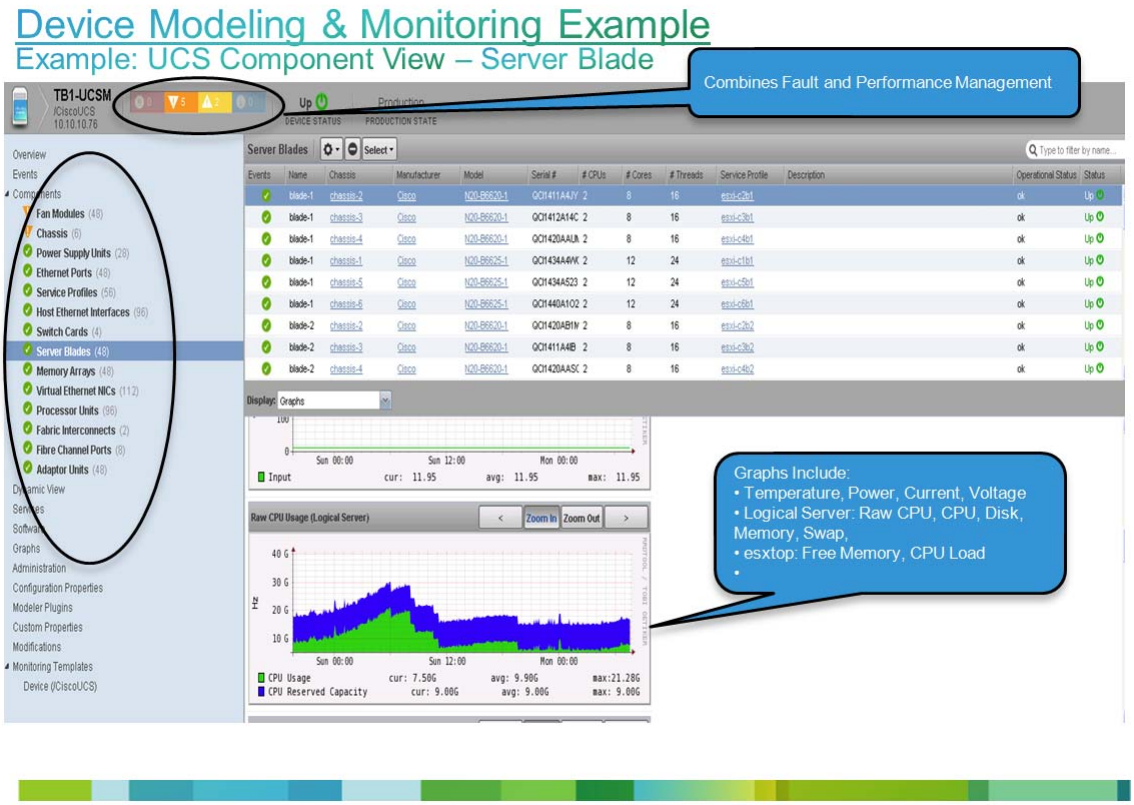
- During individual device discovery, all modeler plug-ins for the device class are automatically considered, and a model per instance of the device is created.
- After discovery modeling, monitoring and event processing automatically starts.

Device Remodeling:

- Model per device instance can dynamically change in response to events (e.g., blade removed, etc.)
- ZenModelerDeamon - per collector configuration happens every 12 hours
- ZenVMwareDeamon (exception for VMware and remodels every 4 hours)
- List of events that trigger remodeling is configurable (default set exists)

An example of unified monitoring using Zenoss CSA is illustrated in Figure 3-7.

Figure 3-7 Unified Monitoring Using Zenoss CSA



Event Processing

In CLSA VMDC, event processing is divided into two categories:

- Basic event processing
- Event processing that is part of RCA and SIA

This section only describes basic event processing functions, while RCA and SIA are discussed in the following sections. The basic event processing functions included in this system are event classification, normalization, de-duplication, enrichment, persistence, and clearing.

Event classification groups similar events in event classes, so that some of the more complex processing may be simplified by looking at event classes rather than each individual event.

Event normalization translates various formats of the raw collected data into a single format that is used by the SAM. Often, the same format or subset of the fields of normalized format can be sent to northbound systems. This function allows simplified integration of northbound systems since they have to deal with a single event format for multiple device types and instrumentation protocols.

Event de-duplication eliminates multiple events that have the exact same content with the exception of the time stamp. After de-duplication, a single event is kept, and typically a counter indicating the number of occurrences of the event is added, as well as a timestamp indicating the first and last occurrence of the duplicate event.

Event persistence archives all events to be used for forensic analysis. In some systems, persistence exists only on post-processed events, while in others, for raw events as well.

Event clearing is used to indicate when the original condition for which the event was raised is removed. Explicit event clearing is done by generating clearing events with the field within the clearing event, which points to the ID of the event that it is clearing. For example, if an interface down event for a specific interface had an ID of ID1, when the interface goes up again, an event with ID2 should be raised, which includes as one of its fields a reference to event ID1. Explicit event clearing is recommended. In addition to explicit clearing, time-based clearing can be utilized as well. Time-based clearing clears the event after a specific time interval elapses from the time that the original event was received.

Root Cause Analysis and Service Impact Analysis

One of the key functions of CLSA VMDC is Root Cause Analysis (RCA) and tenant-based Service Impact Analysis (SIA).

The objective of RCA is to reduce MTTR by determining which events are probable root causes of the problem and which events are just consequences that create noise.

The following are the objectives of tenant-based SIA:

- To prioritize the urgency of failure tickets based on business relevance.
- To determine whether redundancy protected the service.
- To identify failure impacted customers/tenants.
- To prevent future failures by identifying potential service impacting technical risks before they impact service.
- To enable Service Level Agreement (SLA) measurements and reporting.

The following sections are detailed:

- Zenoss SIA and RCA, page 3-14
- VMDC Assurance Service Models, page 3-16
- VMDC RCA and SIA Use Cases, page 3-18

Zenoss SIA and RCA

Zenoss CSA uses model-based SIA, which produces a set of ranked probable root causes as a by-product of SIA. This service impact-based approach to RCA is a fundamentally different approach from legacy rule-based systems:

- Bottom-up. What services are impacted by conditions below (Zenoss) vs.
- Top-down. What is the cause of problem at service level (legacy products)

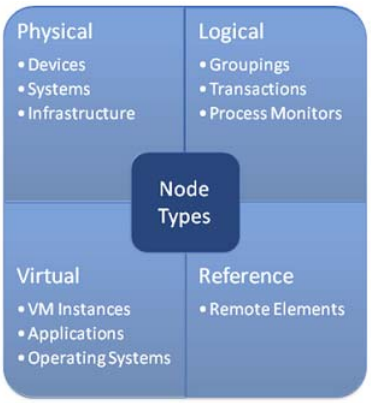
Zenoss does not determine a single root cause, but instead identifies multiple related events (probable root cause events) and presents the following:

- A root cause ranking algorithm is utilized to rank probable root cause events in order of confidence that the event is the actual root cause event. This algorithm ranks impact events based on a variety of criteria, including the severity of the event, service graph depth, and the number of graph branches affected by an event.
- Hierarchical service dependency graphs provide a visual indication of probable root causes leading to a service impact.

Events flow through the graph referencing molecular node policies to determine whether they should be passed, filtered, aggregated, or masked. There are a few key elements of RCA and SIA in Zenoss CSA. Each assurance service within Zenoss is modeled with a service impact tree that consists of a set of nodes, policies applied to the nodes, and the relationships between the nodes:

- The service can be arbitrarily defined and can be a very abstract service that consists of other sub-services, or on other extreme, one can even define a single physical interface as a service. This provides a very flexible framework for service definition.
- Model nodes represent arbitrary components such as physical, logical, or virtual resource. For example, nodes can represent an end-to-end service such as voice, a virtual resource such as a VM, or a physical resource such as a chassis or physical interface. The following four types of nodes are currently supported, as illustrated in Figure 3-8:
  - **Physical**. Systems, infrastructure, and network devices that a service relies on.
  - **Virtual**. Software components that make up a service.
  - **Logical**. Aspects of a service that must be measured or evaluated as a set to determine state (facilitates extension of an impact graph by providing a hook to incorporate arbitrary events into impact analysis).
  - **Reference (future release)**. Provide a link to dependencies managed by an external instance of Zenoss or other management system capable of propagating state information to Zenoss.

Figure 3-8 Node Types



- Policy is defined per node, which allows it to move as the resources move, which is a critical characteristic for the cloud environment. Zenoss refers to this policy as a molecular policy since it is defined per node. Zenoss utilizes a very simple policy that can define the state of the node solely as a function of the state of its children nodes, which allows for service impact "rules" decoupling from device events resulting in the following:
  - "Rules" defined in a single place for any given device or service: device events processing in event processing software modules, service impact processing in service impact graphs (i.e., device events do not need to be considered in service level rules)
  - Simplified development and maintenance of cross-domain service impact and RCA customizations: do not have to correlate device events from multiple devices to determine cross-domain service impact and possible root causes
  - Note that whenever desired, device events can be used as part of service impact "rules" via use of logical nodes whose rules define how to interpret the service impact of specific events based on its type and severity.
  - Policy can be global or contextual:

Global policy applies to device/service type in any service graph.

Contextual policy applies only to device/service in the particular service graph.

– Each node has a default policy applied, which reduces the need for custom configuration. The default policy is often sufficient, but can be modified where required via GUI or API. Figure 3-8 illustrates a sample node policy.

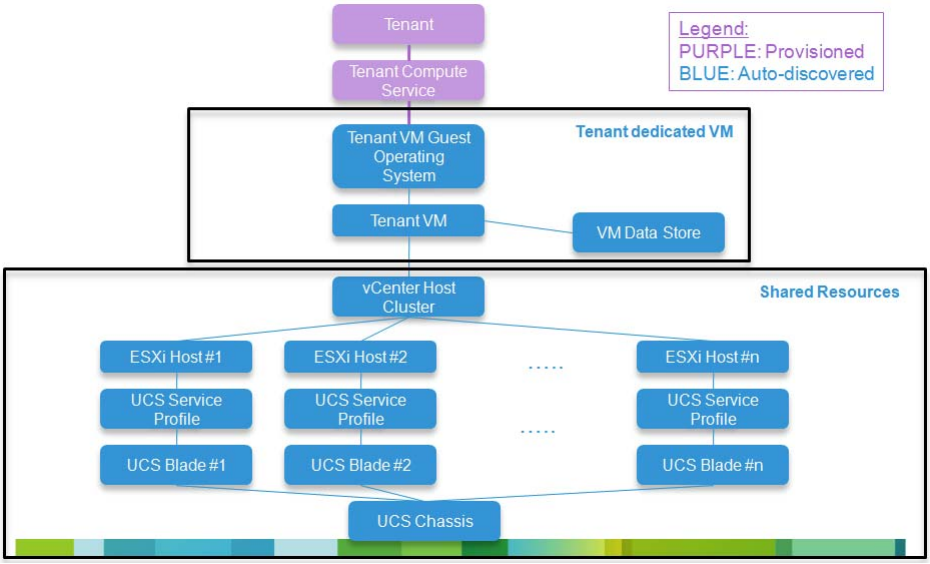
VMDC Assurance Service Models

In order to perform SIA, CLSA VMDC uses service models with polled and asynchronous data to perform SIA and RCA. CLSA VMDC offers an out-of-the-box tenant service model for compute. In future releases, CLSA VMDC will expand the library of out-of-the-box service models that will be validated and delivered as part of this system, however, note that users can easily customize service models as well as create new ones.

Tenant Compute Assurance Service

Figure 3-9 defines the out-of-the-box tenant compute service model to be delivered as part of CLSA VMDC. More details are provided about this service model in Zenoss SIA and RCA, page 3-14.

Figure 3-9 Tenant Compute Assurance Service Model - Generic Application



Service Model Policy

Each node (referred to as the parent node) in the service model has a policy defined that calculates the state of that node based on the state of its children and any explicit events associated with the parent node.

For the particular service model illustrated in Figure 3-9, the specific policies listed in Table 3-3 should be applied.

