# Prime Service Catalog OpenShift Integration 1.0 Design and Implementation Guide

May 23, 2014

# About Cisco Validated Design (CVD) Program

The CVD program consists of systems and solutions designed, tested, and documented to facilitate faster, more reliable, and more predictable customer deployments. For more information visit http://www.cisco.com/go/designzone.

Prime Service Catalog OpenShift Integration 1.0 Design and Implementation Guide

# CONTENTS

# Preface

This document addresses the installation and configuration of Red Hat OpenShift. Installation of the Cisco PSC and Cisco UCS are addressed in other documents. The installation and configuration of Red Hat OpenShift in this document is defined specifically to create a solution for the ordering, provisioning and management of OpenShift services through the Cisco PSC. Users should be qualified in the installation and configuration of Red Hat OpenShift, Cisco PSC, and the Cisco Unified Computing System (UCS).

## Document Objective and Scope

The two most critical problems facing IT departments today are the continuing costs of supporting legacy applications, and the need to be more responsive to service demands on a tight budget.

This document recognizes these two critical issues, and provides a solution that addresses the requirements for both with a single infrastructure for ordering, provisioning, and management of services. This design guide documents a solution providing OpenShift PaaS services through the Cisco PSC.

This design guide provides a comprehensive explanation of procedures required to provide Red Hat OpenShift services through the Cisco Prime Services Catalog. Details of system design and architecture to support the services are explained, including the deployment models for services, and guidelines for implementation and configuration. It is recommended that best practices be followed when deploying Red Hat OpenShift services through the Cisco Prime Services Catalog.

This design and implementation guide provides a comprehensive first step to delivering IaaS and PaaS services through the Cisco PSC.

## Business Dilemma

IT departments are facing two critical problems today:

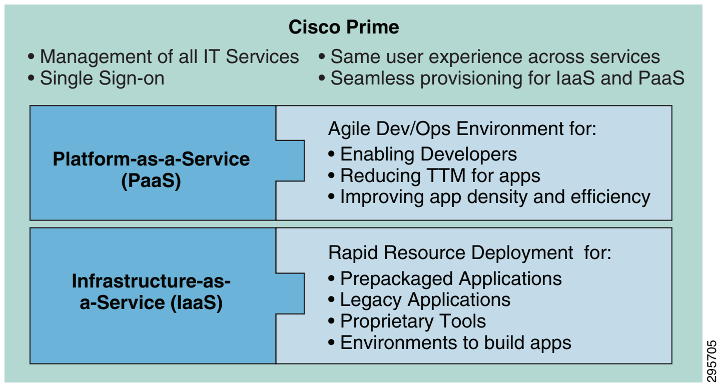
- **Legacy Applications/Server Sprawl**—This is the cost of inefficiency when legacy applications continue to run in a dedicated environment. Servers are running with low utilization, and expanding access to the application requires increasing the number of servers.

- **Increasing Velocity for IT Services and Changes**—The demand for changes to IT services is accelerating at a pace never before experienced in the history of computing while the IT budget is stagnant at best, and, in some cases, shrinking.

IT managers need to solve these two critical issues, while also reaching the objectives for high availability for existing day-to-day operations.

- Solving the IT dilemma requires a strategy that addresses the migration of legacy applications and the tools required to meet the increasing IT demands. A single, comprehensive solution for IaaS, PaaS, and IT services is required.
- The IaaS strategy provides the environment for rapid resource deployment. Virtualizing legacy applications lowers server costs, and controls server sprawl. However, IaaS alone cannot solve the problem. IaaS provides a container, which are useful for applications, tools for managing applications, or an environment for building services. PaaS provides the environment and development tools for building new services and expanding existing services.
- PaaS is a tool kit, not a complete solution. To be effective, PaaS needs to be deployed and managed because without effective management, this tool kit can experience the same issues as other applications running in dedicated environments. Solving the IT dilemma (Figure 3-1) requires an environment that provisions, configures, and manages both IaaS and PaaS services through a single pane-of-glass; provides the same user experience for managing both IaaS and PaaS; and seamlessly and dynamically provides IaaS services for instantiation and expansion of PaaS services.

***Figure 3-1 Solving the IT Dilemma***


Cisco Prime
• Management of all IT Services
• Single Sign-on
• Same user experience across services
• Seamless provisioning for IaaS and PaaS
Platform-as-a-Service (PaaS)
Agile Dev/Ops Environment for:
• Enabling Developers
• Reducing TTM for apps
• Improving app density and efficiency
Infrastructure-as-a-Service (IaaS)
Rapid Resource Deployment for:
• Prepackaged Applications
• Legacy Applications
• Proprietary Tools
• Environments to build apps
295705

The solution presented in this document offers IT departments a dual technology approach to solving the IT dilemma.

- PaaS services are provided for the migration of legacy applications to the cloud environment, reducing costs, controlling server sprawl, and improving efficiency for legacy applications.
- PaaS services are provided as the tool kit to turbo-charge the DevOps world. The value of PaaS is derived not from replacing existing tools, but by offering enhanced capabilities in tools already familiar to the developer.
- Businesses that solve today's IT dilemma will meet the challenges, and excel at satisfying their customers' needs.
- Businesses continuing on the legacy IT path will find it more difficult and expensive to compete.

CHAPTER 1

# Introduction

Cisco's integration of OpenShift 1.0 (by Red Hat) and the Cisco Prime Service Catalog (PSC) creates a solution to provision, configure, and manage Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), Software-as-a-Service (SaaS), and other IT services all through a single pane-of-glass. The combined solution provides a unified portal and service catalog for XaaS (anything-as-a-Service).

The benefits of OpenShift and Cisco Prime extend far beyond the integrated portal/service catalog. All IT services can be managed through the Cisco PSC. When the user signs onto Cisco PSC, he or she can provision, configure, and manage all services in the service catalog without switching between management systems and without signing into each system individually.

The Cisco PSC abstracts the interface to the individual management domains. The user completes the provisioning, configuration, and management of services through Cisco PSC, providing the user with the same interface and the same experience for all of the IT services.

Provisioning of PaaS services through the Cisco PSC is a one-step process done through a menu-driven GUI instead of an arcane command line interface. The Cisco PSC assigns and provisions the IaaS resources when the PaaS service is provisioned. After the IaaS resource exhaustion threshold limit is reached, additional resources are seamlessly added, allowing the solution to scale out without operator intervention.

The Cisco PSC with OpenShift leverages all of the existing strengths of Cisco Prime products in managing IaaS for supporting legacy applications while providing one-touch configuration for PaaS.

For the user, Cisco PSC provides:

- A single-pane-of-glass for provisioning, configuration, and management of IaaS, PaaS, and other IT services.
- Single sign-on for IaaS, PaaS, and other IT services. No need for log on to multiple systems.
- Seamless dynamic provisioning of IaaS resources for PaaS. When the PaaS application is created, IaaS resources are configured automatically. When the resource exhaustion threshold is reached, resources are added dynamically without user intervention.
- Network QoS/Security extended for PaaS. The benefits of the Cisco VLAN technology for network QoS/security are extended for PaaS and the services developed by PaaS.
- Cisco Prime provisions, configures, and manages OpenShift PaaS, thereby reducing deployment time, eliminating configuration errors, and most importantly reducing development costs and time to deployment for new applications.
- Finally, and most importantly, Cisco PSC with OpenShift solves the biggest dilemma confronted by enterprise IT departments. By offering the best-of-breed services for IaaS and PaaS, Cisco PSC with OpenShift addresses the migration of legacy applications to the cloud, and provides the advanced tools required for rapid application development.

IaaS services from the Cisco PSC support the migration of legacy applications to the cloud, gaining control of application management, and at the same time reducing server sprawl and costs.

PaaS services from OpenShift provide the ideal set of tools for rapid application development and deployment in the cloud.

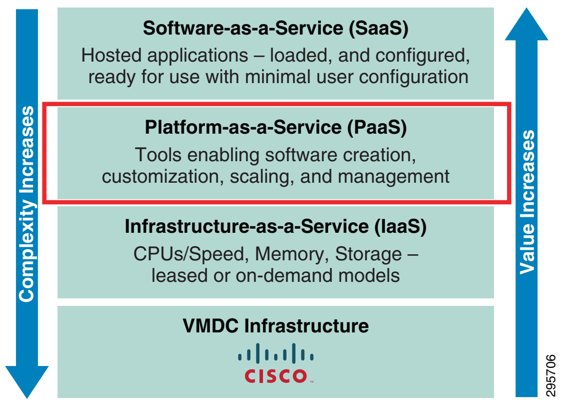
Together, the integrated solution solves the most pressing problems facing IT departments today.

## PaaS Services

IaaS has led the way for IT organizations to begin the process of migrating applications from individual servers to the cloud. IaaS has been effective in centralizing servers, reducing costs, and controlling server sprawl.

With the success of IaaS, IT departments are under great pressure with the business demand for greater access to larger amounts of information increasing geometrically. The pressure to reduce the growth in IT budgets is also increasing. Meeting the geometric increases in demands requires better tools than today's linear development environment can provide. Figure 1-1 shows PaaS service layers.

***Figure 1-1 Service Layers: PaaS Perspective***


Software-as-a-Service (SaaS)
Hosted applications – loaded, and configured, ready for use with minimal user configuration
Platform-as-a-Service (PaaS)
Tools enabling software creation, customization, scaling, and management
Infrastructure-as-a-Service (IaaS)
CPUs/Speed, Memory, Storage – leased or on-demand models
VMDC Infrastructure
Complexity Increases
Value Increases
295706

To understand the benefits of Platform-as-a-Service (PaaS) as part of a cloud infrastructure, it is necessary to understand how PaaS fits into the cloud stack. As emphasized in Figure 1-1, most cloud services are delivered through IaaS, SaaS, or PaaS.

Each of these layers offers a different degree of automation, and value, in the cloud stack. For IaaS, the consumer of the service takes on all of the management above the virtualization of the hardware. The service consumer installs the operating system, the applications, and is responsible for any dependencies for the O/S or application, and any middleware required.

At the opposite end of the stack, SaaS offers the service consumer the lowest management burden, but also the least flexibility. The application is an out-of-the-box commodity, with the service consumer having little to no opportunity to customize. While this works well for standardized applications, most businesses require some level of customization.

PaaS targets the cloud consumers that need the flexibility to configure and develop applications, but removes the burden of managing the lower layer components of the stack. The platform management tasks are part of the platform, freeing developers and administrators to focus on the design and delivery of the application.

While PaaS resides between the IaaS layer and the SaaS layer, the service consumer views PaaS-developed services as just more consumable services. PaaS functions delivered through the Cisco PSC allows end users to order complete application or platform stacks from the Cisco self-service portal. The application blueprint provides end users the ability to order fully configured, multi-tiered cloud applications using standards and automation. This approach bridges the gap between developers and operations and facilitates the collaborative deployment process needed to achieve the goals of DevOps.

More than a development environment, PaaS combines the capabilities of the platform at multiple layers to enable services such as auto-scaling and load balancing, relieving the developer of recreating the platform services for each application.

The PaaS layer empowers the developer with a rich set of tools, leveraging the most important benefits cloud technology offers.

Some of the benefits PaaS provides include:

- Application-centric (vs. VM-centric) management framework.
- Developers focused on application development, not infrastructure management.
- Application as the unit of deployment and management while the infrastructure is transparent.
- Requirements of the development team for app tools and the operations team for app management are satisfied.
- Bottleneck in provisioning and deployment is eliminated
- Codifies the relationship between developers, IT, and globally-distributed clouds.

The benefits PaaS provides are rapidly driving PaaS growth. Without PaaS, many of the services enterprises require at the SaaS level will exceed the limited capacity of the IT resources and budget.

## Cisco Prime-PaaS Deployment

Cisco Prime products address the complete experience lifecycle from service design through fulfillment, assurance, analysis, and optimization. The Cisco Prime product architecture is a pre-integrated management application suite, incorporating a self-populating common inventory model, based on industry standards. The data model for Cisco Prime products abstracts network devices and services to provide powerful experience management capabilities and extends coverage from the service provider core network to the customer premises.

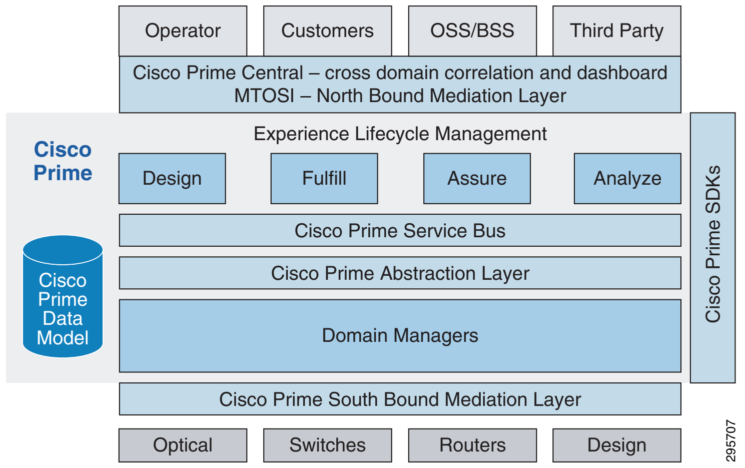
The Cisco Prime product architecture provides a comprehensive management solution to automate the design, fulfillment, assurance, and ongoing management of advanced network services such as video, mobility, and managed cloud services over IP networks. It enables repeatable, policy-driven service provisioning processes within standardized work flows and templates, allowing support personnel without specialized networking knowledge to easily provision, modify, diagnose, and repair complex services. Ultimately, it helps service providers provision services more quickly and consistently, at a lower cost.

The Cisco Prime product architecture is also designed to address complex operational challenges such as pre-population of end-to-end inventory in management systems and cross-domain fault management and troubleshooting. It provides a unified, consistent, and end-to-end view of network services, as well as cohesive work flows for common tasks that extend across multiple domains. As a result, service providers can diagnose, and correct, faults that span multiple domains more rapidly.

Finally, the Cisco Prime product architecture allows service providers to deliver the highly reliable, uninterrupted services their customers expect. Cisco Prime product solutions can be deployed to meet even the most demanding high-availability requirements. This includes both localized high-availability failover, as well as options for geographic disaster recovery and offloading.

The Cisco Prime product architecture delivers all of these benefits through a flexible, end-to-end framework of integrated Cisco Prime product suites, as shown in Figure 1-2.

***Figure 1-2 Cisco Prime Product Architecture***


Operator
Customers
OSS/BSS
Third Party
Cisco Prime Central – cross domain correlation and dashboard
MTOSI – North Bound Mediation Layer
Cisco Prime
Experience Lifecycle Management
Design
Fulfill
Assure
Analyze
Cisco Prime Service Bus
Cisco Prime Abstraction Layer
Cisco Prime Data Model
Domain Managers
Cisco Prime SDKs
Cisco Prime South Bound Mediation Layer
Optical
Switches
Routers
Design
295707

## Cisco Prime Products Architecture Overview

The primary components of the Cisco Prime product architecture are described below.

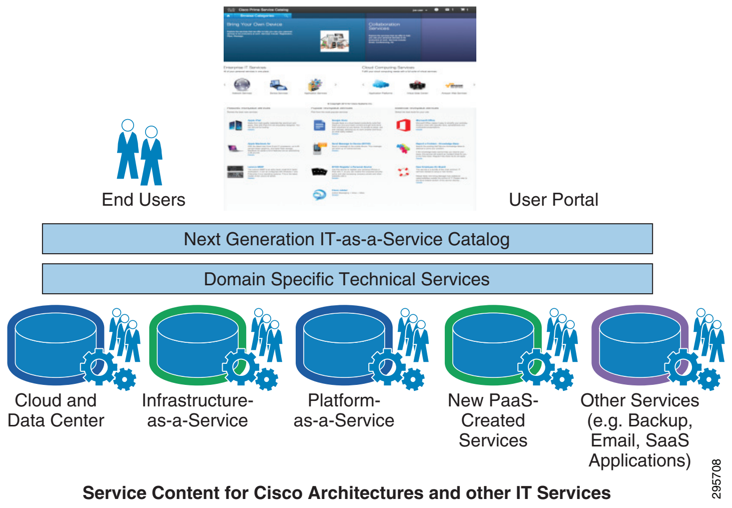
- **Cisco Prime Products Data Mode**l—The Cisco Prime products data model is based on the Multi-Technology Operations Systems Interface (MTOSI) 2.0 industry standard from the TeleManagement Forum (TMF) and is populated from Cisco Element (Domain) Managers.
- **Cisco Prime Products ServiceLink**—ServiceLink coordinates activity among Lifecycle Managers (i.e., among elements controlling design, fulfillment, assurance, and analysis), and between lifecycle and domain managers. It provides a redundant, highly scalable framework with the ability to mediate messages, and embedded Cisco Prime Product Framework Services such as the scheduler, locator service, and persistence.
- **Cisco Prime Products Abstraction Layer—**This component removes the complexity of managing a wide variety of interfaces by abstracting the information into the Cisco Prime products Data Model. The abstraction layer mediates information stored in the Cisco Prime products Data Model and makes it available to all Cisco Prime Suite components and interfaces.
- **Cisco Domain Managers**—Domain managers (or element managers) provide basic fault, configuration, accounting, performance, security (FCAPS) functionality for each specific technology domain. Examples include the Prime Network domain manager for IP/packet services and Prime Optical for optical transport.
- **Cisco Prime Products Southbound Mediation Interfaces (SBI)**—The Cisco Prime products SBI layer provides a common, mediated interface to communicate with any Cisco device. Each domain manager uses this element to abstract the intricacies of device communication via different protocols (e.g. SNMP, CLI, XML, CORBA) using standard interfaces. The SBI also includes a work flow engine that can be customized with graphical, drag-and-drop design tools. These tools simplify domain management in the field and help service providers customize and extend the network model without having to wait for new Cisco Prime products software releases. Additionally, since the SBI is developed in close collaboration with the Cisco hardware business unit, it is designed from the ground up to optimize the interaction between Cisco devices and Cisco Prime products.

- **Cisco Prime Products Northbound Interfaces (NBI)**—These interfaces allow for direct access to lifecycle and domain managers by both Cisco Prime Central and third-party products such as OSS systems. They are standards-based; available in a number of formats, web services, and native XML; and can be further customized in the field. The NBI is also designed for maximum security, employing transport encryption and requiring authenticated access.
- **Cisco Prime Products Software Development Kits (SDK)**—These SDKs provide service providers with maximum flexibility to customize the solution and easily integrate it into their environment. They include APIs and documentation for access to domain managers along with lifecycle managers for both northbound and southbound interfaces.
- **Cisco Prime Products Lifecycle Managers (Design, Fulfill, Assure, Analyze)**—These applications provide end-to-end experience lifecycle management services across all technology domains. The lifecycle managers coordinate with each other through the Cisco Prime Products Service Bus, and rely on the common Cisco Prime Products Data Model for shared device and service context. Examples include Cisco Prime Fulfillment and Cisco Prime Performance Manager.

## Cisco PSC with OpenShift

The Cisco Prime abstraction layer interacts with OpenShift, reducing the complexity for the administrator and the end user, as shown in Figure 1-3.

*Figure 1-3 Cisco PSC Abstracts Domain Specific Controllers*


End Users
User Portal
Next Generation IT-as-a-Service Catalog
Domain Specific Technical Services
Cloud and Data Center
Infrastructure-as-a-Service
Platform-as-a-Service
New PaaS-Created Services
Other Services (e.g. Backup, Email, SaaS Applications)
Service Content for Cisco Architectures and other IT Services
295708

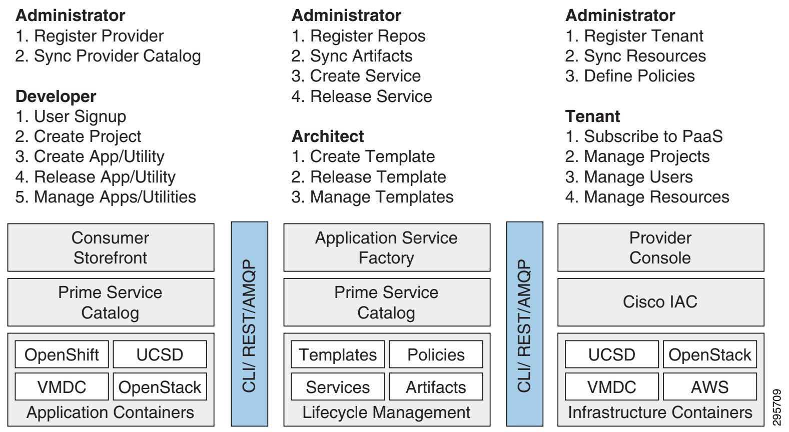
By abstracting the domain specific interactions from the user, Cisco PSC provides the same user experience for all services in the catalog. The user provisions IaaS, PaaS, and other IT services through the same pane of glass, with same graphical user interface.