Table 3-3 Service Model Policy Decisions

Node	Node State	If Child Node State
Tenant Compute Service	UP/DOWN/AT RISK	UP/DOWN/AT RISK
Tenant Guest OS	UP/DOWN/AT RISK	UP/DOWN/AT RISK
Tenant VM	UP/DOWN/AT RISK	UP/DOWN/AT RISK
ESXi Cluster	UP/DOWN AT RISK	All Children up/down At least One Child Down/At Risk
ESXi Host	UP/DOWN/AT RISK	UP/DOWN/AT RISK
UCS Blade	UP/DOWN/AT RISK	UP/DOWN/AT RISK

Out-of-the-box, all nodes use the default policy where the worst impact wins. The one exception is the VMware cluster, which is DOWN if all children are DOWN and DEGRADED if any nodes are DOWN or DEGRADED.

In addition to considering the parent/child policy, the explicit state of the nodes is determined by both availability and events for components the node represents. For VMware and UCS nodes, the explicit node impact status is determined mainly by modeled properties. As modeling occurs or various events are received, Zenoss reassesses the impact state by querying the Zenoss model. For example, when a VM power off event is received, the model is updated and the VM status is reassessed and updated.

Service Model Variations

Note that the model defined in this section illustrates a single-tier application with a single VM. Variation of this service model would be models for the following:

- Multi-tier application, where there would be multiple "tenant dedicated VM" blocks tied to the tenant compute service. The tenant compute service default policy may need to be customized.
- Single-tier application that supports application level redundancy via clustering (e.g., Cisco UC applications such as CUCM). In this case, the model would be modified to include multiples of "tenant dedicated VM" blocks. The default policy used for the "tenant compute service" should be applicable. An example of this service model is illustrated in Figure 3-9.

Service Model Enablement

Most of this model is automatically discovered, while the top node of the service model needs to be provisioned. Typically, provisioning would be done in an automated way when the tenant and VM get onboarded. In CLSA VMDC, there is no integration with the orchestration stack, and as such, the top node of the service model is manually provisioned. Note that in real deployments, per-tenant manual provisioning is not an option, in which case either an available Zenoss API can be used by the orchestration platform of choice, or if not provisioned, the tenant service impact is still possible but results are given in the VM context rather than tenant service context. For example, there would be no automatic mapping between tenant name, tenant service name, and VM ID.

In future CLSA VMDC releases, integration with VMDC orchestration stacks will be implemented and validated. In addition to automatic discovery of the service model from VM down, if operating systems such as Windows or Linux are deployed, they should also be automatically discovered.

Mobility Handling

The host to VM relationship is given by VMware during modeling stage. Whenever VMware generates an event that indicates VM movement, Zenoss reacts and remodels the source and target hosts to update its model. Depending on the event collection interval specified in the Zenoss configuration, the model change can take anywhere from 15 seconds to 3 minutes. With the out-of-the-box configuration, the average time would be about 1 minute.

Redundancy Implications

A service model with three service states accounts for redundancy. The AT RISK state is used to indicate conditions where the service or service model node is still functioning despite a failure of one of its children because redundancy protected the service. For the particular service model shown in Figure 3-9, redundancy capabilities that are accounted for include the following:

- If one of the blades/hosts fails, and the vCenter cluster that VM belongs to has multiple blades/hosts, then the VM node is marked AT RISK as opposed to DOWN based on the status of its children. Note that explicit VM related state and events can result in the state of the VM node being down even though the state of its children alone would result in an AT RISK state
- In a case where there is application level redundancy and thus more than one VM and application deployed for single tier applications, there is also service model redundancy built in on the application/VM level. For example, a service is AT RISK if one of the application nodes/VMs is DOWN because the remaining application/VM nodes provides redundancy for the failed application/VM node.

VMDC RCA and SIA Use Cases

Once service impact models are defined, the data is applied to service impact models to maintain real-time state of the service availability and performance, as well as to determine probable root cause of any failures that may happen. This section provides a list of failure scenarios (use cases) validated as part of the CLSA VMDC test effort, for which the out-of-the-box compute service model can determine correct probable root cause and service state for previously defined services. All of the use cases are validated in an environment where VMware High Availability (HA) is deployed.

Refer to Root Cause Analysis and Service Impact Analysis, page 3-14 for an example workflow illustrating a UCS switch failure event, including screen shots.

Use Case Name (Fault):

- VM Failure
- VM vNIC failure
- VM vMotion - VM vMotion is not a true fault event, since the VM stays up, however, the impact graph does track the VM's host swap.
- ESXi host failure
- UCS Blade failure
- UCS chassis failure
- UCS P/S failure
- UCS FEX failure
- UCS 6100 chassis failure
- UCS 6100 interfaces to UCS 5100 failure
- VM CPU degradation (Threshold Crossing Alert (TCA))

- VM Memory degradation (TCA)
- Host CPU degradation (TCA)
- Host Memory degradation (TCA)

Northbound Interface

One of the key, new functions of CLSA VMDC architecture is a single, normalized Northbound Interface (NBI) provided by the SAM.

The key objectives of the single, normalized interface are:

- **To simplify and reduce the cost of integrating providers existing northbound system with the CLSA VMDC system**. The provider needs to integrate and maintain just one interface rather than multiple dozens of interfaces towards individual devices and/or domain managers. CLSA VMDC is responsible for absorbing updates related to any relevant changes in the underlying system and devices.
- **To enable CLSA VMDC to be inserted in various business and operational deployment environments**. This is achieved by offering a variety of interface protocols, rich filtering capabilities, and notifications with tenant awareness.
- **To enable CLSA VMDC to simplify service assurance of overlaid application based systems that are deployed on top of VMDC infrastructure**. An example of this type of system is the Hosted Collaboration Solution (HCS). This is achieved by providing tenant service level notifications rather than device level notifications, which enables a service overlay (or multi-tier SIA) to be implemented by HCS, and as such, Cloud Service Assurance-HCS (CLSA-HCS) would have to deal with the state of only a handful of services coming from CLSA VMDC, rather than thousands of events coming from individual VMDC devices.

Zenoss northbound integration is supported via:

- JavaScript Object Notation (JSON)/Representational State Transfer Application Programming Interface (ReST API)
- SNMP Traps (ZENOSS-MIB.txt and ZENOSS-IMPACT-MIB.txt)
- Syslog
- Event queues (AMQP and Java/Python wrappers) and event commands (command line call with event context)
- SMTP email

Configurable filtering capabilities are offered to provide different data to different northbound consumers. The following sections describe the interfaces, data, and filtering capabilities in more detail.

Sections

SNMP Northbound Interface

One of the key requirements for CLSA VMDC is to offer asynchronous notifications via SNMP. These notifications are consumed either by the provider's existing northbound systems such as MoM, ticketing, and SLA management systems, or by other Cisco systems deployed on VMDC architecture such as HCS.

Regardless of the source or type of the event, all events should be sent using the same normalized format, however, as discussed in this chapter, there may be differences in the values of the populated fields based on the type of events (e.g., service impact events contain information about service name and state, while device level events do not).

Zenoss SNMP Notification Content

Zenoss CSA uses custom Zenoss MIB implementations for northbound notifications. The original SNMP MIB addresses the resource manager part of the product, but not the service impact part. MIB extensions have been designed to address service impact events and related probable root cause events as a part of the this phase of CLSA VMDC. For a discussion of the new service impact trap, see Zenoss Service Impact SNMP Trap, page 3-21.

Events associated with devices use ZENOSS-MIB for notifications. The ZENOSS-MIB.txt file is located in the following Zenoss directory: $ZENHOME/share/mibs/site. Device level SNMP notifications can be sent to multiple destinations. Refer to the Zenoss Cloud Service Assurance Installation and Administration Guide for more information regarding notifications.

Table 3-4 maps the fields of Zenoss MIBs to the SAM requirements.

Table 3-4 Zenoss MIB Fields

Zenoss MIB Field Name	Description
evtId	Unique identifier ID of the event
evtDedupid	De-duplication ID of the event
evtDevice	Device associated with event
evtComponent	Device component associated with event
evtClass	Event classification
evtKey	Event key used for refining event granularity beyond device and component. Used in de-duplication, automatic clearing.
evtSummary	Event message truncated to 128 characters
evtSeverity	Event severity number: 0=clear(normal), 1=debug, 2=info, 3=warning,4=error, 5=critical
evtState	Event state number: 0=new, 1=acknowledged, 2=suppressed
evtClassKey	Class key for rule processing often matches component
evtGroup	Logical grouping of event sources
evtStateChange	Last time event changed through administrative activity
evtFirstTime	First time an event was received
evtLastTime	Last time an event was received
evtCount	Number of times this event has been seen
evtProdState	Production state of the device or component associated with this event
evtAgent	Collector process that received or created this event

Table 3-4 Zenoss MIB Fields

Zenoss MIB Field Name	Description
evtDeviceClass	Class of device that this event is associated with
evtLocation	Location of device that this event is associated with
evtSystems	Systems containing the device that this event is associated with
evtDeviceGroup	Groups containing the device that this event is associated with
evtIpAddress	IP address that this event was generated or sent from
evtFacility	Syslog facility if the event was initially sent as a syslog
evtPriority	Syslog priority if the event was initially sent as a syslog
evtNtEvId	Windows NT_EVENT_ID if the event was initially received from Windows event log
evtOwnerId	User that acknowledged this event
evtClearId	evtId that cleared this event
evtDevicePriority	Priority of the device that this event is associated with
evtClassMapping	Name of the event class mapping that matched this event

Zenoss Notification Filtering

Filtering capabilities using Zenoss Triggers can be used to customize notifications based on the needs of different northbound consumers:

- Multiple subscribers/receivers may receive notifications.
- Each notification subscriber/receiver may apply a different filter: one receiver may subscribe to service events, another may subscribe to compute events, and a third may subscribe to network events.
- Each system user should be able to apply different filters.

For more information regarding Triggers, refer to the Zenoss Cloud Service Assurance Installation and Administration Guide.

Zenoss Service Impact SNMP Trap

This section defines the SNMP notification for Zenoss Impact, which is new for CLSA VMDC 3.0. The following data is available internally within Zenoss Impact for service related events. This data was used by the notification script in CLSA VMDC 2.2.

- Service Name
- Severity
- Timestamp
- Service state
- URLs to EventDetail, page to acknowledge and close events, device events
- All events in the impact chain. Each event in impact chain includes:
  - Device
  - Component

- Device Class
- Event Class
- Severity
- Timestamp
- Message
- URLs to EventDetail, page to acknowledge and close events, device events

Zenoss Impact provides a flexible framework to define arbitrary services, including support for hierarchical service nesting. In such environments, the question arises for which nodes and/or levels of hierarchy notifications should be sent. Services are collected under Service Organizers. A Service Organizer consists of multiple folders and in each folder there is set of services. In Zenoss Impact, the notification trigger criteria is con

figured for Service Organizer folders and its services and not based on individual nodes and their hierarchy level in the impact tree. This approach provides good balance between flexibility to select notification trigger criteria and simplicity of implementation.

For CLSA VMDC to send notifications per service instance state change, the appropriate structure must be created to organize the services. The following sections discuss the folders and the structure used for the services defined in CLSA VMDC 3.0.

Service Organizers

Service Organizers are located on the left tab in the Impact GUI.

The Shared Services folder includes:

- Service Name X (e.g., Network Aggregation service, Network Core service, etc.)
- Service Name Y

The Customer Name folder includes:

- Tenant Service 1 (e.g., Tenant Compute service, Tenant Network service, etc.)
- Tenant Service 2

Notification Triggers

The user is able to select services and or/folders for which to send notifications. This action is available both in the GUI, as well as via the REST API so that the orchestration system at the time of onboarding the tenant service can select whether or not to enable notification for the service.

The notification policy should be selectable both per folder or per service instance. This enables support for the following use cases:

- Where a single operator or NB system manages and/or provides visibility to all services of single tenant/customer (since one folder is defined per tenant).
- Where different services of the same tenant are managed by different operators/NB systems, e.g., notification for IaaS services are sent to the IaaS operator while notifications for Unified Communications as a Service (UCaaS) services are sent to the UC operator.

Notification Timing

This section defines the guidelines for service impact notification triggers and timing. An attempt is made to balance any delay in notifications indicating change with excessive noise in events sent due to transient state during service impact analysis. In order to have the capability to delay some service impact notifications, there is a timer that can be configured (value range 0-10 minutes with default of three minutes).