The architecture of the Cisco PSC is focused on three main areas:

1. Consumer storefront
2. Service factory
3. Provider console

The integration for this document is focused primarily on the consumer storefront (Figure 1-4).

*Figure 1-4 Cisco PSC Architecture*



- The consumer storefront is targeted at the developer who will use the PaaS environment. It includes functions such as signing up a new PaaS user (developer), and creating, modifying, and deleting PaaS applications. Most of the work for this phase is focused on the Consumer Storefront.
- The application service factory is the work flow that creates stack templates, and publishes services that embed those templates. The work flow is managed by the Stack Designer tool. The stack designer capabilities are beyond the scope of work in this phase. Stack designs used in this phase are pre-built, and exposed through the consumer storefront. Subsequent phases will include stack designer capabilities that will further enhance OpenShift with a drop-and-drag interface.
- The Provider Console is targeted at the provider of PaaS services. While not used in this document, the provider console allows the administrator to spin-up, manage, and delete OpenShift instances in subsequent phases.

Through the Consumer Storefront, this document provides unparalleled ease-of-use to build and expose PaaS services, and to configure and manage IaaS services.

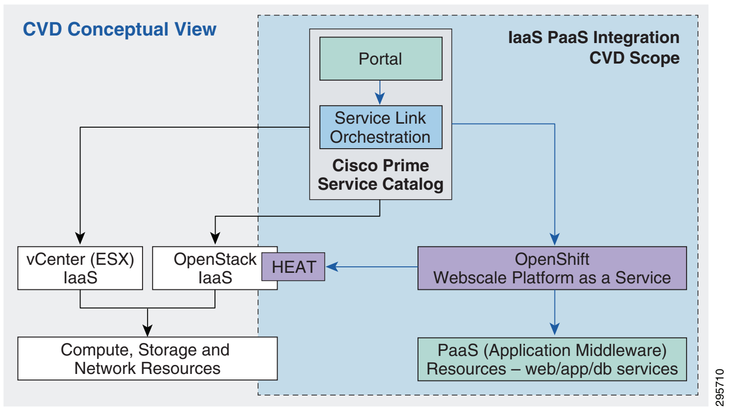
Together, OpenShift and the Cisco PSC create the market leading solution to satisfy IT departments' needs for IaaS for legacy applications, and PaaS to develop and maintain applications.

CHAPTER 2

# System Overview

The Cisco IaaS/PaaS/SaaS Integration Release 1.0 architecture specifies the data flow, the API interactions and the integration points between Cisco PSC, related orchestration engines, and specific resource managers for the relevant IaaS, PaaS, and SaaS layers. This specific document focuses on the architecture, design, implementation and validation of the Cisco Prime Service Catalog OpenShift Integration 1.0 release. Figure 2-1 depicts the goal of this release—to enable Cisco Prime Catalog integration with OpenShift Enterprise 2.0 (OSE 2.0).

***Figure 2-1 Cisco Prime Catalog Integration with OpenShift Enterprise 2.0***


CVD Conceptual View
IaaS PaaS Integration CVD Scope
Portal
Service Link Orchestration
Cisco Prime Service Catalog
vCenter (ESX) IaaS
OpenStack IaaS
HEAT
OpenShift Webscale Platform as a Service
Compute, Storage and Network Resources
PaaS (Application Middleware) Resources – web/app/db services
295710

## System Description and Scope

The solution focuses on the integration aspects rather than the implementation of IaaS or PaaS systems. As such, the scope of the document is limited to implementing, testing and validation of specific use cases outlined in the "Use Cases" section on page 2-2.

To enable the implementation and testing of the integration points between Cisco Prime Catalog and OpenShift Enterprise Broker, the underlying IaaS infrastructure is assumed to be pre-built/built out.

## Use Cases

IaaS-level solutions available in today's market are fairly mature, but an integrated experience in ordering and managing PaaS services and IaaS services through the same management portal and using the same orchestration tool sets is lacking. Customers are looking to Cisco for an integrated IaaS, PaaS, and SaaS ordering and orchestration solution that allows them to expose the power and functionality of their data center infrastructure and value-added platform and software services to their end-users.

The following user cases satisfy a majority of concerns outlined and pave the way for future iterations, providing greater integration between the Service Catalog and the PaaS system (OpenShift).

### Use Case 1: Integrate Provisioning for IaaS and PaaS

End users logging into the PSC can currently order IaaS services (VMs, networks, storage) for underlying IaaS implementations (VSphere or OpenStack based) using packaged Cisco Prime solutions. The same set of users must be able to order PaaS application stacks easily from the same set of Prime Catalog portal pages.

This guide enables the following:

- Single pane provisioning of IaaS and PaaS components.
- Unified authentication store for Portal and PaaS components.
- Delegate proxy from OpenShift to Portal to allow portal to provision components on behalf of Portal users.

### Use Case 2: Application Stack Creation

OpenShift introduces the concept of cartridges for specific web application stack capabilities. PSC must allow portal users to provision and instantiate those cartridges into an OpenShift installation.

This guide enables the following:

- Project / Application name creation.
- Web Application Stack creation (e.g. Python, Java web-app containers).
- Value-add (Add-on) capability creation (e.g. MySQL database).

### Use Case 3: Single Pane Management of Application Stack

Similar to the current PSC capabilities for listing all end user-owned IaaS resources and the ability to take actions against them (start, stop, etc.), the PSC must provide a PaaS component / application-level view of the resources ordered and allows the user to take actions against them.

This guide enables the following capabilities:

- Single pane listing of all end user-owned PaaS applications.
- Ability to stop a particular PaaS application.
- Ability to start a particular PaaS application.
- Provide a view into a scaled PaaS application.
- Ability to delete a PaaS application.

- Listing of relevant information about a PaaS application (URLs, SSH, and GIT access endpoints, application name).

## Use Case 4: Dynamic Scaling of Application (PaaS) Stack Components

OpenShift allows application web stacks to be created that use the inbuilt scaling features, including auto scaling and manual scaling.

This guide enables the following:

- While provisioning via the PSC catalog, expose the ability for the end-user to provision a scaled application.

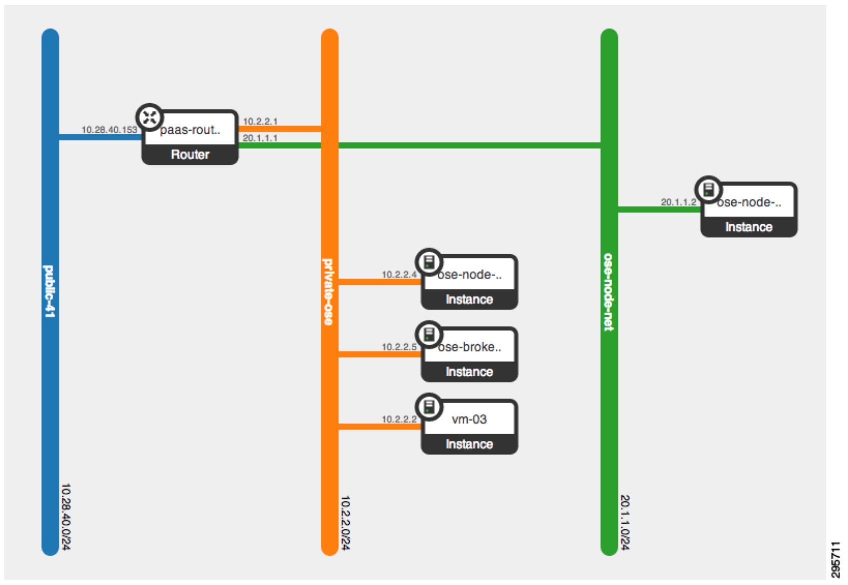
## Use Case 5: Network-based Segmentation of PaaS Districts for Security

By default, OpenShift runs all applications in a shared set of IaaS resources (VMs, hosts, network segments) using Linux container primitives (CGroups, SElinux, etc.) to ensure secure multi-tenancy within the VMS. End users/customers may have a requirement to host a specific set of applications / application stacks in a separate set of IaaS resources from the primary shared pool of VMs.

This guide enables the following:

- Set up OpenShift (using OpenShift districts) to create OpenShift nodes in a different network and expose application stack creation on those VMs as an order attribute. See Figure 2-2.

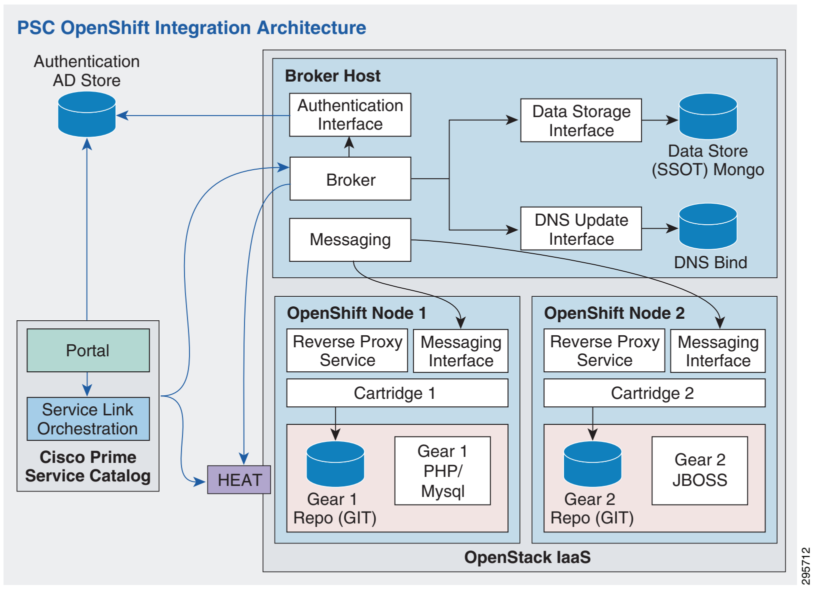
***Figure 2-2 OpenShift Nodes in Public and Private Districts***



## System Architecture

Figure 2-3 shows the products chosen for this solution based on functional availability within the necessary time frame.

*Figure 2-3 Cisco PSC—OpenShift Integration Architecture*



The system architecture follows the current industry best practices for integration:

- API-based integration where possible.
- Use the single source of truth directly via APIs (with no data replication), in order to have real-time data available.
- No proliferation of user identity systems.

As such, the integration in this document uses existing adapters available in the PSC ServiceLink component to talk to the OpenShift Broker via APIs.

If an existing component in the PSC does not meet the integration requirements, this document lists out custom components and scripts developed to aid in the integration.

All calls to the OpenShift broker REST APIs are synchronous in nature. The PSC does not store OpenShift Broker-managed artifact and metadata in the Service Catalog database. Instead, all displays and display updates in the PSC portal are done in real-time via API calls.

## System Components

The following system components are detailed:

- PSC, page 2-5
- OpenStack, page 2-6
- OpenShift, page 2-5
- Supporting Components, page 2-6
  - VMWare ESX Server, page 2-6
  - Microsoft Active Directory, page 2-7
- Cisco Components, page 2-7
- Third-Party Components, page 2-7

## PSC

PSC provides the necessary functionality for the end user and administrative portals and the end user-facing service catalog. PSC is a highly customizable product and, if required, an extensive Advanced Services team specializing in IAC is available to assist the customer with more complex work flows.

In brief, end user requests for compute, network, and/or storage and platform/application resources are received in PSC. The ServiceLink component then instructs the required domain orchestrator's to execute the appropriate tasks in order to fulfill the end user request.

The following sections describe the features and functionalities of IAC that are relevant to meeting the requirements for managing and configuring OpenShift through Cisco Prime Services Catalog.

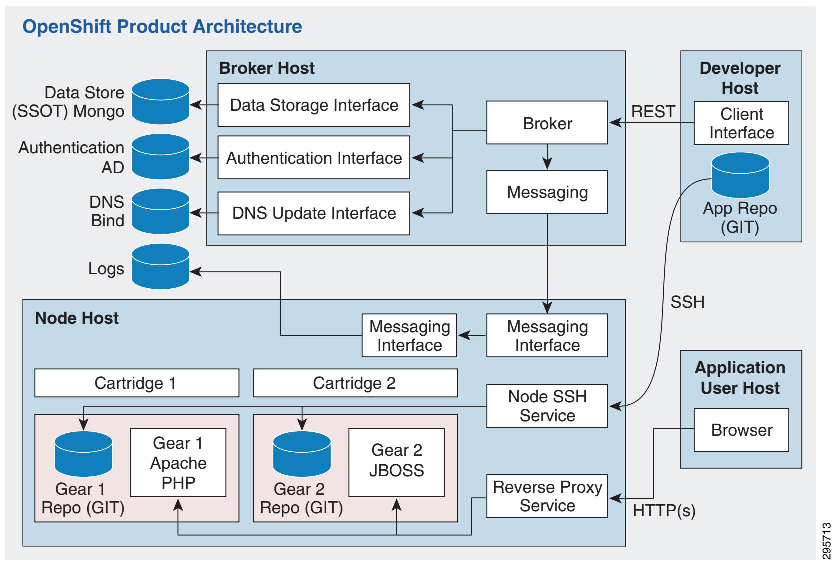
The Cisco PSC with additional IAC portal content is provided as a virtual appliance for ease of deployment. One of the components of SC is the Service Designer, which allows the Cloud Provider to design and package services as products and to catalog these services for end users to browse through and order. The look and feel of the portal is fully customizable, allowing the Cloud Provider to brand the portal as appropriate. Refer to Cisco Service Portal Designer Guide for details.

## OpenShift

OpenShift Enterprise consists of several components. This section defines primary components, and various configurations within the document.

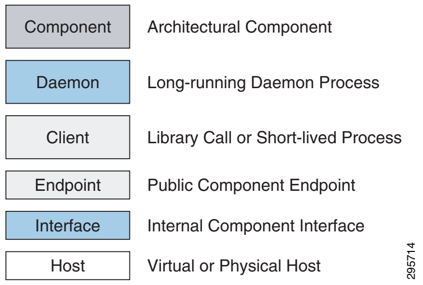
Figure 2-4 shows the Open/Shift product architecture.

***Figure 2-4 OpenShift Product Architecture***



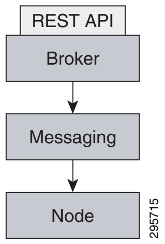
The diagrams in subsequent sections show elements based on the legend in Figure 2-5.

***Figure 2-5 Legend for Elements in this Document***


Component
Architectural Component
Daemon
Long-running Daemon Process
Client
Library Call or Short-lived Process
Endpoint
Public Component Endpoint
Interface
Internal Component Interface
Host
Virtual or Physical Host
295714

An OpenShift Enterprise deployment consists of two logical types of hosts: a broker and one or more nodes. The broker handles the creation and management of user applications, the user authentication service, and manages communication with the appropriate nodes. The nodes run the user applications in contained environments called gears. The broker queries and controls nodes using a messaging service. Figure 2-6 provides a simplified version of the interaction between these two types of hosts.

***Figure 2-6 Broker Interaction with Node Hosting Gears/Applications***



## OpenStack

In this document OpenStack IaaS is used to:

- Demonstrate the dynamic scaling of VMs as required by OpenShift.
- Create networking resources (Subnets/VLANs) where different OpenShift Nodes can be placed to allow for network segmentation requirements of OpenShift clients.
- Allow the IaaS work flows to kick off OpenStack HEAT Orchestration templates in general for creation of OpenStack native resources (e.g. networks, VLANs, VMs, and separate OpenShift installations).

## Supporting Components

Supporting components are those that are not part of the scope of this document, but are required to provide support for the components above(e.g. VMWare ESX for hosting PSC VM image).

### VMWare ESX Server

The packaged PSC component used in this validation is packaged as a VMware VM Image (OVF) file. An ESX server is utilized to stand up the packaged PSC appliance.

Future iterations of this document will use an IaaS-agnostic version of the PSC appliance.

### Microsoft Active Directory

The packaged PSC appliance comes with its own Active Directory installation. The same active directory is leveraged in this document to act as an authentication store for both PSC and the OpenShift Broker.

## Cisco Components

Table 2-1 shows Cisco components.

*Table 2-1 Cisco Components*

| Component Positioning | Product | HW Configuration | Software Version(s) |
|---|---|---|---|
| Data Center | Cisco UCS System | | |
| Cloud | Cisco PSC | | 1.00.04 Jan 9 2008 |

## Third-Party Components

Table 2-2 shows third party components

.

*Table 2-2 Third Party Components*

| Component | Vendor/Partner | Hardware/Software |
|---|---|---|
| IaaS Software | Red Hat | Red Hat OpenStack (HAVANA) |
| PaaS Software | Red Hat | OpenShift Enterprise 2.01 |

CHAPTER 3

# System Design

This chapter describes the topology, hardware, software, and configuration of the design as tested. While configurations in this document may be used as guidelines for other configurations, results reflect hardware and software configurations exclusive to testing conducted for this document.

## IaaS Topology

As mentioned in Chapter 2, "System Overview", the scope of this document does not include the design and validation of the IaaS layer components. The test bed for OpenShift integration requires an IaaS layer component. OpenStack is used to provide the test bed. For the sake of completeness, the design of the IaaS test bed dependency is outlined below.

## Hardware and Software Components of the Architecture

Table 3-1 describes hardware and software components.

.

*Table 3-1 Architectural Hardware and Software Components*

| Vendor | Name | Specification | Purpose |
|---|---|---|---|
| Cisco | Cisco UCS B200 M3 Server | 2.1(3a)B 2CPU 16 Cores,96GB Mem,1TB Storage(1xHDD) | OpenStack Controller |
| Cisco | Cisco UCS B200 M3 Server | 2.1(3a)B 2CPU 16 Cores,96GB Mem,1TB Storage(1xHDD) | OpenStack Compute, Network Node |
| Cisco | Cisco UCS VIC 1240, 1280 | VIC 1240, 1280 | Cisco Virtual Interface Card (adapter) firmware |
| Cisco | Cisco UCS 6248UP Fabric Interconnect | Kernel Version: 5.0(3)N2(2.1.1.3a) | Blade Network and Management |
| Red Hat | Red Hat OpenStack Platform | RHOS 4 (Havana) | OpenStack IaaS |
| Red Hat | Red Hat OpenShift Enterprise | OSE 2.0 | OpenShift PaaS |

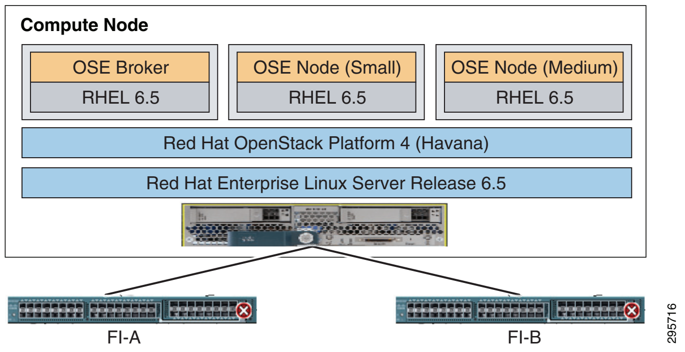
## OpenStack Services Placement

Table 3-2 describes OpenStack services placement. Figure 3-1 depicts the hardware/software stock.

*Table 3-2 OpenStack Services Placement*

| Host Name | Role | Services |
|---|---|---|
| node01 | OpenStack Compute | httpd (horizon), mongod, mysqld, qpid, tgtd, ntpd, neutron-server, cinder-, glance-, nova-api,<br><br>nova-conductor, nova-scheduler, nova-console, nova-metadata-api, nova-novncproxy, ceilometer-*, heat-* |
| node02 | OpenStack Controller, Network Node | libvirtd, ntpd, messagebus, openvswitch, dnsmasq, neutron-dhcp-agent, neutron-l3-agent,<br><br>neutron-metadata-agent,neutron-openvswitch-agent,ceilometer-compute |

*Figure 3-1 Hardware/Software Stack for the Compute Node used for Testing*



## OpenShift Topology

OpenShift, by default, assumes a flat network layout for its deployment, although the deployment is highly configurable based on needs. This document lays out the OpenShift nodes on two different network segments:

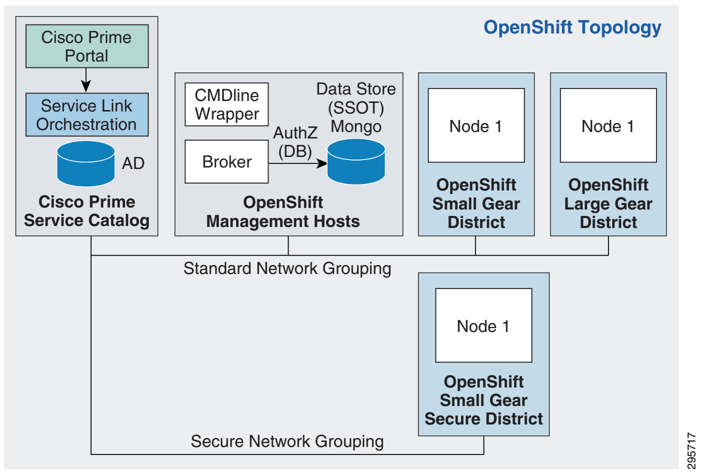
- Standard network segment
- Secure network segment

The OpenShift management nodes and the standard application nodes are placed on one network segment (standard network) and another set of application nodes are placed on a different network segment (secure network).

The secure network segment is protected using ACLs in security groups to permit only http and https traffic from the general network. In the opposite flow, activemq/storm traffic originating from the secure network is blocked from the activemq process on the broker host on the standard network.

The intention is to show that a single OpenShift installation can be deployed across Layer 3 network boundaries and still be managed as a single cohesive OpenShift installation (Figure 3-2).

*Figure 3-2 OpenShift Deployment Logical Network Topology Component Layout on IaaS*


OpenShift Topology
Cisco Prime Portal
Service Link Orchestration
AD
Cisco Prime Service Catalog
CMDline Wrapper
Data Store (SSOT) Mongo
AuthZ (DB)
Broker
OpenShift Management Hosts
Node 1
OpenShift Small Gear District
Node 1
OpenShift Large Gear District
Standard Network Grouping
Node 1
OpenShift Small Gear Secure District
Secure Network Grouping
295717

## Design Principles

The following design principles are provided for consideration.

### Tenancy Model

Implementation of tenancy (and, by extension, multi-tenancy) can be different at different levels of the Cloud Stack even though the core notion of a tenant remains the same. The tenant is a user/set of users/organizations that owns resources, is accountable for the cost of the usage of such resources, and has full control over how those resources are used.

Examples of how tenancy may be broken down at the various layers of the cloud stack are:

- **IaaS Tenancy**—Virtual Data Center, VMs, Subnets, VLANs, Storage pools, total bandwidth.
- **PaaS Tenancy**—Secure OS Containers, DNS namespaces/ domains, application URLs.
- **SaaS Tenancy**—Application-level authorization, groups of users, roles, data encryption.

This solution does not directly address tenancy models and tenancy management. The solution allows for a flexible tenancy model, where a tenant can be defined by the provider with a combination of IaaS-level resources, groups of users, organizational boundaries, and PaaS-level resources.

For the purposes of this solution, an end user is the same as a tenant.

This requires that certain administration tasks for Day Zero be performed as part of the implementation of the solution, either manually or by running installation and configuration scripts and actions.

### Identity and Entitlement

This solution does not directly address identity, access management, or entitlement for different components in the stack. The assumption is that providers have an existing identity management solution with which the components need to integrate. For completeness of the flows, Microsoft Active Directory (AD) was chosen as the single identity store and authentication provider for all components in the stack.

- Each component integrates with the same AD instance(s) to provide a user store and an lDAP- and ADSI-based authentication mechanism.
- CIAC PSC (PSC) uses AD integration for authentication purposes.
- OpenStack Keystone uses AD/LDAP integration for authentication purposes.
- OpenShift Broker uses AD/LDAP integration for authentication purposes.
- System accounts is used wherever possible to identify a component to another component wherever possible; e.g. PSC uses a system account to talk to OpenShift.
- Authorization is performed at each component.

This solution does not take into account the user provisioning and management work flows for AD. For the purposes of verification, users will be created out of band in the Active Directory installation.

## Request Work Flows

The following design principles for implementing user request work flows are provided for consideration.

### IaaS Work Flow

Full lifecycle management of IaaS resources is out of scope of this document. For the purposes of the overall solution proposed in this document, a user can order a virtual machine (VM) and use the existing integration that CIAC has with the hypervisor managers to manage the VM.

### OpenShift PaaS Work Flow

As describe in the Tenancy Model section above, the tenancy for the purposes of this solution is based on the notion of an end user.

To facilitate the creation of resources on the OpenShift environment, a system user (admin user) will be created in OpenShift that will create /delete resources on behalf of the end-user.

1. System user owns all namespaces createdxaaslabs.cisco.com (e.g):
    a. System user is allowed large number of gears; e.g. 1000
2. System user creates app in namespace on behalf of end user:
    a. System user creates / checks end-user in OpenShift.
    b. System user creates namespace for end-user.
    c. System user adds end-user to the app as an edit user.

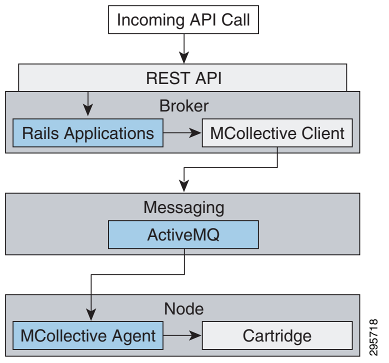
## OpenShift Internal Architecture

This section highlights the internal design of OpenShift. The design is internal to OpenShift and this document does not modify the inherent design in the OpenShift product.

## Communication Mechanisms

Communication from external clients, such as the client tools or the Management Console, occurs through the REST API that is hosted by the broker. The broker then communicates to the nodes using the messaging service component. MCollective queries a set of nodes and communicates securely with individual nodes. Figure 3-3 provides a high-level description of this communication.

*Figure 3-3 OpenShift Enterprise Communication Mechanisms*



## State Management

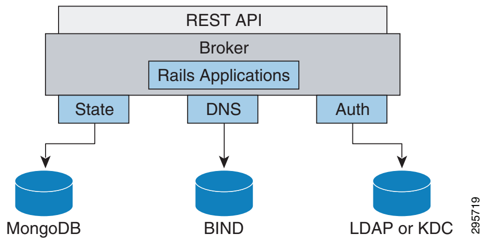
The broker is responsible for managing persistent data for OpenShift Enterprise using three distinct interfaces that represent the complete state. Three interfaces are used because each data store is pluggable and each type of data is usually managed by a separate system. Table 3-3 describes each section of application data, and Figure 3-4 depicts enterprise state management.

.

*Table 3-3 Sections of Application Data*

| Section | Description |
|---|---|
| State | This is the general application state where the data is stored using MongoDB by default. |
| DNS | This is the dynamic DNS state where BIND handles the data by default. |
| Auth | This is the user state for authentication and authorization. This state is stored using any authentication mechanism supported by Apache, such as mod_auth_ldap and mod_auth_kerb. |

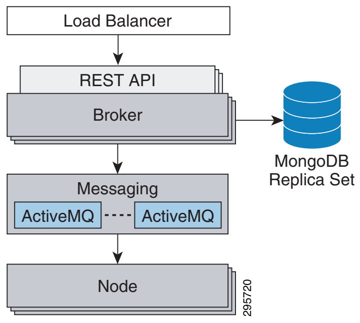
*Figure 3-4 OpenShift Enterprise State Management*



## Redundancy

OpenShift Enterprise incorporates redundancy where each architectural component can be configured redundantly. The broker applications themselves are stateless and can be set up behind a simple HTTP load balancer. The messaging tier is also stateless and MCollective can be configured to use multiple ActiveMQ endpoints. Multiple MongoDB instances can be combined into a replica set for fault tolerance and high availability (Figure 3-5).

***Figure 3-5 Implementing Redundancy in OpenShift Enterprise***



## Configuration Guidelines

Standard configuration guidelines for PSC are followed during installation. Any customizations and deviations from the standard configurations are outlined in the implementation section.

Standard RHOS OpenStack configuration guidelines are followed during installation. Any customizations and deviations from the standard configurations are outlined in the implementation section.

The following design details are provided for consideration.

- PSC Portal, page 3-6
- PSC ServiceLink, page 3-7
- ServiceLink Openssh Adapter, page 3-7
- Service Proxies, page 3-8
- OpenShift Command Line Wrapper, page 3-8
- OpenShift Prime Entitlement Integration, page 3-8

## PSC Portal

The following components are to be created to fulfill the portal content requirements for the integration.

- User Workspace Portal Page—PaaS Order Portlet, page 3-7
- Application Inventory Portal Page, page 3-7
- PaaS Order Form, page 3-7

### User Workspace Portal Page—PaaS Order Portlet

PSC and CIAC ships with Portlets that allow end-users to provision IaaS resources. An additional Portlet is added to the existing User Workspace portal that allows users to order PaaS resources. The Portlet will invoke a PaaS order form that is described further below.

### Application Inventory Portal Page

PSC and CIAC ships with a portal page that allows end-users to view and manage ordered IaaS resources. An additional Portal page is added to PSC. The portal page has content and Ajax call functions embedded in it to allow it to call OpenShift APIs directly on the OpenShift Broker. The page exposes data retrieved from the APIs as viewable and actionable elements in the portal page.

### PaaS Order Form

A PaaS order form is added to PSC. The form uses table values to store the following elements:

- Form display and value capture attributes.
- Pre-defined OpenShift gear sizes.
- Costs associated with order elements.

The form:

- Computes and sets the OpenShift namespace based on the username of the logged in user.
- Retrieves available OpenShift cartridges via an API call to the OpenShift broker.
- Retrieves available additional cartridges via an API call to the OpenShift broker.
- Upon capture of the user-entered values, submits the request to a service defined in the Service Link component, which is outlined below.

## PSC ServiceLink

Several services are created in Service link to allow it to integrate with the back end OpenShift Broker:

### Creating OpenShift Application Service

A service that can be called by PSC service plan components to pass application meta data to the OpenShift Broker asking it to create applications and wait for a reply back.

### Add User Service

A service that can be called by PSC service plan components to pass user meta data to the OpenShift Broker and asking it to create user and namespaces, and wait for a reply back.

### ServiceLink Openssh Adapter

The Add User service described above requires ServiceLink to be able to SSH into the broker box to call a script.

The Add User service is configured to use the OpenSSH adapter.

## Service Proxies

To avoid cross-site scripting and to enable encapsulation of broker API commands, two service proxies are created that run as simple jsp processes inside the ServiceLink j2ee container. Full stateful agents inside ServiceLink in updates to the document will eventually replace the JSPs.

### Creating OpenShift App Service Proxy

Translates ServiceLink XML to OpenShift-supported JSON and calls the OpenShift Broker service on behalf of ServiceLink.

### Broker Read-Only Services

Traps HTTP REST calls from PSC and ServiceLink, proxies the API calls to OpenShift Broker, and returns the results. This service is used by AJAX UI elements in the PSC portal.

## OpenShift Command Line Wrapper

Certain administrative commands in OpenShift are not exposed via REST API calls. Instead, command line APIs are available on the OpenShift Broker host.

These APIs are available as oo-* shell commands on the broker host and are internally implemented using OpenShift Ruby libraries.

Although OpenShift allows pluggable authentication on the broker, it requires that it keep track of the users and userids that are using the OpenShift system. The creation of users in the system can occur via two methods:

- The first time a user authenticates to the OpenShift Console or the OpenShift Broker directly using user credentials.
- An oo-* command is run on the broker hosts for forcing pre-creation of a user.

The PSC integration in the document allows a user to be logged in to PSC. PSC acts on behalf of the user to manipulate resources on the OpenShift system. As such, it is required that PSC be able to create / pre-create OpenShift users on the OpenShift system.

A Ruby wrapper script is created and installed in the OpenShift broker. This aids in the following:

- Create a user in the OpenShift system.
- Disallow user to create namespaces and applications directly.
- Create a namespace (sub-domain container) in the OpenShift System.
- Assign the user as an edit user for the Namespace.

## OpenShift Prime Entitlement Integration

This solution does not directly address entitlement for different components in the stack. The assumption is that providers have an existing identity management solution with which components integrate. Each component integrates with the same AD instance(s) to provide a user store and an LDAP- and ADSI-based authentication mechanism.

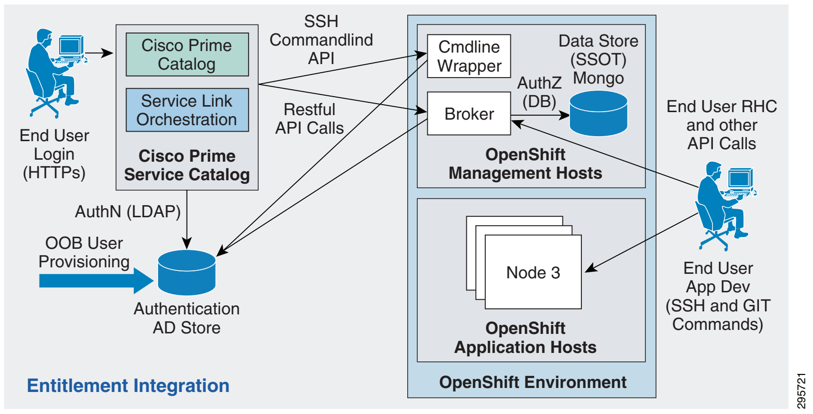
1. CIAC PSC (PSC) uses AD integration for authentication purposes.

2. OpenStack Keystone uses AD/LDAP integration for authentication purposes.
3. OpenShift Broker uses AD/LDAP integration for authentication purposes.
4. System accounts is used wherever possible to identify a component to another component wherever possible; e.g. PSC uses a system account to talk to OpenShift.
5. Authorization is performed at each component.

This solution does not take into account the user provisioning and management work flows for AD. For the purposes of verification, users will be created out of band in the Active Directory installation.

Figure 3-6 shows the AD flows for OpenShift and Cisco Prime Service Catalog in the integrated solution.

***Figure 3-6 OpenShift Prime Entitlement Integration***



## System Level Design Considerations

The following system level design considerations are defined.

- Scalability, page 3-9
- Availability, page 3-10
- Security, page 3-10
- Manageability, page 3-11
- Service Assurance and Monitoring, page 3-11

### Scalability

PSC scalability is not in the scope of this document. The scalability of PSC is addressed directly in the PSC deployment guides. Further assistance is available from Cisco Advanced Services to guide in the deployment of PSC.

OpenShift scalability is not in the scope of this document. The scalability of OpenShift and design considerations for it is addressed directly in the Red Hat OpenShift Enterprise deployment guides. Further assistance is available from Cisco Advanced Services working in conjunction with Red Hat Professional Services to guide in the deployment of OpenShift.

## Availability

PSC availability is not in scope of this document and is addressed directly in the PSC deployment guides. Further assistance is available from Cisco Advanced Services to guide in the deployment of PSC.

OpenShift availability is not in the scope of this document. Availability of OpenShift and design considerations for it is addressed directly in the Red Hat OpenShift Enterprise deployment guides. Further assistance is available from Cisco Advanced Services working in conjunction with Red Hat Professional Services to guide in the deployment of OpenShift.

## Security

All components are web-based and connectivity is either browser or API-based. As such, all web-based threat vectors apply to this system. Although there are multiple deployment options for the system components, the following general security elements apply.

### Connectivity

All browser-based connectivity to the Cloud Portal and Admin consoles (PSC) is authenticated (LDAP credentials) and authorized (LDAP roles).

All API-based connectivity is authenticated (LDAP credentials) and authorized (LDAP roles).

SSL connectivity is, at a minimum, available as optional for all connections.

### Tenant Security Inside OpenShift

All application code runs in pseudo containers that are controlled by Linux control groups (CGroups). CGroups allow grouping of resources (CPU slots, memory, storage, I/O) such that they can be assigned limits and constraints. The CGroups can be applied to OS level processes such that the processes run within the constraints defined in the croup policies.

All application code is further protected by SElinux policies, which are turned by default. Explicit SElinux policies are applied to each CGroups policy on an application-by-application basis.

Connectivity to the Broker nodes for API interaction is done via an http over SSL (https) session.

Connectivity to the Broker nodes for administration purposes is done via authenticated SSH calls into the Linux host that the Brokers and Application hosts are running on.

Connectivity to the OpenShift Application nodes for end-users /developers to push code, view log etc., are done via SSH using an OpenShift-supplied Linux PAM plugin that allows authenticating and identifying applications based on SSH keys and application names.

### Cross Site Scripting (XSS) and Buffer Overflow

Cisco owns and develops all PSC components. As such, the components follow the Cisco Secure Development Lifecycle (CSDL) Rules and Guidelines for Development, including safe libraries and data input validation.

### Data Protection

All connectivity to the databases for the components is at a minimum authenticated via username and password.

SSL connectivity to the databases is supported as an option.

The components do not support any additional data encryption above that provided by the specific vendor database (MS SQL Server, Oracle).

## Manageability

PSC manageability is not in scope of the document and is addressed directly in the PSC deployment guides. Further assistance is available from Cisco Advanced Services to guide in the deployment of PSC.

OpenShift manageability is not in the scope of the document and is addressed directly in the Red Hat OpenShift Enterprise deployment guides. Further assistance is available from Cisco Advanced Services working in conjunction with Red Hat Professional Services to guide in the deployment of OpenShift.

## Service Assurance and Monitoring

Service assurance and monitoring for the two major components (PSC and OpenShift) are not in the scope of this document and their respective deployment guides address this. Further assistance is available from Cisco Advanced Services working in conjunction with Red Hat Professional Services to guide in the deployment of OpenShift.

CHAPTER 4

# System Implementation

The primary focus of this document is the integration between Cisco PSC and Red Hat OpenShift PaaS. Most of this work involves development of PSC extensions leveraging PSC Request Center and ServiceLink components. Almost no changes occurred on the OpenShift side because the majority of management and monitoring operations are exposed via Restful management API on OpenShift Broker.

For this testbed, a fully built and configured CIAC 3.1.1 virtual appliance running Windows 2008R2 is used. Active Directory is set up and configured as part of the virtual appliance.

For the infrastructure, a fully populated UCS 5108 chassis connected to UCS 6248UP Fabric Interconnects are used. Up links on FIs are connected to the Nexus 5K switch. Two VLANs are used to the servers group—a management VLAN with subnet 172.29.87.224/28 and a public VLAN with subnet 172.29.87.240/28.

**Note** User familiarity with UCS hardware configuration via UCS Manager is required so documents that detail UCS Manager configurations only are referenced.

## Red Hat OpenStack Deployment

The following configurations were used in Red Hat OpenStack testbed deployment.

- Controller Node Configuration, page 4-1
  - Installing and Configuring Controller Node, page 4-2
  - Installing OpenStack Identity, page 4-3
  - Installing OpenStack Image Service, page 4-4

## Controller Node Configuration

The following instructions provide guidance for configuring controller node.

The controller node is assigned to node01.ctocllab.cisco.com (172.29.87.229).

Configure the network interfaces on controller node:

```
Interface eth0: ifcfg-eth0

DEVICE=eth0
TYPE=Ethernets
ONBOOT=yes
NM_CONTROLLED=no
```

```
BOOTPROTO=static
IPADDR=172.29.87.229
PREFIX=28
GATEWAY=172.29.87.225
DNS1=172.29.74.154
DOMAIN=ctocllab.cisco.com
DEFROUTE=yes

 Interface eth1: ifcfg-eth1
DEVICE=eth1
TYPE=Ethernet
ONBOOT=yes
NM_CONTROLLED=no
BOOTPROTO=none
PROMISC=yes
```

## Installing and Configuring Controller Node

The following instructions provide guidance for installing and configuring controller node.

The controller node is assigned to [node01.ctocllab.cisco.com (172.29.87.229)].