Service impact notifications are triggered when the following events occur:

- If the service state changes (top-level service in each folder):
  - The notification indicating a service state change should always be sent immediately, regardless of the value of the notification delay timer. This enables the northbound system to immediately detect the change. Also, for northbound systems that are using service state notifications to measure service availability and SLA, this immediate notification enables more accurate service availability measurements.
  - When the service state changes back to UP, the event should serve as a clearing event for the previous service state change event. As such, the ID of the service event that it is clearing must be included.
- If the service state does not change, but most a probable root-cause event changes (i.e., root cause ranked with highest confidence% changes):
  - This trigger honors the notification delay timer, and as such, it is sent only if the event is generated after the notification timer expires.

The following example shows the use of the notification delay timer and the two notification types listed above. Assume that the following conditions exist:

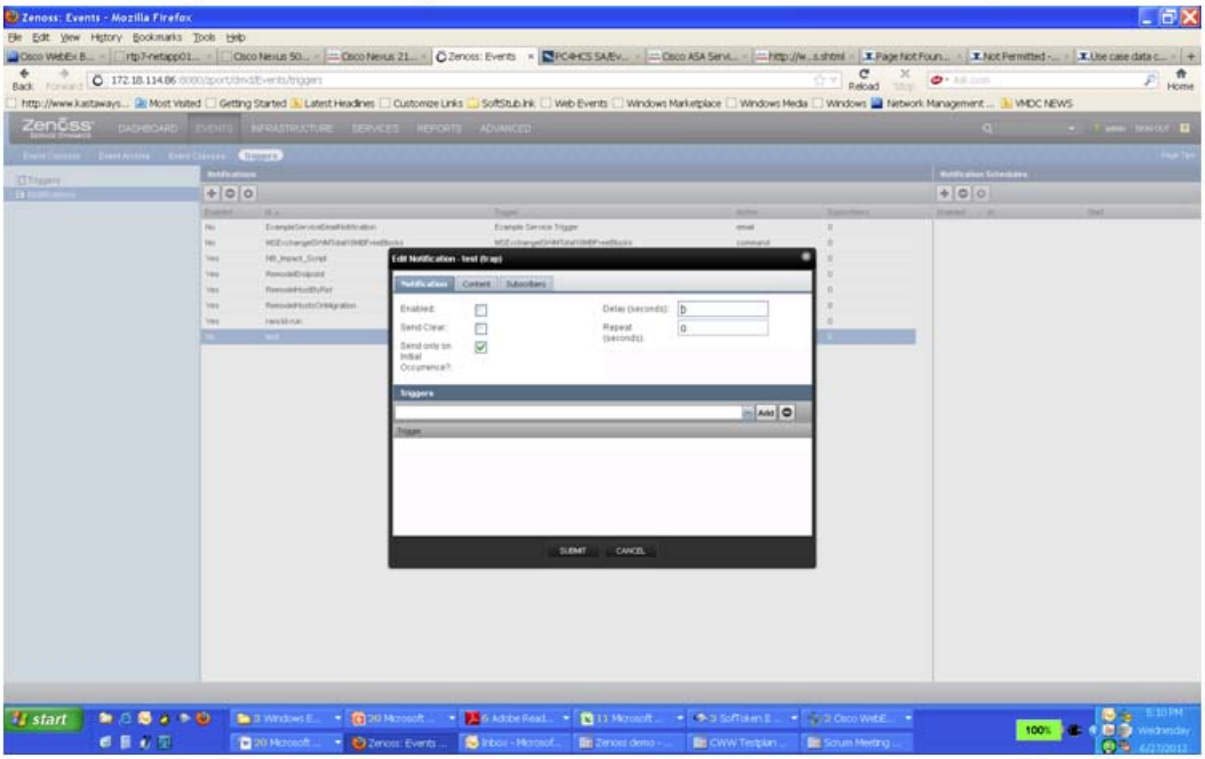
1. The notification delay timer is set to three minutes.
2. The root cause is a UCS blade failure, and the final service state for the IaaS service is AT RISK.

If these conditions exist, the following occurs:

1. At zero seconds, event E1 (VMware event for the VM) arrives. The root cause at that time is RC1= VM Failure.
2. A service impact event is sent northbound indicating that the IaaS state = AT RISK, RC=VM.
3. At one minute, event E2 (VMware event for the host) arrives. The root cause at that time is RC2= Host Failure. Since the notification delay timer is set to three minutes, there are no events sent northbound due to the change of root-cause events. Only one minute has passed since the service state change time.
4. At four minutes, event E3 (UCSM event for blade) arrives. The root cause at that time is RC3=UCS blade failure. A service impact event is sent northbound indicating that the IaaS state = AT RISK, RC= Blade.

Figure 3-10 shows the existing capability that Zenoss has to delay notifications and also to send the clearing events. The same capabilities would be extended to the service impact events.

Figure 3-10 Edit Notification Timer



Notification Content

The following fields should be included in the northbound service impact notification:

- **Folder Name (one up only)**. The customer name would typically be placed here, but keeping the field generic allows flexibility to use folders in any way desired (e.g., to represent shared infrastructure services, reseller, etc.). The operator can include the option to have a full folder path.
- **Folder Type**. The folder type indicates what the folder represent, e.g., for folders representing the customer name, the folder type would have value the value "customer."
- **Service Instance Name and systemwide unique ID**.
- **Service Type**. This field can be used to filter notifications by type of service that the northbound consumer is interested in, even though each instance of the service may be in different folders which are representing different customers.
- **Service State**. The service state is UP, DOWN, AT RISK, or DEGRADED.
- **URLs to Service Impact EventDetail**. This page acknowledges and closes events and device events.
- **Timestamp**.
- **Event clearing ID**. The ID of the event that is being cleared by this event.
- **Probable root-cause event name and systemwide unique ID (event with highest confidence level)**.
- **Probable root-cause confidence level**.
- **Probable root-cause device, component, and severity**.

- **Impact chain and ID to events in impact chain**. The ID can be used to retrieve the impact chain via REST API upon receipt of the notification.
- **URLs to probable root-cause EventDetail**. This page acknowledges and closes events and device events.

Note In CLSA VMDC 3.0, the following fields are not supported: Folder Type, Service Type, and URLs to probable root-cause event detail. In addition, the Event Clearing ID is implemented slightly differently than proposed above. The Service Instance Name & system wide unique ID is implemented in a field called zenImpactUUID. The initial and clearing events have the same zenImpactUUID, however they have states new and cleared.

Root-cause Event Notification

In addition to sending probable root-cause events as part of service impact notification, there is also a need to be able to send only probable root-cause events. For example, in cases of more catastrophic failures where a single root-cause event impacts a larger number of services, northbound systems that are not service focused may prefer to receive only one notification representing the root-cause event and not receive multiple service impacting notifications.

Even in this case, it is desirable to provide the relationship between the root-cause event and the services it impacted. This can be done by including a list of services impacted by the same root-cause event in the root-cause event notification URL or ID.

Root-cause notification is not a separate notification in CLSA VMDC 3.0; instead, the root-cause event is communicated as a field via the service impact notification.

WS or ReST API

The JSON API can be used to obtain the following:

- Device model and attributes data
- Performance data
- Event data
- Service data

Most of the information visible via the GUI can also be obtained via the JSON API.

In addition to retrieving data, the JSON API can also be used for the following:

- Managing events (acknowledge, clear, close)
- Adding devices to be monitored
- Setting production state
- Initiating discovery and modeling of devices
- Managing thresholds
- Managing reports
- Other configurations

More information on the JSON API can be found at the following URL:

http://community.zenoss.org/community/documentation/official_documentation/api

Northbound Integration Use Case Examples

This section includes typical use cases that illustrate the rich filtering capabilities of the NBI.

Sections

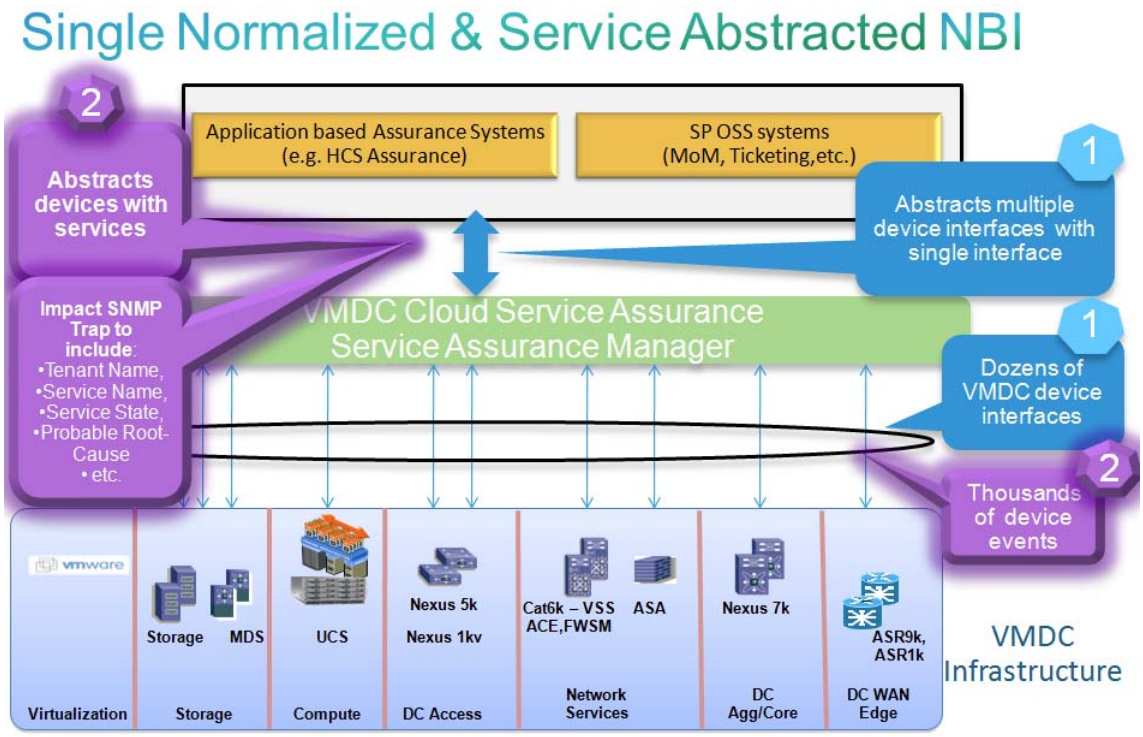
Abstraction via Single Interface

One of the key functions of the SAM layer as defined in CLSA VMDC architecture is the capability to provide a single, normalized NBI that is consistent regardless of the formats of data used by the underlying VMDC components. This allows simplified integration and ongoing interface maintenance with providers existing OSS systems as:

- There is only one integration point as opposed to the number of integration points being proportional to the number of VMDC devices and domain managers.
- Changes in any of the underlying interfaces on managed devices are absorbed by the SAM as opposed to the provider having to update OSS systems every time there is a change in one of the managed components.

Figure 3-11 illustrates how the VMDC system is abstracted via a single interface to the provider's existing OSS system. The purple areas represent the enhancements for CLSA VMDC 3.0.