Register system with Red Hat Network for a pre-registered user id.

```
#> subscription-manager register
```

List available or consumed subscriptions for registered system.

```
#> subscription-manager list [--available|--consumed]
```

Attach the system to a given Pool Id for RHOS 4 (Havana).

```
Subscription Name: Red Hat Cloud Infrastructure Business Partner Self-Supported NFR
(4-sockets)
Provides:          Red Hat OpenStack
                   JBoss Enterprise Application Platform
                   Red Hat Enterprise Linux Server
                   Red Hat OpenStack Beta
                   Red Hat Enterprise Virtualization
                   Red Hat Enterprise MRG Messaging 2
                   Red Hat CloudForms
                   Red Hat Enterprise Linux 7 Public Beta
                   Red Hat Beta
SKU:               ...
Contract:          ...
Account:           ...
Serial:            ...
Pool ID:           <pool-id>

#> subscription-manager attach --pool=<pool-id>
```

Install yum-utils to enable relevant openstack rpms for openstack-4.0.

```
#> yum install -y yum-utils
#> yum install -y yum-plugin-priorities
#> yum-config-manager --enable rhel-6-server-openstack-4.0-rpms \
        --setopt="rhel-6-server-openstack-4.0-rpms.priority=1"
#> yum update -y
#> reboot
```

Install and configure the Database server.

```
#> yum install -y mysql-server
#> service mysqld start
#> chkconfig mysqld on
```

Configure firewall to allow tcp traffic for msql (add the following line to /etc/sysconfig/iptables).

```
"-A INPUT -p tcp -m multiport --dports 3306 -j ACCEPT"
#> service iptables restart
```

Set the database administrator password.

```
#> /usr/bin/mysqladmin -u root password "PASSWORD"
```

Install and configure the Message Broker (qpid).

```
#> yum install -y qpid-cpp-server qpid-cpp-server-ssl
```

Register qpid users.

```
#> saslpasswd2 -f /var/lib/qpidd/qpidd.sasldb -u QPID cinder
#> saslpasswd2 -f /var/lib/qpidd/qpidd.sasldb -u QPID neutron
#> saslpasswd2 -f /var/lib/qpidd/qpidd.sasldb -u QPID nova
#> sasldblistusers2 -f /var/lib/qpidd/qpidd.sasldb
```

Configure firewall to allow tcp traffic for qpid (add the following line to /etc/sysconfig/iptables).

```
"-A INPUT -p tcp -m tcp --dport 5672  -j ACCEPT"
#> service iptables restart
```

Start the service.

```
#> service qpidd start
#> chkconfig qpidd on
```

## Installing OpenStack Identity

The following instructions provide guidance for installing the OpenStack identity service.

Keystone Configuration (/etc/keystone/keystone.conf).

```
[DEFAULT]
admin_token = 12de4ec0f1a924f3b20e
bind_host = 172.29.87.229
compute_port = 8774
debug = False
verbose = False
[sql]
connection = mysql://keystone:<passwd>@172.29.87.229/keystone
[identity]
driver = keystone.identity.backends.sql.Identity
[catalog]
driver = keystone.catalog.backends.sql.Catalog
[token]
driver = keystone.token.backends.sql.Token
[signing]
token_format = UUID
[auth]
methods = password,token
password = keystone.auth.plugins.password.Password
token = keystone.auth.plugins.token.Token
```

Certain python package dependencies for openstack-keystone need to be explicitly installed.

```
#> yum install python-setuptools
#> easy_install pip
#> pip install six
#> pip install sqlalchemy
```

Configure firewall (add the following line to /etc/sysconfig/iptables).

```
"-A INPUT -p tcp -m multiport --dports 5000,35357 -j ACCEPT"
#> service iptables restart
```

## Installing OpenStack Image Service

The following instructions provide guidance for installing OpenStack image service.

### Glance configuration

(/etc/glance/glance-api.conf)

```
[DEFAULT]
default_store = file
bind_host = 0.0.0.0
bind_port = 9292
backlog = 4096
sql_idle_timeout = 3600
workers = 1
registry_host = 0.0.0.0
registry_port = 9191
registry_client_protocol = http

qpid_notification_exchange = glance
qpid_notification_topic = notifications
qpid_hostname = localhost
qpid_port = 5672
qpid_username =
qpid_password =
qpid_sasl_mechanisms =
qpid_reconnect_timeout = 0
qpid_reconnect_limit = 0
qpid_reconnect_interval_min = 0
qpid_reconnect_interval_max = 0
qpid_reconnect_interval = 0
qpid_protocol = tcp
qpid_tcp_nodelay = True

delayed_delete = False

scrub_time = 43200
sql_connection = mysql://glance:<passwd>@172.29.87.229/glance

[keystone_authtoken]
auth_host = 172.29.87.229
auth_port = 35357
auth_protocol = http
admin_tenant_name = services
admin_user = glance
admin_password = <passwd>
[paste_deploy]
flavor = keystone

(/etc/glance/glance-registry.conf)
[DEFAULT]
bind_host = 0.0.0.0
bind_port = 9191
backlog = 4096
sql_idle_timeout = 3600
api_limit_max = 1000
limit_param_default = 25
```

```
sql_connection = mysql://glance:<passwd>@172.29.87.229/glance
[keystone_authtoken]
auth_host = 172.29.87.229
auth_port = 35357
auth_protocol = http
admin_tenant_name = services
admin_user = glance
admin_password = <password>
[paste_deploy]
flavor = keystone
```

Configure firewall (add the following line to /etc/sysconfig/iptables).

```
"-A INPUT -p tcp -m multiport --dports 9292 -j ACCEPT"
#> chown -R glance:glance /var/log/glance/registry.log
```

Verify Glance service installation.

Download image and add to glance.

```
#> wget http://cdn.download.cirros-cloud.net/0.3.1/cirros-0.3.1-x86_64-disk.img
#> glance image-create --name="CirrOS 0.3.1" --disk-format=qcow2
--container-format=bare --is-public=true < cirros-0.3.1-x86_64-disk.img
#> glance image-list
+--------------------------------------+----------------------------------+------------
-+------------------+------------+--------+
| ID                                   | Name                             | Disk Format
| Container Format | Size       | Status |
+--------------------------------------+----------------------------------+------------
-+------------------+------------+--------+
| e71d8b33-d737-47d3-b5ed-32b085d0b47f | CirrOS 0.3.1                     | qcow2
| bare             | 13147648   | active |
+--------------------------------------+----------------------------------+------------
-+------------------+------------+--------+
```

## Installing OpenStack Compute Service

Refer to Installing the OpenStack Compute Service.

Install and Configure VNC Proxy.

```
#> yum install -y openstack-nova-novncproxy
#> yum install -y openstack-nova-console

(/etc/nova/nova.conf)
novncproxy_host = 172.29.87.229
novncproxy_port=6080
novncproxy_base_url=http://172.29.87.229:6080/vnc_auto.html
vncserver_listen=0.0.0.0
vnc_enabled=true
vnc_keymap=en-us
```

Configure firewall (add the following line to /etc/sysconfig/iptables).

```
"-A INPUT -m state --state NEW -m tcp -p tcp --dport 6080 -j ACCEPT"
#> service iptables restart
#> service openstack-nova-consoleauth start
#> chkconfig openstack-nova-consoleauth on
#> service openstack-nova-novncproxy start
#> chkconfig openstack-nova-novncproxy on
```

Create Compute Service Database and Identity Records.

Install Compute packages.

```
#> yum install -y openstack-nova-api \
   openstack-nova-conductor openstack-nova-scheduler \
   python-cinderclient
* Change file ownership:
#> chown -R root:nova api-paste.ini
#> chown -R root:nova nova.conf
#> chown -R root:nova policy.json
#> chown -R root:nova rootwrap.conf
#> chown -R nova:nova /var/log/nova/nova-api.log
* Start messagebus
#> service messagebus start
#> service messagebus status
#> chkconfig messagebus on

(/etc/nova/nova.conf)
[DEFAULT]
rpc_backend = nova.openstack.common.rpc.impl_qpid
my_ip = 172.29.87.229
auth_strategy =keystone
sql_connection = mysql://nova:<passwd>@172.29.87.229/nova
enabled_apis=ec2,osapi_compute
osapi_compute_listen=172.29.87.229
osapi_compute_listen_port=8774
metadata_listen=0.0.0.0
metadata_listen_port=8700
api_paste_config=/etc/nova/api-paste.ini
service_neutron_metadata_proxy=True
neutron_metadata_proxy_shared_secret=cisco123
novncproxy_host = 172.29.87.229
novncproxy_port=6080
network_api_class=nova.network.neutronv2.api.API
metadata_host=172.29.87.229
neutron_url=http://172.29.87.229:9696
neutron_admin_username=neutron
neutron_admin_password=cisco123
neutron_admin_tenant_name=services
neutron_admin_auth_url=http://172.29.87.229:35357/v2.0
security_group_api=nova
debug=true
qpid_hostname = 172.29.87.229
qpid_username = nova
qpid_password = cisco123
qpid_port = 5672
scheduler_default_filters=AllHostsFilter
firewall_driver=nova.virt.libvirt.firewall.IptablesFirewallDriver
allow_same_net_traffic=true
libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtHybridOVSBridgeDriver
novncproxy_base_url=http://172.29.87.229:6080/vnc_auto.html
vncserver_listen=0.0.0.0
vnc_enabled=true
vnc_keymap=en-us
[keystone_authtoken]
auth_host = 172.29.87.229
auth_protocol = http
auth_port = 35357
admin_user = nova
admin_tenant_name = services
admin_password = <password>
```

Compute Services to start and chkconfig.

```
#> service openstack-nova-api restart
#> service openstack-nova-conductor restart
#> service openstack-nova-console restart
#> service openstack-nova-consoleauth restart
#> service openstack-nova-metadata-api restart
#> service openstack-nova-novncproxy restart
#> service openstack-nova-scheduler  restart
#> chkconfig openstack-nova-api on
#> chkconfig openstack-nova-conductor on
#> chkconfig openstack-nova-console on
#> chkconfig openstack-nova-consoleauth on
#> chkconfig openstack-nova-metadata-api on
#> chkconfig openstack-nova-novncproxy on
#> chkconfig openstack-nova-scheduler  on
```

Openstack Block Storage.

Install and Configure Block Storage.

Volume Service Specific Configuration: The block storage driver used in this configuration is LVM. It uses a file mounted via a loop device where a LVM has been created.

Create a new logical volume.

```
#> lvcreate -L 800G -n lv_vol_ephemeral vg_node01
  lv_vol_ephemeral                            vg_node01      -wi-ao---- 800.00g
#> mkfs -t ext4 /dev/vg_node01/lv_vol_ephemeral
#> mkdir /os_scratch
```

Add entry in /etc/fstab.

```
"/dev/vg_node01/lv_vol_ephemeral /os_scratch     ext4    defaults        1 3"
#> mount /os_scratch
#> dd if=/dev/zero of=/os_scratch/cinder-volumes bs=1 count=0 seek=800G
#> losetup -fv /os_scratch/cinder-volumes
```

Check the loop device associated with /os_scratch/cinder-volumes.

```
#> losetup -a
/dev/loop0: [fd02]:13 (/os_scratch/cinder-volumes)
```

Create the volume group associated with it.

```
#> vgcreate cinder-volumes /dev/loop0
#> vgs
  VG             #PV #LV #SN Attr   VSize   VFree
  cinder-volumes   1   4   0 wz--n- 800.00g 715.00g
  vg_node01        1   3   0 wz--n- 931.02g  77.02g

#> echo "include /etc/cinder/volumes/*" >> /etc/tgt/targets.conf
#> yum install scsi-target-utils
#> service tgtd start
#> chkconfig tgtd on
```

(/etc/cinder/cinder.conf).

```
[DEFAULT]
auth_strategy = keystone
rpc_backend = cinder.openstack.common.rpc.impl_qpid
qpid_hostname = 172.29.87.229
qpid_username = cinder
qpid_password = <passwd>
sql_connection = mysql://cinder:<passwd>@172.29.87.229/cinder
service_down_time=180
volume_group=cinder-volumes
volume_driver=cinder.volume.drivers.lvm.LVMISCSIDriver
[keystone_authtoken]
auth_host = 172.29.87.229
admin_tenant_name = services
```

```
admin_user = cinder
admin_password = <passwd>
```

Validate the setup.

```
#> cinder create --display-name vol-test 1
+---------------------+--------------------------------------+
|       Property      |                Value                 |
+---------------------+--------------------------------------+
|     attachments     |                  []                  |
|  availability_zone  |                 nova                 |
|       bootable      |                false                 |
|      created_at     |      2014-02-06T17:42:16.996571      |
| display_description |                 None                 |
|     display_name    |               vol-test               |
|          id         | dea6e442-69af-4925-bb58-4decf301a13e |
|       metadata      |                  {}                  |
|         size        |                  1                   |
|     snapshot_id     |                 None                 |
|     source_volid    |                 None                 |
|        status       |               creating               |
|     volume_type     |                 None                 |
+---------------------+--------------------------------------+
#> lvs
  LV                                          VG             Attr       LSize   Pool
Origin Data%  Move Log Cpy%Sync Convert
  volume-dea6e442-69af-4925-bb58-4decf301a13e cinder-volumes -wi-ao----   1.00g
  ....
```

## Installing and Configuring Neutron Services

The neutron plugin used in this configuration is Open vSwitch (openstack-neutron-openvswitch). L3 agent abstracts the router that can connect to provide gateway services for L2 networks. The Compute node in this configuration hosts the network services for L3 agent, DHCP agent, and the neutron metadata agent that proxies to the nova metadata service. The neutron server is hosted on the Controller node and the L3-agent, L2-agent, DHCP agent and Metadata agent run on the compute node.

### Installing Networking Pre-requisites on the Controller

The following instructions provide guidance for installing networking pre-requisites on the controller.

[Create the Openstack Networking Database|[https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux_OpenStack_Platform/4/html/Installation_and_Configuration_Guide/sect-Networking_Prerequisite_Configuration.html]]

[Create the Networking identity Records|https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux_OpenStack_Platform/4/html/Installation_and_Configuration_Guide/Creating_the_Service_Endpoint.html]

Common Networking Configuration.

Disable Network Manager.

Install the relevant packages.

```
#> yum install -y openstack-neutron \
   openstack-utils \
   openstack-selinux
```

Configure the firewall (add entry to /etc/sysconfig/iptables).

```
"-A INPUT -p tcp -m multiport --dports 9696 -j ACCEPT"
```

```
#> service iptables restart
```

(/etc/neutron/neutron.conf).

```
[DEFAULT]
auth_strategy = keystone
rpc_backend = neutron.openstack.common.rpc.impl_qpid
qpid_hostname = 172.29.87.229
qpid_username = neutron
qpid_password = <passwd>
core_plugin = neutron.plugins.openvswitch.ovs_neutron_plugin.OVSNeutronPluginV2
ovs_use_veth = True
allow_overlapping_ips = True
debug = True
dhcp_lease_duration = 604800
dhcp_lease_time = 604800
[quotas]
[agent]
root_helper = sudo neutron-rootwrap /etc/neutron/rootwrap.conf
[keystone_authtoken]
auth_host = 172.29.87.229
admin_tenant_name = services
admin_user = neutron
admin_password = <passwd>
[database]
[service_providers]
Launch Networking Service
service neutron-server start
chkconfig neutron-server on
```

## Installing Horizon Dashboard

The following instructions provide guidance for installing Horizon Dashboard.

```
#> yum install -y mod_wsgi httpd
#> yum install -y memcached python-memcached
#> yum install -y openstack-dashboard
#> service httpd start
#> chkconfig httpd on
```

Check if httpd is running.

```
#> service --status-all | grep httpd
```

Configure connections and logging.

Edit /etc/openstack-dashboard/local_settings.

Configure local memory cache settings.

```
CACHES = {
    'default': {
        'BACKEND' : 'django.core.cache.backends.locmem.LocMemCache'
    }
}
OPENSTACK_HOST = "172.29.87.229"
OPENSTACK_KEYSTONE_URL = "http://%s:5000/v2.0" % OPENSTACK_HOST
OPENSTACK_KEYSTONE_DEFAULT_ROLE = "admin"
```

Restart httpd service.

```
#> service httpd restart
```

Configure SELinux.

Allow connections from httpd service to the Identity server if SELinux is configured in 'Enforcing or 'Permissive' mode

```
#> getenforce

Enforcing
#> setsebool -P httpd_can_network_connect on
```

Configure firewall (add the following line to /etc/sysconfig/iptables).

```
"-A INPUT -p tcp --dport 443 -j ACCEPT"
#> service iptables restart
```

Validate dashboard installation: http://172.29.87.229/dashboard.

Install Openstack Orchestration Service.

Install Openstack Telemetry Service (Ceilometer).

Install and Configure Compute and Network Node (node02.ctocllab.cisco.com (172.29.87.230).

## Installing and Configuring Compute Node

The following instructions provide guidance for installing and configuring compute node.

Register system with Red Hat Network for a pre-registered userid.

```
#> subscription-manager register
```

List available or consumed subscriptions for registered system.

```
#> subscription-manager list [--available|--consumed]
```

Attach the system to a given Pool Id for RHOS 4 (Havana).

```
Subscription Name: Red Hat Cloud Infrastructure Business Partner Self-Supported NFR
(4-sockets)
Provides:          Red Hat OpenStack
                   JBoss Enterprise Application Platform
                   Red Hat Enterprise Linux Server
                   Red Hat OpenStack Beta
                   Red Hat Enterprise Virtualization
                   Red Hat Enterprise MRG Messaging 2
                   Red Hat CloudForms
                   Red Hat Enterprise Linux 7 Public Beta
                   Red Hat Beta
SKU:               ...
Contract:          ...
Account:           ...
Serial:            ...
Pool ID:           <pool-id>
#> subscription-manager attach --pool=<pool-id>
```

Install yum-utils to enable relevant openstack rpms for openstack-4.0.

```
#> yum install -y yum-utils
#> yum install -y yum-plugin-priorities
#> yum-config-manager --enable rhel-6-server-openstack-4.0-rpms \
        --setopt="rhel-6-server-openstack-4.0-rpms.priority=1"
#> yum update -y
#> reboot
```

Check hardware virtualization support by checking presence of svm or vmx CPU extensions.

```
#> grep -E 'svm|vmx' /proc/cpuinfo
```

Verify kvm modules are loaded.

```
#> lsmod | grep kvm
```

Output must include kvm_intel or kvm_amd.

Check prerequisites.

```
#> yum install ntp
#> service ntpd start
#> chkconfig ntpd on
#> service libvirtd status
libvirtd (pid  2276) is running...
#> service messagebus status
```

Install nova-compute.

```
#> yum install openstack-nova-compute
#> chown root:nova /etc/nova/nova.conf
```

(/etc/nova/nova.conf).

```
[DEFAULT]
rpc_backend = nova.openstack.common.rpc.impl_qpid
my_ip = 172.29.87.230
auth_strategy =keystone
sql_connection = mysql://nova:<passwd>@172.29.87.229/nova
enabled_apis=ec2,osapi_compute,metadata
metadata_listen=0.0.0.0
metadata_listen_port=8775
api_paste_config=/etc/nova/api-paste.ini
service_neutron_metadata_proxy=True
neutron_metadata_proxy_shared_secret=<passwd>
novncproxy_port=6080
glance_host=172.29.87.229
glance_api_servers=$glance_host:$glance_port
network_api_class=nova.network.neutronv2.api.API
metadata_host=$my_ip
metadata_port=8775
neutron_url=http://172.29.87.229:9696
neutron_admin_username=neutron
neutron_admin_password=<passwd>
neutron_admin_tenant_name=services
neutron_admin_auth_url=http://172.29.87.229:35357/v2.0
security_group_api=neutron
debug=true
qpid_hostname = 172.29.87.229
qpid_username = nova
qpid_password = <passwd>
qpid_port = 5672
firewall_driver=nova.virt.firewall.NoopFirewallDriver
allow_same_net_traffic=true
libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtHybridOVSBridgeDriver
novncproxy_base_url=http://172.29.87.229:6080/vnc_auto.html
vncserver_listen=0.0.0.0
vncserver_proxyclient_address=172.29.87.230
vnc_enabled=true
vnc_keymap=en-us
instance_usage_audit = True
instance_usage_audit_period = hour
notify_on_state_change = vm_and_task_state
notification_driver=nova.openstack.common.notifier.rpc_notifier
notification_driver = ceilometer.compute.nova_notifier
[keystone_authtoken]
```

```
auth_host = 172.29.87.229
auth_protocol = http
auth_port = 35357
admin_user = nova
admin_tenant_name = services
admin_password = <password>
```

Start the compute service.

```
#> service openstack-nova-compute start
#> chkconfig openstack-nova-compute on
```

Install and Configure Network.

```
#> yum install openstack-neutron openstack-neutron-openvswitch
#> yum install bridge-utils -y
```

Verify that openvswitch package is installed.

```
#> rpm -qa | grep openvswitch
Start openvswitch service
#> service openvswitch start
#> chkconfig openvswitch on
```

The host running Open vSwitch agent requires that the ovs bridge named br-int be created.

```
#> ovs-vsctl add-br br-int
```

Configure external network access by creating an external bridge.

```
#> ovs-vsctl add-br br-ex
```

The external bridge to the interface on the compute node (ensure it is running in promiscuous mode).

```
Interface eth1 config: ifcfg-eth1
DEVICE=eth1
TYPE=Ethernet
ONBOOT=yes
NM_CONTROLLED=no
BOOTPROTO=none
PROMISC=yes

#> ovs-vsctl add-port br-ex eth1
```

(/etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini)

```
[securitygroup]
firewall_driver =
neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver
[OVS]
tenant_network_type = vlan
network_vlan_ranges = physnet1:885:886:887
bridge_mappings = physnet1:br-ex
[DATABASE]
sql_connection = mysql://neutron:<passwd>@172.29.87.229/ovs_neutron
[SECURITYGROUP]
firewall_driver =
neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver
/etc/neutron/l3-agent.ini
[DEFAULT]
debug = False
interface_driver = neutron.agent.linux.interface.OVSInterfaceDriver
ovs_use_veth = True
use_namespaces = True
metadata_ip = 172.29.87.229
metadata_port = 8700
```

```
(/etc/neutron/dhcp-agent.ini)
[DEFAULT]
debug = True
interface_driver = neutron.agent.linux.interface.OVSInterfaceDriver
dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq
enable_isolated_metadata = True
root_helper = sudo neutron-rootwrap /etc/neutron/rootwrap.conf

(/etc/neutron/metadata.ini)
[DEFAULT]
debug = True
auth_url = http://172.29.87.229:35357/v2.0
auth_region = regionOne
admin_tenant_name = services
admin_user = neutron
admin_password = <passwd>
nova_metadata_ip = 172.29.87.229
nova_metadata_port = 8700
metadata_proxy_shared_secret = <passwd>

(/etc/neutron/neutron.conf)
[DEFAULT]
auth_strategy = keystone
rpc_backend = neutron.openstack.common.rpc.impl_qpid
qpid_hostname = 172.29.87.229
qpid_username = neutron
qpid_password = <passwd>
core_plugin = neutron.plugins.openvswitch.ovs_neutron_plugin.OVSNeutronPluginV2
ovs_use_veth = True
allow_overlapping_ips = True
debug = True
[quotas]
[agent]
root_helper = sudo neutron-rootwrap /etc/neutron/rootwrap.conf
[keystone_authtoken]
auth_host = 172.29.87.229
admin_tenant_name = services
admin_user = neutron
admin_password = <passwd>
[database]
[service_providers]
```

Start the network services and chkconfig on.

```
#> service neutron-dhcp-agent start
#> service neutron-l3-agent start
#> service neutron-metadata-agent start
#> service neutron-openvswitch-agent start
#> chkconfig neutron-dhcp-agent on
#> chkconfig neutron-l3-agent on
#> chkconfig neutron-metadata-agent on
#> chkconfig neutron-openvswitch-agent on
```

## Red Hat OpenShift Deployment

The ability to spin up OpenShift Node infrastructure on-demand forms the basis for advanced features such as auto-scaling the PaaS infrastructure.

When deployed on OpenStack, OpenShift is able to provision and de-provision Node VMs without manual provisioning steps using Heat templates.

This provides the ability to auto-scale up and down. Currently scaling decisions are made within the broker using platform agnostic scaling scripts. OpenShift applies a push approach in which it initiates the scale events based on the information is has already gathered.

As the Node comes on line and starts the MCollective service, it becomes available to pick up messages from the broker.

## Installing the Broker and Node Infrastructure

The following instructions provide guidance for installing the broker and node infrastructure.

Download the latest qcow2 RHEL 6.5 Guest Image from RHN.

Red Hat Common (for RHEL 6 Server x86_64).

Download the latest heat templates for OpenShift Enterprise (Broker and Node).

Use diskimage-builder to prepare the RHEL 6.5 image.

```
#> git clone https://github.com/openstack/diskimage-builder.git
```

The heat templates used to spin up the broker and the node will require enterprise licenses and pool ids for subscriptions to register the systems for the broker and the node at the time of instantiation.

Prepare to run diskimage-builder.

```
#> mkdir $HOME/tmp
```

Export path to the do DIB elements for OpenShift Enterprise.

```
#> export
ELEMENTS_PATH=heat-templates/elements:heat-templates/openshift-enterprise/dib/elements
```

Host the downloaded file on a local httpd server since builder uses an http endpoint to download the image and remote image locations on RHN tend to update the ISO images on frequently.

```
#> export DIB_CLOUD_IMAGES=http://localhost/
#> export DIB_RHSM_OSE_POOL=<ose-pool-id>
#> export DIB_RHSM_POOL=<ose-pool-id>
#> export DIB_RHSM_USER=<registered user>
#> export DIB_RHSM_PASSWORD=<password>
#> export TMP_DIR=$HOME/tmp
#> export DIB_IMAGE_SIZE=10
#> export DIB_OSE_VERSION=2.0
#> export DIB_YUM_VALIDATOR_VERSION=2.0
```

Do not set DIB_RHSM_USER and DIB_RHSM_PASSWORD. It fails during subscription manager register phase. The OSE Pool Id should suffice.

Unit subscriptions do not get removed from systems that fail builds in the midst. The outcome is that at some point, the build will exhaust them and error out with no more subscriptions available from pool.

Remove subscriptions attached to the build server from https://access.redhat.com/management/consumers (All Units)

Bug tracked at https://bugzilla.redhat.com/show_bug.cgi?id=1004483

Run the diskimage-builder for the broker to generate a rhel image using DIBs for openshift-enterprise-broker

```
#> diskimage-builder/bin/disk-image-create --no-tmpfs -a amd64 vm rhel
openshift-enterprise-broker -o RHEL65upgraded-x86_64-broker-v3
```

Run the diskimage-builder for the node to generate a rhel image using DIBs for openshift-enterprise-node.

```
#> diskimage-builder/bin/disk-image-create --no-tmpfs -a amd64 vm rhel
openshift-enterprise-node -o RHEL65-x86_64-node-v2

Upload the Broker image (RHEL65upgraded-x86_64-broker-v3.qcow2) to the target
OpenStack deployment
#> glance image-create --name=RHEL65upgraded-x86_64-broker-v3.qcow2
--disk-format=qcow2 --container-format=bare --is-public=true <
RHEL65upgraded-x86_64-broker-v3.qcow2

Upload the Node image (RHEL65-x86_64-node-v2.qcow2) to the target OpenStack deployment
#> glance image-create --name=RHEL65-x86_64-node-v2 --disk-format=qcow2
--container-format=bare --is-public=true < RHEL65-x86_64-node-v2.qcow2

Console root login has been disabled for the images. Used virt-sysprep to inject the
passward at first-boot. The virt-sysprep utility is part of the ibguestfs-tools-c
package.
#> virt-sysprep --firstboot <script-file.sh>  -a <image-filename>

At the end of the image build, the umount2: Invalid argument, umount:
/root/tmp/image.BKl7ZzO2: not mounted error can be ignored
```

Openstack VM boot time console logs do not show up, and needs to be added to grub command line in the image, however grub is currently not enabled on rhel 6.5, but is in progress.

Openstack Metadata service is exclusively used by cloud-init scripts in the broker/node images to run user data passed on during boot time. Enabling config drive has no effect. In fact, the broker wait conditions also exclusively use the Metadata service of Openstack to communicate cloud-init completion.

## Automated Deployments using Heat Templates

Spin up the broker and the node instances using subscription manager on a given network with heat cli

```
#> heat create openshift-1
--template-url=http://172.29.87.229/heat-templates/OpenShift-1B1N-neutron-cisco.yaml \
--parameters="key_name=<keypair>;prefix=<domain name>;upstreamDNS=<dns ip address>;\
broker_image_name=RHEL65upgraded-x86_64-broker-v2;node_image_name=RHEL65-x86_64-node-v
2;\
BrokerHostname=ose-broker-1.ctocllab.cisco.com;NodeHostname=ose-node-1.ctocllab.cisco.
com;\
ConfInstallMethod=rhsm;ConfSMRegName=<rhn-register-user>;ConfSMRegPass=<password>;\
ConfSMRegPool=<ose-poold-id>;private_net_id=<uuid-of-internal-neutron-network>;\
public_net_id=uuid-of-external-neutron-networ;private_subnet_id=<uuid-of-internal-neut
ron-subnet>;\
ose_version=2.0;yum_validator_version=2.0"
```

The broker and node will take several minutes to complete cloud-init configuration.

Once completed, access the VM via the VNC console and run oo-diagnostics on the broker and the node.

## OpenShift AD Configuration

Perform the following procedure to configure OpenShift AD.

**Step 1** Log on to the OpenShift broker server.

**Step 2** Edit broker HTTPD configuration file at:

/var/www/openshift/broker/httpd/conf.d/openshift-origin-auth-remote-user.conf

**Step 3** Add the following configuration:

```
LoadModule ldap_module modules/mod_ldap.so
LoadModule authnz_ldap_module modules/mod_authnz_ldap.so
<Location /broker>
    AuthName "OpenShift"
    AuthType Basic
    AuthBasicProvider ldap
    AuthLDAPURL
"ldap://<AD-SERVER>:389/DC=CISCODEMO,DC=local?sAMAccountName?sub?(objectClass=*)" NONE
    AuthLDAPBindDN "Administrator@ciscodemo.local"
    AuthLDAPBindPassword "PASSWORD"
    require valid-user
```

**Step 4** Restart OpenShift broker service:

```
#> service openshift-broker restart
```

**Step 5** Edit console configuration file at:
/var/www/openshift/console/httpd/conf.d/openshift-origin-auth-remote-user.conf

**Step 6** Add the following configuration:

```
LoadModule ldap_module modules/mod_ldap.so
LoadModule authnz_ldap_module modules/mod_authnz_ldap.so
<Location /console>
    AuthName "OpenShift"
    AuthType Basic
    AuthBasicProvider ldap
    AuthLDAPURL
"ldap://<AD-SERVER>:389/DC=CISCODEMO,DC=local?sAMAccountName?sub?(objectClass=*)" NONE
    AuthLDAPBindDN "Administrator@ciscodemo.local"
    AuthLDAPBindPassword "PASSWORD"
    require valid-user
```

**Step 7** Restart OpenShift console service:

```
#> service openshift-console restart
```

## Creating OpenShift District

Districts define a set of node hosts within which gears can be reliably moved to manage the resource usage of those nodes. While not strictly required for a basic OpenShift installation, their use is recommended where administrators might ever need to move gears between nodes. It's also possible to create multiple districts and designate different security level on each district by leveraging network firewall and access control policies associated with node network address.

To use districts, the broker's MCollective plugin must be configured to enable districts.

**Step 1** Edit the /etc/openshift/plugins.d/openshift-origin-msg-broker-mcollective.conf configuration file and confirm the following parameters are set:

```
DISTRICTS_ENABLED=true
NODE_PROFILE_ENABLED=true
DISTRICTS_REQUIRE_FOR_APP_CREATE=true
```

**Step 2** Execute the following command on the broker host:

```
#> oo-admin-ctl-district -c create -n smal_district -p small
```

Where small_district is the name of the new district and small is the profile of the gears that will be provisioned on this district.

**Step 3** Add node host to small_district that was created:

```
#> oo-admin-ctl-district -c add-node -n small_district -i node.example.com
```

It is important to note that the server identity (node.example.com here) is the node's hostname as configured on that node, which could be different from the PUBLIC_HOSTNAME configured in /etc/openshift/node.conf on the node. The PUBLIC_HOSTNAME is used in CNAME records and must resolve to the host via DNS; the hostname could be something completely different and may not resolve in DNS at all.

## CIAC Configuration

CIAC needs to be configured to create services and portals outlined in the design section. The following instructions lay out the necessary configurations.

### Active Directory Integration

Perform the following procedure to create integrate active directory.

**Step 1** Launch Cisco Service Portal in browser with administration user.

**Step 2** Sample URL: http://<PSC-HOST>/RequestCenter.

**Step 3** From the Administration module in the module selection drop down, click **Directories**.

**Step 4** Click Add to add a new data source.

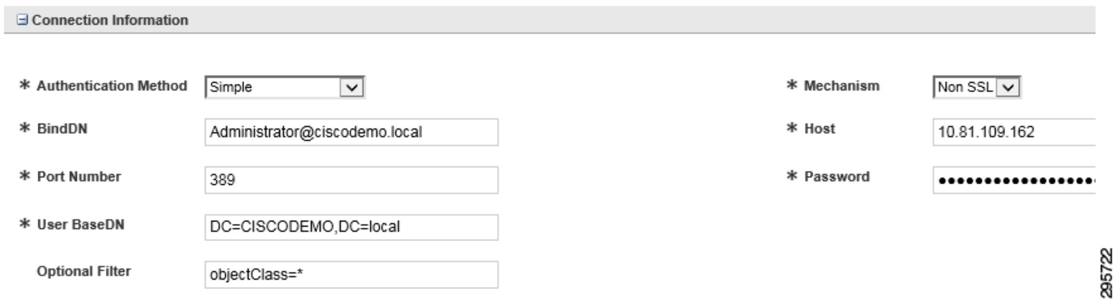
**Step 5** Enter a name for Data Source Name.

**Step 6** Select LDAP for the Protocol.

**Step 7** Select MS Active Directory for Server Product.

**Step 8** Set the connection information, as shown in Figure 4-1.

*Figure 4-1 AD Connection*



Connection Information

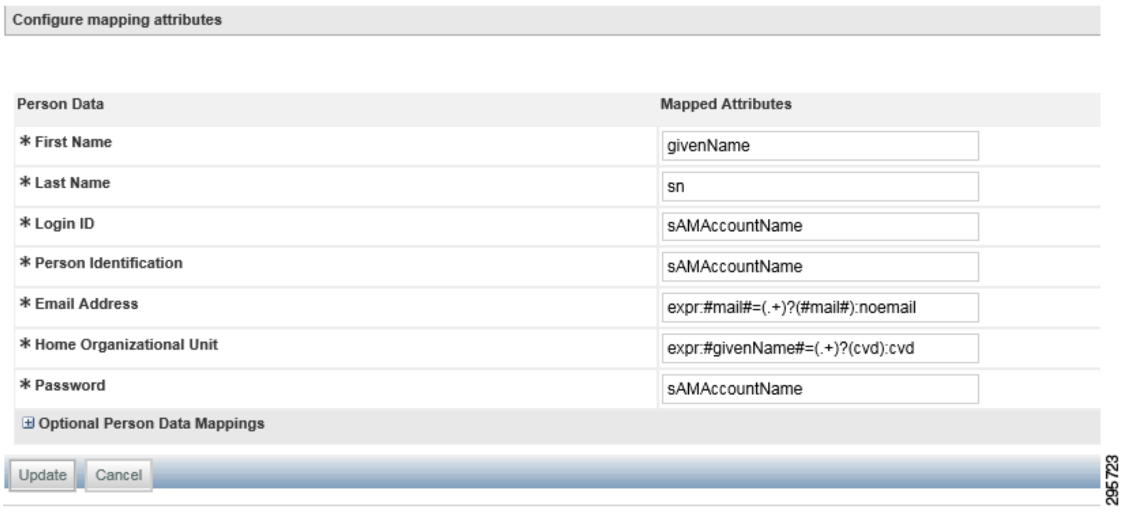
| | | | |
|---|---|---|---|
| * Authentication Method | Simple | * Mechanism | Non SSL |
| * BindDN | Administrator@ciscodemo.local | * Host | 10.81.109.162 |
| * Port Number | 389 | * Password | ••••••••••••••••• |
| * User BaseDN | DC=CISCODEMO,DC=local | | |
| Optional Filter | objectClass=* | | |

**Step 9** Click the Mappings tab (right side).

**Step 10** Enter a mapping name.

**Step 11** Configure mapping attributes, as shown in Figure 4-2.

*Figure 4-2 Configure Mapping Attributes*


Configure mapping attributes

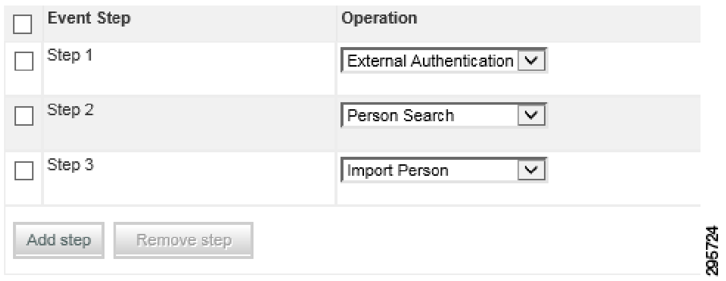
| Person Data | Mapped Attributes |
|---|---|
| * First Name | givenName |
| * Last Name | sn |
| * Login ID | sAMAccountName |
| * Person Identification | sAMAccountName |
| * Email Address | expr:#mail#=(.+)?(#mail#):noemail |
| * Home Organizational Unit | expr:#givenName#=(.+)?(cvd):cvd |
| * Password | sAMAccountName |

**Step 12** Click the Events tab (right side).

**Step 13** Click Edit at Login event.

**Step 14** Add the steps as shown in Figure 4-3.

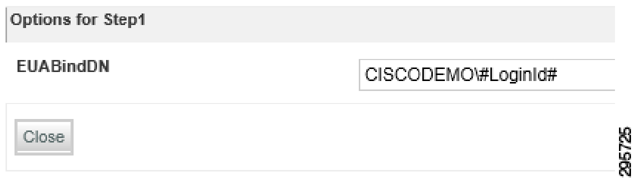
*Figure 4-3 Add Event Steps*



| ☐ | Event Step | Operation |
|---|---|---|
| ☐ | Step 1 | External Authentication |
| ☐ | Step 2 | Person Search |
| ☐ | Step 3 | Import Person |

a. For Step 1, click Options and set the value as shown in Figure 4-4.

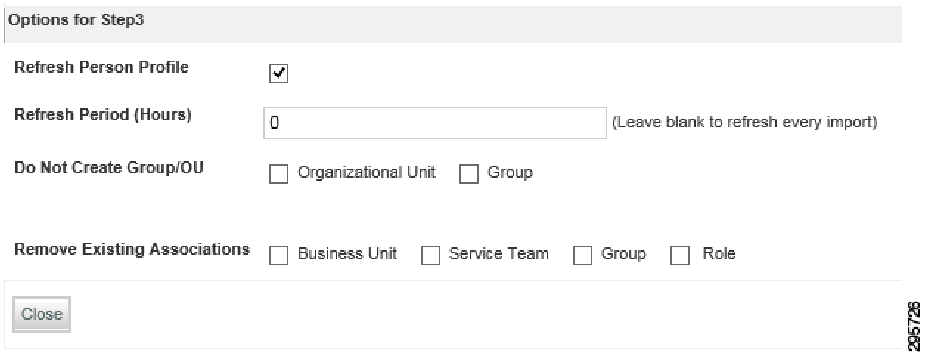
*Figure 4-4 Set Values for Step 1*


Options for Step1

EUABindDN: CISCODEMO\#LoginId#

b. For Step 3, click Options and set the value as shown in Figure 4-5.