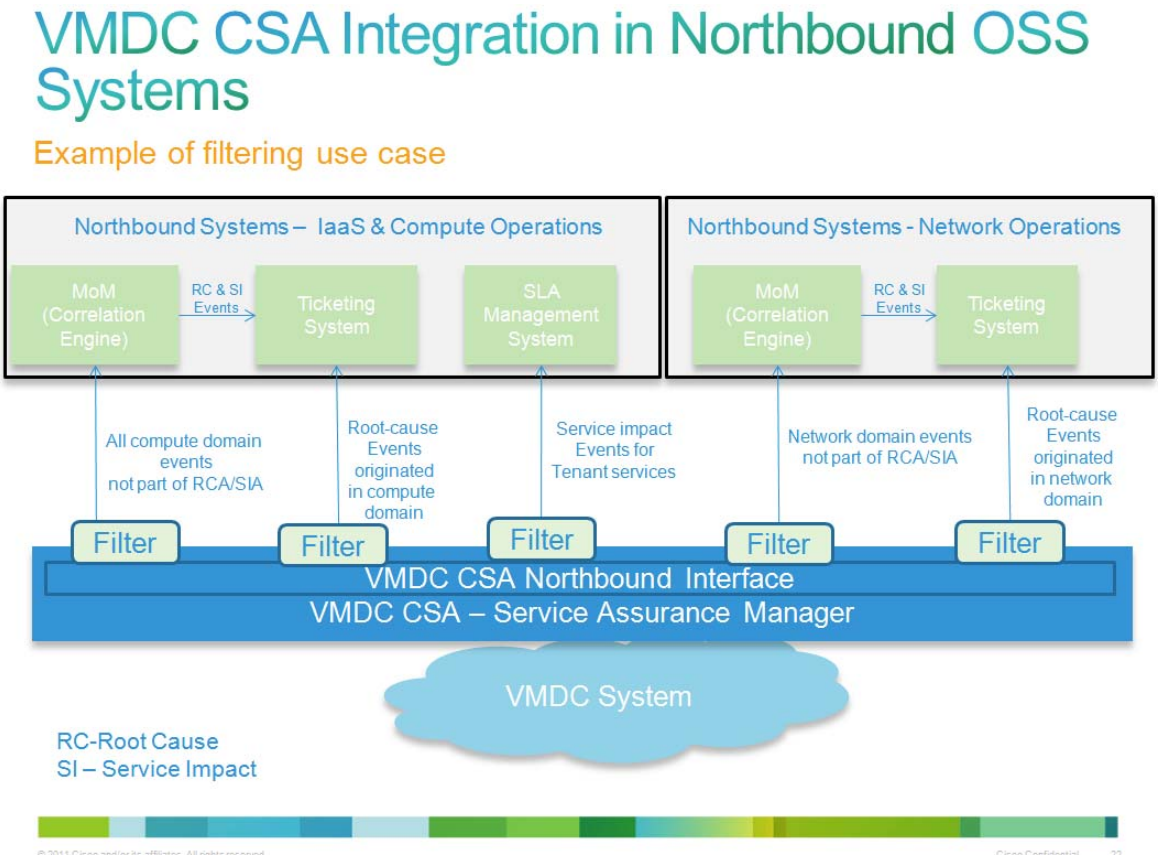
Figure 3-11 Single Normalized and Service Abstraction NBI



Integration With Multiple Northbound Systems

This use case example illustrates the need for different types of notifications and northbound filtering capabilities.

Figure 3-12 VMDC CSA Integration in Northbound OSS Systems



In this use case example (Figure 3-12), there is an IT department with two operations teams: one for network operations and one for server/compute operations. In addition, each one of the teams has a ticketing system and a MoM capable of further event processing and RCA. Assume also that the server/compute operations team has an SLA management system used to measure and manage SLA compliance.

Using the extensive filtering capabilities of the northbound notifications, the needs of both of these operations teams and their various northbound systems can be satisfied with a single instance of the service assurance system. In this example, five northbound notification destinations are configured, each with a different filter (also known as a notification trigger) as follows:

- All root cause events originated by vCenter or UCSM are sent to the Compute Operations ticketing system.
- All service-impact events originated by vCenter or UCSM are sent to the Compute Operations SLA management system.
- All other compute events that may require additional analysis are sent to the Compute Operations MoM.
- All root cause events originated by network devices are sent to the Network Operations ticketing system.

- All other compute events that may require additional analysis are sent to the Network Operations MoM.

Abstraction Through Service Overlays

This use case illustrates the need for service impact notifications from CLSA VMDC. This use case is a prerequisite for integrating CLSA VMDC into CLSA-HCS. To deliver HCS services (voice, voicemail, etc.) to the end customer/tenant, multiple services need to be provided to the customer, which are referred to as service overlays. In a scenario for top-level service such as HCS, there are a number of benefits to only processing abstracted events related to a few underlying services:

- Complexity of its fault management system can be reduced significantly if it is only receiving events related to few underlying services (IaaS, MPLS VPN WAN service, etc.) rather than having to deal with device level events from tens of underlying components.
- More flexibility to support various business and operational deployment models that vary in which domains and services are owned and operated by the provider offering top-level (e.g., HCS) services.

Figure 3-13 and Figure 3-14 illustrate the service overlay approach for application-based services such as HCS, and the need for service level abstraction from the underlying infrastructure system.

Figure 3-13 HCS Services and Operational Domains

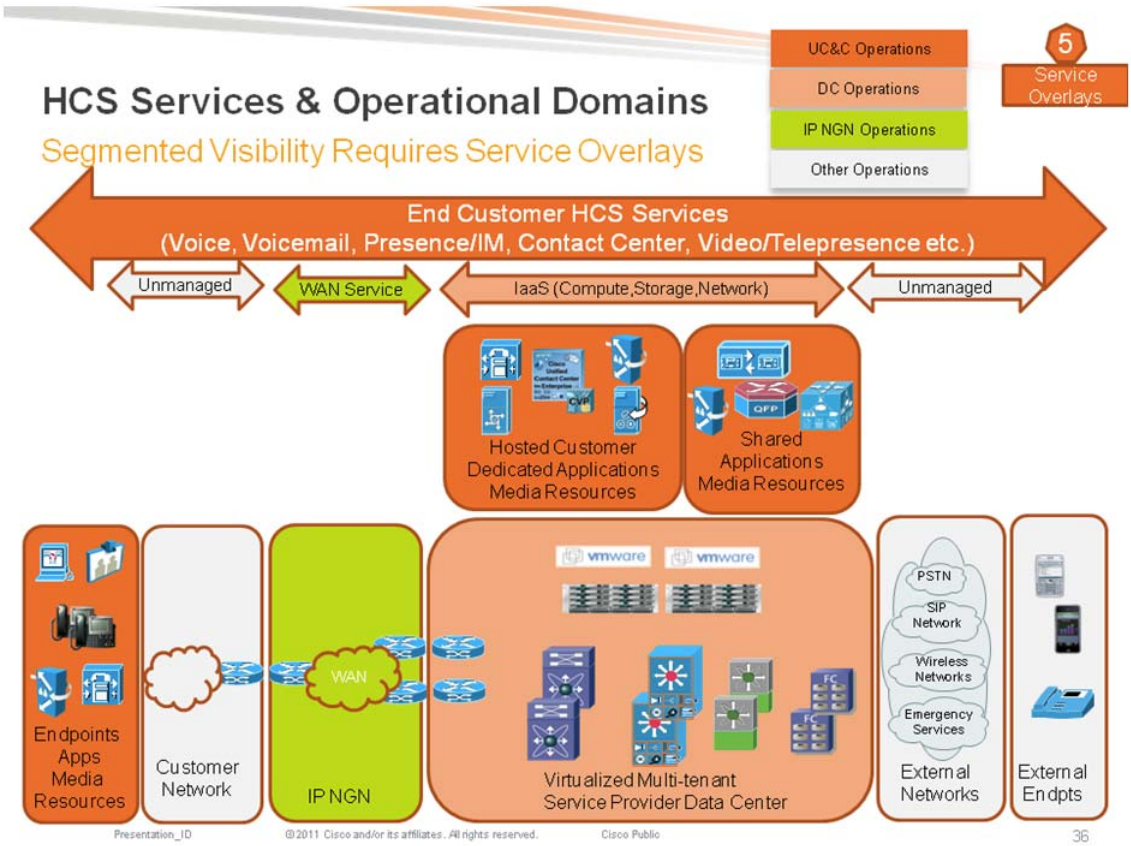
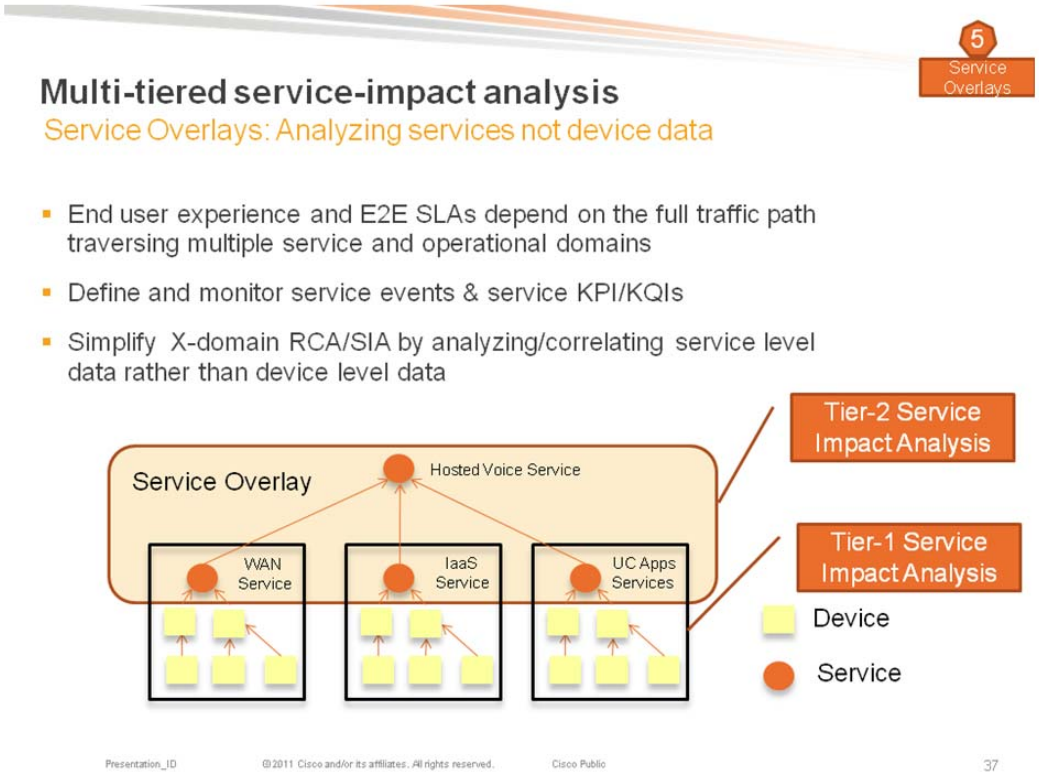


Figure 3-14 Multi-tiered SIA



Performance Management

The following performance management capabilities are provided out-of-the-box in CLSA VMDC:

- KPI statistics resource monitoring and trending:
  - Resource monitoring is partially validated as part of CLSA VMDC.
- Performance service impact models for compute and storage:
  - TCAs utilized as part of SIA
  - Validated as part of CLSA VMDC
- Application response time measurements:
  - Not validated as part of CLSA VMDC
  - For details, refer to product documentation on www.zenoss.com.
- Performance reporting:
  - Not validated as part of CLSA VMDC
  - For details, refer to product documentation on www.zenoss.com.

Dashboards

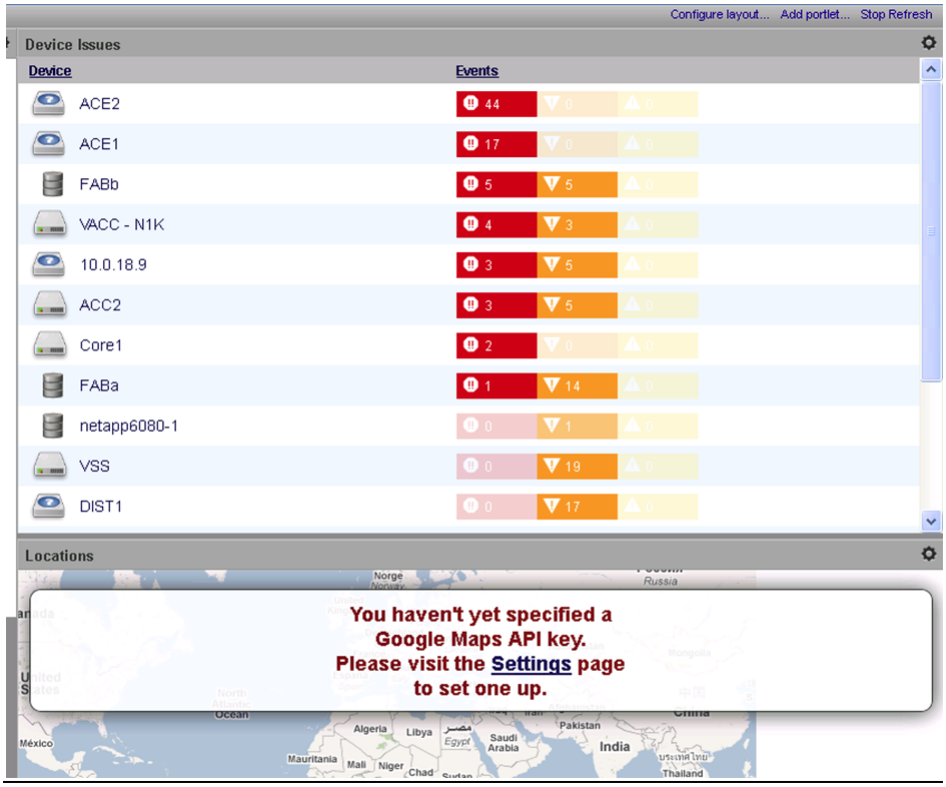
CLSA VMDC features aggregated SP dashboards, as well as both device level and service level dashboards that operators can use to obtain more details. The following are the key dashboard categories for Zenoss CSA:

- Aggregated systemwide resources status dashboards
- Service inventory and status dashboards
- Infrastructure/resource monitoring dashboards
- Event dashboards

Aggregated Systemwide Resources Status Dashboards

These dashboards list all devices with events systemwide, sorted by number of highest priority events, as shown in Figure 3-15.