*Figure 4-5 Set Values for Step 3*

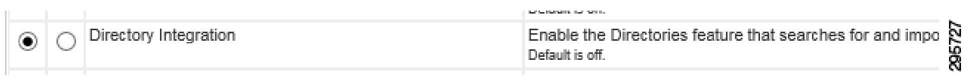

Options for Step3

| | |
|---|---|
| Refresh Person Profile | ✓ |
| Refresh Period (Hours) | 0 (Leave blank to refresh every import) |
| Do Not Create Group/OU | ☐ Organizational Unit ☐ Group |
| Remove Existing Associations | ☐ Business Unit ☐ Service Team ☐ Group ☐ Role |

**Step 15** Click the Settings tab (on the top).

**Step 16** Turn on directory integration by selecting the radio button, as shown in Figure 4-6, and then click Update.

*Figure 4-6 Turn on Directory Integration*

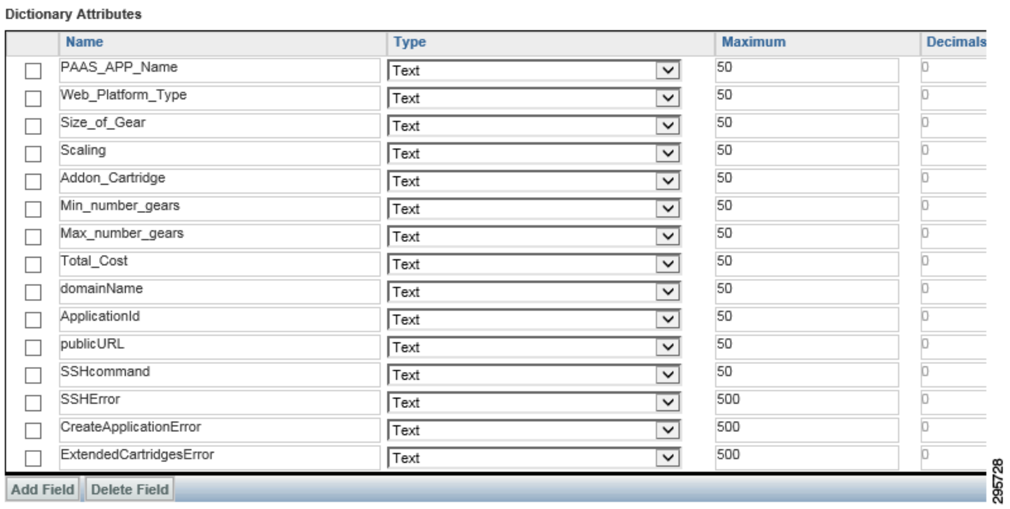

● ○ Directory Integration — Enable the Directories feature that searches for and impo
Default is off.

## Creating a New "PAAS Application" Service

Perform the following procedure to create a new PAAS application service.

**Step 1** Create a dictionary using the interface shown in Figure 4-7.

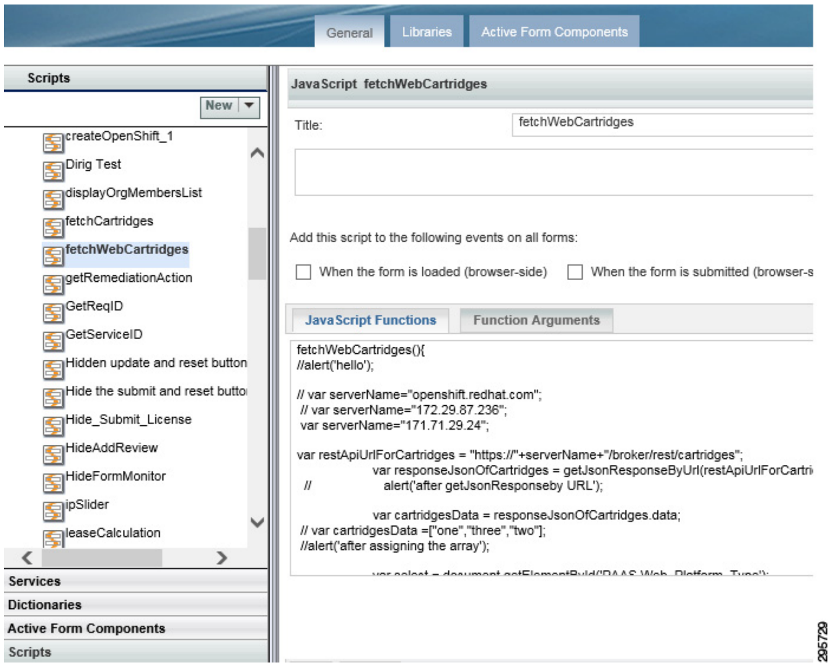
*Figure 4-7 Create a Dictionary*


Dictionary Attributes

| | Name | Type | Maximum | Decimals |
|---|---|---|---|---|
| ☐ | PAAS_APP_Name | Text | 50 | 0 |
| ☐ | Web_Platform_Type | Text | 50 | 0 |
| ☐ | Size_of_Gear | Text | 50 | 0 |
| ☐ | Scaling | Text | 50 | 0 |
| ☐ | Addon_Cartridge | Text | 50 | 0 |
| ☐ | Min_number_gears | Text | 50 | 0 |
| ☐ | Max_number_gears | Text | 50 | 0 |
| ☐ | Total_Cost | Text | 50 | 0 |
| ☐ | domainName | Text | 50 | 0 |
| ☐ | ApplicationId | Text | 50 | 0 |
| ☐ | publicURL | Text | 50 | 0 |
| ☐ | SSHcommand | Text | 50 | 0 |
| ☐ | SSHError | Text | 500 | 0 |
| ☐ | CreateApplicationError | Text | 500 | 0 |
| ☐ | ExtendedCartridgesError | Text | 500 | 0 |

**Step 2** Create the necessary Javascripts as shown in Figure 4-8, Figure 4-9, and Figure 4-10.

*Figure 4-8 Create the Necessary Javascripts 1*



JavaScript fetchWebCartridges

Title: fetchWebCartridges

Add this script to the following events on all forms:

☐ When the form is loaded (browser-side) ☐ When the form is submitted (browser-s

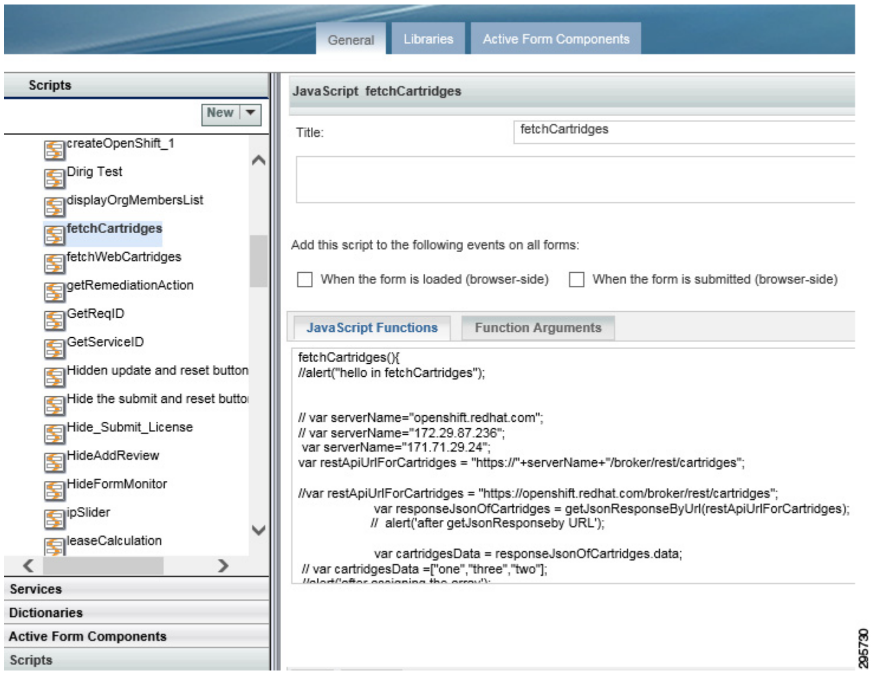
```
fetchWebCartridges(){
//alert('hello');

// var serverName="openshift.redhat.com";
 // var serverName="172.29.87.236";
 var serverName="171.71.29.24";

var restApiUrlForCartridges = "https://"+serverName+"/broker/rest/cartridges";
                  var responseJsonOfCartridges = getJsonResponseByUrl(restApiUrlForCartri
 //                 alert('after getJsonResponseby URL');

                  var cartridgesData = responseJsonOfCartridges.data;
 // var cartridgesData =["one","three","two"];
 //alert('after assigning the array');
```

*Figure 4-9 Create the Necessary Javascripts 2*



JavaScript fetchCartridges

Title: fetchCartridges

Add this script to the following events on all forms:

☐ When the form is loaded (browser-side) ☐ When the form is submitted (browser-side)

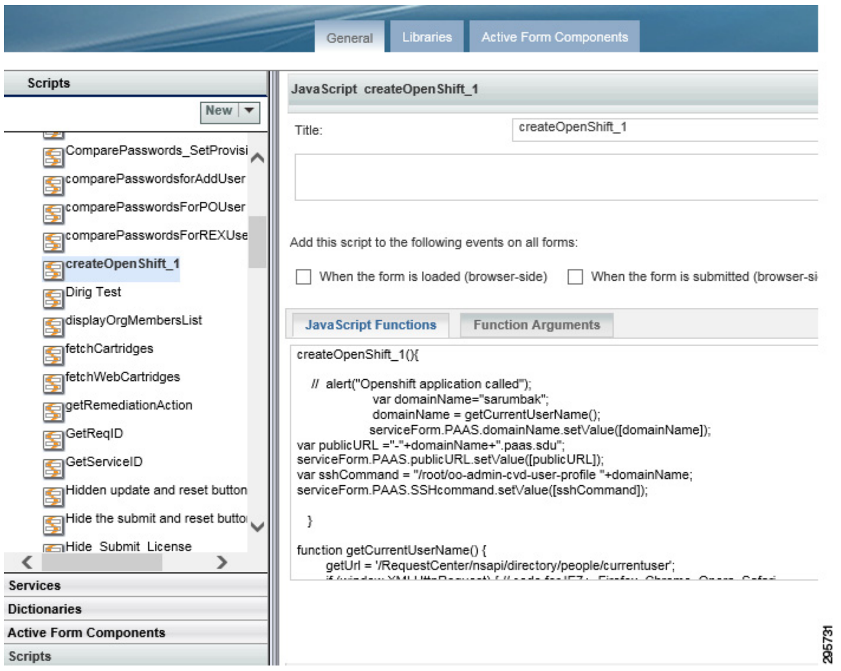
```
fetchCartridges(){
//alert("hello in fetchCartridges");


// var serverName="openshift.redhat.com";
// var serverName="172.29.87.236";
 var serverName="171.71.29.24";
var restApiUrlForCartridges = "https://"+serverName+"/broker/rest/cartridges";

//var restApiUrlForCartridges = "https://openshift.redhat.com/broker/rest/cartridges";
                  var responseJsonOfCartridges = getJsonResponseByUrl(restApiUrlForCartridges);
                 //  alert('after getJsonResponseby URL');

                  var cartridgesData = responseJsonOfCartridges.data;
 // var cartridgesData =["one","three","two"];
```

*Figure 4-10 Create the Javascripts Needed 3*



JavaScript createOpenShift_1

Title: createOpenShift_1

Add this script to the following events on all forms:

☐ When the form is loaded (browser-side) ☐ When the form is submitted (browser-si

```
createOpenShift_1(){

   //  alert("Openshift application called");
               var domainName="sarumbak";
               domainName = getCurrentUserName();
              serviceForm.PAAS.domainName.setValue([domainName]);
var publicURL ="-"+domainName+".paas.sdu";
serviceForm.PAAS.publicURL.setValue([publicURL]);
var sshCommand = "/root/oo-admin-cvd-user-profile "+domainName;
serviceForm.PAAS.SSHcommand.setValue([sshCommand]);


   }

function getCurrentUserName() {
       getUrl = '/RequestCenter/nsapi/directory/people/currentuser';
```

## Creating an Active Form Component

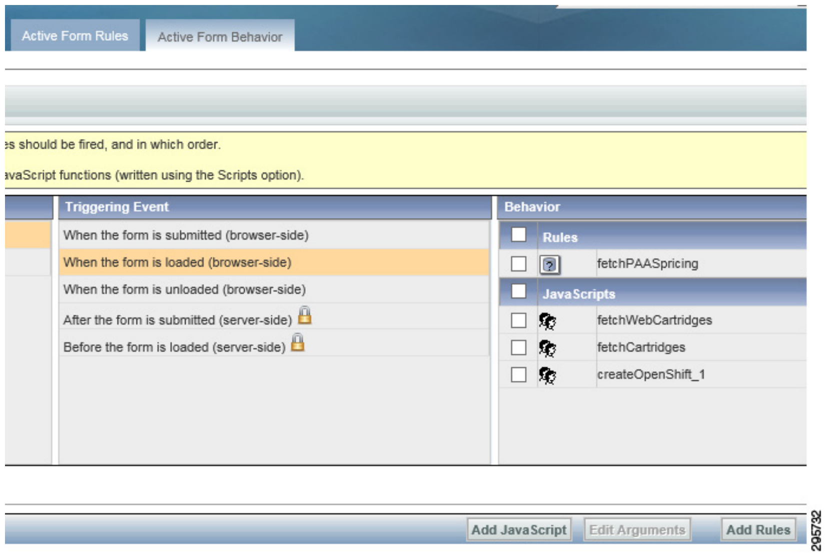
Perform the following procedure to create an active form component.

**Step 1** Add the dictionary that was created in the first step.

**Step 2** Set the display properties of each field.

**Step 3** Set the Active Form behavior; add the scripts (that were created in the previous step), as shown in Figure 4-11:

*Figure 4-11 Add Scripts*



ɜs should be fired, and in which order.

ɪvaScript functions (written using the Scripts option).

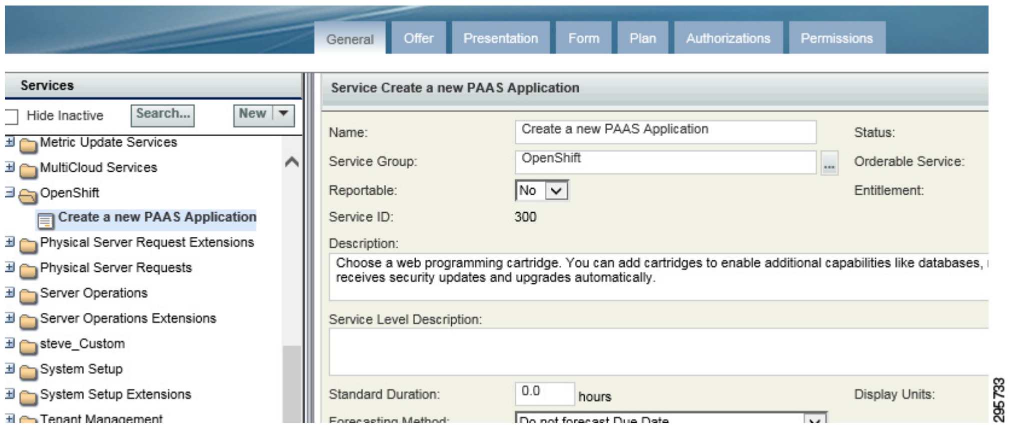
| Triggering Event | Behavior |
|---|---|
| When the form is submitted (browser-side) | Rules |
| When the form is loaded (browser-side) | fetchPAASpricing |
| When the form is unloaded (browser-side) | JavaScripts |
| After the form is submitted (server-side) | fetchWebCartridges |
| Before the form is loaded (server-side) | fetchCartridges |
| | createOpenShift_1 |

## Creating a Service

Perform the following procedure to create a service.

**Step 1** Create a service with desired name and set the description as shown in Figure 4-12.

*Figure 4-12 Create a Service*



Service Create a new PAAS Application

| | | |
|---|---|---|
| Name: | Create a new PAAS Application | Status: |
| Service Group: | OpenShift | Orderable Service: |
| Reportable: | No | Entitlement: |
| Service ID: | 300 | |

Description:
Choose a web programming cartridge. You can add cartridges to enable additional capabilities like databases,
receives security updates and upgrades automatically.

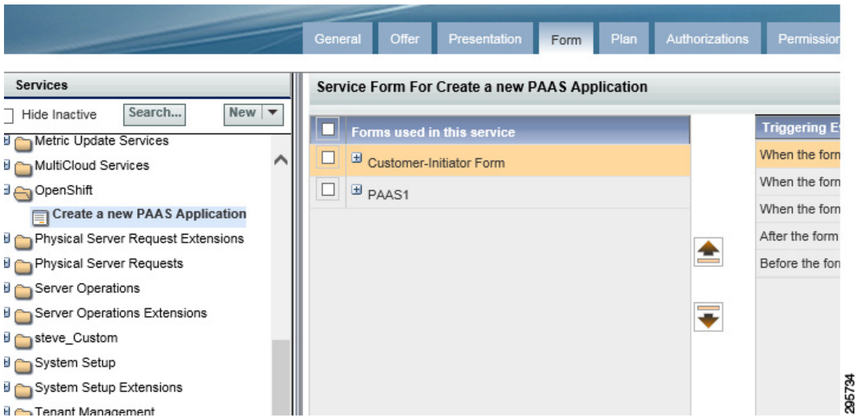
Service Level Description:

Standard Duration: 0.0 hours Display Units:

Forecasting Method: Do not forecast Due Date

**Step 2** Add the Active form component that was created to this service, as shown in Figure 4-13.

*Figure 4-13 Add Active Form Component*



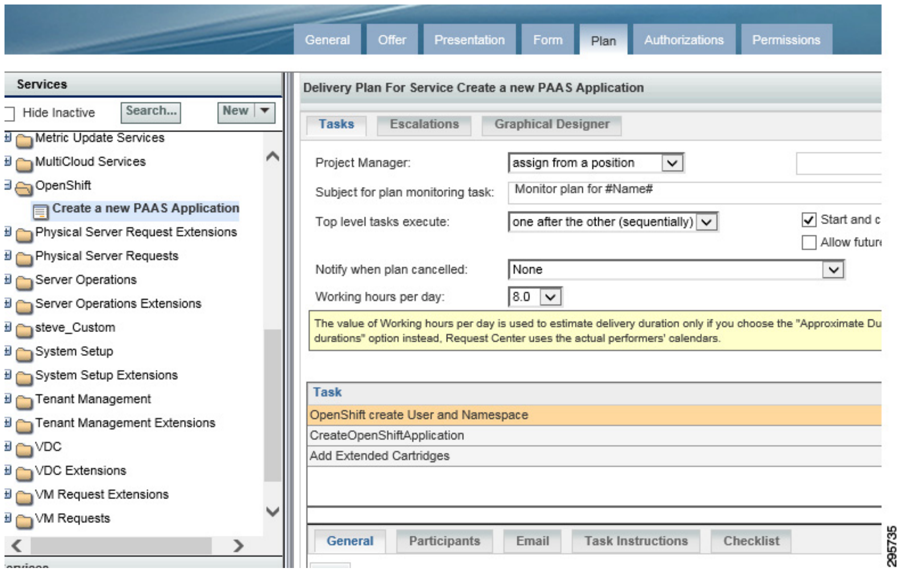
Service Form For Create a new PAAS Application

| Forms used in this service |
|---|
| Customer-Initiator Form |
| PAAS1 |

When the form
When the form
When the form
After the form
Before the form

**Step 3** Set the delivery plan, as shown in Figure 4-14.

*Figure 4-14 Set Delivery Plan*



Delivery Plan For Service Create a new PAAS Application

| | |
|---|---|
| Project Manager: | assign from a position |
| Subject for plan monitoring task: | Monitor plan for #Name# |
| Top level tasks execute: | one after the other (sequentially) |
| Notify when plan cancelled: | None |
| Working hours per day: | 8.0 |

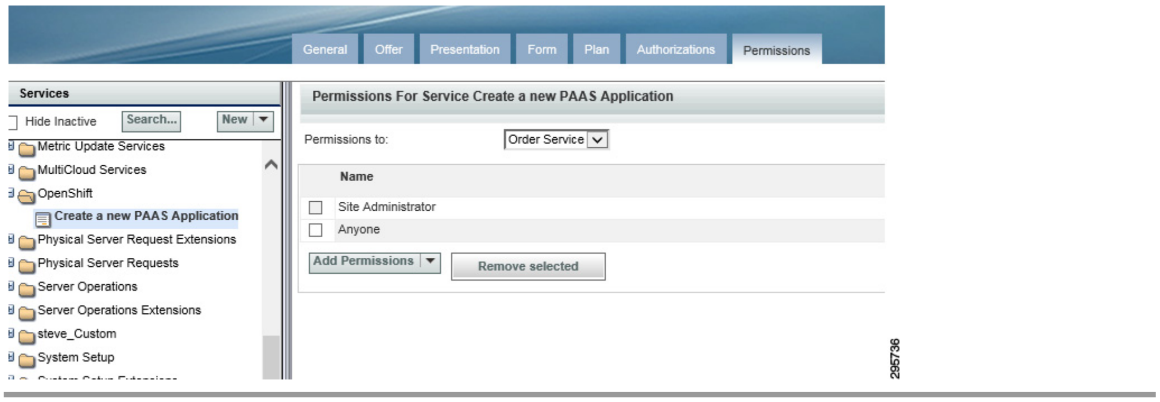
[✓] Start and c
[ ] Allow futur

The value of Working hours per day is used to estimate delivery duration only if you choose the "Approximate Du durations" option instead, Request Center uses the actual performers' calendars.

| Task |
|---|
| OpenShift create User and Namespace |
| CreateOpenShiftApplication |
| Add Extended Cartridges |

**Step 4** Set permissions, as shown in Figure 4-15.

*Figure 4-15 Set Permissions*



Permissions For Service Create a new PAAS Application

Permissions to: Order Service

| | Name |
|---|---|
| ☐ | Site Administrator |
| ☐ | Anyone |

## ServiceLink Agent Configuration

Three ServiceLink agents need to be configured.