Figure 3-15 Aggregated Systemwide Resources Status Dashboard



Service Inventory and Status Dashboards

These dashboards show the availability and performance state of all services in the system, as shown in Figure 3-16.

Figure 3-16 Service Inventory and Status Dashboard

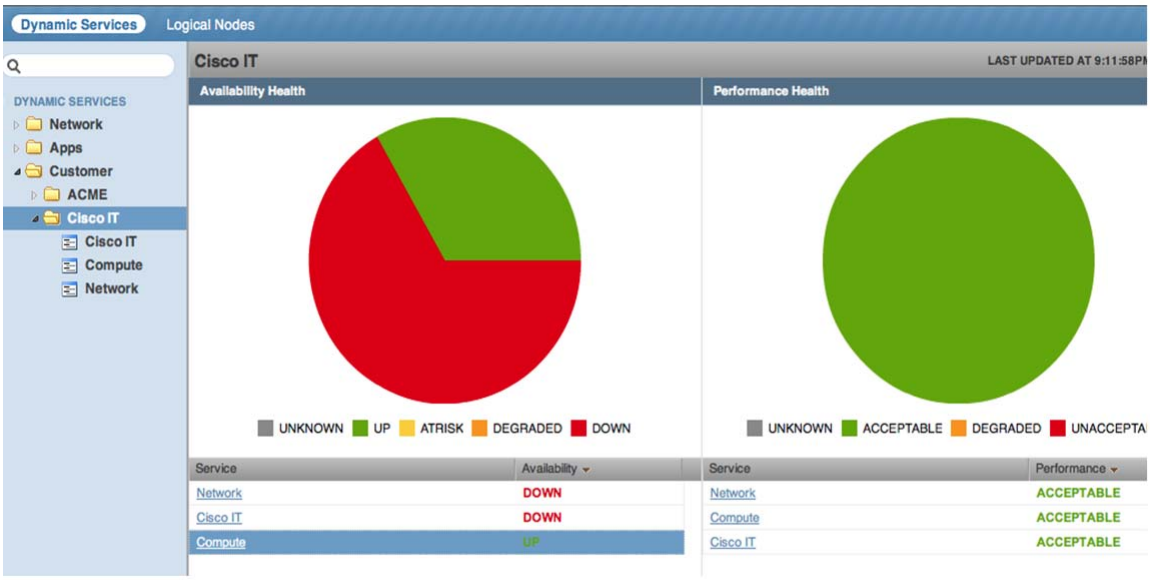
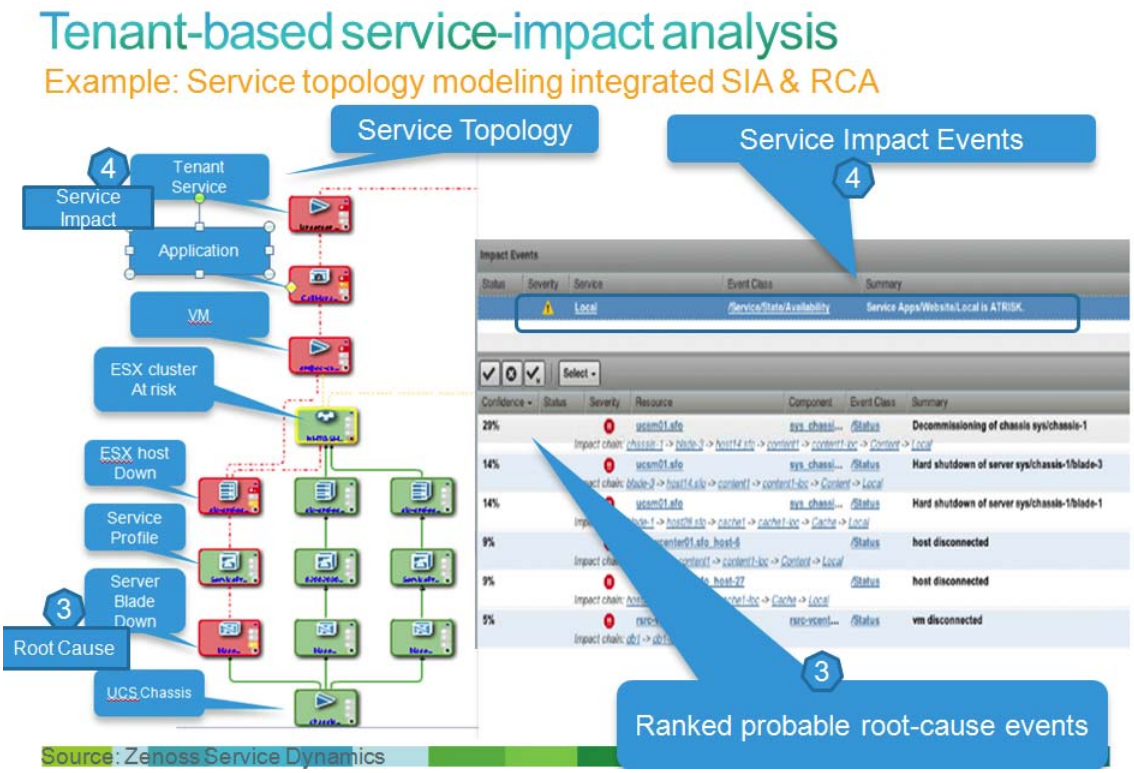


Figure 3-17 shows a per-service detailed dashboard, which lists service impact events and related probable root cause events, as well as a visualization of the service model tree.

Figure 3-17 Per-service Detailed Dashboard



Infrastructure/Resource Monitoring Dashboards

These dashboards list the inventory of all devices and their status, as shown in Figure 3-18.

Figure 3-18 Infrastructure Dashboard

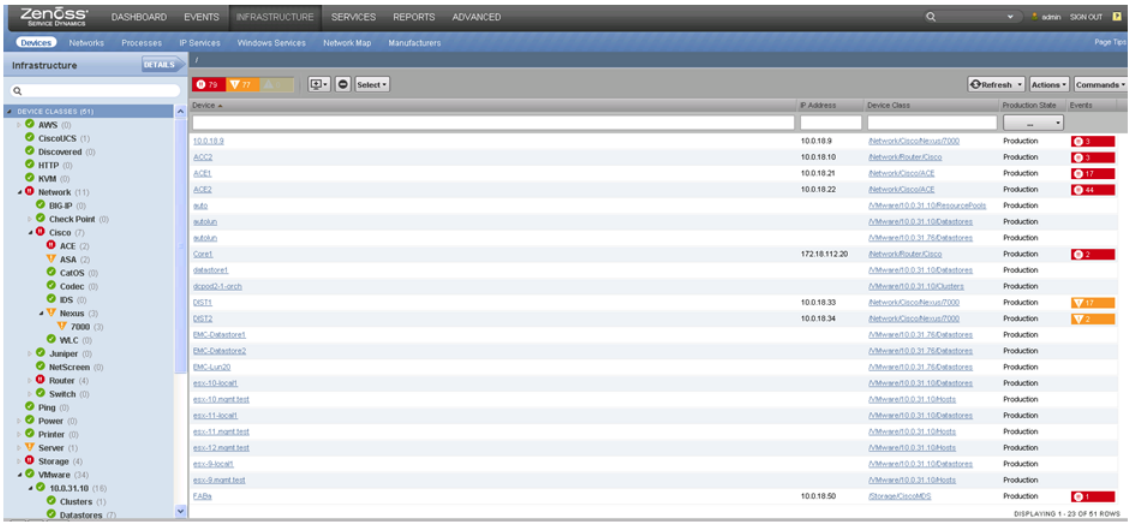


Figure 3-19 and Figure 3-20 show a detailed component dashboard and graphical view (example UCS server blade).

Figure 3-19 Detailed Component Dashboard

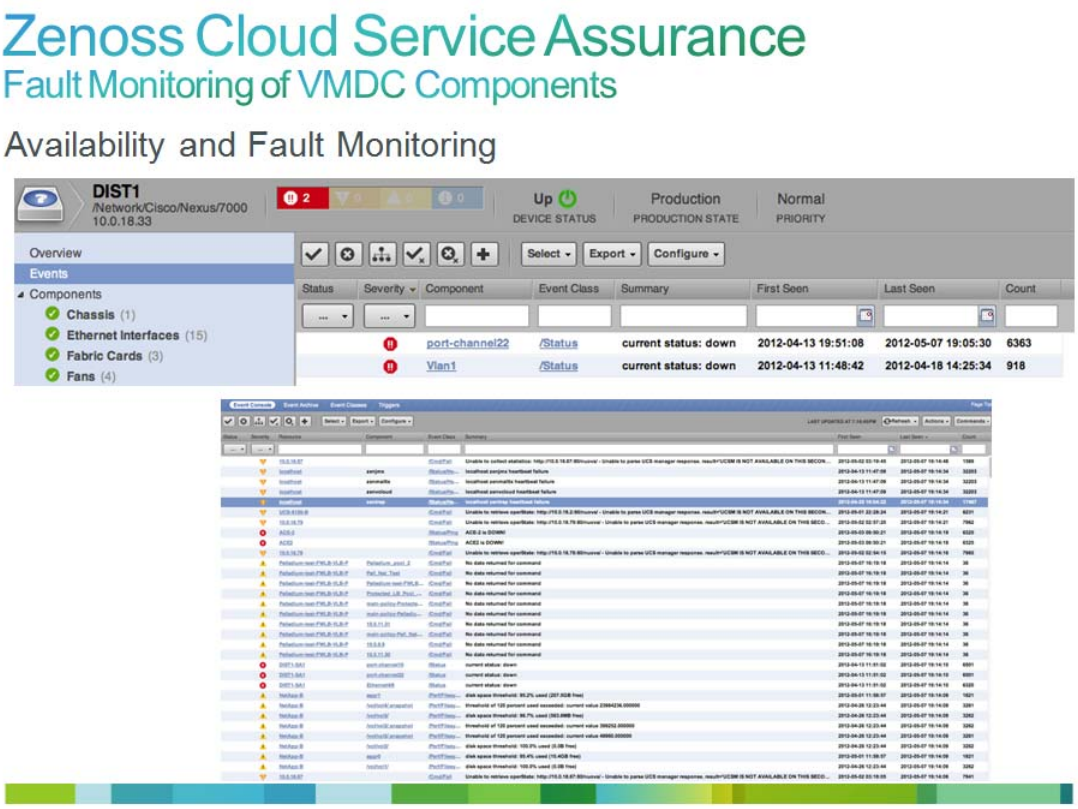
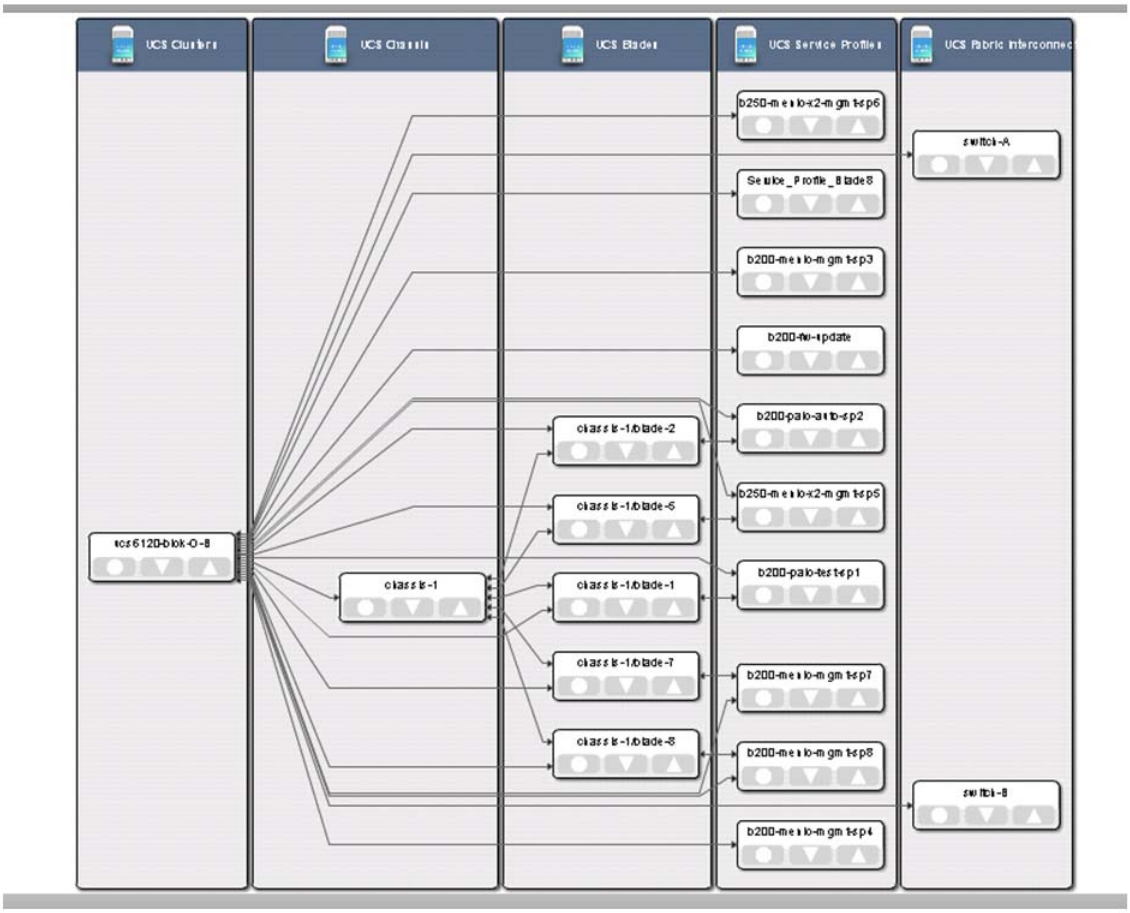