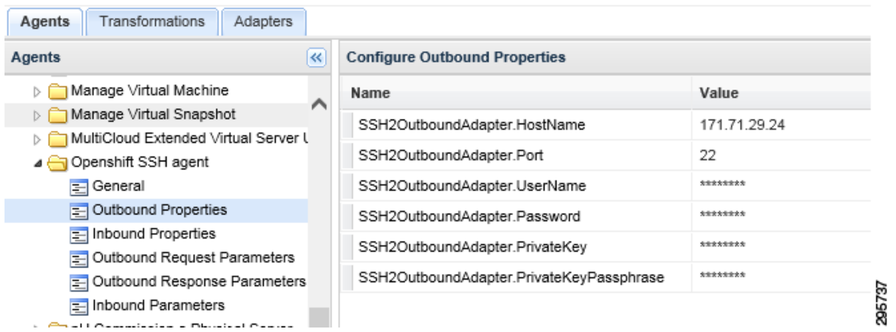
- OpenShift SSH Agent, page 4-24
- Creating an OpenShift Application Agent, page 4-25
- Add Extended Cartridges Agent, page 4-27

### OpenShift SSH Agent

Perform the following procedure to configure OpenShift SSH Agent.

**Step 1** Configure ServiceLink agent Outbound Properties, as shown in Figure 4-16.

*Figure 4-16 Configure ServiceLink Agent Outbound Properties*

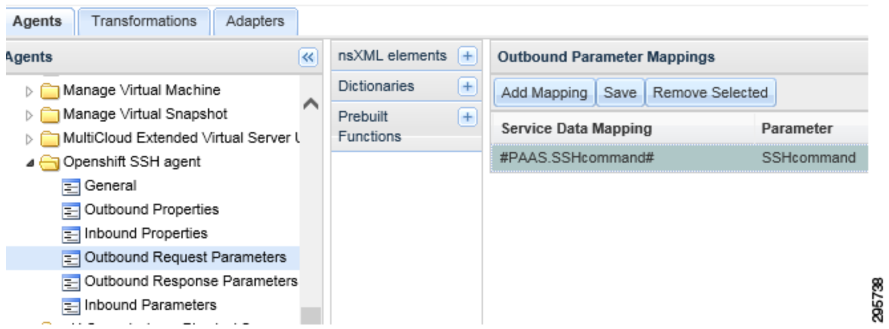


Configure Outbound Properties

| Name | Value |
|---|---|
| SSH2OutboundAdapter.HostName | 171.71.29.24 |
| SSH2OutboundAdapter.Port | 22 |
| SSH2OutboundAdapter.UserName | ******** |
| SSH2OutboundAdapter.Password | ******** |
| SSH2OutboundAdapter.PrivateKey | ******** |
| SSH2OutboundAdapter.PrivateKeyPassphrase | ******** |

**Step 2** Configure Outbound Request parameters, as shown in Figure 4-17.

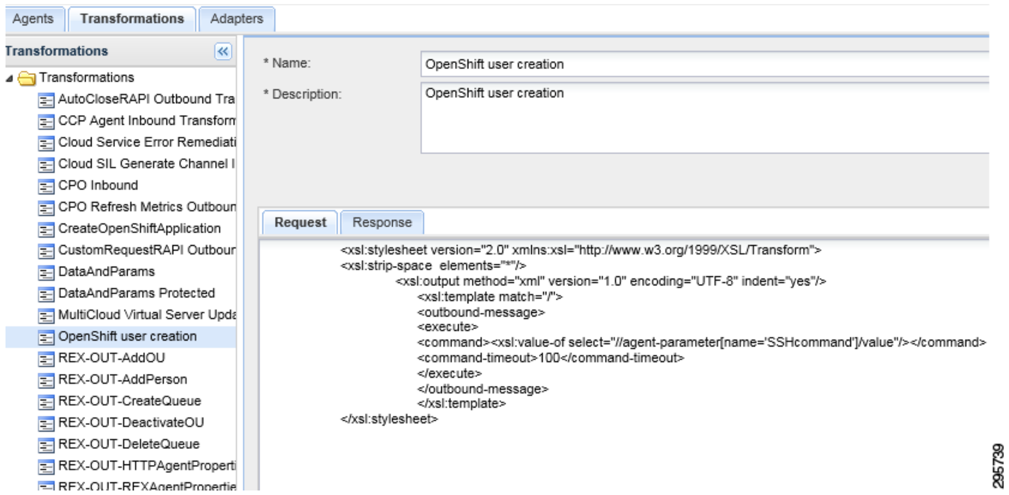
*Figure 4-17 Configure Outbound Request Parameters*



| Service Data Mapping | Parameter |
|---|---|
| #PAAS.SSHcommand# | SSHcommand |

**Step 3** Configure Outbound Transformation request, as shown in Figure 4-18.

*Figure 4-18 Configure Outbound Transformation Request*


Agents
Transformations
Adapters
Transformations
Transformations
AutoCloseRAPI Outbound Tra
CCP Agent Inbound Transform
Cloud Service Error Remediati
Cloud SIL Generate Channel I
CPO Inbound
CPO Refresh Metrics Outboun
CreateOpenShiftApplication
CustomRequestRAPI Outbour
DataAndParams
DataAndParams Protected
MultiCloud Virtual Server Upda
OpenShift user creation
REX-OUT-AddOU
REX-OUT-AddPerson
REX-OUT-CreateQueue
REX-OUT-DeactivateOU
REX-OUT-DeleteQueue
REX-OUT-HTTPAgentProperti
Request
Response
295739

* Name: OpenShift user creation

* Description: OpenShift user creation

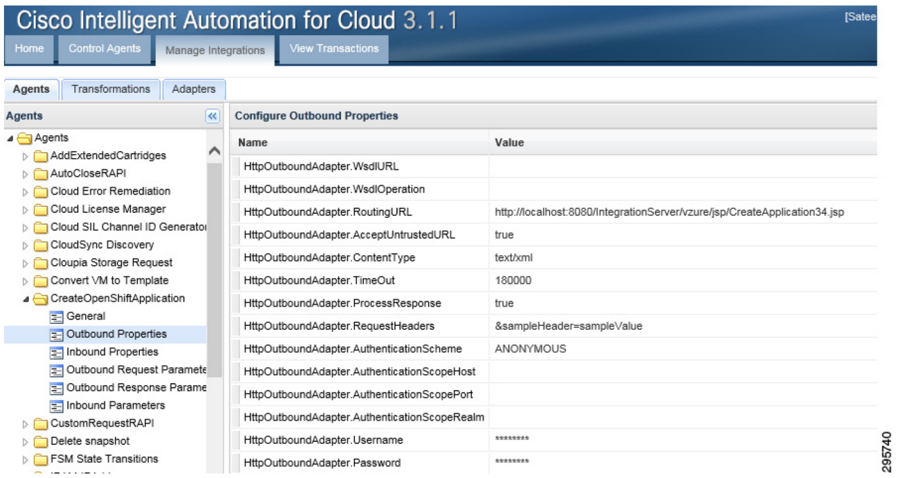
```
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:strip-space  elements="*"/>
        <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
           <xsl:template match="/">
           <outbound-message>
           <execute>
           <command><xsl:value-of select="//agent-parameter[name='SSHcommand']/value"/></command>
           <command-timeout>100</command-timeout>
           </execute>
           </outbound-message>
           </xsl:template>
</xsl:stylesheet>
```

## Creating an OpenShift Application Agent

Perform the following procedure to create an OpenShift application agent.

**Step 1** Configure ServiceLink agent Outbound Properties, as shown in Figure 4-19.

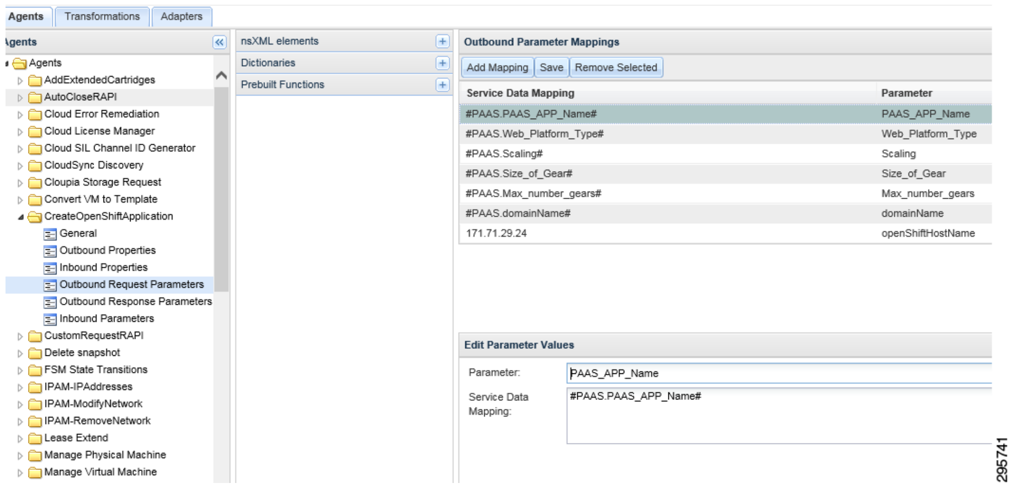
*Figure 4-19* ***Configure ServiceLink Agent Outbound Properties***



| Name | Value |
|---|---|
| HttpOutboundAdapter.WsdlURL | |
| HttpOutboundAdapter.WsdlOperation | |
| HttpOutboundAdapter.RoutingURL | http://localhost:8080/IntegrationServer/vzure/jsp/CreateApplication34.jsp |
| HttpOutboundAdapter.AcceptUntrustedURL | true |
| HttpOutboundAdapter.ContentType | text/xml |
| HttpOutboundAdapter.TimeOut | 180000 |
| HttpOutboundAdapter.ProcessResponse | true |
| HttpOutboundAdapter.RequestHeaders | &sampleHeader=sampleValue |
| HttpOutboundAdapter.AuthenticationScheme | ANONYMOUS |
| HttpOutboundAdapter.AuthenticationScopeHost | |
| HttpOutboundAdapter.AuthenticationScopePort | |
| HttpOutboundAdapter.AuthenticationScopeRealm | |
| HttpOutboundAdapter.Username | ******** |
| HttpOutboundAdapter.Password | ******** |

**Step 2** Configure Outbound Request parameters, as shown in Figure 4-20.

*Figure 4-20* ***Configure Outbound Request Parameters***



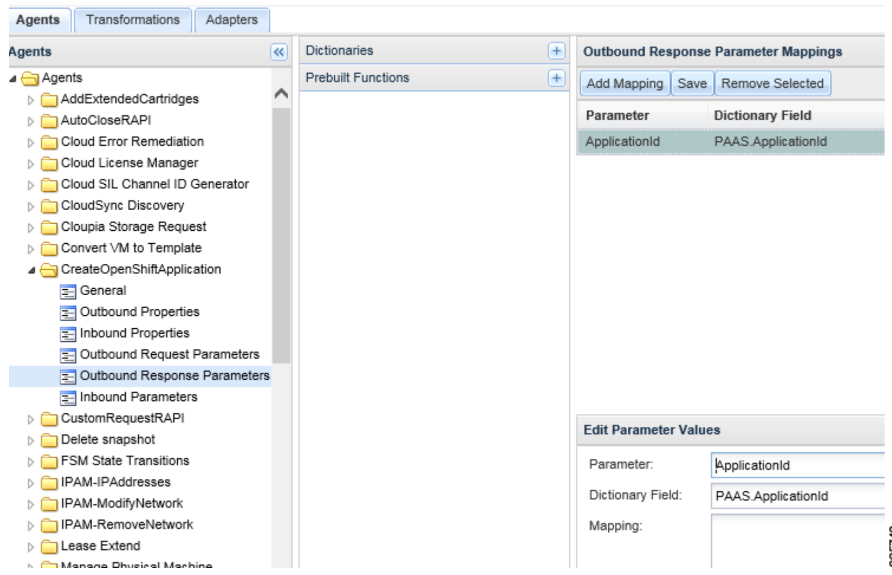
| Service Data Mapping | Parameter |
|---|---|
| #PAAS.PAAS_APP_Name# | PAAS_APP_Name |
| #PAAS.Web_Platform_Type# | Web_Platform_Type |
| #PAAS.Scaling# | Scaling |
| #PAAS.Size_of_Gear# | Size_of_Gear |
| #PAAS.Max_number_gears# | Max_number_gears |
| #PAAS.domainName# | domainName |
| 171.71.29.24 | openShiftHostName |

Parameter: PAAS_APP_Name

Service Data Mapping: #PAAS.PAAS_APP_Name#

**Step 3** Configure Outbound Response Parameters, as shown in Figure 4-21.

*Figure 4-21 Configure Outbound Response Parameters*



Outbound Response Parameter Mappings

| Parameter | Dictionary Field |
|---|---|
| ApplicationId | PAAS.ApplicationId |

Edit Parameter Values

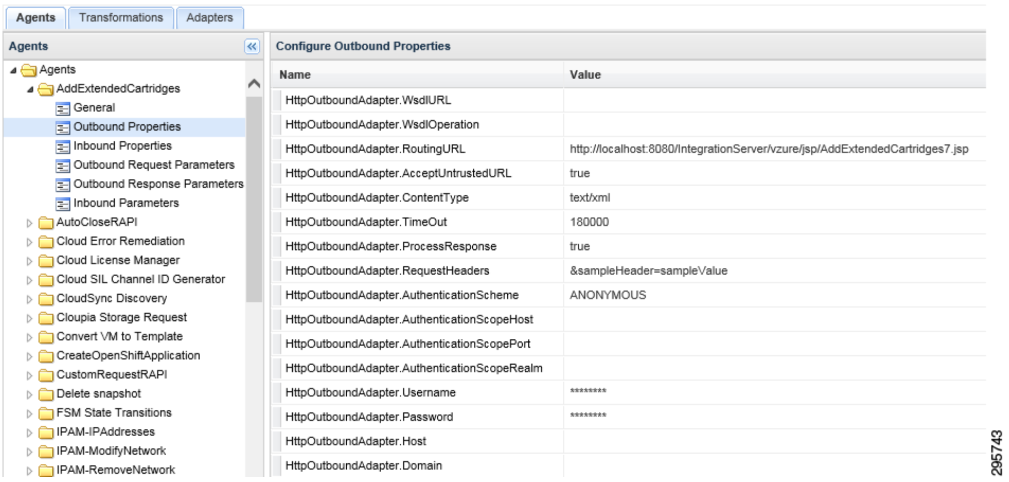
| | |
|---|---|
| Parameter: | ApplicationId |
| Dictionary Field: | PAAS.ApplicationId |
| Mapping: | |

## Add Extended Cartridges Agent

Perform the following procedure to add extended cartridges agent.

**Step 1** Configure ServiceLink agent Outbound Properties, as shown in Figure 4-22.

*Figure 4-22 Configure ServiceLink Agent Outbound Properties*

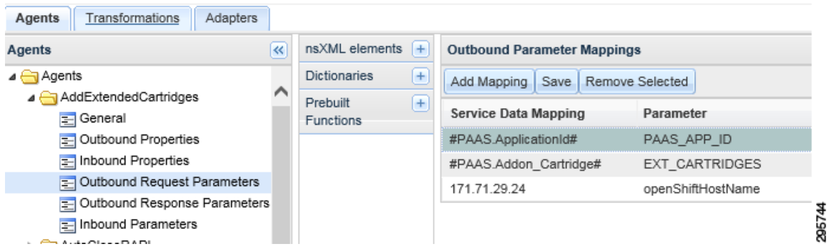


Configure Outbound Properties

| Name | Value |
|---|---|
| HttpOutboundAdapter.WsdlURL | |
| HttpOutboundAdapter.WsdlOperation | |
| HttpOutboundAdapter.RoutingURL | http://localhost:8080/IntegrationServer/vzure/jsp/AddExtendedCartridges7.jsp |
| HttpOutboundAdapter.AcceptUntrustedURL | true |
| HttpOutboundAdapter.ContentType | text/xml |
| HttpOutboundAdapter.TimeOut | 180000 |
| HttpOutboundAdapter.ProcessResponse | true |
| HttpOutboundAdapter.RequestHeaders | &sampleHeader=sampleValue |
| HttpOutboundAdapter.AuthenticationScheme | ANONYMOUS |
| HttpOutboundAdapter.AuthenticationScopeHost | |
| HttpOutboundAdapter.AuthenticationScopePort | |
| HttpOutboundAdapter.AuthenticationScopeRealm | |
| HttpOutboundAdapter.Username | ******** |
| HttpOutboundAdapter.Password | ******** |
| HttpOutboundAdapter.Host | |
| HttpOutboundAdapter.Domain | |

**Step 2** Configure Outbound Request parameters, as shown in Figure 4-23.

*Figure 4-23 Configure Outbound Request Parameters*



Outbound Parameter Mappings

| Service Data Mapping | Parameter |
|---|---|
| #PAAS.ApplicationId# | PAAS_APP_ID |
| #PAAS.Addon_Cartridge# | EXT_CARTRIDGES |
| 171.71.29.24 | openShiftHostName |

## Creating Portal Pages

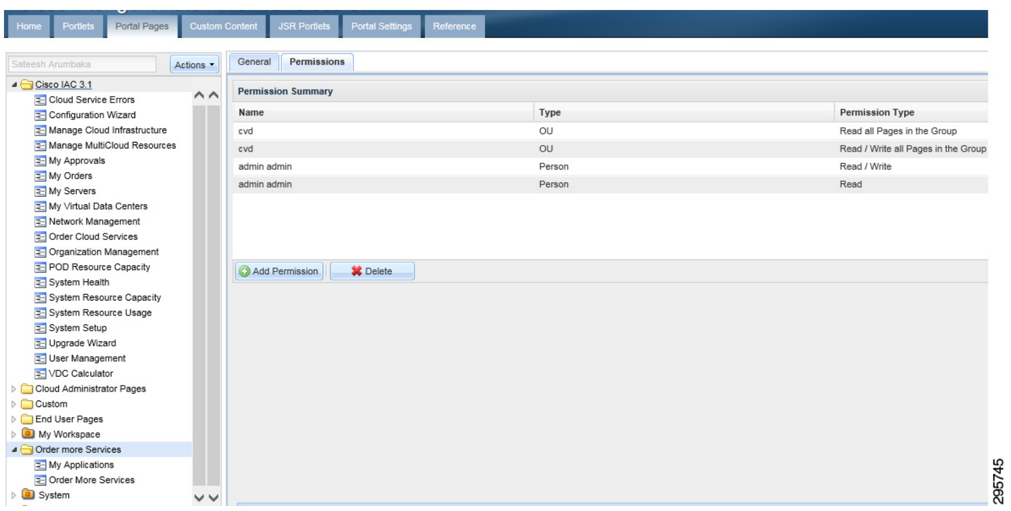
Perform the following procedure to create portal pages.

**Step 1** Log in to Portal Designer module and create a new Portal Page Group.

**Step 2** Set the permissions so that the Organization Unit has read and write permissions to all the pages in the Portal Page group.

**Step 3** Create two Portal pages in this group (Figure 4-24).

*Figure 4-24 Create Portal Pages*

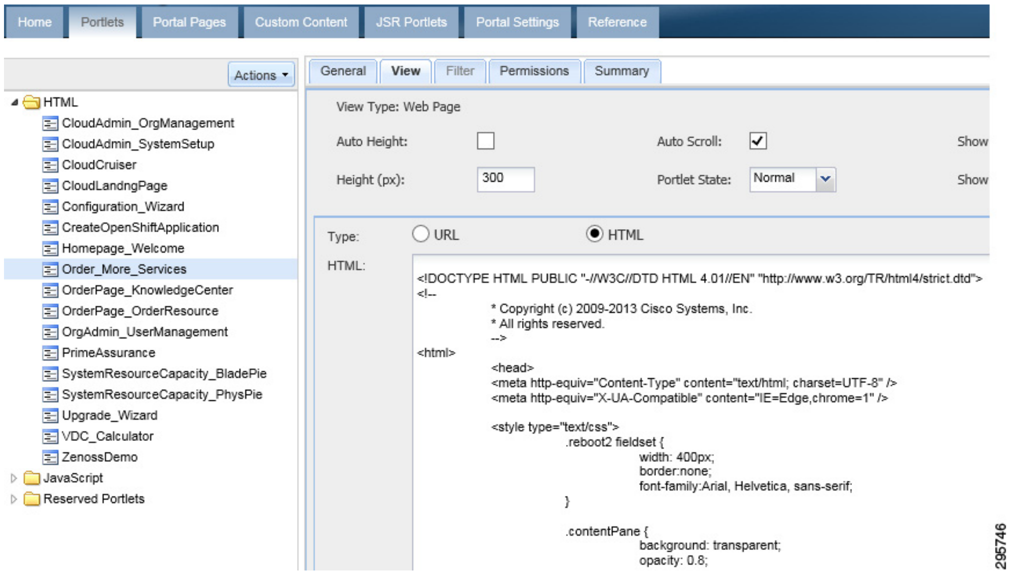


Permission Summary

| Name | Type | Permission Type |
|---|---|---|
| cvd | OU | Read all Pages in the Group |
| cvd | OU | Read / Write all Pages in the Group |
| admin admin | Person | Read / Write |
| admin admin | Person | Read |

**Step 4** Create three portlets, as shown in Figure 4-25, Figure 4-26, and Figure 4-27.

*Figure 4-25 Create Portlets 1*



- HTML
  - CloudAdmin_OrgManagement
  - CloudAdmin_SystemSetup
  - CloudCruiser
  - CloudLandngPage
  - Configuration_Wizard
  - CreateOpenShiftApplication
  - Homepage_Welcome
  - Order_More_Services
  - OrderPage_KnowledgeCenter
  - OrderPage_OrderResource
  - OrgAdmin_UserManagement
  - PrimeAssurance
  - SystemResourceCapacity_BladePie
  - SystemResourceCapacity_PhysPie
  - Upgrade_Wizard
  - VDC_Calculator
  - ZenossDemo
- JavaScript
- Reserved Portlets

View Type: Web Page

Auto Height: ☐ Auto Scroll: ✓ Show

Height (px): 300 Portlet State: Normal Show

Type: URL [● HTML]

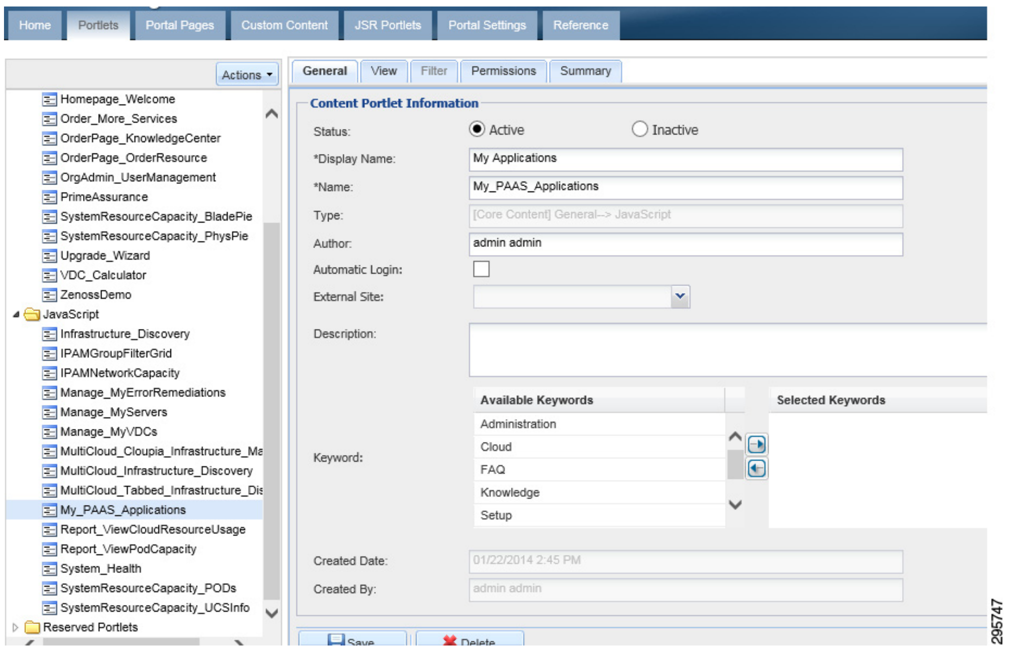
HTML:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<!--
            * Copyright (c) 2009-2013 Cisco Systems, Inc.
            * All rights reserved.
            -->
<html>
            <head>
            <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
            <meta http-equiv="X-UA-Compatible" content="IE=Edge,chrome=1" />

            <style type="text/css">
                        .reboot2 fieldset {
                                    width: 400px;
                                    border:none;
                                    font-family:Arial, Helvetica, sans-serif;
                        }

                        .contentPane {
                                    background: transparent;
                                    opacity: 0.8;
```

*Figure 4-26 Create Portlets 2*

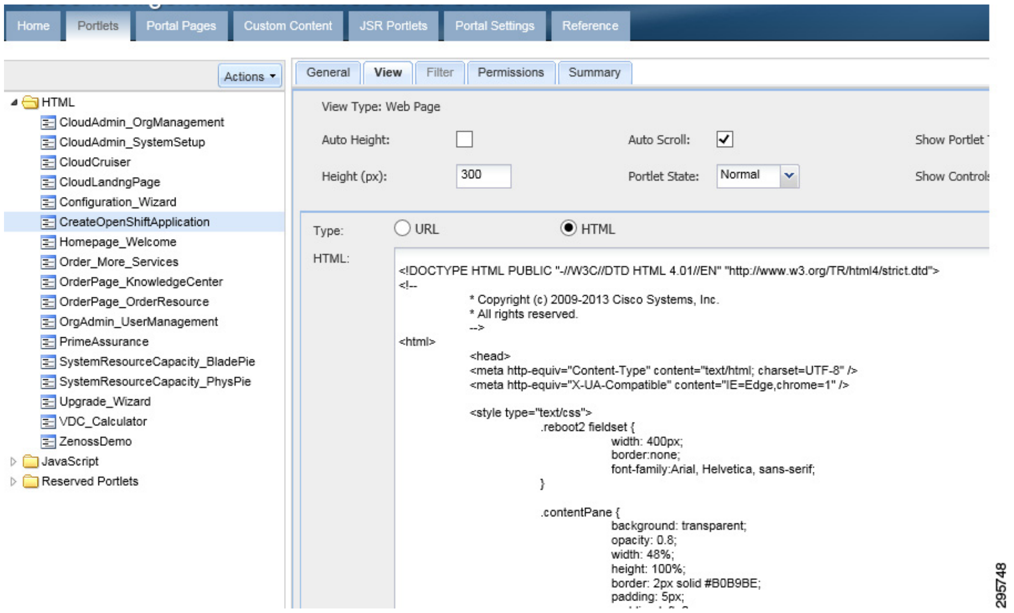


- Homepage_Welcome
- Order_More_Services
- OrderPage_KnowledgeCenter
- OrderPage_OrderResource
- OrgAdmin_UserManagement
- PrimeAssurance
- SystemResourceCapacity_BladePie
- SystemResourceCapacity_PhysPie
- Upgrade_Wizard
- VDC_Calculator
- ZenossDemo
- JavaScript
  - Infrastructure_Discovery
  - IPAMGroupFilterGrid
  - IPAMNetworkCapacity
  - Manage_MyErrorRemediations
  - Manage_MyServers
  - Manage_MyVDCs
  - MultiCloud_Cloupia_Infrastructure_Ma
  - MultiCloud_Infrastructure_Discovery
  - MultiCloud_Tabbed_Infrastructure_Dis
  - My_PAAS_Applications
  - Report_ViewCloudResourceUsage
  - Report_ViewPodCapacity
  - System_Health
  - SystemResourceCapacity_PODs
  - SystemResourceCapacity_UCSInfo
- Reserved Portlets

**Content Portlet Information**

| | |
|---|---|
| Status: | [● Active] Inactive |
| *Display Name: | My Applications |
| *Name: | My_PAAS_Applications |
| Type: | [Core Content] General--> JavaScript |
| Author: | admin admin |
| Automatic Login: | ☐ |
| External Site: | |
| Description: | |
| Keyword: | Available Keywords: Administration, Cloud, FAQ, Knowledge, Setup; Selected Keywords: |
| Created Date: | 01/22/2014 2:45 PM |
| Created By: | admin admin |

*Figure 4-27 Create Portlets 3*



- HTML
  - CloudAdmin_OrgManagement
  - CloudAdmin_SystemSetup
  - CloudCruiser
  - CloudLandngPage
  - Configuration_Wizard
  - CreateOpenShiftApplication
  - Homepage_Welcome
  - Order_More_Services
  - OrderPage_KnowledgeCenter
  - OrderPage_OrderResource
  - OrgAdmin_UserManagement
  - PrimeAssurance
  - SystemResourceCapacity_BladePie
  - SystemResourceCapacity_PhysPie
  - Upgrade_Wizard
  - VDC_Calculator
  - ZenossDemo
- JavaScript
- Reserved Portlets

View Type: Web Page

Auto Height: ☐ Auto Scroll: ✓ Show Portlet

Height (px): 300 Portlet State: Normal Show Controls

Type: URL [✓ HTML]

HTML:

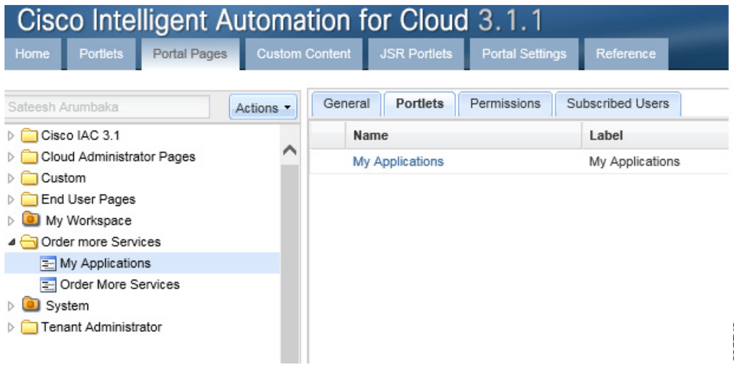
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<!--
          * Copyright (c) 2009-2013 Cisco Systems, Inc.
          * All rights reserved.
          -->
<html>
          <head>
          <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
          <meta http-equiv="X-UA-Compatible" content="IE=Edge,chrome=1" />

          <style type="text/css">
                    .reboot2 fieldset {
                              width: 400px;
                              border:none;
                              font-family:Arial, Helvetica, sans-serif;
                    }

                    .contentPane {
                              background: transparent;
                              opacity: 0.8;
                              width: 48%;
                              height: 100%;
                              border: 2px solid #B0B9BE;
                              padding: 5px;
```

**Step 5** Set the permissions on all three of the Portlets so that the Organization Unit has read and write permissions. Add these portlets to the portal pages, as shown in Figure 4-28 and Figure 4-29.

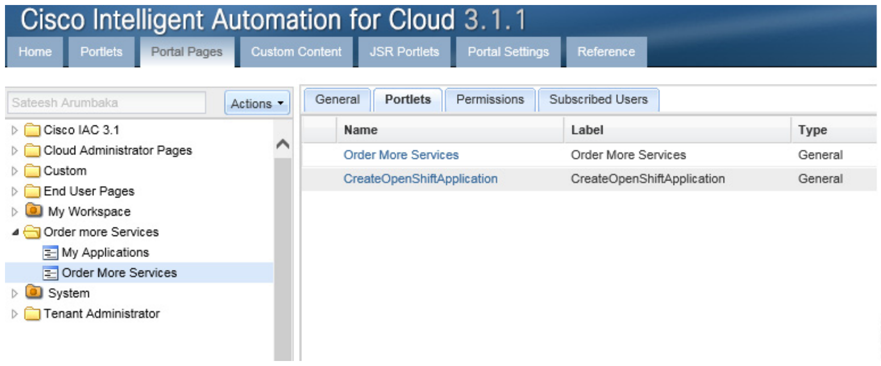
*Figure 4-28 Add Portlets to Portal Pages 1*



- Cisco IAC 3.1
- Cloud Administrator Pages
- Custom
- End User Pages
- My Workspace
- Order more Services
  - My Applications
  - Order More Services
- System
- Tenant Administrator

| Name | Label |
|---|---|
| My Applications | My Applications |

*Figure 4-29 Add Portlets to Portal Pages 2*



| Name | Label | Type |
|---|---|---|
| Order More Services | Order More Services | General |
| CreateOpenShiftApplication | CreateOpenShiftApplication | General |

## Deploy Java Server Page (JSP)

Perform the following procedure to deploy a Java server page.

**Step 1** Log on to the Cisco Prime Catalog server.

**Step 2** Deploy the attached JSPs to the following locations:

- C:\jboss-as-7.1.1.Final\standalone\deployments\RequestCenter.war\vzure\jsp
- C:\jboss-as-7.1.1.Final\standalone\deployments\ServiceLink.war\vzure\jsp

## OpenShift Broker admin Script

Perform the following procedure to execute an OpenShift admin script.

**Step 1** Log on to the Broker host.

Add osadmin to the openshift broker as a user

```
#login to host
ssh -i <sshkey> root@<brokerhost>
#create osadmin user
    oo-admin-ctl-user -c -l osadmin
```

**Step 2** Change osadmin user to allow a large number of domains and gears. All domains are going to be owned by osadmin. Domains for users are assigned to osadmin, and the user is granted edit access to the domains. This allows the user to create/delete/manage applications in their domain.

```
#set max domains and max gears
oo-admin-ctl-user -l osadmin --setmaxdomains 10000
    oo-admin-ctl-user -l osadmin --setmaxgears 10000
```

**Step 3** Add the attaché oo-admin-user-profile profile creation scripts into the broker hosts—this is required for PSC to log into the broker host to create openshift users and assign them domains. This script does the following:

1. Creates the user logins.

2. Creates the domain.
3. Assigns the domain to osadmin.
4. Assigns the user edit access to the domain.
5. Limits the max number of gears for the user to 0.
6. Limits the max number of domains for the user to 0.

```
# copy the script to openshift broker host
scp oo-admin-cvd-user-profile root@<brokerhost>
# edit the oo-admin-cvd-user-profile file to set the following parameters
# OSADMIN="osadmin" # set this to the osadmin user as shown
# OSPWD="<password>" # set this to the osadmin directory/ldap password
    # BROKER="<broker host ip address>" # set this to the public ip of the broker
```

## Testing

The methodology used for testing and validating the functionality was based on test cases per use case for the use cases outlined in the System Overview section.

## Use Case 1: Integrated Provisioning for IaaS and PaaS

The following procedure is performed for configuring integrated provisioning for IaaS and PaaS.

**Step 1** Add users to the Active Directory installation in the specified OU of the implementation section.

**Step 2** Log in to PSC.

**Step 3** From the Portal selection dropdown, choose My Workspace.

**Step 4** From the workspace, click the Order more services tab.

**Step 5** On the Platform As A Service Portlet, click Create a new PAAS Application.

### Results

The following results are observed.

- Able to log in with created credentials.
- Able to view Unified portal page with IaaS Cloud Services and PaaS Services.
- Have permissions to order PaaS Application, which shows up, and order form..

## Use Case 2: Application Stack Creation

The following procedure is performed for configuring application stack creation.

**Step 1** Verify the order form shows up after step 5 in Use Case 1: Integrated Provisioning for IaaS and PaaS, page 4-32.

**Step 2** Enter the application name that you want to create, e.g. rubyapp.

**Step 3** Choose platform type (eg.g python-2.7).

**Step 4** Choose Gear Profile Small.

**Step 5** Click False on the Scaling type.

**Step 6** Click Submit Order.

**Step 7** Close the order confirmation form.

## Results

The following results are observed.

- Able to choose different application stacks.
- Able to choose different resource profiles.
- Able to submit order for stack creation.
- Able to confirm order for stack creation.

## Use Case 3: Single Pane Management of Application Stack

The following procedure is performed for configuring single pane management of application stack.

**Step 1** On the workspace page, click the My Applications tab.

**Step 2** On the list of applications, click the application name just created in Use Case 2: Application Stack Creation, page 4-32.

**Step 3** In the Take Action section:

**a.** Click View App to take you to the application URL.

**b.** Click Stop application.

**c.** Click Start application.

**Step 4** For scaled application type, click View Status.

**Step 5** Use git clone to clone the application repository shown in the Application Details section.

**Step 6** In the Take Action section, click Delete application.

**Step 7** Repeat Use Case 2: Application Stack Creation, page 4-32 steps, click Scaled Application, and submit order.

**Step 8** Repeat Use Case 2: Application Stack Creation, page 4-32 steps and create a few different applications with different application stacks.

**Step 9** On the list of applications, verify Steps 3 through 7 for each application.

## Results

The following results are observed.

- Able to view list of applications created.
- Able to act on different actions like start, stop for each application stack.
- Able to delete application.
- Able to browse to application URL.

## Use Case 4: Integrated Provisioning for IaaS and PaaS

The following procedure is performed for configuring integrated provisioning for IaaS and PaaS.

**Step 1** Refer to Use Case 1: Integrated Provisioning for IaaS and PaaS, page 4-32 and Use Case 2: Application Stack Creation, page 4-32, where a direct order for both IaaS and PaaS resources can be placed from a single pane. Validate this use case.

**Note** Ordering of IaaS resources comes with the PSC solution and is not validated separately.

### Results

See results for Use Cases 1, 2, and 3.

## Use Case 5: Network-based Segmentation of PaaS Districts for Security

The following procedure is performed for configuring network-based segmentation of PaaS districts for security.

**Step 1** Repeat steps 1, 2 and 3 in Use Case 2: Application Stack Creation, page 4-32.

**Step 2** Choose Small-Secure* Gear profile.

**Step 3** Submit order.

**Step 4** Repeat steps 1 through 8 in Use Case 3: Single Pane Management of Application Stack, page 4-33 for the secure network application.

**Step 5** Log in to OpenShift node in secure network as admin/root via ssh and verify that application was created.

### Results

The following results are observed.

- Able to choose a secure network profile for an application.
- Able to verify that the application got created in secure network.

## Summary

The following recommended implementation was conducted:

- Use of OpenStack for IaaS.
- Deployment of OpenShift Enterprise into OpenStack using HEAT templates.
- Deployment of two separate network segments in OpenStack to house secure and standard OpenShift nodes.
- Creation of PaaS ordering Services that talk to OpenShift Broker in PSC.
- Creation of PaaS Application management portal pages in Prime Services Catalog.
- Binding OpenShift and OpenStack authentication to LDAP (AD) directory.

APPENDIX A

# Terms

Table A-1 lists acronyms and abbreviations used in this document.

*Table A-1 List of Terms*

| Term | Abbreviation |
|---|---|
| AD | Microsoft Active Directory |
| CIAC | Cisco Intelligent Automation for Cloud |
| CSDL | Cisco Secure Development Lifecycle |
| IaaS | Infrastructure-as-a-Service |
| MTOSI | Multi-Technology Operations Systems Interface |
| NBI | Northbound Interfaces |
| PaaS | Platform-as-a-Service |
| PSC | Prime Service Catalog |
| SaaS | Software-as-a-Service |
| SBI | Southbound Mediation Interface |
| SDK | Software Development Kit |
| TMF | TeleManagement Forum |
| UCS | Cisco Unified Computing System |
| VM | virtual machine |
| XaaS | Anything-as-a-Service |