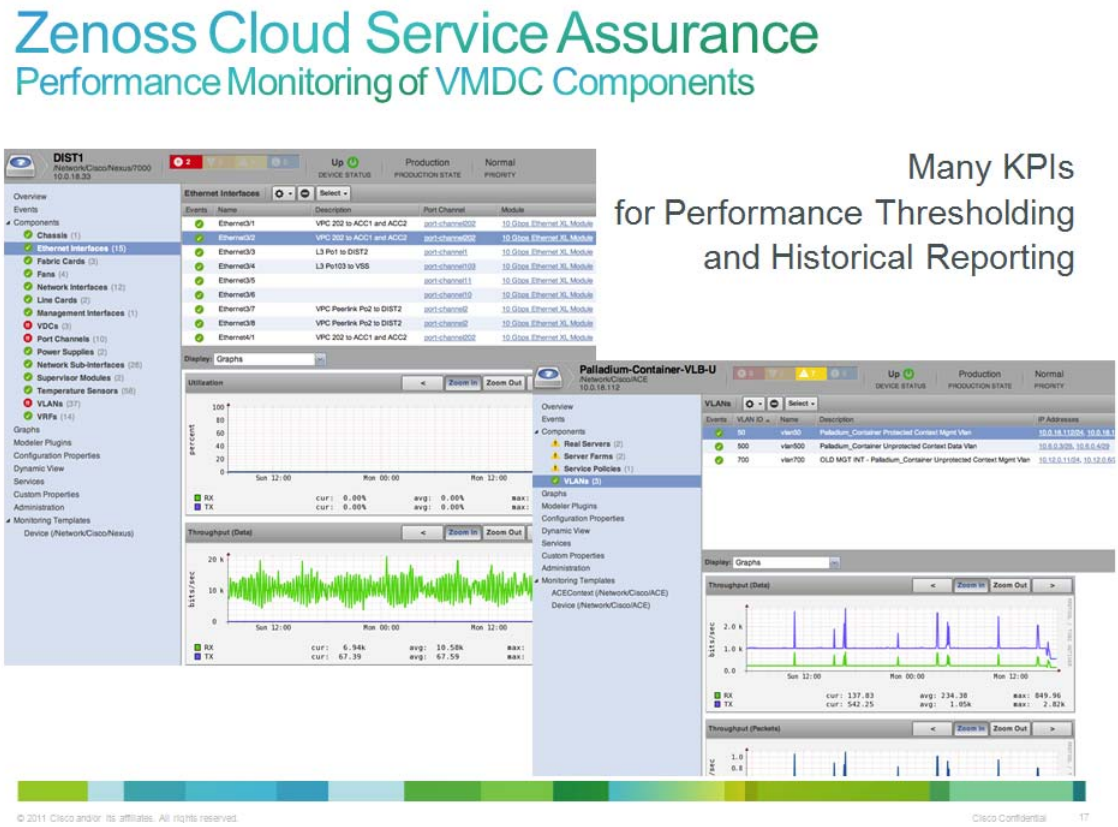
Figure 3-20 UCS Server Blade Graphical View



Event Dashboards

These dashboards show all events in the console (similar consoles exist per component as well), as shown in Figure 3-21.

Figure 3-21 Event Dashboard



Refer to Dashboards, page 3-30 for a use case example of dashboard monitoring.

Reporting

CLSA VMDC provides a range of defined and custom report options, including the following:

- Device Reports
- Event Reports
- Performance Reports
- Graph Reports
- Multi-Graph Reports
- Custom Device Reports

Reports can be exported to external files and systems or can be viewed locally. Reports can also be generated ad hoc or scheduled. Refer to the Zenoss Cloud Service Assurance Installation and Administration Guide for more information.

The following is a list of reports supported out-of-the-box for CLSA VMDC:

- Device Reports (9)
  - All Devices
  - All Monitored Components
  - Device Changes

  - MAC Addresses
  - Model Collection Age
  - New Devices
  - Ping Status Issues
  - SNMP Status Issues
  - Software Inventory
- Custom Device Reports
- Graph Reports
- Multi-graph Reports
- Event Reports (3)
  - All EventClasses
  - All EventMappings
  - All Heartbeats
- Performance Reports (7)
  - Aggregate Reports
  - Availability Report
  - CPU Utilization
  - Filesystem Util Report
  - Interface Utilization
  - Memory Utilization
  - Threshold Summary
- Storage (3)
  - Clients
  - Licenses
  - Disk Firmware
- Enterprise Reports (17)
  - Organizer Graphs
  - 95th Percentile
  - Defined Thresholds
  - Interface Volume
  - Network Topology
  - Customized Performance Templates
  - User Event Activity
  - Notifications and Triggers by Recipient
  - Datapoints by Collector
  - Organizer Availability
  - Maintenance Windows
  - Interface Utilization

  - Event Time to Resolution
  - Data Sources in Use
  - Users Group Membership
  - Cisco Inventory
  - Guest to Datapools
- MSExchange (1)
  - MSExchangeAvailability
- VMware (5)
  - ESXs
  - VMware Utilization
  - VMs
  - Datastores
  - Clusters
- Cisco UCS Reports (2)
  - Hardware Inventory
  - Free Slots

Multiservices

This section discusses the CLSA VMDC approach to multitenancy. VMDC architecture supports multitenant delivery, and CLSA VMDC must therefore support an assurance window into these tenant services to equip cloud providers with the ability to assure logically distinct customer services. A related topic, Role-Based Access Control (RBAC), is also presented in this section.

CLSA VMDC Multitenancy

VMDC provides a multitenancy cloud infrastructure by logically separating tenant services that are implemented on a shared physical infrastructure. Tenants consume a portion of network, storage, and compute resources that have been allocated from the larger pool represented by the cloud. CLSA VMDC delivers cloud provider assurance of shared infrastructure devices and their sub-components. In addition, CLSA VMDC supports the multitenancy aspect of the VMDC architecture through the use of defined Tenant Services.

Zenoss CSA enables an administrator to stitch together service element nodes which taken as a whole comprise a specific tenant service. A CLSA VMDC tenant service begins with the creation of the topmost element node named for the tenant. To this tenant node, underlying VMware vSphere and UCS shared infrastructure elements can be discovered and attached. Refer to VMDC Assurance Service Models, page 3-16 for more details regarding tenant services.

Using the tenant service modeling feature of CLSA VMDC, cloud customers' services can be assured independently. Elements of the tenant service that are unique to that tenant customer such as specific VM's are visible only to the cloud provider or the service owner. Elements of the service that belong to shared infrastructure, such as a UCS chassis or a storage device are visible across multiple tenant services, as would be expected. In fact, if a shared device experiences a fault condition, all services associated with that device should be impacted, however, any fault condition associated with unique elements of a tenant service are not visible to other tenants.

Note This phase of CLSA VMDC only supports cloud provider visibility into dashboards and service impact trees. Tenant customer visibility into service impact trees via customer portals will be supported in future releases.

RBAC Implementation

As a cloud providers' infrastructure increases in scale, it becomes important to provide a segmentation of operations capability to implement a division of responsibility. CLSA VMDC fulfills this need with its RBAC implementation. Beyond this division of responsibility capability, RBAC can also be used to support groups of users with limited visibility into specific tenant services. Table 3-5 lists the out-of-the-box roles that may be used to segment cloud provider operations responsibilities and access.

Table 3-5 Global User Role Definitions

Role	Definition
ZenUser	Provides global read-only access to system objects.
ZenManager	Provides global read-write access to system objects.
Manager	Provides global read-write access to system objects and read-write access to the Zope object database (which includes all devices, users, and event mappings)
ZenOperator	Provides users the ability to manage events, i.e. acknowledge, move to archive, etc.

These predefined roles are global in scope, such that an operator may access all cloud objects, but only be allowed certain operations. In addition to these global user roles, users may be defined that are more limited in scope. These user roles may be assigned to organizational groups to manage a subset of the entire infrastructure or even specific tenant services.

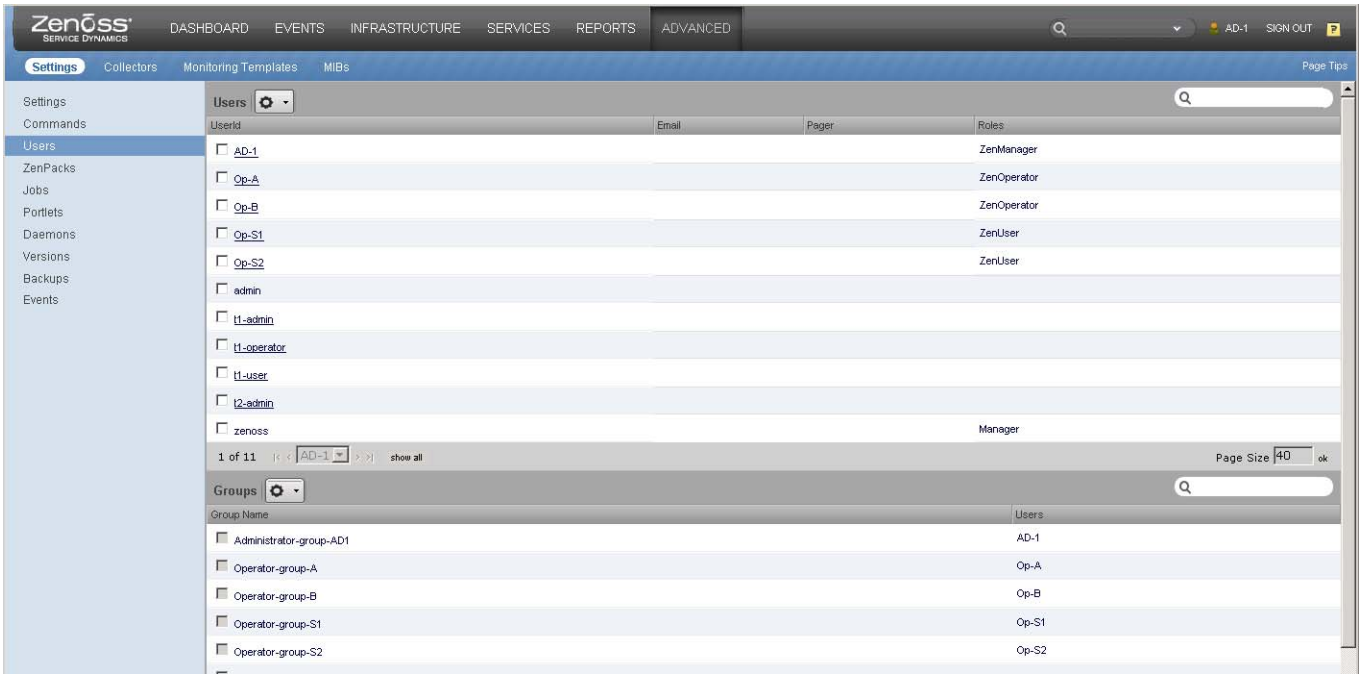
Organizational groups are used to collect subsets of infrastructure and/or services into logical categories for segmented operations. Table 3-6 lists the broad group categories and suggested uses for each group type. Groups can be devised for each tenant such that a customer's tenant services can be assigned to a tenant group.

Table 3-6 Device and Service Group Categories

Group Categories	Group Purpose
Group	Can be used for collecting similar devices, e.g. all switches group, all compute devices group
Systems	Can be used to collect all equipment with a specific data center, e.g. data center A, data center B
Locations	Can be used to collect devices by geographic boundaries, e.g. city, state, or even specific device rack

Figure 3-22 illustrates a list of users with either global or customized group role assignments. Users with the global user roles would belong to the cloud provider.

Figure 3-22 Custom User Groups



Users with non-global roles can belong to either the cloud provider or even a cloud customer. While RBAC is only supported for the cloud provider in this phase, it could be the mechanism to deliver a cloud customer, or multiservices, assurance portal in future phases.

CHAPTER 4

Zenoss Cloud Service Assurance Overview

Zenoss Cloud Service Assurance (CSA) is used as the Service Assurance Manager (SAM) for Cloud Service Assurance for Virtualized Multiservice Data Center (CLSA VMDC). Zenoss CSA consists of the following two elements:

- Core assurance platform
- VMDC ZenPacks that include VMDC specific device plugins that provide out-of-the-box support for VMDC components on the core assurance platform

This chapter discusses the product architecture and provides an overview of the capabilities of the core assurance platform and presents the following topics:

Zenoss Cloud Service Assurance Functional Overview

Zenoss CSA focuses on the assurance and optimization category of cloud operations. The console provides a unified view of assurance and operations. Cloud operations consoles meet a combination of Enterprise role-based security and Service Provider (SP) multiservice needs. Enterprises have traditionally defined multiple organizational roles limiting the actions a user in the role can perform. SPs have traditionally provided each customer a view of just their given resources as a paid service. In cloud-enabled organizations, both needs must be met simultaneously.

As far as customers are concerned, the ability to deliver against the service level they chose from the Service Catalog is paramount. Whether characterizing this mutual understanding as an expectation or a formal agreement, there is a need to track and report against the agreed to metrics in a manner that is easy to understand. This means providing separate views for each customer and workload, and operating as if there are multiple distinct tenants using common cloud resources.

To provide the service level metrics and meet the expectations, the workloads and the supporting infrastructure must be monitored to detect issues that may impact the customer. The Impact and Event and Management collects device, application, and infrastructure information that identifies potential issues and evaluates it to determine which issues are critical. Performance Monitoring collects vital performance statistics at every layer of the infrastructure, evaluates it to determine whether anything should be fed into the Impact Management process, and stores it for long term analytics. These two functions provide reactive management to application, device, and infrastructure issues.

For some issues such as a sudden failure of a power supply, there is no advanced warning. For other issues, there are trends that can be collectively understood and provide early warning of an impending issue. Predictive Analytics attempts to provide proactive analysis of data, searching for identification and remediation of issues before they reach a critical state.

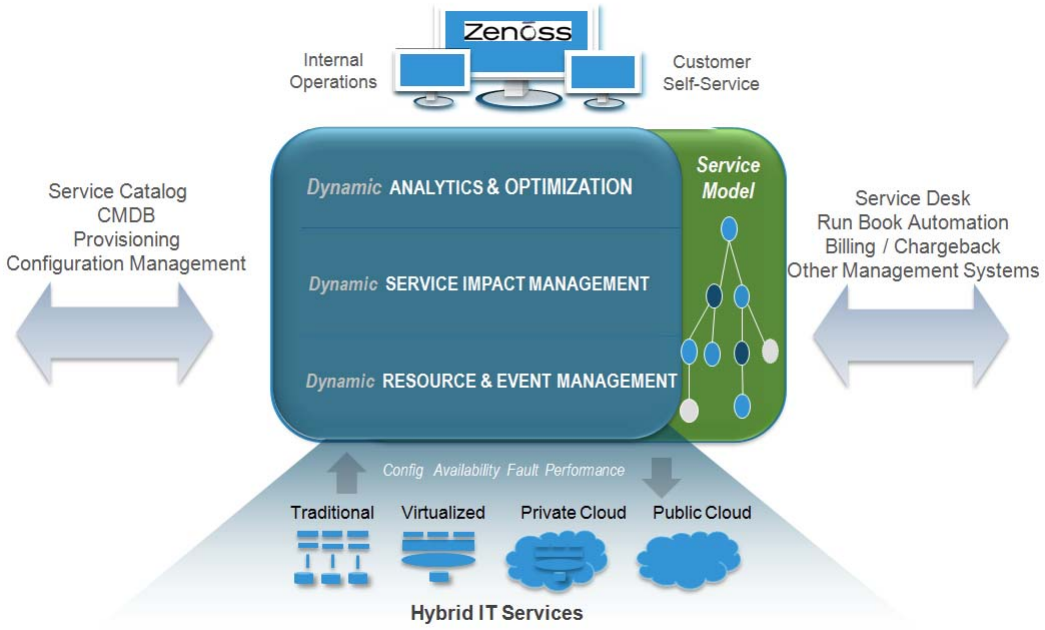
In cloud operations, the discipline of Capacity Analytics changes roles from an assessment of workload-driven resource requirements against the fixed amount available from a dedicated hardware platform. Reacting to the assessment can take weeks as new hardware is provisioned. Within the cloud Data Center (DC), resource allocations can be altered in minutes, and the reaction to changing workload needs should be just as fast. Administrators need to communicate to customers the needs of their workloads and get confirmation that they are willing to bear any increased costs, or allow the resource needs to go unmet and application performance to suffer. Administrators also need to provision enough resources for the overall DC to ensure that peak load commitments can be met, which means understanding capacity at an aggregated level.

Figure 4-1 shows the key functions that are available within Zenoss CSA. There are three key categories of the functions implemented in three software sub-components:

1. Resource Management, which provides discovery, data collection, performance, and event monitoring and notification capabilities.
2. Impact and Event Management, which provides full event management lifecycle, Root Cause Analysis (RCA), and Service Impact Analysis (SIA).
3. Analytics and Optimization, which provides a data warehouse for long term data trending, analytics engine, and reporting capabilities.

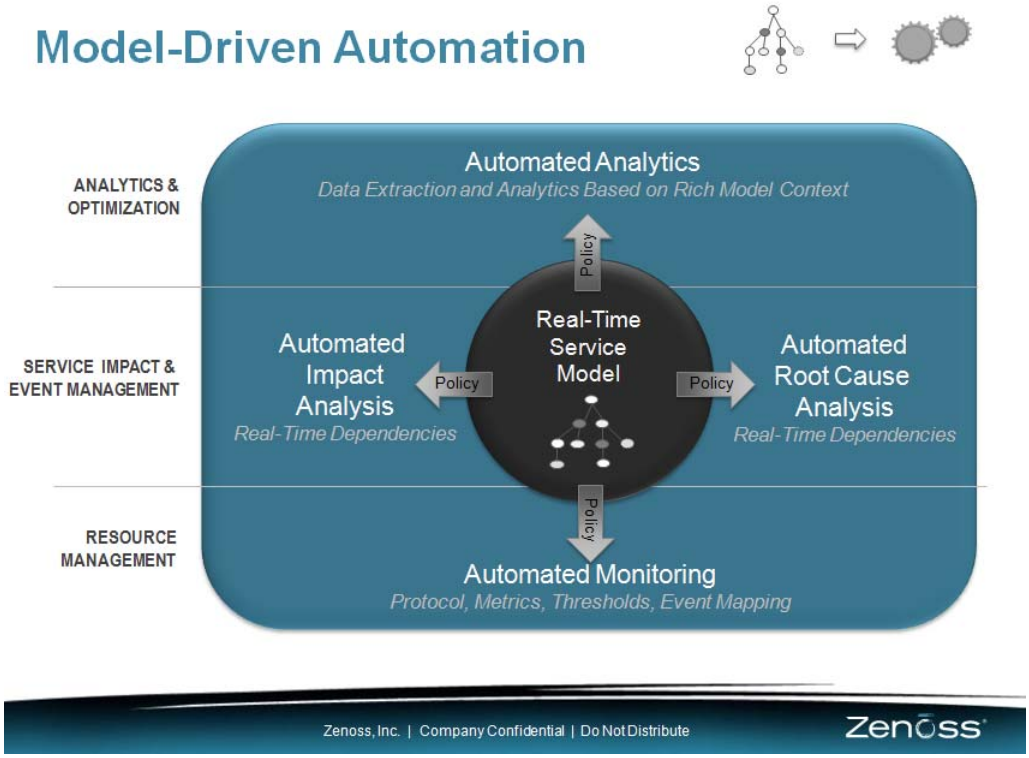
Note Resource Management and Impact and Event Management are the focus of CLSA VMDC. The Analytics and Optimization functions were only evaluated on a best effort basis, but are not fully customized nor validated for VMDC environments.

Figure 4-1 Key Functions of Zenoss CSA



The core issue in the cloud DC is keeping up with its rapid rate of change. When a customer's order for a new IT service is automatically fulfilled and placed into production, there is no time to manually update tools or components. At the heart of Zenoss CSA is a unified, real-time understanding of the entire IT environment, including resources, services, relationships, dependencies, state, and configuration. With this understanding, Zenoss is able to simplify the service assurance process with template-driven resource monitoring and automated impact and RCA. Unlike traditional systems that rely on configuration databases that are updated in batch mode, the Zenoss model is maintained in near real time through a series of discovery and modeling techniques that tap into the stream of configuration changes as they happen across the physical, virtual, and cloud-based infrastructure. As with every other aspect of the product, this model can be extended through its open API (Figure 4-2).

Figure 4-2 Model-Driven Automation



The following sections provide more details on the key functions of Zenoss CSA.

- Dynamic Resource Management, page 4-3
- Dynamic Impact and Event Management, page 4-4
- Dynamic Analytics and Optimization, page 4-6

Dynamic Resource Management

The foundation of service assurance in the hybrid cloud DC is unified, cross-domain resource monitoring and control that brings together configuration, performance, availability, fault, event and log information across the physical, virtual and cloud-based infrastructure and applications, and enables automated actions to be performed at the resource level. Zenoss CSA delivers this capability on a scalable, open platform that is easy to extend, and is able to track dynamic elements and relationships as they evolve in near real time.

Key feature areas include the following:

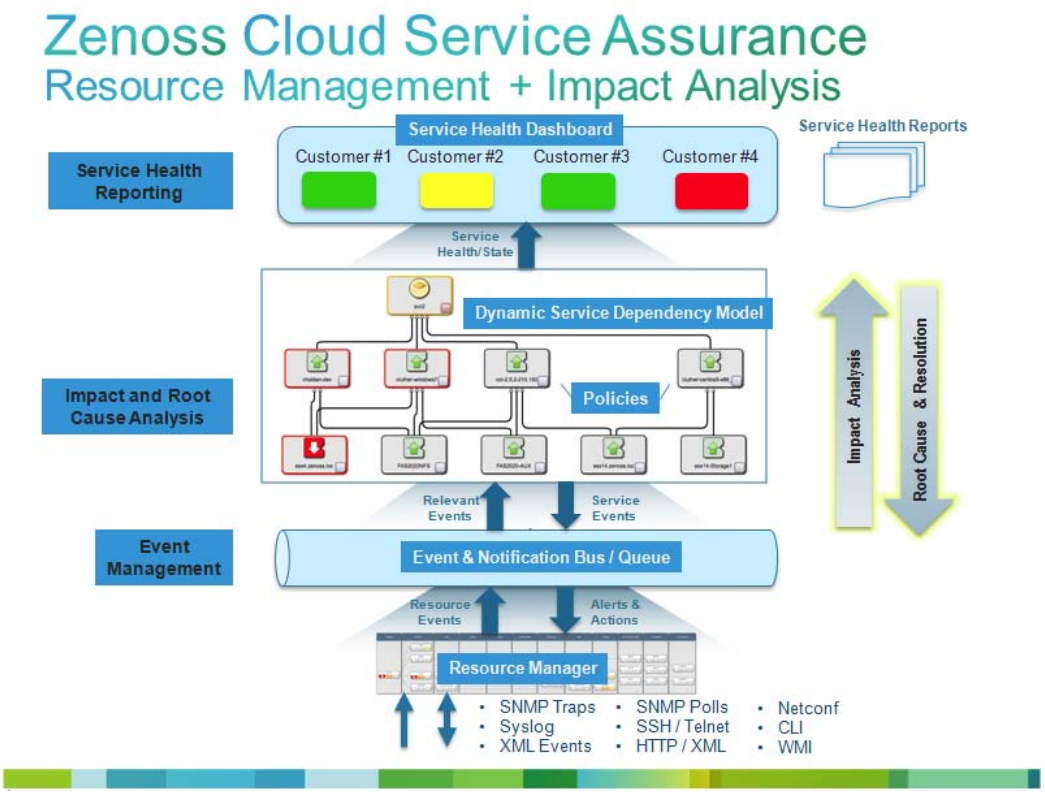
- **Discovery and Modeling**. Automatically maintain real-time inventory and configuration details for the entire IT environment; includes real-time relationship tracking of dynamic relationships common in virtualized and cloud-based infrastructures.
- **Full-stack Monitoring**. Unify and automate performance, availability, and event monitoring of networks, servers, storage and applications across physical, virtual, and cloud-based environments with a single, model-driven, horizontally-scalable, extensible collection platform.
- **Notification and Control**. Rich alerting and remediation framework allows the user to be notified via email, text, or pager based on user-specified policies, or to take direct automated action to address a problem in real time.

Dynamic Impact and Event Management

Maintaining a real-time perspective on service health, and linking health issues to the underlying infrastructure in a reliable, simple, and cost-effective way can be challenging in hybrid data centers. In particular, legacy approaches to impact management and RCA simply break down due to the shared and dynamic nature of virtualized and cloud-based infrastructures.

Figure 4-3 illustrates how Zenoss CSA transforms state information from the resource manager into a stream of events that are processed in near real time by its SIA and RCA engine, leveraging the real-time service model. This processing feeds a service health dashboard and generates service events that are used for alerting, troubleshooting, and to initiate real-time automation and service remediation. The result is real-time service level awareness, rapid triage, and closed-loop automation that thrives in the dynamic, hybrid cloud environment.

Figure 4-3 Service Health Dashboard



Dynamic Service Modeling

Zenoss CSA maintains a real-time service model, and automatically discovers infrastructure dependencies. Service constructs can be easily defined based on logical business constructs to define the infrastructure groupings supporting a specific application service. For example, a Customer Relationship Management (CRM) application service might require e-mail, web, and database services to be present in order to operate. These logical definitions define the collection of services required for the CRM service to be considered functional. Once this logical service hierarchy is defined, the application or OS instances delivering the service functions are associated and Zenoss CSA takes care of the rest. Using advanced dependency modeling capabilities, Zenoss CSA pulls in all relevant infrastructure elements. Virtual Machine (VM) partitions, blades, chassis, storage, network interfaces, and a wide variety of device components are all automatically discovered and mapped into the relevant service dependency graphs.

Dynamic Impact Analysis

Dynamic Impact Analysis identifies which services are affected by conditions in supporting components or infrastructure. For example, a failing fan in a Cisco UCS chassis might result in dozens of virtual machines being moved to new virtual hosts. With Zenoss impact analysis, IT operations can determine which business services will be affected by the fan failure and can plan corrective actions to minimize service level disruptions.

Dynamic RCA

Dynamic RCA allows IT operators to quickly identify the specific events most likely to be the cause of a service impacting condition. In complex IT environments, it is not uncommon for a single component failure to cause a cascade of failures, resulting in an event storm totaling thousands of individual events.

Zenoss CSA includes a proprietary Confidence Ranking Engine built on top of Impact Analysis to quickly triage these events and identify where IT resources should be applied to correct these types of situations. This algorithm filters impact events based on a variety of criteria including severity of the event, service graph depth, and the number of graph branches affected by an event. This ranking algorithm allows IT operators to target resources to address events deemed the most likely cause of a service failure or degradation. Real world deployments of Zenoss CSA have validated the effectiveness of the service impact framework by demonstrating significant event reduction and highly accurate identification of root cause events.

Simple, Modular Policy - "Policy Gates"

Traditional impact managers require complex, top down rule sets to be defined, which require either a static IT infrastructure or a detailed understanding of all possible infrastructure configurations to identify service impact or determine root cause. This approach fails in dynamic virtualized or cloud data centers due to the need to maintain hard dependencies on named infrastructure elements. In contrast, Zenoss CSA uses Policy Gates to define impact rules on an element-by-element basis, and rolls up impact results through the current service model to reach conclusions. The design premise of this system is that the state of any given element in a service graph is determined by analyzing the state of the immediate children of that element. A change in the state of a given element is propagated to the parents of that element, causing the parents to evaluate their own state using their own Policy Gate configurations. The net result of this approach is that functions such as event aggregation, filtering, de-duplication and masking are provided automatically, eliminating the need for highly specialized skills to write impact rules and dramatically reducing human effort in event processing.

Unified, Scale Event Management

Aggregate and manage events for an entire IT environment with a next generation event management system that provides automated event normalization and enrichment, and is easily extended, integrated, and scaled through an embedded message bus. Zenoss CSA is capable of processing in excess of 1,500 events per second with a single event processor. Field deployments have shown that the system is capable of quickly parsing through event storms scaling to thousands of events in seconds, resulting in just a handful of events after processing through the Service Impact and Confidence Ranking.

Dynamic Analytics and Optimization

The final step of the service assurance lifecycle is historical analysis and planning. Deep, cross-domain analytics are needed to perform capacity planning and drive optimization of the environment. Zenoss CSA enables this through its integrated analytics capabilities that directly leverage the real-time service model and all state information from the Resource, Impact, and Event Management modules. Leveraging a powerful, open business intelligence engine, the Zenoss analytics capability provides a scalable and rich analytics platform that addresses tenant reporting needs, management dashboards, capacity planning, and insight for optimization. Specific capabilities include:

- **Turnkey Operations Data Warehouse**. Automatically aggregates and normalizes configuration, performance, and event history for the entire IT environment across physical, virtual, and cloud-based infrastructure and applications.
- **Unified Historical Analytics**. Understand utilization and health trends across an IT's entire infrastructure including tenant-based consumption and availability reporting; gain deep, timely insight through drag-and-drop dashboards, out-of-the-box reports, and powerful ad-hoc analytics all available through a multiservice web portal. The Zenoss Analytics software module is not validated or included as part of CLSA VMDC, however, this software module can be obtained directly from Zenoss, Inc.

- **Predictive Analytics**. Forecast capacity needs and anticipate availability problems through predictive trending that allows for visualization and proactive management of upcoming operational issues and infrastructure requirements

Zenoss Cloud Service Assurance Architecture Highlights

This section highlights the key aspects of the Zenoss CSA architecture that distinguish it from other platforms. Some of the key architecture characteristics are listed below.

- **Unified Design**. End-to-end service assurance and analytics capability designed from the ground up as one product on a common architecture.
- **Horizontal Scaling**. Scale the deployment to manage hundreds of nodes from a single server to 100K nodes in a globally distributed configuration, leveraging low-cost hardware. Scale as needed to manage elastic infrastructure.
- **Agentless, Multi-Protocol**. Agentless collection and control platform that leverages a suite of secure access methods, management APIs, and synthetic transactions to instrument the full stack at scale without the need for proprietary agents.
- **Open Integration and Extensibility Framework**. Rapidly extend, customize, and integrate with other management tools, leveraging open architecture and "ZenPack" plug-in framework. Leverage a global community of extension developers and partners.
- **Integrated RCA and SIA**. RCA is performed as a side product of the SIA, as opposed to traditional systems, which perform two functions using two different sets of rules, models, or even products. This simplifies development of customizations, as well as provides direct relationship between root cause events and service impact events caused by the root cause events.

Figure 4-4 and Figure 4-5 highlight the key aspects of the Zenoss CSA architecture.

Figure 4-4 High-Level Architecture

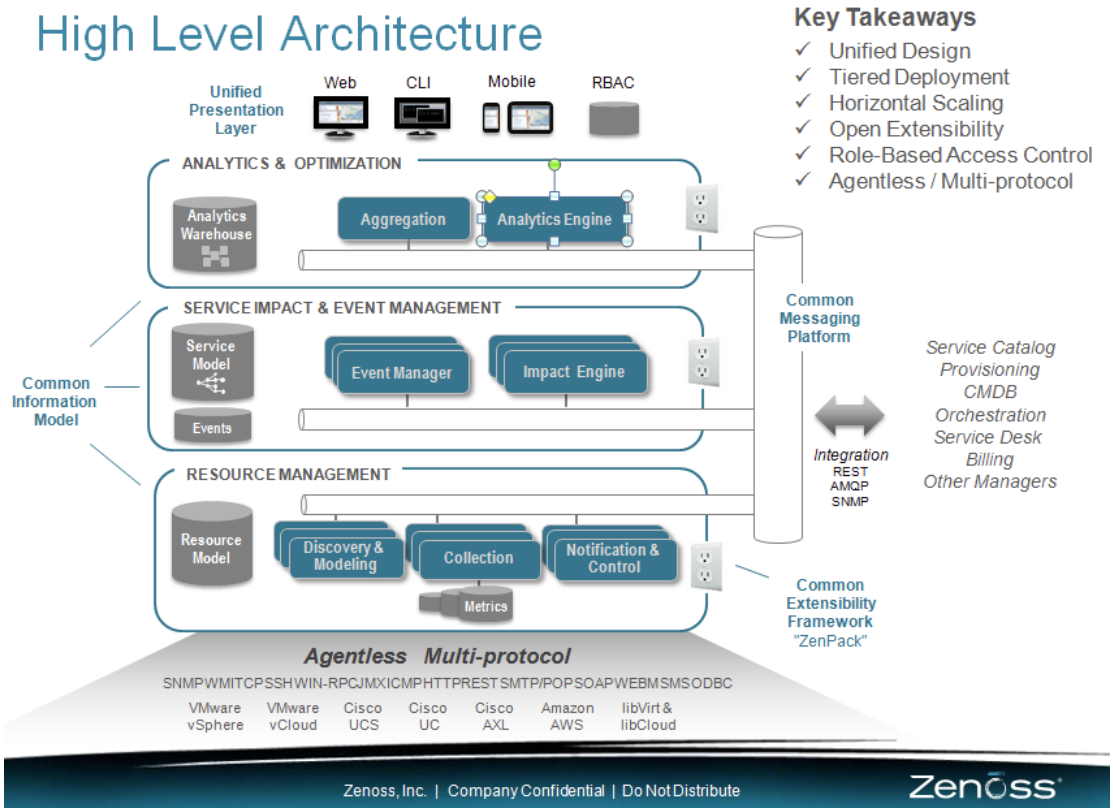


Figure 4-5 Zenoss Product Architecture Overview